aws

개발자 가이드

Amazon Cognito

Amazon Cognito: 개발자 가이드

Table of Contents

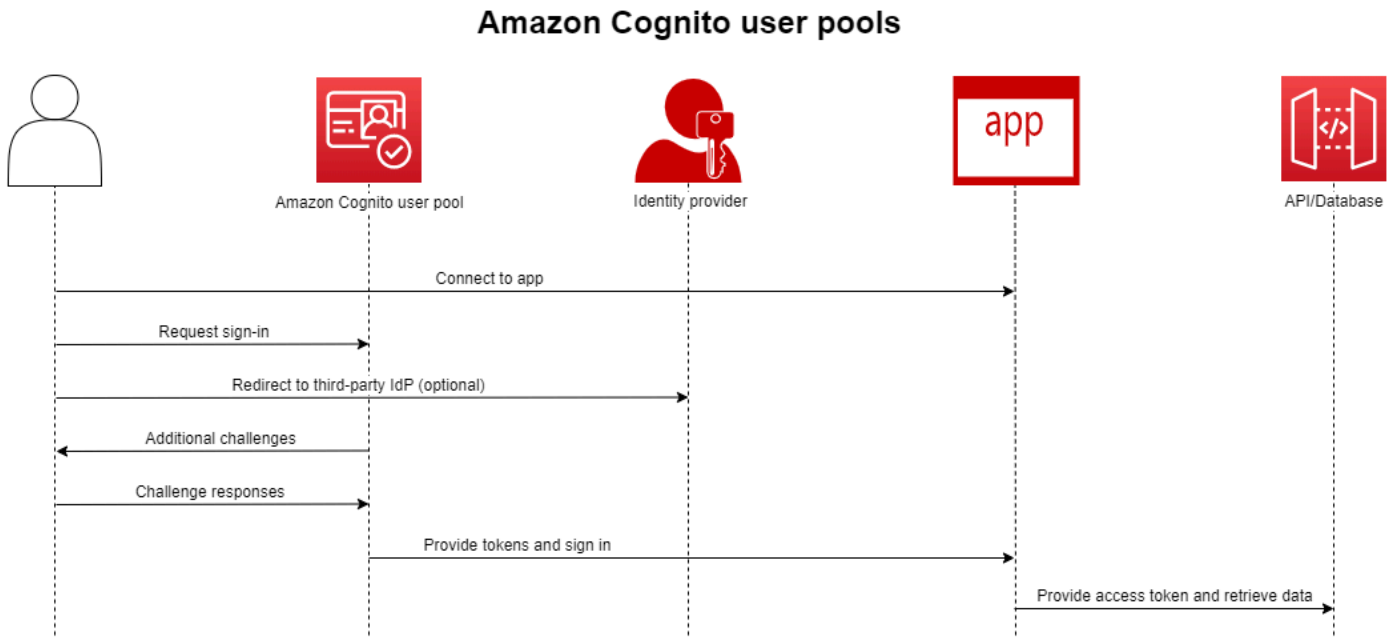
Amazon Cognito란 무엇입니까?

Amazon Cognito는 웹 및 모바일 앱을 위한 자격 증명 플랫폼입니다. Amazon Cognito는 OAuth 2.0 액세스 토큰 및 AWS 보안 인증을 위한 사용자 디렉터리, 인증 서버, 인증 서비스입니다. Amazon Cognito를 사용하면 기본 제공 사용자 디렉터리, 엔터프라이즈 디렉터리, Google 및 Facebook 같은 소비자 ID 제공업체의 사용자를 인증하고 권한을 부여할 수 있습니다.

주제

다음 두 가지 구성 요소가 Amazon Cognito를 구성합니다. 이들은 사용자의 액세스 요구 사항에 따라 독립적으로 또는 함께 작동합니다.

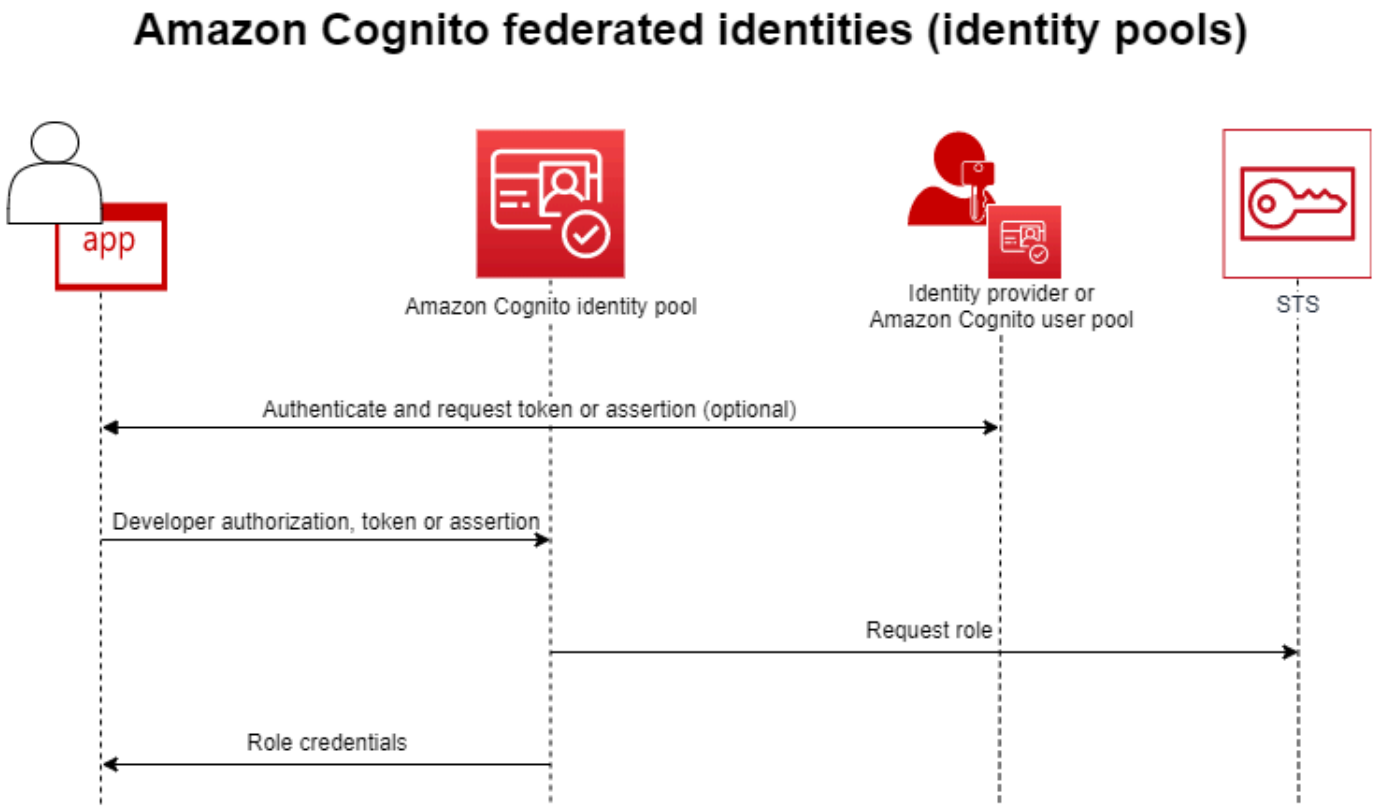
사용자 풀



앱 또는 API에 사용자를 인증하고 권한을 부여하려는 경우 사용자 풀을 생성하세요. 사용자 풀은 셀프 서비스 및 관리자 기반 사용자 생성, 관리, 인증을 모두 갖춘 사용자 디렉터리입니다. 사용자 풀은 독립 디렉터리 및 OIDC ID 제공업체(idP)일 수 있고, 직원 및 고객 자격 증명 서드 파티 공급자에 대한 중간 서비스 공급자(SP)일 수 있습니다. 사용자 풀이 있는 SAML 2.0 및 OIDC에서 조직의 직원 ID에 대한 싱글 사인온 (SSO) 을 앱에 제공할 수 있습니다. IdPs 퍼블릭 OAuth 2.0 ID 스토어인 Amazon, Google, Apple, Facebook의 조직 고객 ID를 위한 SSO를 앱에 제공할 수도 있습니다. 고객 ID 및 액세스 관리(CIAM)에 대한 자세한 내용은 CIAM이란 무엇인가요? 섹션을 참조하십시오.

사용자 풀은 자격 증명 풀과의 통합이 필요하지 않습니다. 사용자 풀에서 인증된 JSON 웹 토큰(JWT)을 앱, 웹 서버 또는 API에 직접 발급할 수 있습니다.

자격 증명 풀



인증된 사용자 또는 익명 사용자가 리소스에 액세스할 수 있도록 승인하려는 경우 Amazon Cognito 자격 증명 풀을 설정하십시오. AWS 자격 증명 풀은 앱이 사용자에게 리소스를 제공할 수 있도록 AWS 자격 증명을 발급합니다. 사용자 풀 또는 SAML 2.0 서비스 같은 신뢰할 수 있는 ID 제공업체를 사용하여 사용자를 인증할 수 있습니다. 필요한 경우 게스트 사용자의 보안 인증도 발급할 수 있습니다. 자격 증명 풀은 역할 기반 액세스 제어와 속성 기반 액세스 제어를 모두 사용하여 사용자의 리소스 액세스 권한을 관리합니다. AWS

자격 증명 풀은 사용자 풀과의 통합이 필요하지 않습니다. 자격 증명 풀은 직원 ID 제공업체와 소비자 ID 제공업체 모두로부터 인증된 클레임을 직접 수락할 수 있습니다.

Amazon Cognito 사용자 풀과 자격 증명 풀을 함께 사용

이 주제를 시작하는 다이어그램에서는 Amazon Cognito를 사용하여 사용자를 인증한 다음 사용자에게 AWS 서비스 액세스 권한을 부여합니다.

1. 앱 사용자는 사용자 풀을 통해 로그인하고 OAuth 2.0 토큰을 받습니다.

2. 앱은 사용자 풀 토큰을 자격 증명 풀과 교환하여 AWS API 및 () 와 함께 사용할 수 있는 임시 AWS 자격 증명으로 교환합니다. AWS Command Line Interface AWS CLI
3. 앱은 사용자에게 자격 증명 세션을 할당하고 Amazon S3 및 Amazon DynamoDB AWS 서비스 등에 대한 승인된 액세스를 제공합니다.

자격 증명 풀과 사용자 풀을 사용하는 더 많은 예제는 일반적인 Amazon Cognito 시나리오를 참조하세요.

Amazon Cognito에서 공동 책임 모델의 클라우드 자체 보안 의무는 SOC 1-3, PCI DSS, ISO 27001을 준수하며, HIPA-BAA 적격입니다. Amazon Cognito에서 클라우드 내 보안이 SOC1-3, ISO 27001, HIPAA-BAA를 준수하도록 설계할 수 있습니다. 단, PCI DSS는 준수하지 않습니다. 자세한 내용은 범위 내AWS 서비스를 참조하세요. 리전 데이터 고려 사항도 참조하세요.

Amazon Cognito의 기능

사용자 풀

Amazon Cognito 사용자 풀은 사용자 디렉터리입니다. 사용자 풀을 사용하면 사용자는 Amazon Cognito를 통해, 또는 서드 파티 IdP를 통해 페더레이션하여 웹 또는 모바일 앱에 로그인할 수 있습니다. 사용자 풀에는 페더레이션 사용자와 로컬 사용자의 사용자 프로필이 있습니다.

로컬 사용자는 가입했거나 사용자 풀에서 직접 생성한 사용자입니다. AWS Management Console, AWS SDK 또는 () 에서 이러한 사용자 프로필을 관리하고 사용자 지정할 수 있습니다. AWS Command Line Interface AWS CLI

Amazon Cognito 사용자 풀은 타사의 IdPs 토큰과 어설션을 수락하고 사용자 속성을 JWT로 수집하여 앱에 발급합니다. Amazon Cognito가 클레임을 중앙 토큰 형식에 매핑하여 상호 작용을 처리하는 동안 하나의 JWT 세트로 IdPs 앱을 표준화할 수 있습니다.

Amazon Cognito 사용자 풀은 독립 실행형 IdP일 수 있습니다. Amazon Cognito는 OpenID Connect(OIDC) 표준을 기반으로 인증 및 권한 부여를 위한 JWT를 생성합니다. 로컬 사용자를 로그인하면 해당 사용자는 사용자 풀을 신뢰할 수 있습니다. 로컬 사용자를 인증할 때 다음 기능에 액세스할 수 있습니다.

- Amazon Cognito 사용자 풀 API를 호출하여 사용자를 인증, 권한 부여, 관리하는 자체 웹 프런트엔드를 구현합니다.
- 사용자 다중 인증(MFA)을 설정합니다. Amazon Cognito는 시간 기반 일회용 암호(TOTP) 및 SMS 메시지 MFA를 지원합니다.

- 악의적으로 제어되는 사용자 계정의 액세스로부터 보호합니다.
- 자체 사용자 지정 다단계 인증 흐름을 생성합니다.
- 다른 디렉터리에서 사용자를 찾아 Amazon Cognito로 마이그레이션합니다.

또한 Amazon Cognito 사용자 풀은 사용자의 서비스 공급자 (SP) 역할과 앱의 IdP라는 이중 역할을 수행할 수 IdPs 있습니다. Amazon Cognito 사용자 풀은 페이스북 및 구글과 IdPs 같은 소비자 또는 Okta 및 Active Directory 페더레이션 서비스 (ADFS) 와 IdPs 같은 인력과 연결할 수 있습니다.

Amazon Cognito 사용자 풀이 발급하는 OAuth 2.0 및 OpenID Connect(OIDC) 토큰을 사용하면 다음과 같은 작업을 수행할 수 있습니다.

- 사용자를 인증하고 사용자 프로필을 설정하는 데 필요한 정보를 제공하는 ID 토큰을 앱에서 수락합니다.
- 사용자의 API 호출을 승인하는 OIDC 범위를 사용하여 API에서 액세스 토큰을 수락합니다.
- Amazon Cognito AWS 자격 증명 풀에서 자격 증명을 검색합니다.

Amazon Cognito 사용자 풀의 기능

기능	설명
OIDC IdP	ID 토큰을 발급하여 사용자를 인증하십시오.
권한 부여 서버	액세스 토큰을 발급하여 API에 대한 사용자 액세스를 승인합니다.
SAML 2.0 SP	SAML 어설션을 ID 및 액세스 토큰으로 변환
OIDC SP	OIDC 토큰을 ID 및 액세스 토큰으로 변환
OAuth 2.0 SP	애플, 페이스북, 아마존, 구글의 ID 토큰을 자신의 ID 및 액세스 토큰으로 변환하세요.
인증 프론트엔드 서비스	호스팅된 UI를 사용하여 사용자를 등록, 관리 및 인증합니다.
자체 UI를 위한 API 지원	지원되는 AWS SDK의 API 요청을 통해 사용자 생성, 관리 및 인증[1]

MFA	SMS 메시지, TOTP 또는 사용자 디바이스를 추가 인증 요소로 사용[1]
보안 모니터링 및 대응	악의적인 활동 및 안전하지 않은 비밀번호로부터 보호[1]
인증 흐름 사용자 지정	자체 인증 메커니즘을 구축하거나 기존 플로우에 사용자 지정 단계를 추가[1]
그룹	자격 증명 풀에 토큰을 전달할 때 논리적인 사용자 그룹화 및 IAM 역할 클레임 계층 구조 생성
ID 토큰을 사용자 지정하세요.	신규 클레임, 수정된 클레임, 숨겨진 클레임으로 ID 토큰을 사용자 지정하세요.
사용자 특성을 맞춤설정하세요.	사용자 속성에 값을 할당하고 고유한 사용자 지정 특성을 추가합니다.

[1] 기능은 로컬 사용자만 사용할 수 있습니다.

사용자 풀에 대한 자세한 내용은 사용자 풀 시작하기 및 Amazon Cognito 사용자 풀 API 참조를 참조하세요.

자격 증명 풀

자격 증명 풀은 사용자나 게스트에게 할당하고 임시 자격 증명을 받을 수 있는 권한을 부여하는 고유 식별자 또는 ID의 모음입니다. AWS SAML 2.0, OpenID Connect(OIDC) 또는 OAuth 2.0 소셜 ID 제공업체(idP)의 신뢰할 수 있는 클레임 형태로 자격 증명 풀에 인증 증명을 제시하면 사용자가 자격 증명 풀의 자격 증명에 연결됩니다. 자격 증명 풀이 해당 자격 증명을 위해 생성하는 토큰은 () 에서 AWS Security Token Service 임시 세션 자격 증명을 검색할 수 있습니다.AWS STS

인증된 ID를 보완하기 위해 IdP 인증 AWS 없이 액세스를 승인하도록 ID 풀을 구성할 수도 있습니다. 자체 사용자 지정 인증 증명을 제공하거나 인증을 제공하지 않을 수 있습니다. 임시 AWS 자격 증명을 요청하는 모든 앱 사용자에게 인증되지 않은 ID를 사용하여 임시 자격 증명을 부여할 수 있습니다. 또한 자격 증명 풀은 개발자 인증 자격 증명과 함께 자체 사용자 지정 스키마를 기반으로 클레임을 수락하고 보안 인증을 발급합니다.

Amazon Cognito 자격 증명 풀을 사용하면 AWS 계정에서 두 가지 방법으로 IAM 정책과 통합할 수 있습니다. 이 두 기능은 함께 사용하거나 개별적으로 사용할 수 있습니다.

역할 기반 액세스 제어

사용자가 자격 증명 풀에 클레임을 전달하면 Amazon Cognito는 자격 증명 풀이 요청하는 IAM 역할을 선택합니다. 역할의 권한을 필요에 맞게 사용자 지정하려면 각 역할에 IAM 정책을 적용합니다. 예를 들어 마케팅 부서에 소속되어 있음을 입증하는 사용자는 마케팅 부서 액세스 요구 사항에 맞게 조정된 정책이 포함된 역할의 보안 인증을 받게 됩니다. Amazon Cognito는 기본 역할, 사용자의 클레임을 쿼리하는 규칙에 기반한 역할 또는 사용자 풀의 사용자 그룹 멤버십에 기반한 역할을 요청할 수 있습니다. IAM이 임시 세션 생성을 위해 자격 증명 풀만 신뢰하도록 역할 신뢰 정책을 구성할 수도 있습니다.

액세스 제어를 위한 속성

자격 증명 풀은 사용자의 클레임에서 속성을 읽고 이를 사용자 임시 세션의 보안 주체 태그에 매핑합니다. 그런 다음 IAM 리소스 기반 정책을 구성하여 자격 증명 풀에서 세션 태그를 전달하는 IAM 보안 주체를 기반으로 리소스에 대한 액세스를 허용하거나 거부할 수 있습니다. 예를 들어 사용자가 마케팅 부서에 근무하고 있음을 입증하는 경우 세션에 태그를 지정하십시오. AWS STS `Department: marketing` Amazon S3 버킷은 `Department` 태그 값이 필요한 aws: PrincipalTag 조건에 따라 읽기 작업을 허용합니다. `marketing`

Amazon Cognito 자격 증명 풀의 기능

기능	설명
아마존 코그니토 사용자 풀 SP	사용자 풀의 ID 토큰을 다음의 웹 자격 증명 자격 증명으로 교환하십시오. AWS STS
SAML 2.0 SP	SAML 어설션을 다음과 같은 웹 ID 자격 증명으로 교환하십시오. AWS STS
OIDC SP	OIDC 토큰을 웹 ID 자격 증명으로 교환하십시오. AWS STS
OAuth 2.0 SP	아마존, 페이스북, 구글, 애플, 트위터의 OAuth 토큰을 웹 ID 자격 증명으로 교환하세요. AWS STS
커스텀 SP	AWS 자격 증명을 사용하면 모든 형식의 클레임을 웹 ID 자격 증명으로 교환할 수 있습니다. AWS STS

인증되지 않은 액세스	인증 없이 액세스가 제한된 웹 ID 자격 증명을 발급하십시오. AWS STS
역할 기반 액세스 제어	클레임을 기반으로 인증된 사용자의 IAM 역할을 선택하고, 자격 증명 풀의 컨텍스트에서만 역할을 위임하도록 구성하십시오.
속성 기반 액세스 제어	클레임을 AWS STS 임시 세션의 주 태그로 전환하고, IAM 정책을 사용하여 보안 주체 태그를 기준으로 리소스 액세스를 필터링합니다.

자격 증명 풀에 대한 자세한 내용은 Amazon Cognito 자격 증명 풀 시작하기 및 Amazon Cognito 자격 증명 풀 API 참조를 참조하세요.

Amazon Cognito 사용자 풀과 자격 증명 풀 비교

기능	설명	사용자 풀	자격 증명 풀
OIDC IdP	OIDC ID 토큰을 발급하여 앱 사용자를 인증하세요.	✓	
API 인증 서버	액세스 토큰을 발급하여 OAuth 2.0 인증 범위를 허용하는 API, 데이터베이스 및 기타 리소스에 대한 사용자 액세스를 승인합니다.	✓	
IAM 웹 ID 인증 서버	임시 AWS 자격 증명으로 AWS STS 교환할 수 있는 토큰 생성		✓
SAML 2.0 SP 및 OIDC IdP	SAML 2.0 IdP의 클레임을 기반으로 맞춤형	✓	

	OIDC 토큰을 발행하십시오.		
OIDC SP 및 OIDC IdP	OIDC IdP의 클레임을 기반으로 맞춤형 OIDC 토큰 발행	✓	
OAuth 2.0 SP 및 IDC IdP	애플, 구글과 같은 OAuth 2.0 소셜 제공업체의 범위를 기반으로 맞춤형 OIDC 토큰을 발행하세요.	✓	
SAML 2.0 SP 및 자격 증명 브로커	SAML 2.0 IdP의 클레임을 기반으로 임시 AWS 자격 증명 발급		✓
OIDC SP 및 자격 증명 브로커	OIDC IdP의 클레임을 기반으로 임시 AWS 자격 증명 발급		✓
OAuth 2.0 SP 및 자격 증명 브로커	Apple 및 Google과 같은 OAuth 2.0 소셜 제공업체의 범위를 기반으로 임시 AWS 자격 증명을 발급합니다.		✓
Amazon Cognito 사용자 풀 SP 및 자격 증명 브로커	Amazon Cognito AWS 사용자 풀의 OIDC 클레임을 기반으로 임시 자격 증명을 발급합니다.		✓
맞춤형 SP 및 자격 증명 브로커	개발자 IAM AWS 인증을 기반으로 임시 자격 증명을 발급합니다.		✓

인증 프론트엔드 서비스	호스팅된 UI를 사용하여 사용자를 등록, 관리 및 인증합니다.	✓
자체 인증 UI를 위한 API 지원	지원되는 AWS SDK의 API 요청을 통해 사용자 생성, 관리 및 인증[1]	✓
MFA	SMS 메시지, TOTP 또는 사용자 디바이스를 추가 인증 요소로 사용[1]	✓
보안 모니터링 및 대응	악의적인 활동 및 안전하지 않은 비밀번호로부터 보호[1]	✓
인증 흐름 사용자 지정	자체 인증 메커니즘을 구축하거나 기존 플로우에 사용자 지정 단계를 추가[1]	✓
그룹	자격 증명 풀에 토큰을 전달할 때 논리적인 사용자 그룹화 및 IAM 역할 클레임 계층 구조 생성	✓
ID 토큰을 사용자 지정하세요.	신규 클레임, 수정된 클레임, 숨겨진 클레임으로 ID 토큰을 사용자 지정하세요.	✓
AWS WAF 웹 ACL	다음을 사용하여 인증 환경에 대한 요청을 모니터링하고 제어합니다. AWS WAF	✓

사용자 속성 사용자 지정	사용자 속성에 값을 할당하고 고유한 사용자 지정 특성을 추가합니다.	✓	
인증되지 않은 액세스	인증 없이 액세스가 제한된 웹 ID 자격 증명을 발급하십시오. AWS STS		✓
역할 기반 액세스 제어	클레임을 기반으로 인증된 사용자의 IAM 역할을 선택하고, 자격 증명 풀의 컨텍스트에서만 역할을 위임하도록 구성하십시오.		✓
속성 기반 액세스 제어	사용자 클레임을 AWS STS 임시 세션의 보안 주체 태그로 변환하고, IAM 정책을 사용하여 보안 주체 태그를 기준으로 리소스 액세스를 필터링합니다.		✓

[1] 기능은 로컬 사용자만 사용할 수 있습니다.

Amazon Cognito 시작하기

예를 들어 사용자 풀 애플리케이션은 을 참조하십시오사용자 풀 시작하기.

자격 증명 풀에 대한 소개는 를 참조하십시오Amazon Cognito 자격 증명 풀 시작하기.

사용자 풀 및 자격 증명 풀의 설정 안내 경험에 대한 링크는 를 참조하십시오아마존 Cognito의 설정 옵션 안내.

동영상, 기사, 설명서 및 기타 샘플 애플리케이션에 대해서는 Amazon Cognito 개발자 리소스를 참조하십시오.

Amazon Cognito를 사용하려면 AWS 계정이 있어야 합니다. 자세한 정보는 시작하기 AWS을 참조하세요.

리전별 가용성

Amazon Cognito는 전 세계 여러 AWS 지역에서 사용할 수 있습니다. 각 리전에서 Amazon Cognito는 다수의 가용 영역으로 배포됩니다. 이러한 가용 영역은 물리적으로 서로 분리되어 있지만, 지연 시간이 짧고 처리량과 중복성이 우수한 프라이빗 네트워크 연결로 통합됩니다. 이러한 가용 영역을 사용하면 AWS Amazon Cognito를 비롯한 서비스에 매우 높은 수준의 가용성과 중복성을 제공하는 동시에 지연 시간을 최소화할 수 있습니다.

현재 Amazon Cognito가 제공되는 모든 리전의 목록은 Amazon Web Services 일반 참조에서 AWS 리전 및 엔드포인트를 참조하세요. 각 리전에서 사용할 수 있는 가용 영역 수에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

Amazon Cognito 요금

Amazon Cognito 요금에 대한 자세한 내용은 Amazon Cognito 요금을 참조하세요.

Amazon Cognito 사용자 풀 및 자격 증명 풀에서 인증이 작동하는 방식

고객이 Amazon Cognito 사용자 풀에 로그인하면 애플리케이션이 JSON 웹 토큰 (JWT) 을 받습니다.

고객이 사용자 풀 토큰 또는 다른 공급자를 사용하여 자격 증명 풀에 로그인하면 애플리케이션은 임시 자격 증명을 받습니다. AWS

사용자 풀 로그인을 사용하면 AWS SDK를 사용하여 인증 및 권한 부여를 완전히 구현할 수 있습니다. 사용자 인터페이스 (UI) 구성 요소를 직접 구축하지 않으려면 사전 구축된 웹 UI (호스팅된 UI) 또는 타사 ID 공급자 (IdP) 의 로그인 페이지를 호출할 수 있습니다.

이 주제에서는 애플리케이션이 Amazon Cognito와 상호 작용하여 ID 토큰으로 인증하고, 액세스 토큰으로 권한을 부여하고, 자격 증명 풀 자격 증명으로 액세스할 수 있는 몇 가지 방법에 대한 AWS 서비스 개요입니다.

주제

- SDK를 통한 사용자 풀 API 인증 및 권한 부여 AWS

- 호스팅된 UI를 사용한 사용자 풀 인증
- 타사 ID 공급자를 통한 사용자 풀 인증
- 자격 증명 풀 인증

SDK를 통한 사용자 풀 API 인증 및 권한 부여 AWS

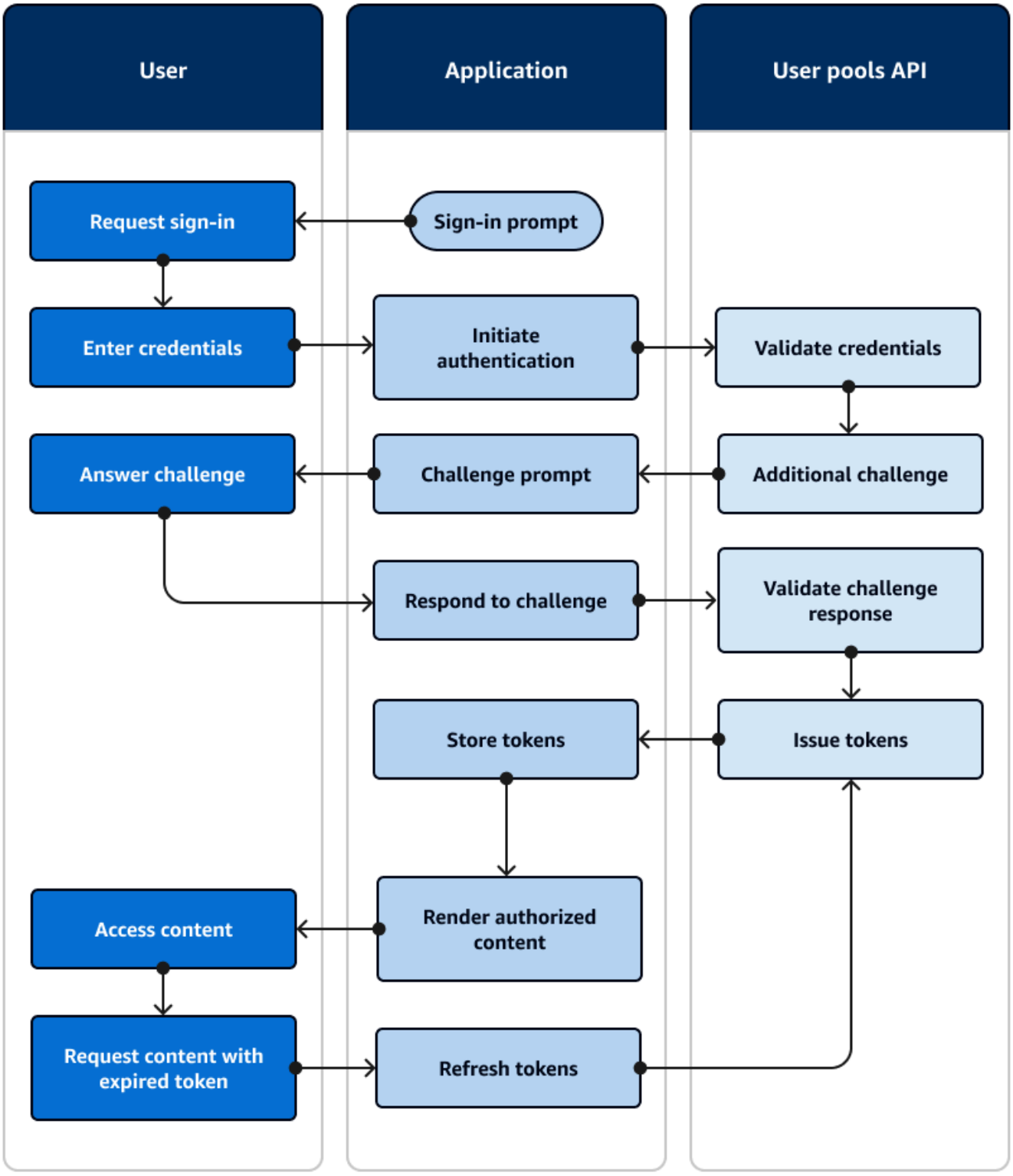
AWS 다양한 개발자 프레임워크에서 Amazon Cognito 사용자 풀 또는 Amazon Cognito 자격 증명 공급자용 구성 요소를 개발했습니다. 이러한 SDK에 내장된 메서드는 Amazon Cognito 사용자 풀 API를 호출합니다. 동일한 사용자 풀 API 네임스페이스는 사용자 풀 구성 및 사용자 인증을 위한 작업을 수행합니다. 자세한 개요는 를 참조하십시오. Amazon Cognito 사용자 풀 API 및 사용자 풀 엔드포인트 사용

API 인증은 애플리케이션에 기존 UI 구성요소가 있고 주로 사용자 풀을 사용자 디렉토리로 사용하는 모델에 적합합니다. 이 디자인은 Amazon Cognito를 대규모 애플리케이션 내의 구성 요소로 추가합니다. 복잡한 챌린지와 대응의 연쇄를 처리하려면 프로그래밍 로직이 필요합니다.

이 애플리케이션은 완전한 OpenID Connect (OIDC) 신뢰 당사자 구현을 구현할 필요가 없습니다. 대신 JWT를 디코딩하고 사용할 수 있습니다. 로컬 사용자를 위한 전체 사용자 풀 기능에 액세스하려면 개발 환경에서 Amazon Cognito SDK를 사용하여 인증을 구축하십시오.

사용자 지정 OAuth 범위를 사용하는 API 인증은 외부 API 인증에 덜 중점을 둡니다. API 인증에서 액세스 토큰에 사용자 지정 범위를 추가하려면 런타임 시 a를 사용하여 토큰을 수정하십시오. 사전 토큰 생성 Lambda 트리거

다음 다이어그램은 API 인증을 위한 일반적인 로그인 세션을 보여줍니다.


User
Application
User pools API
Request sign-in
Sign-in prompt
Enter credentials
Initiate authentication
Validate credentials
Answer challenge
Challenge prompt
Additional challenge
Respond to challenge
Validate challenge response
Store tokens
Issue tokens
Access content
Render authorized content
Request content with expired token
Refresh tokens

API 인증 흐름

1. 사용자가 애플리케이션에 액세스합니다.
2. 사용자는 “로그인” 링크를 선택합니다.
3. 사용자는 사용자 이름과 비밀번호를 입력합니다.
4. 애플리케이션은 InitiateAuthAPI 요청을 보내는 메서드를 호출합니다. 요청은 사용자의 자격 증명을 사용자 풀에 전달합니다.
5. 사용자 풀은 사용자의 자격 증명을 검증하고 사용자가 멀티 팩터 인증 (MFA) 을 활성화했는지 확인합니다.
6. 사용자 풀은 MFA 코드를 요청하는 챌린지로 응답합니다.
7. 애플리케이션은 사용자로부터 MFA 코드를 수집하는 프롬프트를 생성합니다.
8. 애플리케이션은 RespondToAuthChallengeAPI 요청을 보내는 메서드를 호출합니다. 요청은 사용자의 MFA 코드를 전달합니다.
9. 사용자 풀은 사용자의 MFA 코드를 검증합니다.
10. 사용자 풀은 사용자의 JWT로 응답합니다.
11. 애플리케이션은 사용자의 JWT를 디코딩, 검증, 저장 또는 캐시합니다.
12. 애플리케이션은 요청된 액세스 제어 구성 요소를 표시합니다.
13. 사용자는 해당 콘텐츠를 봅니다.
14. 나중에 사용자의 액세스 토큰이 만료되었으며 사용자는 액세스 제어 구성 요소를 보도록 요청합니다.
15. 애플리케이션은 사용자 세션이 지속되어야 한다고 판단합니다. 새로 고침 토큰을 사용하여 InitiateAuth메서드를 다시 호출하고 새 토큰을 검색합니다.

변형 및 사용자 지정

사용자 지정 인증 문제와 같은 추가 문제로 이 흐름을 보강할 수 있습니다. 암호가 유출되었거나 예상치 못한 로그인 특성으로 인해 악의적인 로그인 시도가 발생할 수 있는 사용자의 액세스를 자동으로 제한할 수 있습니다. 이 흐름은 가입, 사용자 속성 업데이트, 암호 재설정 작업에서도 거의 비슷해 보입니다. 이러한 흐름의 대부분은 공개 (클라이언트 측) 및 기밀 (서버 측) API 작업이 중복됩니다.

관련 리소스

- 아마존 코그니토 사용자 풀 API
- 사용자 풀 시작하기

- 웹 및 모바일 앱과 Amazon Cognito 인증 및 권한 부여 통합
- Amazon Cognito 사용자 풀 API 및 사용자 풀 엔드포인트 사용

호스팅된 UI를 사용한 사용자 풀 인증

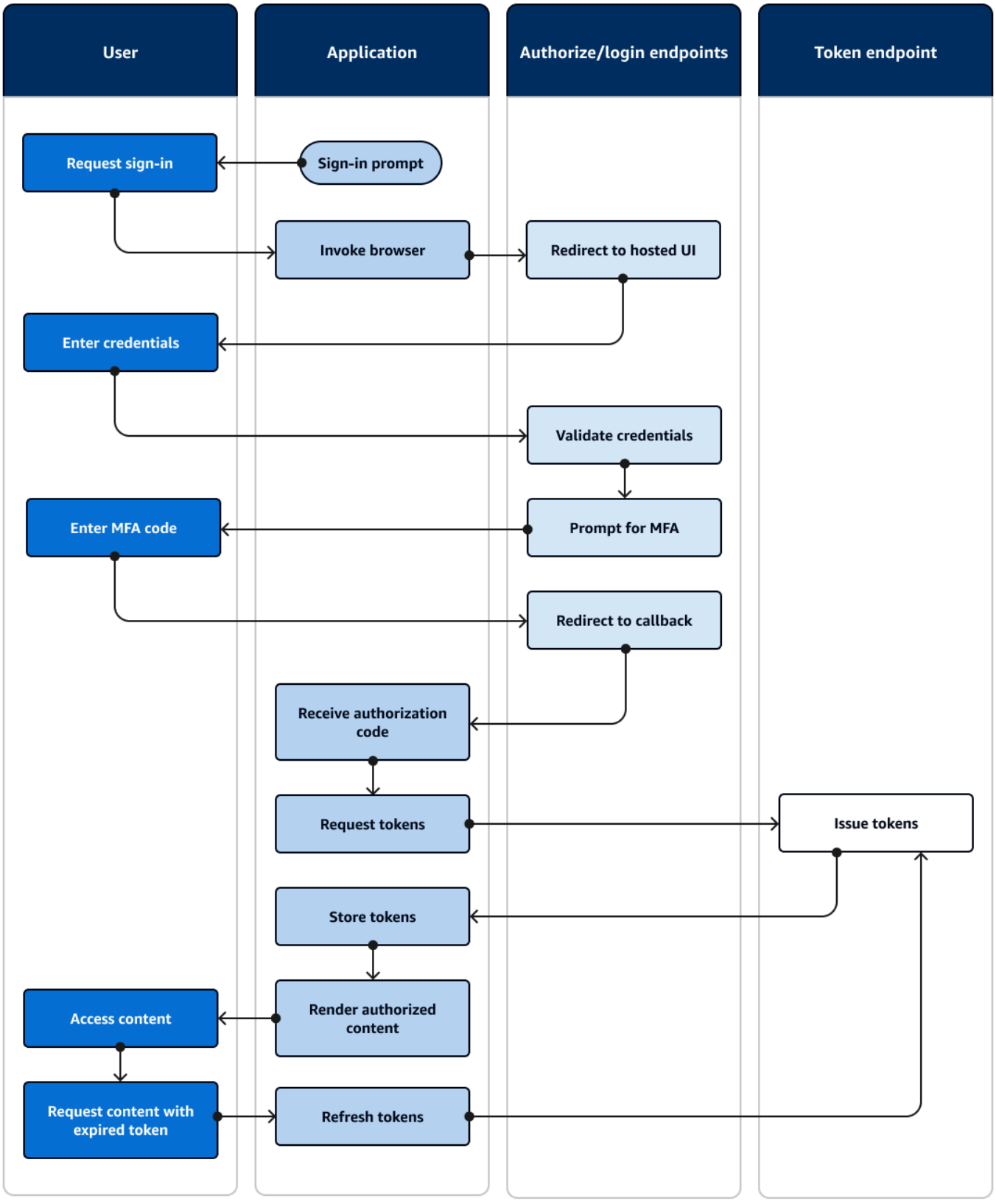
호스팅된 UI는 사용자 풀 및 앱 클라이언트에 연결된 웹 사이트입니다. 사용자를 위해 로그인, 가입 및 암호 재설정 작업을 수행할 수 있습니다. 인증용 호스팅된 UI 구성 요소가 있는 애플리케이션은 개발자의 구현 노력을 줄일 수 있습니다. 애플리케이션은 인증을 위한 UI 구성 요소를 건너뛰고 사용자 브라우저에서 호스팅된 UI를 호출할 수 있습니다.

애플리케이션은 웹 또는 앱 리디렉션 위치를 통해 사용자의 JWT를 수집합니다. 호스팅된 UI를 구현하는 애플리케이션은 OpenID Connect (OIDC) IdP인 것처럼 사용자 풀에 연결하여 인증을 받을 수 있습니다.

호스팅된 UI 인증은 애플리케이션에 권한 부여 서버가 필요하지만 사용자 지정 인증, ID 풀 통합 또는 사용자 속성 셀프 서비스와 같은 기능은 필요하지 않은 모델에 적합합니다. 이러한 고급 옵션 중 일부를 사용하려는 경우 SDK의 사용자 풀 구성 요소를 사용하여 이러한 고급 옵션을 구현할 수 있습니다.

OIDC 구현에 주로 의존하는 호스팅된 UI 및 타사 IdP 인증 모델은 OAuth 2.0 범위를 사용하는 고급 권한 부여 모델에 가장 적합합니다.

다음 다이어그램은 호스팅된 UI 인증을 위한 일반적인 로그인 세션을 보여줍니다.


User
Application
Authorize/login endpoints
Token endpoint
Request sign-in
Sign-in prompt
Invoke browser
Redirect to hosted UI
Enter credentials
Validate credentials
Enter MFA code
Prompt for MFA
Redirect to callback
Receive authorization code
Request tokens
Issue tokens
Store tokens
Access content
Render authorized content
Request content with expired token
Refresh tokens

호스팅된 UI 인증 흐름

1. 사용자가 애플리케이션에 액세스합니다.
2. 사용자는 “로그인” 링크를 선택합니다.
3. 애플리케이션은 사용자를 호스팅된 UI 로그인 프롬프트로 안내합니다.
4. 사용자는 사용자 이름과 비밀번호를 입력합니다.
5. 사용자 풀은 사용자의 자격 증명을 검증하고 사용자가 멀티 팩터 인증 (MFA) 을 활성화했는지 확인합니다.
6. 호스팅된 UI는 사용자에게 MFA 코드를 입력하라는 메시지를 표시합니다.
7. 사용자가 MFA 코드를 입력합니다.
8. 호스팅된 UI는 사용자를 애플리케이션으로 리디렉션합니다.
9. 애플리케이션은 호스팅된 UI가 콜백 URL에 추가한 URL 요청 매개변수에서 인증 코드를 수집합니다.
10. 애플리케이션은 인증 코드와 함께 토큰을 요청합니다.
11. 토큰 엔드포인트는 JWT를 애플리케이션에 반환합니다.
12. 애플리케이션은 사용자의 JWT를 디코딩, 검증, 저장 또는 캐시합니다.
13. 애플리케이션은 요청된 액세스 제어 구성 요소를 표시합니다.
14. 사용자는 해당 콘텐츠를 봅니다.
15. 나중에 사용자의 액세스 토큰이 만료되었으며 사용자는 액세스 제어 구성 요소를 보도록 요청합니다.
16. 애플리케이션은 사용자 세션이 지속되어야 한다고 판단합니다. 새로 고침 토큰을 사용하여 토큰 엔드포인트에서 새 토큰을 요청합니다.

변형 및 사용자 지정

모든 앱 클라이언트에서 CSS를 사용하여 호스팅된 UI의 모양과 느낌을 사용자 지정할 수 있습니다. 또한 자체 ID 공급자, 범위, 사용자 속성에 대한 액세스 및 고급 보안 구성을 사용하여 앱 클라이언트를 구성할 수 있습니다.

관련 리소스

- Amazon Cognito 호스팅 UI 및 페더레이션 엔드포인트 설정 및 사용
- 호스팅 UI로 가입 및 로그인
- 리소스 서버를 통한 범위, M2M 및 API 인증

- 사용자 풀 페더레이션 엔드포인트 및 호스팅 UI 참조

타사 ID 공급자를 통한 사용자 풀 인증

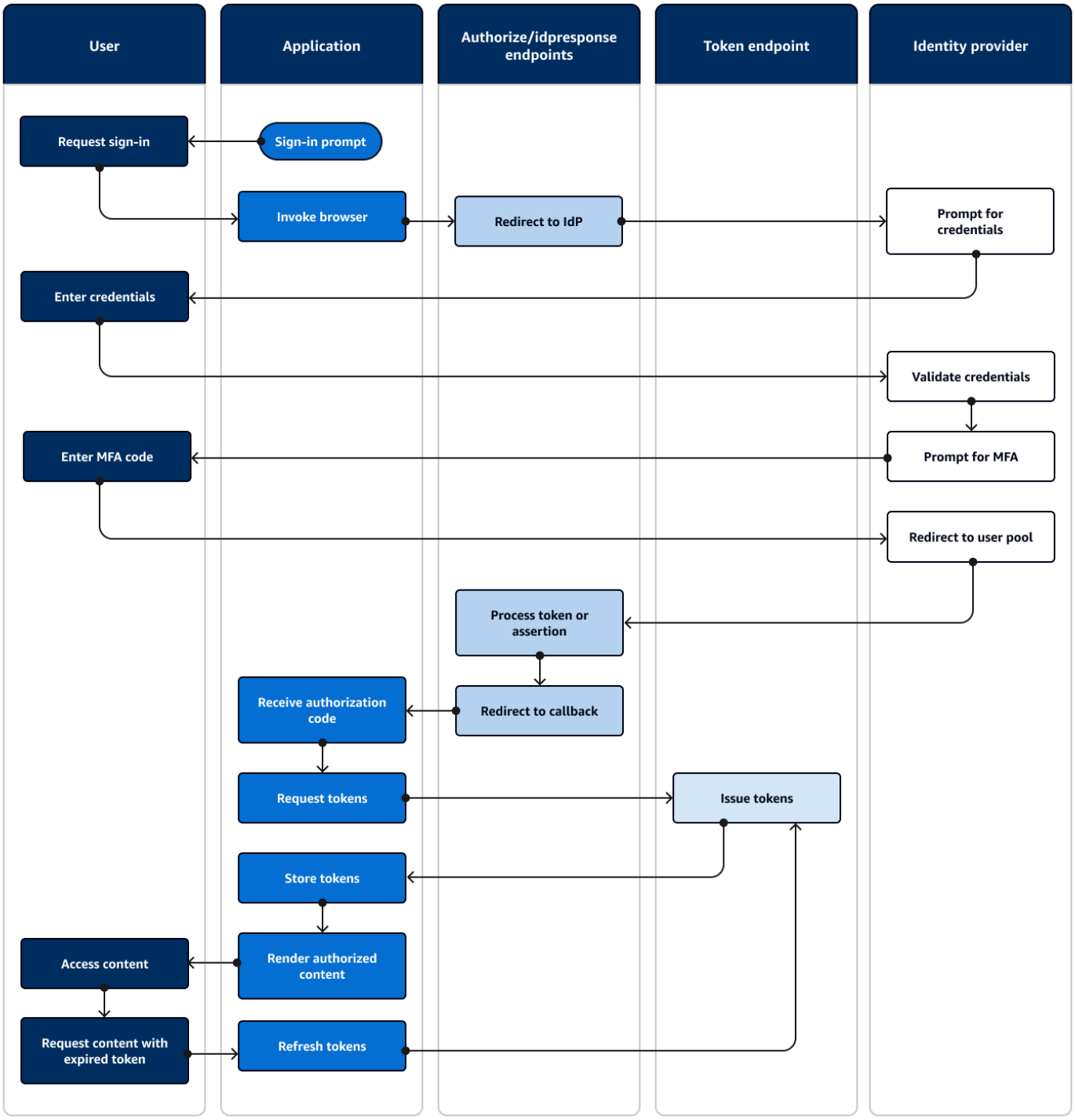
외부 ID 공급자 (IdP) 를 통한 로그인 또는 페더레이션 인증은 호스팅된 UI와 유사한 모델입니다. 애플리케이션은 사용자 풀의 OIDC 의존자이며, 사용자 풀은 IdP에 대한 패스스루 역할을 합니다. IdP는 페이스북이나 구글과 같은 소비자 사용자 디렉터리일 수도 있고 Azure와 같은 SAML 2.0 또는 OIDC 엔터프라이즈 디렉터리일 수도 있습니다.

애플리케이션은 사용자 브라우저의 호스팅된 UI 대신 사용자 풀 인증 서버의 리디렉션 엔드포인트를 호출합니다. 사용자의 관점에서 보면 사용자는 애플리케이션의 로그인 버튼을 선택합니다. 그러면 IdP에 로그인하라는 메시지가 표시됩니다. 호스팅된 UI 인증과 마찬가지로 애플리케이션은 앱의 리디렉션 위치에서 JWT를 수집합니다.

타사 IdP를 사용한 인증은 사용자가 애플리케이션에 가입할 때 새 비밀번호를 제시하고 싶어하지 않는 모델에 적합합니다. 호스팅된 UI 인증을 구현한 애플리케이션에 손쉽게 타사 인증을 추가할 수 있습니다. 실제로 호스팅된 UI와 서드파티 UI는 사용자 브라우저에서 호출하는 내용을 조금만 변경해도 일관된 인증 결과를 IdPs 생성합니다.

호스팅된 UI 인증과 마찬가지로 페더레이션 인증은 OAuth 2.0 범위를 사용하는 고급 인증 모델에 가장 적합합니다.

다음 다이어그램은 페더레이션 인증을 위한 일반적인 로그인 세션을 보여줍니다.



페더레이션 인증 흐름

1. 사용자가 애플리케이션에 액세스합니다.
2. 사용자는 “로그인” 링크를 선택합니다.

3. 애플리케이션은 사용자를 IdP로 로그인 프롬프트로 안내합니다.
4. 사용자는 사용자 이름과 비밀번호를 입력합니다.
5. IdP는 사용자의 자격 증명을 확인하고 사용자가 다단계 인증 (MFA) 을 활성화했는지 확인합니다.
6. IdP는 사용자에게 MFA 코드를 입력하라는 메시지를 표시합니다.
7. 사용자가 MFA 코드를 입력합니다.
8. IdP는 SAML 응답 또는 인증 코드를 사용하여 사용자를 사용자 풀로 리디렉션합니다.
9. 사용자가 인증 코드를 전달한 경우 사용자 풀은 자동으로 코드를 IdP 토큰으로 교환합니다. 사용자 풀은 IdP 토큰의 유효성을 검사하고 새 인증 코드를 사용하여 사용자를 애플리케이션으로 리디렉션합니다.
10. 애플리케이션은 사용자 풀이 콜백 URL에 추가한 URL 요청 매개변수에서 인증 코드를 수집합니다.
11. 애플리케이션은 인증 코드와 함께 토큰을 요청합니다.
12. 토큰 엔드포인트는 JWT를 애플리케이션에 반환합니다.
13. 애플리케이션은 사용자의 JWT를 디코딩, 검증, 저장 또는 캐시합니다.
14. 애플리케이션은 요청된 액세스 제어 구성 요소를 표시합니다.
15. 사용자는 해당 콘텐츠를 봅니다.
16. 나중에 사용자의 액세스 토큰이 만료되었으며 사용자는 액세스 제어 구성 요소를 보도록 요청합니다.
17. 애플리케이션은 사용자 세션이 지속되어야 한다고 판단합니다. 새로 고침 토큰을 사용하여 토큰 엔드포인트에서 새 토큰을 요청합니다.

변형 및 사용자 지정

호스팅된 UI에서 페더레이션 인증을 시작할 수 있습니다. 그러면 사용자가 앱 클라이언트에 할당한 목록 중에서 선택할 수 있습니다. IdPs 호스팅된 UI는 이메일 주소를 입력하라는 메시지를 표시하고 해당 SAML IdP에 사용자 요청을 자동으로 라우팅할 수도 있습니다. 타사 ID 공급자를 통한 인증에는 호스팅된 UI와의 사용자 상호 작용이 필요하지 않습니다. 애플리케이션은 사용자의 권한 부여 서버 요청에 요청 매개변수를 추가하여 사용자가 자동으로 IdP 로그인 페이지로 리디렉션하도록 할 수 있습니다.

관련 리소스

- 서드 파티를 통한 사용자 풀 로그인 추가
- 예제 시나리오: 엔터프라이즈 대시보드에서 Amazon Cognito 앱을 북마크하기
- 리소스 서버를 통한 범위, M2M 및 API 인증
- 사용자 풀 페더레이션 엔드포인트 및 호스팅 UI 참조

자격 증명 풀 인증

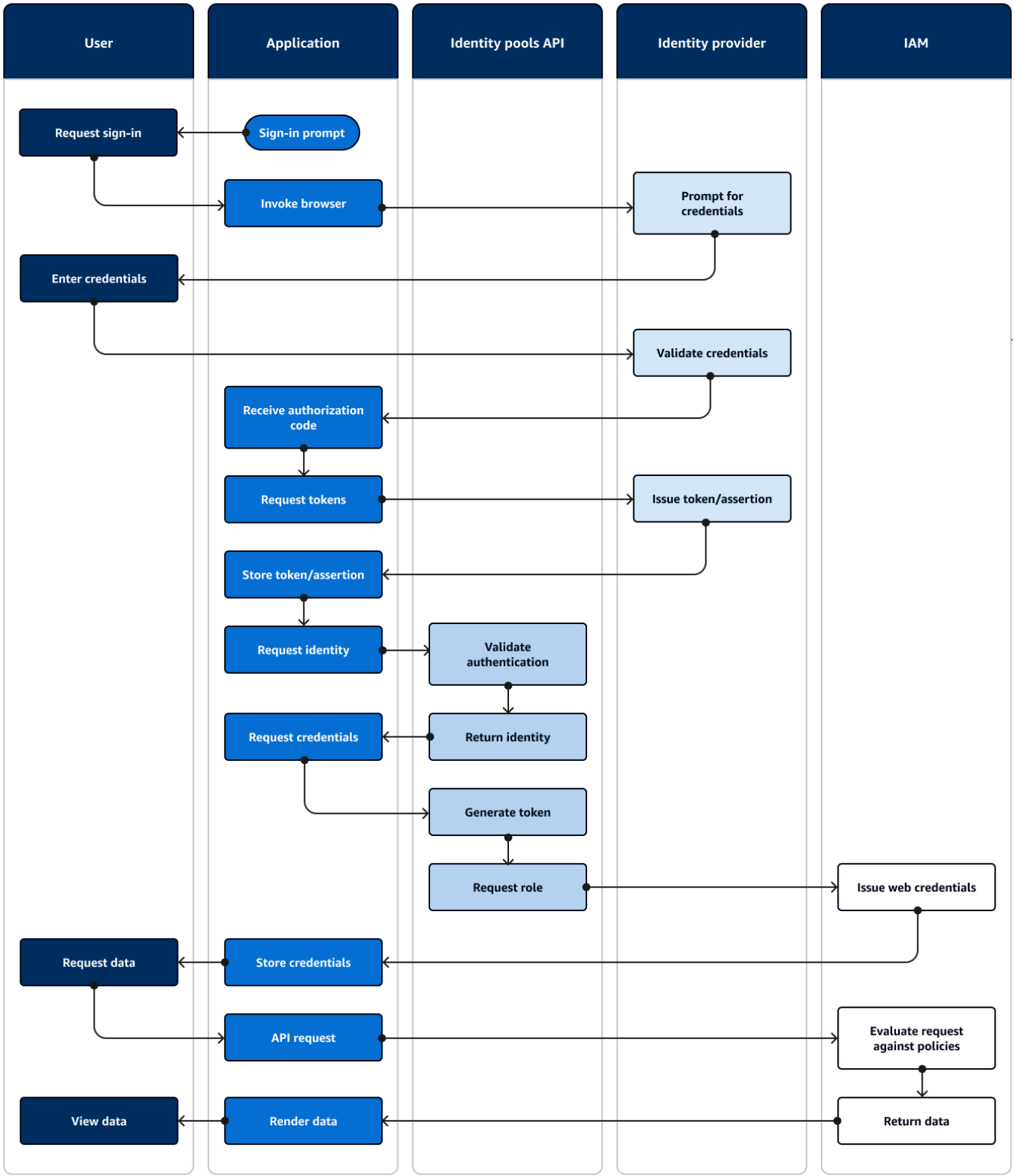
자격 증명 풀은 함수, API 네임스페이스 및 SDK 모델 측면에서 사용자 풀과 구별되는 애플리케이션 구성 요소입니다. 사용자 풀이 토큰 기반 인증 및 권한 부여를 제공하는 반면, 자격 증명 풀은 권한 부여 대상 (IAM) 을 제공합니다. AWS Identity and Access Management

자격 증명 풀에 세트를 할당하고 해당 IdPs 풀로 사용자를 로그인시킬 수 있습니다. 사용자 풀은 자격 증명 IdPs 풀과 긴밀하게 통합되어 ID 풀에 액세스 제어를 위한 가장 많은 옵션을 제공합니다. 동시에 ID 풀을 위한 다양한 인증 옵션이 있습니다. 사용자 풀은 자격 증명 풀에서 임시 AWS 자격 증명으로 가는 경로로 SAML, OIDC, 소셜, 개발자 및 게스트 ID 소스에 합류합니다.

자격 증명 풀을 사용한 인증은 외부 인증으로, 앞서 설명한 사용자 풀 흐름 중 하나 또는 다른 IdP와 독립적으로 개발하는 흐름을 따릅니다. 애플리케이션은 초기 인증을 수행한 후 증명을 자격 증명 풀에 전달하고 그 대가로 임시 세션을 받습니다.

자격 증명 풀을 사용한 인증은 IAM 인증을 통해 애플리케이션 자산 및 데이터에 대한 액세스 제어를 적용하는 모델에 적합합니다. AWS 서비스 사용자 풀의 API 인증과 마찬가지로 성공적인 애플리케이션에는 사용자의 이익을 위해 액세스하려는 각 서비스에 대한 AWS SDK가 포함됩니다. AWS SDK는 자격 증명 풀 인증의 자격 증명을 API 요청에 서명으로 적용합니다.

다음 다이어그램은 IdP를 사용한 ID 풀 인증을 위한 일반적인 로그인 세션을 보여줍니다.


User
Application
Identity pools API
Identity provider
IAM
Request sign-in
Sign-in prompt
Invoke browser
Prompt for credentials
Enter credentials
Validate credentials
Receive authorization code
Request tokens
Issue token/assertion
Store token/assertion
Request identity
Validate authentication
Request credentials
Return identity
Generate token
Request role
Issue web credentials
Request data
Store credentials
API request
Evaluate request against policies
View data
Render data
Return data

페더레이션 인증 흐름

1. 사용자가 애플리케이션에 액세스합니다.
2. 사용자는 “로그인” 링크를 선택합니다.
3. 애플리케이션은 사용자를 IdP로 로그인 프롬프트로 안내합니다.
4. 사용자는 사용자 이름과 비밀번호를 입력합니다.
5. IdP는 사용자의 자격 증명을 확인합니다.
6. IdP는 SAML 응답 또는 인증 코드를 사용하여 사용자를 애플리케이션으로 리디렉션합니다.
7. 사용자가 인증 코드를 전달한 경우 애플리케이션은 코드를 IdP 토큰으로 교환합니다.
8. 애플리케이션은 사용자의 JWT 또는 어설션을 디코딩, 검증, 저장 또는 캐시합니다.
9. 애플리케이션은 API 요청을 보내는 메서드를 호출합니다. GetId 사용자의 토큰 또는 어설션을 전달하고 ID ID를 요청합니다.
10. 자격 증명 풀은 구성된 ID 공급자를 대상으로 토큰 또는 어설션을 검증합니다.
11. 자격 증명 풀은 ID ID를 반환합니다.
12. 애플리케이션은 GetCredentialsForIdentityAPI 요청을 보내는 메서드를 호출합니다. 사용자의 토큰 또는 어설션을 전달하고 IAM 역할을 요청합니다.
13. 자격 증명 풀은 새 JWT를 생성합니다. 새 JWT에는 IAM 역할을 요청하는 클레임이 포함되어 있습니다. 자격 증명 풀은 사용자의 요청과 IdP에 대한 자격 증명 풀 구성의 역할 선택 기준에 따라 역할을 결정합니다.
14. AWS Security Token Service (AWS STS) 는 자격 증명 풀의 AssumeRoleWithWebIdentity요청에 응답합니다. 응답에는 IAM 역할을 사용하는 임시 세션에 대한 API 자격 증명이 포함됩니다.
15. 애플리케이션은 세션 자격 증명을 저장합니다.
16. 사용자는 액세스 보호된 리소스가 필요한 작업을 앱에서 수행합니다. AWS
17. 애플리케이션은 필요한 경우 임시 자격 증명을 API 요청에 서명으로 적용합니다. AWS 서비스
18. IAM은 자격 증명의 역할에 연결된 정책을 평가합니다. 요청과 비교합니다.
19. 요청된 데이터를 AWS 서비스 반환합니다.
20. 애플리케이션은 사용자 인터페이스에서 데이터를 렌더링합니다.
21. 사용자가 데이터를 조회합니다.

변형 및 사용자 지정

사용자 풀을 사용한 인증을 시각화하려면 토큰/어설션 발행 단계 뒤에 이전 사용자 풀 개요 중 하나를 삽입하십시오. 개발자 인증은 ID 요청 이전의 모든 단계를 개발자 자격 증명으로 서명한 요청으로 대체

합니다. 또한 게스트 인증은 ID 요청으로 바로 건너뛰고 인증을 확인하지 않으며 액세스가 제한된 IAM 역할에 대한 자격 증명을 반환합니다.

관련 리소스

- Amazon Cognito 자격 증명 풀
- 사용자 IAM 역할
- 자격 증명 풀 개념
- 자격 증명 풀(페더레이션 자격 증명) 인증 흐름

아마존 코그니토 용어

Amazon Cognito는 웹 및 모바일 앱을 위한 자격 증명을 제공합니다. 자격 증명 및 액세스 관리에서 흔히 사용되는 용어를 바탕으로 구축되었습니다. 범용 ID 및 액세스 용어에 대한 많은 가이드가 제공됩니다. 다음은 몇 가지 예시입니다.

- IDPro 지식 본문의 용어
- AWS 아이덴티티 서비스
- NIST CSRC의 용어집

다음 목록은 Amazon Cognito에 고유하거나 Amazon Cognito에서 특정 컨텍스트를 포함하는 용어를 설명합니다.

주제

- 일반
- 사용자 풀
- 자격 증명 풀

일반

이 목록에 있는 용어는 Amazon Cognito에만 국한된 것이 아니며 ID 및 액세스 관리 전문가들 사이에서 널리 알려져 있습니다. 다음은 전체 용어 목록은 아니지만 이 가이드의 특정 Amazon Cognito 컨텍스트에 대한 가이드입니다.

앱

일반적으로 모바일 애플리케이션입니다. 이 가이드에서 앱은 Amazon Cognito에 연결되는 웹 애플리케이션 또는 모바일 앱의 줄임말인 경우가 많습니다.

속성 기반 액세스 제어(ABAC)

앱이 사용자의 속성 (예: 직책 또는 부서) 을 기반으로 리소스에 대한 액세스 권한을 결정하는 모델입니다. ABAC를 적용하는 Amazon Cognito 도구에는 사용자 풀에 ID 토큰이 포함되고 자격 증명 풀에는 주 태그가 포함됩니다.

권한 부여 서버

JSON 웹 토큰을 생성하는 웹 기반 시스템입니다. Amazon Cognito 사용자 풀 페더레이션 엔드포인트는 사용자 풀의 두 인증 및 권한 부여 방법의 권한 부여 서버 구성 요소입니다. 다른 방법은 사용자 풀 API입니다.

기밀 앱, 서버측 앱

사용자가 애플리케이션 서버의 코드를 사용하여 비밀에 액세스하여 원격으로 연결하는 애플리케이션입니다. 이는 일반적으로 웹 애플리케이션입니다.

ID 제공업체(IdP)

사용자 ID를 저장하고 확인하는 서비스입니다. Amazon Cognito는 외부 공급자에게 인증을 요청하고 앱에 대한 IdP 역할을 할 수 있습니다.

JSON 웹 토큰 (JWT)

인증된 사용자에 대한 클레임이 포함된 JSON 형식의 문서입니다. ID 토큰은 사용자를 인증하고, 액세스 토큰은 사용자를 인증하고, 새로 고침 토큰은 자격 증명을 업데이트합니다. Amazon Cognito는 외부 공급자로부터 토큰을 받아 앱 또는 앱에 토큰을 발행합니다. AWS STS

멀티 팩터 인증(MFA)

사용자가 사용자 이름과 비밀번호를 제공한 후 추가 인증을 제공해야 한다는 요구 사항. Amazon Cognito 사용자 풀에는 로컬 사용자를 위한 MFA 기능이 있습니다.

OAuth 2.0 (소셜) 제공업체

JWT 액세스 및 새로 고침 토큰을 제공하는 사용자 풀 또는 자격 증명 풀에 대한 IdP Amazon Cognito 사용자 풀은 사용자가 인증한 후 소셜 공급자와의 상호 작용을 자동화합니다.

오픈ID 커넥트 (OIDC) 제공업체

사용자 풀 또는 자격 증명 풀에 대한 IdP로, OAuth 사양을 확장하여 ID 토큰을 제공합니다. Amazon Cognito 사용자 풀은 사용자가 인증한 후 OIDC 공급자와의 상호 작용을 자동화합니다.

퍼블릭 앱

코드가 로컬에 저장되고 비밀에 액세스할 수 없는 디바이스에서 독립적으로 작동하는 애플리케이션입니다. 이는 일반적으로 모바일 앱입니다.

리소스 서버

액세스 제어가 가능한 API. 또한 Amazon Cognito 사용자 풀은 리소스 서버를 사용하여 API와 상호 작용하기 위한 구성을 정의하는 구성 요소를 설명합니다.

역할 기반 액세스 제어(RBAC)

사용자의 기능적 지정을 기반으로 액세스 권한을 부여하는 모델입니다. Amazon Cognito 자격 증명 풀은 IAM 역할을 구분하여 RBAC를 구현합니다.

서비스 공급자 (SP), 신뢰 당사자 (RP)

IdP를 사용하여 사용자를 신뢰할 수 있는지 확인하는 애플리케이션입니다. Amazon Cognito는 외부 SP에는 SP 역할을 하고 앱 IdPs 기반 SP에는 IdP 역할을 합니다.

SAML 제공자

사용자가 Amazon Cognito에 전달하는 디지털 서명된 어설 션 문서를 생성하는 사용자 풀 또는 자격 증명 풀에 대한 IdP입니다.

범용 고유 식별자 (UUID)

개체에 적용되는 128비트 레이블입니다. Amazon Cognito UUID는 사용자 풀 또는 자격 증명 풀별로 고유합니다.

사용자 디렉터리

해당 정보를 다른 시스템에 제공하는 사용자 및 해당 속성 모음입니다. Amazon Cognito 사용자 풀은 사용자 디렉터리이며 외부 사용자 디렉터리의 사용자를 통합하기 위한 도구이기도 합니다.

사용자 풀

이 가이드의 다음 목록에 있는 용어는 사용자 풀의 특정 기능 또는 구성을 가리킵니다.

아마존 코그니토 사용자 풀 API

AWS SDK를 사용하여 앱에 추가할 수 있는 인증 및 권한 부여 API 작업 세트입니다. API는 로컬 사용자 및 연결된 사용자를 로그인할 수 있습니다.

조정 인증

잠재적 악의적 활동을 탐지하고 사용자 프로필에 추가 보안을 적용하는 고급 보안 기능입니다.

고급 보안 기능

사용자 보안을 위한 도구를 추가하는 선택적 구성 요소입니다.

앱 클라이언트

사용자 풀의 설정을 하나의 앱에 대한 IdP로 정의하는 구성요소입니다.

콜백 URL, 리디렉션 URI

앱 클라이언트의 설정 및 사용자 풀 페더레이션 엔드포인트에 대한 요청의 매개변수. 콜백 URL은 앱에서 인증된 사용자의 초기 목적지입니다.

손상된 보안 인증

공격자가 알고 있을 수 있는 사용자 비밀번호를 탐지하고 사용자 프로필에 추가 보안을 적용하는 고급 보안 기능입니다.

확인

새 사용자가 로그인할 수 있도록 하기 위한 사전 요구 사항이 충족되었는지 확인하는 프로세스입니다. 확인은 일반적으로 이메일 주소 또는 전화번호 인증을 통해 이루어집니다.

사용자 지정 인증

추가 사용자 문제 및 대응을 정의하는 Lambda 트리거로 인증 프로세스를 확장합니다.

디바이스 인증

MFA를 신뢰할 수 있는 디바이스의 ID를 사용하는 로그인으로 대체하는 인증 프로세스입니다.

외부 제공업체, 타사 제공업체

사용자 풀과 신뢰 관계가 있는 IdP.

페더레이션 사용자

외부 공급자로부터 인증을 받은 사용자 풀의 사용자.

페더레이션 엔드포인트

사용자 풀 도메인의 웹 페이지 세트로, 상호 작용을 위한 서비스 IdPs 및 앱을 호스팅합니다.

호스팅된 UI

사용자 풀 도메인의 대화형 웹 페이지 세트로, 사용자 인증을 위한 서비스를 호스팅합니다.

Lambda 트리거

사용자 AWS Lambda 풀이 사용자 인증 프로세스의 주요 지점에서 자동으로 호출할 수 있는 기능입니다. Lambda 트리거를 사용하여 인증 결과를 사용자 지정할 수 있습니다.

로컬 사용자

외부 공급자와의 인증을 통해 생성되지 않은 사용자 풀 사용자 디렉터리의 사용자 프로필입니다.

연결된 사용자

ID가 로컬 사용자와 병합된 외부 공급자의 사용자.

토큰 사용자 지정

런타임에 사용자 ID 또는 액세스 토큰을 수정하는 사전 토큰 생성 Lambda 트리거의 결과.

사용자 풀, 아마존 Cognito 자격 증명 공급자,**`cognito-idp`**, Amazon Cognito 사용자 풀

OIDC와 연동되는 애플리케이션을 위한 인증 및 권한 부여 서비스가 포함된 AWS 리소스입니다. IdPs

사용자 풀 도메인

사용자 풀에 추가하는 웹 사이트 이름. 도메인은 호스팅된 UI 및 페더레이션 엔드포인트의 기본 URL입니다.

확인

사용자가 이메일 주소 또는 전화번호를 소유하고 있는지 확인하는 프로세스입니다. 사용자 풀은 새 이메일 주소 또는 전화번호를 입력한 사용자에게 코드를 보냅니다. Amazon Cognito에 코드를 제출하면 메시지 대상의 소유권을 확인하고 사용자 풀에서 추가 메시지를 수신할 수 있습니다. 또한 확인을 참조하십시오.

사용자 프로필, 사용자 계정

사용자 디렉터리에 있는 사용자의 항목입니다. 모든 사용자의 사용자 풀에는 프로필이 있습니다.

자격 증명 풀

이 가이드의 다음 목록에 있는 용어는 자격 증명 풀의 특정 기능 또는 구성을 가리킵니다.

액세스 제어를 위한 속성

자격 증명 풀에서의 속성 기반 액세스 제어 구현. 자격 증명 풀은 사용자 속성을 사용자 자격 증명에 태그로 적용합니다.

기본 (클래식) 인증

사용자 자격 증명 요청을 사용자 지정할 수 있는 인증 프로세스입니다.

개발자 인증 자격 증명

개발자 자격 증명으로 자격 증명 풀 사용자 자격 증명을 승인하는 인증 프로세스입니다.

개발자 자격 증명

자격 증명 풀 관리자의 IAM API 키.

향상된 인증

자격 증명 풀에서 정의한 로직에 따라 IAM 역할을 선택하고 보안 주체 태그를 적용하는 인증 흐름입니다.

자격 증명

앱 사용자와 사용자 자격 증명을 자격 증명 풀과 신뢰 관계가 있는 외부 사용자 디렉터리의 프로필에 연결하는 UUID입니다.

자격 증명 풀, 아마존 Cognito 페더레이션 자격 증명, 아마존 Cognito 자격 증명, **cognito-identity**

임시 자격 증명을 사용하는 애플리케이션을 위한 인증 및 권한 부여 서비스가 포함된 AWS 리소스입니다. AWS

인증되지 않은 자격 증명

자격 증명 풀 IdP로 로그인하지 않은 사용자 사용자가 인증을 받기 전에 단일 IAM 역할에 대해 제한된 사용자 자격 증명을 생성하도록 허용할 수 있습니다.

사용자 보안 인증

자격 증명 풀 인증 후 사용자가 받는 임시 AWS API 키.

이 서비스를 SDK와 함께 사용 AWS

AWS 소프트웨어 개발 키트 (SDK) 는 널리 사용되는 여러 프로그래밍 언어에 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 예제
AWS CLI	AWS CLI 코드 예제
AWS SDK for Go	AWS SDK for Go 코드 예제
AWS SDK for Java	AWS SDK for Java 코드 예제
AWS SDK for JavaScript	AWS SDK for JavaScript 코드 예제
AWS SDK for Kotlin	AWS SDK for Kotlin 코드 예제
AWS SDK for .NET	AWS SDK for .NET 코드 예제
AWS SDK for PHP	AWS SDK for PHP 코드 예제
AWS Tools for PowerShell	PowerShell 코드 예제를 위한 도구
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 코드 예제
AWS SDK for Ruby	AWS SDK for Ruby 코드 예제
AWS SDK for Rust	AWS SDK for Rust 코드 예제
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP 코드 예제
AWS SDK for Swift	AWS SDK for Swift 코드 예제

예제 사용 가능 여부

필요한 예제를 찾을 수 없습니까? 이 페이지 하단의 피드백 제공 링크를 사용하여 코드 예시를 요청하세요.

시작하기 AWS

Amazon Cognito로 작업을 시작하기 전에 몇 가지 필수 AWS 리소스를 준비하십시오.

가입하여 다음을 수행하십시오. AWS 계정

계정이 없는 경우 다음 단계를 완료하여 계정을 만드세요. AWS 계정

가입하려면 AWS 계정

1. https://portal.aws.amazon.com/billing/signup을 여세요.
2. 온라인 지시 사항을 따르세요.

   등록 절차 중에는 전화를 받고 키패드로 인증 코드를 입력하는 과정이 있습니다.

   에 AWS 계정가입하면 AWS 계정 루트 사용자a가 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스 액세스 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

AWS 가입 절차가 완료된 후 확인 이메일을 보냅니다. 언제든지 https://aws.amazon.com/으로 가서 내 계정(My Account)을 선택하여 현재 계정 활동을 보고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

등록한 AWS 계정후에는 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 보호하고 AWS IAM Identity Center활성화하고 생성하십시오 AWS 계정 루트 사용자.

보안을 유지하세요. AWS 계정 루트 사용자

1. 루트 사용자를 선택하고 AWS 계정 이메일 주소를 입력하여 계정 AWS Management Console소유자로 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

   루트 사용자를 사용하여 로그인하는 데 도움이 필요하면AWS 로그인 사용 설명서의 루트 사용자로 로그인을 참조하세요.
2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

   지침은 IAM 사용 설명서의 AWS 계정 루트 사용자 (콘솔) 에 대한 가상 MFA 디바이스 활성화를 참조하십시오.

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

지침은 AWS IAM Identity Center 사용 설명서의 AWS IAM Identity Center설정을 참조하세요.

2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

   를 ID 소스로 사용하는 방법에 대한 자습서는 사용 설명서의 기본값으로 IAM Identity Center 디렉터리사용자 액세스 구성을 참조하십시오. IAM Identity Center 디렉터리 AWS IAM Identity Center

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 사용자를 사용하여 로그인하는 데 도움이 필요하면 사용 설명서의 AWS 액세스 포털에 로그인을 참조하십시오.AWS 로그인

추가 사용자에게 액세스 권한 할당

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

   지침은AWS IAM Identity Center 사용 설명서의 Create a permission set를 참조하세요.

2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

   지침은AWS IAM Identity Center 사용 설명서의 Add groups를 참조하세요.

사용자 풀 시작하기

이 섹션의 가이드를 사용하여 초기 사용자 풀 리소스를 만들 수 있습니다. step-by-step 연습을 위해 React JavaScript 개발자 환경의 기본 웹 애플리케이션부터 시작하세요. 여기서부터 호스팅된 사용자 인터페이스 (호스팅된 UI), 외부 소셜 또는 SAML 2.0 ID 공급자와의 페더레이션된 로그인 () 과 같은 기능을 계속 추가할 수 있습니다. IdPs

기능 세트를 확장하고 Amazon Cognito의 더 많은 구성 요소를 통합하려는 경우, Amazon Cognito 사용자 풀 장에서 사용자 풀로 수행할 수 있는 모든 작업에 대한 전체 설명을 읽어보십시오.

이 섹션의 예제 사용자 풀 및 애플리케이션은 애플리케이션 리소스와 Amazon Cognito 사용자 풀의 기본 통합을 보여줍니다. 나중에 사용 가능한 옵션을 더 많이 사용하도록 사용자 풀을 조정할 수 있습니다. 그런 다음 애플리케이션을 업데이트하여 새 API를 채택하고 호스팅된 UI와 IdPs 상호 작용할 수 있습니다.

이 섹션의 자습서에서는 사용자 지정 UI와 SDK를 사용한 API 기반 인증을 사용하는 애플리케이션을 생성합니다. AWS 이러한 방식으로 빌드한 애플리케이션은 로컬 사용자를 인증하는 데 적합합니다. 미리 빌드된 UI, 일부 사용자 풀 기능의 자동 처리 및 연동 사용자 인증을 포함하는 애플리케이션으로 시작하려면 다음으로 건너뛰십시오. 호스팅된 UI가 있는 앱 클라이언트 추가

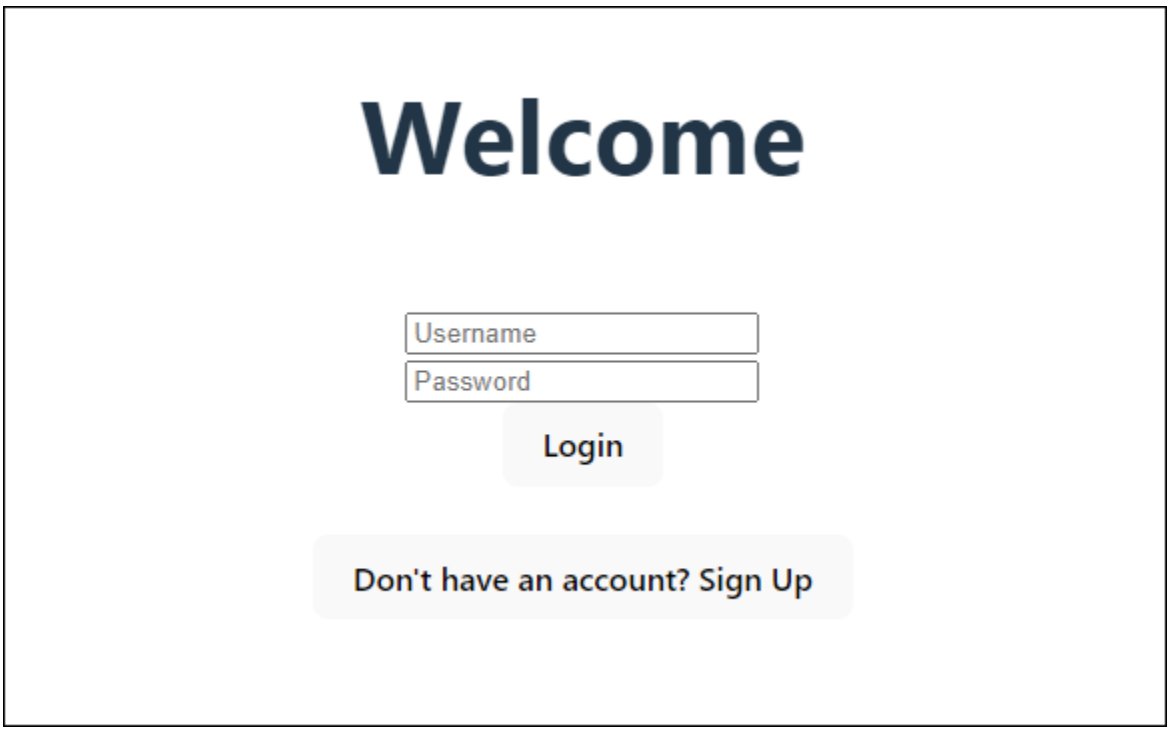
주제

- 예제 React 단일 페이지 애플리케이션을 설정하세요.
- Flutter로 예제 안드로이드 앱을 설정해 보세요.
- 다음 단계

예제 React 단일 페이지 애플리케이션을 설정하세요.

이 튜토리얼에서는 사용자 가입, 확인, 로그인을 테스트할 수 있는 React 단일 페이지 애플리케이션을 만들어 보겠습니다. React는 사용자 인터페이스 (UI) 에 중점을 둔 웹 및 모바일 앱용 JavaScript 기반 라이브러리입니다. 이 예제 애플리케이션은 Amazon Cognito 사용자 풀의 몇 가지 기본 기능을 보여줍니다. 이미 React를 사용한 웹 앱 개발 경험이 있다면 에서 예제 앱을 다운로드하십시오. GitHub

다음 스크린샷은 생성할 애플리케이션의 초기 인증 페이지입니다.



사용자 풀 생성 프로시저는 예제 애플리케이션과 호환되는 사용자 풀을 설정합니다. 다음 요구 사항을 충족하는 사용자 풀이 있는 경우 이 단계를 건너뛸 수 있습니다.

- 사용자는 자신의 이메일 주소로 로그인할 수 있습니다. Cognito 사용자 풀 로그인 옵션: 이메일.
- 사용자 이름은 대소문자를 구분하지 않습니다. 사용자 이름 요구 사항: 사용자 이름 대소문자를 구분하도록 설정이 선택되지 않았습니다.
- 다단계 인증 (MFA) 은 필요하지 않습니다. MFA 적용: 선택적 MFA.
- 사용자 풀은 이메일 메시지로 사용자 프로필 확인을 위한 속성을 확인합니다. 확인할 속성: 이메일 메시지 전송, 이메일 주소 확인.
- 이메일은 유일한 필수 속성입니다. 필수 속성: 이메일.
- 사용자는 사용자 풀에 자신을 등록할 수 있습니다. 셀프 등록: 셀프 등록 활성화가 선택되었습니다.
- 초기 앱 클라이언트는 사용자 이름과 비밀번호로 로그인할 수 있는 공개 클라이언트입니다. 앱 유형: 퍼블릭 클라이언트, 인증 흐름:. `ALLOW_USER_PASSWORD_AUTH`

사용자 풀 생성

새 사용자 풀 생성

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 AWS 자격 증명을 입력합니다.

2. 사용자 풀 생성 버튼을 선택합니다. 이 옵션을 표시하려면 왼쪽 탐색 창에서 사용자 풀을 선택해야 할 수도 있습니다.
3. 페이지 오른쪽 상단에서 사용자 풀 생성(Create a user pool)을 선택하여 사용자 풀 생성 마법사를 시작합니다.
4. 로그인 환경 구성에서 이 사용자 풀에 사용할 ID 공급자 (IdPs) 를 선택할 수 있습니다. 자세한 정보는 서드 파티를 통한 사용자 풀 로그인 추가을 참조하세요.

   a. 인증 제공자에서 공급자 유형에는 Cognito 사용자 풀만 선택되어 있는지 확인합니다.
   b. Cognito 사용자 풀 로그인 옵션의 경우 사용자 이름을 선택합니다. 추가 사용자 이름 요구 사항을 선택하지 마세요.
   c. 다른 모든 옵션은 기본값으로 유지하고 다음을 선택합니다.

5. 보안 요구 사항 구성에서 암호 정책, 다단계 인증 (MFA) 요구 사항 및 사용자 계정 복구 옵션을 선택할 수 있습니다. 자세한 정보는 Amazon Cognito 사용자 풀의 보안 기능 사용을 참조하세요.

   a. 암호 정책의 경우 암호 정책 모드가 Cognito 기본값으로 설정되어 있는지 확인합니다.
   b. 멀티 팩터 인증에서 MFA를 적용하려면 옵션 MFA를 선택합니다.
   c. MFA 방법의 경우 인증자 앱 및 SMS 메시지를 선택합니다.
   d. 사용자 계정 복구의 경우 셀프 서비스 계정 복구 활성화가 선택되어 있고 사용자 계정 복구 메시지 전송 방법이 이메일 전용으로 설정되어 있는지 확인합니다.
   e. 다른 모든 옵션은 기본값으로 유지하고 다음을 선택합니다.

6. 가입 환경 구성에서 새 사용자로 가입할 때 새 사용자가 자신의 ID를 확인하는 방법과 사용자 가입 과정에서 필수 또는 선택 사항으로 설정해야 하는 속성을 결정할 수 있습니다. 자세한 정보는 사용자 풀의 사용자 관리을 참조하세요.

   a. 셀프 등록 활성화가 선택되었는지 확인합니다. 이 설정을 사용하면 사용자 풀이 열려 인터넷상의 모든 사용자가 가입할 수 있습니다. 이는 예제 애플리케이션을 위한 것이지만 프로덕션 환경에서는 이 설정을 주의해서 적용하십시오.
   b. Cognito 지원 확인 및 확인에서 Cognito가 확인 및 확인을 위해 메시지를 자동으로 보내도록 허용 확인란이 선택되어 있는지 확인합니다.
   c. 확인할 속성이 이메일 메시지 전송, 이메일 주소 확인으로 설정되어 있는지 확인합니다.
   d. 속성 변경 확인에서 기본 옵션이 선택되어 있는지 확인합니다. 즉, 업데이트 보류 시 원래 속성 값 유지가 선택되고, 업데이트 보류 중 활성 속성 값이 이메일 주소로 설정되어 있는지 확인합니다.
   e. 필수 속성에서 이전 선택 항목을 기반으로 하는 필수 속성에 이메일이 표시되는지 확인합니다.

> **Important**
>
> 이 예제 응용 프로그램의 경우 사용자 풀이 phone_number를 필수 속성으로 설정해서는 안 됩니다. phone_number가 필수 속성으로 표시된 경우 이전 선택 사항을 검토하고 업데이트하십시오.
>
> - 사용자 계정 복구 메시지를 위한 선택적 MFA, 이메일 전용 전송 방법
> - 이메일 메시지 전송, 확인할 속성에 대한 이메일 주소 확인

f. 다른 모든 옵션은 기본값으로 유지하고 다음을 선택합니다.

7. 메시지 전송 구성에서 가입, 계정 확인, MFA 및 계정 복구를 위해 사용자에게 이메일 및 SMS 메시지를 보내도록 Amazon Simple Email Service 및 Amazon Simple Notification Service와의 통합을 구성할 수 있습니다. 자세한 내용은 Amazon Cognito 사용자 풀에 대한 이메일 설정 및 Amazon Cognito 사용자 풀의 SMS 메시지 설정 섹션을 참조하세요.

   a. 이메일 공급자의 경우 Cognito로 이메일 보내기를 선택하고 Amazon Cognito에서 제공하는 기본 이메일 발신자를 사용합니다. 이메일 용량이 적은 이 설정은 애플리케이션 테스트에 충분합니다. Amazon Simple Email Service (Amazon SES) 를 통해 이메일 주소를 확인하고 Amazon SES로 이메일 보내기를 선택하면 반송할 수 있습니다.

   b. SMS의 경우 새 IAM 역할 생성을 선택하고 IAM 역할 이름을 입력합니다. 그러면 Amazon Cognito에 SMS 메시지를 전송할 수 있는 권한을 부여하는 역할이 생성됩니다.

   c. 다른 모든 옵션은 기본값으로 유지하고 다음을 선택합니다.

8. 앱 통합에서 사용자 풀의 이름을 지정하고, 호스팅된 UI를 구성하고, 앱 클라이언트를 생성할 수 있습니다. 자세한 정보는 호스팅된 UI가 있는 앱 클라이언트 추가을 참조하세요. 예제 애플리케이션은 호스팅된 UI를 사용하지 않습니다.

   a. 사용자 풀 이름에 사용자 풀 이름을 입력합니다.

   b. Cognito 호스팅 UI 사용을 선택하지 마세요.

   c. 초기 앱 클라이언트에서 앱 유형이 퍼블릭 클라이언트로 설정되어 있는지 확인합니다.

   d. 클라이언트 암호에서 클라이언트 암호를 생성하지 않음이 선택되어 있는지 확인합니다.

   e. [앱 클라이언트 이름(App client name)]을 입력합니다.

   f. 고급 앱 클라이언트 설정을 확장합니다. 인증 흐름 목록에 `ALLOW_USER_PASSWORD_AUTH` 추가합니다.

   g. 다른 모든 옵션은 기본값으로 유지하고 다음을 선택합니다.

9. 검토 및 작성 화면에서 선택 사항을 검토하고 필요에 따라 선택 내용을 수정합니다. 사용자 풀 구성에 만족하면 사용자 풀 생성을 선택하여 계속 진행하십시오.
10. 사용자 풀 페이지에서 새 사용자 풀을 선택합니다.
11. 사용자 풀 개요에서 사용자 풀 ID를 기록해 둡니다. 예제 애플리케이션을 만들 때 이 문자열을 제공하게 됩니다.
12. 앱 통합 탭을 선택하고 앱 클라이언트 및 분석 섹션을 찾으십시오. 새 앱 클라이언트를 선택합니다. 클라이언트 ID를 기록해 둡니다.

관련 리소스

- Amazon Cognito 사용자 풀
- 사용자 풀 인증 흐름
- 사용자 풀에 토큰 사용

애플리케이션 생성

이 애플리케이션을 빌드하려면 개발자 환경을 설정해야 합니다. 개발자 환경 요구 사항은 다음과 같습니다.

1. Node.js 설치 및 업데이트되었습니다.
2. 노드 패키지 관리자 (npm) 가 설치되고 버전 10.2.3 이상으로 업데이트되었습니다.
3. 웹 브라우저의 TCP 포트 5173에서 환경에 액세스할 수 있습니다.

예제 React 웹 애플리케이션을 만들려면

1. 개발자 환경에 로그인하고 애플리케이션의 상위 디렉토리로 이동합니다.

```
cd ~/path/to/project/folder/
```

2. 새 React 서비스를 만드세요.

```
npm create vite@latest frontend-client -- --template react-ts
```

3. 의 AWS 코드 예제 저장소에서 `cognito-developer-guide-react-example` 프로젝트 폴더를 GitHub 복제합니다.

```
cd ~/some/other/path
```

```
git clone https://github.com/awsdocs/aws-doc-sdk-examples.git
```

```
cp -r ./aws-doc-sdk-examples/javascriptv3/example_code/cognito-identity-provider/
scenarios/cognito-developer-guide-react-example/frontend-client ~/path/to/project/
folder/frontend-client
```

4. 프로젝트의 src 디렉터리로 이동합니다.

```
cd ~/path/to/project/folder/frontend-client/src
```

5. 다음 값을 config.ts 편집하고 바꾸십시오.

   a. YOUR_AWS_REGION AWS 리전 코드로 바꾸십시오. 예를 들면 us-east-1입니다.

   b. 테스트용으로 지정한 사용자 풀의 YOUR_COGNITO_USER_POOL_ID ID로 바꾸십시오. 예를 들면 us-east-1_EXAMPLE입니다. 사용자 풀은 이전 단계에서 입력한 AWS 리전 것과 같아야 합니다.

   c. 테스트용으로 지정한 앱 클라이언트의 YOUR_COGNITO_APP_CLIENT_ID ID로 바꾸십시오. 예를 들면 1example23456789입니다. 앱 클라이언트는 이전 단계의 사용자 풀에 있어야 합니다.

6. 다른 localhost IP에서 예제 애플리케이션에 액세스하려면 해당 줄을 package.json 편집하고 "dev": "vite", 로 "dev": "vite --host 0.0.0.0", 변경하십시오.

7. 애플리케이션을 설치합니다.

```
npm install
```

8. 애플리케이션을 실행합니다.

```
npm run dev
```

9. http://localhost:5173또는 에서 웹 브라우저를 사용하여 애플리케이션에 http://[IP address]:5173 액세스합니다.

10. 유효한 이메일 주소로 새 사용자를 등록하십시오.

11. 이메일 메시지에서 확인 코드를 검색하세요. 신청서에 확인 코드를 입력합니다.

12. 사용자 이름과 비밀번호로 로그인합니다.

Amazon Lightsail을 사용하여 React 개발자 환경 만들기

Amazon Lightsail을 사용하여 가상 클라우드 서버를 생성하면 이 애플리케이션을 빠르게 시작할 수 있습니다.

Lightsail을 사용하면 이 예제 애플리케이션의 사전 요구 사항으로 사전 구성된 소규모 서버 인스턴스를 빠르게 만들 수 있습니다. 브라우저 기반 클라이언트를 사용하여 인스턴스에 SSH로 연결하고 퍼블릭 또는 프라이빗 IP 주소로 웹 서버에 연결할 수 있습니다.

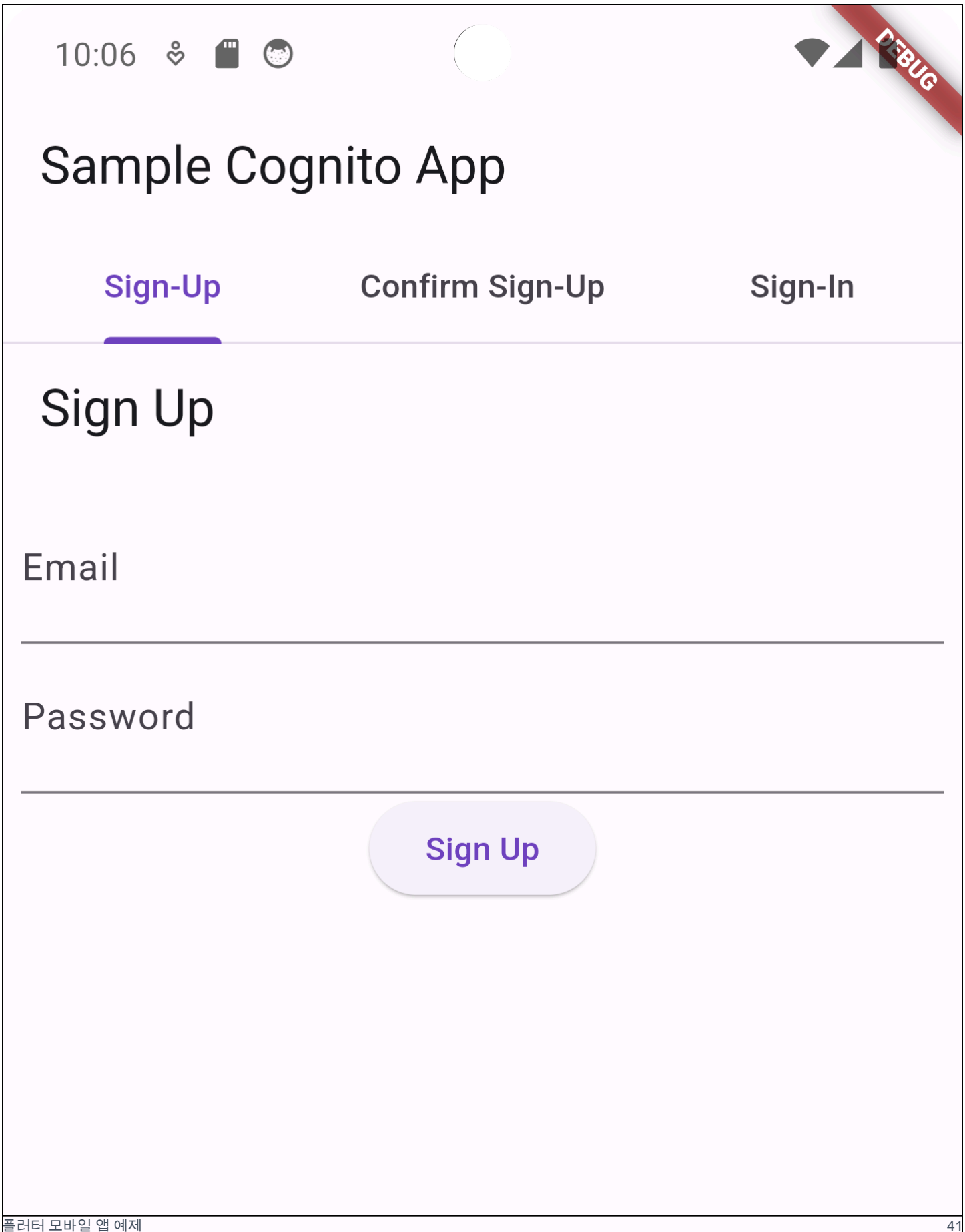
이 예제 애플리케이션을 위한 Lightsail 인스턴스를 만들려면

1. Lightsail 콘솔로 이동합니다. 메시지가 표시되면 자격 증명을 입력합니다. AWS
2. 인스턴스 생성을 선택합니다.
3. 플랫폼 선택에서 Linux/Unix를 선택합니다.
4. 블루프린트 선택에서 Node.js 를 선택합니다.
5. 인스턴스 식별에서 개발 환경에 친숙한 이름을 지정합니다.
6. 인스턴스 생성을 선택합니다.
7. Lightsail에서 인스턴스를 생성한 후 인스턴스를 선택하고 Connect 탭에서 SSH를 사용한 연결을 선택합니다.
8. SSH 세션이 브라우저 창에서 열립니다. `npm -v`실행하여 인스턴스에 Node.js `node -v` 및 최소 npm 버전 10.2.3이 프로비저닝되었는지 확인합니다.
9. React 애플리케이션 구성을 진행하세요.

Flutter로 예제 안드로이드 앱을 설정해 보세요.

이 자습서에서는 Android Studio에서 기기를 에뮬레이션하고 사용자 가입, 확인, 로그인을 테스트할 수 있는 모바일 애플리케이션을 만들어 보겠습니다. 이 예제 애플리케이션은 Flutter에서 안드로이드용 기본 Amazon Cognito 사용자 풀 모바일 클라이언트를 생성합니다. 이미 Flutter를 사용한 모바일 앱 개발 경험이 있다면 에서 예제 앱을 다운로드하십시오. GitHub

다음 스크린샷은 가상 Android 기기에서 실행되는 앱을 보여줍니다.


10:06
DEBUG
Sample Cognito App
Sign-Up
Confirm Sign-Up
Sign-In
Sign Up
Email
Password
Sign Up

사용자 풀 생성 절차는 예제 애플리케이션과 호환되는 사용자 풀을 설정합니다. 다음 요구 사항을 충족하는 사용자 풀이 있는 경우 이 단계를 건너뛸 수 있습니다.

- 사용자는 자신의 이메일 주소로 로그인할 수 있습니다. Cognito 사용자 풀 로그인 옵션: 이메일.
- 사용자 이름은 대소문자를 구분하지 않습니다. 사용자 이름 요구 사항: 사용자 이름 대소문자를 구분하도록 설정이 선택되지 않았습니다.
- 다단계 인증 (MFA) 은 필요하지 않습니다. MFA 적용: 선택적 MFA.
- 사용자 풀은 이메일 메시지로 사용자 프로필 확인을 위한 속성을 확인합니다. 확인할 속성: 이메일 메시지 전송, 이메일 주소 확인.
- 이메일은 유일한 필수 속성입니다. 필수 속성: 이메일.
- 사용자는 사용자 풀에 자신을 등록할 수 있습니다. 셀프 등록: 셀프 등록 활성화가 선택되었습니다.
- 초기 앱 클라이언트는 사용자 이름과 비밀번호로 로그인할 수 있는 공개 클라이언트입니다. 앱 유형: 퍼블릭 클라이언트, 인증 흐름:. `ALLOW_USER_PASSWORD_AUTH`

사용자 풀 생성

새 사용자 풀 생성

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 AWS 자격 증명을 입력합니다.
2. 사용자 풀 생성 버튼을 선택합니다. 이 옵션을 표시하려면 왼쪽 탐색 창에서 사용자 풀을 선택해야 할 수도 있습니다.
3. 페이지 오른쪽 상단에서 사용자 풀 생성(Create a user pool)을 선택하여 사용자 풀 생성 마법사를 시작합니다.
4. 로그인 환경 구성에서 이 사용자 풀에 사용할 ID 공급자 (IdPs) 를 선택할 수 있습니다. 자세한 정보는 서드 파티를 통한 사용자 풀 로그인 추가을 참조하세요.

   a. 인증 제공자에서 공급자 유형에는 Cognito 사용자 풀만 선택되어 있는지 확인합니다.

   b. Cognito 사용자 풀 로그인 옵션의 경우 사용자 이름을 선택합니다. 추가 사용자 이름 요구 사항을 선택하지 마세요.

   c. 다른 모든 옵션은 기본값으로 유지하고 다음을 선택합니다.
5. 보안 요구 사항 구성에서 암호 정책, 다단계 인증 (MFA) 요구 사항 및 사용자 계정 복구 옵션을 선택할 수 있습니다. 자세한 정보는 Amazon Cognito 사용자 풀의 보안 기능 사용을 참조하세요.

   a. 암호 정책의 경우 암호 정책 모드가 Cognito 기본값으로 설정되어 있는지 확인합니다.

   b. 멀티 팩터 인증에서 MFA를 적용하려면 옵션 MFA를 선택합니다.

c. MFA 방법의 경우 인증자 앱 및 SMS 메시지를 선택합니다.

d. 사용자 계정 복구의 경우 셀프 서비스 계정 복구 활성화가 선택되어 있고 사용자 계정 복구 메시지 전송 방법이 이메일 전용으로 설정되어 있는지 확인합니다.

e. 다른 모든 옵션은 기본값으로 유지하고 다음을 선택합니다.

6. 가입 환경 구성에서 새 사용자로 가입할 때 새 사용자가 자신의 ID를 확인하는 방법과 사용자 가입 과정에서 필수 또는 선택 사항으로 설정해야 하는 속성을 결정할 수 있습니다. 자세한 정보는 사용자 풀의 사용자 관리을 참조하세요.

   a. 셀프 등록 활성화가 선택되었는지 확인합니다. 이 설정을 사용하면 사용자 풀이 열려 인터넷 상의 모든 사용자가 가입할 수 있습니다. 이는 예제 애플리케이션을 위한 것이지만 프로덕션 환경에서는 이 설정을 주의해서 적용하십시오.

   b. Cognito 지원 확인 및 확인에서 Cognito가 확인 및 확인을 위해 메시지를 자동으로 보내도록 허용 확인란이 선택되어 있는지 확인합니다.

   c. 확인할 속성이 이메일 메시지 전송, 이메일 주소 확인으로 설정되어 있는지 확인합니다.

   d. 속성 변경 확인에서 기본 옵션이 선택되어 있는지 확인합니다. 즉, 업데이트 보류 시 원래 속성 값 유지가 선택되고, 업데이트 보류 중 활성 속성 값이 이메일 주소로 설정되어 있는지 확인합니다.

   e. 필수 속성에서 이전 선택 항목을 기반으로 하는 필수 속성에 이메일이 표시되는지 확인합니다.

   > **Important**
   >
   > 이 예제 응용 프로그램의 경우 사용자 풀이 phone_number를 필수 속성으로 설정해서는 안 됩니다. phone_number가 필수 속성으로 표시된 경우 이전 선택 사항을 검토하고 업데이트하십시오.
   >
   > - 사용자 계정 복구 메시지를 위한 선택적 MFA, 이메일 전용 전송 방법
   > - 이메일 메시지 전송, 확인할 속성에 대한 이메일 주소 확인

   f. 다른 모든 옵션은 기본값으로 유지하고 다음을 선택합니다.

7. 메시지 전송 구성에서 가입, 계정 확인, MFA 및 계정 복구를 위해 사용자에게 이메일 및 SMS 메시지를 보내도록 Amazon Simple Email Service 및 Amazon Simple Notification Service와의 통합을 구성할 수 있습니다. 자세한 내용은 Amazon Cognito 사용자 풀에 대한 이메일 설정 및 Amazon Cognito 사용자 풀의 SMS 메시지 설정 섹션을 참조하세요.

a. 이메일 공급자의 경우 Cognito로 이메일 보내기를 선택하고 Amazon Cognito에서 제공하는 기본 이메일 발신자를 사용합니다. 이메일 용량이 적은 이 설정은 애플리케이션 테스트에 충분합니다. Amazon Simple Email Service (Amazon SES) 를 통해 이메일 주소를 확인하고 Amazon SES로 이메일 보내기를 선택하면 반송할 수 있습니다.

b. SMS의 경우 새 IAM 역할 생성을 선택하고 IAM 역할 이름을 입력합니다. 그러면 Amazon Cognito에 SMS 메시지를 전송할 수 있는 권한을 부여하는 역할이 생성됩니다.

c. 다른 모든 옵션은 기본값으로 유지하고 다음을 선택합니다.

8. 앱 통합에서 사용자 풀의 이름을 지정하고, 호스팅된 UI를 구성하고, 앱 클라이언트를 생성할 수 있습니다. 자세한 정보는 호스팅된 UI가 있는 앱 클라이언트 추가을 참조하세요. 예제 애플리케이션은 호스팅된 UI를 사용하지 않습니다.

a. 사용자 풀 이름에 사용자 풀 이름을 입력합니다.

b. Cognito 호스팅 UI 사용을 선택하지 마세요.

c. 초기 앱 클라이언트에서 앱 유형이 퍼블릭 클라이언트로 설정되어 있는지 확인합니다.

d. 클라이언트 암호에서 클라이언트 암호를 생성하지 않음이 선택되어 있는지 확인합니다.

e. [앱 클라이언트 이름(App client name)]을 입력합니다.

f. 고급 앱 클라이언트 설정을 확장합니다. 인증 흐름 목록에 `ALLOW_USER_PASSWORD_AUTH` 추가합니다.

g. 다른 모든 옵션은 기본값으로 유지하고 다음을 선택합니다.

9. 검토 및 작성 화면에서 선택 사항을 검토하고 필요에 따라 선택 내용을 수정합니다. 사용자 풀 구성에 만족하면 사용자 풀 생성을 선택하여 계속 진행하십시오.

10. 사용자 풀 페이지에서 새 사용자 풀을 선택합니다.

11. 사용자 풀 개요에서 사용자 풀 ID를 기록해 둡니다. 예제 애플리케이션을 만들 때 이 문자열을 제공하게 됩니다.

12. 앱 통합 탭을 선택하고 앱 클라이언트 및 분석 섹션을 찾으십시오. 새 앱 클라이언트를 선택합니다. 클라이언트 ID를 기록해 둡니다.

관련 리소스

- Amazon Cognito 사용자 풀
- 사용자 풀 인증 흐름
- 사용자 풀에 토큰 사용

애플리케이션 생성

예제 Android 앱을 만들려면

1. Android 스튜디오 및 명령줄 도구를 설치합니다.
2. 안드로이드 스튜디오에서 Flutter 플러그인을 설치합니다.
3. 이 예제 앱의 `cognito_flutter_mobile_app` 디렉터리 콘텐츠로 새 Android 스튜디오 프로젝트를 생성합니다.

   - `assets/config.json`수정하고 이전에 만든 사용자 풀 및 앱 클라이언트의 `<< YOUR CLIENT ID>>` ID로 `<<YOUR USER POOL ID>>` 바꿉니다.
4. Flutter를 설치하세요.

   a. PATH 변수에 플러터를 추가하세요.

   b. 다음 명령으로 라이선스를 수락합니다.

   ```
   flutter doctor --android-licenses
   ```

   c. Flutter 환경을 확인하고 누락된 구성 요소를 모두 설치하십시오.

   ```
   flutter doctor
   ```

   - 구성 요소가 누락된 경우 `flutter doctor -v` 실행하여 문제 해결 방법을 알아보세요.

   d. 새 Flutter 프로젝트의 디렉터리로 변경하고 종속 항목을 설치하세요.

   - `flutter pub add amazon_cognito_identity_dart_2`를 실행합니다.

   e. `flutter pub add flutter_secure_storage`를 실행합니다.
5. 가상 안드로이드 기기를 만드세요.

   1. Android 스튜디오 GUI에서 기기 관리자를 사용하여 새 기기를 생성합니다.
   2. CLI에서 를 실행합니다. `flutter emulators --create --name android-device`
6. 가상 안드로이드 디바이스를 실행합니다.

   1. Android 스튜디오 GUI에서 가상 기기 옆에 있는 시작

      아이콘을 선택합니다.
   2. CLI에서 를 실행합니다. `flutter emulators --launch android-device`

7. 가상 디바이스에서 앱을 실행합니다.

   1. Android 스튜디오 GUI에서 배포

      아이콘을 선택합니다.

   2. CLI에서 를 실행합니다. `flutter run`

8. Android 스튜디오에서 실행 중인 가상 기기로 이동합니다.
9. 유효한 이메일 주소로 새 사용자를 등록하세요.
10. 이메일 메시지에서 확인 코드를 검색하세요. 신청서에 확인 코드를 입력합니다.
11. 사용자 이름과 비밀번호로 로그인합니다.

다음 단계

자습서를 따라 예제 애플리케이션을 완성한 후에는 사용자 풀 구현의 범위를 넓힐 수 있습니다. 추가 사용자 풀을 만들거나, 다른 응용 프로그램의 사용자 풀 기능을 사용자 정의하거나, 외부 ID 공급자를 추가할 수 있습니다. Amazon Cognito 사용자 풀을 프로덕션 애플리케이션에 적용하려는 계획을 세울 때 추가 예제와 자습서를 평가할 수 있습니다.

다음은 몇 가지 추가 Amazon Cognito 사용자 풀 기능입니다.

- 기본 제공 로그인 및 가입 웹 페이지 사용자 정의
- 사용자 풀에 MFA 추가
- 사용자 풀에 고급 보안 기능 추가
- Lambda 트리거를 사용하여 사용자 풀 워크플로 사용자 정의
- Amazon Cognito 사용자 풀에서 Amazon Pinpoint 분석 사용

Amazon Cognito 인증 및 권한 부여 모델의 개요는 을 참조하십시오. Amazon Cognito 사용자 풀 및 자격 증명 풀에서 인증이 작동하는 방식

사용자 풀 인증에 성공한 AWS 서비스 후 다른 항목에 액세스하려면 을 참조하십시오로그인 후 자격 증명 풀을 AWS 서비스 사용하여 액세스.

AWS Management Console 및 사용자 풀 SDK를 사용하는 것 외에도 를 사용하여 사용자 풀을 관리할 수 있습니다. AWS Command Line Interface

주제

- 새 사용자 풀 생성
- 호스팅된 UI가 있는 앱 클라이언트 추가
- 사용자 풀에 소셜 로그인 추가(선택 사항)
- 사용자 풀에 SAML 자격 증명 공급자로 로그인 추가(선택 사항)

새 사용자 풀 생성

사용자 풀이 있으면 사용자는 Amazon Cognito를 통해 웹 또는 모바일 앱에 로그인할 수 있습니다.

새 사용자 풀 생성

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 AWS 자격 증명을 입력합니다.
2. 사용자 풀 생성 버튼을 선택합니다. 이 옵션을 표시하려면 왼쪽 탐색 창에서 사용자 풀을 선택해야 할 수도 있습니다.
3. 페이지 오른쪽 상단에서 사용자 풀 생성(Create a user pool)을 선택하여 사용자 풀 생성 마법사를 시작합니다.
4. 로그인 환경 구성에서 이 사용자 풀에 사용할 ID 공급자 (IdPs) 를 선택할 수 있습니다. 자세한 정보는 서드 파티를 통한 사용자 풀 로그인 추가을 참조하세요.

   a. 인증 제공자에서 공급자 유형에는 Cognito 사용자 풀만 선택되어 있는지 확인합니다.
   b. Cognito 사용자 풀 로그인 옵션의 경우 사용자 이름을 선택합니다. 추가 사용자 이름 요구 사항을 선택하지 마세요.
   c. 다른 모든 옵션은 기본값으로 유지하고 다음을 선택합니다.
5. 보안 요구 사항 구성에서 암호 정책, 다단계 인증 (MFA) 요구 사항 및 사용자 계정 복구 옵션을 선택할 수 있습니다. 자세한 정보는 Amazon Cognito 사용자 풀의 보안 기능 사용을 참조하세요.

   a. 암호 정책의 경우 암호 정책 모드가 Cognito 기본값으로 설정되어 있는지 확인합니다.
   b. 멀티 팩터 인증에서 MFA를 적용하려면 옵션 MFA를 선택합니다.
   c. MFA 방법의 경우 인증자 앱 및 SMS 메시지를 선택합니다.
   d. 사용자 계정 복구의 경우 셀프 서비스 계정 복구 활성화가 선택되어 있고 사용자 계정 복구 메시지 전송 방법이 이메일 전용으로 설정되어 있는지 확인합니다.
   e. 다른 모든 옵션은 기본값으로 유지하고 다음을 선택합니다.
6. 가입 환경 구성에서 새 사용자로 가입할 때 새 사용자가 자신의 ID를 확인하는 방법과 사용자 가입 과정에서 필수 또는 선택 사항으로 설정해야 하는 속성을 결정할 수 있습니다. 자세한 정보는 사용자 풀의 사용자 관리을 참조하세요.

a. 셀프 등록 활성화가 선택되었는지 확인합니다. 이 설정을 사용하면 사용자 풀이 열려 인터넷 상의 모든 사용자가 가입할 수 있습니다. 이는 예제 애플리케이션을 위한 것이지만 프로덕션 환경에서는 이 설정을 주의해서 적용하십시오.

b. Cognito 지원 확인 및 확인에서 Cognito가 확인 및 확인을 위해 메시지를 자동으로 보내도록 허용 확인란이 선택되어 있는지 확인합니다.

c. 확인할 속성이 이메일 메시지 전송, 이메일 주소 확인으로 설정되어 있는지 확인합니다.

d. 속성 변경 확인에서 기본 옵션이 선택되어 있는지 확인합니다. 즉, 업데이트 보류 시 원래 속성 값 유지가 선택되고, 업데이트 보류 중 활성 속성 값이 이메일 주소로 설정되어 있는지 확인합니다.

e. 필수 속성에서 이전 선택 항목을 기반으로 하는 필수 속성에 이메일이 표시되는지 확인합니다.

> **Important**
>
> 이 예제 응용 프로그램의 경우 사용자 풀이 phone_number를 필수 속성으로 설정해서는 안 됩니다. phone_number가 필수 속성으로 표시된 경우 이전 선택 사항을 검토하고 업데이트하십시오.
>
> - 사용자 계정 복구 메시지를 위한 선택적 MFA, 이메일 전용 전송 방법
> - 이메일 메시지 전송, 확인할 속성에 대한 이메일 주소 확인

f. 다른 모든 옵션은 기본값으로 유지하고 다음을 선택합니다.

7. 메시지 전송 구성에서 가입, 계정 확인, MFA 및 계정 복구를 위해 사용자에게 이메일 및 SMS 메시지를 보내도록 Amazon Simple Email Service 및 Amazon Simple Notification Service와의 통합을 구성할 수 있습니다. 자세한 내용은 Amazon Cognito 사용자 풀에 대한 이메일 설정 및 Amazon Cognito 사용자 풀의 SMS 메시지 설정 섹션을 참조하세요.

   a. 이메일 공급자의 경우 Cognito로 이메일 보내기를 선택하고 Amazon Cognito에서 제공하는 기본 이메일 발신자를 사용합니다. 이메일 용량이 적은 이 설정은 애플리케이션 테스트에 충분합니다. Amazon Simple Email Service (Amazon SES) 를 통해 이메일 주소를 확인하고 Amazon SES로 이메일 보내기를 선택하면 반송할 수 있습니다.

   b. SMS의 경우 새 IAM 역할 생성을 선택하고 IAM 역할 이름을 입력합니다. 그러면 Amazon Cognito에 SMS 메시지를 전송할 수 있는 권한을 부여하는 역할이 생성됩니다.

   c. 다른 모든 옵션은 기본값으로 유지하고 다음을 선택합니다.

8. 앱 통합에서 사용자 풀의 이름을 지정하고, 호스팅된 UI를 구성하고, 앱 클라이언트를 생성할 수 있습니다. 자세한 정보는 호스팅된 UI가 있는 앱 클라이언트 추가을 참조하세요. 예제 애플리케이션은 호스팅된 UI를 사용하지 않습니다.

   a. 사용자 풀 이름에 사용자 풀 이름을 입력합니다.

   b. Cognito 호스팅 UI 사용을 선택하지 마세요.

   c. 초기 앱 클라이언트에서 앱 유형이 퍼블릭 클라이언트로 설정되어 있는지 확인합니다.

   d. 클라이언트 암호에서 클라이언트 암호를 생성하지 않음이 선택되어 있는지 확인합니다.

   e. [앱 클라이언트 이름(App client name)]을 입력합니다.

   f. 고급 앱 클라이언트 설정을 확장합니다. 인증 흐름 목록에 `ALLOW_USER_PASSWORD_AUTH` 추가합니다.

   g. 다른 모든 옵션은 기본값으로 유지하고 다음을 선택합니다.

9. 검토 및 작성 화면에서 선택 사항을 검토하고 필요에 따라 선택 내용을 수정합니다. 사용자 풀 구성에 만족하면 사용자 풀 생성을 선택하여 계속 진행하십시오.

10. 사용자 풀 페이지에서 새 사용자 풀을 선택합니다.

11. 사용자 풀 개요에서 사용자 풀 ID를 기록해 둡니다. 예제 애플리케이션을 만들 때 이 문자열을 제공하게 됩니다.

12. 앱 통합 탭을 선택하고 앱 클라이언트 및 분석 섹션을 찾으십시오. 새 앱 클라이언트를 선택합니다. 클라이언트 ID를 기록해 둡니다.

사용자 풀을 생성하려면

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 AWS 자격 증명을 입력합니다.
2. 사용자 풀(User Pools)을 선택합니다.
3. 페이지 오른쪽 상단에서 [사용자 풀 생성(Create a user pool)]을 선택하여 사용자 풀 생성 마법사를 시작합니다.
4. [로그인 환경 구성(Configure sign-in experience)]에서 이 사용자 풀에 사용할 페더레이션 공급자를 선택합니다. 자세한 정보는 서드 파티를 통한 사용자 풀 로그인 추가을 참조하세요.
5. 보안 요구 사항 구성(Configure security requirements)에서 암호 정책, 멀티 팩터 인증(MFA) 요구 사항, 사용자 계정 복구 옵션을 선택합니다. 자세한 정보는 Amazon Cognito 사용자 풀의 보안 기능 사용을 참조하세요.
6. [가입 환경 구성(Configure sign-up experience)]에서 새 사용자가 가입할 때 자격 증명을 확인하는 방법과 사용자 가입 흐름 중에 필수 또는 선택 사항이어야 하는 속성을 결정합니다. 자세한 정보는 사용자 풀의 사용자 관리을 참조하세요.

> **Important**
>
> 사용자 풀에서 사용자 가입을 활성화하면 인터넷에 있는 누구나 계정에 가입하고 앱에 로그인할 수 있습니다. 퍼블릭 가입이 가능하도록 앱을 공개하려는 경우가 아니면 사용자 풀에서 자체 등록을 활성화하지 마세요. 이 설정을 변경하려면 사용자 풀 콘솔의 가입 경험 탭에서 셀프 서비스 등록을 업데이트하거나 또는 API AllowAdminCreateUserOnly요청의 값을 업데이트하십시오. CreateUserPool UpdateUserPool
> 사용자 풀에서 설정할 수 있는 보안 기능에 대한 자세한 내용은 Amazon Cognito 사용자 풀의 보안 기능 사용 섹션을 참조하세요.

7. [메시지 전달 구성(Configure message delivery)]에서 가입, 계정 확인, MFA, 계정 복구를 위해 사용자에게 이메일과 SMS 메시지를 전송하도록 Amazon Simple Email Service 및 Amazon Simple Notification Service와의 통합을 구성합니다. 자세한 내용은 Amazon Cognito 사용자 풀에 대한 이메일 설정 및 Amazon Cognito 사용자 풀의 SMS 메시지 설정 섹션을 참조하세요.
8. [앱 통합(Integrate your app)]에서 사용자 풀의 이름을 지정하고 호스트된 UI를 구성한 다음, 앱 클라이언트를 생성합니다. 자세한 정보는 호스팅된 UI가 있는 앱 클라이언트 추가을 참조하세요.
9. 검토 및 생성 화면에서 선택 사항을 검토하고 필요에 따라 선택 사항을 수정하십시오. 사용자 풀 구성에 만족하면 사용자 풀 생성을 선택하여 계속 진행하십시오.

관련 리소스

사용자 풀에 대한 자세한 내용은 Amazon Cognito 사용자 풀 섹션을 참조하세요.

참조: 사용자 풀 인증 흐름 및 사용자 풀에 토큰 사용

호스팅된 UI가 있는 앱 클라이언트 추가

사용자 풀을 생성한 후 호스팅된 UI의 빌트인 웹페이지를 불러오는 애플리케이션용 앱 클라이언트를 생성할 수 있습니다. 호스팅된 UI에서 사용자는 다음을 수행할 수 있습니다.

- 사용자 프로필에 가입하세요.
- 타사 ID 공급자로 로그인합니다.
- 다단계 인증을 사용하거나 사용하지 않고 로그인할 수 있습니다.
- 비밀번호를 재설정하세요.

호스팅된 UI 로그인을 위한 앱 클라이언트를 만들려면

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 AWS 자격 증명을 입력합니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다. 새 사용자 풀을 생성하는 경우 마법사 중에 앱 클라이언트를 설정하고 호스트된 UI를 구성하라는 메시지가 표시됩니다.
4. 사용자 풀의 [앱 통합(App integration)] 탭을 선택합니다.
5. 도메인 옆의 작업을 선택한 다음, 사용자 지정 도메인 생성 또는 Amazon Cognito 도메인 생성을 선택합니다. 사용자 풀 도메인을 이미 구성한 경우 새 사용자 지정 도메인을 생성하기 전에 Amazon Cognito 도메인 삭제 또는 사용자 지정 도메인 삭제를 선택합니다.
6. Amazon Cognito 도메인에서 사용할 사용 가능한 도메인 접두사를 입력합니다. 사용자 지정 도메인을 설정하는 방법에 대한 자세한 내용은 호스팅 UI에 고유한 도메인 사용을 참조하세요.
7. 생성을 선택합니다.
8. 동일한 사용자 풀에 대한 [앱 통합(App integration)] 탭으로 다시 이동한 다음, [앱 클라이언트(App clients)]를 찾습니다. [앱 클라이언트 생성(Create an app client)]을 선택합니다.
9. [애플리케이션 유형(Application type)]을 선택합니다. 선택 내용에 따라 몇 가지 권장 설정이 제공됩니다. 호스트된 UI를 사용하는 앱은 퍼블릭 클라이언트입니다.
10. [앱 클라이언트 이름(App client name)]을 입력합니다.
11. 이 연습에서는 [클라이언트 암호 생성 안 함(Don't generate client secret)]을 선택합니다. 클라이언트 암호는 중앙 집중식 애플리케이션에서 사용자를 인증하는 기밀 앱에서 사용됩니다. 이 연습에서는 사용자에게 호스트된 UI 로그인 페이지를 제공하며 클라이언트 암호가 필요하지 않습니다.
12. 앱에 허용할 인증 흐름을 선택합니다. `USER_SRP_AUTH`가 선택되었는지 확인합니다.
13. 필요에 따라 [토큰 만료(token expiration)], [고급 보안 구성(Advanced security configuration)], [속성 읽기 및 쓰기 권한(Attribute read and write permissions)]을 사용자 정의합니다. 자세한 내용은 앱 클라이언트 설정 구성을 참조하세요.
14. 콜백 URL 추가(Add a callback URL)를 통해 앱 클라이언트의 콜백 URL을 추가합니다. 호스트된 UI 인증 후에 여기로 이동됩니다. 앱에서 로그아웃을 구현할 수 있을 때까지는 허용된 로그아웃 URL을 추가할 필요가 없습니다.

    iOS 또는 Android 앱의 경우에는 `myapp://`와 같은 콜백 URL을 사용할 수 있습니다.
15. 앱 클라이언트의 [자격 증명 공급자(Identity providers)]를 선택합니다. 최소한 공급자로 Amazon Cognito 사용자 풀을 사용하도록 설정합니다.

> **Note**
>
> OpenID Connect (OIDCIdPs) 또는 IdPs SAML을 통해서뿐만 아니라 페이스북, 아마존, 구글, 애플과 같은 외부 ID 공급자 () 를 사용하여 로그인하려면 먼저 제3자를 통한 사용자 풀 로그인 추가에 나와 있는 것처럼 구성하십시오. 그런 다음 앱 클라이언트 설정 페이지로 돌아가서 활성화하십시오.

16. [OAuth 2.0 권한 부여 유형(OAuth 2.0 Grant Types)]을 선택합니다. Authorization code grant(인증 코드 권한 부여)를 선택하여 사용자 풀 토큰으로 교환되는 인증 코드를 반환합니다. 토큰은 최종 사용자에게 직접 노출되지 않기 때문에 침해될 가능성이 낮습니다. 하지만 사용자 풀 토큰에 대한 인증 코드를 교환하려면 백엔드에 사용자 지정 애플리케이션이 필요합니다. 보안상의 이유로 모바일 앱에서는 PKCE(Proof Key for Code Exchange)와 함께 인증 코드 권한 부여 흐름을 사용하는 것이 좋습니다.

    [암시적 허용(Implicit grant)]를 선택하여 Amazon Cognito에서 사용자 풀 JSON 웹 토큰(JWT)이 반환되도록 합니다. 토큰에 대한 인증 코드를 교환할 수 있는 백엔드가 없을 때 이 흐름을 사용할 수 있습니다. 또한 토큰 디버깅에도 유용합니다.

    > **Note**
    >
    > Authorization code grant(인증 코드 권한 부여) 및 Implicit code grant(암시적 코드 부여)를 모두 활성화하고 각각을 필요한 경우 사용할 수 있습니다.
    > 사용자를 대신해서가 아니라 자체적으로 액세스 토큰을 요청해야 하는 경우에만 Client credentials(클라이언트 자격 증명)를 선택합니다.

17. 구체적으로 하나를 제외하려는 경우가 아니면 모든 OpenID Connect 범위(OpenID Connect scopes)를 선택합니다.
18. 구성한 사용자 지정 범위를 선택합니다. 사용자 정의 범위는 일반적으로 기밀 클라이언트에서 사용됩니다.
19. 생성을 선택합니다.

로그인 페이지를 보려면

앱 클라이언트 페이지에서 호스팅된 UI 보기를 선택하여 앱 클라이언트 ID, 범위, 권한 부여 및 콜백 URL 매개변수로 미리 채워진 로그인 페이지의 새 브라우저 탭을 엽니다.

다음 URL을 사용하여 호스트된 UI 로그인 웹 페이지를 수동으로 볼 수 있습니다. response_type을 기록합니다. 이 경우 response_type=code는 인증 코드 권한 부여입니다.

```
https://your_domain/login?
response_type=code&client_id=your_app_client_id&redirect_uri=your_callback_url
```

다음 URL을 사용하여 호스팅 UI 로그인 웹 페이지를 볼 수 있습니다. 여기서 response_type=token은 묵시적 코드 부여입니다. 로그인 성공 이후 Amazon Cognito는 사용자 풀 토큰을 웹 브라우저의 주소 표시줄로 반환합니다.

```
https://your_domain/login?
response_type=token&client_id=your_app_client_id&redirect_uri=your_callback_url
```

#idtoken= 파라미터 응답 이후 JSON 웹 토큰(JWT) 자격 증명을 찾을 수 있습니다.

다음 URL은 암시적 권한 부여 요청의 샘플 응답입니다. 사용자의 자격 증명 토큰 문자열이 훨씬 더 길어집니다.

```
https://www.example.com/
#id_token=123456789tokens123456789&expires_in=3600&token_type=Bearer
```

Amazon Cognito 사용자 풀 토큰은 RS256 알고리즘을 사용하여 서명됩니다. 를 사용하여 사용자 풀 토큰을 디코딩하고 확인할 수 있습니다. AWS Lambda자세한 내용은 웹 사이트에서 Amazon Cognito JWT 토큰 디코딩 및 확인을 참조하십시오. AWS GitHub

도메인 이름(Domain name) 페이지에 도메인이 표시됩니다. 일반 설정 페이지에 앱 클라이언트 ID와 콜백 URL이 표시됩니다. 콘솔에서 변경한 내용이 즉시 표시되지 않으면 몇 분 정도 기다린 다음 브라우저를 새로 고치십시오.

사용자 풀에 소셜 로그인 추가(선택 사항)

앱 사용자가 Facebook, Google, Amazon 및 Apple과 같은 소셜 자격 증명 공급자(IdP)를 통해 로그인하도록 할 수 있습니다. 사용자가 직접 또는 타사를 통해 로그인하는지 여부와 무관하게 모든 사용자는 사용자 풀에 프로필을 보유합니다. 소셜 로그인 자격 증명 공급자를 통한 로그인 추가를 원치 않는 경우 이 단계를 건너뛰세요.

소셜 IdP에 등록

Amazon Cognito에서 소셜 IdP를 생성하려면 소셜 IdP에 애플리케이션을 등록하여 클라이언트 ID와 클라이언트 암호를 받아야 합니다.

Facebook으로 앱을 등록하려면

1. Facebook에서 개발자 계정을 생성합니다.
2. Facebook 자격 증명으로 로그인합니다.
3. 내 앱(My Apps) 메뉴에서 Create New App(새 앱 생성)을 선택합니다.

   기존 Facebook 앱이 없는 경우 다른 옵션이 표시됩니다. 앱 생성을 선택합니다.
4. 앱 생성 페이지에서 앱의 사용 사례를 선택한 후 다음을 선택합니다.
5. Facebook 앱의 이름을 입력하고 앱 생성을 선택합니다.
6. 왼쪽 탐색 모음에서 앱 설정을 선택한 다음 기본 사항을 선택합니다.
7. 앱 ID(App ID)와 앱 보안(App Secret)을 메모합니다. 다음 섹션에서 이 둘을 사용합니다.
8. 페이지 하단에서 + 플랫폼 추가를 선택합니다.
9. 플랫폼 선택 화면에서 플랫폼을 선택하고 다음을 선택합니다.
10. 변경 사항 저장(Save changes)을 선택합니다.
11. [앱 도메인(App Domains)]에서 사용자 풀 도메인을 입력합니다.

    ```
    https://your_user_pool_domain
    ```
12. 변경 사항 저장(Save changes)을 선택합니다.
13. 탐색 표시줄에서 제품을 선택한 다음 Facebook 로그인에서 구성을 선택합니다.
14. Facebook 로그인 구성 메뉴에서 설정을 선택합니다.

    유효한 OAuth 리디렉션 URI(Valid OAuth Redirect URIs)에 리디렉션 URL을 입력합니다. 리디렉션 URL은 사용자 풀 도메인과 `/oauth2/idpresponse` 엔드포인트로 구성됩니다.

    ```
    https://your_user_pool_domain/oauth2/idpresponse
    ```
15. 변경 사항 저장(Save changes)을 선택합니다.

Amazon으로 앱을 등록하려면

1. Amazon에서 개발자 계정을 생성합니다.

2. Amazon 자격 증명으로 로그인합니다.
3. Amazon 보안 프로파일을 생성하여 Amazon 클라이언트 ID와 클라이언트 암호를 받아야 합니다.

   페이지 상단의 탐색 표시줄에서 앱 및 서비스를 선택한 다음 Login with Amazon을 선택합니다.
4. 보안 프로필 생성(Create a Security Profile)을 선택합니다.
5. 보안 프로파일 이름(Security Profile Name), 보안 프로파일 설명(Security Profile Description), 개인 정보 보호 정책 동의 URL(Consent Privacy Notice URL)을 입력합니다.
6. 저장(Save)을 선택합니다.
7. 클라이언트 ID(Client ID) 및 클라이언트 암호(Client Secret)를 선택하여 클라이언트 ID 및 암호를 표시합니다. 다음 섹션에서 이 둘을 사용합니다.
8. 톱니 모양 아이콘을 마우스로 가리키고 [웹 설정(Web Settings)]을 선택한 다음, [편집(Edit)]을 선택합니다.
9. [허용된 원본(Allowed Origins)]에 사용자 풀 도메인을 입력합니다.

   ```
   https://<your-user-pool-domain>
   ```
10. [허용된 반환 URL(Allowed Return URLs)]에 `/oauth2/idpresponse` 엔드포인트가 있는 사용자 풀 도메인을 입력합니다.

    ```
    https://<your-user-pool-domain>/oauth2/idpresponse
    ```
11. 저장(Save)을 선택합니다.

Google로 앱을 등록하려면

Google Cloud Platform의 OAuth 2.0에 대한 자세한 내용은 개발자용 Google Workspace 설명서의 인증 및 권한 부여 알아보기를 참조하세요.

1. Google에서 개발자 계정을 생성합니다.
2. Google Cloud Platform 콘솔에 로그인합니다.
3. 상단 탐색 모음에서 프로젝트 선택(Select a project)을 선택합니다. Google 플랫폼에 프로젝트가 이미 있는 경우 이 메뉴에는 대신 기본 프로젝트가 표시됩니다.
4. 새 프로젝트(NEW PROJECT)를 선택합니다.
5. 프로젝트의 이름을 입력한 다음 생성(CREATE)을 선택합니다.
6. 왼쪽 내비게이션 바에서 API 및 서비스를 선택한 다음 Oauth 동의 화면을 선택합니다.

7. 앱 정보, 앱 도메인, 승인된 도메인, 개발자 연락처 정보를 입력합니다. 인증된 도메인에는 사용자 지정 도메인의 루트가 `amazoncognito.com` 포함되어야 합니다. 예를 들면 `example.com`입니다. 저장 후 계속(SAVE AND CONTINUE)을 선택합니다.
8. 1. 범위에서 범위 추가 또는 제거를 선택한 다음 최소한 다음과 같은 OAuth 범위를 선택합니다.
    1. `.../auth/userinfo.email`
    2. `.../auth/userinfo.profile`
    3. openid
9. 사용자 테스트(Test users)에서 사용자 추가(Add users)를 선택합니다. 이메일 주소와 기타 승인된 테스트 사용자를 입력한 다음 저장 후 계속을 선택합니다.
10. 왼쪽 탐색 막대를 다시 확장하고 API 및 서비스를 선택한 다음 자격 증명을 선택합니다.
11. 자격 증명 생성을 선택한 다음 OAuth 클라이언트 ID를 선택합니다.
12. 애플리케이션 유형(Application type)을 선택하고 클라이언트 이름을 지정합니다.
13. 승인된 JavaScript 출처에서 URI 추가를 선택합니다. 사용자 풀 도메인을 입력합니다.

    ```
    https://<your-user-pool-domain>
    ```
14. 권한 있는 리디렉션 URI(Authorized redirect URIs)에서 URI 추가(ADD URI)를 선택합니다. 사용자 풀 도메인의 `/oauth2/idpresponse` 엔드포인트에 대한 경로를 입력합니다.

    ```
    https://<your-user-pool-domain>/oauth2/idpresponse
    ```
15. 생성(CREATE)을 선택합니다.
16. 내 클라이언트 ID와 내 클라이언트 암호에서 Google이 표시하는 값을 안전하게 저장합니다. Google IdP를 추가할 때 이러한 값을 Amazon Cognito에 입력합니다.

Apple으로 앱을 등록하려면

Apple로 로그인 설정에 대한 자세한 내용은 Apple 개발자 설명서의 Apple로 로그인용 환경 구성을 참조하세요.

1. Apple에서 개발자 계정을 생성합니다.
2. Apple 자격 증명으로 로그인합니다.
3. 왼쪽 탐색 모음에서 인증서, 식별자 및 프로필(Certificates, Identifiers & Profiles)을 선택합니다.
4. 왼쪽 탐색 모음에서 식별자(Identifiers)를 선택합니다.
5. 식별자(Identifiers) 페이지에서 + 아이콘을 선택합니다.

6. 새 식별자 등록(Register a New Identifier) 페이지에서 앱 ID(App IDs)를 선택한 다음 계속(Continue)을 선택합니다.
7. 유형 선택 페이지에서 앱을 선택한 다음 계속을 선택합니다.
8. 앱 ID 등록(Register an App ID) 페이지에서 다음을 수행합니다.

    1. 설명(Description)에 설명을 입력합니다.
    2. 앱 ID 접두사(App ID Prefix)에서 번들 ID(Bundle ID)를 입력합니다. [앱 ID 접두사(App ID Prefix)] 아래의 값을 적어 둡니다. 2단계: 사용자 풀에 소셜 IdP 추가에서 자격 증명 공급자로 Apple을 선택한 후 이 값을 사용합니다.
    3. 기능(Capabilities)에서 Apple로 로그인(Sign In with Apple)을 선택한 다음 편집(Edit)을 선택합니다.
    4. Apple로 로그인: 앱 ID 구성(Sign in with Apple: App ID Configuration) 페이지에서 앱을 기본 앱으로 설정하거나 다른 앱 ID와 함께 그룹화하도록 선택한 다음, 저장(Save)을 선택합니다.
    5. 계속을 선택합니다.
9. 앱 ID 확인(Confirm your App ID) 페이지에서 등록(Register)을 선택합니다.
10. 식별자(Identifiers) 페이지에서 + 아이콘을 선택합니다.
11. 새 식별자 등록(Register a New Identifier) 페이지에서 서비스 ID(Services IDs)를 선택한 다음 계속(Continue)을 선택합니다.
12. 서비스 ID 등록(Register a Services ID) 페이지에서 다음을 수행합니다.

    1. 설명(Description)에 설명을 입력합니다.
    2. 식별자(Identifier) 아래에 식별자를 입력합니다. 에서 Apple을 ID 공급자로 선택한 후 이 값이 필요하므로 이 서비스 ID를 기록해 두십시오2단계: 사용자 풀에 소셜 IdP 추가.
    3. 계속, 등록을 차례로 선택합니다.
13. 식별자 페이지에서 방금 생성한 서비스 ID를 선택합니다.

    1. Apple로 로그인(Sign In with Apple)을 선택한 다음 구성(Configure)을 선택합니다.
    2. 웹 인증 구성(Web Authentication Configuration) 페이지에서 앞서 생성한 앱 ID를 기본 앱 ID(Primary App ID)로 선택합니다.
    3. 웹 사이트 URL(Website URLs)에서 + 아이콘을 선택합니다.
    4. 도메인 및 하위 도메인(Domains and subdomains)에서 `https://` 접두사를 사용하지 않고 사용자 풀 도메인을 입력합니다.

        ```
        <your-user-pool-domain>
        ```

5. 반환 URL(Return URLs)에 사용자 풀 도메인의 `/oauth2/idpresponse` 엔드포인트에 대한 경로를 입력합니다.

   ```
   https://<your-user-pool-domain>/oauth2/idpresponse
   ```

6. [다음] 을 선택한 다음 [완료] 를 선택합니다. 도메인을 확인할 필요는 없습니다.
7. 계속(Continue)과 저장(Save)을 차례로 선택합니다.

14. 왼쪽 탐색 모음에서 키(Keys)를 선택합니다.
15. 키(Keys) 페이지에 + 아이콘을 선택합니다.
16. 새 키 등록(Register a New Key) 페이지에서 다음을 수행합니다.

    1. [키 이름(Key Name)] 아래에 키 이름을 입력합니다.
    2. Apple로 로그인(Sign In with Apple)을 선택한 다음 구성(Configure)을 선택합니다.
    3. 키 구성 페이지에서 이전에 기본 앱 ID로 만든 앱 ID를 선택합니다. 저장을 선택합니다.
    4. 계속(Continue), 등록(Register)을 차례로 선택합니다.

17. 키 다운로드 페이지에서 다운로드를 선택하여 개인 키를 다운로드하고 표시된 키 ID를 메모한 다음 완료를 선택합니다. 이 프라이빗 키와 이 페이지에 표시되는 키 ID(Key ID) 값은 2단계: 사용자 풀에 소셜 IdP 추가에서 자격 증명 공급자로 Apple을 선택한 후에 필요합니다.

사용자 풀에 소셜 IdP 추가

이 섹션에서는 이전 섹션의 클라이언트 ID와 클라이언트 암호를 사용하여 사용자 풀에 소셜 IdP를 구성합니다.

다음을 사용하여 사용자 풀 소셜 ID 공급자를 구성하려면 AWS Management Console

1. Amazon Cognito 콘솔로 이동합니다. AWS 자격 증명을 입력하라는 메시지가 표시될 수 있습니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. [로그인 환경(Sign-in experience)] 탭을 선택합니다. [페더레이션 로그인(Federated sign-in)]을 찾아서 [자격 증명 공급자 추가(Add an identity provider)]를 선택합니다.
5. Facebook, Google, Login with Amazon 또는 Sign in with Apple과 같은 소셜 자격 증명 공급자를 선택합니다.
6. 선택한 소셜 ID 공급자에 따라 다음 단계 중에서 선택합니다.

- Google 및 Login with Amazon — 이전 섹션에서 생성한 앱 클라이언트 ID와 앱 클라이언트 암호를 입력합니다.
- Facebook — 이전 섹션에서 생성한 앱 클라이언트 ID와 앱 클라이언트 암호를 입력한 다음 API 버전 (예: 버전 2.12) 을 선택합니다. 가능한 최신 버전을 선택하는 것이 좋습니다. 각 Facebook API에는 수명 주기와 사용 중단 날짜가 있습니다. Facebook 범위와 속성은 API 버전마다 다를 수 있습니다. Facebook에서 소셜 ID 로그인을 테스트하여 페더레이션이 의도한 대로 작동하는지 확인하는 것이 좋습니다.
- Apple로 로그인 — 이전 섹션에서 생성한 서비스 ID, 팀 ID, 키 ID 및 개인 키를 입력합니다.

7. 사용하려는 승인된 범위의 이름을 입력합니다. 범위는 앱에서 액세스하고자 하는 사용자 속성(예: `name` 및 `email`)을 정의합니다. Facebook의 경우 쉼표로 구분해야 합니다. Google 및 Login with Amazon의 경우 공백으로 구분해야 합니다. Apple로 로그인의 경우, 액세스하려는 범위의 확인란을 선택합니다.

소셜 자격 증명 공급자	예제 범위
Facebook	`public_profile, email`
Google	`profile email openid`
Login with Amazon	`profile postal_code`
Apple로 로그인	`email name`

앱에 이러한 속성을 제공하는 데 동의하라는 메시지가 앱 사용자에게 표시됩니다. 소셜 공급자 범위에 대한 자세한 내용은 Google, Facebook, Login with Amazon 또는 Sign in with Apple의 설명서를 참조하세요.

Sign in with Apple의 경우 다음은 범위가 반환되지 않을 수 있는 사용자 시나리오입니다.

- 최종 사용자가 Apple의 로그인 페이지를 나간 후 오류가 발생합니다 (Amazon Cognito의 내부 오류 또는 개발자가 작성한 모든 오류로 인해 발생할 수 있음).
- 서비스 ID 식별자는 사용자 풀 및/또는 기타 인증 서비스에서 사용됩니다.
- 사용자가 로그인한 후 개발자가 추가 범위를 추가합니다. 사용자는 인증할 때와 토큰을 새로 고칠 때만 새 정보를 검색합니다.

- 개발자가 사용자를 삭제하면 사용자는 Apple ID 프로필에서 앱을 제거하지 않고 다시 로그인합니다.

8. 자격 증명 공급자의 속성을 사용자 풀에 매핑합니다. 자세한 정보는 매핑에 대해 알아야 할 사항을 참조하세요.
9. 생성을 선택합니다.
10. [앱 클라이언트 통합(App client integration)] 탭의 목록에서 [앱 클라이언트(App clients)] 중 하나를 선택한 다음, [호스트된 UI 설정 편집(Edit hosted UI settings)]을 선택합니다. [자격 증명 공급자(Identity providers)]에서 앱 클라이언트에 새 소셜 자격 증명 공급자를 추가합니다.
11. 변경 사항 저장(Save changes)을 선택합니다.

소셜 IdP 구성 테스트

이전 두 섹션의 요소를 사용하여 로그인 URL을 생성할 수 있습니다. 이것을 사용하여 소셜 IdP 구성을 테스트합니다.

```
https://mydomain.us-east-1.amazoncognito.com/login?
response_type=code&client_id=1example23456789&redirect_uri=https://www.example.com
```

사용자 풀 도메인 이름(Domain name) 콘솔 페이지에서 사용자의 도메인을 찾을 수 있습니다. client_id는 앱 클라이언트 설정(App client settings) 페이지에 있습니다. redirect_uri 파라미터용 콜백 URL을 사용합니다. 이것은 인증 성공 이후에 사용자가 리디렉션되는 페이지의 URL입니다.

> **Note**
>
> Amazon Cognito는 5분 이내에 완료되지 않는 인증 요청을 취소하고 사용자를 호스팅 UI로 리디렉션합니다. 페이지에 `Something went wrong` 오류 메시지가 표시됩니다.

사용자 풀에 SAML 자격 증명 공급자로 로그인 추가(선택 사항)

앱 사용자가 SAML 자격 증명 공급자(IdP)를 통해 로그인하도록 할 수 있습니다. 사용자가 직접 또는 타사를 통해 로그인하는지 여부와 무관하게 모든 사용자는 사용자 풀에 프로필을 보유합니다. SAML 자격 증명 공급자를 통한 로그인 추가를 원치 않는 경우 이 단계를 건너뛰세요.

자세한 정보는 사용자 풀과 함께 SAML ID 공급자 사용을 참조하세요.

SAML ID 공급자를 업데이트하고 사용자 풀을 구성해야 합니다. 사용자 풀을 SAML 2.0 ID 공급자의 신뢰 당사자 또는 애플리케이션으로 추가하는 방법에 대한 자세한 내용은 SAML ID 공급자의 설명서를 참조하십시오.

또한 SAML ID 공급자에게 어설션 소비자 서비스 (ACS) 엔드포인트를 제공해야 합니다. SAML ID 제공업체의 SAML 2.0 POST 바인딩에 대해 사용자 풀 도메인에서 다음 엔드포인트 구성합니다. 사용자 풀 도메인에 대한 자세한 내용은 을 참조하십시오. 사용자 풀 도메인 구성

```
https://Your user pool domain/saml2/idpresponse
With an Amazon Cognito domain:
https://<yourDomainPrefix>.auth.<region>.amazoncognito.com/saml2/idpresponse
With a custom domain:
https://Your custom domain/saml2/idpresponse
```

Amazon Cognito 콘솔의 도메인 이름 탭에서 사용자 풀의 도메인 접두사와 지역 값을 찾을 수 있습니다.

일부 SAML 자격 증명 공급자의 경우 고객 URI 또는 SP 엔티티 ID라고도 하는 서비스 공급자 (SP) `urn` 를 다음과 같은 형식으로 제공해야 합니다.

```
urn:amazon:cognito:sp:<yourUserPoolID>
```

사용자 풀 ID는 Amazon Cognito 콘솔의 일반 설정(General settings) 탭에서 확인할 수 있습니다.

사용자 풀에 필요한 속성에 대한 속성 값을 제공하려면 SAML 자격 증명 공급자도 구성해야 합니다. 일반적으로 `email`은 사용자 풀에 대해 필요한 속성입니다. 이러한 경우 SAML 자격 증명 공급자는 SAML 어설션에 `email` 값(클레임)을 제공해야 합니다.

Amazon Cognito 사용자 풀은 사후 바인딩 엔드포인트와의 SAML 2.0 페더레이션을 지원합니다. 이렇게 하면 사용자 풀이 사용자 에이전트를 통해 ID 공급자로부터 SAML 응답을 직접 수신하므로 앱에서 SAML 어설션 응답을 검색하거나 파싱할 필요가 없습니다.

사용자 풀에 SAML 2.0 자격 증명 공급자를 구성하려면

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 자격 증명을 입력합니다. AWS
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. [로그인 환경(Sign-in experience)] 탭을 선택합니다. [페더레이션 로그인(Federated sign-in)]을 찾아서 [자격 증명 공급자 추가(Add an identity provider)]를 선택합니다.

5. [SAML] 소셜 자격 증명 공급자를 선택합니다.
6. [식별자(Identifiers)]를 쉼표로 구분하여 입력합니다. 식별자는 Amazon Cognito에 사용자가 로그인할 때 입력하는 이메일 주소를 확인해야 한다고 알려줍니다. 그런 다음 도메인에 해당하는 공급자로 사용자를 안내합니다.
7. 사용자가 로그아웃할 때 Amazon Cognito에서 서명된 로그아웃 요청을 공급자에게 보내도록 하려는 경우 [로그아웃 흐름 추가(Add sign-out flow)]를 선택합니다. 호스트된 UI를 구성할 때 생성되는 `https://<your Amazon Cognito domain>/saml2/logout` 엔드포인트에 로그아웃 응답을 보내도록 SAML 2.0 자격 증명 공급자를 구성해야 합니다. `saml2/logout`엔드포인트는 POST 바인딩을 사용합니다.

   > **Note**
   >
   > 이 옵션을 선택하고 SAML ID 공급자가 서명된 로그아웃 요청을 예상하는 경우 Amazon Cognito에서 제공하는 서명 인증서를 SAML IdP와 함께 구성해야 합니다.
   > SAML IdP는 서명된 로그아웃 요청을 처리하고 Amazon Cognito 세션에서 사용자를 로그아웃시킵니다.

8. 메타데이터 문서 소스(Metadata document source)를 선택합니다. 자격 증명 공급자가 퍼블릭 URL에서 SAML 메타데이터를 제공하는 경우 메타데이터 문서 URL(Metadata document URL)을 선택하고 해당 퍼블릭 URL을 입력할 수 있습니다. 그렇지 않은 경우 메타데이터 문서 업로드(Upload metadata document)를 선택한 다음, 이전에 공급자로부터 다운로드한 메타데이터 파일을 선택합니다.

   > **Note**
   >
   > 제공자에 퍼블릭 엔드포인트가 있는 경우 파일을 업로드하는 대신 메타데이터 문서 URL을 입력하는 것이 좋습니다. 이렇게 하면 Amazon Cognito가 메타데이터를 자동으로 새로 고칠 수 있습니다. 일반적으로 메타데이터 새로 고침은 6시간마다 또는 메타데이터가 만료되기 전 중 더 빠른 시간에 발생합니다.

9. [SAML 공급자와 앱 간에 속성 매핑(Map attributes between your SAML provider and your app)]을 선택하여 SAML 공급자 속성을 사용자 풀의 사용자 프로파일에 매핑합니다. 속성 맵에 사용자 풀 필수 속성을 포함합니다.

   예를 들어 [사용자 풀 속성(User pool attribute)] `email`을 선택한 경우 자격 증명 공급자의 SAML 어설션에 표시된 대로 SAML 속성 이름을 입력합니다. 자격 증명 공급자가 참조용으로 샘플

SAML 어설션을 제공할 수도 있습니다. email과 같은 간단한 이름을 사용하는 자격 증명 공급자도 있고, 다음 예제와 같이 URL 포맷의 속성 이름을 사용하는 자격 증명 공급자도 있습니다.

```
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
```

10. 생성(Create)을 선택합니다.

Amazon Cognito 자격 증명 풀 시작하기

Amazon Cognito 자격 증명 풀을 통해 사용자 고유의 자격 증명을 생성하고 사용자의 권한을 할당할 수 있습니다. 자격 증명 풀은 다음을 포함할 수 있습니다.

- Amazon Cognito 사용자 풀의 사용자
- Facebook, Google, Apple 또는 SAML 자격 증명 공급자와 같은 외부 자격 증명 공급자로 인증하는 사용자
- 기존의 고유한 인증 프로세스를 통해 인증된 사용자

AWS 자격 증명 풀을 사용하면 Amazon API Gateway를 통해 다른 리소스에 직접 AWS 서비스 액세스하거나 리소스에 액세스하기 위해 정의한 권한이 포함된 임시 자격 증명을 얻을 수 있습니다.

주제

Amazon Cognito에서 자격 증명 풀 생성

Amazon Cognito 콘솔을 통해 자격 증명 풀을 만들거나 AWS Command Line Interface 또는 Amazon Cognito API를 사용할 수 있습니다.

콘솔에서 새 자격 증명 풀을 만들려면

1. Amazon Cognito 콘솔에 로그인하고 자격 증명 풀을 선택합니다.
2. 자격 증명 풀 생성을 선택합니다.
3. 자격 증명 풀 신뢰 구성에서 인증된 액세스, 게스트 액세스 또는 둘 다에 대해 자격 증명 풀을 설정하도록 선택합니다.

    - 인증된 액세스를 선택한 경우 자격 증명 풀에서 인증된 자격 증명의 소스로 설정하려는 자격 증명 유형을 하나 이상 선택합니다. 사용자 지정 개발자 공급자를 구성하는 경우 자격 증명 풀을 생성한 후에는 이를 수정하거나 삭제할 수 없습니다.
4. 권한 구성에서 자격 증명 풀의 인증된 사용자 또는 게스트 사용자의 기본 IAM 역할을 선택합니다.

a. Amazon Cognito가 기본 권한 및 자격 증명 풀과의 신뢰 관계가 있는 새 역할을 생성하도록 하려면 새 IAM 역할 생성을 선택합니다. 새 역할을 식별하는 IAM 역할 이름을 입력합니다 (예: `myidentitypool_authenticatedrole`). Amazon Cognito가 새 IAM 역할에 할당할 권한을 검토하려면 정책 문서 보기를 선택합니다.

b. 사용하려는 역할이 이미 AWS 계정 있는 경우 기존 IAM 역할을 사용하도록 선택할 수 있습니다. `cognito-identity.amazonaws.com`을 포함하도록 IAM 역할 신뢰 정책을 구성해야 합니다. 특정 자격 증명 풀의 인증된 사용자로부터 요청이 시작되었다는 증거가 제시되는 경우에만 Amazon Cognito가 역할을 맡을 수 있도록 역할 신뢰 정책을 구성합니다. 자세한 정보는 역할 트러스트 및 권한을 참조하세요.

5. Connect ID 제공자에 ID 풀 신뢰 구성에서 선택한 ID 제공자 (IdPs) 의 세부 정보를 입력합니다. OAuth 앱 클라이언트 정보를 제공하거나, Amazon Cognito 사용자 풀을 선택하거나, IAM IdP를 선택하거나, 개발자 공급자의 사용자 지정 식별자를 입력하라는 메시지가 표시될 수 있습니다.

   a. 각 IdP의 역할 설정을 선택합니다. 인증된 역할을 구성할 때 설정한 기본 역할을 해당 IdP의 사용자에게 할당하거나 규칙을 사용하여 역할 선택을 선택할 수 있습니다. Amazon Cognito 사용자 풀 IdP를 사용하면 토큰에서 preferred_role을 포함한 역할을 선택할 수도 있습니다. `cognito:preferred_role` 클레임에 대한 자세한 내용은 그룹에 우선 순위 값 할당을 참조하세요.

      i. 규칙을 사용하여 역할 선택을 선택한 경우 사용자 인증의 소스 클레임, 클레임 비교 기준으로 사용할 운영자, 이 역할 선택과 일치하도록 하는 값, 역할 할당이 일치할 때 할당할 역할을 입력합니다. 다른 조건에 따라 추가 규칙을 생성하려면 다른 항목 추가를 선택합니다.

      ii. 역할 해결을 선택합니다. 사용자의 클레임이 규칙과 일치하지 않는 경우 보안 인증을 거부하거나 인증된 역할의 보안 인증을 발급할 수 있습니다.

   b. 각 IdP별로 액세스 제어를 위한 속성을 구성합니다. 액세스 제어를 위한 속성은 사용자 클레임을 Amazon Cognito가 임시 세션에 적용하는 보안 주체 태그에 매핑합니다. 세션에 적용하는 태그를 기반으로 사용자 액세스를 필터링하는 IAM 정책을 구축할 수 있습니다.

      i. 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.

      ii. `sub` 및 `aud` 클레임 기반 보안 주체 태그를 적용하려면 기본 매핑 사용을 선택합니다.

      iii. 보안 주체 태그에 대한 속성의 자체 사용자 지정 스키마를 생성하려면 사용자 지정 매핑 사용을 선택합니다. 그런 다음 태그에 표시하려는 각 클레임에서 소싱하려는 태그 키를 입력합니다.

6. 속성 구성에서 자격 증명 풀 이름 아래에 이름을 입력합니다.

7. 기본(Classic) 인증에서 기본 흐름 활성화 여부를 선택합니다. 기본 흐름이 활성화되면 선택한 역할을 IdPs 우회하여 직접 AssumeRoleWithWebIdentity호출할 수 있습니다. 자세한 정보는 자격 증명 풀(페더레이션 자격 증명) 인증 흐름을 참조하세요.
8. 자격 증명 풀에 태그를 적용하려면 태그에서 태그 추가를 선택합니다.
9. 검토 및 생성에서 새 자격 증명 풀에 대한 선택 사항을 확인합니다. 편집을 선택하여 마법사로 돌아가서 설정을 변경합니다. 완료하면 자격 증명 풀 생성을 선택합니다.

SDK 설정

Amazon Cognito 자격 증명 풀을 사용하려면 AWS SDK for Java, 또는 를 설정하십시오 AWS Amplify. AWS SDK for .NET자세한 내용은 다음 항목을 참조하십시오.

- 개발자 안내서의 SDK 설정 JavaScriptAWS SDK for Java
- Amplify 개발자 센터의 Amplify 설명서
- AWS SDK for .NET 개발자 안내서의 Amazon Cognito 보안 인증 공급자

자격 증명 공급자 통합

Amazon Cognito 자격 증명 풀(페더레이션 자격 증명)은 Amazon Cognito 사용자 풀, Amazon, Facebook, Google, Apple 및 SAML ID 제공업체를 비롯한 페더레이션 ID 제공업체, 인증되지 않은 자격 증명을 통한 사용자 인증을 지원합니다. 이 기능은 자체 백엔드 인증 프로세스를 통해 사용자를 등록하고 인증할 수 있는 개발자 인증 자격 증명(자격 증명 풀)도 지원합니다.

Amazon Cognito 사용자 풀을 사용하여 사용자 디렉터리를 만드는 데 대한 자세한 내용은 Amazon Cognito 사용자 풀 및 로그인 후 자격 증명 풀을 AWS 서비스 사용하여 액세스 섹션을 참조하세요.

외부 자격 증명 공급자 사용에 대한 자세한 내용은 자격 증명 풀 외부 자격 증명 공급자 섹션을 참조하세요.

자체 백엔드 인증 프로세스 통합에 대한 자세한 내용은 개발자 인증 자격 증명(자격 증명 풀) 섹션을 참조하세요.

자격 증명 얻기

Amazon Cognito AWS 자격 증명 풀은 게스트 (미인증) 인 사용자와 인증을 받고 토큰을 받은 사용자에게 임시 자격 증명을 제공합니다. 앱은 이러한 AWS 자격 증명을 사용하여 Amazon API Gateway를

AWS 통해 내부 AWS 또는 외부의 백엔드에 안전하게 액세스할 수 있습니다. 자격 증명 얻기 섹션을 참조하세요.

아마존 Cognito의 설정 옵션 안내

구조화된 가이드 환경에서 Amazon Cognito의 기능을 평가하는 것이 좋습니다. 다음은 사용자 풀 및 자격 증명 풀을 통해 맞춤형 경험을 제공하는 몇 가지 외부 리소스입니다.

워크숍을 완료하세요.

AWS 워크샵 스튜디오에서는 대부분의 Amazon Cognito 기능을 설정하는 과정을 안내하는 워크숍을 주최합니다. 이러한 기능에는 사용자 풀 API, 사용자 풀 호스팅된 UI, 자격 증명 풀, 보안 구성 등이 포함됩니다.

예제에서 애플리케이션 코드를 추가하세요.

이 가이드의 코드 예제 장에는 사용자 풀 및 자격 증명 풀과 함께 사용할 수 있는 애플리케이션 코드가 있습니다. 코드 예제 장의 사용자 풀 섹션에는 개별 작업을 다루는 짧은 스니펫과 다양한 프로그래밍 언어의 end-to-end 예제 애플리케이션을 위한 더 긴 예제가 있습니다.

다음을 사용하여 풀스택 애플리케이션을 생성하십시오. AWS Amplify

AWS Amplify애플리케이션과 사용자 인터페이스를 개발하고 호스팅하려는 개발자를 AWS 서비스 위한 것입니다. Amazon Cognito는 Amplify의 인증 구성 요소입니다. 애플리케이션에 인증을 추가하면 Amplify는 Amazon Cognito 사용자 풀 및 자격 증명 풀 리소스의 배포를 자동화할 수 있습니다. 웹 및 모바일 앱과 Amazon Cognito 인증 및 권한 부여 통합 섹션도 참조하세요.

더 많은 Amazon Cognito 애플리케이션 리소스는 다음에서 확인할 수 있습니다. GitHub

- Amazon Cognito용.NET을 사용한 인증 흐름 예제
- Amazon Cognito 비밀번호 없는 인증
- PetStoreAmazon 검증 권한을 사용한 예제
- ABAC+ 자격 증명 풀을 사용하여 리소스에 액세스하는 AWS 샘플 React 앱
- CDK를 사용한 Amazon Cognito 및 API Gateway 기반 머신 간 인증 AWS
- Amazon Cognito, API Gateway, IAM을 사용하여 세분화된 권한 부여 구축
- CloudFront권한 부여 @edge

워크숍 더 보기

- Amazon Cognito를 사용하여 비밀번호 없는 인증을 구현하고 WebAuthn

- Amazon Cognito 사용자 풀을 사용하는 멀티테넌트 SaaS 자격 증명
- 아마존 Cognito JWT 딥 다이브

웹 및 모바일 앱과 Amazon Cognito 인증 및 권한 부여 통합

앱을 Amazon Cognito 앱 클라이언트와 통합하면 사용자 인증 및 권한 부여를 위한 API 작업을 간접적으로 호출할 수 있습니다. Amazon AWS AmplifyCognito를 웹 및 모바일 앱과 통합하는 데 사용하는 것이 좋습니다. AWS Amplify 프런트엔드 웹 및 모바일 개발자가 풀스택 애플리케이션을 쉽게 구축, 연결 및 호스팅할 수 있는 완벽한 솔루션이며 AWS, 사용 사례가 AWS 서비스 발전함에 따라 유연하게 활용할 수 있습니다. Amplify Auth는 주로 Amazon Cognito를 사용하여 인증 기능을 구축합니다.

주제

- 인증: AWS Amplify
- AWS SDK를 사용하는 인증
- Amazon Verified Permissions를 통한 권한 부여

Amazon Cognito의 일반적인 구현에서는 시각적 도구와 API를 함께 사용합니다. Amazon Cognito 콘솔은 Amazon Cognito 사용자 풀과 자격 증명 풀을 설정하고 관리하기 위한 시각적 인터페이스입니다. 호스팅된 UI는 Amazon Cognito 사용자 풀을 빠르게 테스트하고 배포할 수 있는 ready-to-use 웹 기반 로그인 애플리케이션입니다. 또한 대부분의 Amazon Cognito 배포에서는 사용자 풀 및 자격 증명 풀과 상호 작용하는 코드를 앱에 추가해야 합니다. 예를 들어 앱은 사용자 로그인을 위해 호스팅 UI를 간접적으로 호출한 다음 앱 코드에서 토큰 엔드포인트를 직접적으로 호출하여 사용자의 인증 코드를 토큰으로 교환할 수 있습니다. 그런 다음 앱은 사용자 토큰을 해석 및 저장하고 인증 및 권한 부여를 위해 적절한 컨텍스트에서 토큰을 제시해야 합니다. Amplify는 이러한 프로세스를 위한 내장 함수가 포함된 안내 통합 도구를 추가합니다.

또한 Amazon Cognito 리소스를 완전히 코드로 구축할 수도 있습니다. 사용자 지정 앱 코드를 시작하려면 AWS SDK용 Amazon Cognito 코드 예제를 참조하세요. OpenID Connect 자격 증명 공급자로서 Amazon Cognito와 통합하려면 OpenID Connect 개발자 도구를 사용합니다.

Amazon Cognito 인증 및 권한 부여를 사용하기 전에 앱 플랫폼을 선택하고 서비스와 통합할 코드를 준비합니다. 사용 가능한 플랫폼에 대해서는 AWS SDK를 사용하는 인증 섹션을 참조하세요. Amazon Cognito 및 기타 제품을 위한 명령줄 SDK로 AWS 서비스, Amazon Cognito API에 익숙해질 수 있는 귀중한 자료입니다. AWS CLI

> **Note**
>
> Amazon Cognito의 일부 구성 요소는 API만을 사용하여 구성할 수 있습니다. 예를 들어, 또는 API 요청에서 CreateUserPool클래스의 속성을 `LambdaConfig` 업데이트하는 요청으

로 사용자 풀 사용자 지정 SMS 또는 이메일 발신자 Lambda 트리거만 설정할 수 있습니다. UserPoolUpdateUserPool

Amazon Cognito 사용자 풀 API는 여러 API 작업 클래스와 네임스페이스를 공유합니다. 한 클래스는 사용자 풀과 해당 프로세스, 자격 증명 공급자 및 사용자를 구성합니다. 또 다른 클래스에는 퍼블릭 클라이언트의 사용자가 로그인, 로그아웃 및 프로필 관리를 수행할 수 있는 인증되지 않은 작업이 포함됩니다. API 작업의 최종 클래스는 기밀 서버 측 클라이언트에서 자체 AWS 자격 증명으로 승인한 사용자 작업을 수행합니다. 앱 코드 구현을 시작하기 전에 의도한 앱 아키텍처를 알아야 합니다. 자세한 정보는 Amazon Cognito 사용자 풀 API 및 사용자 풀 엔드포인트 사용을 참조하세요.

인증: AWS Amplify

AWS Amplify 웹 및 모바일 애플리케이션 구축을 위한 완벽한 솔루션입니다. Amplify를 통해 Amplify 라이브러리를 사용하여 기존 리소스에 연결하거나 Amplify 명령줄 인터페이스(CLI)를 사용하여 새 리소스를 생성 및 구성할 수 있습니다. Amplify에는 앱에서 로그인 및 가입 경험을 설정하고 사용자 지정할 수 있는 Authenticator와 같은 연결된 UI 구성 요소도 있습니다.

프런트 엔드 앱에서 Amplify 인증 기능을 사용하려면 플랫폼별 다음 설명서를 참조하세요.

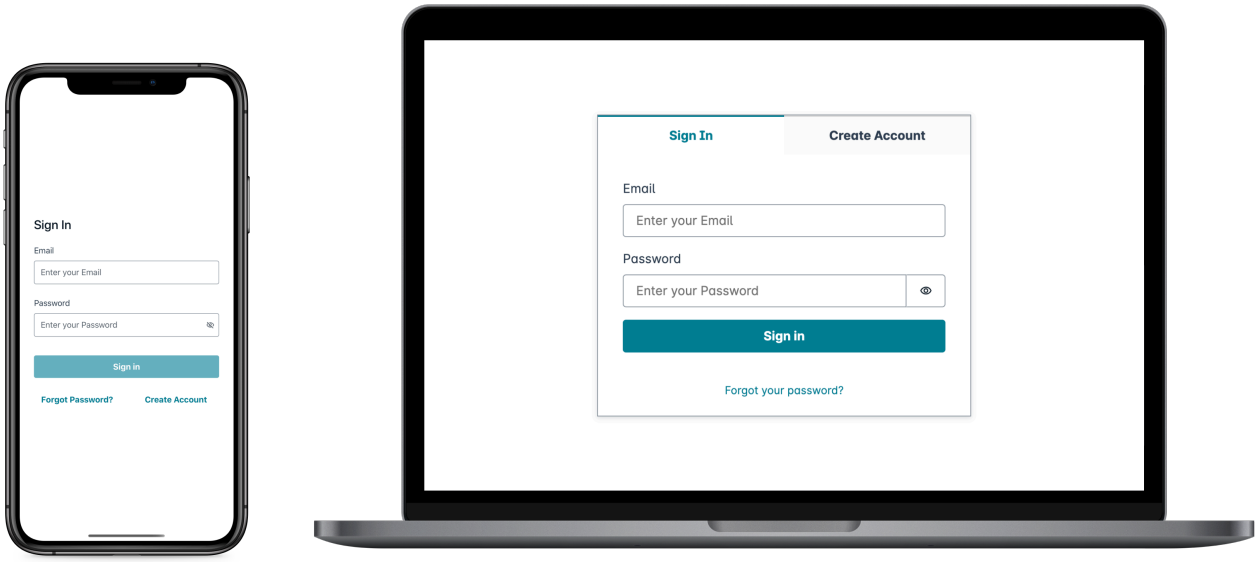
- 에 대한 인증을 Amplify JavaScript
- iOS에 대한 Amplify 인증
- Android에 대한 Amplify 인증
- Flutter에 대한 Amplify 인증

Amplify 라이브러리는 오픈 소스이며 에서 사용할 수 있습니다. GitHub Amplify Auth가 Amazon Cognito 인증을 구현하는 방법에 대해 자세히 알아보려면 다음 라이브러리를 방문하세요.

- amplify-js
- amplify-swift
- amplify-flutter
- amplify-android

Amplify로 사용자 인터페이스(UI) 생성

Amazon Cognito 사용자 풀 호스팅 UI는 웹 또는 모바일 앱에 대한 인증 프런트 엔드의 필수 요구 사항을 충족할 수 있습니다. 호스팅 UI가 수용하는 파라미터 이상으로 사용자 인터페이스(UI)를 사용자 지정하려면 사용자 지정 앱을 만드세요. Amplify UI는 다양한 언어로 된 사용자 지정 가능한 프런트 엔드 구성 요소 모음입니다.



사용자 지정 인증 구성 요소를 시작하려면 해당 Authenticator 구성 요소에 대한 다음 설명서를 참조하세요.

- Android용 Authenticator
- Angular용 Authenticator
- Flutter용 Authenticator
- React용 Authenticator
- React Native용 Authenticator
- Swift용 Authenticator
- Vue용 Authenticator

AWS SDK를 사용하는 인증

안전한 백엔드를 사용하여 Amazon Cognito와 상호 작용하는 자체 자격 증명 마이크로서비스를 구축하려면 원하는 언어로 된 SDK를 사용하여 Amazon Cognito 사용자 풀 및 Amazon Cognito 자격 증명 풀 API에 연결하십시오. AWS

각 API 작업에 대한 자세한 내용은 Amazon Cognito 사용자 풀 API 참조 및 Amazon Cognito API 참조를 참조하세요. 이 문서에는 지원되는 플랫폼에서 다양한 SDK를 사용하기 위한 리소스가 포함된 다음 사항도 참조하십시오 섹션도 포함되어 있습니다.

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS Java V2용 SDK
- AWS 에 대한 SDK JavaScript
- AWS PHP V3용 SDK
- AWS 파이썬용 SDK
- AWS 루비 V3용 SDK

Amazon Verified Permissions를 통한 권한 부여

Amazon Verified Permissions는 사용자가 구축하는 애플리케이션에 대한 권한 부여 서비스입니다. Amazon Cognito 사용자 풀을 자격 증명 소스로 추가하면 앱에서 허용 또는 거부 결정을 위해 사용자 풀 액세스 또는 자격 증명(ID) 토큰을 Verified Permissions에 전달할 수 있습니다. Verified Permissions는 Cedar 정책 언어로 작성한 정책을 기반으로 사용자의 속성과 요청 컨텍스트를 고려합니다. 요청 컨텍스트에는 요청한 문서, 이미지 또는 기타 리소스의 식별자와 사용자가 리소스에 대해 수행하려는 작업이 포함될 수 있습니다.

앱은 API 요청의 검증된 권한에 IsAuthorizedWithToken대한 사용자 ID 또는 액세스 토큰을 제공할 수 있습니다. BatchIsAuthorizedWithToken 이러한 API 작업은 사용자를 `Principal` A로 받아들이고 `Resource` 사용자가 액세스하려는 `Action` 서버에 대한 권한 부여 결정을 내립니다. 추가 사용자 `Context` 지정은 세부적인 액세스 결정에 기여할 수 있습니다.

앱이 `IsAuthorizedWithToken` API 요청에 토큰을 제시하면 Verified Permissions는 다음과 같은 검증을 수행합니다.

1. 사용자 풀이 요청된 정책 스토어에 대해 구성된 Verified Permissions 자격 증명 소스입니다.
2. 액세스 또는 자격 증명 토큰의 `client_id` 또는 `aud` 클레임이 각각 Verified Permissions에 제공한 사용자 풀 앱 클라이언트 ID와 일치합니다. 이 클레임을 확인하려면 Verified Permissions 자격 증명 소스에서 클라이언트 ID 검증 구성 작업을 수행해야 합니다.
3. 토큰이 만료되지 않았습니다.
4. 토큰의 `token_use` 클레임 값은 전달한 매개변수와 `IsAuthorizedWithToken` 일치합니다. `token_use`클레임을 파라미터에 전달한 `access` 경우와 `accessToken` 파라미터에 전달한 `id` 경우에만 클레임이 `identityToken` 유효합니다.
5. 토큰의 서명이 사용자 풀의 게시된 JSON 웹 키(JWK)에서 가져온 것입니다. `https://cognito-idp.`*`Region`*`.amazonaws.com/`*`your user pool ID`*`/.well-known/jwks.json`에서 JWK를 볼 수 있습니다.

취소된 토큰 및 삭제된 사용자

Verified Permissions는 자격 증명 소스와 사용자 토큰의 만료 시간을 통해 알고 있는 정보만 검증합니다. Verified Permissions는 토큰 취소 또는 사용자 존재 여부를 확인하지 않습니다. 사용자 토큰을 취소했거나 사용자 풀에서 사용자 프로필을 삭제한 경우에도 Verified Permissions는 토큰이 만료될 때까지 토큰을 유효한 것으로 간주합니다.

정책 평가

사용자 풀을 정책 스토어의 자격 증명 소스로 구성합니다. 요청에서 사용자의 토큰을 Verified Permissions에 제출하도록 앱을 구성합니다. 각 요청에 대해 Verified Permissions는 토큰의 클레임을 정책과 비교합니다. Verified Permissions 정책은 AWS의 IAM 정책과 마찬가지로 보안 주체, 리소스 및 작업을 선언합니다. 검증된 권한은 허용된 작업과 `Allow` 일치하고 `Deny` 명시적 작업과 일치하지 않으면 요청에 응답하고 그렇지 않으면 로 응답합니다. `Deny` 자세한 내용은 Amazon Verified Permissions 사용 설명서의 Amazon Verified Permissions 정책을 참조하세요.

토큰 사용자 지정

Verified Permissions에 제시하려는 사용자 주장을 변경, 추가, 제거하려면 a를 사용하여 액세스 및 ID 토큰의 콘텐츠를 사용자 지정하십시오. 사전 토큰 생성 Lambda 트리거 사전 토큰 생성 트리거를 사용하여 토큰에 클레임을 추가하고 수정할 수 있습니다. 예를 들어 데이터베이스에서 추가 사용자 속성을 쿼리하고 ID 토큰으로 인코딩할 수 있습니다.

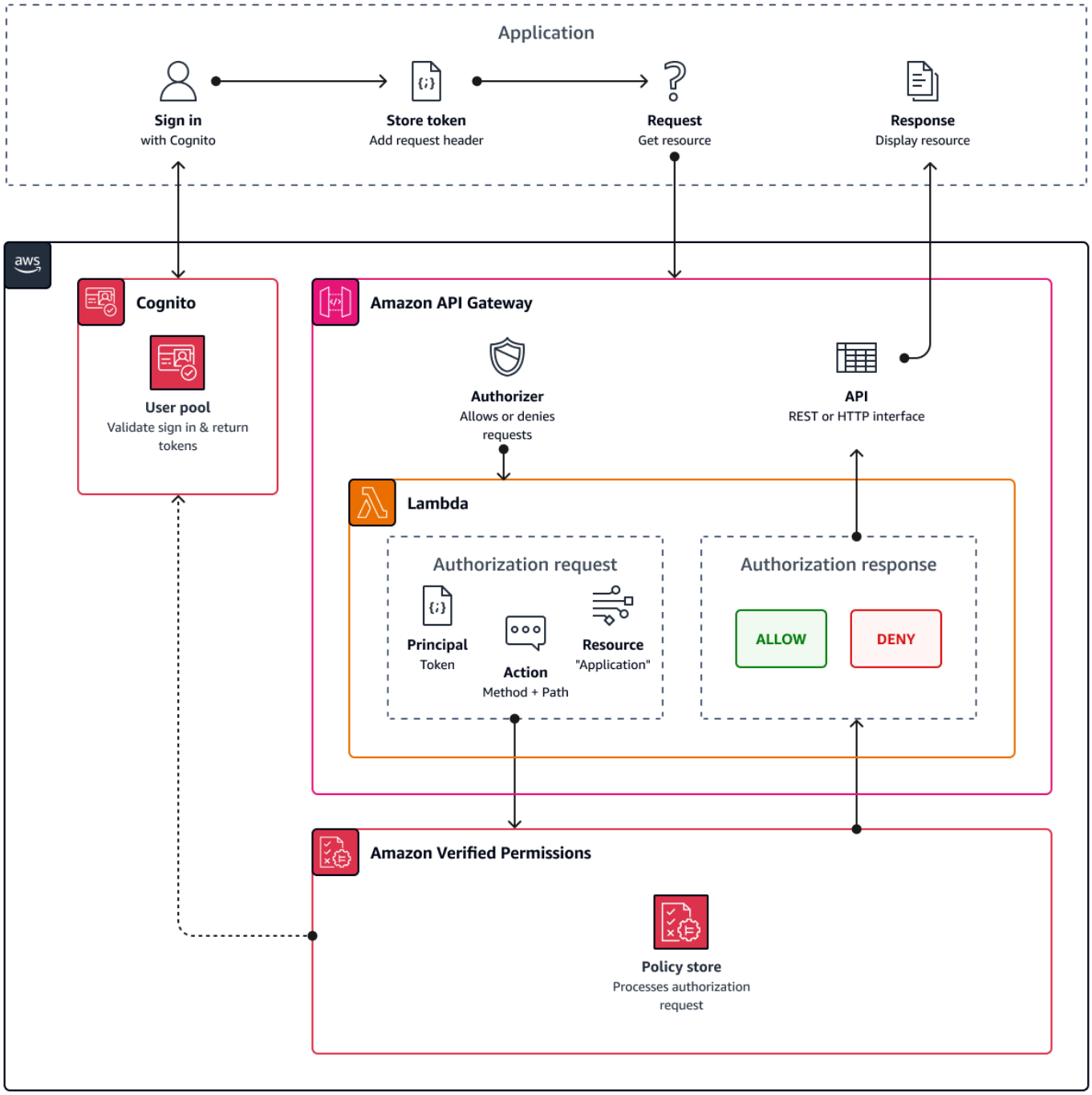
> **Note**
>
> Verified Permissions가 클레임을 처리하는 방식 때문에 `cognito`, `dev` 또는 `custom`이라는 이름의 클레임을 사전 토큰 생성 함수에 추가하지 마세요. 이러한 예약된 클레임 접두사를 `cognito:username`과 같이 콜론으로 구분된 형식이 아닌 전체 클레임 이름으로 제시하면 권한 부여 요청이 실패합니다.

검증된 권한이 Amazon Cognito 토큰의 클레임을 권한 부여 정책에 매핑하는 방법에 대한 자세한 내용은 Amazon Cognito 토큰을 검증된 권한 스키마에 매핑 섹션을 참조하세요.

검증된 권한을 통한 API 인증

ID 또는 액세스 토큰은 검증된 권한을 사용하여 백엔드 Amazon API Gateway REST API에 대한 요청을 승인할 수 있습니다. 사용자 풀 및 API로 즉시 연결되는 링크가 있는 정책 스토어를 생성할 수 있습니다. Cognito 및 API Gateway로 설정 시작 옵션을 사용하면 검증된 권한은 사용자 풀 ID 소스를 정책 스토어에 추가하고 Lambda 권한 부여자를 API에 추가합니다. 애플리케이션이 사용자 풀 베어러 토큰을 API에 전달하면 Lambda 권한 부여자가 검증된 권한을 호출합니다. 권한 부여자는 토큰을 보안 주체로 전달하고 요청 경로와 메서드를 작업으로 전달합니다.

다음 다이어그램은 검증된 권한이 있는 API Gateway API의 권한 부여 흐름을 보여줍니다. 자세한 내용은 Amazon 검증 권한 사용 설명서의 API 연결 정책 스토어를 참조하십시오.



검증된 권한은 사용자 풀 그룹을 중심으로 API 인증을 구성합니다. ID와 액세스 토큰 모두 `cognito:groups` 클레임을 포함하므로 정책 저장소는 다양한 애플리케이션 컨텍스트에서 API에 대한 RBAC (역할 기반 액세스 제어) 를 관리할 수 있습니다.

정책 저장소 설정 선택

정책 저장소에서 ID 소스를 구성할 때는 액세스 토큰을 처리할지 ID 토큰을 처리할지 여부를 선택해야 합니다. 이 결정은 정책 엔진 운영 방식에 있어 매우 중요합니다. ID 토큰에는 사용자 속성이 포함됩니다. 액세스 토큰에는 사용자 액세스 제어 정보 (OAuth 범위) 가 포함됩니다. 두 토큰 유형 모두 그룹 구성원 정보를 포함하지만 일반적으로 검증된 권한 정책 저장소가 있는 RBAC용 액세스 토큰을 사용하는 것이 좋습니다. 액세스 토큰은 권한 부여 결정에 영향을 줄 수 있는 범위를 통해 그룹 구성원 자격을 증가시킵니다. 액세스 토큰의 클레임은 권한 부여 요청의 컨텍스트가 됩니다.

또한 사용자 풀을 ID 소스로 구성할 때 사용자 및 그룹 엔티티 유형을 구성해야 합니다. 엔티티 유형은 검증된 권한 정책에서 참조할 수 있는 보안 주체, 작업 및 리소스 식별자입니다. 정책 저장소의 엔티티는 멤버 자격 관계를 가질 수 있으며, 여기서 한 엔티티는 상위 엔티티의 구성원이 될 수 있습니다. 멤버십이 있으면 주도자 그룹, 작업 그룹 및 리소스 그룹을 참조할 수 있습니다. 사용자 풀 그룹의 경우 지정하는 사용자 엔티티 유형이 그룹 엔티티 유형의 멤버여야 합니다. API 연결 정책 저장소를 설정하거나 Verified Permissions 콘솔의 설정 안내를 따르는 경우 정책 저장소는 자동으로 이 상위-구성원 관계를 가집니다.

ID 토큰은 RBAC를 ABAC (속성 기반 액세스 제어) 와 결합할 수 있습니다. API 연결 정책 저장소를 생성한 후 사용자 특성 및 그룹 구성원으로 정책을 개선할 수 있습니다. ID 토큰의 특성 클레임은 권한 부여 요청의 주요 특성이 됩니다. 정책은 주요 특성에 따라 권한 부여 결정을 내릴 수 있습니다.

제공하는 허용 가능한 앱 클라이언트 목록과 일치하는 `aud` 또는 `client_id` 클레임이 있는 토큰을 수락하도록 정책 저장소를 구성할 수도 있습니다.

역할 기반 API 권한 부여에 대한 예제 정책

다음 예제 정책은 예제 REST API에 대한 검증된 권한 정책 저장소를 설정하여 생성되었습니다. PetStore

```
permit(
  principal in PetStore::UserGroup::"us-east-1_EXAMPLE|MyGroup",
  action in [ PetStore::Action::"get /pets", PetStore::Action::"get /pets/{petId}" ],
  resource
  );
```

확인된 권한은 다음과 같은 경우 애플리케이션의 권한 부여 요청에 대한 `Allow` 결정을 반환합니다.

1. 애플리케이션이 `Authorization` 헤더의 ID 또는 액세스 토큰을 베어러 토큰으로 전달했습니다.
2. 애플리케이션이 문자열이 `MyGroup` 포함된 `cognito:groups` 클레임과 함께 토큰을 전달했습니다.

3. 애플리케이션에서 예를 들어, `https://myapi.example.com/pets` 또는 를 `HTTP GET https://myapi.example.com/pets/scrappy` 요청했습니다.

Amazon Cognito 사용자에 대한 정책 예시

또한 사용자 풀은 API 요청 이외의 조건에서 검증된 권한에 대한 권한 부여 요청을 생성할 수 있습니다. 애플리케이션의 모든 액세스 제어 결정을 정책 저장소에 제출할 수 있습니다. 예를 들어 요청이 네트워크를 전송하기 전에 속성 기반 액세스 제어로 Amazon DynamoDB 또는 Amazon S3 보안을 보완하여 할당량 사용을 줄일 수 있습니다.

다음 예에서는 Cedar 정책 언어를 사용하여 하나의 사용자 풀 앱 클라이언트로 인증하는 Finance 사용자가 `example_image.png`를 읽고 쓸 수 있도록 허용합니다. 앱의 사용자인 John은 앱 클라이언트로부터 ID 토큰을 받아 GET 요청에서 권한 부여가 필요한 URL `https://example.com/images/example_image.png`에 전달합니다. John의 ID 토큰에는 사용자 풀 앱 클라이언트 ID `1234567890example`에 대한 `aud` 클레임이 있습니다. 또한 사전 토큰 생성 Lambda 함수가 John에 대한 값 `Finance1234`를 가진 새 클레임 `costCenter`를 삽입했습니다.

```
permit (
   principal,
   actions in [ExampleCorp::Action::"readFile", "writeFile"],
   resource == ExampleCorp::Photo::"example_image.png"
)
when {
   principal.aud == "1234567890example" &&
   principal.custom.costCenter like "Finance*"
};
```

다음 요청 본문은 `Allow` 응답을 생성합니다.

```
{
   "accesstoken": "[John's ID token]",
   "action": {
      "actionId": "readFile",
      "actionType": "Action"
   },
   "resource": {
      "entityId": "example_image.png",
      "entityType": "Photo"
   }
}
```

Verified Permissions 정책에 보안 주체를 지정하려면 다음과 같은 형식을 사용합니다.

```
permit (
   principal == [Namespace]::[Entity]::"[user pool ID]"|"[user sub]",
   action,
   resource
);
```

다음은 서브가 있는 ID 또는 사용자 ID (사용자 IDus-east-1_Example) 를 가진 사용자 풀의 사용자를 위한 보안 주체 예제입니다. 973db890-092c-49e4-a9d0-912a4c0a20c7

```
principal == ExampleCorp::User::"us-east-1_Example|973db890-092c-49e4-
a9d0-912a4c0a20c7",
```

검증된 권한 정책에서 사용자 그룹을 지정하려면 다음 형식을 사용하십시오.

```
permit (
   principal in [Namespace]::[Group Entity]::"[Group name]",
   action,
   resource
);
```

다음은 예제입니다.

속성 기반 액세스 제어

앱에 대한 검증된 권한을 통한 권한 부여 및 자격 증명용 Amazon Cognito AWS 자격 증명 풀의 액세스 제어 기능 속성은 모두 속성 기반 액세스 제어 (ABAC) 의 한 형태입니다. 다음은 Verified Permissions와 Amazon Cognito ABAC의 기능을 비교한 것입니다. ABAC에서 시스템은 엔터티의 속성을 검사하고 사용자가 정의한 조건에 따라 권한 부여 결정을 내립니다.

Service	프로세스	Result
Amazon Verified Permissions	사용자 풀 JWT의 분석을 통해 `Allow` OR `Deny` 결정을 반환합니다.	Cedar 정책 평가에 따라 애플리케이션 리소스에 대한 액세스가 성공하거나 실패합니다.
Amazon Cognito 자격	속성에 따라 사용자에게 세션 태그를 할당합니다. IAM 정책 조건은 태그	IAM 역할의 임시 AWS 자격 증명이 포함된 태그가 지정된 세션입니다.

Service	프로세스	Result
증명 풀 (액세스 제어를 위한 속성)	`Allow` 또는 `Deny` 사용자 액세스를 확인할 수 있습니다. AWS 서비스	

AWS SDK를 사용한 Amazon Cognito 코드 예시

다음 코드 예시는 Amazon Cognito를 AWS 소프트웨어 개발 키트(SDK)와 함께 사용하는 방법을 보여 줍니다.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 이 서비스를 SDK와 함께 사용 AWS 섹션을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

코드 예시

        - AWS SDK 또는 ConfirmForgotPassword CLI와 함께 사용
        - AWS SDK 또는 ConfirmSignUp CLI와 함께 사용
        - AWS SDK 또는 CreateUserPool CLI와 함께 사용
        - AWS SDK 또는 CreateUserPoolClient CLI와 함께 사용
        - AWS SDK 또는 DeleteUser CLI와 함께 사용
        - AWS SDK 또는 ForgotPassword CLI와 함께 사용
        - AWS SDK 또는 InitiateAuth CLI와 함께 사용
        - AWS SDK 또는 ListUserPools CLI와 함께 사용
        - AWS SDK 또는 ListUsers CLI와 함께 사용
        - AWS SDK 또는 ResendConfirmationCode CLI와 함께 사용
        - AWS SDK 또는 RespondToAuthChallenge CLI와 함께 사용
        - AWS SDK 또는 SignUp CLI와 함께 사용
        - AWS SDK 또는 UpdateUserPool CLI와 함께 사용
        - AWS SDK 또는 VerifySoftwareToken CLI와 함께 사용
    - SDK를 사용하는 Amazon Cognito 자격 증명 공급자의 시나리오 AWS
        - SDK를 사용하여 Lambda 함수를 사용하여 알려진 Amazon Cognito 사용자를 자동으로 확인합니다. AWS
        - SDK를 사용하여 Lambda 함수를 사용하여 알려진 Amazon Cognito 사용자를 자동으로 마이그레이션합니다. AWS
        - SDK를 사용하여 MFA가 필요한 Amazon Cognito 사용자 풀에 사용자를 등록합니다. AWS
        - SDK를 사용한 Amazon Cognito 사용자 인증 후 Lambda 함수를 사용하여 사용자 지정 활동 데이터를 작성합니다. AWS
- SDK를 사용한 Amazon Cognito Sync의 코드 예제 AWS
    - SDK를 사용한 Amazon Cognito 동기화 작업 AWS
        - AWS SDK 또는 ListIdentityPoolUsage CLI와 함께 사용

SDK를 사용한 Amazon Cognito 자격 증명의 코드 예제 AWS

다음 코드 예제는 AWS 소프트웨어 개발 키트 (SDK) 와 함께 Amazon Cognito ID를 사용하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

교차 서비스 예시는 여러 AWS 서비스전반에서 작동하는 샘플 애플리케이션입니다.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 을 참조하십시오. 이 서비스를 SDK와 함께 사용 AWS 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

코드 예시

- SDK를 사용한 Amazon Cognito 자격 증명 관련 작업 AWS
  - AWS SDK 또는 CreateIdentityPool CLI와 함께 사용
  - AWS SDK 또는 DeleteIdentityPool CLI와 함께 사용
  - AWS SDK 또는 DescribeIdentityPool CLI와 함께 사용
  - AWS SDK 또는 GetCredentialsForIdentity CLI와 함께 사용
  - AWS SDK 또는 GetIdentityPoolRoles CLI와 함께 사용
  - AWS SDK 또는 ListIdentityPools CLI와 함께 사용
  - AWS SDK 또는 SetIdentityPoolRoles CLI와 함께 사용
  - AWS SDK 또는 UpdateIdentityPool CLI와 함께 사용
- SDK를 사용한 Amazon Cognito 자격 증명의 크로스 서비스 예제 AWS
  - Amazon Transcribe 앱 구축
  - Amazon Textract 탐색기 애플리케이션 생성

SDK를 사용한 Amazon Cognito 자격 증명 관련 작업 AWS

다음 코드 예제는 SDK를 사용하여 개별 Amazon Cognito 자격 증명 작업을 수행하는 방법을 보여줍니다. AWS 이들 발췌문은 Amazon Cognito Identity API를 직접적으로 호출하며, 컨텍스트에서 실행되어야 하는 더 큰 프로그램에서 발췌한 코드입니다. 각 예제에는 코드 GitHub 설정 및 실행 지침을 찾을 수 있는 링크가 포함되어 있습니다.

다음 예제에는 가장 일반적으로 사용되는 작업만 포함되어 있습니다. 전체 목록은 Amazon Cognito Identity API Reference(Amazon Cognito 자격 증명 API 참조)를 참조하세요.

예제

- AWS SDK 또는 CreateIdentityPool CLI와 함께 사용

- AWS SDK 또는 DeleteIdentityPool CLI와 함께 사용
- AWS SDK 또는 DescribeIdentityPool CLI와 함께 사용
- AWS SDK 또는 GetCredentialsForIdentity CLI와 함께 사용
- AWS SDK 또는 GetIdentityPoolRoles CLI와 함께 사용
- AWS SDK 또는 ListIdentityPools CLI와 함께 사용
- AWS SDK 또는 SetIdentityPoolRoles CLI와 함께 사용
- AWS SDK 또는 UpdateIdentityPool CLI와 함께 사용

AWS SDK 또는 **CreateIdentityPool** CLI와 함께 사용

다음 코드 예제는 CreateIdentityPool의 사용 방법을 보여줍니다.

CLI

AWS CLI

Cognito 자격 증명 풀 공급자를 사용하여 자격 증명 풀 생성

이 예제에서는 라는 자격 증명 풀을 생성합니다 MyIdentityPool. Cognito 자격 증명 풀 공급자가 있습니다. 인증되지 않은 자격 증명은 허용되지 않습니다.

명령:

```
aws cognito-identity create-identity-pool --identity-pool-name
 MyIdentityPool --no-allow-unauthenticated-identities --cognito-
identity-providers ProviderName="cognito-idp.us-west-2.amazonaws.com/us-
west-2_aaaaaaaaa",ClientId="3n4b5urk1ft4fl3mg5e62d9ado",ServerSideTokenCheck=false
```

출력:

```
{
  "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
  "IdentityPoolName": "MyIdentityPool",
  "AllowUnauthenticatedIdentities": false,
  "CognitoIdentityProviders": [
      {
          "ProviderName": "cognito-idp.us-west-2.amazonaws.com/us-
west-2_111111111",
```

```
            "ClientId": "3n4b5urk1ft4fl3mg5e62d9ado",
            "ServerSideTokenCheck": false
        }
    ]
}
```

- API 세부 정보는 AWS CLI 명령 CreateIdentityPool참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc

/**
 * 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 CreateIdentityPool {
    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <identityPoolName>\s
```

```
                Where:
                    identityPoolName - The name to give your identity pool.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityPoolName = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String identityPoolId = createIdPool(cognitoClient, identityPoolName);
        System.out.println("Unity pool ID " + identityPoolId);
        cognitoClient.close();
    }

    public static String createIdPool(CognitoIdentityClient cognitoClient, String
 identityPoolName) {
        try {
            CreateIdentityPoolRequest poolRequest =
 CreateIdentityPoolRequest.builder()
                    .allowUnauthenticatedIdentities(false)
                    .identityPoolName(identityPoolName)
                    .build();

            CreateIdentityPoolResponse response =
 cognitoClient.createIdentityPool(poolRequest);
            return response.identityPoolId();

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API CreateIdentityPool참조를 참조하십시오.

PowerShell

도구: PowerShell

예 1: 인증되지 않은 ID를 허용하는 새 자격 증명 풀을 생성합니다.

```
New-CGIIdentityPool -AllowUnauthenticatedIdentities $true -IdentityPoolName
 CommonTests13
```

출력:

```
LoggedAt                        : 8/12/2015 4:56:07 PM
AllowUnauthenticatedIdentities  : True
DeveloperProviderName           :
IdentityPoolId                  : us-east-1:15d49393-ab16-431a-b26e-EXAMPLEGUID3
IdentityPoolName                : CommonTests13
OpenIdConnectProviderARNs       : {}
SupportedLoginProviders         : {}
ResponseMetadata                : Amazon.Runtime.ResponseMetadata
ContentLength                   : 136
HttpStatusCode                  : OK
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 CreateIdentityPool.AWS Tools for PowerShell

Swift

SDK for Swift

> **Note**
>
> 이 사전 릴리스 설명서는 평가판 버전 SDK에 관한 것입니다. 내용은 변경될 수 있습니다.

> **Note**
>
> 자세한 내용은 다음과 같습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

새 자격 증명 풀을 생성합니다.

```
    /// Create a new identity pool and return its ID.
    ///
    /// - Parameters:
    ///     - name: The name to give the new identity pool.
    ///
    /// - Returns: A string containing the newly created pool's ID, or `nil`
    ///   if an error occurred.
    ///
    func createIdentityPool(name: String) async throws -> String? {
        let cognitoInputCall = CreateIdentityPoolInput(developerProviderName:
"com.exampleco.CognitoIdentityDemo",
                                                       identityPoolName: name)

        let result = try await cognitoIdentityClient.createIdentityPool(input:
cognitoInputCall)
        guard let poolId = result.identityPoolId else {
            return nil
        }

        return poolId
    }
```

- 자세한 내용은 AWS SDK for Swift 개발자 안내서를 참조하십시오.
- API 세부 정보는 Swift API용AWS SDK 참조를 참조하십시오 CreateIdentityPool.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. 이 서비스를 SDK와 함께 사용 AWS 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **DeleteIdentityPool** CLI와 함께 사용

다음 코드 예제는 DeleteIdentityPool의 사용 방법을 보여줍니다.

CLI

AWS CLI

자격 증명 풀 삭제

다음 delete-identity-pool 예시에서는 지정된 자격 증명 풀을 삭제합니다.

명령:

```
aws cognito-identity delete-identity-pool \
    --identity-pool-id "us-west-2:11111111-1111-1111-1111-111111111111"
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 DeleteIdentityPool참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.DeleteIdentityPoolRequest;

/**
 * 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 DeleteIdentityPool {

    public static void main(String[] args) {
        final String usage = """

                Usage:
```

```
                    <identityPoolId>\s

                Where:
                    identityPoolId - The Id value of your identity pool.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityPoold = args[0];
        CognitoIdentityClient cognitoIdClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();

        deleteIdPool(cognitoIdClient, identityPoold);
        cognitoIdClient.close();
    }

    public static void deleteIdPool(CognitoIdentityClient cognitoIdClient, String
 identityPoold) {
        try {

            DeleteIdentityPoolRequest identityPoolRequest =
 DeleteIdentityPoolRequest.builder()
                    .identityPoolId(identityPoold)
                    .build();

            cognitoIdClient.deleteIdentityPool(identityPoolRequest);
            System.out.println("Done");

        } catch (AwsServiceException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API DeleteIdentityPool참조를 참조하십시오.

PowerShell

도구: PowerShell

예 1: 특정 자격 증명 풀을 삭제합니다.

```
Remove-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 DeleteIdentityPool.

Swift

SDK for Swift

> **Note**
>
> 이 사전 릴리스 설명서는 평가판 버전 SDK에 관한 것입니다. 내용은 변경될 수 있습니다.

> **Note**
>
> 자세한 내용은 다음과 같습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

지정된 자격 증명 풀을 삭제합니다.

```
    /// Delete the specified identity pool.
    ///
    /// - Parameters:
    ///   - id: The ID of the identity pool to delete.
    ///
    func deleteIdentityPool(id: String) async throws {
        let input = DeleteIdentityPoolInput(
            identityPoolId: id
        )

        _ = try await cognitoIdentityClient.deleteIdentityPool(input: input)
    }
```

- 자세한 내용은 AWS SDK for Swift 개발자 안내서를 참조하십시오.
- API 세부 정보는 Swift API용AWS SDK 참조를 참조하십시오 DeleteIdentityPool.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. 이 서비스를 SDK와 함께 사용 AWS 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **DescribeIdentityPool** CLI와 함께 사용

다음 코드 예제는 DescribeIdentityPool의 사용 방법을 보여줍니다.

CLI

AWS CLI

자격 증명 풀을 설명하려면

이 예제에서는 자격 증명 풀을 설명합니다.

명령:

```
aws cognito-identity describe-identity-pool --identity-pool-id "us-
west-2:11111111-1111-1111-1111-111111111111"
```

출력:

```
{
  "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
  "IdentityPoolName": "MyIdentityPool",
  "AllowUnauthenticatedIdentities": false,
  "CognitoIdentityProviders": [
      {
          "ProviderName": "cognito-idp.us-west-2.amazonaws.com/us-
west-2_111111111",
          "ClientId": "3n4b5urk1ft4fl3mg5e62d9ado",
          "ServerSideTokenCheck": false
      }
  ]
}
```

- API 세부 정보는 AWS CLI 명령 DescribeIdentityPool참조를 참조하십시오.

PowerShell

도구: PowerShell

예 1: ID를 기준으로 특정 자격 증명 풀에 대한 정보를 검색합니다.

```
Get-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

출력:

```
LoggedAt                        : 8/12/2015 4:29:40 PM
AllowUnauthenticatedIdentities  : True
DeveloperProviderName           :
IdentityPoolId                  : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
IdentityPoolName                : CommonTests1
OpenIdConnectProviderARNs       : {}
SupportedLoginProviders         : {}
ResponseMetadata                : Amazon.Runtime.ResponseMetadata
ContentLength                   : 142
HttpStatusCode                  : OK
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 DescribeIdentityPool.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. 이 서비스를 SDK와 함께 사용 AWS 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **GetCredentialsForIdentity** CLI와 함께 사용

다음 코드 예시에서는 `GetCredentialsForIdentity`을 사용하는 방법을 보여 줍니다.

Java

SDK for Java 2.x

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc

/**
 * 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 GetIdentityCredentials {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <identityId>\s

            Where:
                identityId - The Id of an existing identity in the format
 REGION:GUID.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityId = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        getCredsForIdentity(cognitoClient, identityId);
        cognitoClient.close();
```

```
    }

    public static void getCredsForIdentity(CognitoIdentityClient cognitoClient,
String identityId) {
        try {
            GetCredentialsForIdentityRequest getCredentialsForIdentityRequest =
GetCredentialsForIdentityRequest
                    .builder()
                    .identityId(identityId)
                    .build();

            GetCredentialsForIdentityResponse response = cognitoClient
                    .getCredentialsForIdentity(getCredentialsForIdentityRequest);
            System.out.println(
                    "Identity ID " + response.identityId() + ", Access key ID " +
response.credentials().accessKeyId());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API GetCredentialsForIdentity참조를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오이 서비스를 SDK와 함께 사용 AWS. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **GetIdentityPoolRoles** CLI와 함께 사용

다음 코드 예제는 GetIdentityPoolRoles의 사용 방법을 보여줍니다.

CLI

AWS CLI

자격 증명 풀 역할을 가져오려면

이 예에서는 자격 증명 풀 역할을 가져옵니다.

명령:

```
aws cognito-identity get-identity-pool-roles --identity-pool-id "us-
west-2:11111111-1111-1111-1111-111111111111"
```

출력:

```
{
  "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
  "Roles": {
      "authenticated": "arn:aws:iam::111111111111:role/
Cognito_MyIdentityPoolAuth_Role",
      "unauthenticated": "arn:aws:iam::111111111111:role/
Cognito_MyIdentityPoolUnauth_Role"
  }
}
```

- API 세부 정보는 AWS CLI 명령 GetIdentityPoolRoles참조를 참조하십시오.

PowerShell

도구: PowerShell

예 1: 특정 자격 증명 풀의 역할에 대한 정보를 가져옵니다.

```
Get-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

출력:

```
LoggedAt         : 8/12/2015 4:33:51 PM
IdentityPoolId   : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
Roles            : {[unauthenticated, arn:aws:iam::123456789012:role/
CommonTests1Role]}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 165
HttpStatusCode   : OK
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 GetIdentityPoolRoles.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. 이 서비스를 SDK와 함께 사용 AWS 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **ListIdentityPools** CLI와 함께 사용

다음 코드 예제는 ListIdentityPools의 사용 방법을 보여줍니다.

CLI

AWS CLI

자격 증명 풀 나열

이 예시에는 자격 증명 풀이 나열되어 있습니다. 최대 20개의 자격 증명이 나열되어 있습니다.

명령:

```
aws cognito-identity list-identity-pools --max-results 20
```

출력:

```
{
  "IdentityPools": [
      {
          "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
          "IdentityPoolName": "MyIdentityPool"
      },
      {
          "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
          "IdentityPoolName": "AnotherIdentityPool"
      },
      {
          "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
          "IdentityPoolName": "IdentityPoolRegionA"
      }
  ]
}
```

- API 세부 정보는 AWS CLI 명령 ListIdentityPools참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc

/**
 * 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 ListIdentityPools {
    public static void main(String[] args) {
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listIdPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listIdPools(CognitoIdentityClient cognitoClient) {
        try {
            ListIdentityPoolsRequest poolsRequest =
 ListIdentityPoolsRequest.builder()
```

```
                    .maxResults(15)
                    .build();

            ListIdentityPoolsResponse response =
 cognitoClient.listIdentityPools(poolsRequest);
            response.identityPools().forEach(pool -> {
                System.out.println("Pool ID: " + pool.identityPoolId());
                System.out.println("Pool name: " + pool.identityPoolName());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API ListIdentityPools참조를 참조하십시오.

PowerShell

도구: PowerShell

예 1: 기존 자격 증명 풀 목록을 검색합니다.

```
Get-CGIIdentityPoolList
```

출력:

```
IdentityPoolId
 IdentityPoolName
--------------
 ----------------
us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1                    CommonTests1
us-east-1:118d242d-204e-4b88-b803-EXAMPLEGUID2                    Tests2
us-east-1:15d49393-ab16-431a-b26e-EXAMPLEGUID3                    CommonTests13
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 ListIdentityPools.

Swift

SDK for Swift

> **Note**
>
> 이 사전 릴리스 설명서는 평가판 버전 SDK에 관한 것입니다. 내용은 변경될 수 있습니다.

> **Note**
>
> 자세한 내용은 다음과 같습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

이름이 지정된 자격 증명 풀의 ID를 찾습니다.

```
    /// Return the ID of the identity pool with the specified name.
    ///
    /// - Parameters:
    ///   - name: The name of the identity pool whose ID should be returned.
    ///
    /// - Returns: A string containing the ID of the specified identity pool
    ///   or `nil` on error or if not found.
    ///
    func getIdentityPoolID(name: String) async throws -> String? {
        var token: String? = nil

        // Iterate over the identity pools until a match is found.

        repeat {
            /// `token` is a value returned by `ListIdentityPools()` if the
            /// returned list of identity pools is only a partial list. You
            /// use the `token` to tell Amazon Cognito that you want to
            /// continue where you left off previously. If you specify `nil`
            /// or you don't provide the token, Amazon Cognito will start at
            /// the beginning.

            let listPoolsInput = ListIdentityPoolsInput(maxResults: 25,
nextToken: token)
```

```
            /// Read pages of identity pools from Cognito until one is found
            /// whose name matches the one specified in the `name` parameter.
            /// Return the matching pool's ID. Each time we ask for the next
            /// page of identity pools, we pass in the token given by the
            /// previous page.

            let output = try await cognitoIdentityClient.listIdentityPools(input:
listPoolsInput)

            if let identityPools = output.identityPools {
                for pool in identityPools {
                    if pool.identityPoolName == name {
                        return pool.identityPoolId!
                    }
                }
            }

            token = output.nextToken
        } while token != nil

        return nil
    }
```

기존 자격 증명 풀의 ID를 가져오거나 존재하지 않으면 생성합니다.

```
    /// Return the ID of the identity pool with the specified name.
    ///
    /// - Parameters:
    ///   - name: The name of the identity pool whose ID should be returned
    ///
    /// - Returns: A string containing the ID of the specified identity pool.
    ///   Returns `nil` if there's an error or if the pool isn't found.
    ///
    public func getOrCreateIdentityPoolID(name: String) async throws -> String? {
        // See if the pool already exists. If it doesn't, create it.

        guard let poolId = try await self.getIdentityPoolID(name: name) else {
            return try await self.createIdentityPool(name: name)
        }

        return poolId
    }
```

- 자세한 내용은 AWS SDK for Swift 개발자 안내서를 참조하십시오.
- API 세부 정보는 Swift APIAWS SDK 참조를 참조하십시오 ListIdentityPools.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. 이 서비스를 SDK와 함께 사용 AWS 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **SetIdentityPoolRoles** CLI와 함께 사용

다음 코드 예제는 SetIdentityPoolRoles의 사용 방법을 보여줍니다.

CLI

AWS CLI

자격 증명 풀 역할을 설정하려면

다음 set-identity-pool-roles 예에서는 자격 증명 풀 역할을 설정합니다.

```
aws cognito-identity set-identity-pool-roles \
    --identity-pool-id "us-west-2:11111111-1111-1111-1111-111111111111" \
    --roles authenticated="arn:aws:iam::111111111111:role/
Cognito_MyIdentityPoolAuth_Role"
```

- API 세부 정보는 AWS CLI 명령 SetIdentityPoolRoles참조를 참조하십시오.

PowerShell

도구: PowerShell

예 1: 인증되지 않은 IAM 역할을 갖도록 특정 자격 증명 풀을 구성합니다.

```
Set-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -Role @{ "unauthenticated" = "arn:aws:iam::123456789012:role/
CommonTests1Role" }
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오. SetIdentityPoolRolesAWS Tools for PowerShell

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. 이 서비스를 SDK와 함께 사용 AWS 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **UpdateIdentityPool** CLI와 함께 사용

다음 코드 예제는 UpdateIdentityPool의 사용 방법을 보여줍니다.

CLI

AWS CLI

자격 증명 풀을 업데이트하려면

이 예에서는 자격 증명 풀을 업데이트합니다. 이름을 로 설정합니다 MyIdentityPool. Cognito를 ID 공급자로 추가합니다. 인증되지 않은 ID는 허용하지 않습니다.

명령:

```
aws cognito-identity update-identity-pool --identity-pool-id "us-
west-2:11111111-1111-1111-1111-111111111111" --identity-pool-name
 "MyIdentityPool" --no-allow-unauthenticated-identities --cognito-
identity-providers ProviderName="cognito-idp.us-west-2.amazonaws.com/us-
west-2_111111111",ClientId="3n4b5urk1ft4fl3mg5e62d9ado",ServerSideTokenCheck=false
```

출력:

```
{
  "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
  "IdentityPoolName": "MyIdentityPool",
  "AllowUnauthenticatedIdentities": false,
  "CognitoIdentityProviders": [
      {
          "ProviderName": "cognito-idp.us-west-2.amazonaws.com/us-
west-2_111111111",
          "ClientId": "3n4b5urk1ft4fl3mg5e62d9ado",
          "ServerSideTokenCheck": false
      }
  ]
}
```

- API 세부 정보는 명령 참조를 참조하십시오 UpdateIdentityPool.AWS CLI

PowerShell

도구: PowerShell

예 1: 일부 자격 증명 풀 속성 (이 경우에는 자격 증명 풀의 이름) 을 업데이트합니다.

```
Update-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -IdentityPoolName NewPoolName
```

출력:

```
LoggedAt                        : 8/12/2015 4:53:33 PM
AllowUnauthenticatedIdentities  : False
DeveloperProviderName           :
IdentityPoolId                  : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
IdentityPoolName                : NewPoolName
OpenIdConnectProviderARNs       : {}
SupportedLoginProviders         : {}
ResponseMetadata                : Amazon.Runtime.ResponseMetadata
ContentLength                   : 135
HttpStatusCode                  : OK
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 UpdateIdentityPool.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. 이 서비스를 SDK와 함께 사용 AWS 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

SDK를 사용한 Amazon Cognito 자격 증명의 크로스 서비스 예제 AWS

다음 샘플 애플리케이션은 AWS SDK를 사용하여 Amazon Cognito ID를 다른 애플리케이션과 결합합니다. AWS 서비스각 예제에는 애플리케이션 GitHub 설정 및 실행 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다.

예제

- Amazon Transcribe 앱 구축
- Amazon Textract 탐색기 애플리케이션 생성

Amazon Transcribe 앱 구축

다음 코드 예제에서는 Amazon Transcribe를 사용하여 브라우저에서 음성 녹음을 텍스트로 기록하고 표시하는 방법을 보여줍니다.

JavaScript

JavaScript (v3) 용 SDK

Amazon Transcribe를 사용하여 브라우저에서 음성 녹음을 텍스트로 기록하고 표시하는 앱을 만듭니다. 이 앱은 두 개의 Amazon Simple Storage Service (Amazon S3) 버킷을 사용합니다. 하나는 애플리케이션 코드를 호스팅하고 다른 하나는 트랜스크립션을 저장합니다. 이 앱은 Amazon Cognito 사용자 풀을 사용하여 사용자를 인증합니다. 인증된 사용자는 필요한 서비스에 액세스할 수 있는 AWS Identity and Access Management (IAM) 권한을 가집니다. AWS

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 전체 예제를 참조하십시오. GitHub

이 예시는 AWS SDK for JavaScript v3 개발자 안내서에서도 확인할 수 있습니다.

이 예시에서 사용되는 서비스

- Amazon Cognito 자격 증명
- Amazon S3
- Amazon Transcribe

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오이 서비스를 SDK와 함께 사용 AWS. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

Amazon Textract 탐색기 애플리케이션 생성

다음 코드 예제에서는 대화형 애플리케이션을 통해 Amazon Textract 출력을 탐색하는 방법을 보여줍니다.

JavaScript

JavaScript (v3) 용 SDK

를 사용하여 Amazon AWS SDK for JavaScript Textract를 사용하여 문서 이미지에서 데이터를 추출하고 대화형 웹 페이지에 표시하는 React 애플리케이션을 구축하는 방법을 보여 줍니다. 이 예제는 웹 브라우저에서 실행되며 자격 증명을 위해 인증된 Amazon Cognito 자격 증명이 필요

합니다. 이 애플리케이션은 스토리지로 Amazon Simple Storage Service (Amazon S3)를 사용하고 알림을 위해 Amazon Simple Notification Service(Amazon SNS) 주제를 구독하는 Amazon Simple Queue Service(Amazon SQS) 대기열을 폴링합니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 전체 예제를 참조하십시오. GitHub

이 예시에서 사용되는 서비스

- Amazon Cognito 자격 증명
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오이 서비스를 SDK와 함께 사용 AWS. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

SDK를 사용하는 Amazon Cognito 자격 증명 공급자의 코드 예제 AWS

다음 코드 예제는 AWS 소프트웨어 개발 키트 (SDK) 와 함께 Amazon Cognito 자격 증명 공급자를 사용하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

AWS SDK 개발자 안내서의 전체 목록과 코드 예제는 을 참조하십시오. 이 서비스를 SDK와 함께 사용 AWS 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

시작하기

Hello Amazon Cognito

다음 코드 예제에서는 Amazon Cognito 사용을 시작하는 방법을 보여줍니다.

C++

SDK for C++

> **Note**
>
> 자세한 내용은 에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

C MakeLists .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 cognito-idp)

# Set this project's name.
project("hello_cognito")

# 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_cognito.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

hello_cognito.cpp 소스 파일의 코드입니다.

```
#include <aws/core/Aws.h>
#include <aws/cognito-idp/CognitoIdentityProviderClient.h>
#include <aws/cognito-idp/model/ListUserPoolsRequest.h>
#include <iostream>

/*
 *  A "Hello Cognito" starter application which initializes an Amazon Cognito
 client and lists the Amazon Cognito
 *  user pools.
 *
 *  main function
 *
 *  Usage: 'hello_cognito'
 *
 */

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;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
```

```
        // clientConfig.region = "us-east-1";

        Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
cognitoClient(clientConfig);

        Aws::String nextToken; // Used for pagination.
        std::vector<Aws::String> userPools;

        do {
            Aws::CognitoIdentityProvider::Model::ListUserPoolsRequest
listUserPoolsRequest;
            if (!nextToken.empty()) {
                listUserPoolsRequest.SetNextToken(nextToken);
            }

            Aws::CognitoIdentityProvider::Model::ListUserPoolsOutcome
listUserPoolsOutcome =
                    cognitoClient.ListUserPools(listUserPoolsRequest);

            if (listUserPoolsOutcome.IsSuccess()) {
                for (auto &userPool:
listUserPoolsOutcome.GetResult().GetUserPools()) {

                    userPools.push_back(userPool.GetName());
                }

                nextToken = listUserPoolsOutcome.GetResult().GetNextToken();
            } else {
                std::cerr << "ListUserPools error: " <<
listUserPoolsOutcome.GetError().GetMessage() << std::endl;
                result = 1;
                break;
            }


        } while (!nextToken.empty());
        std::cout << userPools.size() << " user pools found." << std::endl;
        for (auto &userPool: userPools) {
            std::cout << "   user pool: " << userPool << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
```

```
}
```

- API에 대한 자세한 내용은 API 레퍼런스를 참조하십시오 ListUserPools.AWS SDK for C++

Go

SDK for Go V2

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
package main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics 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
 }
```

```
 cognitoClient := cognitoidentityprovider.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the user pools for your account.")
 var pools []types.UserPoolDescriptionType
 paginator := cognitoidentityprovider.NewListUserPoolsPaginator(
  cognitoClient, &cognitoidentityprovider.ListUserPoolsInput{MaxResults:
 aws.Int32(10)})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(context.TODO())
  if err != nil {
   log.Printf("Couldn't get user pools. Here's why: %v\n", err)
  } else {
   pools = append(pools, output.UserPools...)
  }
 }
 if len(pools) == 0 {
  fmt.Println("You don't have any user pools!")
 } else {
  for _, pool := range pools {
   fmt.Printf("\t%v: %v\n", *pool.Name, *pool.Id)
  }
 }
}
```

- API 세부 정보는 AWS SDK for Go API ListUserPools참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
```

```
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;

/**
 * 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 ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllUserPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listAllUserPools(CognitoIdentityProviderClient
 cognitoClient) {
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                    .maxResults(10)
                    .build();

            ListUserPoolsResponse response =
 cognitoClient.listUserPools(request);
            response.userPools().forEach(userpool -> {
                System.out.println("User pool " + userpool.name() + ", User ID "
 + userpool.id());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
}
```

- API 세부 정보는 AWS SDK for Java 2.x API ListUserPools참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import {
  paginateListUserPools,
  CognitoIdentityProviderClient,
} from "@aws-sdk/client-cognito-identity-provider";

const client = new CognitoIdentityProviderClient({});

export const helloCognito = async () => {
  const paginator = paginateListUserPools({ client }, {});

  const userPoolNames = [];

  for await (const page of paginator) {
    const names = page.UserPools.map((pool) => pool.Name);
    userPoolNames.push(...names);
  }

  console.log("User pool names: ");
  console.log(userPoolNames.join("\n"));
  return userPoolNames;
};
```

- API 세부 정보는 AWS SDK for JavaScript API ListUserPools참조를 참조하십시오.

코드 예시

- SDK를 사용하는 Amazon Cognito 자격 증명 공급자를 위한 작업 AWS
  - AWS SDK 또는 AdminCreateUser CLI와 함께 사용
  - AWS SDK 또는 AdminGetUser CLI와 함께 사용
  - AWS SDK 또는 AdminInitiateAuth CLI와 함께 사용
  - AWS SDK 또는 AdminRespondToAuthChallenge CLI와 함께 사용
  - AWS SDK 또는 AdminSetUserPassword CLI와 함께 사용
  - AWS SDK 또는 AssociateSoftwareToken CLI와 함께 사용
  - AWS SDK 또는 ConfirmDevice CLI와 함께 사용
  - AWS SDK 또는 ConfirmForgotPassword CLI와 함께 사용
  - AWS SDK 또는 ConfirmSignUp CLI와 함께 사용
  - AWS SDK 또는 CreateUserPool CLI와 함께 사용
  - AWS SDK 또는 CreateUserPoolClient CLI와 함께 사용
  - AWS SDK 또는 DeleteUser CLI와 함께 사용
  - AWS SDK 또는 ForgotPassword CLI와 함께 사용
  - AWS SDK 또는 InitiateAuth CLI와 함께 사용
  - AWS SDK 또는 ListUserPools CLI와 함께 사용
  - AWS SDK 또는 ListUsers CLI와 함께 사용
  - AWS SDK 또는 ResendConfirmationCode CLI와 함께 사용
  - AWS SDK 또는 RespondToAuthChallenge CLI와 함께 사용
  - AWS SDK 또는 SignUp CLI와 함께 사용
  - AWS SDK 또는 UpdateUserPool CLI와 함께 사용
  - AWS SDK 또는 VerifySoftwareToken CLI와 함께 사용
- SDK를 사용하는 Amazon Cognito 자격 증명 공급자의 시나리오 AWS
  - SDK를 사용하여 Lambda 함수를 사용하여 알려진 Amazon Cognito 사용자를 자동으로 확인합니다. AWS
  - SDK를 사용하여 Lambda 함수를 사용하여 알려진 Amazon Cognito 사용자를 자동으로 마이그레이션합니다. AWS
  - SDK를 사용하여 MFA가 필요한 Amazon Cognito 사용자 풀에 사용자를 등록합니다. AWS
  - SDK를 사용한 Amazon Cognito 사용자 인증 후 Lambda 함수를 사용하여 사용자 지정 활동 데이터를 작성합니다. AWS

SDK를 사용하는 Amazon Cognito 자격 증명 공급자를 위한 작업 AWS

다음 코드 예제는 SDK를 사용하여 개별 Amazon Cognito 자격 증명 공급자 작업을 수행하는 방법을 보여줍니다. AWS 이들 발췌문은 Amazon Cognito Identity Provider API를 직접적으로 호출하며, 컨텍스트에서 실행되어야 하는 더 큰 프로그램에서 발췌한 코드입니다. 각 예제에는 코드 GitHub 설정 및 실행 지침을 찾을 수 있는 링크가 포함되어 있습니다.

다음 예제에는 가장 일반적으로 사용되는 작업만 포함되어 있습니다. 전체 목록은 Amazon Cognito Identity Provider API Reference(Amazon Cognito 자격 증명 공급자 API 참조)를 참조하세요.

예제

- AWS SDK 또는 AdminCreateUser CLI와 함께 사용
- AWS SDK 또는 AdminGetUser CLI와 함께 사용
- AWS SDK 또는 AdminInitiateAuth CLI와 함께 사용
- AWS SDK 또는 AdminRespondToAuthChallenge CLI와 함께 사용
- AWS SDK 또는 AdminSetUserPassword CLI와 함께 사용
- AWS SDK 또는 AssociateSoftwareToken CLI와 함께 사용
- AWS SDK 또는 ConfirmDevice CLI와 함께 사용
- AWS SDK 또는 ConfirmForgotPassword CLI와 함께 사용
- AWS SDK 또는 ConfirmSignUp CLI와 함께 사용
- AWS SDK 또는 CreateUserPool CLI와 함께 사용
- AWS SDK 또는 CreateUserPoolClient CLI와 함께 사용
- AWS SDK 또는 DeleteUser CLI와 함께 사용
- AWS SDK 또는 ForgotPassword CLI와 함께 사용
- AWS SDK 또는 InitiateAuth CLI와 함께 사용
- AWS SDK 또는 ListUserPools CLI와 함께 사용
- AWS SDK 또는 ListUsers CLI와 함께 사용
- AWS SDK 또는 ResendConfirmationCode CLI와 함께 사용
- AWS SDK 또는 RespondToAuthChallenge CLI와 함께 사용
- AWS SDK 또는 SignUp CLI와 함께 사용
- AWS SDK 또는 UpdateUserPool CLI와 함께 사용

- AWS SDK 또는 VerifySoftwareToken CLI와 함께 사용

AWS SDK 또는 **AdminCreateUser** CLI와 함께 사용

다음 코드 예제는 AdminCreateUser의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- Amazon Cognito 사용자 인증 후 Lambda 함수를 사용하여 사용자 지정 활동 데이터 작성

CLI

AWS CLI

사용자를 만들려면

다음 admin-create-user 예제에서는 지정된 설정의 이메일 주소 및 전화번호를 사용하여 사용자를 생성합니다.

```
aws cognito-idp admin-create-user \
    --user-pool-id us-west-2_aaaaaaaaa \
    --username diego \
    --user-attributes Name=email,Value=diego@example.com
 Name=phone_number,Value="+15555551212" \
    --message-action SUPPRESS
```

출력:

```
{
    "User": {
        "Username": "diego",
        "Attributes": [
            {
                "Name": "sub",
                "Value": "7325c1de-b05b-4f84-b321-9adc6e61f4a2"
            },
            {
                "Name": "phone_number",
                "Value": "+15555551212"
            },
            {
```

```
                "Name": "email",
                "Value": "diego@example.com"
            }
        ],
        "UserCreateDate": 1548099495.428,
        "UserLastModifiedDate": 1548099495.428,
        "Enabled": true,
        "UserStatus": "FORCE_CHANGE_PASSWORD"
    }
}
```

- API 세부 정보는 AWS CLI 명령 AdminCreateUser참조를 참조하십시오.

Go

SDK for Go V2

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(userPoolId string, userName string,
 userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(context.TODO(),
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
```

```
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}
```

- API 세부 정보는 AWS SDK for Go API AdminCreateUser참조를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오이 서비스를 SDK와 함께 사용 AWS. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **AdminGetUser** CLI와 함께 사용

다음 코드 예제는 AdminGetUser의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- MFA가 필요한 사용자 풀에 사용자 가입시키기

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
```

```
    /// Get the specified user from an Amazon Cognito user pool with
administrator access.
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="poolId">The Id of the Amazon Cognito user pool.</param>
    /// <returns>Async task.</returns>
    public async Task<UserStatusType> GetAdminUserAsync(string userName, string
poolId)
    {
        AdminGetUserRequest userRequest = new AdminGetUserRequest
        {
            Username = userName,
            UserPoolId = poolId,
        };

        var response = await _cognitoService.AdminGetUserAsync(userRequest);

        Console.WriteLine($"User status {response.UserStatus}");
        return response.UserStatus;
    }
```

- API 세부 정보는 AWS SDK for .NET API AdminGetUser참조를 참조하십시오.

C++

SDK for C++

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);
```

```
    Aws::CognitoIdentityProvider::Model::AdminGetUserRequest request;
    request.SetUsername(userName);
    request.SetUserPoolId(userPoolID);

    Aws::CognitoIdentityProvider::Model::AdminGetUserOutcome outcome =
            client.AdminGetUser(request);

    if (outcome.IsSuccess()) {
        std::cout << "The status for " << userName << " is " <<

 Aws::CognitoIdentityProvider::Model::UserStatusTypeMapper::GetNameForUserStatusType(
                          outcome.GetResult().GetUserStatus()) << std::endl;
        std::cout << "Enabled is " << outcome.GetResult().GetEnabled() <<
std::endl;
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminGetUser. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

- API 세부 정보는 AWS SDK for C++ API [AdminGetUser](#)참조를 참조하십시오.

CLI

AWS CLI

사용자 가져오기

이 예시에서는 사용자 이름 jane@example.com에 대한 정보를 가져옵니다.

명령:

```
aws cognito-idp admin-get-user --user-pool-id us-west-2_aaaaaaaaa --username
 jane@example.com
```

출력:

```
{
  "Username": "4320de44-2322-4620-999b-5e2e1c8df013",
  "Enabled": true,
  "UserStatus": "FORCE_CHANGE_PASSWORD",
```

```
  "UserCreateDate": 1548108509.537,
  "UserAttributes": [
      {
          "Name": "sub",
          "Value": "4320de44-2322-4620-999b-5e2e1c8df013"
      },
      {
          "Name": "email_verified",
          "Value": "true"
      },
      {
          "Name": "phone_number_verified",
          "Value": "true"
      },
      {
          "Name": "phone_number",
          "Value": "+01115551212"
      },
      {
          "Name": "email",
          "Value": "jane@example.com"
      }
  ],
  "UserLastModifiedDate": 1548108509.537
}
```

- API 세부 정보는 AWS CLI 명령 AdminGetUser참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   public static void getAdminUser(CognitoIdentityProviderClient
identityProviderClient, String userName,
           String poolId) {
       try {
```

```
            AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
                    .username(userName)
                    .userPoolId(poolId)
                    .build();

            AdminGetUserResponse response =
 identityProviderClient.adminGetUser(userRequest);
            System.out.println("User status " + response.userStatusAsString());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API AdminGetUser참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const adminGetUser = ({ userPoolId, username }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new AdminGetUserCommand({
    UserPoolId: userPoolId,
    Username: username,
  });

  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API AdminGetUser참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getAdminUser(userNameVal: String?, poolIdVal: String?) {
    val userRequest = AdminGetUserRequest {
        username = userNameVal
        userPoolId = poolIdVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminGetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 AdminGetUser.

Python

SDK for Python(Boto3)

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""
```

```
   def __init__(self, cognito_idp_client, user_pool_id, client_id,
client_secret=None):
       """
       :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
client.
       :param user_pool_id: The ID of an existing Amazon Cognito user pool.
       :param client_id: The ID of a client application registered with the user
pool.
       :param client_secret: The client secret, if the client has a secret.
       """
       self.cognito_idp_client = cognito_idp_client
       self.user_pool_id = user_pool_id
       self.client_id = client_id
       self.client_secret = client_secret


   def sign_up_user(self, user_name, password, user_email):
       """
       Signs up a new user with Amazon Cognito. This action prompts Amazon
Cognito
       to send an email to the specified email address. The email contains a
code that
       can be used to confirm the user.

       When the user already exists, the user status is checked to determine
whether
       the user has been confirmed.

       :param user_name: The user name that identifies the new user.
       :param password: The password for the new user.
       :param user_email: The email address for the new user.
       :return: True when the user is already confirmed with Amazon Cognito.
                Otherwise, false.
       """
       try:
           kwargs = {
               "ClientId": self.client_id,
               "Username": user_name,
               "Password": password,
               "UserAttributes": [{"Name": "email", "Value": user_email}],
           }
           if self.client_secret is not None:
               kwargs["SecretHash"] = self._secret_hash(user_name)
           response = self.cognito_idp_client.sign_up(**kwargs)
```

```
            confirmed = response["UserConfirmed"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "UsernameExistsException":
                response = self.cognito_idp_client.admin_get_user(
                    UserPoolId=self.user_pool_id, Username=user_name
                )
                logger.warning(
                    "User %s exists and is %s.", user_name,
response["UserStatus"]
                )
                confirmed = response["UserStatus"] == "CONFIRMED"
            else:
                logger.error(
                    "Couldn't sign up %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return confirmed
```

- API에 대한 자세한 내용은 파이썬용AWS SDK (Boto3) API 레퍼런스를 참조하십시오 AdminGetUser.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. 이 서비스를 SDK와 함께 사용 AWS 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **AdminInitiateAuth** CLI와 함께 사용

다음 코드 예제는 AdminInitiateAuth의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- MFA가 필요한 사용자 풀에 사용자 가입시키기

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Initiate an admin auth request.
    /// </summary>
    /// <param name="clientId">The client ID to use.</param>
    /// <param name="userPoolId">The ID of the user pool.</param>
    /// <param name="userName">The username to authenticate.</param>
    /// <param name="password">The user's password.</param>
    /// <returns>The session to use in challenge-response.</returns>
    public async Task<string> AdminInitiateAuthAsync(string clientId, string
userPoolId, string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var request = new AdminInitiateAuthRequest
        {
            ClientId = clientId,
            UserPoolId = userPoolId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.AdminInitiateAuthAsync(request);
        return response.Session;
    }
```

- API 세부 정보는 AWS SDK for .NET API AdminInitiateAuth참조를 참조하십시오.

C++

SDK for C++

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthRequest request;
    request.SetClientId(clientID);
    request.SetUserPoolId(userPoolID);
    request.AddAuthParameters("USERNAME", userName);
    request.AddAuthParameters("PASSWORD", password);
    request.SetAuthFlow(

Aws::CognitoIdentityProvider::Model::AuthFlowType::ADMIN_USER_PASSWORD_AUTH);


    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthOutcome outcome =
            client.AdminInitiateAuth(request);

    if (outcome.IsSuccess()) {
        std::cout << "Call to AdminInitiateAuth was successful." << std::endl;
        sessionResult = outcome.GetResult().GetSession();
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminInitiateAuth. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

- API 세부 정보는 AWS SDK for C++ API AdminInitiateAuth참조를 참조하십시오.

CLI

AWS CLI

권한 부여 시작

이 예시에서는 사용자 이름 jane@example.com에 대해 ADMIN_NO_SRP_AUTH 흐름을 사용하여 권한 부여를 시작합니다.

클라이언트에는 서버 기반 인증을 위한 로그인 API(ADMIN_NO_SRP_AUTH)가 활성화되어 있어야 합니다.

반환 값의 세션 정보를 사용하여 admin-respond-to-auth -hallenge를 호출합니다.

명령:

```
aws cognito-idp admin-initiate-auth --user-pool-id us-west-2_aaaaaaaaa --client-
id 3n4b5urk1ft4fl3mg5e62d9ado --auth-flow ADMIN_NO_SRP_AUTH --auth-parameters
 USERNAME=jane@example.com,PASSWORD=password
```

출력:

```
{
  "ChallengeName": "NEW_PASSWORD_REQUIRED",
  "Session": "SESSION",
  "ChallengeParameters": {
      "USER_ID_FOR_SRP": "84514837-dcbc-4af1-abff-f3c109334894",
      "requiredAttributes": "[]",
      "userAttributes": "{\"email_verified\":\"true\",\"phone_number_verified\":
\"true\",\"phone_number\":\"+01xxx5550100\",\"email\":\"jane@example.com\"}"
  }
}
```

- API 세부 정보는 AWS CLI 명령 AdminInitiateAuth참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
           String clientId, String userName, String password, String userPoolId)
{
       try {
           Map<String, String> authParameters = new HashMap<>();
           authParameters.put("USERNAME", userName);
           authParameters.put("PASSWORD", password);

           AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
                   .clientId(clientId)
                   .userPoolId(userPoolId)
                   .authParameters(authParameters)
                   .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                   .build();

           AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
           System.out.println("Result Challenge is : " +
response.challengeName());
           return response;

       } catch (CognitoIdentityProviderException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return null;
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API AdminInitiateAuth참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const adminInitiateAuth = ({ clientId, userPoolId, username, password }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new AdminInitiateAuthCommand({
    ClientId: clientId,
    UserPoolId: userPoolId,
    AuthFlow: AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
    AuthParameters: { USERNAME: username, PASSWORD: password },
  });

  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API AdminInitiateAuth참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun checkAuthMethod(clientIdVal: String, userNameVal: String,
 passwordVal: String, userPoolIdVal: String): AdminInitiateAuthResponse {
    val authParas = mutableMapOf<String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal

    val authRequest = AdminInitiateAuthRequest {
        clientId = clientIdVal
        userPoolId = userPoolIdVal
        authParameters = authParas
        authFlow = AuthFlowType.AdminUserPasswordAuth
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminInitiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 AdminInitiateAuth.

Python

SDK for Python(Boto3)

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
```

```
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def start_sign_in(self, user_name, password):
        """
        Starts the sign-in process for a user by using administrator credentials.
        This method of signing in is appropriate for code running on a secure
 server.

        If the user pool is configured to require MFA and this is the first sign-
in
        for the user, Amazon Cognito returns a challenge response to set up an
        MFA application. When this occurs, this function gets an MFA secret from
        Amazon Cognito and returns it to the caller.

        :param user_name: The name of the user to sign in.
        :param password: The user's password.
        :return: The result of the sign-in attempt. When sign-in is successful,
 this
                 returns an access token that can be used to get AWS credentials.
 Otherwise,
                 Amazon Cognito returns a challenge to set up an MFA application,
                 or a challenge to enter an MFA code from a registered MFA
 application.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "AuthFlow": "ADMIN_USER_PASSWORD_AUTH",
                "AuthParameters": {"USERNAME": user_name, "PASSWORD": password},
            }
            if self.client_secret is not None:
```

```
                kwargs["AuthParameters"]["SECRET_HASH"] =
self._secret_hash(user_name)
            response = self.cognito_idp_client.admin_initiate_auth(**kwargs)
            challenge_name = response.get("ChallengeName", None)
            if challenge_name == "MFA_SETUP":
                if (
                    "SOFTWARE_TOKEN_MFA"
                    in response["ChallengeParameters"]["MFAS_CAN_SETUP"]
                ):
                    response.update(self.get_mfa_secret(response["Session"]))
                else:
                    raise RuntimeError(
                        "The user pool requires MFA setup, but the user pool is
not "
                        "configured for TOTP MFA. This example requires TOTP
MFA."
                    )
        except ClientError as err:
            logger.error(
                "Couldn't start sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response
```

- API에 대한 자세한 내용은 파이썬용AWS SDK (Boto3) API 레퍼런스를 참조하십시오 AdminInitiateAuth.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. 이 서비스를 SDK와 함께 사용 AWS 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **AdminRespondToAuthChallenge** CLI와 함께 사용

다음 코드 예제는 AdminRespondToAuthChallenge의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- MFA가 필요한 사용자 풀에 사용자 가입시키기

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Respond to an admin authentication challenge.
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="clientId">The client ID.</param>
    /// <param name="mfaCode">The multi-factor authentication code.</param>
    /// <param name="session">The current application session.</param>
    /// <param name="clientId">The user pool ID.</param>
    /// <returns>The result of the authentication response.</returns>
    public async Task<AuthenticationResultType> AdminRespondToAuthChallengeAsync(
        string userName,
        string clientId,
        string mfaCode,
        string session,
        string userPoolId)
    {
        Console.WriteLine("SOFTWARE_TOKEN_MFA challenge is generated");

        var challengeResponses = new Dictionary<string, string>();
        challengeResponses.Add("USERNAME", userName);
        challengeResponses.Add("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

        var respondToAuthChallengeRequest = new
AdminRespondToAuthChallengeRequest
        {
            ChallengeName = ChallengeNameType.SOFTWARE_TOKEN_MFA,
```

```
            ClientId = clientId,
            ChallengeResponses = challengeResponses,
            Session = session,
            UserPoolId = userPoolId,
        };

        var response = await
_cognitoService.AdminRespondToAuthChallengeAsync(respondToAuthChallengeRequest);
        Console.WriteLine($"Response to Authentication
{response.AuthenticationResult.TokenType}");
        return response.AuthenticationResult;
    }
```

- API 세부 정보는 AWS SDK for .NET API AdminRespondToAuthChallenge참조를 참조하십시오.

C++

SDK for C++

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeRequest
request;
        request.AddChallengeResponses("USERNAME", userName);
        request.AddChallengeResponses("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
        request.SetChallengeName(
```

```
Aws::CognitoIdentityProvider::Model::ChallengeNameType::SOFTWARE_TOKEN_MFA);
        request.SetClientId(clientID);
        request.SetUserPoolId(userPoolID);
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeOutcome
outcome =
                client.AdminRespondToAuthChallenge(request);

        if (outcome.IsSuccess()) {
            std::cout << "Here is the response to the challenge.\n" <<

outcome.GetResult().GetAuthenticationResult().Jsonize().View().WriteReadable()
                      << std::endl;

            accessToken =
outcome.GetResult().GetAuthenticationResult().GetAccessToken();
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::AdminRespondToAuthChallenge. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- API 세부 정보는 AWS SDK for C++ API AdminRespondToAuthChallenge참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   // Respond to an authentication challenge.
   public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
           String userName, String clientId, String mfaCode, String session) {
       System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
       Map<String, String> challengeResponses = new HashMap<>();

       challengeResponses.put("USERNAME", userName);
       challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

       AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
               .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
               .clientId(clientId)
               .challengeResponses(challengeResponses)
               .session(session)
               .build();

       AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
               .adminRespondToAuthChallenge(respondToAuthChallengeRequest);

System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
               + respondToAuthChallengeResult.authenticationResult());
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API AdminRespondToAuthChallenge참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const adminRespondToAuthChallenge = ({
```

```
  userPoolId,
  clientId,
  username,
  totp,
  session,
}) => {
  const client = new CognitoIdentityProviderClient({});
  const command = new AdminRespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: totp,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API AdminRespondToAuthChallenge참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(userName: String, clientIdVal: String?,
 mfaCode: String, sessionVal: String?) {
    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponsesOb = mutableMapOf<String, String>()
    challengeResponsesOb["USERNAME"] = userName
```

```
    challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest = AdminRespondToAuthChallengeRequest {
        challengeName = ChallengeNameType.SoftwareTokenMfa
        clientId = clientIdVal
        challengeResponses = challengeResponsesOb
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val respondToAuthChallengeResult =
 identityProviderClient.adminRespondToAuthChallenge(adminRespondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult()
 ${respondToAuthChallengeResult.authenticationResult}")
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 AdminRespondToAuthChallenge.

Python

SDK for Python(Boto3)

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

연결된 MFA 애플리케이션에서 생성한 코드를 제공하여 MFA 문제에 응답하세요.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
```

```
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def respond_to_mfa_challenge(self, user_name, session, mfa_code):
        """
        Responds to a challenge for an MFA code. This completes the second step
of
        a two-factor sign-in. When sign-in is successful, it returns an access
token
        that can be used to get AWS credentials from Amazon Cognito.

        :param user_name: The name of the user who is signing in.
        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :param mfa_code: A code generated by the associated MFA application.
        :return: The result of the authentication. When successful, this contains
an
                 access token for the user.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "ChallengeName": "SOFTWARE_TOKEN_MFA",
                "Session": session,
                "ChallengeResponses": {
                    "USERNAME": user_name,
                    "SOFTWARE_TOKEN_MFA_CODE": mfa_code,
                },
            }
            if self.client_secret is not None:
                kwargs["ChallengeResponses"]["SECRET_HASH"] = self._secret_hash(
                    user_name
                )
```

```
            response =
self.cognito_idp_client.admin_respond_to_auth_challenge(**kwargs)
            auth_result = response["AuthenticationResult"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ExpiredCodeException":
                logger.warning(
                    "Your MFA code has expired or has been used already. You
might have "
                    "to wait a few seconds until your app shows you a new code."
                )
            else:
                logger.error(
                    "Couldn't respond to mfa challenge for %s. Here's why: %s:
%s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return auth_result
```

- API에 대한 자세한 내용은 파이썬용AWS SDK (Boto3) API 레퍼런스를 참조하십시오 AdminRespondToAuthChallenge.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. 이 서비스를 SDK와 함께 사용 AWS 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **AdminSetUserPassword** CLI와 함께 사용

다음 코드 예시에서는 AdminSetUserPassword을 사용하는 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- Amazon Cognito 사용자 인증 후 Lambda 함수를 사용하여 사용자 지정 활동 데이터 작성

Go

SDK for Go V2

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(userPoolId string, userName
 string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(context.TODO(),
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

- API 세부 정보는 AWS SDK for Go API AdminSetUserPassword참조를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오이 서비스를 SDK와 함께 사용 AWS. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **AssociateSoftwareToken** CLI와 함께 사용

다음 코드 예제는 AssociateSoftwareToken의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- MFA가 필요한 사용자 풀에 사용자 가입시키기

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Get an MFA token to authenticate the user with the authenticator.
    /// </summary>
    /// <param name="session">The session name.</param>
    /// <returns>The session name.</returns>
    public async Task<string> AssociateSoftwareTokenAsync(string session)
    {
        var softwareTokenRequest = new AssociateSoftwareTokenRequest
        {
            Session = session,
        };

        var tokenResponse = await
_cognitoService.AssociateSoftwareTokenAsync(softwareTokenRequest);
        var secretCode = tokenResponse.SecretCode;
```

```
        Console.WriteLine($"Use the following secret code to set up the
authenticator: {secretCode}");

        return tokenResponse.Session;
    }
```

- API 세부 정보는 AWS SDK for .NET API AssociateSoftwareToken참조를 참조하십시오.

C++

SDK for C++

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenRequest
request;
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenOutcome
outcome =
                client.AssociateSoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "Enter this setup key into an authenticator app, for
example Google Authenticator."
                    << std::endl;
            std::cout << "Setup key: " << outcome.GetResult().GetSecretCode()
                      << std::endl;
```

```
#ifdef USING_QR
            printAsterisksLine();
            std::cout << "\nOr scan the QR code in the file '" << QR_CODE_PATH <<
 "."
                      << std::endl;

            saveQRCode(std::string("otpauth://totp/") + userName + "?secret=" +
                       outcome.GetResult().GetSecretCode());
#endif // USING_QR
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with
 CognitoIdentityProvider::AssociateSoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- API 세부 정보는 AWS SDK for C++ API AssociateSoftwareToken참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
        AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
                .session(session)
                .build();

        AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
                .associateSoftwareToken(softwareTokenRequest);
        String secretCode = tokenResponse.secretCode();
```

```
        System.out.println("Enter this token into Google Authenticator");
        System.out.println(secretCode);
        return tokenResponse.session();
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API AssociateSoftwareToken참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const associateSoftwareToken = (session) => {
  const client = new CognitoIdentityProviderClient({});
  const command = new AssociateSoftwareTokenCommand({
    Session: session,
  });

  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API AssociateSoftwareToken참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest = AssociateSoftwareTokenRequest {
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val tokenResponse =
 identityProviderClient.associateSoftwareToken(softwareTokenRequest)
        val secretCode = tokenResponse.secretCode
        println("Enter this token into Google Authenticator")
        println(secretCode)
        return tokenResponse.session
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 AssociateSoftwareToken.

Python

SDK for Python(Boto3)

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
```

```
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def get_mfa_secret(self, session):
        """
        Gets a token that can be used to associate an MFA application with the
user.

        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :return: An MFA token that can be used to set up an MFA application.
        """
        try:
            response =
self.cognito_idp_client.associate_software_token(Session=session)
        except ClientError as err:
            logger.error(
                "Couldn't get MFA secret. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response
```

- API에 대한 자세한 내용은 파이썬용AWS SDK (Boto3) API 레퍼런스를 참조하십시오 AssociateSoftwareToken.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. 이 서비스를 SDK와 함께 사용 AWS 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **ConfirmDevice** CLI와 함께 사용

다음 코드 예제는 ConfirmDevice의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- MFA가 필요한 사용자 풀에 사용자 가입시키기

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Initiates and confirms tracking of the device.
    /// </summary>
    /// <param name="accessToken">The user's access token.</param>
    /// <param name="deviceKey">The key of the device from Amazon Cognito.</
param>
    /// <param name="deviceName">The device name.</param>
    /// <returns></returns>
    public async Task<bool> ConfirmDeviceAsync(string accessToken, string
 deviceKey, string deviceName)
    {
        var request = new ConfirmDeviceRequest
        {
            AccessToken = accessToken,
            DeviceKey = deviceKey,
            DeviceName = deviceName
        };

        var response = await _cognitoService.ConfirmDeviceAsync(request);
        return response.UserConfirmationNecessary;
    }
```

- API 세부 정보는 AWS SDK for .NET API ConfirmDevice참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const confirmDevice = ({ deviceKey, accessToken, passwordVerifier, salt }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ConfirmDeviceCommand({
    DeviceKey: deviceKey,
    AccessToken: accessToken,
    DeviceSecretVerifierConfig: {
      PasswordVerifier: passwordVerifier,
      Salt: salt,
    },
  });

  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API ConfirmDevice참조를 참조하십시오.

Python

SDK for Python(Boto3)

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""
```

```
   def __init__(self, cognito_idp_client, user_pool_id, client_id,
client_secret=None):
       """
       :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
client.
       :param user_pool_id: The ID of an existing Amazon Cognito user pool.
       :param client_id: The ID of a client application registered with the user
pool.
       :param client_secret: The client secret, if the client has a secret.
       """
       self.cognito_idp_client = cognito_idp_client
       self.user_pool_id = user_pool_id
       self.client_id = client_id
       self.client_secret = client_secret


   def confirm_mfa_device(
       self,
       user_name,
       device_key,
       device_group_key,
       device_password,
       access_token,
       aws_srp,
   ):
       """
       Confirms an MFA device to be tracked by Amazon Cognito. When a device is
       tracked, its key and password can be used to sign in without requiring a
new
       MFA code from the MFA application.

       :param user_name: The user that is associated with the device.
       :param device_key: The key of the device, returned by Amazon Cognito.
       :param device_group_key: The group key of the device, returned by Amazon
Cognito.
       :param device_password: The password that is associated with the device.
       :param access_token: The user's access token.
       :param aws_srp: A class that helps with Secure Remote Password (SRP)
                       calculations. The scenario associated with this example
uses
                       the warrant package.
       :return: True when the user must confirm the device. Otherwise, False.
When
```

```
                False, the device is automatically confirmed and tracked.
        """
        srp_helper = aws_srp.AWSSRP(
            username=user_name,
            password=device_password,
            pool_id="_",
            client_id=self.client_id,
            client_secret=None,
            client=self.cognito_idp_client,
        )
        device_and_pw = f"{device_group_key}{device_key}:{device_password}"
        device_and_pw_hash = aws_srp.hash_sha256(device_and_pw.encode("utf-8"))
        salt = aws_srp.pad_hex(aws_srp.get_random(16))
        x_value = aws_srp.hex_to_long(aws_srp.hex_hash(salt +
device_and_pw_hash))
        verifier = aws_srp.pad_hex(pow(srp_helper.val_g, x_value,
srp_helper.big_n))
        device_secret_verifier_config = {
            "PasswordVerifier": base64.standard_b64encode(
                bytearray.fromhex(verifier)
            ).decode("utf-8"),
            "Salt":
base64.standard_b64encode(bytearray.fromhex(salt)).decode("utf-8"),
        }
        try:
            response = self.cognito_idp_client.confirm_device(
                AccessToken=access_token,
                DeviceKey=device_key,
                DeviceSecretVerifierConfig=device_secret_verifier_config,
            )
            user_confirm = response["UserConfirmationNecessary"]
        except ClientError as err:
            logger.error(
                "Couldn't confirm mfa device %s. Here's why: %s: %s",
                device_key,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return user_confirm
```

- API에 대한 자세한 내용은 파이썬용AWS SDK (Boto3) API 레퍼런스를 참조하십시오 ConfirmDevice.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. 이 서비스를 SDK와 함께 사용 AWS 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **ConfirmForgotPassword** CLI와 함께 사용

다음 코드 예제는 ConfirmForgotPassword의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- Lambda 함수를 사용하여 알려진 사용자를 자동으로 마이그레이션

CLI

AWS CLI

잊어버린 비밀번호를 확인하려면

이 예시는 사용자 이름 diego@example.com 의 비밀번호를 잊어버렸는지 확인합니다.

명령:

```
aws cognito-idp confirm-forgot-password --client-id 3n4b5urk1ft4fl3mg5e62d9ado --
username=diego@example.com --password PASSWORD --confirmation-code CONF_CODE
```

- API 세부 정보는 AWS CLI 명령 ConfirmForgotPassword참조를 참조하십시오.

Go

SDK for Go V2

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(clientId string, code string,
 userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(context.TODO(),
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}
```

- API 세부 정보는 AWS SDK for Go API ConfirmForgotPassword참조를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오이 서비스를 SDK와 함께 사용 AWS. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **ConfirmSignUp** CLI와 함께 사용

다음 코드 예제는 ConfirmSignUp의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- MFA가 필요한 사용자 풀에 사용자 가입시키기

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Confirm that the user has signed up.
    /// </summary>
    /// <param name="clientId">The Id of this application.</param>
    /// <param name="code">The confirmation code sent to the user.</param>
    /// <param name="userName">The username.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ConfirmSignupAsync(string clientId, string code,
string userName)
    {
        var signUpRequest = new ConfirmSignUpRequest
        {
            ClientId = clientId,
            ConfirmationCode = code,
            Username = userName,
        };

        var response = await _cognitoService.ConfirmSignUpAsync(signUpRequest);
        if (response.HttpStatusCode == HttpStatusCode.OK)
        {
            Console.WriteLine($"{userName} was confirmed");
            return true;
        }
        return false;
    }
```

- API 세부 정보는 AWS SDK for .NET API ConfirmSignUp참조를 참조하십시오.

C++

SDK for C++

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::ConfirmSignUpRequest request;
        request.SetClientId(clientID);
        request.SetConfirmationCode(confirmationCode);
        request.SetUsername(userName);

        Aws::CognitoIdentityProvider::Model::ConfirmSignUpOutcome outcome =
                client.ConfirmSignUp(request);

        if (outcome.IsSuccess()) {
            std::cout << "ConfirmSignup was Successful."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::ConfirmSignUp. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- API 세부 정보는 AWS SDK for C++ API ConfirmSignUp참조를 참조하십시오.

CLI

AWS CLI

가입 확인

이 예시는 사용자 이름 diego@example.com 가입을 확인합니다.

명령:

```
aws cognito-idp confirm-sign-up --client-id 3n4b5urk1ft4fl3mg5e62d9ado --
username=diego@example.com --confirmation-code CONF_CODE
```

- API 세부 정보는 AWS CLI 명령 ConfirmSignUp참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
           String userName) {
       try {
           ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
                   .clientId(clientId)
                   .confirmationCode(code)
                   .username(userName)
                   .build();

           identityProviderClient.confirmSignUp(signUpRequest);
           System.out.println(userName + " was confirmed");

       } catch (CognitoIdentityProviderException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
```

```
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API ConfirmSignUp참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const confirmSignUp = ({ clientId, username, code }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ConfirmSignUpCommand({
    ClientId: clientId,
    Username: username,
    ConfirmationCode: code,
  });

  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API ConfirmSignUp참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun confirmSignUp(clientIdVal: String?, codeVal: String?, userNameVal:
 String?) {
    val signUpRequest = ConfirmSignUpRequest {
        clientId = clientIdVal
        confirmationCode = codeVal
        username = userNameVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 ConfirmSignUp.

Python

SDK for Python(Boto3)

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
```

```
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def confirm_user_sign_up(self, user_name, confirmation_code):
        """
        Confirms a previously created user. A user must be confirmed before they
        can sign in to Amazon Cognito.

        :param user_name: The name of the user to confirm.
        :param confirmation_code: The confirmation code sent to the user's
registered
                                  email address.
        :return: True when the confirmation succeeds.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "ConfirmationCode": confirmation_code,
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            self.cognito_idp_client.confirm_sign_up(**kwargs)
        except ClientError as err:
            logger.error(
                "Couldn't confirm sign up for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return True
```

- API에 대한 자세한 내용은 파이썬용AWS SDK (Boto3) API 레퍼런스를 참조하십시오 ConfirmSignUp.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. 이 서비스를 SDK와 함께 사용 AWS 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **CreateUserPool** CLI와 함께 사용

다음 코드 예제는 CreateUserPool의 사용 방법을 보여줍니다.

CLI

AWS CLI

최소 구성 사용자 풀 생성

이 예제에서는 기본값을 MyUserPool 사용하여 이름이 지정된 사용자 풀을 만듭니다. 필수 속성도 없고 애플리케이션 클라이언트도 없습니다. MFA 및 고급 보안이 비활성화되었습니다.

명령:

```
aws cognito-idp create-user-pool --pool-name MyUserPool
```

출력:

```
{
  "UserPool": {
      "SchemaAttributes": [
          {
              "Name": "sub",
              "StringAttributeConstraints": {
                  "MinLength": "1",
                  "MaxLength": "2048"
              },
              "DeveloperOnlyAttribute": false,
              "Required": true,
              "AttributeDataType": "String",
              "Mutable": false
          },
          {
              "Name": "name",
              "StringAttributeConstraints": {
                  "MinLength": "0",
                  "MaxLength": "2048"
              },
              "DeveloperOnlyAttribute": false,
```

```
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "given_name",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "family_name",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "middle_name",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "nickname",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
```

```
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "preferred_username",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "profile",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "picture",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "website",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
```

```
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "email",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "AttributeDataType": "Boolean",
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "Name": "email_verified",
            "Mutable": true
        },
        {
            "Name": "gender",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "birthdate",
            "StringAttributeConstraints": {
                "MinLength": "10",
                "MaxLength": "10"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
```

```
            {
                "Name": "zoneinfo",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "locale",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "phone_number",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "AttributeDataType": "Boolean",
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "Name": "phone_number_verified",
                "Mutable": true
            },
            {
                "Name": "address",
                "StringAttributeConstraints": {
                    "MinLength": "0",
```

```
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "updated_at",
                "NumberAttributeConstraints": {
                    "MinValue": "0"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "Number",
                "Mutable": true
            }
        ],
        "MfaConfiguration": "OFF",
        "Name": "MyUserPool",
        "LastModifiedDate": 1547833345.777,
        "AdminCreateUserConfig": {
            "UnusedAccountValidityDays": 7,
            "AllowAdminCreateUserOnly": false
        },
        "EmailConfiguration": {},
        "Policies": {
            "PasswordPolicy": {
                "RequireLowercase": true,
                "RequireSymbols": true,
                "RequireNumbers": true,
                "MinimumLength": 8,
                "RequireUppercase": true
            }
        },
        "CreationDate": 1547833345.777,
        "EstimatedNumberOfUsers": 0,
        "Id": "us-west-2_aaaaaaaaa",
        "LambdaConfig": {}
    }
}
```

두 개의 필수 속성으로 사용자 풀을 생성하는 방법

이 예제에서는 사용자 풀을 생성합니다 MyUserPool. 풀은 이메일을 사용자 이름 속성으로 받아들이도록 구성되어 있습니다. 또한 Amazon Simple Email Service를 사용하여 이메일 소스 주소를 검증된 주소로 설정합니다.

명령:

```
aws cognito-idp create-user-pool --pool-name MyUserPool --username-
attributes "email" --email-configuration=SourceArn="arn:aws:ses:us-
east-1:111111111111:identity/
jane@example.com",ReplyToEmailAddress="jane@example.com"
```

출력:

```
{
  "UserPool": {
      "SchemaAttributes": [
          {
              "Name": "sub",
              "StringAttributeConstraints": {
                  "MinLength": "1",
                  "MaxLength": "2048"
              },
              "DeveloperOnlyAttribute": false,
              "Required": true,
              "AttributeDataType": "String",
              "Mutable": false
          },
          {
              "Name": "name",
              "StringAttributeConstraints": {
                  "MinLength": "0",
                  "MaxLength": "2048"
              },
              "DeveloperOnlyAttribute": false,
              "Required": false,
              "AttributeDataType": "String",
              "Mutable": true
          },
          {
              "Name": "given_name",
              "StringAttributeConstraints": {
                  "MinLength": "0",
                  "MaxLength": "2048"
```

```
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "family_name",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "middle_name",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "nickname",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "preferred_username",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
```

```
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "profile",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "picture",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "website",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "email",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
```

```
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "AttributeDataType": "Boolean",
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "Name": "email_verified",
            "Mutable": true
        },
        {
            "Name": "gender",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "birthdate",
            "StringAttributeConstraints": {
                "MinLength": "10",
                "MaxLength": "10"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "zoneinfo",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
```

```
            "Mutable": true
        },
        {
            "Name": "locale",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "phone_number",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "AttributeDataType": "Boolean",
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "Name": "phone_number_verified",
            "Mutable": true
        },
        {
            "Name": "address",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "updated_at",
```

```
                "NumberAttributeConstraints": {
                    "MinValue": "0"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "Number",
                "Mutable": true
            }
        ],
        "MfaConfiguration": "OFF",
        "Name": "MyUserPool",
        "LastModifiedDate": 1547837788.189,
        "AdminCreateUserConfig": {
            "UnusedAccountValidityDays": 7,
            "AllowAdminCreateUserOnly": false
        },
        "EmailConfiguration": {
            "ReplyToEmailAddress": "jane@example.com",
            "SourceArn": "arn:aws:ses:us-east-1:111111111111:identity/
jane@example.com"
        },
        "Policies": {
            "PasswordPolicy": {
                "RequireLowercase": true,
                "RequireSymbols": true,
                "RequireNumbers": true,
                "MinimumLength": 8,
                "RequireUppercase": true
            }
        },
        "UsernameAttributes": [
            "email"
        ],
        "CreationDate": 1547837788.189,
        "EstimatedNumberOfUsers": 0,
        "Id": "us-west-2_aaaaaaaaa",
        "LambdaConfig": {}
    }
}
```

- API 세부 정보는 AWS CLI 명령 CreateUserPool참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolResponse;

/**
 * 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 CreateUserPool {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <userPoolName>\s

                Where:
                    userPoolName - The name to give your user pool when it's
 created.
                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userPoolName = args[0];
        CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String id = createPool(cognitoClient, userPoolName);
        System.out.println("User pool ID: " + id);
        cognitoClient.close();
    }

    public static String createPool(CognitoIdentityProviderClient cognitoClient,
String userPoolName) {
        try {
            CreateUserPoolRequest request = CreateUserPoolRequest.builder()
                    .poolName(userPoolName)
                    .build();

            CreateUserPoolResponse response =
cognitoClient.createUserPool(request);
            return response.userPool().id();

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API CreateUserPool참조를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오이 서비스를 SDK와 함께 사용 AWS. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **CreateUserPoolClient** CLI와 함께 사용

다음 코드 예제는 CreateUserPoolClient의 사용 방법을 보여줍니다.

CLI

AWS CLI

사용자 풀 클라이언트를 만들려면

이 예에서는 USER_PASSWORD_AUTH 및 ADMIN_NO_SRP_AUTH라는 두 개의 명시적 권한 부여 흐름을 사용하여 새 사용자 풀 클라이언트를 생성합니다.

명령:

```
aws cognito-idp create-user-pool-client --user-pool-id us-west-2_aaaaaaaaa
  --client-name MyNewClient --no-generate-secret --explicit-auth-flows
 "USER_PASSWORD_AUTH" "ADMIN_NO_SRP_AUTH"
```

출력:

```
{
  "UserPoolClient": {
      "UserPoolId": "us-west-2_aaaaaaaaa",
      "ClientName": "MyNewClient",
      "ClientId": "6p3bs000no6a4ue1idruvd05ad",
      "LastModifiedDate": 1548697449.497,
      "CreationDate": 1548697449.497,
      "RefreshTokenValidity": 30,
      "ExplicitAuthFlows": [
          "USER_PASSWORD_AUTH",
          "ADMIN_NO_SRP_AUTH"
      ],
      "AllowedOAuthFlowsUserPoolClient": false
  }
}
```

- API AWS CLI 세부 정보는 명령 참조를 참조하십시오. CreateUserPoolClient

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientReques
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientRespon

/**
 * A user pool client app is an application that authenticates with Amazon
 * Cognito user pools.
 * When you create a user pool, you can configure app clients that allow mobile
 * or web applications
 * to call API operations to authenticate users, manage user attributes and
 * profiles,
 * and implement sign-up and sign-in flows.
 *
 * 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 CreateUserPoolClient {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <clientName> <userPoolId>\s
```

```
                Where:
                    clientName - The name for the user pool client to create.
                    userPoolId - The ID for the user pool.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clientName = args[0];
        String userPoolId = args[1];
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createPoolClient(cognitoClient, clientName, userPoolId);
        cognitoClient.close();
    }

    public static void createPoolClient(CognitoIdentityProviderClient
 cognitoClient, String clientName,
            String userPoolId) {
        try {
            CreateUserPoolClientRequest request =
 CreateUserPoolClientRequest.builder()
                    .clientName(clientName)
                    .userPoolId(userPoolId)
                    .build();

            CreateUserPoolClientResponse response =
 cognitoClient.createUserPoolClient(request);
            System.out.println("User pool " +
 response.userPoolClient().clientName() + " created. ID: "
                    + response.userPoolClient().clientId());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API CreateUserPoolClient참조를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오이 서비스를 SDK와 함께 사용 AWS. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **DeleteUser** CLI와 함께 사용

다음 코드 예제는 DeleteUser의 사용 방법을 보여줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- Lambda 함수를 사용하여 알려진 사용자를 자동으로 확인
- Lambda 함수를 사용하여 알려진 사용자를 자동으로 마이그레이션
- Amazon Cognito 사용자 인증 후 Lambda 함수를 사용하여 사용자 지정 활동 데이터 작성

C++

SDK for C++

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::DeleteUserRequest request;
        request.SetAccessToken(accessToken);

        Aws::CognitoIdentityProvider::Model::DeleteUserOutcome outcome =
```

```
                client.DeleteUser(request);

        if (outcome.IsSuccess()) {
            std::cout << "The user " << userName << " was deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::DeleteUser. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

- API 세부 정보는 AWS SDK for C++ API DeleteUser참조를 참조하십시오.

CLI

AWS CLI

사용자 삭제

이 예시는 사용자를 삭제합니다.

명령:

```
aws cognito-idp delete-user --access-token ACCESS_TOKEN
```

- API 세부 정보는 AWS CLI 명령 DeleteUser참조를 참조하십시오.

Go

SDK for Go V2

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(userAccessToken string) error {
 _, err := actor.CognitoClient.DeleteUser(context.TODO(),
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}
```

- API 세부 정보는 AWS SDK for Go API DeleteUser참조를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오이 서비스를 SDK와 함께 사용 AWS. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **ForgotPassword** CLI와 함께 사용

다음 코드 예제는 ForgotPassword의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- Lambda 함수를 사용하여 알려진 사용자를 자동으로 마이그레이션

CLI

AWS CLI

암호를 강제로 변경하려면

다음 forgot-password 예시에서는 jane@example.com 으로 비밀번호를 변경하라는 메시지를 보냅니다.

```
aws cognito-idp forgot-password --client-id 38fjsnc484p94kpqsnet7mpld0 --username
 jane@example.com
```

출력:

```
{
    "CodeDeliveryDetails": {
        "Destination": "j***@e***.com",
        "DeliveryMedium": "EMAIL",
        "AttributeName": "email"
    }
}
```

- API 세부 정보는 AWS CLI 명령 ForgotPassword참조를 참조하십시오.

Go

SDK for Go V2

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(clientId string, userName string)
 (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(context.TODO(),
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
```

```
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}
```

- API 세부 정보는 AWS SDK for Go API ForgotPassword참조를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오이 서비스를 SDK와 함께 사용 AWS. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **InitiateAuth** CLI와 함께 사용

다음 코드 예제는 InitiateAuth의 사용 방법을 보여줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- Lambda 함수를 사용하여 알려진 사용자를 자동으로 확인
- Lambda 함수를 사용하여 알려진 사용자를 자동으로 마이그레이션
- MFA가 필요한 사용자 풀에 사용자 가입시키기
- Amazon Cognito 사용자 인증 후 Lambda 함수를 사용하여 사용자 지정 활동 데이터 작성

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Initiate authorization.
```

```
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The name of the user who is authenticating.</
param>
    /// <param name="password">The password for the user who is authenticating.</
param>
    /// <returns>The response from the initiate auth request.</returns>
    public async Task<InitiateAuthResponse> InitiateAuthAsync(string clientId,
 string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var authRequest = new InitiateAuthRequest

        {
            ClientId = clientId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.InitiateAuthAsync(authRequest);
        Console.WriteLine($"Result Challenge is : {response.ChallengeName}");

        return response;
    }
```

- API 세부 정보는 AWS SDK for .NET API InitiateAuth참조를 참조하십시오.

Go

SDK for Go V2

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(clientId string, userName string, password
 string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(context.TODO(),
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}
```

- API 세부 정보는 AWS SDK for Go API InitiateAuth참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const initiateAuth = ({ username, password, clientId }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new InitiateAuthCommand({
    AuthFlow: AuthFlowType.USER_PASSWORD_AUTH,
    AuthParameters: {
      USERNAME: username,
      PASSWORD: password,
    },
    ClientId: clientId,
  });

  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API InitiateAuth참조를 참조하십시오.

Python

SDK for Python(Boto3)

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예시에서는 추적된 디바이스로 인증을 시작하는 방법을 보여줍니다. 로그인을 완료하려면 클라이언트가 보안 원격 암호(SRP) 문제에 올바르게 응답해야 합니다.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
```

```
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_in_with_tracked_device(
        self,
        user_name,
        password,
        device_key,
        device_group_key,
        device_password,
        aws_srp,
    ):
        """
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA
 code.

        Signing in with a tracked device requires that the client respond to the
 SRP
        protocol. The scenario associated with this example uses the warrant
 package
        to help with SRP calculations.

        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains
 an
                 access token for the user.
        """
```

```
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name,
                password=device_password,
                pool_id="_",
                client_id=self.client_id,
                client_secret=None,
                client=self.cognito_idp_client,
            )

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id,
                AuthFlow="USER_PASSWORD_AUTH",
                AuthParameters={
                    "USERNAME": user_name,
                    "PASSWORD": password,
                    "DEVICE_KEY": device_key,
                },
            )
            if response_init["ChallengeName"] != "DEVICE_SRP_AUTH":
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
{response_init['ChallengeName']}."
                )

            auth_params = srp_helper.get_auth_params()
            auth_params["DEVICE_KEY"] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_SRP_AUTH",
                ChallengeResponses=auth_params,
            )
            if response_auth["ChallengeName"] != "DEVICE_PASSWORD_VERIFIER":
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}."
                )

            challenge_params = response_auth["ChallengeParameters"]
            challenge_params["USER_ID_FOR_SRP"] = device_group_key + device_key
            cr = srp_helper.process_challenge(challenge_params, {"USERNAME":
user_name})
            cr["USERNAME"] = user_name
            cr["DEVICE_KEY"] = device_key
```

```
            response_verifier =
self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_PASSWORD_VERIFIER",
                ChallengeResponses=cr,
            )
            auth_tokens = response_verifier["AuthenticationResult"]
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return auth_tokens
```

- API에 대한 자세한 내용은 파이썬용AWS SDK (Boto3) API 레퍼런스를 참조하십시오 InitiateAuth.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. 이 서비스를 SDK와 함께 사용 AWS 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **ListUserPools** CLI와 함께 사용

다음 코드 예제는 ListUserPools의 사용 방법을 보여줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
```

```
    /// List the Amazon Cognito user pools for an account.
    /// </summary>
    /// <returns>A list of UserPoolDescriptionType objects.</returns>
    public async Task<List<UserPoolDescriptionType>> ListUserPoolsAsync()
    {
        var userPools = new List<UserPoolDescriptionType>();

        var userPoolsPaginator = _cognitoService.Paginators.ListUserPools(new
 ListUserPoolsRequest());

        await foreach (var response in userPoolsPaginator.Responses)
        {
            userPools.AddRange(response.UserPools);
        }

        return userPools;
    }
```

- API 세부 정보는 AWS SDK for .NET API ListUserPools참조를 참조하십시오.

CLI

AWS CLI

사용자 풀 나열

이 예시에서는 최대 20개의 사용자 풀을 나열합니다.

명령:

```
aws cognito-idp list-user-pools --max-results 20
```

출력:

```
{
  "UserPools": [
      {
         "CreationDate": 1547763720.822,
         "LastModifiedDate": 1547763720.822,
         "LambdaConfig": {},
         "Id": "us-west-2_aaaaaaaaa",
```

```
            "Name": "MyUserPool"
        }
    ]
}
```

- API 세부 정보는 AWS CLI 명령 ListUserPools참조를 참조하십시오.

Go

SDK for Go V2

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
package main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics 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
 }
 cognitoClient := cognitoidentityprovider.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the user pools for your account.")
 var pools []types.UserPoolDescriptionType
 paginator := cognitoidentityprovider.NewListUserPoolsPaginator(
  cognitoClient, &cognitoidentityprovider.ListUserPoolsInput{MaxResults:
 aws.Int32(10)})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(context.TODO())
  if err != nil {
   log.Printf("Couldn't get user pools. Here's why: %v\n", err)
  } else {
   pools = append(pools, output.UserPools...)
  }
 }
 if len(pools) == 0 {
  fmt.Println("You don't have any user pools!")
 } else {
  for _, pool := range pools {
   fmt.Printf("\t%v: %v\n", *pool.Name, *pool.Id)
  }
 }
}
```

- API 세부 정보는 AWS SDK for Go API ListUserPools참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
```

```
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;

/**
 * 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 ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllUserPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listAllUserPools(CognitoIdentityProviderClient
 cognitoClient) {
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                    .maxResults(10)
                    .build();

            ListUserPoolsResponse response =
 cognitoClient.listUserPools(request);
            response.userPools().forEach(userpool -> {
                System.out.println("User pool " + userpool.name() + ", User ID "
 + userpool.id());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API ListUserPools참조를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn show_pools(client: &Client) -> Result<(), Error> {
    let response = client.list_user_pools().max_results(10).send().await?;
    let pools = response.user_pools();
    println!("User pools:");
    for pool in pools {
        println!("  ID:              {}", pool.id().unwrap_or_default());
        println!("  Name:            {}", pool.name().unwrap_or_default());
        println!("  Lambda Config:   {:?}", pool.lambda_config().unwrap());
        println!(
            "  Last modified:   {}",
            pool.last_modified_date().unwrap().to_chrono_utc()?
        );
        println!(
            "  Creation date:   {:?}",
            pool.creation_date().unwrap().to_chrono_utc()
        );
        println!();
    }
    println!("Next token: {}", response.next_token().unwrap_or_default());

    Ok(())
}
```

- API에 대한 자세한 내용은 Rust용AWS SDK API 레퍼런스를 참조하십시오 ListUserPools.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. 이 서비스를 SDK와 함께 사용 AWS 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **ListUsers** CLI와 함께 사용

다음 코드 예제는 ListUsers의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- MFA가 필요한 사용자 풀에 사용자 가입시키기

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Get a list of users for the Amazon Cognito user pool.
/// </summary>
/// <param name="userPoolId">The user pool ID.</param>
/// <returns>A list of users.</returns>
public async Task<List<UserType>> ListUsersAsync(string userPoolId)
{
    var request = new ListUsersRequest
    {
        UserPoolId = userPoolId
    };

    var users = new List<UserType>();

    var usersPaginator = _cognitoService.Paginators.ListUsers(request);
    await foreach (var response in usersPaginator.Responses)
    {
```

```
            users.AddRange(response.Users);
        }

        return users;
    }
```

- API 세부 정보는 AWS SDK for .NET API ListUsers참조를 참조하십시오.

CLI

AWS CLI

사용자 나열

이 예시에서는 최대 20개의 사용자를 나열합니다.

명령:

```
aws cognito-idp list-users --user-pool-id us-west-2_aaaaaaaaa --limit 20
```

출력:

```
{
  "Users": [
      {
          "Username": "22704aa3-fc10-479a-97eb-2af5806bd327",
          "Enabled": true,
          "UserStatus": "FORCE_CHANGE_PASSWORD",
          "UserCreateDate": 1548089817.683,
          "UserLastModifiedDate": 1548089817.683,
          "Attributes": [
              {
                  "Name": "sub",
                  "Value": "22704aa3-fc10-479a-97eb-2af5806bd327"
              },
              {
                  "Name": "email_verified",
                  "Value": "true"
              },
              {
                  "Name": "email",
```

```
                "Value": "mary@example.com"
            }
        ]
    }
  ]
}
```

- API 세부 정보는 AWS CLI 명령 ListUsers참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersResponse;

/**
 * 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) {

        final String usage = """
```

```
                Usage:
                    <userPoolId>\s

                Where:
                    userPoolId - The ID given to your user pool when it's
created.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userPoolId = args[0];
        CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllUsers(cognitoClient, userPoolId);
        listUsersFilter(cognitoClient, userPoolId);
        cognitoClient.close();
    }

    public static void listAllUsers(CognitoIdentityProviderClient cognitoClient,
String userPoolId) {
        try {
            ListUsersRequest usersRequest = ListUsersRequest.builder()
                    .userPoolId(userPoolId)
                    .build();

            ListUsersResponse response = cognitoClient.listUsers(usersRequest);
            response.users().forEach(user -> {
                System.out.println("User " + user.username() + " Status " +
user.userStatus() + " Created "
                        + user.userCreateDate());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
    // Shows how to list users by using a filter.
    public static void listUsersFilter(CognitoIdentityProviderClient
 cognitoClient, String userPoolId) {

        try {
            String filter = "email = \"tblue@noserver.com\"";
            ListUsersRequest usersRequest = ListUsersRequest.builder()
                    .userPoolId(userPoolId)
                    .filter(filter)
                    .build();

            ListUsersResponse response = cognitoClient.listUsers(usersRequest);
            response.users().forEach(user -> {
                System.out.println("User with filter applied " + user.username()
 + " Status " + user.userStatus()
                        + " Created " + user.userCreateDate());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API ListUsers참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const listUsers = ({ userPoolId }) => {
  const client = new CognitoIdentityProviderClient({});
```

```
  const command = new ListUsersCommand({
    UserPoolId: userPoolId,
  });

  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API ListUsers참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listAllUsers(userPoolId: String) {

    val request = ListUsersRequest {
        this.userPoolId = userPoolId
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { cognitoClient ->
        val response = cognitoClient.listUsers(request)
        response.users?.forEach { user ->
            println("The user name is ${user.username}")
        }
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 ListUsers.

Python

SDK for Python(Boto3)

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def list_users(self):
        """
        Returns a list of the users in the current user pool.

        :return: The list of users.
        """
        try:
            response =
 self.cognito_idp_client.list_users(UserPoolId=self.user_pool_id)
            users = response["Users"]
        except ClientError as err:
            logger.error(
                "Couldn't list users for %s. Here's why: %s: %s",
```

```
                self.user_pool_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return users
```

- API에 대한 자세한 내용은 파이썬용AWS SDK (Boto3) API 레퍼런스를 참조하십시오 ListUsers.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. 이 서비스를 SDK와 함께 사용 AWS 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **ResendConfirmationCode** CLI와 함께 사용

다음 코드 예제는 ResendConfirmationCode의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- MFA가 필요한 사용자 풀에 사용자 가입시키기

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Send a new confirmation code to a user.
/// </summary>
/// <param name="clientId">The Id of the client application.</param>
```

```
    /// <param name="userName">The username of user who will receive the code.</
param>
    /// <returns>The delivery details.</returns>
    public async Task<CodeDeliveryDetailsType> ResendConfirmationCodeAsync(string
 clientId, string userName)
    {
        var codeRequest = new ResendConfirmationCodeRequest
        {
            ClientId = clientId,
            Username = userName,
        };

        var response = await
 _cognitoService.ResendConfirmationCodeAsync(codeRequest);

        Console.WriteLine($"Method of delivery is
 {response.CodeDeliveryDetails.DeliveryMedium}");

        return response.CodeDeliveryDetails;
    }
```

- API 세부 정보는 AWS SDK for .NET API ResendConfirmationCode참조를 참조하십시오.

C++

SDK for C++

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);
```

```
        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeRequest
request;
        request.SetUsername(userName);
        request.SetClientId(clientID);

        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeOutcome
outcome =
                client.ResendConfirmationCode(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "CognitoIdentityProvider::ResendConfirmationCode was
successful."
                    << std::endl;
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::ResendConfirmationCode. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- API 세부 정보는 AWS SDK for C++ API [ResendConfirmationCode](#)참조를 참조하십시오.

CLI

AWS CLI

확인 코드 다시 보내기

다음 `resend-confirmation-code` 예시에서는 사용자 `jane`에게 확인 코드를 보냅니다.

```
aws cognito-idp resend-confirmation-code \
    --client-id 12a3b456c7de890f11g123hijk \
    --username jane
```

출력:

```
{
    "CodeDeliveryDetails": {
```

```
        "Destination": "j***@e***.com",
        "DeliveryMedium": "EMAIL",
        "AttributeName": "email"
    }
}
```

자세한 내용은 Amazon Cognito 개발자 안내서의 사용자 계정 가입 및 확인 섹션을 참조하세요.

- API 세부 정보는 AWS CLI 명령 ResendConfirmationCode참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
            String userName) {
       try {
            ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
                    .clientId(clientId)
                    .username(userName)
                    .build();

            ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
            System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

       } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API ResendConfirmationCode참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const resendConfirmationCode = ({ clientId, username }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ResendConfirmationCodeCommand({
    ClientId: clientId,
    Username: username,
  });

  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API ResendConfirmationCode참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun resendConfirmationCode(clientIdVal: String?, userNameVal: String?) {
    val codeRequest = ResendConfirmationCodeRequest {
```

```
        clientId = clientIdVal
        username = userNameVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 ResendConfirmationCode.

Python

SDK for Python(Boto3)

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret
```

```
    def resend_confirmation(self, user_name):
        """
        Prompts Amazon Cognito to resend an email with a new confirmation code.

        :param user_name: The name of the user who will receive the email.
        :return: Delivery information about where the email is sent.
        """
        try:
            kwargs = {"ClientId": self.client_id, "Username": user_name}
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.resend_confirmation_code(**kwargs)
            delivery = response["CodeDeliveryDetails"]
        except ClientError as err:
            logger.error(
                "Couldn't resend confirmation to %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return delivery
```

- API에 대한 자세한 내용은 파이썬용AWS SDK (Boto3) API 레퍼런스를 참조하십시오 ResendConfirmationCode.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. 이 서비스를 SDK와 함께 사용 AWS 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **RespondToAuthChallenge** CLI와 함께 사용

다음 코드 예제는 RespondToAuthChallenge의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- MFA가 필요한 사용자 풀에 사용자 가입시키기

CLI

AWS CLI

인증 문제에 응답

이 예시에서는 initiate-auth로 시작된 인증 문제에 응답합니다. 이것은 NEW_PASSWORD_REQUIRED 문제에 대한 응답입니다. 사용자 jane@example.com의 암호를 설정합니다.

명령:

```
aws cognito-idp respond-to-auth-challenge --client-id 3n4b5urk1ft4fl3mg5e62d9ado
 --challenge-name NEW_PASSWORD_REQUIRED --challenge-responses
 USERNAME=jane@example.com,NEW_PASSWORD="password" --session "SESSION_TOKEN"
```

출력:

```
{
  "ChallengeParameters": {},
  "AuthenticationResult": {
      "AccessToken": "ACCESS_TOKEN",
      "ExpiresIn": 3600,
      "TokenType": "Bearer",
      "RefreshToken": "REFRESH_TOKEN",
      "IdToken": "ID_TOKEN",
      "NewDeviceMetadata": {
          "DeviceKey": "us-west-2_fec070d2-fa88-424a-8ec8-b26d7198eb23",
          "DeviceGroupKey": "-wt2ha1Zd"
      }
  }
}
```

- API 세부 정보는 AWS CLI 명령 RespondToAuthChallenge참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const respondToAuthChallenge = ({
  clientId,
  username,
  session,
  userPoolId,
  code,
}) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new RespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: code,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API RespondToAuthChallenge참조를 참조하십시오.

Python

SDK for Python(Boto3)

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

추적되는 디바이스로 로그인하세요. 로그인을 완료하려면 클라이언트가 보안 원격 암호(SRP) 문제에 올바르게 응답해야 합니다.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_in_with_tracked_device(
        self,
        user_name,
        password,
        device_key,
        device_group_key,
        device_password,
        aws_srp,
    ):
        """
```

```
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA
 code.

        Signing in with a tracked device requires that the client respond to the
 SRP
        protocol. The scenario associated with this example uses the warrant
 package
        to help with SRP calculations.

        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains
 an
                 access token for the user.
        """
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name,
                password=device_password,
                pool_id="_",
                client_id=self.client_id,
                client_secret=None,
                client=self.cognito_idp_client,
            )

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id,
                AuthFlow="USER_PASSWORD_AUTH",
                AuthParameters={
                    "USERNAME": user_name,
                    "PASSWORD": password,
                    "DEVICE_KEY": device_key,
                },
            )
            if response_init["ChallengeName"] != "DEVICE_SRP_AUTH":
```

```
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
{response_init['ChallengeName']}."
                )

            auth_params = srp_helper.get_auth_params()
            auth_params["DEVICE_KEY"] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_SRP_AUTH",
                ChallengeResponses=auth_params,
            )
            if response_auth["ChallengeName"] != "DEVICE_PASSWORD_VERIFIER":
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}."
                )

            challenge_params = response_auth["ChallengeParameters"]
            challenge_params["USER_ID_FOR_SRP"] = device_group_key + device_key
            cr = srp_helper.process_challenge(challenge_params, {"USERNAME":
user_name})
            cr["USERNAME"] = user_name
            cr["DEVICE_KEY"] = device_key
            response_verifier =
self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_PASSWORD_VERIFIER",
                ChallengeResponses=cr,
            )
            auth_tokens = response_verifier["AuthenticationResult"]
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return auth_tokens
```

- API에 대한 자세한 내용은 파이썬용AWS SDK (Boto3) API 레퍼런스를 참조하십시오 RespondToAuthChallenge.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. 이 서비스를 SDK와 함께 사용 AWS 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **SignUp** CLI와 함께 사용

다음 코드 예제는 SignUp의 사용 방법을 보여줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- Lambda 함수를 사용하여 알려진 사용자를 자동으로 확인
- Lambda 함수를 사용하여 알려진 사용자를 자동으로 마이그레이션
- MFA가 필요한 사용자 풀에 사용자 가입시키기

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Sign up a new user.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The username to use.</param>
    /// <param name="password">The user's password.</param>
    /// <param name="email">The email address of the user.</param>
    /// <returns>A Boolean value indicating whether the user was confirmed.</
returns>
    public async Task<bool> SignUpAsync(string clientId, string userName, string
 password, string email)
    {
        var userAttrs = new AttributeType
```

```
        {
            Name = "email",
            Value = email,
        };

        var userAttrsList = new List<AttributeType>();

        userAttrsList.Add(userAttrs);

        var signUpRequest = new SignUpRequest
        {
            UserAttributes = userAttrsList,
            Username = userName,
            ClientId = clientId,
            Password = password
        };

        var response = await _cognitoService.SignUpAsync(signUpRequest);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API 세부 정보는 AWS SDK for .NET API SignUp참조를 참조하십시오.

C++

SDK for C++

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);
```

```
        Aws::CognitoIdentityProvider::Model::SignUpRequest request;
        request.AddUserAttributes(
                Aws::CognitoIdentityProvider::Model::AttributeType().WithName(
                        "email").WithValue(email));
        request.SetUsername(userName);
        request.SetPassword(password);
        request.SetClientId(clientID);
        Aws::CognitoIdentityProvider::Model::SignUpOutcome outcome =
                client.SignUp(request);

        if (outcome.IsSuccess()) {
            std::cout << "The signup request for " << userName << " was
successful."
                      << std::endl;
        }
        else if (outcome.GetError().GetErrorType() ==

Aws::CognitoIdentityProvider::CognitoIdentityProviderErrors::USERNAME_EXISTS) {
            std::cout
                    << "The username already exists. Please enter a different
username."
                    << std::endl;
            userExists = true;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::SignUpRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- API 세부 정보는 AWS SDK for C++ API SignUp참조를 참조하십시오.

CLI

AWS CLI

사용자 가입

이 예시에서는 jane@example.com에 가입합니다.

명령:

```
aws cognito-idp sign-up --client-id 3n4b5urk1ft4fl3mg5e62d9ado --
username jane@example.com --password PASSWORD --user-attributes
 Name="email",Value="jane@example.com" Name="name",Value="Jane"
```

출력:

```
{
  "UserConfirmed": false,
  "UserSub": "e04d60a6-45dc-441c-a40b-e25a787d4862"
}
```

- API 세부 정보는 AWS CLI 명령 SignUp참조를 참조하십시오.

Go

SDK for Go V2

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(clientId string, userName string, password
 string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(context.TODO(),
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
```

```
    {Name: aws.String("email"), Value: aws.String(userEmail)},
   },
  })
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    log.Println(*invalidPassword.Message)
   } else {
    log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
   }
  } else {
   confirmed = output.UserConfirmed
  }
  return confirmed, err
}
```

- API 세부 정보는 AWS SDK for Go API SignUp참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public static void signUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String userName,
            String password, String email) {
        AttributeType userAttrs = AttributeType.builder()
                .name("email")
                .value(email)
                .build();

        List<AttributeType> userAttrsList = new ArrayList<>();
        userAttrsList.add(userAttrs);
        try {
            SignUpRequest signUpRequest = SignUpRequest.builder()
```

```
                    .userAttributes(userAttrsList)
                    .username(userName)
                    .clientId(clientId)
                    .password(password)
                    .build();

            identityProviderClient.signUp(signUpRequest);
            System.out.println("User has been signed up ");

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API SignUp참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const signUp = ({ clientId, username, password, email }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new SignUpCommand({
    ClientId: clientId,
    Username: username,
    Password: password,
    UserAttributes: [{ Name: "email", Value: email }],
  });

  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API SignUp참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun signUp(clientIdVal: String?, userNameVal: String?, passwordVal:
 String?, emailVal: String?) {
    val userAttrs = AttributeType {
        name = "email"
        value = emailVal
    }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)
    val signUpRequest = SignUpRequest {
        userAttributes = userAttrsList
        username = userNameVal
        clientId = clientIdVal
        password = passwordVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 SignUp.

Python

SDK for Python(Boto3)

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_up_user(self, user_name, password, user_email):
        """
        Signs up a new user with Amazon Cognito. This action prompts Amazon
 Cognito
        to send an email to the specified email address. The email contains a
 code that
        can be used to confirm the user.

        When the user already exists, the user status is checked to determine
 whether
        the user has been confirmed.

        :param user_name: The user name that identifies the new user.
```

```
        :param password: The password for the new user.
        :param user_email: The email address for the new user.
        :return: True when the user is already confirmed with Amazon Cognito.
                 Otherwise, false.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "Password": password,
                "UserAttributes": [{"Name": "email", "Value": user_email}],
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.sign_up(**kwargs)
            confirmed = response["UserConfirmed"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "UsernameExistsException":
                response = self.cognito_idp_client.admin_get_user(
                    UserPoolId=self.user_pool_id, Username=user_name
                )
                logger.warning(
                    "User %s exists and is %s.", user_name,
response["UserStatus"]
                )
                confirmed = response["UserStatus"] == "CONFIRMED"
            else:
                logger.error(
                    "Couldn't sign up %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return confirmed
```

- API에 대한 자세한 내용은 파이썬용AWS SDK (Boto3) API 레퍼런스를 참조하십시오 SignUp.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. 이 서비스를 SDK와 함께 사용 AWS 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **UpdateUserPool** CLI와 함께 사용

다음 코드 예제는 UpdateUserPool의 사용 방법을 보여줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- Lambda 함수를 사용하여 알려진 사용자를 자동으로 확인
- Lambda 함수를 사용하여 알려진 사용자를 자동으로 마이그레이션
- Amazon Cognito 사용자 인증 후 Lambda 함수를 사용하여 사용자 지정 활동 데이터 작성

CLI

AWS CLI

사용자 풀을 업데이트하려면

이 예에서는 사용자 풀에 태그를 추가합니다.

명령:

```
aws cognito-idp update-user-pool --user-pool-id us-west-2_aaaaaaaaa --user-pool-
tags Team=Blue,Area=West
```

- API 세부 정보는 AWS CLI 명령 UpdateUserPool참조를 참조하십시오.

Go

SDK for Go V2

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(userPoolId string,
 triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(context.TODO(),
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
```

```
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(context.TODO(),
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}
```

- API 세부 정보는 AWS SDK for Go API UpdateUserPool참조를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오이 서비스를 SDK와 함께 사용 AWS. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **VerifySoftwareToken** CLI와 함께 사용

다음 코드 예제는 VerifySoftwareToken의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- MFA가 필요한 사용자 풀에 사용자 가입시키기

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
```

```
    /// Verify the TOTP and register for MFA.
    /// </summary>
    /// <param name="session">The name of the session.</param>
    /// <param name="code">The MFA code.</param>
    /// <returns>The status of the software token.</returns>
    public async Task<VerifySoftwareTokenResponseType>
VerifySoftwareTokenAsync(string session, string code)
    {
        var tokenRequest = new VerifySoftwareTokenRequest
        {
            UserCode = code,
            Session = session,
        };

        var verifyResponse = await
_cognitoService.VerifySoftwareTokenAsync(tokenRequest);

        return verifyResponse.Status;
    }
```

- API 세부 정보는 AWS SDK for .NET API VerifySoftwareToken참조를 참조하십시오.

C++

SDK for C++

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenRequest request;
```

```
        request.SetUserCode(userCode);
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenOutcome outcome =
                client.VerifySoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout << "Verification of the code was successful."
                      << std::endl;
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::VerifySoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- API 세부 정보는 AWS SDK for C++ API VerifySoftwareToken참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    // Verify the TOTP and register for MFA.
    public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
        try {
            VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
                    .userCode(code)
                    .session(session)
                    .build();
```

```
            VerifySoftwareTokenResponse verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest);
            System.out.println("The status of the token is " +
 verifyResponse.statusAsString());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API VerifySoftwareToken참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const verifySoftwareToken = (totp) => {
  const client = new CognitoIdentityProviderClient({});

  // The 'Session' is provided in the response to 'AssociateSoftwareToken'.
  const session = process.env.SESSION;

  if (!session) {
    throw new Error(
      "Missing a valid Session. Did you run 'admin-initiate-auth'?",
    );
  }

  const command = new VerifySoftwareTokenCommand({
    Session: session,
    UserCode: totp,
  });

  return client.send(command);
```

```
};
```

- API 세부 정보는 AWS SDK for JavaScript API VerifySoftwareToken참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(sessionVal: String?, codeVal: String?) {
    val tokenRequest = VerifySoftwareTokenRequest {
        userCode = codeVal
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 VerifySoftwareToken.

Python

SDK for Python(Boto3)

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def verify_mfa(self, session, user_code):
        """
        Verify a new MFA application that is associated with a user.

        :param session: Session information returned from a previous call to
 initiate
                        authentication.
        :param user_code: A code generated by the associated MFA application.
        :return: Status that indicates whether the MFA application is verified.
        """
        try:
            response = self.cognito_idp_client.verify_software_token(
                Session=session, UserCode=user_code
```

```
            )
        except ClientError as err:
            logger.error(
                "Couldn't verify MFA. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response
```

- API에 대한 자세한 내용은 파이썬용AWS SDK (Boto3) API 레퍼런스를 참조하십시오 VerifySoftwareToken.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. 이 서비스를 SDK와 함께 사용 AWS 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

SDK를 사용하는 Amazon Cognito 자격 증명 공급자의 시나리오 AWS

다음 코드 예제는 SDK를 사용하여 Amazon Cognito ID 공급자에서 일반적인 시나리오를 구현하는 방법을 보여줍니다. AWS 이러한 시나리오에서는 Amazon Cognito 자격 증명 공급자 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여줍니다. 각 시나리오에는 코드 설정 및 실행 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. GitHub

예제

- SDK를 사용하여 Lambda 함수를 사용하여 알려진 Amazon Cognito 사용자를 자동으로 확인합니다. AWS
- SDK를 사용하여 Lambda 함수를 사용하여 알려진 Amazon Cognito 사용자를 자동으로 마이그레이션합니다. AWS
- SDK를 사용하여 MFA가 필요한 Amazon Cognito 사용자 풀에 사용자를 등록합니다. AWS
- SDK를 사용한 Amazon Cognito 사용자 인증 후 Lambda 함수를 사용하여 사용자 지정 활동 데이터를 작성합니다. AWS

SDK를 사용하여 Lambda 함수를 사용하여 알려진 Amazon Cognito 사용자를 자동으로 확인합니다. AWS

다음 코드 예제는 Lambda 함수를 사용하여 알려진 Amazon Cognito 사용자를 자동으로 확인하는 방법을 보여줍니다.

- `PreSignUp` 트리거에 대해 Lambda 함수를 호출하도록 사용자 풀을 구성합니다.
- Amazon Cognito를 사용하여 사용자 가입시키기
- Lambda 함수는 DynamoDB 테이블을 스캔하고 알려진 사용자를 자동으로 확인합니다.
- 새 사용자로 로그인한 다음 리소스를 정리합니다.

Go

SDK for Go V2

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
// AutoConfirm separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type AutoConfirm struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewAutoConfirm constructs a new auto confirm runner.
func NewAutoConfirm(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) AutoConfirm {
 scenario := AutoConfirm{
  helper:       helper,
  questioner:   questioner,
```

```
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddPreSignUpTrigger adds a Lambda handler as an invocation target for the
 PreSignUp trigger.
func (runner *AutoConfirm) AddPreSignUpTrigger(userPoolId string, functionArn
 string) {
 log.Printf("Let's add a Lambda function to handle the PreSignUp trigger from
 Cognito.\n" +
  "This trigger happens when a user signs up, and lets your function take action
 before the main Cognito\n" +
  "sign up processing occurs.\n")
 err := runner.cognitoActor.UpdateTriggers(
  userPoolId,
  actions.TriggerInfo{Trigger: actions.PreSignUp, HandlerArn:
 aws.String(functionArn)})
 if err != nil {
  panic(err)
 }
 log.Printf("Lambda function %v added to user pool %v to handle the PreSignUp
 trigger.\n",
  functionArn, userPoolId)
}

// SignUpUser signs up a user from the known user table with a password you
 specify.
func (runner *AutoConfirm) SignUpUser(clientId string, usersTable string)
 (string, string) {
 log.Println("Let's sign up a user to your Cognito user pool. When the user's
 email matches an email in the\n" +
  "DynamoDB known users table, it is automatically verified and the user is
 confirmed.")

 knownUsers, err := runner.helper.GetKnownUsers(usersTable)
 if err != nil {
  panic(err)
 }
 userChoice := runner.questioner.AskChoice("Which user do you want to use?\n",
 knownUsers.UserNameList())
```

```
 user := knownUsers.Users[userChoice]

 var signedUp bool
 var userConfirmed bool
 password := runner.questioner.AskPassword("Enter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !signedUp {
  log.Printf("Signing up user '%v' with email '%v' to Cognito.\n", user.UserName,
 user.UserEmail)
  userConfirmed, err = runner.cognitoActor.SignUp(clientId, user.UserName,
 password, user.UserEmail)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("Enter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   signedUp = true
  }
 }
 log.Printf("User %v signed up, confirmed = %v.\n", user.UserName, userConfirmed)

 log.Println(strings.Repeat("-", 88))

 return user.UserName, password
}

// SignInUser signs in a user.
func (runner *AutoConfirm) SignInUser(clientId string, userName string, password
 string) string {
 runner.questioner.Ask("Press Enter when you're ready to continue.")
 log.Printf("Let's sign in as %v...\n", userName)
 authResult, err := runner.cognitoActor.SignIn(clientId, userName, password)
 if err != nil {
  panic(err)
 }
 log.Printf("Successfully signed in. Your access token starts with: %v...\n",
 (*authResult.AccessToken)[:10])
 log.Println(strings.Repeat("-", 88))
 return *authResult.AccessToken
}
```

```
// Run runs the scenario.
func (runner *AutoConfirm) Run(stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup()
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]
 runner.helper.PopulateUserTable(stackOutputs["TableName"])

 runner.AddPreSignUpTrigger(stackOutputs["UserPoolId"],
 stackOutputs["AutoConfirmFunctionArn"])
 runner.resources.triggers = append(runner.resources.triggers, actions.PreSignUp)
 userName, password := runner.SignUpUser(stackOutputs["UserPoolClientId"],
 stackOutputs["TableName"])
 runner.helper.ListRecentLogEvents(stackOutputs["AutoConfirmFunction"])
 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
  runner.SignInUser(stackOutputs["UserPoolClientId"], userName, password))

 runner.resources.Cleanup()

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Lambda 함수를 사용하여 `PreSignUp` 트리거를 처리합니다.

```
const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string `dynamodbav:"UserName"`
 UserEmail string `dynamodbav:"UserEmail"`
}

// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
 userEmail, err := attributevalue.Marshal(user.UserEmail)
 if err != nil {
  panic(err)
 }
 return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the PreSignUp event by looking up a user in an Amazon
 DynamoDB table and
// specifying whether they should be confirmed and verified.
func (h *handler) HandleRequest(ctx context.Context, event
 events.CognitoEventUserPoolsPreSignup) (events.CognitoEventUserPoolsPreSignup,
 error) {
 log.Printf("Received presignup from %v for user '%v'", event.TriggerSource,
 event.UserName)
 if event.TriggerSource != "PreSignUp_SignUp" {
  // Other trigger sources, such as PreSignUp_AdminInitiateAuth, ignore the
 response from this handler.
  return event, nil
 }
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserEmail: event.Request.UserAttributes["email"],
 }
 log.Printf("Looking up email %v in table %v.\n", user.UserEmail, tableName)
 output, err := h.dynamoClient.GetItem(ctx, &dynamodb.GetItemInput{
  Key:       user.GetKey(),
  TableName: aws.String(tableName),
 })
```

```
 if err != nil {
  log.Printf("Error looking up email %v.\n", user.UserEmail)
  return event, err
 }
 if output.Item == nil {
  log.Printf("Email %v not found. Email verification is required.\n",
 user.UserEmail)
  return event, err
 }

 err = attributevalue.UnmarshalMap(output.Item, &user)
 if err != nil {
  log.Printf("Couldn't unmarshal DynamoDB item. Here's why: %v\n", err)
  return event, err
 }

 if user.UserName != event.UserName {
  log.Printf("UserEmail %v found, but stored UserName '%v' does not match
 supplied UserName '%v'. Verification is required.\n",
   user.UserEmail, user.UserName, event.UserName)
 } else {
  log.Printf("UserEmail %v found with matching UserName %v. User is confirmed.
\n", user.UserEmail, user.UserName)
  event.Response.AutoConfirmUser = true
  event.Response.AutoVerifyEmail = true
 }

 return event, err
}

func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

일반적인 작업을 수행하는 구조체를 생성합니다.

```
// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(stackName string) (actions.StackOutputs, error)
 PopulateUserTable(tableName string)
 GetKnownUsers(tableName string) (actions.UserList, error)
 AddKnownUser(tableName string, user actions.User)
 ListRecentLogEvents(functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
```

```
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(stackName string)
 (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(tableName string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(tableName string) (actions.UserList,
 error) {
 knownUsers, err := helper.dynamoActor.Scan(tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(tableName string, user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
```

```
func (helper ScenarioHelper) ListRecentLogEvents(functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Amazon Cognito 작업을 래핑하는 구조체를 생성합니다.

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)
```

```
type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(userPoolId string,
 triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(context.TODO(),
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(context.TODO(),
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}
```

```
// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(clientId string, userName string, password
 string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(context.TODO(),
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(clientId string, userName string, password
 string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(context.TODO(),
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
```

```
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(clientId string, userName string)
 (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(context.TODO(),
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(clientId string, code string,
 userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(context.TODO(),
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
```

```
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(userAccessToken string) error {
 _, err := actor.CognitoClient.DeleteUser(context.TODO(),
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(userPoolId string, userName string,
 userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(context.TODO(),
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
```

```
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(userPoolId string, userName
 string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(context.TODO(),
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

DynamoDB 작업을 래핑하는 구조체를 생성합니다.

```
// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
```

```
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(tableName string) error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
```

```
 _, err = actor.DynamoClient.BatchWriteItem(context.TODO(),
 &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(tableName string) (UserList, error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(context.TODO(), &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(tableName string, user User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(context.TODO(), &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

CloudWatch Logs 액션을 래핑하는 구조체를 만드세요.

```
type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(functionName string)
 (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(context.TODO(),
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(functionName string,
 logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(context.TODO(),
 &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
```

```
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

액션을 래핑하는 구조체를 만드세요. AWS CloudFormation

```
// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(stackName string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(context.TODO(),
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

리소스를 정리합니다.

```
// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup() {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 \n" +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
```

```
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
 HandlerArn: nil}
  }
  err := resources.cognitoActor.UpdateTriggers(resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
   panic(err)
  }
  log.Println("Removed Cognito triggers from user pool.")
 } else {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
 }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 다음 주제를 참조하십시오.
  - DeleteUser
  - InitiateAuth
  - SignUp
  - UpdateUserPool

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. 이 서비스를 SDK와 함께 사용 AWS 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

SDK를 사용하여 Lambda 함수를 사용하여 알려진 Amazon Cognito 사용자를 자동으로 마이그레이션합니다. AWS

다음 코드 예제는 Lambda 함수를 사용하여 알려진 Amazon Cognito 사용자를 자동으로 마이그레이션하는 방법을 보여줍니다.

- `MigrateUser` 트리거에 대해 Lambda 함수를 호출하도록 사용자 풀을 구성합니다.
- 사용자 풀에 없는 사용자 이름과 이메일을 사용하여 Amazon Cognito에 로그인합니다.
- Lambda 함수는 DynamoDB 테이블을 스캔하고 알려진 사용자를 사용자 풀로 자동으로 마이그레이션합니다.

- 암호 찾기 흐름을 수행하여 마이그레이션된 사용자의 암호를 재설정합니다.
- 새 사용자로 로그인한 다음 리소스를 정리합니다.

Go

SDK for Go V2

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
import (
 "errors"
 "fmt"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// MigrateUser separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type MigrateUser struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewMigrateUser constructs a new migrate user runner.
```

```
func NewMigrateUser(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) MigrateUser {
 scenario := MigrateUser{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddMigrateUserTrigger adds a Lambda handler as an invocation target for the
 MigrateUser trigger.
func (runner *MigrateUser) AddMigrateUserTrigger(userPoolId string, functionArn
 string) {
 log.Printf("Let's add a Lambda function to handle the MigrateUser trigger from
 Cognito.\n" +
  "This trigger happens when an unknown user signs in, and lets your function
 take action before Cognito\n" +
  "rejects the user.\n\n")
 err := runner.cognitoActor.UpdateTriggers(
  userPoolId,
  actions.TriggerInfo{Trigger: actions.UserMigration, HandlerArn:
 aws.String(functionArn)})
 if err != nil {
  panic(err)
 }
 log.Printf("Lambda function %v added to user pool %v to handle the MigrateUser
 trigger.\n",
  functionArn, userPoolId)

 log.Println(strings.Repeat("-", 88))
}

// SignInUser adds a new user to the known users table and signs that user in to
 Amazon Cognito.
func (runner *MigrateUser) SignInUser(usersTable string, clientId string) (bool,
 actions.User) {
 log.Println("Let's sign in a user to your Cognito user pool. When the username
 and email matches an entry in the\n" +
  "DynamoDB known users table, the email is automatically verified and the user
 is migrated to the Cognito user pool.")
```

```
 user := actions.User{}
 user.UserName = runner.questioner.Ask("\nEnter a username:")
 user.UserEmail = runner.questioner.Ask("\nEnter an email that you own. This
 email will be used to confirm user migration\n" +
  "during this example:")

 runner.helper.AddKnownUser(usersTable, user)

 var err error
 var resetRequired *types.PasswordResetRequiredException
 var authResult *types.AuthenticationResultType
 signedIn := false
 for !signedIn && resetRequired == nil {
  log.Printf("Signing in to Cognito as user '%v'. The expected result is a
 PasswordResetRequiredException.\n\n", user.UserName)
  authResult, err = runner.cognitoActor.SignIn(clientId, user.UserName, "_")
  if err != nil {
   if errors.As(err, &resetRequired) {
    log.Printf("\nUser '%v' is not in the Cognito user pool but was found in the
 DynamoDB known users table.\n"+
     "User migration is started and a password reset is required.",
 user.UserName)
   } else {
    panic(err)
   }
  } else {
   log.Printf("User '%v' successfully signed in. This is unexpected and probably
 means you have not\n"+
    "cleaned up a previous run of this scenario, so the user exist in the Cognito
 user pool.\n"+
    "You can continue this example and select to clean up resources, or manually
 remove\n"+
    "the user from your user pool and try again.", user.UserName)
   runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)
   signedIn = true
  }
 }

 log.Println(strings.Repeat("-", 88))
 return resetRequired != nil, user
}
```

```
// ResetPassword starts a password recovery flow.
func (runner *MigrateUser) ResetPassword(clientId string, user actions.User) {
 wantCode := runner.questioner.AskBool(fmt.Sprintf("In order to migrate the user
 to Cognito, you must be able to receive a confirmation\n"+
  "code by email at %v. Do you want to send a code (y/n)?", user.UserEmail), "y")
 if !wantCode {
  log.Println("To complete this example and successfully migrate a user to
 Cognito, you must enter an email\n" +
   "you own that can receive a confirmation code.")
  return
 }
 codeDelivery, err := runner.cognitoActor.ForgotPassword(clientId, user.UserName)
 if err != nil {
  panic(err)
 }
 log.Printf("\nA confirmation code has been sent to %v.",
 *codeDelivery.Destination)
 code := runner.questioner.Ask("Check your email and enter it here:")

 confirmed := false
 password := runner.questioner.AskPassword("\nEnter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !confirmed {
  log.Printf("\nConfirming password reset for user '%v'.\n", user.UserName)
  err = runner.cognitoActor.ConfirmForgotPassword(clientId, code, user.UserName,
 password)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("\nEnter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   confirmed = true
  }
 }
 log.Printf("User '%v' successfully confirmed and migrated.\n", user.UserName)
 log.Println("Signing in with your username and password...")
 authResult, err := runner.cognitoActor.SignIn(clientId, user.UserName, password)
 if err != nil {
  panic(err)
 }
```

```
 log.Printf("Successfully signed in. Your access token starts with: %v...\n",
 (*authResult.AccessToken)[:10])
 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)

 log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
func (runner *MigrateUser) Run(stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup()
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]

 runner.AddMigrateUserTrigger(stackOutputs["UserPoolId"],
 stackOutputs["MigrateUserFunctionArn"])
 runner.resources.triggers = append(runner.resources.triggers,
 actions.UserMigration)
 resetNeeded, user := runner.SignInUser(stackOutputs["TableName"],
 stackOutputs["UserPoolClientId"])
 if resetNeeded {
  runner.helper.ListRecentLogEvents(stackOutputs["MigrateUserFunction"])
  runner.ResetPassword(stackOutputs["UserPoolClientId"], user)
 }

 runner.resources.Cleanup()

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
```

```
}
```

Lambda 함수를 사용하여 `MigrateUser` 트리거를 처리합니다.

```
const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string `dynamodbav:"UserName"`
 UserEmail string `dynamodbav:"UserEmail"`
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the MigrateUser event by looking up a user in an Amazon
 DynamoDB table and
// specifying whether they should be migrated to the user pool.
func (h *handler) HandleRequest(ctx context.Context, event
 events.CognitoEventUserPoolsMigrateUser)
 (events.CognitoEventUserPoolsMigrateUser, error) {
 log.Printf("Received migrate trigger from %v for user '%v'",
 event.TriggerSource, event.UserName)
 if event.TriggerSource != "UserMigration_Authentication" {
  return event, nil
 }
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserName: event.UserName,
 }
 log.Printf("Looking up user '%v' in table %v.\n", user.UserName, tableName)
 filterEx := expression.Name("UserName").Equal(expression.Value(user.UserName))
 expr, err := expression.NewBuilder().WithFilter(filterEx).Build()
 if err != nil {
  log.Printf("Error building expression to query for user '%v'.\n",
 user.UserName)
  return event, err
 }
```

```
 output, err := h.dynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName:                 aws.String(tableName),
  FilterExpression:          expr.Filter(),
  ExpressionAttributeNames:  expr.Names(),
  ExpressionAttributeValues: expr.Values(),
 })
 if err != nil {
  log.Printf("Error looking up user '%v'.\n", user.UserName)
  return event, err
 }
 if output.Items == nil || len(output.Items) == 0 {
  log.Printf("User '%v' not found, not migrating user.\n", user.UserName)
  return event, err
 }

 var users []UserInfo
 err = attributevalue.UnmarshalListOfMaps(output.Items, &users)
 if err != nil {
  log.Printf("Couldn't unmarshal DynamoDB items. Here's why: %v\n", err)
  return event, err
 }

 user = users[0]
 log.Printf("UserName '%v' found with email %v. User is migrated and must reset
 password.\n", user.UserName, user.UserEmail)
 event.CognitoEventUserPoolsMigrateUserResponse.UserAttributes =
 map[string]string{
  "email":          user.UserEmail,
  "email_verified": "true", // email_verified is required for the forgot password
 flow.
 }
 event.CognitoEventUserPoolsMigrateUserResponse.FinalUserStatus =
 "RESET_REQUIRED"
 event.CognitoEventUserPoolsMigrateUserResponse.MessageAction = "SUPPRESS"

 return event, err
}

func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
```

```
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

일반적인 작업을 수행하는 구조체를 생성합니다.

```
// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(stackName string) (actions.StackOutputs, error)
 PopulateUserTable(tableName string)
 GetKnownUsers(tableName string) (actions.UserList, error)
 AddKnownUser(tableName string, user actions.User)
 ListRecentLogEvents(functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
```

```
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(stackName string)
 (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(tableName string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(tableName string) (actions.UserList,
 error) {
 knownUsers, err := helper.dynamoActor.Scan(tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(tableName string, user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
```

```
 err := helper.dynamoActor.AddUser(tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Amazon Cognito 작업을 래핑하는 구조체를 생성합니다.

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}
```

```
// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(userPoolId string,
 triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(context.TODO(),
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(context.TODO(),
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
```

```
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(clientId string, userName string, password
 string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(context.TODO(),
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(clientId string, userName string, password
 string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(context.TODO(),
 &cognitoidentityprovider.InitiateAuthInput{
```

```
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(clientId string, userName string)
 (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(context.TODO(),
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(clientId string, code string,
 userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(context.TODO(),
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
```

```
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(userAccessToken string) error {
 _, err := actor.CognitoClient.DeleteUser(context.TODO(),
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(userPoolId string, userName string,
 userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(context.TODO(),
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
```

```
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(userPoolId string, userName
 string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(context.TODO(),
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

DynamoDB 작업을 래핑하는 구조체를 생성합니다.

```
// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(tableName string) error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
```

```
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(context.TODO(),
 &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(tableName string) (UserList, error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(context.TODO(), &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(tableName string, user User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
```

```
 }
 _, err = actor.DynamoClient.PutItem(context.TODO(), &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

CloudWatch Logs 액션을 래핑하는 구조체를 만드세요.

```
type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(functionName string)
 (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(context.TODO(),
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
```

```
func (actor CloudWatchLogsActions) GetLogEvents(functionName string,
 logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(context.TODO(),
 &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

액션을 래핑하는 구조체를 만드세요. AWS CloudFormation

```

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(stackName string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(context.TODO(),
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
```

```
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

리소스를 정리합니다.

```
// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup() {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 \n" +
    "that were created for this scenario.")
  }
 }()
```

```
 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
 HandlerArn: nil}
  }
  err := resources.cognitoActor.UpdateTriggers(resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
   panic(err)
  }
  log.Println("Removed Cognito triggers from user pool.")
 } else {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
 }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 다음 주제를 참조하십시오.
  - ConfirmForgotPassword
  - DeleteUser
  - ForgotPassword
  - InitiateAuth
  - SignUp
  - UpdateUserPool

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. 이 서비스를 SDK와 함께 사용 AWS 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

SDK를 사용하여 MFA가 필요한 Amazon Cognito 사용자 풀에 사용자를 등록합니다. AWS

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 사용자 이름, 암호 및 이메일 주소로 사용자를 가입시키고 확인합니다.
- MFA 애플리케이션을 사용자와 연결하여 다중 인증을 설정합니다.
- 암호와 MFA 코드를 사용하여 로그인합니다.

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
namespace CognitoBasics;

public class CognitoBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon Cognito.
        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<IAmazonCognitoIdentityProvider>()
```

```
            .AddTransient<CognitoWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<CognitoBasics>();

        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        var cognitoWrapper = host.Services.GetRequiredService<CognitoWrapper>();

        Console.WriteLine(new string('-', 80));
        UiMethods.DisplayOverview();
        Console.WriteLine(new string('-', 80));

        // clientId - The app client Id value that you get from the AWS CDK
script.
        var clientId = configuration["ClientId"]; // "*** REPLACE WITH CLIENT ID
VALUE FROM CDK SCRIPT";

        // poolId - The pool Id that you get from the AWS CDK script.
        var poolId = configuration["PoolId"]!; // "*** REPLACE WITH POOL ID VALUE
FROM CDK SCRIPT";
        var userName = configuration["UserName"];
        var password = configuration["Password"];
        var email = configuration["Email"];

        // If the username wasn't set in the configuration file,
        // get it from the user now.
        if (userName is null)
        {
            do
            {
                Console.Write("Username: ");
                userName = Console.ReadLine();
            }
            while (string.IsNullOrEmpty(userName));
        }
        Console.WriteLine($"\nUsername: {userName}");
```

```
        // If the password wasn't set in the configuration file,
        // get it from the user now.
        if (password is null)
        {
            do
            {
                Console.Write("Password: ");
                password = Console.ReadLine();
            }
            while (string.IsNullOrEmpty(password));
        }

        // If the email address wasn't set in the configuration file,
        // get it from the user now.
        if (email is null)
        {
            do
            {
                Console.Write("Email: ");
                email = Console.ReadLine();
            } while (string.IsNullOrEmpty(email));
        }

        // Now sign up the user.
        Console.WriteLine($"\nSigning up {userName} with email address:
{email}");
        await cognitoWrapper.SignUpAsync(clientId, userName, password, email);

        // Add the user to the user pool.
        Console.WriteLine($"Adding {userName} to the user pool");
        await cognitoWrapper.GetAdminUserAsync(userName, poolId);

        UiMethods.DisplayTitle("Get confirmation code");
        Console.WriteLine($"Conformation code sent to {userName}.");
        Console.Write("Would you like to send a new code? (Y/N) ");
        var answer = Console.ReadLine();

        if (answer!.ToLower() == "y")
        {
            await cognitoWrapper.ResendConfirmationCodeAsync(clientId, userName);
            Console.WriteLine("Sending a new confirmation code");
        }
```

```
        Console.Write("Enter confirmation code (from Email): ");
        var code = Console.ReadLine();

        await cognitoWrapper.ConfirmSignupAsync(clientId, code, userName);

        UiMethods.DisplayTitle("Checking status");
        Console.WriteLine($"Rechecking the status of {userName} in the user
 pool");
        await cognitoWrapper.GetAdminUserAsync(userName, poolId);

        Console.WriteLine($"Setting up authenticator for {userName} in the user
 pool");
        var setupResponse = await cognitoWrapper.InitiateAuthAsync(clientId,
 userName, password);

        var setupSession = await
 cognitoWrapper.AssociateSoftwareTokenAsync(setupResponse.Session);
        Console.Write("Enter the 6-digit code displayed in Google Authenticator:
 ");
        var setupCode = Console.ReadLine();

        var setupResult = await
 cognitoWrapper.VerifySoftwareTokenAsync(setupSession, setupCode);
        Console.WriteLine($"Setup status: {setupResult}");

        Console.WriteLine($"Now logging in {userName} in the user pool");
        var authSession = await cognitoWrapper.AdminInitiateAuthAsync(clientId,
 poolId, userName, password);

        Console.Write("Enter a new 6-digit code displayed in Google
 Authenticator: ");
        var authCode = Console.ReadLine();

        var authResult = await
 cognitoWrapper.AdminRespondToAuthChallengeAsync(userName, clientId, authCode,
 authSession, poolId);
        Console.WriteLine($"Authenticated and received access token:
 {authResult.AccessToken}");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Cognito scenario is complete.");
        Console.WriteLine(new string('-', 80));
    }
}
```

```
using System.Net;

namespace CognitoActions;

/// <summary>
/// Methods to perform Amazon Cognito Identity Provider actions.
/// </summary>
public class CognitoWrapper
{
    private readonly IAmazonCognitoIdentityProvider _cognitoService;

    /// <summary>
    /// Constructor for the wrapper class containing Amazon Cognito actions.
    /// </summary>
    /// <param name="cognitoService">The Amazon Cognito client object.</param>
    public CognitoWrapper(IAmazonCognitoIdentityProvider cognitoService)
    {
        _cognitoService = cognitoService;
    }

    /// <summary>
    /// List the Amazon Cognito user pools for an account.
    /// </summary>
    /// <returns>A list of UserPoolDescriptionType objects.</returns>
    public async Task<List<UserPoolDescriptionType>> ListUserPoolsAsync()
    {
        var userPools = new List<UserPoolDescriptionType>();

        var userPoolsPaginator = _cognitoService.Paginators.ListUserPools(new
 ListUserPoolsRequest());

        await foreach (var response in userPoolsPaginator.Responses)
        {
            userPools.AddRange(response.UserPools);
        }

        return userPools;
    }


    /// <summary>
    /// Get a list of users for the Amazon Cognito user pool.
```

```
    /// </summary>
    /// <param name="userPoolId">The user pool ID.</param>
    /// <returns>A list of users.</returns>
    public async Task<List<UserType>> ListUsersAsync(string userPoolId)
    {
        var request = new ListUsersRequest
        {
            UserPoolId = userPoolId
        };

        var users = new List<UserType>();

        var usersPaginator = _cognitoService.Paginators.ListUsers(request);
        await foreach (var response in usersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
    }


    /// <summary>
    /// Respond to an admin authentication challenge.
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="clientId">The client ID.</param>
    /// <param name="mfaCode">The multi-factor authentication code.</param>
    /// <param name="session">The current application session.</param>
    /// <param name="clientId">The user pool ID.</param>
    /// <returns>The result of the authentication response.</returns>
    public async Task<AuthenticationResultType> AdminRespondToAuthChallengeAsync(
        string userName,
        string clientId,
        string mfaCode,
        string session,
        string userPoolId)
    {
        Console.WriteLine("SOFTWARE_TOKEN_MFA challenge is generated");

        var challengeResponses = new Dictionary<string, string>();
        challengeResponses.Add("USERNAME", userName);
        challengeResponses.Add("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
```

```
        var respondToAuthChallengeRequest = new
AdminRespondToAuthChallengeRequest
        {
            ChallengeName = ChallengeNameType.SOFTWARE_TOKEN_MFA,
            ClientId = clientId,
            ChallengeResponses = challengeResponses,
            Session = session,
            UserPoolId = userPoolId,
        };

        var response = await
_cognitoService.AdminRespondToAuthChallengeAsync(respondToAuthChallengeRequest);
        Console.WriteLine($"Response to Authentication
{response.AuthenticationResult.TokenType}");
        return response.AuthenticationResult;
    }


    /// <summary>
    /// Verify the TOTP and register for MFA.
    /// </summary>
    /// <param name="session">The name of the session.</param>
    /// <param name="code">The MFA code.</param>
    /// <returns>The status of the software token.</returns>
    public async Task<VerifySoftwareTokenResponseType>
VerifySoftwareTokenAsync(string session, string code)
    {
        var tokenRequest = new VerifySoftwareTokenRequest
        {
            UserCode = code,
            Session = session,
        };

        var verifyResponse = await
_cognitoService.VerifySoftwareTokenAsync(tokenRequest);

        return verifyResponse.Status;
    }


    /// <summary>
    /// Get an MFA token to authenticate the user with the authenticator.
    /// </summary>
    /// <param name="session">The session name.</param>
```

```
    /// <returns>The session name.</returns>
    public async Task<string> AssociateSoftwareTokenAsync(string session)
    {
        var softwareTokenRequest = new AssociateSoftwareTokenRequest
        {
            Session = session,
        };

        var tokenResponse = await
_cognitoService.AssociateSoftwareTokenAsync(softwareTokenRequest);
        var secretCode = tokenResponse.SecretCode;

        Console.WriteLine($"Use the following secret code to set up the
authenticator: {secretCode}");

        return tokenResponse.Session;
    }


    /// <summary>
    /// Initiate an admin auth request.
    /// </summary>
    /// <param name="clientId">The client ID to use.</param>
    /// <param name="userPoolId">The ID of the user pool.</param>
    /// <param name="userName">The username to authenticate.</param>
    /// <param name="password">The user's password.</param>
    /// <returns>The session to use in challenge-response.</returns>
    public async Task<string> AdminInitiateAuthAsync(string clientId, string
userPoolId, string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var request = new AdminInitiateAuthRequest
        {
            ClientId = clientId,
            UserPoolId = userPoolId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.AdminInitiateAuthAsync(request);
        return response.Session;
```

```
    }

    /// <summary>
    /// Initiate authorization.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The name of the user who is authenticating.</
param>
    /// <param name="password">The password for the user who is authenticating.</
param>
    /// <returns>The response from the initiate auth request.</returns>
    public async Task<InitiateAuthResponse> InitiateAuthAsync(string clientId,
 string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var authRequest = new InitiateAuthRequest

        {
            ClientId = clientId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.InitiateAuthAsync(authRequest);
        Console.WriteLine($"Result Challenge is : {response.ChallengeName}");

        return response;
    }

    /// <summary>
    /// Confirm that the user has signed up.
    /// </summary>
    /// <param name="clientId">The Id of this application.</param>
    /// <param name="code">The confirmation code sent to the user.</param>
    /// <param name="userName">The username.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ConfirmSignupAsync(string clientId, string code,
 string userName)
    {
        var signUpRequest = new ConfirmSignUpRequest
        {
```

```
            ClientId = clientId,
            ConfirmationCode = code,
            Username = userName,
        };

        var response = await _cognitoService.ConfirmSignUpAsync(signUpRequest);
        if (response.HttpStatusCode == HttpStatusCode.OK)
        {
            Console.WriteLine($"{userName} was confirmed");
            return true;
        }
        return false;
    }


    /// <summary>
    /// Initiates and confirms tracking of the device.
    /// </summary>
    /// <param name="accessToken">The user's access token.</param>
    /// <param name="deviceKey">The key of the device from Amazon Cognito.</
param>
    /// <param name="deviceName">The device name.</param>
    /// <returns></returns>
    public async Task<bool> ConfirmDeviceAsync(string accessToken, string
 deviceKey, string deviceName)
    {
        var request = new ConfirmDeviceRequest
        {
            AccessToken = accessToken,
            DeviceKey = deviceKey,
            DeviceName = deviceName
        };

        var response = await _cognitoService.ConfirmDeviceAsync(request);
        return response.UserConfirmationNecessary;
    }


    /// <summary>
    /// Send a new confirmation code to a user.
    /// </summary>
    /// <param name="clientId">The Id of the client application.</param>
    /// <param name="userName">The username of user who will receive the code.</
param>
```

```
    /// <returns>The delivery details.</returns>
    public async Task<CodeDeliveryDetailsType> ResendConfirmationCodeAsync(string
clientId, string userName)
    {
        var codeRequest = new ResendConfirmationCodeRequest
        {
            ClientId = clientId,
            Username = userName,
        };

        var response = await
_cognitoService.ResendConfirmationCodeAsync(codeRequest);

        Console.WriteLine($"Method of delivery is
{response.CodeDeliveryDetails.DeliveryMedium}");

        return response.CodeDeliveryDetails;
    }


    /// <summary>
    /// Get the specified user from an Amazon Cognito user pool with
administrator access.
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="poolId">The Id of the Amazon Cognito user pool.</param>
    /// <returns>Async task.</returns>
    public async Task<UserStatusType> GetAdminUserAsync(string userName, string
poolId)
    {
        AdminGetUserRequest userRequest = new AdminGetUserRequest
        {
            Username = userName,
            UserPoolId = poolId,
        };

        var response = await _cognitoService.AdminGetUserAsync(userRequest);

        Console.WriteLine($"User status {response.UserStatus}");
        return response.UserStatus;
    }


    /// <summary>
```

```
    /// Sign up a new user.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The username to use.</param>
    /// <param name="password">The user's password.</param>
    /// <param name="email">The email address of the user.</param>
    /// <returns>A Boolean value indicating whether the user was confirmed.</
returns>
    public async Task<bool> SignUpAsync(string clientId, string userName, string
 password, string email)
    {
        var userAttrs = new AttributeType
        {
            Name = "email",
            Value = email,
        };

        var userAttrsList = new List<AttributeType>();

        userAttrsList.Add(userAttrs);

        var signUpRequest = new SignUpRequest
        {
            UserAttributes = userAttrsList,
            Username = userName,
            ClientId = clientId,
            Password = password
        };

        var response = await _cognitoService.SignUpAsync(signUpRequest);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 다음 주제를 참조하십시오.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken

- ConfirmDevice
- ConfirmSignUp
- InitiateAuth
- ListUsers
- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

C++

SDK for C++

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Scenario that adds a user to an Amazon Cognito user pool.
/*!
  \sa gettingStartedWithUserPools()
  \param clientID: Client ID associated with an Amazon Cognito user pool.
  \param userPoolID: An Amazon Cognito user pool ID.
  \param clientConfig: Aws client configuration.
  \return bool: Successful completion.
 */
bool AwsDoc::Cognito::gettingStartedWithUserPools(const Aws::String &clientID,
                                                  const Aws::String &userPoolID,
                                                  const
 Aws::Client::ClientConfiguration &clientConfig) {
    printAsterisksLine();
    std::cout
            << "Welcome to the Amazon Cognito example scenario."
```

```
            << std::endl;
    printAsterisksLine();

    std::cout
            << "This scenario will add a user to an Amazon Cognito user pool."
            << std::endl;
    const Aws::String userName = askQuestion("Enter a new username: ");
    const Aws::String password = askQuestion("Enter a new password: ");
    const Aws::String email = askQuestion("Enter a valid email for the user: ");

    std::cout << "Signing up " << userName << std::endl;

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);
    bool userExists = false;
    do {
        // 1. Add a user with a username, password, and email address.
        Aws::CognitoIdentityProvider::Model::SignUpRequest request;
        request.AddUserAttributes(
                Aws::CognitoIdentityProvider::Model::AttributeType().WithName(
                        "email").WithValue(email));
        request.SetUsername(userName);
        request.SetPassword(password);
        request.SetClientId(clientID);
        Aws::CognitoIdentityProvider::Model::SignUpOutcome outcome =
                client.SignUp(request);

        if (outcome.IsSuccess()) {
            std::cout << "The signup request for " << userName << " was
successful."
                      << std::endl;
        }
        else if (outcome.GetError().GetErrorType() ==

Aws::CognitoIdentityProvider::CognitoIdentityProviderErrors::USERNAME_EXISTS) {
            std::cout
                    << "The username already exists. Please enter a different
username."
                    << std::endl;
            userExists = true;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::SignUpRequest. "
                      << outcome.GetError().GetMessage()
```

```
                    << std::endl;
            return false;
        }
    } while (userExists);

    printAsterisksLine();
    std::cout << "Retrieving status of " << userName << " in the user pool."
              << std::endl;
    // 2. Confirm that the user was added to the user pool.
    if (!checkAdminUserStatus(userName, userPoolID, client)) {
        return false;
    }

    std::cout << "A confirmation code was sent to " << email << "." << std::endl;

    bool resend = askYesNoQuestion("Would you like to send a new code? (y/n) ");
    if (resend) {
        // Request a resend of the confirmation code to the email address.
(ResendConfirmationCode)
        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeRequest
request;
        request.SetUsername(userName);
        request.SetClientId(clientID);

        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeOutcome
outcome =
                client.ResendConfirmationCode(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "CognitoIdentityProvider::ResendConfirmationCode was
successful."
                    << std::endl;
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::ResendConfirmationCode. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();
```

```
   {
       // 4. Send the confirmation code that's received in the email.
(ConfirmSignUp)
       const Aws::String confirmationCode = askQuestion(
               "Enter the confirmation code that was emailed: ");
       Aws::CognitoIdentityProvider::Model::ConfirmSignUpRequest request;
       request.SetClientId(clientID);
       request.SetConfirmationCode(confirmationCode);
       request.SetUsername(userName);

       Aws::CognitoIdentityProvider::Model::ConfirmSignUpOutcome outcome =
               client.ConfirmSignUp(request);

       if (outcome.IsSuccess()) {
           std::cout << "ConfirmSignup was Successful."
                     << std::endl;
       }
       else {
           std::cerr << "Error with CognitoIdentityProvider::ConfirmSignUp. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
           return false;
       }
   }

   std::cout << "Rechecking the status of " << userName << " in the user pool."
             << std::endl;
   if (!checkAdminUserStatus(userName, userPoolID, client)) {
       return false;
   }

   printAsterisksLine();

   std::cout << "Initiating authorization using the username and password."
             << std::endl;

   Aws::String session;
   // 5. Initiate authorization with username and password. (AdminInitiateAuth)
   if (!adminInitiateAuthorization(clientID, userPoolID,  userName, password,
session, client)) {
       return false;
   }
```

```
    printAsterisksLine();

    std::cout
            << "Starting setup of time-based one-time password (TOTP) multi-
factor authentication (MFA)."
            << std::endl;

    {
        // 6. Request a setup key for one-time password (TOTP)
        //    multi-factor authentication (MFA). (AssociateSoftwareToken)
        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenRequest
 request;
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenOutcome
 outcome =
                client.AssociateSoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "Enter this setup key into an authenticator app, for
 example Google Authenticator."
                    << std::endl;
            std::cout << "Setup key: " << outcome.GetResult().GetSecretCode()
                      << std::endl;
#ifdef USING_QR
            printAsterisksLine();
            std::cout << "\nOr scan the QR code in the file '" << QR_CODE_PATH <<
 "."
                      << std::endl;

            saveQRCode(std::string("otpauth://totp/") + userName + "?secret=" +
                       outcome.GetResult().GetSecretCode());
#endif // USING_QR
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with
 CognitoIdentityProvider::AssociateSoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }
```

```
    askQuestion("Type enter to continue...", alwaysTrueTest);

    printAsterisksLine();

    {
        Aws::String userCode = askQuestion(
                "Enter the 6 digit code displayed in the authenticator app: ");

        //  7. Send the MFA code copied from an authenticator app.
(VerifySoftwareToken)
        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenRequest request;
        request.SetUserCode(userCode);
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenOutcome outcome =
                client.VerifySoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout << "Verification of the code was successful."
                      << std::endl;
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::VerifySoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();
    std::cout << "You have completed the MFA authentication setup." << std::endl;
    std::cout << "Now, sign in." << std::endl;

    // 8. Initiate authorization again with username and password.
(AdminInitiateAuth)
    if (!adminInitiateAuthorization(clientID, userPoolID, userName, password,
session, client)) {
        return false;
    }

    Aws::String accessToken;
    {
```

```
        Aws::String mfaCode = askQuestion(
                "Re-enter the 6 digit code displayed in the authenticator app:
");

        // 9. Send a new MFA code copied from an authenticator app.
(AdminRespondToAuthChallenge)
        Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeRequest
request;
        request.AddChallengeResponses("USERNAME", userName);
        request.AddChallengeResponses("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
        request.SetChallengeName(

Aws::CognitoIdentityProvider::Model::ChallengeNameType::SOFTWARE_TOKEN_MFA);
        request.SetClientId(clientID);
        request.SetUserPoolId(userPoolID);
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeOutcome
outcome =
                client.AdminRespondToAuthChallenge(request);

        if (outcome.IsSuccess()) {
            std::cout << "Here is the response to the challenge.\n" <<

outcome.GetResult().GetAuthenticationResult().Jsonize().View().WriteReadable()
                      << std::endl;

            accessToken =
outcome.GetResult().GetAuthenticationResult().GetAccessToken();
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::AdminRespondToAuthChallenge. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }

        std::cout << "You have successfully added a user to Amazon Cognito."
                  << std::endl;
    }

    if (askYesNoQuestion("Would you like to delete the user that you just added?
(y/n) ")) {
```

```
        // 10. Delete the user that you just added. (DeleteUser)
        Aws::CognitoIdentityProvider::Model::DeleteUserRequest request;
        request.SetAccessToken(accessToken);

        Aws::CognitoIdentityProvider::Model::DeleteUserOutcome outcome =
                client.DeleteUser(request);

        if (outcome.IsSuccess()) {
            std::cout << "The user " << userName << " was deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::DeleteUser. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    return true;
}

//! Routine which checks the user status in an Amazon Cognito user pool.
/*!
 \sa checkAdminUserStatus()
 \param userName: A username.
 \param userPoolID: An Amazon Cognito user pool ID.
 \return bool: Successful completion.
 */
bool AwsDoc::Cognito::checkAdminUserStatus(const Aws::String &userName,
                                           const Aws::String &userPoolID,
                                           const
 Aws::CognitoIdentityProvider::CognitoIdentityProviderClient &client) {
    Aws::CognitoIdentityProvider::Model::AdminGetUserRequest request;
    request.SetUsername(userName);
    request.SetUserPoolId(userPoolID);

    Aws::CognitoIdentityProvider::Model::AdminGetUserOutcome outcome =
            client.AdminGetUser(request);

    if (outcome.IsSuccess()) {
        std::cout << "The status for " << userName << " is " <<

 Aws::CognitoIdentityProvider::Model::UserStatusTypeMapper::GetNameForUserStatusType(
                          outcome.GetResult().GetUserStatus()) << std::endl;
```

```
        std::cout << "Enabled is " << outcome.GetResult().GetEnabled() <<
 std::endl;
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminGetUser. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which starts authorization of an Amazon Cognito user.
//! This routine requires administrator credentials.
/*!
 \sa adminInitiateAuthorization()
 \param clientID: Client ID of tracked device.
 \param userPoolID: An Amazon Cognito user pool ID.
 \param userName: A username.
 \param password: A password.
 \param sessionResult: String to receive a session token.
 \return bool: Successful completion.
 */
bool AwsDoc::Cognito::adminInitiateAuthorization(const Aws::String &clientID,
                                                 const Aws::String &userPoolID,
                                                 const Aws::String &userName,
                                                 const Aws::String &password,
                                                 Aws::String &sessionResult,
                                                 const
 Aws::CognitoIdentityProvider::CognitoIdentityProviderClient &client) {
    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthRequest request;
    request.SetClientId(clientID);
    request.SetUserPoolId(userPoolID);
    request.AddAuthParameters("USERNAME", userName);
    request.AddAuthParameters("PASSWORD", password);
    request.SetAuthFlow(

 Aws::CognitoIdentityProvider::Model::AuthFlowType::ADMIN_USER_PASSWORD_AUTH);


    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthOutcome outcome =
            client.AdminInitiateAuth(request);

    if (outcome.IsSuccess()) {
```

```
        std::cout << "Call to AdminInitiateAuth was successful." << std::endl;
        sessionResult = outcome.GetResult().GetSession();
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminInitiateAuth. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 다음 주제를 참조하십시오.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

Java

SDK for Java 2.x

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChalleng
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChalleng
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenRequ
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenResp
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AttributeType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AuthFlowType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ChallengeNameType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ConfirmSignUpRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeRequ
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeResp
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.SignUpRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenRequest
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenRespons
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.HashMap;
```

```
import java.util.List;
import java.util.Map;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * TIP: To set up the required user pool, run the AWS Cloud Development Kit (AWS
 * CDK) script provided in this GitHub repo at
 * resources/cdk/cognito_scenario_user_pool_with_mfa.
 *
 * This code example performs the following operations:
 *
 * 1. Invokes the signUp method to sign up a user.
 * 2. Invokes the adminGetUser method to get the user's confirmation status.
 * 3. Invokes the ResendConfirmationCode method if the user requested another
 * code.
 * 4. Invokes the confirmSignUp method.
 * 5. Invokes the AdminInitiateAuth to sign in. This results in being prompted
 * to set up TOTP (time-based one-time password). (The response is
 * “ChallengeName”: “MFA_SETUP”).
 * 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private
 * key. This can be used with Google Authenticator.
 * 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for
 * MFA.
 * 8. Invokes the AdminInitiateAuth to sign in again. This results in being
 * prompted to submit a TOTP (Response: “ChallengeName”: “SOFTWARE_TOKEN_MFA”).
 * 9. Invokes the AdminRespondToAuthChallenge to get back a token.
 */

public class CognitoMVP {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws NoSuchAlgorithmException,
 InvalidKeyException {
        final String usage = """
```

```
                Usage:
                    <clientId> <poolId>

                Where:
                    clientId - The app client Id value that you can get from the
AWS CDK script.
                    poolId - The pool Id that you can get from the AWS CDK
script.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clientId = args[0];
        String poolId = args[1];
        CognitoIdentityProviderClient identityProviderClient =
CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Cognito example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter your user name");
        Scanner in = new Scanner(System.in);
        String userName = in.nextLine();

        System.out.println("*** Enter your password");
        String password = in.nextLine();

        System.out.println("*** Enter your email");
        String email = in.nextLine();

        System.out.println("1. Signing up " + userName);
        signUp(identityProviderClient, clientId, userName, password, email);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Getting " + userName + " in the user pool");
        getAdminUser(identityProviderClient, userName, poolId);
```

```
        System.out
                .println("*** Conformation code sent to " + userName + ". Would
you like to send a new code? (Yes/No)");
        System.out.println(DASHES);

        System.out.println(DASHES);
        String ans = in.nextLine();

        if (ans.compareTo("Yes") == 0) {
            resendConfirmationCode(identityProviderClient, clientId, userName);
            System.out.println("3. Sending a new confirmation code");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Enter confirmation code that was emailed");
        String code = in.nextLine();
        confirmSignUp(identityProviderClient, clientId, code, userName);
        System.out.println("Rechecking the status of " + userName + " in the user
pool");
        getAdminUser(identityProviderClient, userName, poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Invokes the initiateAuth to sign in");
        AdminInitiateAuthResponse authResponse =
initiateAuth(identityProviderClient, clientId, userName, password,
                poolId);
        String mySession = authResponse.session();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Invokes the AssociateSoftwareToken method to
generate a TOTP key");
        String newSession = getSecretForAppMFA(identityProviderClient,
mySession);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter the 6-digit code displayed in Google
Authenticator");
        String myCode = in.nextLine();
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("7. Verify the TOTP and register for MFA");
        verifyTOTP(identityProviderClient, newSession, myCode);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Re-enter a 6-digit code displayed in Google
Authenticator");
        String mfaCode = in.nextLine();
        AdminInitiateAuthResponse authResponse1 =
initiateAuth(identityProviderClient, clientId, userName, password,
                poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9.  Invokes the AdminRespondToAuthChallenge");
        String session2 = authResponse1.session();
        adminRespondToAuthChallenge(identityProviderClient, userName, clientId,
mfaCode, session2);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("All Amazon Cognito operations were successfully
performed");
        System.out.println(DASHES);
    }

    // Respond to an authentication challenge.
    public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
            String userName, String clientId, String mfaCode, String session) {
        System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
        Map<String, String> challengeResponses = new HashMap<>();

        challengeResponses.put("USERNAME", userName);
        challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

        AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
                .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
                .clientId(clientId)
                .challengeResponses(challengeResponses)
                .session(session)
```

```
                .build();

        AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
                .adminRespondToAuthChallenge(respondToAuthChallengeRequest);

System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
                + respondToAuthChallengeResult.authenticationResult());
    }

    // Verify the TOTP and register for MFA.
    public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
        try {
            VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
                    .userCode(code)
                    .session(session)
                    .build();

            VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
            System.out.println("The status of the token is " +
verifyResponse.statusAsString());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
            String clientId, String userName, String password, String userPoolId)
{
        try {
            Map<String, String> authParameters = new HashMap<>();
            authParameters.put("USERNAME", userName);
            authParameters.put("PASSWORD", password);

            AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
                    .clientId(clientId)
                    .userPoolId(userPoolId)
```

```
                    .authParameters(authParameters)
                    .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                    .build();

            AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
            System.out.println("Result Challenge is : " +
response.challengeName());
            return response;

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }

    public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
        AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
                .session(session)
                .build();

        AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
                .associateSoftwareToken(softwareTokenRequest);
        String secretCode = tokenResponse.secretCode();
        System.out.println("Enter this token into Google Authenticator");
        System.out.println(secretCode);
        return tokenResponse.session();
    }

    public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
            String userName) {
        try {
            ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
                    .clientId(clientId)
                    .confirmationCode(code)
                    .username(userName)
                    .build();

            identityProviderClient.confirmSignUp(signUpRequest);
```

```
            System.out.println(userName + " was confirmed");

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
            String userName) {
        try {
            ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
                    .clientId(clientId)
                    .username(userName)
                    .build();

            ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
            System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void signUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String userName,
            String password, String email) {
        AttributeType userAttrs = AttributeType.builder()
                .name("email")
                .value(email)
                .build();

        List<AttributeType> userAttrsList = new ArrayList<>();
        userAttrsList.add(userAttrs);
        try {
            SignUpRequest signUpRequest = SignUpRequest.builder()
                    .userAttributes(userAttrsList)
                    .username(userName)
                    .clientId(clientId)
```

```
                    .password(password)
                    .build();

            identityProviderClient.signUp(signUpRequest);
            System.out.println("User has been signed up ");

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getAdminUser(CognitoIdentityProviderClient
 identityProviderClient, String userName,
            String poolId) {
        try {
            AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
                    .username(userName)
                    .userPoolId(poolId)
                    .build();

            AdminGetUserResponse response =
 identityProviderClient.adminGetUser(userRequest);
            System.out.println("User status " + response.userStatusAsString());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하십시오.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp

- InitiateAuth
- ListUsers
- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

최상의 경험을 위해 GitHub 리포지토리를 복제하고 이 예제를 실행하세요. 다음 코드는 전체 예제 애플리케이션의 샘플을 나타냅니다.

```
import { log } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { signUp } from "../../../actions/sign-up.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getSecondValuesFromEntries } from "@aws-doc-sdk-examples/lib/utils/util-
csv.js";

const validateClient = (clientId) => {
  if (!clientId) {
    throw new Error(
      `App client id is missing. Did you run 'create-user-pool'?`,
    );
  }
};

const validateUser = (username, password, email) => {
  if (!(username && password && email)) {
    throw new Error(
      `Username, password, and email must be provided as arguments to the 'sign-
up' command.`,
```

```
    );
  }
};

const signUpHandler = async (commands) => {
  const [_, username, password, email] = commands;

  try {
    validateUser(username, password, email);
    /**
     * @type {string[]}
     */
    const values = getSecondValuesFromEntries(FILE_USER_POOLS);
    const clientId = values[0];
    validateClient(clientId);
    log(`Signing up.`);
    await signUp({ clientId, username, password, email });
    log(`Signed up. A confirmation email has been sent to: ${email}.`);
    log(`Run 'confirm-sign-up ${username} <code>' to confirm your account.`);
  } catch (err) {
    log(err);
  }
};

export { signUpHandler };

const signUp = ({ clientId, username, password, email }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new SignUpCommand({
    ClientId: clientId,
    Username: username,
    Password: password,
    UserAttributes: [{ Name: "email", Value: email }],
  });

  return client.send(command);
};

import { log } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { confirmSignUp } from "../../../actions/confirm-sign-up.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getSecondValuesFromEntries } from "@aws-doc-sdk-examples/lib/utils/util-
csv.js";
```

```
const validateClient = (clientId) => {
  if (!clientId) {
    throw new Error(
      `App client id is missing. Did you run 'create-user-pool'?`,
    );
  }
};

const validateUser = (username) => {
  if (!username) {
    throw new Error(
      `Username name is missing. It must be provided as an argument to the
 'confirm-sign-up' command.`,
    );
  }
};

const validateCode = (code) => {
  if (!code) {
    throw new Error(
      `Verification code is missing. It must be provided as an argument to the
 'confirm-sign-up' command.`,
    );
  }
};

const confirmSignUpHandler = async (commands) => {
  const [_, username, code] = commands;

  try {
    validateUser(username);
    validateCode(code);
    /**
     * @type {string[]}
     */
    const values = getSecondValuesFromEntries(FILE_USER_POOLS);
    const clientId = values[0];
    validateClient(clientId);
    log(`Confirming user.`);
    await confirmSignUp({ clientId, username, code });
    log(
      `User confirmed. Run 'admin-initiate-auth ${username} <password>' to sign
 in.`,
```

```
    );
  } catch (err) {
    log(err);
  }
};

export { confirmSignUpHandler };

const confirmSignUp = ({ clientId, username, code }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ConfirmSignUpCommand({
    ClientId: clientId,
    Username: username,
    ConfirmationCode: code,
  });

  return client.send(command);
};

import qrcode from "qrcode-terminal";
import { log } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { adminInitiateAuth } from "../../../actions/admin-initiate-auth.js";
import { associateSoftwareToken } from "../../../actions/associate-software-
token.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getFirstEntry } from "@aws-doc-sdk-examples/lib/utils/util-csv.js";

const handleMfaSetup = async (session, username) => {
  const { SecretCode, Session } = await associateSoftwareToken(session);

  // Store the Session for use with 'VerifySoftwareToken'.
  process.env.SESSION = Session;

  console.log(
    "Scan this code in your preferred authenticator app, then run 'verify-
software-token' to finish the setup.",
  );
  qrcode.generate(
    `otpauth://totp/${username}?secret=${SecretCode}`,
    { small: true },
    console.log,
  );
};
```

```
const handleSoftwareTokenMfa = (session) => {
  // Store the Session for use with 'AdminRespondToAuthChallenge'.
  process.env.SESSION = session;
};

const validateClient = (id) => {
  if (!id) {
    throw new Error(
      `User pool client id is missing. Did you run 'create-user-pool'?`,
    );
  }
};

const validateId = (id) => {
  if (!id) {
    throw new Error(`User pool id is missing. Did you run 'create-user-pool'?`);
  }
};

const validateUser = (username, password) => {
  if (!(username && password)) {
    throw new Error(
      `Username and password must be provided as arguments to the 'admin-
initiate-auth' command.`,
    );
  }
};

const adminInitiateAuthHandler = async (commands) => {
  const [_, username, password] = commands;

  try {
    validateUser(username, password);

    const [userPoolId, clientId] = getFirstEntry(FILE_USER_POOLS);
    validateId(userPoolId);
    validateClient(clientId);

    log("Signing in.");
    const { ChallengeName, Session } = await adminInitiateAuth({
      clientId,
      userPoolId,
      username,
```

```
      password,
    });

    if (ChallengeName === "MFA_SETUP") {
      log("MFA setup is required.");
      return handleMfaSetup(Session, username);
    }

    if (ChallengeName === "SOFTWARE_TOKEN_MFA") {
      handleSoftwareTokenMfa(Session);
      log(`Run 'admin-respond-to-auth-challenge ${username} <totp>'`);
    }
  } catch (err) {
    log(err);
  }
};

export { adminInitiateAuthHandler };

const adminInitiateAuth = ({ clientId, userPoolId, username, password }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new AdminInitiateAuthCommand({
    ClientId: clientId,
    UserPoolId: userPoolId,
    AuthFlow: AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
    AuthParameters: { USERNAME: username, PASSWORD: password },
  });

  return client.send(command);
};

import { log } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { adminRespondToAuthChallenge } from "../../../actions/admin-respond-to-
auth-challenge.js";
import { getFirstEntry } from "@aws-doc-sdk-examples/lib/utils/util-csv.js";
import { FILE_USER_POOLS } from "./constants.js";

const verifyUsername = (username) => {
  if (!username) {
    throw new Error(
      `Username is missing. It must be provided as an argument to the 'admin-
respond-to-auth-challenge' command.`,
    );
```

```
  }
};

const verifyTotp = (totp) => {
  if (!totp) {
    throw new Error(
      `Time-based one-time password (TOTP) is missing. It must be provided as an
 argument to the 'admin-respond-to-auth-challenge' command.`,
    );
  }
};

const storeAccessToken = (token) => {
  process.env.AccessToken = token;
};

const adminRespondToAuthChallengeHandler = async (commands) => {
  const [_, username, totp] = commands;

  try {
    verifyUsername(username);
    verifyTotp(totp);

    const [userPoolId, clientId] = getFirstEntry(FILE_USER_POOLS);
    const session = process.env.SESSION;

    const { AuthenticationResult } = await adminRespondToAuthChallenge({
      clientId,
      userPoolId,
      username,
      totp,
      session,
    });

    storeAccessToken(AuthenticationResult.AccessToken);

    log("Successfully authenticated.");
  } catch (err) {
    log(err);
  }
};

export { adminRespondToAuthChallengeHandler };
```

```
const respondToAuthChallenge = ({
  clientId,
  username,
  session,
  userPoolId,
  code,
}) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new RespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: code,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
};

import { log } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { verifySoftwareToken } from "../../../actions/verify-software-token.js";

const validateTotp = (totp) => {
  if (!totp) {
    throw new Error(
      `Time-based one-time password (TOTP) must be provided to the 'validate-
software-token' command.`,
    );
  }
};
const verifySoftwareTokenHandler = async (commands) => {
  const [_, totp] = commands;

  try {
    validateTotp(totp);

    log("Verifying TOTP.");
    await verifySoftwareToken(totp);
    log("TOTP Verified. Run 'admin-initiate-auth' again to sign-in.");
  } catch (err) {
```

```
    console.log(err);
  }
};

export { verifySoftwareTokenHandler };

const verifySoftwareToken = (totp) => {
  const client = new CognitoIdentityProviderClient({});

  // The 'Session' is provided in the response to 'AssociateSoftwareToken'.
  const session = process.env.SESSION;

  if (!session) {
    throw new Error(
      "Missing a valid Session. Did you run 'admin-initiate-auth'?",
    );
  }

  const command = new VerifySoftwareTokenCommand({
    Session: session,
    UserCode: totp,
  });

  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 다음 주제를 참조하십시오.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp

- VerifySoftwareToken

Kotlin

SDK for Kotlin

> **Note**
>
> 더 많은 내용이 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 TIP: To set up the required user pool, run the AWS Cloud Development
 Kit (AWS CDK) script provided in this GitHub repo at resources/cdk/
cognito_scenario_user_pool_with_mfa.

 This code example performs the following operations:

 1. Invokes the signUp method to sign up a user.
 2. Invokes the adminGetUser method to get the user's confirmation status.
 3. Invokes the ResendConfirmationCode method if the user requested another code.
 4. Invokes the confirmSignUp method.
 5. Invokes the initiateAuth to sign in. This results in being prompted to
 set up TOTP (time-based one-time password). (The response is “ChallengeName”:
 “MFA_SETUP”).
 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private key.
 This can be used with Google Authenticator.
 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for
 MFA.
 8. Invokes the AdminInitiateAuth to sign in again. This results in being
 prompted to submit a TOTP (Response: “ChallengeName”: “SOFTWARE_TOKEN_MFA”).
 9. Invokes the AdminRespondToAuthChallenge to get back a token.
 */
```

```
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <clientId> <poolId>
        Where:
            clientId - The app client Id value that you can get from the AWS CDK
 script.
            poolId - The pool Id that you can get from the AWS CDK script.
    """

    if (args.size != 2) {
        println(usage)
        exitProcess(1)
    }

    val clientId = args[0]
    val poolId = args[1]

    // Use the console to get data from the user.
    println("*** Enter your use name")
    val inOb = Scanner(System.`in`)
    val userName = inOb.nextLine()
    println(userName)

    println("*** Enter your password")
    val password: String = inOb.nextLine()

    println("*** Enter your email")
    val email = inOb.nextLine()

    println("*** Signing up $userName")
    signUp(clientId, userName, password, email)

    println("*** Getting $userName in the user pool")
    getAdminUser(userName, poolId)

    println("*** Conformation code sent to $userName. Would you like to send a
 new code? (Yes/No)")
    val ans = inOb.nextLine()

    if (ans.compareTo("Yes") == 0) {
        println("*** Sending a new confirmation code")
        resendConfirmationCode(clientId, userName)
    }
```

```
    println("*** Enter the confirmation code that was emailed")
    val code = inOb.nextLine()
    confirmSignUp(clientId, code, userName)

    println("*** Rechecking the status of $userName in the user pool")
    getAdminUser(userName, poolId)

    val authResponse = checkAuthMethod(clientId, userName, password, poolId)
    val mySession = authResponse.session
    val newSession = getSecretForAppMFA(mySession)
    println("*** Enter the 6-digit code displayed in Google Authenticator")
    val myCode = inOb.nextLine()

    // Verify the TOTP and register for MFA.
    verifyTOTP(newSession, myCode)
    println("*** Re-enter a 6-digit code displayed in Google Authenticator")
    val mfaCode: String = inOb.nextLine()
    val authResponse1 = checkAuthMethod(clientId, userName, password, poolId)
    val session2 = authResponse1.session
    adminRespondToAuthChallenge(userName, clientId, mfaCode, session2)
}

suspend fun checkAuthMethod(clientIdVal: String, userNameVal: String,
 passwordVal: String, userPoolIdVal: String): AdminInitiateAuthResponse {
    val authParas = mutableMapOf<String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal

    val authRequest = AdminInitiateAuthRequest {
        clientId = clientIdVal
        userPoolId = userPoolIdVal
        authParameters = authParas
        authFlow = AuthFlowType.AdminUserPasswordAuth
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminInitiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}

suspend fun resendConfirmationCode(clientIdVal: String?, userNameVal: String?) {
```

```
    val codeRequest = ResendConfirmationCodeRequest {
        clientId = clientIdVal
        username = userNameVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
    }
}

// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(userName: String, clientIdVal: String?,
 mfaCode: String, sessionVal: String?) {
    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponsesOb = mutableMapOf<String, String>()
    challengeResponsesOb["USERNAME"] = userName
    challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest = AdminRespondToAuthChallengeRequest {
        challengeName = ChallengeNameType.SoftwareTokenMfa
        clientId = clientIdVal
        challengeResponses = challengeResponsesOb
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val respondToAuthChallengeResult =
 identityProviderClient.adminRespondToAuthChallenge(adminRespondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult()
 ${respondToAuthChallengeResult.authenticationResult}")
    }
}

// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(sessionVal: String?, codeVal: String?) {
    val tokenRequest = VerifySoftwareTokenRequest {
        userCode = codeVal
        session = sessionVal
    }
```

```
    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}

suspend fun getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest = AssociateSoftwareTokenRequest {
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val tokenResponse =
 identityProviderClient.associateSoftwareToken(softwareTokenRequest)
        val secretCode = tokenResponse.secretCode
        println("Enter this token into Google Authenticator")
        println(secretCode)
        return tokenResponse.session
    }
}

suspend fun confirmSignUp(clientIdVal: String?, codeVal: String?, userNameVal:
 String?) {
    val signUpRequest = ConfirmSignUpRequest {
        clientId = clientIdVal
        confirmationCode = codeVal
        username = userNameVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
    }
}

suspend fun getAdminUser(userNameVal: String?, poolIdVal: String?) {
    val userRequest = AdminGetUserRequest {
        username = userNameVal
        userPoolId = poolIdVal
    }
```

```
    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminGetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}

suspend fun signUp(clientIdVal: String?, userNameVal: String?, passwordVal:
 String?, emailVal: String?) {
    val userAttrs = AttributeType {
        name = "email"
        value = emailVal
    }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)
    val signUpRequest = SignUpRequest {
        userAttributes = userAttrsList
        username = userNameVal
        clientId = clientIdVal
        password = passwordVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API reference의 다음 주제를 참조하세요.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth

- ListUsers
- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

Python

SDK for Python(Boto3)

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

시나리오에 사용된 Amazon Cognito 함수를 래핑하는 클래스를 만듭니다.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def _secret_hash(self, user_name):
        """
        Calculates a secret hash from a user name and a client secret.
```

```
        :param user_name: The user name to use when calculating the hash.
        :return: The secret hash.
        """
        key = self.client_secret.encode()
        msg = bytes(user_name + self.client_id, "utf-8")
        secret_hash = base64.b64encode(
            hmac.new(key, msg, digestmod=hashlib.sha256).digest()
        ).decode()
        logger.info("Made secret hash for %s: %s.", user_name, secret_hash)
        return secret_hash

    def sign_up_user(self, user_name, password, user_email):
        """
        Signs up a new user with Amazon Cognito. This action prompts Amazon
Cognito
        to send an email to the specified email address. The email contains a
code that
        can be used to confirm the user.

        When the user already exists, the user status is checked to determine
whether
        the user has been confirmed.

        :param user_name: The user name that identifies the new user.
        :param password: The password for the new user.
        :param user_email: The email address for the new user.
        :return: True when the user is already confirmed with Amazon Cognito.
                 Otherwise, false.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "Password": password,
                "UserAttributes": [{"Name": "email", "Value": user_email}],
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.sign_up(**kwargs)
            confirmed = response["UserConfirmed"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "UsernameExistsException":
                response = self.cognito_idp_client.admin_get_user(
```

```
                    UserPoolId=self.user_pool_id, Username=user_name
                )
                logger.warning(
                    "User %s exists and is %s.", user_name,
response["UserStatus"]
                )
                confirmed = response["UserStatus"] == "CONFIRMED"
            else:
                logger.error(
                    "Couldn't sign up %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return confirmed


    def resend_confirmation(self, user_name):
        """
        Prompts Amazon Cognito to resend an email with a new confirmation code.

        :param user_name: The name of the user who will receive the email.
        :return: Delivery information about where the email is sent.
        """
        try:
            kwargs = {"ClientId": self.client_id, "Username": user_name}
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.resend_confirmation_code(**kwargs)
            delivery = response["CodeDeliveryDetails"]
        except ClientError as err:
            logger.error(
                "Couldn't resend confirmation to %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return delivery


    def confirm_user_sign_up(self, user_name, confirmation_code):
```

```
        """
        Confirms a previously created user. A user must be confirmed before they
        can sign in to Amazon Cognito.

        :param user_name: The name of the user to confirm.
        :param confirmation_code: The confirmation code sent to the user's
registered
                                  email address.
        :return: True when the confirmation succeeds.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "ConfirmationCode": confirmation_code,
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            self.cognito_idp_client.confirm_sign_up(**kwargs)
        except ClientError as err:
            logger.error(
                "Couldn't confirm sign up for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return True


    def list_users(self):
        """
        Returns a list of the users in the current user pool.

        :return: The list of users.
        """
        try:
            response =
self.cognito_idp_client.list_users(UserPoolId=self.user_pool_id)
            users = response["Users"]
        except ClientError as err:
            logger.error(
                "Couldn't list users for %s. Here's why: %s: %s",
```

```
                self.user_pool_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return users


    def start_sign_in(self, user_name, password):
        """
        Starts the sign-in process for a user by using administrator credentials.
        This method of signing in is appropriate for code running on a secure
 server.

        If the user pool is configured to require MFA and this is the first sign-
in
        for the user, Amazon Cognito returns a challenge response to set up an
        MFA application. When this occurs, this function gets an MFA secret from
        Amazon Cognito and returns it to the caller.

        :param user_name: The name of the user to sign in.
        :param password: The user's password.
        :return: The result of the sign-in attempt. When sign-in is successful,
 this
                 returns an access token that can be used to get AWS credentials.
 Otherwise,
                 Amazon Cognito returns a challenge to set up an MFA application,
                 or a challenge to enter an MFA code from a registered MFA
 application.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "AuthFlow": "ADMIN_USER_PASSWORD_AUTH",
                "AuthParameters": {"USERNAME": user_name, "PASSWORD": password},
            }
            if self.client_secret is not None:
                kwargs["AuthParameters"]["SECRET_HASH"] =
 self._secret_hash(user_name)
            response = self.cognito_idp_client.admin_initiate_auth(**kwargs)
            challenge_name = response.get("ChallengeName", None)
            if challenge_name == "MFA_SETUP":
```

```
                if (
                    "SOFTWARE_TOKEN_MFA"
                    in response["ChallengeParameters"]["MFAS_CAN_SETUP"]
                ):
                    response.update(self.get_mfa_secret(response["Session"]))
                else:
                    raise RuntimeError(
                        "The user pool requires MFA setup, but the user pool is
not "
                        "configured for TOTP MFA. This example requires TOTP
MFA."
                    )
        except ClientError as err:
            logger.error(
                "Couldn't start sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response


    def get_mfa_secret(self, session):
        """
        Gets a token that can be used to associate an MFA application with the
user.

        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :return: An MFA token that can be used to set up an MFA application.
        """
        try:
            response =
self.cognito_idp_client.associate_software_token(Session=session)
        except ClientError as err:
            logger.error(
                "Couldn't get MFA secret. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response


    def verify_mfa(self, session, user_code):
        """
        Verify a new MFA application that is associated with a user.

        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :param user_code: A code generated by the associated MFA application.
        :return: Status that indicates whether the MFA application is verified.
        """
        try:
            response = self.cognito_idp_client.verify_software_token(
                Session=session, UserCode=user_code
            )
        except ClientError as err:
            logger.error(
                "Couldn't verify MFA. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response


    def respond_to_mfa_challenge(self, user_name, session, mfa_code):
        """
        Responds to a challenge for an MFA code. This completes the second step
of
        a two-factor sign-in. When sign-in is successful, it returns an access
token
        that can be used to get AWS credentials from Amazon Cognito.

        :param user_name: The name of the user who is signing in.
        :param session: Session information returned from a previous call to
initiate
                        authentication.
```

```
        :param mfa_code: A code generated by the associated MFA application.
        :return: The result of the authentication. When successful, this contains
an
                 access token for the user.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "ChallengeName": "SOFTWARE_TOKEN_MFA",
                "Session": session,
                "ChallengeResponses": {
                    "USERNAME": user_name,
                    "SOFTWARE_TOKEN_MFA_CODE": mfa_code,
                },
            }
            if self.client_secret is not None:
                kwargs["ChallengeResponses"]["SECRET_HASH"] = self._secret_hash(
                    user_name
                )
            response =
self.cognito_idp_client.admin_respond_to_auth_challenge(**kwargs)
            auth_result = response["AuthenticationResult"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ExpiredCodeException":
                logger.warning(
                    "Your MFA code has expired or has been used already. You
might have "
                    "to wait a few seconds until your app shows you a new code."
                )
            else:
                logger.error(
                    "Couldn't respond to mfa challenge for %s. Here's why: %s:
%s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return auth_result


    def confirm_mfa_device(
```

```
        self,
        user_name,
        device_key,
        device_group_key,
        device_password,
        access_token,
        aws_srp,
    ):
        """
        Confirms an MFA device to be tracked by Amazon Cognito. When a device is
        tracked, its key and password can be used to sign in without requiring a
new
        MFA code from the MFA application.

        :param user_name: The user that is associated with the device.
        :param device_key: The key of the device, returned by Amazon Cognito.
        :param device_group_key: The group key of the device, returned by Amazon
Cognito.
        :param device_password: The password that is associated with the device.
        :param access_token: The user's access token.
        :param aws_srp: A class that helps with Secure Remote Password (SRP)
                        calculations. The scenario associated with this example
uses
                        the warrant package.
        :return: True when the user must confirm the device. Otherwise, False.
When
                 False, the device is automatically confirmed and tracked.
        """
        srp_helper = aws_srp.AWSSRP(
            username=user_name,
            password=device_password,
            pool_id="_",
            client_id=self.client_id,
            client_secret=None,
            client=self.cognito_idp_client,
        )
        device_and_pw = f"{device_group_key}{device_key}:{device_password}"
        device_and_pw_hash = aws_srp.hash_sha256(device_and_pw.encode("utf-8"))
        salt = aws_srp.pad_hex(aws_srp.get_random(16))
        x_value = aws_srp.hex_to_long(aws_srp.hex_hash(salt +
device_and_pw_hash))
        verifier = aws_srp.pad_hex(pow(srp_helper.val_g, x_value,
srp_helper.big_n))
        device_secret_verifier_config = {
```

```
            "PasswordVerifier": base64.standard_b64encode(
                bytearray.fromhex(verifier)
            ).decode("utf-8"),
            "Salt":
base64.standard_b64encode(bytearray.fromhex(salt)).decode("utf-8"),
        }
        try:
            response = self.cognito_idp_client.confirm_device(
                AccessToken=access_token,
                DeviceKey=device_key,
                DeviceSecretVerifierConfig=device_secret_verifier_config,
            )
            user_confirm = response["UserConfirmationNecessary"]
        except ClientError as err:
            logger.error(
                "Couldn't confirm mfa device %s. Here's why: %s: %s",
                device_key,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return user_confirm


    def sign_in_with_tracked_device(
        self,
        user_name,
        password,
        device_key,
        device_group_key,
        device_password,
        aws_srp,
    ):
        """
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA
code.

        Signing in with a tracked device requires that the client respond to the
SRP
        protocol. The scenario associated with this example uses the warrant
package
        to help with SRP calculations.
```

```
        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains
 an
                 access token for the user.
        """
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name,
                password=device_password,
                pool_id="_",
                client_id=self.client_id,
                client_secret=None,
                client=self.cognito_idp_client,
            )

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id,
                AuthFlow="USER_PASSWORD_AUTH",
                AuthParameters={
                    "USERNAME": user_name,
                    "PASSWORD": password,
                    "DEVICE_KEY": device_key,
                },
            )
            if response_init["ChallengeName"] != "DEVICE_SRP_AUTH":
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
 {response_init['ChallengeName']}."
                )

            auth_params = srp_helper.get_auth_params()
            auth_params["DEVICE_KEY"] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
```

```
                ChallengeName="DEVICE_SRP_AUTH",
                ChallengeResponses=auth_params,
            )
            if response_auth["ChallengeName"] != "DEVICE_PASSWORD_VERIFIER":
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}."
                )

            challenge_params = response_auth["ChallengeParameters"]
            challenge_params["USER_ID_FOR_SRP"] = device_group_key + device_key
            cr = srp_helper.process_challenge(challenge_params, {"USERNAME":
 user_name})
            cr["USERNAME"] = user_name
            cr["DEVICE_KEY"] = device_key
            response_verifier =
 self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_PASSWORD_VERIFIER",
                ChallengeResponses=cr,
            )
            auth_tokens = response_verifier["AuthenticationResult"]
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return auth_tokens
```

시나리오를 실행하는 클래스를 생성합니다. 이 예시에서는 Amazon Cognito에서 추적할 MFA 디바이스를 등록하고 추적된 디바이스의 암호 및 정보를 사용하여 로그인하는 방법도 보여줍니다. 이렇게 하면 새 MFA 코드를 입력할 필요가 없습니다.

```
def run_scenario(cognito_idp_client, user_pool_id, client_id):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
```

```
    print("-" * 88)
    print("Welcome to the Amazon Cognito user signup with MFA demo.")
    print("-" * 88)

    cog_wrapper = CognitoIdentityProviderWrapper(
        cognito_idp_client, user_pool_id, client_id
    )

    user_name = q.ask("Let's sign up a new user. Enter a user name: ",
q.non_empty)
    password = q.ask("Enter a password for the user: ", q.non_empty)
    email = q.ask("Enter a valid email address that you own: ", q.non_empty)
    confirmed = cog_wrapper.sign_up_user(user_name, password, email)
    while not confirmed:
        print(
            f"User {user_name} requires confirmation. Check {email} for "
            f"a verification code."
        )
        confirmation_code = q.ask("Enter the confirmation code from the email: ")
        if not confirmation_code:
            if q.ask("Do you need another confirmation code (y/n)? ",
q.is_yesno):
                delivery = cog_wrapper.resend_confirmation(user_name)
                print(
                    f"Confirmation code sent by {delivery['DeliveryMedium']} "
                    f"to {delivery['Destination']}."
                )
        else:
            confirmed = cog_wrapper.confirm_user_sign_up(user_name,
confirmation_code)
    print(f"User {user_name} is confirmed and ready to use.")
    print("-" * 88)

    print("Let's get a list of users in the user pool.")
    q.ask("Press Enter when you're ready.")
    users = cog_wrapper.list_users()
    if users:
        print(f"Found {len(users)} users:")
        pp(users)
    else:
        print("No users found.")
    print("-" * 88)
```

```
    print("Let's sign in and get an access token.")
    auth_tokens = None
    challenge = "ADMIN_USER_PASSWORD_AUTH"
    response = {}
    while challenge is not None:
        if challenge == "ADMIN_USER_PASSWORD_AUTH":
            response = cog_wrapper.start_sign_in(user_name, password)
            challenge = response["ChallengeName"]
        elif response["ChallengeName"] == "MFA_SETUP":
            print("First, we need to set up an MFA application.")
            qr_img = qrcode.make(
                f"otpauth://totp/{user_name}?secret={response['SecretCode']}"
            )
            qr_img.save("qr.png")
            q.ask(
                "Press Enter to see a QR code on your screen. Scan it into an MFA
"
                "application, such as Google Authenticator."
            )
            webbrowser.open("qr.png")
            mfa_code = q.ask(
                "Enter the verification code from your MFA application: ",
q.non_empty
            )
            response = cog_wrapper.verify_mfa(response["Session"], mfa_code)
            print(f"MFA device setup {response['Status']}")
            print("Now that an MFA application is set up, let's sign in again.")
            print(
                "You might have to wait a few seconds for a new MFA code to
appear in "
                "your MFA application."
            )
            challenge = "ADMIN_USER_PASSWORD_AUTH"
        elif response["ChallengeName"] == "SOFTWARE_TOKEN_MFA":
            auth_tokens = None
            while auth_tokens is None:
                mfa_code = q.ask(
                    "Enter a verification code from your MFA application: ",
q.non_empty
                )
                auth_tokens = cog_wrapper.respond_to_mfa_challenge(
                    user_name, response["Session"], mfa_code
                )
            print(f"You're signed in as {user_name}.")
```

```
            print("Here's your access token:")
            pp(auth_tokens["AccessToken"])
            print("And your device information:")
            pp(auth_tokens["NewDeviceMetadata"])
            challenge = None
        else:
            raise Exception(f"Got unexpected challenge
 {response['ChallengeName']}")
    print("-" * 88)

    device_group_key = auth_tokens["NewDeviceMetadata"]["DeviceGroupKey"]
    device_key = auth_tokens["NewDeviceMetadata"]["DeviceKey"]
    device_password = base64.standard_b64encode(os.urandom(40)).decode("utf-8")

    print("Let's confirm your MFA device so you don't have re-enter MFA tokens
 for it.")
    q.ask("Press Enter when you're ready.")
    cog_wrapper.confirm_mfa_device(
        user_name,
        device_key,
        device_group_key,
        device_password,
        auth_tokens["AccessToken"],
        aws_srp,
    )
    print(f"Your device {device_key} is confirmed.")
    print("-" * 88)

    print(
        f"Now let's sign in as {user_name} from your confirmed device
 {device_key}.\n"
        f"Because this device is tracked by Amazon Cognito, you won't have to re-
enter an MFA code."
    )
    q.ask("Press Enter when ready.")
    auth_tokens = cog_wrapper.sign_in_with_tracked_device(
        user_name, password, device_key, device_group_key, device_password,
 aws_srp
    )
    print("You're signed in. Your access token is:")
    pp(auth_tokens["AccessToken"])
    print("-" * 88)
```

```
    print("Don't forget to delete your user pool when you're done with this
 example.")
    print("\nThanks for watching!")
    print("-" * 88)


def main():
    parser = argparse.ArgumentParser(
        description="Shows how to sign up a new user with Amazon Cognito and
 associate "
        "the user with an MFA application for multi-factor authentication."
    )
    parser.add_argument(
        "user_pool_id", help="The ID of the user pool to use for the example."
    )
    parser.add_argument(
        "client_id", help="The ID of the client application to use for the
 example."
    )
    args = parser.parse_args()
    try:
        run_scenario(boto3.client("cognito-idp"), args.user_pool_id,
 args.client_id)
    except Exception:
        logging.exception("Something went wrong with the demo.")


if __name__ == "__main__":
    main()
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 GetRoutingControlState를 참조하십시오.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth

- ListUsers
- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오이 서비스를 SDK와 함께 사용 AWS. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

SDK를 사용한 Amazon Cognito 사용자 인증 후 Lambda 함수를 사용하여 사용자 지정 활동 데이터를 작성합니다. AWS

다음 코드 예제는 Amazon Cognito 사용자 인증 후 Lambda 함수를 사용하여 사용자 지정 작업 데이터를 쓰는 방법을 보여줍니다.

- 관리자 함수를 사용하여 사용자 풀에 사용자를 추가합니다.
- `PostAuthentication` 트리거에 대해 Lambda 함수를 호출하도록 사용자 풀을 구성합니다.
- 새로운 사용자를 Amazon Cognito에 로그인시킵니다.
- Lambda 함수는 로그 및 DynamoDB 테이블에 사용자 지정 정보를 CloudWatch 기록합니다.
- DynamoDB 테이블에서 사용자 지정 데이터를 가져오기 및 표시한 다음 리소스를 정리합니다.

Go

SDK for Go V2

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
// ActivityLog separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
```

```
type ActivityLog struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewActivityLog constructs a new activity log runner.
func NewActivityLog(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) ActivityLog {
 scenario := ActivityLog{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddUserToPool selects a user from the known users table and uses administrator
 credentials to add the user to the user pool.
func (runner *ActivityLog) AddUserToPool(userPoolId string, tableName string)
 (string, string) {
 log.Println("To facilitate this example, let's add a user to the user pool using
 administrator privileges.")
 users, err := runner.helper.GetKnownUsers(tableName)
 if err != nil {
  panic(err)
 }
 user := users.Users[0]
 log.Printf("Adding known user %v to the user pool.\n", user.UserName)
 err = runner.cognitoActor.AdminCreateUser(userPoolId, user.UserName,
 user.UserEmail)
 if err != nil {
  panic(err)
 }
 pwSet := false
 password := runner.questioner.AskPassword("\nEnter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !pwSet {
  log.Printf("\nSetting password for user '%v'.\n", user.UserName)
```

```
  err = runner.cognitoActor.AdminSetUserPassword(userPoolId, user.UserName,
 password)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("\nEnter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   pwSet = true
  }
 }

 log.Println(strings.Repeat("-", 88))

 return user.UserName, password
}

// AddActivityLogTrigger adds a Lambda handler as an invocation target for the
 PostAuthentication trigger.
func (runner *ActivityLog) AddActivityLogTrigger(userPoolId string,
 activityLogArn string) {
 log.Println("Let's add a Lambda function to handle the PostAuthentication
 trigger from Cognito.\n" +
  "This trigger happens after a user is authenticated, and lets your function
 take action, such as logging\n" +
  "the outcome.")
 err := runner.cognitoActor.UpdateTriggers(
  userPoolId,
  actions.TriggerInfo{Trigger: actions.PostAuthentication, HandlerArn:
 aws.String(activityLogArn)})
 if err != nil {
  panic(err)
 }
 runner.resources.triggers = append(runner.resources.triggers,
 actions.PostAuthentication)
 log.Printf("Lambda function %v added to user pool %v to handle
 PostAuthentication Cognito trigger.\n",
  activityLogArn, userPoolId)

 log.Println(strings.Repeat("-", 88))
}
```

```
// SignInUser signs in as the specified user.
func (runner *ActivityLog) SignInUser(clientId string, userName string, password
 string) {
 log.Printf("Now we'll sign in user %v and check the results in the logs and the
 DynamoDB table.", userName)
 runner.questioner.Ask("Press Enter when you're ready.")
 authResult, err := runner.cognitoActor.SignIn(clientId, userName, password)
 if err != nil {
  panic(err)
 }
 log.Println("Sign in successful.",
  "The PostAuthentication Lambda handler writes custom information to CloudWatch
 Logs.")

 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)
}

// GetKnownUserLastLogin gets the login info for a user from the Amazon DynamoDB
 table and displays it.
func (runner *ActivityLog) GetKnownUserLastLogin(tableName string, userName
 string) {
 log.Println("The PostAuthentication handler also writes login data to the
 DynamoDB table.")
 runner.questioner.Ask("Press Enter when you're ready to continue.")
 users, err := runner.helper.GetKnownUsers(tableName)
 if err != nil {
  panic(err)
 }
 for _, user := range users.Users {
  if user.UserName == userName {
   log.Println("The last login info for the user in the known users table is:")
   log.Printf("\t%+v", *user.LastLogin)
  }
 }
 log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
func (runner *ActivityLog) Run(stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup()
```

```
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]
 runner.helper.PopulateUserTable(stackOutputs["TableName"])
 userName, password := runner.AddUserToPool(stackOutputs["UserPoolId"],
 stackOutputs["TableName"])

 runner.AddActivityLogTrigger(stackOutputs["UserPoolId"],
 stackOutputs["ActivityLogFunctionArn"])
 runner.SignInUser(stackOutputs["UserPoolClientId"], userName, password)
 runner.helper.ListRecentLogEvents(stackOutputs["ActivityLogFunction"])
 runner.GetKnownUserLastLogin(stackOutputs["TableName"], userName)

 runner.resources.Cleanup()

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Lambda 함수를 사용하여 PostAuthentication 트리거를 처리합니다.

```
const TABLE_NAME = "TABLE_NAME"

// LoginInfo defines structured login data that can be marshalled to a DynamoDB
 format.
type LoginInfo struct {
 UserPoolId string `dynamodbav:"UserPoolId"`
 ClientId   string `dynamodbav:"ClientId"`
 Time       string `dynamodbav:"Time"`
```

```
}

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string    `dynamodbav:"UserName"`
 UserEmail string    `dynamodbav:"UserEmail"`
 LastLogin LoginInfo `dynamodbav:"LastLogin"`
}

// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
 userEmail, err := attributevalue.Marshal(user.UserEmail)
 if err != nil {
  panic(err)
 }
 return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the PostAuthentication event by writing custom data to
 the logs and
// to an Amazon DynamoDB table.
func (h *handler) HandleRequest(ctx context.Context,
 event events.CognitoEventUserPoolsPostAuthentication)
 (events.CognitoEventUserPoolsPostAuthentication, error) {
 log.Printf("Received post authentication trigger from %v for user '%v'",
 event.TriggerSource, event.UserName)
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserName:  event.UserName,
  UserEmail: event.Request.UserAttributes["email"],
  LastLogin: LoginInfo{
   UserPoolId: event.UserPoolID,
   ClientId:   event.CallerContext.ClientID,
   Time:       time.Now().Format(time.UnixDate),
  },
 }
 // Write to CloudWatch Logs.
 fmt.Printf("%#v", user)
```

```
 // Also write to an external system. This examples uses DynamoDB to demonstrate.
 userMap, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshal to DynamoDB map. Here's why: %v\n", err)
 } else if len(userMap) == 0 {
  log.Printf("User info marshaled to an empty map.")
 } else {
  _, err := h.dynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
   Item:      userMap,
   TableName: aws.String(tableName),
  })
  if err != nil {
   log.Printf("Couldn't write to DynamoDB. Here's why: %v\n", err)
  } else {
   log.Printf("Wrote user info to DynamoDB table %v.\n", tableName)
  }
 }

 return event, nil
}

func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

일반적인 작업을 수행하는 구조체를 생성합니다.

```
// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(stackName string) (actions.StackOutputs, error)
 PopulateUserTable(tableName string)
```

```
 GetKnownUsers(tableName string) (actions.UserList, error)
 AddKnownUser(tableName string, user actions.User)
 ListRecentLogEvents(functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(stackName string)
 (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(stackName), nil
}
```

```
// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(tableName string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(tableName string) (actions.UserList,
 error) {
 knownUsers, err := helper.dynamoActor.Scan(tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(tableName string, user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(functionName)
 if err != nil {
  panic(err)
 }
```

```
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Amazon Cognito 작업을 래핑하는 구조체를 생성합니다.

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
```

```
func (actor CognitoActions) UpdateTriggers(userPoolId string,
 triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(context.TODO(),
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(context.TODO(),
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(clientId string, userName string, password
 string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(context.TODO(),
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
```

```
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(clientId string, userName string, password
 string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(context.TODO(),
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}
```

```
// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(clientId string, userName string)
 (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(context.TODO(),
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(clientId string, code string,
 userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(context.TODO(),
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}
```

```
// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(userAccessToken string) error {
 _, err := actor.CognitoClient.DeleteUser(context.TODO(),
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(userPoolId string, userName string,
 userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(context.TODO(),
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
```

```
func (actor CognitoActions) AdminSetUserPassword(userPoolId string, userName
 string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(context.TODO(),
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

DynamoDB 작업을 래핑하는 구조체를 생성합니다.

```
// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
```

```
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(tableName string) error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(context.TODO(),
 &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
 return err
```

```
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(tableName string) (UserList, error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(context.TODO(), &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(tableName string, user User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(context.TODO(), &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

CloudWatch Logs 액션을 래핑하는 구조체를 만드세요.

```
type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
```

```
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(functionName string)
 (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(context.TODO(),
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(functionName string,
 logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(context.TODO(),
 &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

액션을 래핑하는 구조체를 만드세요. AWS CloudFormation

```
// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(stackName string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(context.TODO(),
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

리소스를 정리합니다.

```
// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger
```

```
 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup() {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 \n" +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
 HandlerArn: nil}
  }
  err := resources.cognitoActor.UpdateTriggers(resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
```

```
    panic(err)
   }
   log.Println("Removed Cognito triggers from user pool.")
  } else {
   log.Println("Be sure to remove resources when you're done with them to avoid
  unexpected charges!")
  }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 다음 주제를 참조하십시오.
  - AdminCreateUser
  - AdminSetUserPassword
  - DeleteUser
  - InitiateAuth
  - UpdateUserPool

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. 이 서비스를 SDK와 함께 사용 AWS 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

SDK를 사용한 Amazon Cognito Sync의 코드 예제 AWS

다음 코드 예제는 AWS 소프트웨어 개발 키트 (SDK) 와 함께 Amazon Cognito Sync를 사용하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. 이 서비스를 SDK와 함께 사용 AWS 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

코드 예시

- SDK를 사용한 Amazon Cognito 동기화 작업 AWS
  - AWS SDK 또는 ListIdentityPoolUsage CLI와 함께 사용

SDK를 사용한 Amazon Cognito 동기화 작업 AWS

다음 코드 예제는 SDK를 사용하여 개별 Amazon Cognito Sync 작업을 수행하는 방법을 보여줍니다. AWS 이들 발췌문은 Amazon Cognito Sync API를 직접적으로 호출하며, 컨텍스트에서 실행되어야 하는 더 큰 프로그램에서 발췌한 코드입니다. 각 예제에는 코드 GitHub 설정 및 실행 지침을 찾을 수 있는 링크가 포함되어 있습니다.

다음 예제에는 가장 일반적으로 사용되는 작업만 포함되어 있습니다. 전체 목록은 Amazon Cognito Sync API Reference(Amazon Cognito Sync API 참조)를 참조하세요.

예제

- AWS SDK 또는 ListIdentityPoolUsage CLI와 함께 사용

AWS SDK 또는 **ListIdentityPoolUsage** CLI와 함께 사용

다음 코드 예시에서는 `ListIdentityPoolUsage`을 사용하는 방법을 보여 줍니다.

Rust

SDK for Rust

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn show_pools(client: &Client) -> Result<(), Error> {
    let response = client
        .list_identity_pool_usage()
        .max_results(10)
        .send()
        .await?;

    let pools = response.identity_pool_usages();
    println!("Identity pools:");

    for pool in pools {
        println!(
            "  Identity pool ID:    {}",
```

```
            pool.identity_pool_id().unwrap_or_default()
        );
        println!(
            "  Data storage:        {}",
            pool.data_storage().unwrap_or_default()
        );
        println!(
            "  Sync sessions count: {}",
            pool.sync_sessions_count().unwrap_or_default()
        );
        println!(
            "  Last modified:       {}",
            pool.last_modified_date().unwrap().to_chrono_utc()?
        );
        println!();
    }

    println!("Next token: {}", response.next_token().unwrap_or_default());

    Ok(())
}
```

- API에 대한 자세한 내용은 Rust용AWS SDK API 레퍼런스를 참조하십시오 ListIdentityPoolUsage.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. 이 서비스를 SDK와 함께 사용 AWS 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

멀티 테넌트 애플리케이션 모범 사례

Amazon Cognito 사용자 풀은 Amazon Cognito 할당량 내에 있어야 하는 대량의 요청을 생성하는 멀티 테넌트 애플리케이션과 함께 작동합니다. 고객층이 늘어날 때 이 용량을 확장하려면 추가 할당량 용량을 구매하면 됩니다.

> **Note**
>
> Amazon Cognito 할당량은 및 단위별로 적용됩니다. AWS 계정 AWS 리전이러한 할당량은 애플리케이션의 모든 테넌트 간에 공유됩니다. Amazon Cognito 서비스 할당량을 검토하고 할당량이 애플리케이션의 예상 볼륨 및 예상 테넌트 수를 충족하는지 확인하십시오.

이 섹션에서는 동일한 지역 및 내의 Amazon Cognito 리소스 간에 테넌트를 분리하기 위해 구현할 수 있는 방법을 설명합니다. AWS 계정또한 테넌트를 둘 이상의 지역 AWS 계정 또는 지역으로 나누고 각 지역에 자체 할당량을 할당할 수 있습니다. 다중 지역 다중 테넌시의 다른 이점으로는 가능한 최고 수준의 격리, 전 세계에 분산된 사용자를 위한 최단 네트워크 전송 시간, 조직의 기존 배포 모델 준수 등이 있습니다.

단일 지역 멀티테넌시는 고객과 관리자에게도 이점을 제공할 수 있습니다.

다음 목록은 공유 리소스를 사용하는 멀티테넌시의 몇 가지 이점을 설명합니다.

멀티테넌시의 장점

일반 사용자 디렉터리

멀티 테넌시는 고객이 둘 이상의 애플리케이션에 계정을 보유한 모델을 지원합니다. 타사 공급자의 ID를 일관된 단일 사용자 풀 프로필에 연결할 수 있습니다. 사용자 프로필이 테넌트별로 고유한 경우 단일 사용자 풀을 사용하는 모든 다중 테넌시 전략은 사용자 관리에 진입할 수 있는 단일 지점이 있습니다.

공통 보안

공유 사용자 풀에서 보안에 대한 단일 표준을 만들고 모든 테넌트에 동일한 고급 보안, MFA (다중 요소 인증) AWS WAF및 표준을 적용할 수 있습니다. AWS WAF 웹 ACL은 연결하는 리소스와 AWS 리전 동일해야 하므로 멀티 테넌시는 복잡한 리소스에 대한 공유 액세스를 제공합니다. 다중 리전 Amazon Cognito 애플리케이션에서 일관된 보안 구성을 유지하려면 리소스 간에 구성을 복제하는 운영 표준을 적용해야 합니다.

일반적인 사용자 지정

를 사용하여 사용자 풀과 자격 증명 풀을 사용자 지정할 수 AWS Lambda있습니다. 사용자 풀의 Lambda 트리거 구성과 자격 증명 풀의 Amazon Cognito 이벤트 구성은 복잡할 수 있습니다. Lambda 함수는 사용자 풀 또는 자격 증명 풀과 AWS 리전 동일해야 합니다. 공유 Lambda 함수는 사용자 지정 인증 흐름, 사용자 마이그레이션, 토큰 생성 및 지역 내 기타 기능에 대한 표준을 적용할 수 있습니다.

공통 메시징

Amazon Simple Notification Service (Amazon SNS) 를 사용하려면 사용자에게 SMS 메시지를 보내려면 먼저 지역에 추가 구성을 해야 합니다. Amazon Simple Email Service (Amazon SES) 가 검증한 자격 증명 및 지역 내에 포함된 도메인을 사용하여 이메일 메시지를 보낼 수 있습니다.

멀티 테넌시를 사용하면 이 구성 및 유지 관리 오버헤드를 모든 테넌트 간에 공유할 수 있습니다. Amazon SNS와 Amazon SES를 모두 사용할 수 있는 것은 AWS 리전아니기 때문에 리소스를 지역 간에 분할하려면 추가 고려가 필요합니다.

사용자 지정 메시징 공급자를 사용하면 단일 Lambda 함수를 공통적으로 사용자 지정하여 메시지 전송을 관리할 수 있습니다.

호스팅된 UI는 이미 인증된 사용자를 인식할 수 있도록 브라우저에 세션 쿠키를 설정합니다. 사용자 풀에서 로컬 사용자를 인증하면 해당 세션 쿠키가 동일한 사용자 풀의 모든 앱 클라이언트에 대해 로컬 사용자를 인증합니다. 로컬 사용자는 외부 IdP를 통한 페더레이션 없이 사용자 풀 디렉터리에만 존재합니다. 이러한 세션 쿠키는 1시간 동안 유효합니다. 세션 쿠키 기간은 변경할 수 없습니다.

호스팅된 UI 세션 쿠키를 사용하여 앱 클라이언트에서의 로그인을 방지하는 두 가지 방법이 있습니다.

- 사용자를 테넌트별 사용자 풀로 분리하십시오.
- 호스팅된 UI 로그인을 Amazon Cognito 사용자 풀 API 로그인으로 대체하십시오.

주제

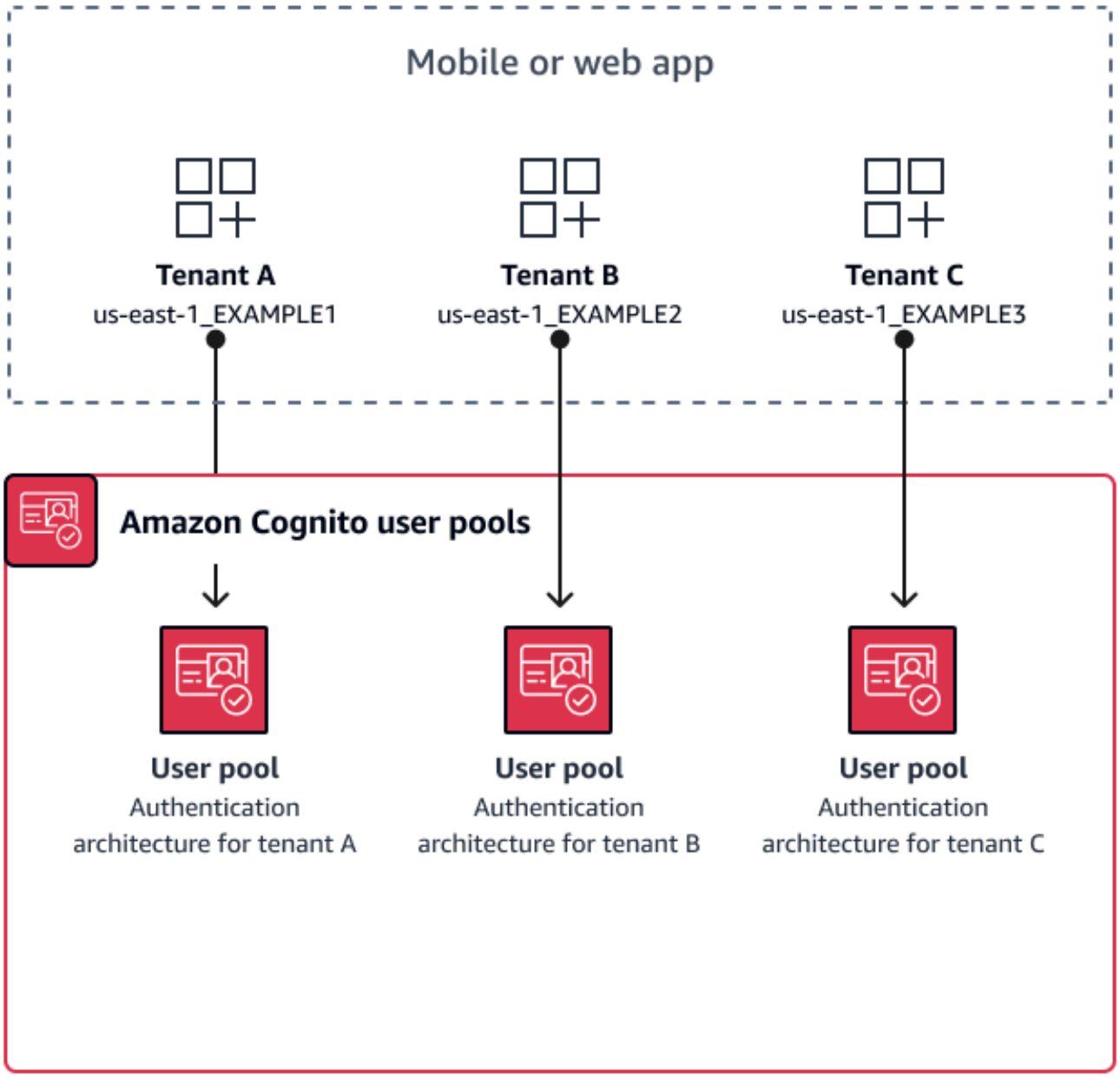
- 사용자 풀 멀티테넌시 모범 사례
- 앱-클라이언트 멀티테넌시 모범 사례
- 사용자 풀 그룹 멀티테넌시 모범 사례
- 사용자 지정 속성 멀티테넌시 모범 사례
- 멀티 테넌시 보안 권장 사항

사용자 풀 멀티테넌시 모범 사례

앱의 각 테넌트에 대한 사용자 풀을 생성합니다. 이 접근 방식은 각 테넌트별로 최대한의 격리를 제공합니다. 각 테넌트별로 서로 다른 구성을 구현할 수 있습니다. 사용자 풀별 테넌트 격리를 통해 유연하게 user-to-tenant 매핑할 수 있습니다. 동일한 사용자에 대해 여러 개의 프로필을 생성할 수 있습니다. 그러나 각 사용자는 액세스 권한이 있는 각 테넌트별로 개별적으로 가입해야 합니다.

이 접근 방식을 사용하면 각 테넌트에 대해 호스트된 UI를 독립적으로 설정하고 사용자를 테넌트별 애플리케이션 인스턴스로 리디렉션할 수 있습니다. 또한 이 접근 방식을 사용하여 Amazon API Gateway와 같은 백엔드 서비스와 통합할 수 있습니다.

다음 다이어그램은 전용 사용자 풀이 있는 각 테넌트를 보여줍니다.



사용자 풀 멀티테넌시 구현 시기

격리와 사용자 지정이 주요 관심사인 경우 사용자 풀이 여러 개 있는 아키텍처에서는 사용자와 테넌트 간의 관계가 복잡할 수 있습니다. 교육용 테넌트가 두 명인 경우를 예로 들어 보겠습니다. 동일한 사용자가 한 앱에서는 액세스가 제한된 학생이고 다른 앱에서는 높은 수준의 권한을 가진 교사일 수 있습니다. 한 앱에서는 MFA가 필요하지만 다른 앱에서는 필요하지 않거나 암호 정책이 다를 수 있습니다. 로

컬 사용자는 호스팅된 UI를 사용하여 사용자 풀의 여러 앱 클라이언트에 로그인할 수 있으므로 호스팅된 UI로 둘 이상의 테넌트가 로그인하도록 하려는 경우에도 사용자 풀 멀티 테넌시가 적합합니다.

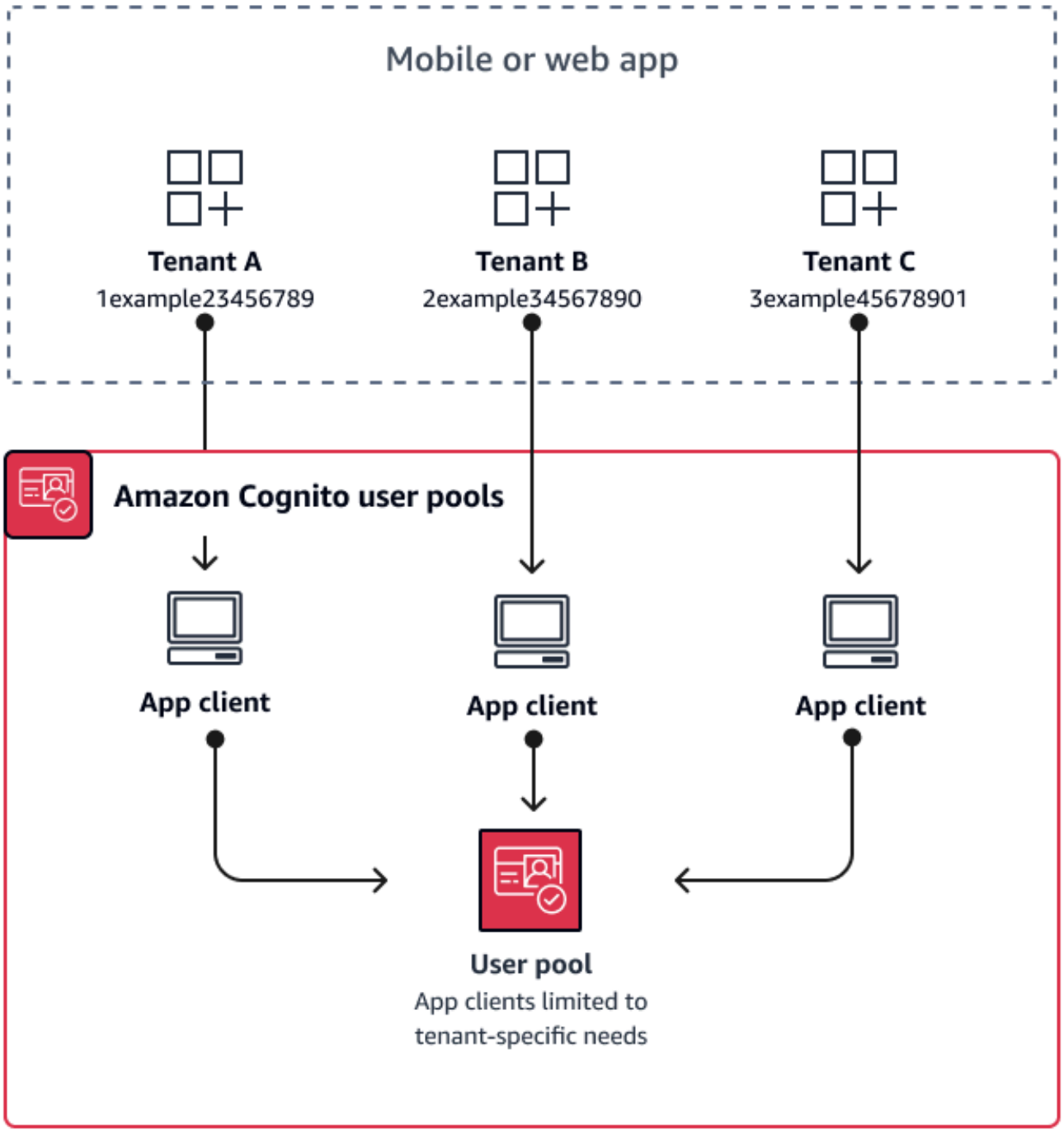
노력 수준

이 방법을 사용하기 위해서는 많은 개발 및 운영상의 작업이 요구됩니다. 앱 제품군에 일관되고 예측 가능한 결과를 보장하려면 Amazon Cognito 리소스를 자동화 도구와 통합하고 인증 아키텍처가 점점 복잡해져도 기준을 유지해야 합니다. 앱을 위한 단일 출발점을 만들려면 사용자를 올바른 리소스로 라우팅하는 초기 결정을 캡처할 수 있는 사용자 인터페이스 (UI) 요소를 구축해야 합니다.

앱-클라이언트 멀티테넌시 모범 사례

앱의 각 테넌트에 대한 앱 클라이언트를 생성합니다. 앱-클라이언트 멀티 테넌시를 사용하면 모든 사용자를 테넌트 연결 앱 클라이언트에 할당하고 단일 사용자 프로필을 유지할 수 있습니다. 사용자 풀의 일부 또는 모든 ID 공급자 (IdPs) 를 앱 클라이언트에 할당할 수 있으므로 테넌트 앱 클라이언트는 테넌트별 IdP를 사용한 로그인을 허용할 수 있습니다. 사용자가 여러 테넌트에 있는 경우 일관된 사용자 경험을 위해 해당 프로필을 여러 IdPs 테넌트와 연결할 수 있습니다.

다음 다이어그램은 공유 사용자 풀에 전용 앱 클라이언트가 있는 각 테넌트를 보여줍니다.



앱-클라이언트 멀티테넌시 구현 시기

Lambda 트리거, 암호 정책, 이메일과 SMS 메시지의 콘텐츠 및 전송 방법과 같은 사용자 풀 수준의 설정에 대한 범용 구성을 선택할 수 있는 경우 공유 사용자 풀의 사용자는 모든 앱 클라이언트에 로그인

할 수 있으므로 앱-클라이언트 멀티 테넌시는 app-client-specific IdPs Amazon Cognito 사용자 풀 API로 로그인하는 데 적합합니다. 앱-클라이언트 멀티 테넌시는 사용자가 여러 애플리케이션 간에 전환할 수 있도록 허용하려는 one-to-many 환경에도 적합합니다.

노력 수준

앱-클라이언트 멀티테넌시에는 적당한 노력이 필요합니다. 앱-클라이언트 멀티 테넌시의 주요 문제는 테넌트가 호스팅된 UI 쿠키를 제공하고 앱 간에 전환할 수 있는지 여부입니다. 앱-클라이언트 멀티테넌시 아키텍처에서는 격리가 필요한 경우 호스팅된 UI 로그인을 사용하지 마십시오. 앱 클라이언트 로직이 내장된 상태에서 모바일 앱 또는 웹 앱 링크를 배포하거나 사용자의 테넌시를 결정하는 초기 UI 요소를 구축할 수 있습니다. 여러 사용자 풀과 자격 증명 풀에 걸쳐 구성을 표준화하고 유지 관리할 필요가 없으므로 작업 수준이 낮습니다.

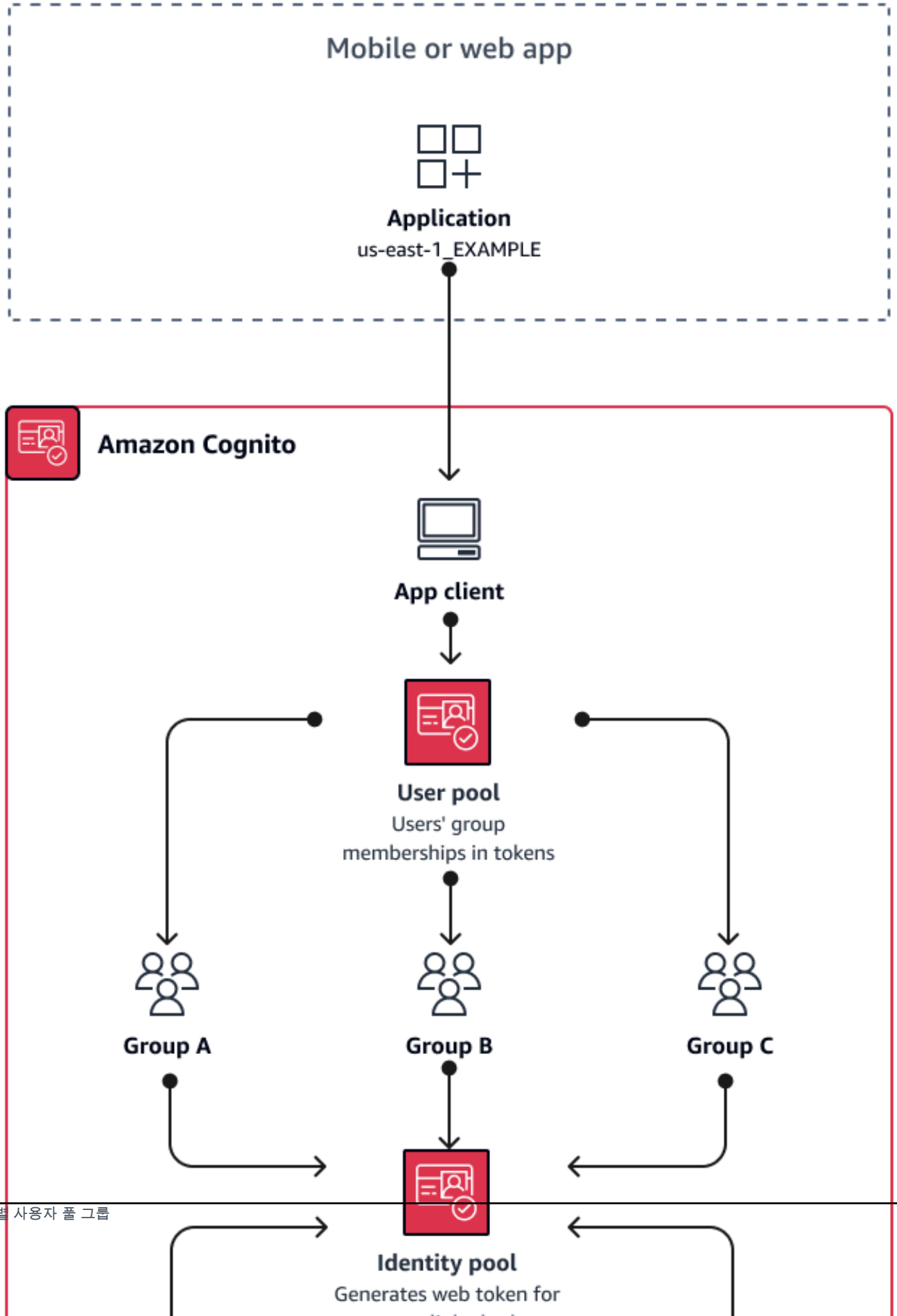
사용자 풀 그룹 멀티테넌시 모범 사례

그룹 기반 멀티테넌시는 아키텍처에 자격 증명 풀이 있는 Amazon Cognito 사용자 풀이 필요할 때 가장 효과적입니다.

사용자 풀 ID 및 액세스 토큰에는 클레임이 포함되어 있습니다. `cognito:groups` 또한 ID 토큰에는 `cognito:preferred_role` 클레임이 `cognito:roles` 포함됩니다. 앱 인증의 기본 결과가 자격 증명 풀의 임시 AWS 자격 증명인 경우, 사용자의 그룹 멤버십은 사용자가 받는 IAM 역할 및 권한을 결정할 수 있습니다.

예를 들어, 각각 고유한 Amazon S3 버킷에 애플리케이션 자산을 저장하는 세 개의 테넌트를 생각해 보십시오. 각 테넌트의 사용자를 관련 그룹에 할당하고, 그룹의 선호 역할을 구성하고, 해당 역할에 버킷에 대한 읽기 액세스 권한을 부여합니다.

다음 다이어그램은 IAM 역할에 대한 적격성을 결정하는 사용자 풀의 전용 그룹과 앱 클라이언트 및 사용자 풀을 공유하는 테넌트를 보여줍니다.


Mobile or web app
Application
us-east-1_EXAMPLE
Amazon Cognito
App client
User pool
Users' group
memberships in tokens
Group A
Group B
Group C
Identity pool
Generates web token for

그룹 멀티테넌시 구현 시기

AWS 리소스 액세스가 주요 관심사인 경우 Amazon Cognito 사용자 풀의 그룹 사용자 풀은 역할 기반 액세스 제어 (RBAC) 를 위한 메커니즘입니다. 사용자 풀에 여러 그룹을 구성하고 그룹 우선 순위에 따라 복잡한 RBAC 결정을 내릴 수 있습니다. 자격 증명 풀은 우선 순위가 가장 높은 역할, 그룹 클레임의 모든 역할 또는 사용자 토큰의 다른 클레임에 대한 자격 증명을 할당할 수 있습니다.

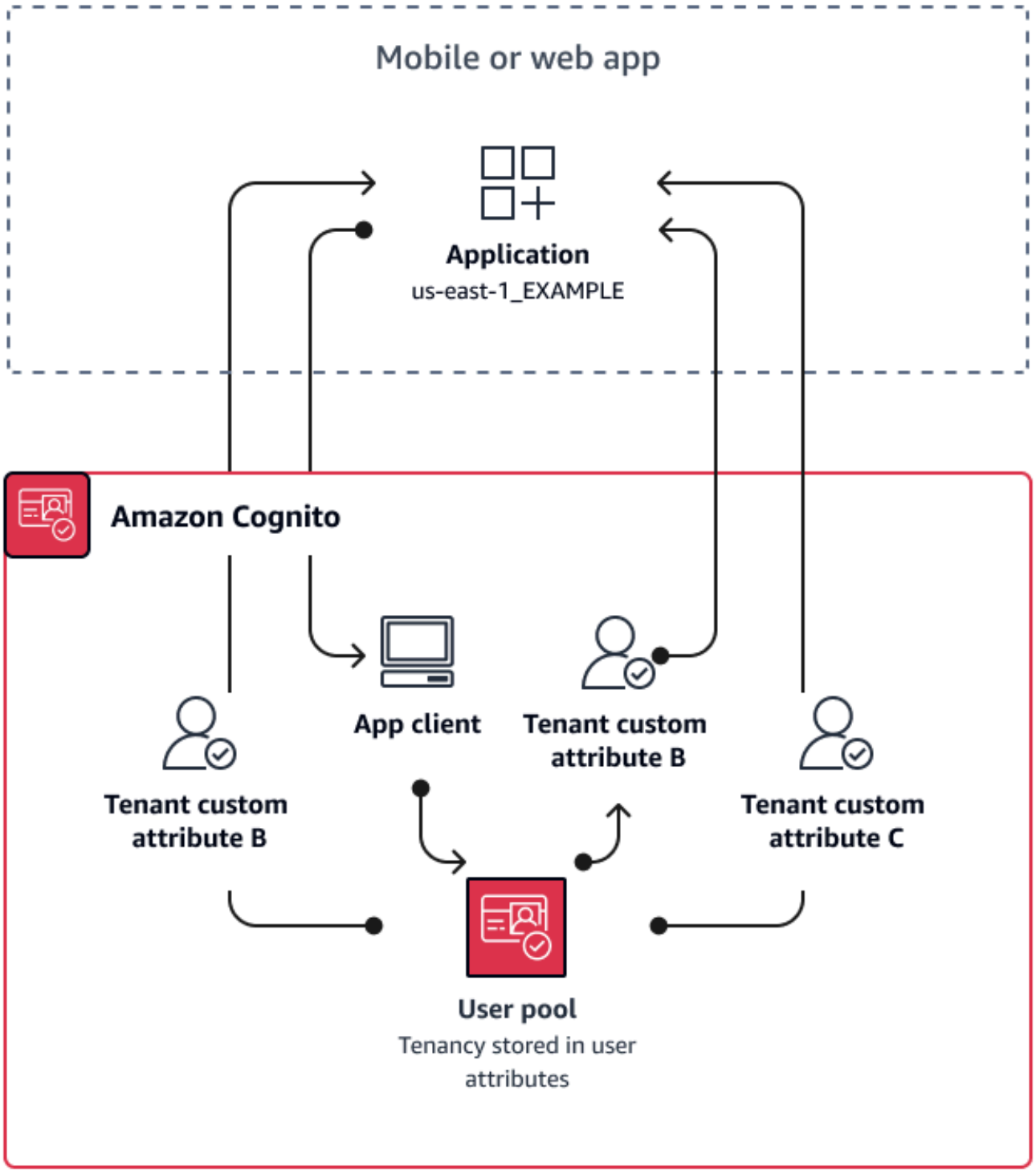
노력 수준

그룹 멤버십만으로 멀티테넌시를 유지하려는 노력의 수준은 낮습니다. 그러나 IAM 역할 선택을 위한 기본 제공 용량 이상으로 사용자 풀 그룹의 역할을 확장하려면 사용자 토큰의 그룹 멤버십을 처리하는 애플리케이션 로직을 구축하고 클라이언트에서 수행할 작업을 결정해야 합니다. Amazon 검증 권한을 앱과 통합하여 클라이언트 측 승인 결정을 내릴 수 있습니다. 그룹 식별자는 현재 Verified Permissions IsAuthorizedWithTokenAPI 작업에서 처리되지 않지만, 그룹 멤버십 클레임을 포함하여 토큰의 콘텐츠를 파싱하는 사용자 지정 코드를 개발할 수 있습니다.

사용자 지정 속성 멀티테넌시 모범 사례

Amazon Cognito는 사용자가 선택한 이름을 가진 사용자 지정 속성을 지원합니다. 사용자 지정 속성이 유용한 시나리오 중 하나는 공유 사용자 풀의 사용자 테넌시를 구분하는 경우입니다. 사용자에게 와 같은 `custom:tenantID` 속성 값을 할당하면 앱이 그에 따라 테넌트별 리소스에 대한 액세스 권한을 할당할 수 있습니다. 테넌트 ID를 정의하는 사용자 지정 속성은 앱 클라이언트에서 변경할 수 없거나 읽기 전용이어야 합니다.

다음 다이어그램은 앱 클라이언트와 사용자 풀을 공유하는 테넌트를 보여줍니다. 사용자 풀에는 해당 테넌트가 속한 테넌트를 나타내는 사용자 지정 속성이 있습니다.



사용자 지정 속성이 테넌시를 결정하는 경우 단일 애플리케이션 또는 로그인 URL을 배포할 수 있습니다. 사용자가 로그인하면 앱에서 `custom:tenantID` 클레임을 처리하고 로드할 자산, 적용할 브

랜딩, 표시할 기능을 결정할 수 있습니다. 사용자 속성을 기반으로 고급 액세스 제어 결정을 내리려면 Amazon Verified Permissions에서 자격 증명 공급자로 사용자 풀을 설정하고 ID 또는 액세스 토큰의 내용으로부터 액세스 결정을 생성하십시오.

사용자 지정 속성 멀티테넌시를 구현해야 하는 시기

테넌시가 표면 수준인 경우 테넌트 속성은 브랜딩 및 레이아웃 결과에 기여할 수 있습니다. 테넌트 간에 상당한 격리를 유지하려는 경우 사용자 지정 속성이 최선의 선택은 아닙니다. 사용자 풀 또는 앱-클라이언트 수준에서 구성해야 하는 테넌트 간의 차이 (예: MFA 또는 호스팅된 UI 브랜딩) 를 사용하려면 사용자 지정 속성이 제공하지 않는 방식으로 테넌트를 구분해야 합니다. 자격 증명 풀을 사용하면 ID 토큰의 사용자 지정 속성 클레임에서 사용자의 IAM 역할을 선택할 수도 있습니다.

노력 수준

사용자 지정 속성 멀티테넌시는 앱에서 테넌트 기반 권한 부여 결정의 의무를 이전하기 때문에 작업 수준이 높은 경향이 있습니다. OIDC 클레임을 파싱하는 클라이언트 구성이나 Amazon Verified Permissions에 대해 이미 잘 알고 있다면 이 접근 방식에는 최소한의 노력이 필요할 수 있습니다.

멀티 테넌시 보안 권장 사항

보다 안전한 애플리케이션을 만들려면 다음을 권장합니다.

- Amazon 검증 권한으로 앱의 테넌시를 검증하십시오. 애플리케이션에서 사용자의 요청을 허용하기 전에 사용자 풀, 앱 클라이언트, 그룹 또는 사용자 지정 속성 권한을 검사하는 정책을 구축하십시오. AWS Amazon Cognito 사용자 풀을 염두에 두고 검증된 권한 자격 증명 소스를 생성했습니다. 검증된 권한에는 멀티테넌시 관리에 대한 추가 지침이 있습니다.
- 도메인 일치를 기반으로 테넌트에 대한 사용자 액세스 권한을 부여하는 데 확인된 이메일 주소만 사용합니다. 앱에서 확인하거나 외부 IdP가 확인 증명을 제공하는 경우에만 이메일 주소와 전화 번호를 신뢰하세요. 이러한 권한 설정에 대한 자세한 내용은 속성 권한 및 범위를 참조하세요.
- 테넌트를 식별하는 사용자 프로필 속성에는 변경할 수 없거나 읽기 전용인 사용자 지정 특성을 사용하십시오. 사용자를 만들거나 사용자 풀에 사용자를 등록할 때만 변경 불가 속성 값을 설정할 수 있습니다. 또한 앱 클라이언트에 이러한 속성에 대한 읽기 전용 권한을 부여합니다.
- 테넌트의 외부 IdP와 애플리케이션 클라이언트 간의 1:1 매핑을 사용하여 테넌트 간 무단 액세스를 방지할 수 있습니다. 외부 IdP에서 인증되었으며 유효한 Amazon Cognito 세션 쿠키가 있는 사용자는 동일한 IdP를 신뢰하는 다른 테넌트 앱에 액세스할 수 있습니다.
- 애플리케이션에서 테넌트 일치 및 권한 부여 로직을 구현할 때 테넌트에 대한 사용자 액세스 권한을 부여하는 기준을 사용자가 직접 수정할 수 없도록 합니다. 또한 페더레이션에 외부 IdP 가 사용되는 경우 테넌트 ID 공급자 관리자가 사용자 액세스를 수정할 수 없도록 합니다.

일반적인 Amazon Cognito 시나리오

이 주제에서는 Amazon Cognito를 사용하는 6가지 일반적인 시나리오에 대해 설명합니다.

Amazon Cognito의 두 가지 주요 구성 요소는 사용자 풀과 자격 증명 풀입니다. 사용자 풀은 웹과 모바일 앱 사용자의 가입 및 로그인 옵션을 제공하는 사용자 디렉터리입니다. 자격 증명 풀은 사용자에게 다른 AWS 자격 증명에 대한 액세스 권한을 부여하는 임시 자격 증명을 제공합니다 AWS 서비스.

사용자 풀은 Amazon Cognito의 사용자 디렉터리입니다. 앱 사용자는 사용자 풀을 통해 직접 로그인하거나 타사 ID 공급자 (IdP) 를 통해 페더레이션할 수 있습니다. 사용자 풀은 페이스북, 구글, 아마존, 애플을 통한 소셜 로그인과 OpenID Connect (OIDC) 및 SAML에서 반환되는 토큰을 처리하는 오버헤드를 관리합니다. IdPs 사용자가 직접 또는 타사를 통해 로그인하는지 여부와 무관하게 사용자 풀의 모든 멤버는 디렉터리 프로필을 보유하며, SDK를 통해 액세스할 수 있습니다.

자격 증명 풀을 통해 사용자는 Amazon S3 및 DynamoDB와 같은 AWS 서비스에 액세스하기 위한 임시 AWS 자격 증명을 얻을 수 있습니다. 자격 증명 풀은 익명의 게스트 사용자뿐 아니라 타사를 통한 페더레이션도 지원합니다. IdPs

주제

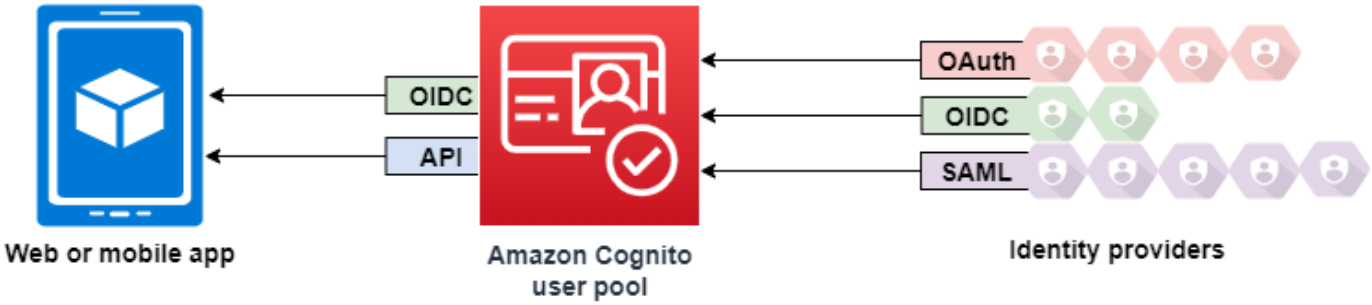
- 사용자 풀로 인증
- 사용자 풀로 서버 측 리소스 액세스
- 사용자 풀로 API Gateway 및 Lambda를 사용하여 리소스 액세스
- 사용자 풀과 자격 증명 풀을 사용하여 AWS 서비스에 액세스하십시오.
- 서드 파티에 인증 및 자격 증명 풀로 AWS 서비스 액세스
- Amazon Cognito를 사용하여 AWS AppSync 리소스에 액세스하기

사용자 풀로 인증

사용자가 사용자 풀을 사용하여 인증하도록 설정할 수 있습니다. 앱 사용자는 사용자 풀을 통해 직접 로그인하거나 타사 ID 공급자 (IdP) 를 통해 페더레이션할 수 있습니다. 사용자 풀은 페이스북, 구글, 아마존, 애플을 통한 소셜 로그인과 OpenID Connect (OIDC) 및 SAML에서 반환되는 토큰을 처리하는 오버헤드를 관리합니다. IdPs

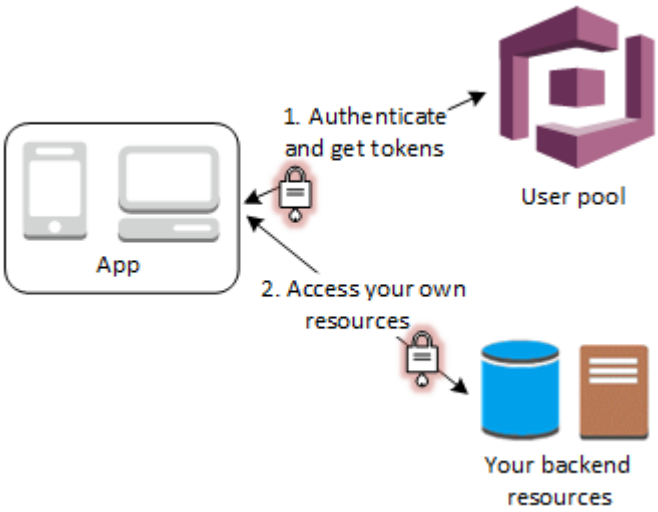
인증 성공 이후 웹 또는 모바일 앱은 Amazon Cognito에서 사용자 풀 토큰을 받습니다. 이러한 토큰을 사용하여 앱이 다른 AWS 서비스에 액세스할 수 있도록 허용하는 AWS 자격 증명을 검색할 수도 있고, 이를 사용하여 서버 측 리소스 또는 Amazon API Gateway에 대한 액세스를 제어할 수도 있습니다.

자세한 내용은 사용자 풀 인증 흐름 및 사용자 풀에 토큰 사용 섹션을 참조하세요.



사용자 풀로 서버 측 리소스 액세스

사용자 풀 로그인 성공 이후 웹 또는 모바일 앱은 Amazon Cognito에서 사용자 풀 토큰을 받습니다. 이러한 토큰을 사용하여 서버 측 리소스에 대한 액세스를 제어할 수 있습니다. 또한 사용자 풀 그룹을 생성하여 권한을 관리하고 다른 유형의 사용자를 표시할 수도 있습니다. 그룹을 사용하여 리소스에 액세스할 권한을 제어하는 방법에 대한 자세한 내용은 사용자 풀에 그룹 추가 섹션을 참조하세요.



사용자 풀에 대한 도메인이 구성되면 Amazon Cognito는 호스트된 웹 UI를 프로비저닝합니다. 이를 사용하여 앱에 가입 및 로그인 페이지를 추가할 수 있습니다. 이 OAuth 2.0 기반을 사용하면 사용자가 보호된 리소스에 액세스할 수 있는 자체 리소스 서버를 만들 수 있습니다. 자세한 정보는 리소스 서버를 통한 범위, M2M 및 API 인증을 참조하세요.

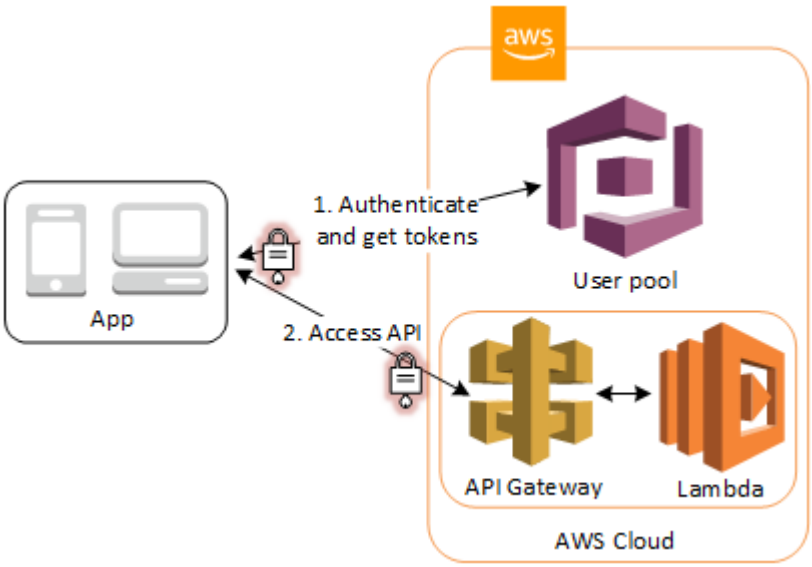
사용자 풀 인증에 대한 자세한 내용은 사용자 풀 인증 흐름 및 사용자 풀에 토큰 사용 섹션을 참조하세요.

사용자 풀로 API Gateway 및 Lambda를 사용하여 리소스 액세스

사용자가 API Gateway를 통해 API에 액세스하도록 할 수 있습니다. API Gateway는 성공적인 사용자 풀 인증에서 토큰을 검증하고, 토큰을 사용하여 사용자에게 Lambda 함수 또는 자체 API를 비롯한 리소스에 대한 액세스 권한을 부여합니다.

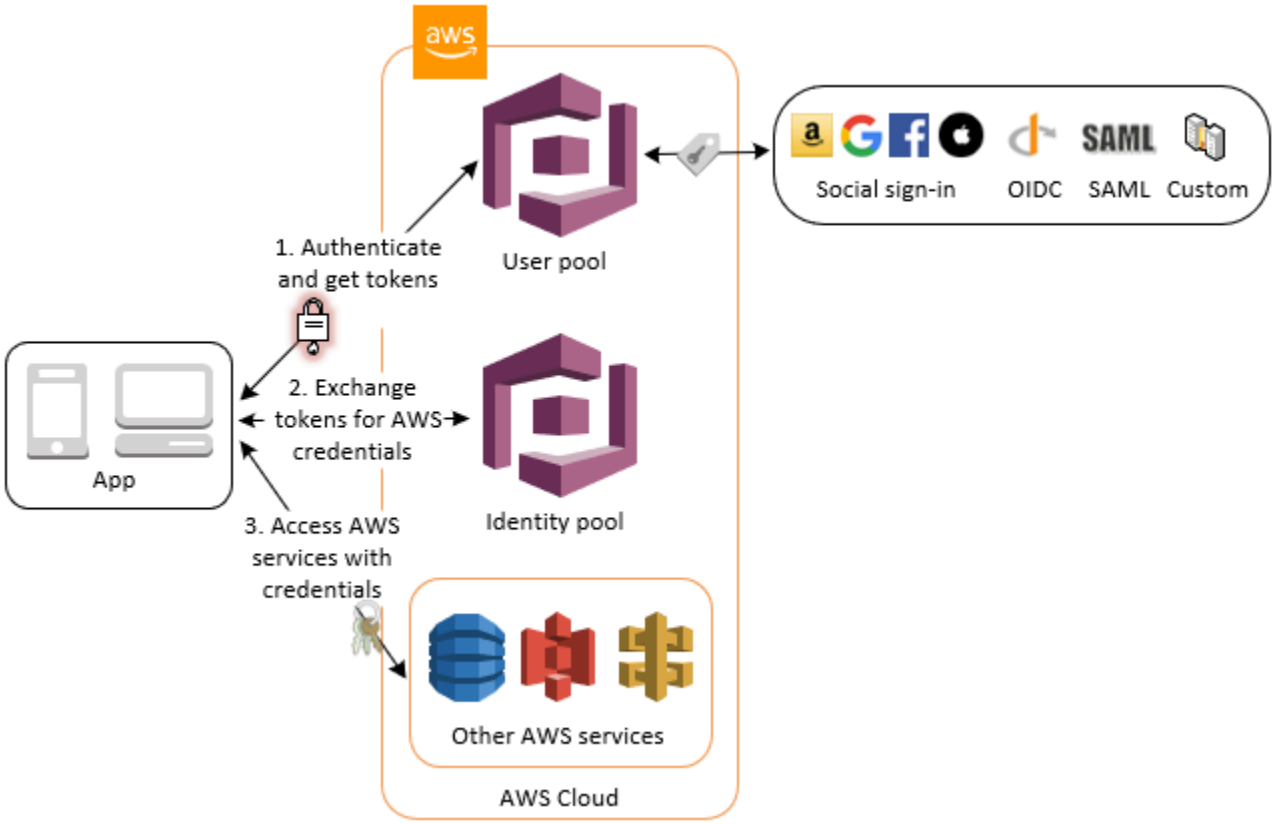
사용자 풀에 그룹을 사용하여 IAM 역할에 그룹 멤버십을 매핑함으로써 API Gateway를 통해 권한을 제어할 수 있습니다. 사용자가 속한 그룹은 앱 사용자가 로그인할 때 사용자 풀에서 제공한 ID 토큰에 포함됩니다. 사용자 풀 그룹에 대한 자세한 내용은 사용자 풀에 그룹 추가 섹션을 참조하세요.

Amazon Cognito 권한 부여자 Lambda 함수에서 확인을 위해 API Gateway에 대한 요청과 함께 사용자 풀 토큰을 제출할 수 있습니다. API Gateway 대한 자세한 내용은 Amazon Cognito 사용자 풀에 API Gateway 사용을 참조하세요.



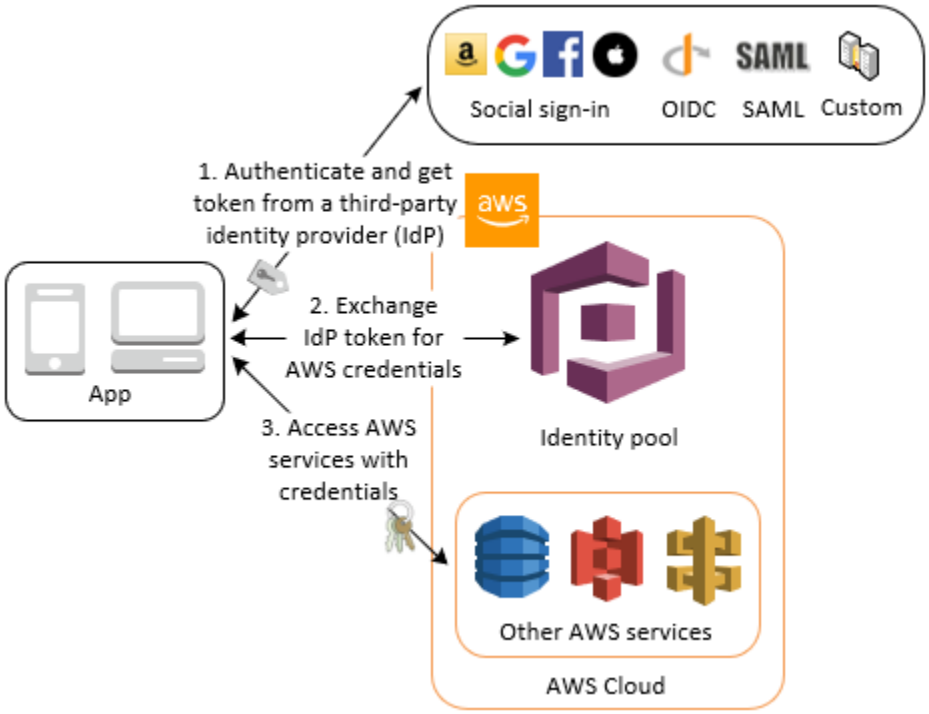
사용자 풀과 자격 증명 풀을 사용하여 AWS 서비스에 액세스하십시오.

사용자 풀 인증 성공 이후 앱은 Amazon Cognito에서 사용자 풀 토큰을 받습니다. 자격 증명 풀을 사용하여 다른 AWS 서비스에 대한 임시 액세스와 교환할 수 있습니다. 자세한 내용은 로그인 후 자격 증명 풀을 AWS 서비스 사용하여 액세스 및 Amazon Cognito 자격 증명 풀 시작하기 섹션을 참조하세요.



서드 파티에 인증 및 자격 증명 풀로 AWS 서비스 액세스

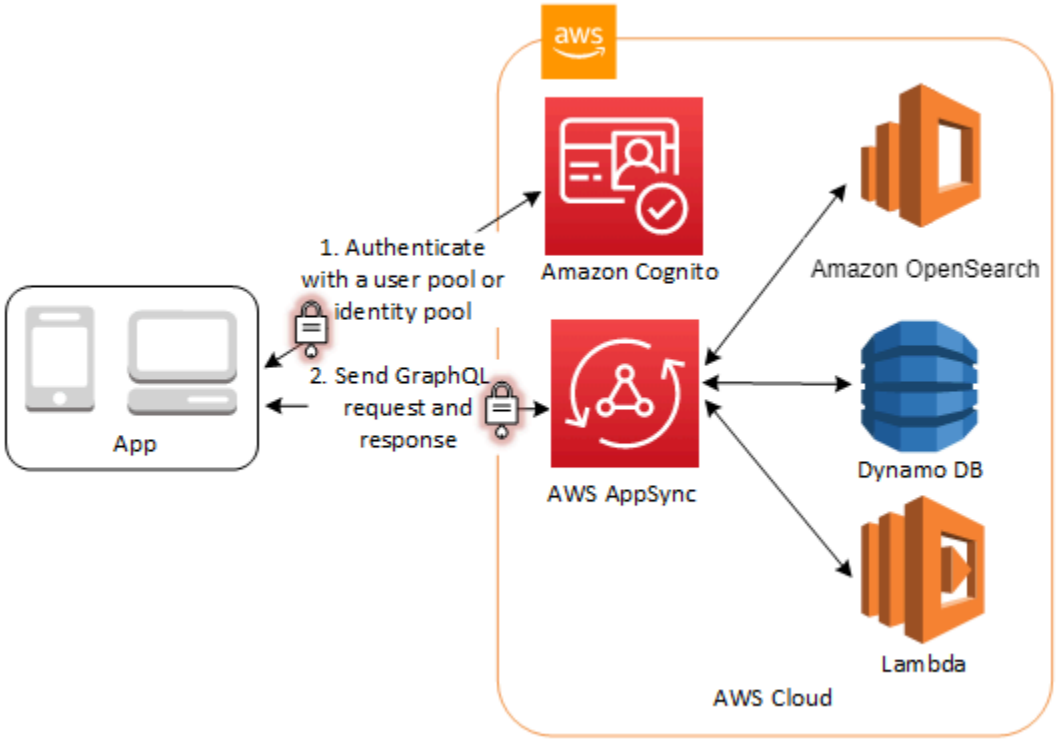
자격 증명 풀을 통해 사용자가 AWS 서비스에 액세스할 수 있도록 할 수 있습니다. 자격 증명 풀에는 서드 파티 자격 증명 공급자에 의해 인증된 사용자의 IdP 토큰이 필요합니다(익명 게스트인 경우 필요하지 않음). 대신 자격 증명 풀은 다른 AWS 서비스에 액세스하는 데 사용할 수 있는 임시 AWS 자격 증명을 부여합니다. 자세한 정보는 Amazon Cognito 자격 증명 풀 시작하기을 참조하세요.



Amazon Cognito를 사용하여 AWS AppSync 리소스에 액세스하기

Amazon Cognito 사용자 풀 인증을 성공적으로 완료한 토큰으로 사용자에게 AWS AppSync 리소스에 대한 액세스 권한을 부여할 수 있습니다. 자세한 내용은 AWS AppSync 개발자 안내서의 AMAZON_COGNITO_USER_POOLS 인증을 참조하세요.

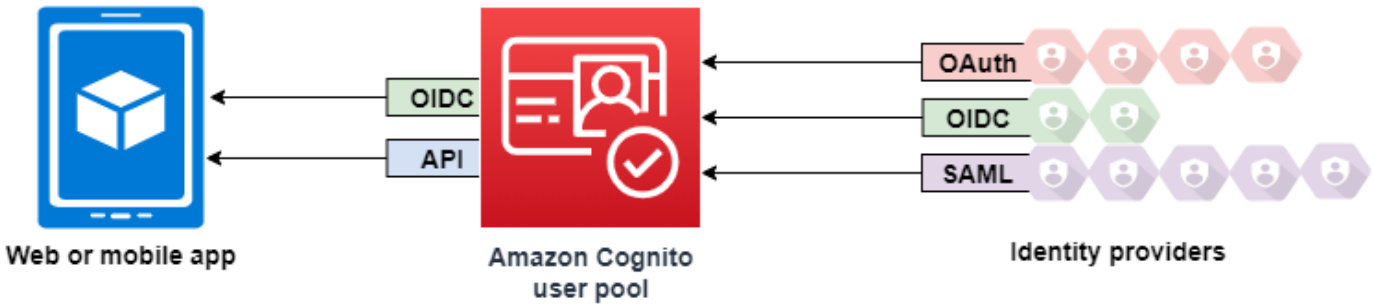
자격 증명 풀에서 AWS AppSync 받은 IAM 자격 증명을 사용하여 GraphQL API에 대한 요청에 서명할 수도 있습니다. AWS_IAM 권한 부여를 참조하십시오.


aws
Amazon Cognito
1. Authenticate with a user pool or identity pool
2. Send GraphQL request and response
App
AWS AppSync
Amazon OpenSearch
Dynamo DB
Lambda
AWS Cloud

Amazon Cognito 사용자 풀

Amazon Cognito 사용자 풀은 웹 및 모바일 앱 인증과 권한 부여를 위한 사용자 디렉터리입니다. 앱의 관점에서 보면 Amazon Cognito 사용자 풀은 OpenID Connect(OIDC) ID 제공업체(idP)입니다. 사용자 풀은 보안, 아이덴티티 페더레이션, 앱 통합, 사용자 경험 사용자 지정을 위한 기능 계층을 추가합니다.

예를 들어 사용자의 세션이 신뢰할 수 있는 출처에서 온 것인지 확인할 수 있습니다. Amazon Cognito 디렉터리를 외부 ID 제공업체와 결합할 수 있습니다. 선호하는 AWS SDK를 사용하여 앱에 가장 적합한 API 인증 모델을 선택할 수 있습니다. Amazon Cognito의 기본 동작을 수정하거나 점검하는 AWS Lambda 함수를 추가할 수도 있습니다.



주제

- 특성
- 사용자 풀을 사용한 인증
- Amazon Cognito 사용자 풀 API 및 사용자 풀 엔드포인트 사용
- 사용자 풀 구성 업데이트
- Amazon Cognito 호스팅 UI 및 페더레이션 엔드포인트 설정 및 사용
- 리소스 서버를 통한 범위, M2M 및 API 인증
- 서드 파티를 통한 사용자 풀 로그인 추가
- Lambda 트리거를 사용하여 사용자 풀 워크플로 사용자 정의
- Amazon Cognito 사용자 풀에서 Amazon Pinpoint 분석 사용
- 사용자 풀의 사용자 관리
- Amazon Cognito 사용자 풀에 대한 이메일 설정
- Amazon Cognito 사용자 풀의 SMS 메시지 설정
- 사용자 풀에 토큰 사용
- 성공적인 사용자 풀 인증 후 리소스 액세스

- Amazon Cognito 사용자 풀의 보안 기능 사용

특성

Amazon Cognito 사용자 풀에는 다음과 같은 기능이 있습니다.

가입

Amazon Cognito 사용자 풀에는 사용자 풀에 사용자 프로필을 추가하는 사용자 기반 방법, 관리자 기반 방법, 프로그래밍 방법이 있습니다. Amazon Cognito 사용자 풀은 다음과 같은 가입 모델을 지원합니다. 앱에서 이러한 모델을 결합하여 사용할 수 있습니다.

> **Important**
>
> 사용자 풀에서 사용자 가입을 활성화하면 인터넷에 있는 누구나 계정에 가입하고 앱에 로그인할 수 있습니다. 퍼블릭 가입이 가능하도록 앱을 공개하려는 경우가 아니면 사용자 풀에서 자체 등록을 활성화하지 마세요. 이 설정을 변경하려면 사용자 풀 콘솔의 가입 경험 탭에서 셀프 서비스 가입을 업데이트하거나 또는 API AllowAdminCreateUserOnly요청의 값을 업데이트하세요. CreateUserPool UpdateUserPool
> 사용자 풀에서 설정할 수 있는 보안 기능에 대한 자세한 내용은 Amazon Cognito 사용자 풀의 보안 기능 사용 섹션을 참조하세요.

1. 사용자는 앱에서 정보를 입력하고 사용자 풀에 고유한 사용자 프로필을 생성할 수 있습니다. API 가입 작업을 호출하여 사용자 풀에 사용자를 등록할 수 있습니다. 이러한 가입 작업을 누구에게나 공개하거나 클라이언트 암호 또는 자격 증명으로 승인할 수 있습니다. AWS
2. Amazon Cognito에 정보를 전달할 권한을 부여할 수 있는 서드 파티 IdP로 사용자를 리디렉션할 수 있습니다. Amazon Cognito는 OIDC ID 토큰, OAuth 2.0 `userInfo` 데이터, SAML 2.0 어설션을 사용자 풀의 사용자 프로필로 처리합니다. 속성 매핑 규칙에 따라 Amazon Cognito가 수신하기를 원하는 속성을 제어할 수 있습니다.
3. 퍼블릭 또는 페더레이션 가입을 건너뛰고 자체 데이터 소스 및 스키마를 기반으로 사용자를 생성할 수 있습니다. Amazon Cognito 콘솔 또는 API에서 직접 사용자를 추가합니다. CSV 파일에서 사용자를 가져옵니다. 기존 디렉터리에서 새 사용자를 찾고 기존 데이터로 사용자 프로필을 채우는 just-in-time AWS Lambda 함수를 실행하세요.

사용자가 가입한 후에는 Amazon Cognito가 액세스 및 ID 토큰에 나열하는 그룹에 사용자를 추가할 수 있습니다. ID 토큰을 자격 증명 풀에 전달할 때 사용자 풀 그룹을 IAM 역할에 연결할 수도 있습니다.

관련 주제

- 사용자 풀의 사용자 관리
- Amazon Cognito 사용자 풀 API 및 사용자 풀 엔드포인트 사용
- SDK를 사용하는 Amazon Cognito 자격 증명 공급자의 코드 예제 AWS

로그인

Amazon Cognito는 독립 실행형 사용자 디렉터리이자 앱에 대한 ID 제공업체(idP)가 될 수 있습니다. 사용자는 Amazon Cognito에서 호스팅하는 UI를 사용하거나 Amazon Cognito 사용자 풀 API를 통해 자체 UI를 사용하여 로그인할 수 있습니다. 프런트엔드 사용자 지정 UI 뒤에 있는 애플리케이션 계층은 몇 가지 방법 중 하나로 백엔드에서 요청을 승인하여 합법적인 요청을 확인할 수 있습니다.

필요할 경우 Amazon Cognito에 기본 제공된 사용자 디렉터리와 결합된 외부 디렉터리를 사용하여 사용자를 로그인하려면 다음 통합을 추가할 수 있습니다.

1. OAuth 2.0 소셜 로그인으로 로그인하고 소비자 사용자 데이터를 가져옵니다. Amazon Cognito는 OAuth 2.0을 통해 Google, Facebook, Amazon, Apple 로그인을 지원합니다.
2. SAML 및 OIDC 로그인으로 로그인하고 기업 사용자 데이터를 가져옵니다. SAML 또는 OpenID Connect(OIDC) ID 제공업체(idP)의 클레임을 수락하도록 Amazon Cognito를 구성할 수도 있습니다.
3. 외부 사용자 프로필을 기본 사용자 프로필에 연결합니다. 연결된 사용자는 서드 파티 사용자 자격 증명으로 로그인하고 기본 제공 디렉터리에서 사용자에게 할당한 액세스 권한을 받을 수 있습니다.

관련 주제

- 서드 파티를 통한 사용자 풀 로그인 추가
- 페더레이션 사용자를 기존 사용자 프로필에 연결

M 인증 achine-to-machine

일부 세션은 human-to-machine 상호 작용이 아닙니다. 자동화된 프로세스를 통해 API에 대한 요청을 승인할 수 있는 서비스 계정이 필요할 수 있습니다. OAuth 2.0 범위의 machine-to-machine 권한 부여를 위한 액세스 토큰을 생성하려면 클라이언트 자격 증명 부여를 생성하는 앱 클라이언트를 추가할 수 있습니다.

관련 주제

- 리소스 서버를 통한 범위, M2M 및 API 인증

호스팅된 UI

사용자 인터페이스를 구축하고 싶지 않은 경우 사용자 지정된 Amazon Cognito 호스팅 UI를 사용자에게 제공할 수 있습니다. 호스팅 UI는 가입, 로그인, 다중 인증(MFA), 암호 재설정을 위한 일련의 웹 페이지입니다. 호스팅된 UI를 기존 도메인에 추가하거나 하위 도메인에서 접두사 식별자를 사용할 수 있습니다. AWS

관련 주제

- Amazon Cognito 호스팅 UI 및 페더레이션 엔드포인트 설정 및 사용
- 사용자 풀 도메인 구성

보안

로컬 사용자는 SMS 메시지의 코드나 다중 인증(MFA) 코드를 생성하는 앱을 사용하여 추가 인증 요소를 제공할 수 있습니다. 앱에서 MFA를 설정 및 처리하기 위한 메커니즘을 구축하거나 호스팅 UI가 MFA를 관리하도록 할 수 있습니다. Amazon Cognito 사용자 풀은 사용자가 신뢰할 수 있는 디바이스에서 로그인할 때 MFA를 우회할 수 있습니다.

처음부터 사용자에게 MFA를 요구하기를 원치 않는 경우, 조건부로 MFA를 요구할 수 있습니다. Amazon Cognito는 고급 보안 기능을 통해 잠재적인 악성 활동을 탐지하고 MFA를 설정하도록 사용자에게 요구하거나 로그인을 차단할 수 있습니다.

사용자 풀로 향하는 네트워크 트래픽이 악의적일 수 있는 경우 이를 모니터링하고 AWS WAF 웹 ACL로 조치를 취할 수 있습니다.

관련 주제

- 사용자 풀에 MFA 추가
- 사용자 풀에 고급 보안 기능 추가
- 웹 ACL을 사용자 풀과 AWS WAF 연결

사용자 지정 사용자 경험

사용자의 가입, 로그인 또는 프로필 업데이트 단계 대부분에서 Amazon Cognito가 요청을 처리하는 방법을 사용자 지정할 수 있습니다. Lambda 트리거를 사용하면 사용자 지정 조건에 따라 ID 토큰을 수정하거나 가입 요청을 거부할 수 있습니다. 자체 사용자 지정 인증 흐름을 생성할 수 있습니다.

사용자 지정 CSS 및 로고를 업로드하여 사용자에게 친숙한 모양과 느낌을 호스팅 UI에 제공할 수 있습니다.

관련 주제

- Lambda 트리거를 사용하여 사용자 풀 워크플로 사용자 정의
- 사용자 정의 인증 챌린지 Lambda 트리거
- 기본 제공 로그인 및 가입 웹 페이지 사용자 정의

모니터링 및 분석

Amazon Cognito 사용자 풀은 호스팅 UI에 대한 요청을 비롯한 API 요청을 AWS CloudTrail에 로깅합니다. Amazon CloudWatch Logs에서 성능 지표를 검토하고, Lambda CloudWatch 트리거로 사용자 지정 로그를 푸시하고, Service Quotas 콘솔에서 API 요청 볼륨을 모니터링할 수 있습니다.

API 요청의 디바이스 및 세션 데이터를 Amazon Pinpoint 캠페인에 로깅할 수도 있습니다. Amazon Pinpoint를 사용하면 사용자 활동 분석을 기반으로 앱에서 푸시 알림을 보낼 수 있습니다.

관련 주제

- 를 사용하여 Amazon Cognito API 호출을 로깅합니다. AWS CloudTrail
- 및 Service Quotas의 CloudWatch 할당량 및 사용량 추적
- Amazon Cognito 사용자 풀에서 Amazon Pinpoint 분석 사용

Amazon Cognito 자격 증명 풀 통합

Amazon Cognito의 나머지 절반은 자격 증명 풀입니다. 자격 증명 풀은 예를 들어 Amazon DynamoDB 또는 Amazon S3에 대한 사용자의 API 요청을 승인하고 모니터링하는 자격 증명을 제공합니다. AWS 서비스사용자 풀에서 사용자를 분류하는 방식에 따라 데이터를 보호하는 자격 증명 기반 액세스 정책을 구축할 수 있습니다. 자격 증명 풀은 또한 사용자 풀 인증과 별개로 다양한 ID 제공업체(idP)의 토큰 및 SAML 2.0 어설션을 수락할 수 있습니다.

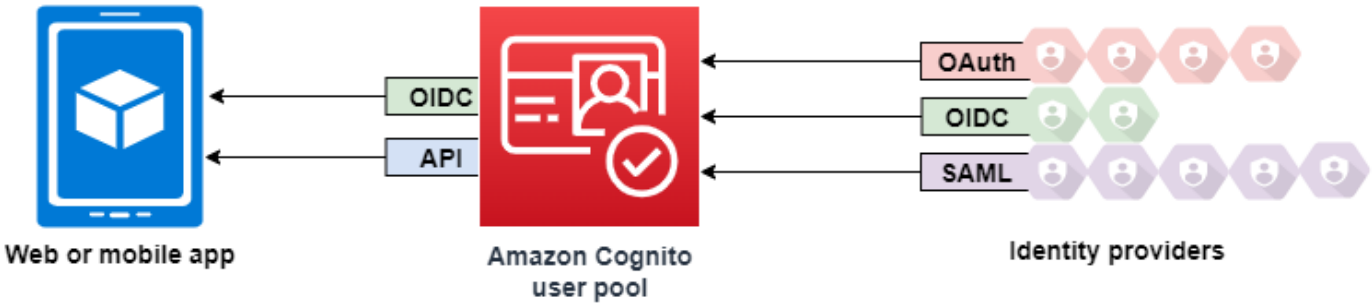
관련 주제

- 로그인 후 자격 증명 풀을 AWS 서비스 사용하여 액세스
- Amazon Cognito 자격 증명 풀

사용자 풀을 사용한 인증

앱 사용자는 사용자 풀을 통해 직접 로그인하거나 타사 ID 공급자 (IdP) 를 통해 페더레이션할 수 있습니다. 사용자 풀은 페이스북, 구글, 아마존, 애플을 통한 소셜 로그인과 OpenID Connect (OIDC) 및 SAML에서 반환되는 토큰을 처리하는 오버헤드를 관리합니다. IdPs

인증 성공 이후 Amazon Cognito는 앱으로 사용자 풀 토큰을 반환합니다. 이 토큰을 사용하여 자체 서버 측 리소스 또는 Amazon API Gateway에 대한 액세스 권한을 부여할 수 있습니다. 또는 다른 서비스에 액세스하기 위한 AWS 자격 증명으로 교환할 수도 있습니다. AWS



웹 또는 모바일 앱에서의 사용자 풀 토큰 처리 및 관리는 Amazon Cognito SDK를 통해 클라이언트 측에서 제공됩니다. 마찬가지로 유효한(기간이 만료되지 않은) 새로 고침 토큰이 존재하고 ID 및 액세스 토큰의 유효 기간이 최소 5분 남아 있는 경우, Mobile SDK for iOS 및 Mobile SDK for Android는 ID 및 액세스 토큰을 자동으로 새로 고침합니다. Android 및 iOS용 SDK 및 샘플 코드에 대한 JavaScript 자세한 내용은 Amazon Cognito 사용자 풀 SDK를 참조하십시오.

앱 사용자가 성공적으로 로그인하면 Amazon Cognito가 세션을 생성하고 인증된 사용자를 위한 ID, 액세스 및 새로 고침 토큰을 반환합니다.

JavaScript

```
// Amazon Cognito creates a session which includes the id, access, and refresh
 tokens of an authenticated user.

var authenticationData = {
```

```
        Username : 'username',
        Password : 'password',
    };
    var authenticationDetails = new
 AmazonCognitoIdentity.AuthenticationDetails(authenticationData);
    var poolData = { UserPoolId : 'us-east-1_ExaMPle',
        ClientId : '1example23456789'
    };
    var userPool = new AmazonCognitoIdentity.CognitoUserPool(poolData);
    var userData = {
        Username : 'username',
        Pool : userPool
    };
    var cognitoUser = new AmazonCognitoIdentity.CognitoUser(userData);
    cognitoUser.authenticateUser(authenticationDetails, {
        onSuccess: function (result) {
            var accessToken = result.getAccessToken().getJwtToken();

            /* Use the idToken for Logins Map when Federating User Pools with
 identity pools or when passing through an Authorization Header to an API Gateway
 Authorizer */
            var idToken = result.idToken.jwtToken;
        },

        onFailure: function(err) {
            alert(err);
        },

});
```

Android

```
// Session is an object of the type CognitoUserSession, and includes the id, access,
 and refresh tokens for a user.

String idToken = session.getIdToken().getJWTToken();
String accessToken = session.getAccessToken().getJWT();
```

iOS - swift

```
// AWSCognitoIdentityUserSession includes id, access, and refresh tokens for a user.

- (AWSTask<AWSCognitoIdentityUserSession *> *)getSession;
```

iOS - objective-C

```
// AWSCognitoIdentityUserSession includes the id, access, and refresh tokens for a
 user.

[[user getSession:@"username" password:@"password" validationData:nil scopes:nil]
 continueWithSuccessBlock:^id _Nullable(AWSTask<AWSCognitoIdentityUserSession *> *
 _Nonnull task) {
    // success, task.result has user session
    return nil;
}];
```

주제

- 사용자 풀 인증 흐름
- 사용자 풀 앱 클라이언트
- 사용자 풀의 사용자 디바이스 작업

사용자 풀 인증 흐름

Amazon Cognito에서는 다양한 방법으로 사용자를 인증할 수 있습니다. 도메인 유무에 관계없이 모든 사용자 풀은 사용자 풀 API에서 사용자를 인증할 수 있습니다. 사용자 풀에 도메인을 추가하면 사용자 풀 엔드포인트를 사용할 수 있습니다. 사용자 풀 API는 다양한 권한 부여 모델 및 API 요청의 요청 흐름을 지원합니다.

사용자의 자격 증명을 확인하기 위해 Amazon Cognito는 암호 외에 새로운 문제 유형도 통합하는 인증 흐름을 지원합니다. Amazon Cognito 인증을 사용하려면 일반적으로 두 API 작업을 다음 순서로 구현해야 합니다.

Public authentication

1. InitiateAuth
2. RespondToAuthChallenge

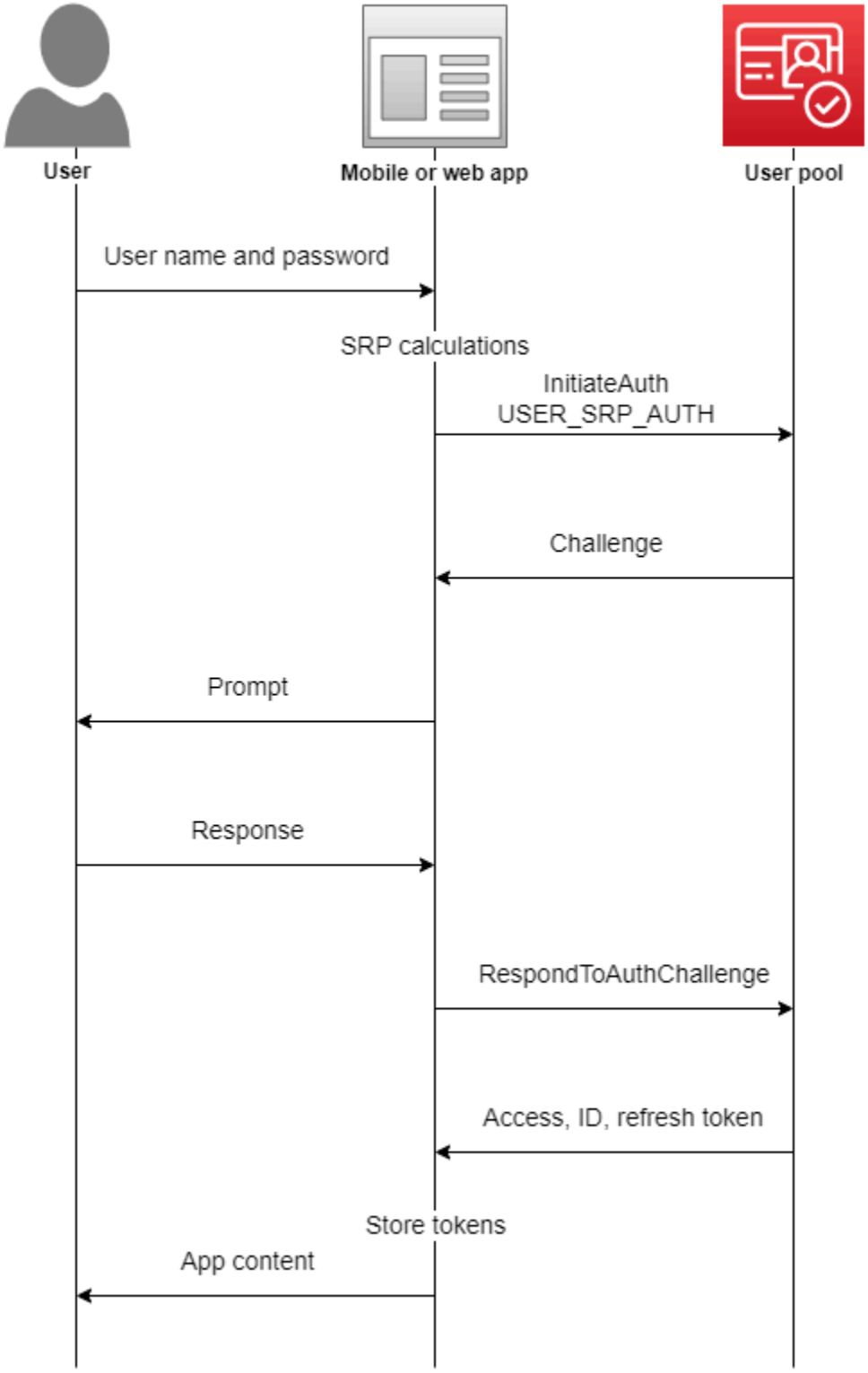
`InitiateAuth` 및 `RespondToAuthChallenge`는 클라이언트측 퍼블릭 앱 클라이언트와 함께 사용하기 위한 인증되지 않은 API입니다.

Server-side authentication

1. AdminInitiateAuth
2. AdminRespondToAuthChallenge

`AdminInitiateAuth` 및 `AdminRespondToAuthChallenge`는 IAM 보안 인증 정보를 필요로 하며 서버측 기밀 앱 클라이언트에 적합합니다.

인증이 실패하거나 Amazon Cognito가 사용자에게 토큰이 발행할 때까지 사용자는 연속 문제에 대답하여 인증합니다. 모든 사용자 지정 인증 흐름을 지원하기 위해 서로 다른 문제가 포함된 프로세스에서 Amazon Cognito를 통해 이러한 단계를 반복할 수 있습니다.


User
Mobile or web app
User pool
User name and password
SRP calculations
InitiateAuth
USER_SRP_AUTH
Challenge
Prompt
Response
RespondToAuthChallenge
Access, ID, refresh token
Store tokens
App content

일반적으로 앱은 사용자로부터 정보를 수집하라는 메시지를 생성하고 해당 정보를 API 요청으로 Amazon Cognito에 제출합니다. 다중 인증(MFA)으로 사용자를 구성한 사용자 풀의 `InitiateAuth` 흐름을 고려해 보겠습니다.

1. 사용자 이름과 암호를 입력하라는 메시지가 앱에 표시됩니다.
2. `InitiateAuth`에 사용자 이름과 암호를 파라미터로 포함합니다.
3. Amazon Cognito는 `SMS_MFA` 챌린지 및 세션 식별자를 반환합니다.
4. 앱에서 사용자에게 휴대폰에서 MFA 코드를 입력하라는 메시지를 표시합니다.
5. `RespondToAuthChallenge` 요청에 해당 코드와 세션 식별자를 포함합니다.

사용자 풀의 기능에 따라 앱이 Amazon Cognito에서 토큰을 검색하기 전에 `InitiateAuth`에 대한 몇 가지 챌린지에 응답할 수 있습니다. Amazon Cognito는 각 요청에 대한 응답에 세션 문자열을 포함합니다. API 요청을 인증 흐름에 결합하려면 이전 요청에 대한 응답의 세션 문자열을 각 후속 요청에 포함합니다. 기본적으로 사용자는 세션 문자열이 만료되기 전에 3분 동안 각 챌린지를 완료해야 합니다. 이 기간을 조정하려면 앱 클라이언트 Authentication flow session duration(인증 흐름 세션 기간)을 변경합니다. 다음 절차에서는 앱 클라이언트 구성에서 이 설정을 변경하는 방법을 설명합니다.

> **Note**
>
> 인증 흐름 세션 기간 설정은 Amazon Cognito 사용자 풀 API를 사용한 인증에 적용됩니다. Amazon Cognito 호스팅 UI는 세션 기간을 멀티 팩터 인증의 경우 3분, 암호 재설정 코드의 경우 8분으로 설정합니다.

Amazon Cognito console

앱 클라이언트 인증 흐름 세션 기간을 구성하는 방법(AWS Management Console)

1. 사용자 풀의 App integration(앱 통합) 탭에서 App clients and analytics(앱 클라이언트 및 분석) 컨테이너의 앱 클라이언트 이름을 선택합니다.
2. 앱 클라이언트 정보 컨테이너에서 편집을 선택합니다.
3. Authentication flow session duration(인증 흐름 세션 기간) 값을 SMS MFA 코드에 대해 원하는 유효 기간(분)으로 변경합니다. 이렇게 하면 사용자가 앱 클라이언트에서 인증 챌린지를 완료해야 하는 시간도 변경됩니다.
4. 변경 사항 저장를 선택합니다.

Amazon Cognito API

앱 클라이언트 인증 흐름 세션 기간을 구성하는 방법(Amazon Cognito API)

1. `DescribeUserPoolClient` 요청의 기존 사용자 풀 설정으로 `UpdateUserPoolClient` 요청을 준비합니다. `UpdateUserPoolClient` 요청에는 모든 기존 앱 클라이언트 속성이 포함되어야 합니다.
2. `AuthSessionValidity` 값을 SMS MFA 코드에 대해 원하는 유효 기간(분)으로 변경합니다. 이렇게 하면 사용자가 앱 클라이언트에서 인증 챌린지를 완료해야 하는 시간도 변경됩니다.

앱 클라이언트에 대한 자세한 내용은 사용자 풀 앱 클라이언트 섹션을 참조하세요.

AWS Lambda 트리거를 사용하여 사용자 인증 방식을 사용자 지정할 수 있습니다. 이러한 트리거는 인증 흐름의 일부로서 고유 챌린지를 발행 및 확인할 수 있습니다.

보안 백엔드 서버에 대해 관리자 인증 흐름을 사용할 수 있으며, 사용자 마이그레이션 인증 흐름을 사용하면 사용자에게 암호 재설정을 요구하지 않고도 사용자 마이그레이션을 수행할 수 있습니다.

로그인 시도 실패에 대한 Amazon Cognito 잠금 동작

암호를 사용한 인증되지 않은 로그인 시도 또는 IAM 인증 로그인 시도가 5회 실패하면 Amazon Cognito는 1초 동안 사용자를 잠금 처리합니다. 이후 한 번 더 실패할 때마다 잠금 시간이 두 배로 늘어나 최대 15분 정도까지 늘어납니다. 잠금 기간 중에 시도하면 `Password attempts exceeded` 예외가 발생하며, 이후 잠금 기간에 영향을 주지는 않습니다. 누적 로그인 시도 실패 n(`Password attempts exceeded`예외 제외)회에 대해, Amazon Cognito는 2^ (n-5)초 동안 사용자를 차단합니다. 잠금을 n=0 초기 상태로 재설정하려면 사용자는 잠금 기간이 만료된 후 성공적으로 로그인하거나 잠금이 설정된 후 연속으로 15분 동안 로그인 시도를 하지 않아야 합니다. 이러한 동작은 변경될 수 있습니다. 암호 기반 인증도 수행하지 않으면 이 동작은 사용자 지정 챌린지에는 적용되지 않습니다.

주제

- 클라이언트 측 인증 흐름
- 서버 측 인증 흐름
- 사용자 지정 인증 흐름
- 기본 제공 인증 흐름 및 챌린지
- 사용자 지정 인증 흐름 및 챌린지
- 사용자 지정 인증 흐름에서 SRP 암호 확인 사용
- 관리 인증 흐름

- 사용자 마이그레이션 인증 흐름

클라이언트 측 인증 흐름

다음 프로세스는 AWS Amplify 또는 AWS SDK로 생성한 사용자 클라이언트 측 앱을 대상으로 수행됩니다.

1. 사용자가 사용자 이름 및 암호를 앱에 입력합니다.
2. 앱이 사용자의 사용자 이름 및 Secure Remote Password(SRP) 세부 정보를 사용하여 `InitiateAuth` 작업을 호출합니다.

   이 API 작업은 인증 파라미터를 반환합니다.

   > **Note**
   >
   > 이 앱은 AWS SDK에 기본 제공된 Amazon Cognito SRP 기능을 사용하여 SRP 세부 정보를 생성합니다.

3. 앱이 `RespondToAuthChallenge` 작업을 호출합니다. 호출에 성공하면 Amazon Cognito에서 사용자 토큰을 반환하고 인증 흐름이 완료됩니다.

   Amazon Cognito에 다른 문제가 필요한 경우 `RespondToAuthChallenge`에 대한 호출이 토큰을 반환하지 않습니다. 대신 이러한 호출은 세션을 반환합니다.
4. `RespondToAuthChallenge`가 세션을 반환하는 경우 이번에는 앱이 세션 및 챌린지 응답(예: MFA 코드)과 함께 `RespondToAuthChallenge`를 다시 호출합니다.

서버 측 인증 흐름

사용자 앱이 없지만 대신 Java, Ruby 또는 Node.js 보안 백엔드 또는 서버 측 앱을 사용하는 경우 Amazon Cognito 사용자 풀에 대해 인증된 서버 측 API를 사용할 수 있습니다.

서버 측 앱의 경우 사용자 풀 인증은 클라이언트 측 앱에 대한 인증과 유사합니다. 단, 다음은 예외입니다.

- 서버 측 앱은 `AdminInitiateAuth` API 작업(`InitiateAuth` 대신)을 호출합니다. 이 작업을 수행하려면 및 를 포함하는 `cognito-idp:AdminInitiateAuth` 권한을 가진 AWS 자격 증명이 필요합니다. `cognito-idp:AdminRespondToAuthChallenge` 이 작업은 필수 인증 파라미터를 반환합니다.

- 서버 측 앱에 인증 파라미터가 있으면 `RespondToAuthChallenge` 대신 `AdminRespondToAuthChallenge` API 작업을 호출합니다. AWS 자격 증명을 제공한 경우에만 `AdminRespondToAuthChallenge` API 작업이 성공합니다.

AWS 자격 증명을 사용하여 Amazon Cognito API 요청에 서명하는 방법에 대한 자세한 내용은 AWS 일반 참조의 서명 버전 4 서명 프로세스를 참조하십시오.

다음 방법 중 하나로 명시적으로 활성화하지 않는 한 `AdminInitiateAuth` 및 `AdminRespondToAuthChallenge` API 작업에서는 관리자 로그인을 위한 username-and-password 사용자 자격 증명을 수락할 수 없습니다.

- `CreateUserPoolClient` 또는 `UpdateUserPoolClient`를 호출할 때 `ExplicitAuthFlow` 파라미터의 `ALLOW_ADMIN_USER_PASSWORD_AUTH`(이전에 `ADMIN_NO_SRP_AUTH`로 알려짐)를 포함합니다.
- 앱 클라이언트의 인증 흐름 목록에 `ALLOW_ADMIN_USER_PASSWORD_AUTH`를 추가합니다. 앱 클라이언트 및 분석(App clients and analytics) 아래에 있는 사용자 풀의 앱 통합(App integration) 탭에서 앱 클라이언트를 구성합니다. 자세한 내용은 사용자 풀 앱 클라이언트 섹션을 참조하세요.

사용자 지정 인증 흐름

또한 Amazon Cognito 사용자 풀을 사용하면 트리거를 사용하여 챌린지/응답 기반 인증 모델을 생성하는 데 도움이 되는 사용자 지정 인증 흐름을 사용할 수 있습니다. AWS Lambda

> **Note**
>
> 손상된 자격 증명 및 사용자 지정 인증 흐름을 통한 적응형 인증에는 고급 보안 기능을 사용할 수 없습니다. 자세한 내용은 사용자 풀에 고급 보안 기능 추가 섹션을 참조하세요.

사용자 지정 인증 흐름은 여러 요구 사항에 맞추어 사용자 지정 챌린지 및 응답 사이클을 허용하도록 고안되었습니다. 이 흐름은 사용할 인증 유형을 나타내고 초기 인증 파라미터를 제공하는 `InitiateAuth` API 작업을 호출하면서 시작됩니다. Amazon Cognito는 다음 정보 유형 중 하나를 사용하여 `InitiateAuth` 호출에 응답합니다.

- 세션 및 파라미터와 함께 사용자에 대한 챌린지
- 사용자가 인증에 실패할 경우 오류

- InitiateAuth 호출에서 사용자 로그인에 필요한 파라미터가 제공된 경우 ID, 액세스 및 새로 고침 토큰. (일반적으로 사용자나 앱은 먼저 챌린지에 답변해야 하지만 사용자 지정 코드에서 이를 결정해야 합니다.)

Amazon Cognito가 문제와 함께 InitiateAuth 호출에 응답하면 앱은 더 많은 입력을 수집하며 RespondToAuthChallenge 작업을 호출합니다. 이 호출은 문제 응답을 제공하고 세션에 다시 전달합니다. Amazon Cognito는 InitiateAuth 호출과 유사한 방식으로 RespondToAuthChallenge 호출에 응답합니다. 사용자가 로그인한 경우 Amazon Cognito에서 토큰을 제공하거나 사용자가 로그인하지 않은 경우 Amazon Cognito에서 다른 문제 또는 오류를 제공합니다. Amazon Cognito에서 다른 문제를 반환한 경우 사용자가 성공적으로 로그인하거나 오류가 반환될 때까지 시퀀스가 반복되고 앱에서 RespondToAuthChallenge를 호출합니다. InitiateAuth 및 RespondToAuthChallenge API 작업에 대한 자세한 내용은 API 설명서를 참조하세요.

기본 제공 인증 흐름 및 챌린지

Amazon Cognito에는 표준 인증을 위한 AuthFlow 및 ChallengeName 값이 기본으로 포함되어 Secure Remote Password(SRP) 프로토콜을 통해 사용자 이름과 암호가 확인됩니다. AWS SDK는 Amazon Cognito를 통해 이러한 흐름을 기본적으로 지원합니다.

이 흐름은 USER_SRP_AUTH를 InitiateAuth에 AuthFlow로 보내면 시작됩니다. AuthParameters의 USERNAME 및 SRP_A 값을 보내기도 합니다. InitiateAuth 호출이 성공하면 PASSWORD_VERIFIER가 챌린지 파라미터의 ChallengeName 및 SRP_B로 응답에 포함됩니다. 그러면 앱이 PASSWORD_VERIFIER ChallengeName 및 ChallengeResponses의 필수 파라미터를 사용하여 RespondToAuthChallenge를 호출합니다. RespondToAuthChallenge에 대한 호출이 성공하고 사용자가 로그인하는 경우 Amazon Cognito에서 토큰을 발급합니다. 사용자에 대해 멀티 팩터 인증(MFA)을 활성화한 경우 Amazon Cognito에서 SMS_MFA의 ChallengeName을 반환합니다. 앱은 RespondToAuthChallenge에 대한 다른 호출을 통해 필요한 코드를 제공할 수 있습니다.

사용자 지정 인증 흐름 및 챌린지

앱은 CUSTOM_AUTH를 Authflow로 지정해 InitiateAuth를 호출하여 사용자 지정 인증 흐름을 시작할 수 있습니다. 사용자 지정 인증 흐름을 사용하면 세 개의 Lambda 트리거에서 문제 및 응답의 확인을 제어합니다.

- DefineAuthChallenge Lambda 트리거는 이전 문제와 응답의 세션 배열을 입력으로 사용합니다. 그리고 나서 다음 문제 이름 및 사용자가 인증되었는지 그리고 토큰이 부여될 수 있는지를 나타내는 부울을 생성합니다. 이 Lambda 트리거는 문제를 통해 사용자의 경로를 제어하는 상태 시스템입니다.

- CreateAuthChallenge Lambda 트리거는 문제 이름을 입력으로 사용하고 문제 및 파라미터를 생성하여 응답을 평가합니다. DefineAuthChallenge에서 CUSTOM_CHALLENGE를 다음 문제로 반환하면 인증 흐름에서 CreateAuthChallenge를 호출합니다. 이 CreateAuthChallenge Lambda 트리거는 다음 문제 유형을 문제 메타데이터 파라미터로 전달합니다.
- VerifyAuthChallengeResponse Lambda 함수가 응답을 평가하고 부울을 반환하여 응답이 유효한지 여부를 나타냅니다.

사용자 지정 인증 흐름은 SRP 암호 확인 및 SMS를 통한 MFA와 같은 기본 제공 챌린지를 조합하여 사용할 수 있습니다. 또한 CAPTCHA 또는 본인 확인 질문과 같은 사용자 지정 챌린지를 사용할 수 있습니다.

사용자 지정 인증 흐름에서 SRP 암호 확인 사용

사용자 지정 인증 흐름에 SRP를 포함하려면 SRP를 시작해야 합니다.

- 앱은 사용자 지정 흐름에서 SRP 암호 확인을 시작하기 위해 CUSTOM_AUTH를 Authflow로 지정하여 InitiateAuth를 호출합니다. AuthParameters 맵에서는 앱으로부터의 요청에 SRP_A:(SRP A 값) 및 CHALLENGE_NAME: SRP_A가 포함됩니다.
- 이 CUSTOM_AUTH 흐름이 challengeName: SRP_A 및 challengeResult: true의 초기 세션에서 DefineAuthChallenge Lambda 트리거를 호출합니다. Lambda 함수는 challengeName: PASSWORD_VERIFIER, issueTokens: false 및 failAuthentication: false로 응답해야 합니다.
- 다음 앱은 challengeName: PASSWORD_VERIFIER 및 challengeResponses 맵의 SRP에 필요한 다른 파라미터와 함께 RespondToAuthChallenge를 호출해야 합니다.
- Amazon Cognito에서 암호를 확인하면 RespondToAuthChallenge에서 challengeName: PASSWORD_VERIFIER 및 challengeResult: true의 두 번째 세션과 함께 DefineAuthChallenge Lambda 트리거를 호출합니다. 이때 DefineAuthChallenge Lambda 트리거는 challengeName: CUSTOM_CHALLENGE로 응답하여 사용자 지정 문제를 시작할 수 있습니다.
- 사용자에 대해 MFA가 활성화된 경우 Amazon Cognito가 암호를 확인한 후 사용자에게 MFA를 사용하거나 설정하거나 로그인하라는 메시지가 표시됩니다.

> **Note**
>
> Amazon Cognito 호스팅 로그인 웹 페이지는 사용자 정의 인증 챌린지 Lambda 트리거를 활성화할 수 없습니다.

Lambda 트리거에 대한 자세한 내용은 Lambda 트리거를 사용하여 사용자 풀 워크플로 사용자 정의 섹션을 참조하세요.

관리 인증 흐름

인증에 대한 모범 사례는 암호 확인에 SRP와 함께 사용자 지정 인증 흐름에 설명된 API 작업을 사용하는 것입니다. AWS SDK는 이 접근 방식을 사용하며, 이 접근 방식은 SDK가 SRP를 사용하는 데 도움이 됩니다. 그러나 SRP 계산을 원치 않는 경우 보안 백엔드 서버에 관리 API 작업 세트를 대신 사용할 수 있습니다. 이러한 백엔드 관리자 구현의 경우 `InitiateAuth` 대신 `AdminInitiateAuth`를 사용합니다. 또한, `AdminRespondToAuthChallenge` 대신 `RespondToAuthChallenge`를 사용합니다. 암호를 일반 텍스트로 제출할 수 있으므로 이러한 작업을 사용할 때 SRP 계산을 수행할 필요가 없습니다. 예:

```
AdminInitiateAuth Request {
    "AuthFlow":"ADMIN_USER_PASSWORD_AUTH",
    "AuthParameters":{
        "USERNAME":"<username>",
        "PASSWORD":"<password>"
        },
    "ClientId":"<clientId>",
    "UserPoolId":"<userPoolId>"
}
```

이 관리 인증 작업에는 개발자 자격 증명이 필요하며 AWS 서명 버전 4(SigV4) 서명 프로세스가 사용됩니다. 이 작업은 Node.js를 포함하여 Lambda 함수에 편리하게 쓸 수 있는 표준 AWS SDK에서 사용할 수 있습니다. 이러한 작업을 사용하고 이러한 작업에서 일반 텍스트로 된 암호를 수락하도록 하려면 콘솔에서 앱에 대해 이러한 작업을 활성화해야 합니다. `CreateUserPoolClient` 또는 `UpdateUserPoolClient` 호출 시 `ExplicitAuthFlow` 파라미터에 `ADMIN_USER_PASSWORD_AUTH`를 전달해도 됩니다. `InitiateAuth` 및 `RespondToAuthChallenge` 작업은 `ADMIN_USER_PASSWORD_AUTH` `AuthFlow`를 수락하지 않습니다.

AdminInitiateAuth 응답 ChallengeParameters에 USER_ID_FOR_SRP 속성이 있으면 별칭(이메일 주소 또는 전화번호)이 아닌 사용자의 실제 사용자 이름이 이 속성에 포함됩니다. AdminRespondToAuthChallenge를 호출할 때 ChallengeResponses에서 USERNAME 파라미터에 이 사용자 이름을 전달해야 합니다.

> **Note**
>
> 백엔드 관리 구현은 관리 인증 흐름을 사용하기 때문에 이러한 흐름에서 디바이스 추적을 지원하지 않습니다. 디바이스 추적이 설정되면 관리 인증이 성공하지만 액세스 토큰을 새로 고치는 호출은 실패합니다.

사용자 마이그레이션 인증 흐름

사용자 마이그레이션 Lambda 트리거는 기존 사용자 관리 시스템에서 사용자 풀로의 마이그레이션을 도와줍니다. USER_PASSWORD_AUTH 인증 흐름을 선택하면 사용자 마이그레이션 과정에서 사용자가 암호를 재설정할 필요가 없습니다. 이 흐름은 인증 시 암호화된 SSL 연결을 통해 사용자의 암호를 서비스로 보냅니다.

모든 사용자를 마이그레이션하면 이 흐름을 보다 안전한 SRP 흐름으로 전환합니다. SRP 흐름은 네트워크를 통해 암호를 보내지 않습니다.

Lambda 트리거에 대해 자세히 알아보려면 Lambda 트리거를 사용하여 사용자 풀 워크플로 사용자 정의 섹션을 참조하세요.

Lambda 트리거를 사용한 사용자 마이그레이션에 대한 자세한 내용은 사용자 마이그레이션 Lambda 트리거를 사용하여 사용자 풀로 사용자 가져오기 섹션을 참조하세요.

사용자 풀 앱 클라이언트

사용자 풀 앱 클라이언트는 Amazon Cognito로 인증되는 모바일 또는 웹 애플리케이션과 상호 작용하는 사용자 풀 내의 구성입니다. 앱 클라이언트는 인증된 API 작업과 인증되지 않은 API 작업을 호출하고 사용자 속성 전체 또는 일부를 읽거나 수정할 수 있습니다. 등록, 로그인 및 암호 찾기 작업을 처리하려면 앱 클라이언트에 대해 앱이 자체 식별을 수행해야 합니다. 이러한 API 요청에는 앱 클라이언트 ID를 사용한 자체 식별 및 선택적 클라이언트 암호를 사용한 권한 부여가 포함되어야 합니다. 개발자는 권한이 있는 클라이언트 앱만 이러한 미인증 작업을 호출할 수 있도록 앱 클라이언트 ID 또는 암호를 보호할 책임이 있습니다. 또한 자격 증명을 사용하여 인증된 API 요청에 서명하도록 앱을 구성하는 경우 사용자 검사로부터 AWS 자격 증명을 보호해야 합니다.

사용자 풀에 여러 앱 클라이언트를 생성할 수 있습니다. 앱 클라이언트는 앱의 코드 플랫폼 또는 사용자 풀의 별도 테넌트에 연결할 수 있습니다. 예를 들어 서버 측 애플리케이션에 대한 앱과 다른 Android 앱을 생성할 수 있습니다. 각 앱에는 고유한 앱 클라이언트 ID가 있습니다.

앱 클라이언트 유형

Amazon Cognito에서 앱 클라이언트를 생성할 때 표준 OAuth 클라이언트 유형 퍼블릭 클라이언트(public client) 및 기밀 클라이언트(confidential client)에 따라 옵션을 미리 채울 수 있습니다. 기밀 클라이언트(confidential client)를 클라이언트 암호(client secret)와 함께 구성합니다. 클라이언트 유형에 대한 자세한 내용은 IETF RFC 6749 #2.1을 참조하세요.

퍼블릭 클라이언트(Public client)

퍼블릭 클라이언트는 브라우저 또는 모바일 디바이스에서 실행됩니다. 퍼블릭 클라이언트에는 신뢰할 수 있는 서버 측 리소스가 없으므로 클라이언트 암호가 없습니다.

기밀 클라이언트(Confidential client)

기밀 클라이언트에는 인증되지 않은 API 작업에 대해 클라이언트 암호(client secret)로 신뢰할 수 있는 서버 측 리소스가 있습니다. 앱은 백엔드 서버에서 데몬 또는 셸 스크립트로 실행될 수 있습니다.

클라이언트 암호(client secret)

클라이언트 암호는 앱이 앱 클라이언트에 대한 모든 API 요청에서 사용해야 하는 고정 문자열입니다. 앱 클라이언트에는 `client_credentials`를 부여할 클라이언트 암호가 있어야 합니다. 자세한 내용은 IETF RFC 6749 #2.3.1을 참조하세요.

앱을 생성한 후에는 보안 암호를 변경할 수 없습니다. 보안 암호를 교체하려는 경우 새 보안 암호로 새 앱을 생성할 수 있습니다. 앱을 삭제하여 해당 앱 클라이언트 ID를 사용하는 앱의 액세스를 차단할 수도 있습니다.

기밀 클라이언트와 클라이언트 암호를 공개 앱과 함께 사용할 수 있습니다. Amazon CloudFront 프록시를 사용하여 전송 `SECRET_HASH` 중인 이메일을 추가합니다. 자세한 내용은 블로그에서 Amazon CloudFront 프록시를 사용하여 Amazon Cognito용 퍼블릭 클라이언트 보호를 참조하십시오. AWS

JSON 웹 토큰

Amazon Cognito 앱 클라이언트는 다음 유형의 JSON 웹 토큰(JWT)을 발급할 수 있습니다.

자격 증명(ID) 토큰

사용자가 사용자 풀에서 인증되었음을 검증할 수 있는 명령문입니다. OpenID Connect(OIDC)는 OAuth 2.0에서 정의한 액세스 및 새로 고침 토큰 표준에 ID 토큰 사양을 추가했습니다. ID 토큰에는 앱이 사용자 프로필을 생성하고 리소스를 프로비저닝하는 데 사용할 수 있는 사용자 속성과 같은 자격 증명 정보가 들어 있습니다. 자세한 정보는 ID 토큰 사용을 참조하세요.

액세스 토큰

사용자 액세스 권한에 대한 검증 가능한 문입니다. 액세스 토큰에는 OIDC 및 OAuth 2.0의 기능인 범위가 포함됩니다. 앱은 백엔드 리소스에 대한 범위를 제공하고 사용자 풀이 사용자 또는 시스템에 API의 데이터 또는 자체 사용자 데이터에 액세스할 수 있는 권한을 부여했음을 입증할 수 있습니다. 사용자 지정 범위가 있는 액세스 토큰(주로 M2M client-credentials 권한 부여)은 리소스 서버에 대한 액세스를 승인합니다. 자세한 정보는 액세스 토큰 사용을 참조하세요.

새로 고침 토큰

사용자 토큰이 만료될 때 앱이 사용자 풀에 제공할 수 있는 암호화된 초기 인증 문입니다. refresh-token 요청은 만료되지 않은 새 액세스 및 ID 토큰을 반환합니다. 자세한 정보는 새로 고침 토큰 사용을 참조하세요.

Amazon Cognito 콘솔에 있는 사용자 풀의 앱 통합 탭에서 각 앱 클라이언트에 대해 이러한 토큰의 만료를 설정할 수 있습니다.

앱 클라이언트 용어

다음 용어는 Amazon Cognito 콘솔에서 사용할 수 있는 앱 클라이언트 속성입니다.

허용된 콜백 URL

콜백 URL은 로그인 성공 후 사용자가 리디렉션되는 위치를 나타냅니다. 하나 이상의 콜백 URL을 선택합니다. 콜백 URL은 다음을 수행해야 합니다.

- 절대 URI이어야 합니다.
- 클라이언트로 미리 등록되어야 합니다.
- 조각 구성요소가 없어야 합니다.

OAuth 2.0 - 리디렉션 엔드포인트를 참조하세요.

테스트 목적으로만 `http://localhost`를 사용하는 경우를 제외하고 Amazon Cognito에서는 HTTP가 아니라 HTTPS가 필요합니다.

`myapp://example` 같은 앱 콜백 URL도 지원됩니다.

허용된 로그아웃 URL

로그아웃 URL은 로그아웃 후 사용자가 리디렉션되는 위치를 나타냅니다.

속성 읽기 및 쓰기 권한

사용자 풀에는 각각 고유한 앱 클라이언트와 IdPs 앱을 보유한 고객이 많을 수 있습니다. 앱과 관련된 사용자 속성에 대해서만 읽기 및 쓰기 액세스 권한을 갖도록 앱 클라이언트를 구성할 수 있습니다. machine-to-machine (M2M) 권한 부여와 같은 경우에는 어떤 사용자 속성에도 액세스 권한을 부여할 수 없습니다.

속성 읽기 및 쓰기 권한 구성 시 고려 사항

- 앱 클라이언트를 생성하고 속성 읽기 및 쓰기 권한을 사용자 지정하지 않으면 Amazon Cognito는 모든 사용자 풀 속성에 읽기 및 쓰기 권한을 부여합니다.
- 변경할 수 없는 사용자 지정 속성에 쓰기 액세스 권한을 부여할 수 있습니다. 앱 클라이언트는 사용자를 생성 또는 가입시킬 때만 변경 불가능한 사용자 지정 속성에 값을 쓸 수 있습니다. 이후에는 사용자의 변경 불가능한 사용자 지정 속성에 값을 쓸 수 없습니다.
- 앱 클라이언트는 사용자 풀의 필수 속성에 대한 쓰기 권한을 가져야 합니다. Amazon Cognito 콘솔은 필수 속성을 쓰기 가능으로 자동 설정합니다.
- 앱 클라이언트가 `email_verified` 또는 `phone_number_verified`에 대한 쓰기 액세스 권한을 갖도록 허용할 수 없습니다. 사용자 풀 관리자는 이러한 값을 수정할 수 있습니다. 사용자는 속성 확인을 통해서만 이러한 속성의 값을 변경할 수 있습니다.

인증 흐름

앱 클라이언트에서 로그인을 허용하는 방법입니다. 앱은 사용자 이름 및 암호를 사용한 인증, 보안 원격 암호(SRP), Lambda 트리거를 사용한 사용자 지정 인증, 토큰 새로 고침을 지원할 수 있습니다. 보안 모범 사례에 따라 SRP 인증을 기본 로그인 방법으로 사용합니다. 호스팅 UI는 SRP를 사용하여 사용자를 자동으로 로그인시킵니다.

사용자 지정 범위

사용자 지정 범위는 리소스 서버에서 자체 리소스 서버에 대해 정의할 수 있습니다. 형식은 *`resource-server-identifier/scope###`*. 리소스 서버를 통한 범위, M2M 및 API 인증를 참조하세요.

기본 리디렉션 URI

타사 사용자에 대한 인증 요청의 `redirect_uri` 파라미터를 대체합니다. IdPs CreateUserPoolClient또는 UpdateUserPoolClientAPI 요청의 `DefaultRedirectURI` 파라미터로

이 앱 클라이언트 설정을 구성합니다. 또한 이 URL은 앱 `CallbackURLs` 클라이언트의 멤버여야 합니다. Amazon Cognito는 다음과 같은 경우 인증된 세션을 이 URL로 리디렉션합니다.

1. 앱 클라이언트에는 하나의 ID 공급자가 할당되고 여러 콜백 URL이 정의되어 있습니다. 매개변수가 포함되지 않은 경우 사용자 풀은 인증 요청을 권한 부여 서버의 기본 리디렉션 URI로 리디렉션합니다. `redirect_uri`
2. 앱 클라이언트에는 ID 제공자 1개가 할당되고 콜백 URL 1개가 정의되어 있습니다. 이 시나리오에서는 기본 콜백 URL을 정의할 필요가 없습니다. `redirect_uri`매개변수가 포함되지 않은 요청은 사용 가능한 하나의 콜백 URL로 리디렉션됩니다.

자격 증명 공급자

사용자 풀 외부 ID 공급자 (IdPs) 중 일부 또는 전체를 선택하여 사용자를 인증할 수 있습니다. 앱 클라이언트는 사용자 풀의 로컬 사용자만 인증할 수도 있습니다. 앱 클라이언트에 IdP 추가 시 IdP에 대한 권한 부여 링크를 생성하여 호스팅 UI 로그인 페이지에 표시할 수 있습니다. 여러 개를 할당할 수 IdPs 있지만 최소한 하나는 할당해야 합니다. 외부 사용에 대한 자세한 내용은 IdPs 을 참조하십시오서드 파티를 통한 사용자 풀 로그인 추가.

OpenID Connect 범위

다음 `OAuth` 범위에서 하나 이상을 선택하여 액세스 토큰에 대해 요청될 수 있는 액세스 권한을 지정합니다.

- `openid` 범위는 사용자가 ID 토큰과 사용자의 고유 ID를 검색하고 싶어 한다고 선언합니다. 또한 요청의 추가 범위에 따라 전체 또는 일부 사용자 속성을 요청합니다. Amazon Cognito는 사용자가 `openid` 범위를 요청할 때만 ID 토큰을 반환합니다. `openid` 범위는 만료 및 키 ID와 같은 구조적 ID 토큰 클레임을 승인하고 UserInfo 엔드포인트의 응답으로 받는 사용자 속성을 결정합니다.
  - 요청한 유일한 범위가 `openid`가 되는 경우 Amazon Cognito는 현재 앱 클라이언트가 읽을 수 있는 모든 사용자 속성으로 ID 토큰을 채웁니다. 이 범위의 액세스 토큰에 대한 `userInfo` 응답만으로도 모든 사용자 속성이 반환됩니다.
  - `phone`, `email`, `profile` 또는 ID 토큰과 같은 다른 범위로 `openid`를 요청하고 사용자의 고유 ID와 추가 범위에 의해 정의된 속성을 `userInfo`가 반환하는 경우입니다.
- `phone` 범위는 `phone_number` 및 `phone_number_verified` 클레임에 대한 액세스 권한을 부여합니다. 이 범위는 `openid` 범위를 통해서만 요청할 수 있습니다.
- `email` 범위는 `email` 및 `email_verified` 클레임에 대한 액세스 권한을 부여합니다. 이 범위는 `openid` 범위를 통해서만 요청할 수 있습니다.
- `aws.cognito.signin.user.admin`범위는 액세스 토큰이 필요한 Amazon Cognito 사용자 풀 API 작업 (예: 및) 에 대한 액세스 권한을 부여합니다. UpdateUserAttributesVerifyUserAttribute

- `profile` 범위는 클라이언트가 읽을 수 있는 모든 사용자 속성에 대한 액세스 권한을 부여합니다. 이 범위는 `openid` 범위를 통해서만 요청할 수 있습니다.

범위에 대한 자세한 내용은 표준 OIDC 범위 목록을 참조하세요.

OAuth 권한 부여 유형

OAuth 권한 부여는 사용자 풀 토큰을 검색하는 인증 방법입니다. Amazon Cognito는 다음 유형의 권한 부여를 지원합니다. 이러한 OAuth 권한 부여를 앱에 통합하려면 도메인을 사용자 풀에 추가해야 합니다.

인증 코드 권한 부여

인증 코드 권한 부여는 앱이 Token 엔드포인트를 사용하여 사용자 풀 토큰과 교환할 수 있는 코드를 생성합니다 인증 코드를 교환하면 앱이 ID, 액세스 및 새로 고침 토큰을 받습니다. 이 OAuth 흐름은 암시적 권한 부여와 마찬가지로 사용자의 브라우저에서 발생합니다. 인증 코드 권한 부여는 Amazon Cognito에서 제공하는 가장 안전한 권한 부여입니다. 토큰이 사용자 세션에 표시되지 않기 때문입니다. 대신 앱은 요청을 생성하여 토큰을 반환하고 이를 보호된 스토리지에 캐시합니다. 자세한 내용은 IETF 6749 #1.3.1의 인증 코드를 참조하세요.

> **Note**
>
> 퍼블릭 클라이언트 앱의 보안 모범 사례는 인증 코드 권한 부여 OAuth 흐름만 활성화하고 코드 교환용 증명 키(PKCE)를 구현하여 토큰 교환을 제한하는 것입니다. PKCE를 사용하는 경우 클라이언트가 토큰 엔드포인트에 원래 인증 요청에서 제시한 것과 동일한 암호를 제공한 경우에만 인증 코드를 교환할 수 있습니다. PKCE에 대한 자세한 내용은 IETF RFC 7636을 참조하세요.

암시적 허용

암시적 권한 부여는 권한 부여 엔드포인트에서 직접 사용자의 브라우저 세션에 액세스 및 ID 토큰을 전달하지만 새로 고침 토큰은 전달하지 않습니다. 암시적 권한 부여를 사용하면 토큰 엔드포인트를 별도로 요청할 필요가 없지만, 암시적 권한 부여는 PKCE와 호환되지 않으며 새로 고침 토큰을 반환하지 않습니다. 이 권한 부여는 인증 코드 권한 부여를 완료할 수 없는 테스트 시나리오와 앱 아키텍처에도 적용할 수 있습니다. 자세한 내용은 IETF RFC 6749 #1.3.2의 암시적 권한 부여를 참조하세요. 인증 코드 권한 부여 및 암시적 권한 부여를 모두 활성화하고 각 권한 부여를 필요한 경우 사용할 수 있습니다.

클라이언트 보안 인증 권한 부여

클라이언트 자격 증명 부여는 machine-to-machine (M2M) 통신을 위한 것입니다. 인증 코드 및 암시적 권한 부여는 인증된 인간 사용자에게 토큰을 발급합니다. 클라이언트 보안 인증은 비대화형 시스템에서 API에 범위 기반 권한을 부여합니다. 앱은 토큰 엔드포인트에서 직접 클라이언트 보안 인증 정보를 요청하고 액세스 토큰을 받을 수 있습니다. 자세한 내용은 IETF RFC 6749 #1.3.4의 클라이언트 보안 인증을 참조하세요. 클라이언트 암호가 있고 인증 코드 또는 암시적 권한 부여를 지원하지 않는 앱 클라이언트에서만 클라이언트 보안 인증 권한 부여를 활성화할 수 있습니다.

> **Note**
>
> 사용자 자격으로 클라이언트 보안 인증 정보 흐름을 호출하지 않으므로, 이 권한 부여는 액세스 토큰에 사용자 지정 범위만 추가할 수 있습니다. 사용자 지정 범위는 자체 리소스 서버에 대해 정의할 수 있습니다. `openid` 및 `profile` 같은 기본 범위는 비 인간 사용자에게 적용되지 않습니다.
>
> ID 토큰은 사용자 속성의 검증이기 때문에 M2M 커뮤니케이션과는 관련이 없으며 클라이언트 보안 인증 권한 부여를 통해서도 발급되지 않습니다. 리소스 서버를 통한 범위, M2M 및 API 인증를 참조하세요.

클라이언트 자격 증명 부여는 청구서에 비용을 추가합니다. AWS 자세한 내용은 Amazon Cognito 요금을 참조하세요.

앱 클라이언트 생성

AWS Management Console

앱 클라이언트를 생성하려면(콘솔)

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 AWS 자격 증명을 입력합니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. [앱 통합(App integration)] 탭을 선택합니다.
5. [앱 클라이언트(App clients)]에서 [앱 클라이언트 생성(Create an app client)]을 선택합니다.
6. [앱 유형(App type)]에서 [퍼블릭 클라이언트(Public client)], [기밀 클라이언트(Confidential client)] 또는 [기타(Other)]를 선택합니다.
7. [앱 클라이언트 이름(App client name)]을 입력합니다.

8. [클라이언트 암호 생성]을 선택하여 Amazon Cognito에서 클라이언트 암호를 자동으로 생성하도록 합니다. 클라이언트 암호는 일반적으로 기밀 클라이언트와 연결됩니다.
9. 앱 클라이언트에서 허용할 [인증 흐름(Authentication flows)]을 선택합니다.
10. Authentication flow session duration(인증 흐름 세션 기간)을 구성합니다. 세션 토큰이 만료되기 전에 사용자가 각 인증 챌린지를 완료해야 하는 시간입니다.
11. (선택 사항) 토큰 만료를 구성하려는 경우 다음 단계를 완료합니다.

    a. 앱 클라이언트의 [새로 고침 토큰 만료(Refresh token expiration)]를 지정합니다. 기본값은 30일입니다. 이 값을 1시간에서 10년 사이의 값으로 변경할 수 있습니다.

    b. 앱 클라이언트의 [액세스 토큰 만료(Access token expiration)]를 지정합니다. 기본값은 1시간입니다. 이 값을 5분에서 24시간 사이의 값으로 변경할 수 있습니다.

    c. 앱 클라이언트의 [ID 토큰 만료(ID token expiration)]를 지정합니다. 기본값은 1시간입니다. 이 값을 5분에서 24시간 사이의 값으로 변경할 수 있습니다.

    > **Important**
    >
    > 호스트된 UI를 사용하여 1시간 미만의 토큰 수명을 구성할 경우 사용자는 현재 1시간으로 고정된 세션 쿠키 기간에 따라 토큰을 사용할 수 있습니다.

12. [토큰 철회 사용(Enable token revocation)]에서 이 앱 클라이언트에 토큰 철회를 사용할지 여부를 선택합니다. 사용할 경우 Amazon Cognito가 발급하는 토큰의 크기가 늘어납니다.
13. 이 앱 클라이언트에 대해 [사용자 존재 오류 메시지 방지] 여부를 선택합니다. Amazon Cognito는 존재하지 않는 사용자에 대한 로그인 요청에 사용자 이름이나 암호가 잘못되었다는 일반적인 메시지로 응답합니다.
14. 이 앱 클라이언트에서 호스팅된 UI를 사용하려면 호스팅된 UI 설정을 구성하십시오.

    a. 허용된 콜백 URL을 하나 이상 입력합니다. 이러한 URL은 사용자가 인증을 완료한 후 Amazon Cognito에서 사용자를 리디렉션하도록 하려는 웹 또는 앱 URL입니다.

    b. 허용된 로그아웃 URL을 하나 이상 입력합니다. 이러한 URL은 앱에서 Logout 엔드포인트 요청에 수락하기를 원하는 URL입니다.

    c. 앱의 사용자를 로그인시킬 수 있도록 하려는 자격 증명 공급자를 하나 이상 선택합니다. 기존 항목 중 원하는 조합을 선택할 수 IdPs 있습니다. 사용자 풀만 사용하여 사용자를 인증하거나 사용자 풀에 구성한 한 명 이상의 IdPs 제3자를 사용하여 사용자를 인증할 수 있습니다.

    d. 앱 클라이언트에서 수락할 OAuth 2.0 권한 부여 유형을 선택합니다.

- 인증 코드 부여를 선택하여 Token 엔드포인트로 앱에서 토큰으로 교환할 수 있는 코드를 앱에 전달합니다.
- 암시적 부여를 선택하여 ID와 액세스 토큰을 앱에 직접 전달합니다. 암시적 권한 부여 흐름은 토큰을 사용자에게 직접 노출합니다.
- 사용자 자격 증명이 아니라 클라이언트 암호에 대한 지식을 기반으로 앱에 액세스 토큰을 전달하려면 클라이언트 자격 증명을 선택합니다. 클라이언트 자격 증명 부여 흐름은 권한 부여 코드 및 암시적 권한 부여 흐름과 상호 배타적입니다.

e. 사용자의 앱 클라이언트에 권한을 부여할 [OpenID Connect 범위]를 선택합니다. 사용자 풀 API를 통해 `aws.cognito.signin.user.admin` 범위만 포함하는 액세스 토큰을 생성할 수 있습니다. 추가 범위의 경우 Token 엔드포인트에서 액세스 토큰을 요청해야 합니다.

f. 앱 클라이언트에서 승인하려는 사용자 지정 범위를 선택합니다. 사용자 지정 범위는 대부분 타사 API에 대한 액세스를 승인하는 데 사용됩니다.

15. 이 앱 클라이언트에 대한 [속성 읽기 및 쓰기 권한]을 구성합니다. 앱 클라이언트는 사용자 풀 속성 스키마의 제한된 하위 집합에 대한 읽기 및 쓰기 권한을 가질 수 있습니다.
16. 앱 클라이언트 생성을 선택합니다.
17. [클라이언트 ID(Client id)]를 적어 둡니다. 클라이언트 ID는 가입 및 로그인 요청에서 앱 클라이언트를 식별합니다.

AWS CLI

```
aws cognito-idp create-user-pool-client --user-pool-id MyUserPoolID --client-
name myApp
```

> **Note**
>
> CLI가 원격 파라미터 파일로 처리하지 않도록 콜백 및 로그아웃 URL에 JSON 형식을 사용합니다.
>
> ```
> --callback-urls "["https://example.com"]"
> --logout-urls "["https://example.com"]"
> ```

자세한 내용은 AWS CLI 명령 참조를 참조하십시오. create-user-pool-client

Amazon Cognito user pools API

CreateUserPoolClientAPI 요청 생성. 기본값으로 설정하지 않으려는 모든 파라미터의 값을 지정해야 합니다.

사용자 풀 앱 클라이언트 (AWS CLI 및 AWS API) 업데이트

에서 AWS CLI다음 명령을 입력합니다.

```
aws cognito-idp update-user-pool-client --user-pool-id  "MyUserPoolID" --client-id
 "MyAppClientID" --allowed-o-auth-flows-user-pool-client --allowed-o-auth-flows "code"
 "implicit" --allowed-o-auth-scopes "openid" --callback-urls "["https://example.com"]"
 --supported-identity-providers "["MySAMLIdP", "LoginWithAmazon"]"
```

명령이 성공하면 확인 메시지가 AWS CLI 반환됩니다.

```
{
    "UserPoolClient": {
        "ClientId": "MyClientID",
        "SupportedIdentityProviders": [
            "LoginWithAmazon",
            "MySAMLIdP"
        ],
        "CallbackURLs": [
            "https://example.com"
        ],
        "AllowedOAuthScopes": [
            "openid"
        ],
        "ClientName": "Example",
        "AllowedOAuthFlows": [
            "implicit",
            "code"
        ],
        "RefreshTokenValidity": 30,
        "AuthSessionValidity": 3,
        "CreationDate": 1524628110.29,
        "AllowedOAuthFlowsUserPoolClient": true,
        "UserPoolId": "MyUserPoolID",
        "LastModifiedDate": 1530055177.553
    }
}
```

자세한 내용은 AWS CLI 명령 참조를 참조하십시오 update-user-pool-client.

AWS API: UpdateUserPoolClient

사용자 풀 앱 클라이언트 (AWS CLI 및 AWS API) 에 대한 정보 가져오기

```
aws cognito-idp describe-user-pool-client --user-pool-id MyUserPoolID --client-
id MyClientID
```

자세한 내용은 AWS CLI 명령 참조를 참조하십시오 describe-user-pool-client.

AWS API: DescribeUserPoolClient

사용자 풀 (AWS CLI 및 AWS API) 의 모든 앱 클라이언트 정보 나열

```
aws cognito-idp list-user-pool-clients --user-pool-id "MyUserPoolID" --max-results 3
```

자세한 내용은 AWS CLI 명령 참조를 참조하십시오 list-user-pool-clients.

AWS API: ListUserPoolClients

사용자 풀 앱 클라이언트 (AWS CLI 및 AWS API) 삭제

```
aws cognito-idp delete-user-pool-client --user-pool-id "MyUserPoolID" --client-id
 "MyAppClientID"
```

자세한 내용은 AWS CLI 명령 참조를 참조하십시오. delete-user-pool-client

AWS API: DeleteUserPoolClient

사용자 풀의 사용자 디바이스 작업

Amazon Cognito 사용자 풀 API로 로컬 사용자 풀 사용자를 로그인하면 고급 보안 기능의 사용자 활동 로그를 각 디바이스와 연결할 수 있으며, 선택적으로 사용자가 신뢰할 수 있는 디바이스를 사용하는 경우 다중 인증(MFA)을 건너뛰도록 허용할 수 있습니다. Amazon Cognito는 디바이스 정보가 아직 포함되어 있지 않은 모든 로그인에 대한 응답에 디바이스 키를 포함시킵니다. 디바이스 키의 형식은 *Region_UUID*입니다. 디바이스 키, SRP(Secure Remote Password) 라이브러리, 디바이스 인증을 허용하는 사용자 풀을 사용하면, 앱 사용자에게 현재 디바이스를 신뢰하라는 메시지를 표시하고 로그인 시 MFA 코드를 묻는 메시지를 더 이상 표시하지 않을 수 있습니다.

주제

기억된 디바이스 설정

Amazon Cognito 사용자 풀을 사용하면 각 사용자의 디바이스를 고유한 디바이스 식별자인 디바이스 키와 연결할 수 있습니다. 로그인 시 디바이스 키를 제시하고 디바이스 인증을 수행하는 경우 두 가지 기능을 활용할 수 있습니다.

1. 고급 보안 기능을 사용하면 보안 및 분석 목적으로 특정 디바이스에서의 사용자 활동을 모니터링할 수 있습니다. 사용자가 로그인하면 각 사용자와 디바이스를 인증하고 활동 로그에 디바이스 정보를 추가하는 옵션이 앱에 있습니다.
2. 디바이스 기억 기능은 신뢰할 수 있는 디바이스 인증 흐름도 지원합니다. 여기서 사용자는 앱의 보안 요구 사항에 맞는 기간 동안 MFA 없이 로그인하도록 선택할 수 있습니다. 사용자에게 MFA 코드를 제출하라는 메시지를 다시 표시하려는 경우 디바이스의 기억된 상태를 변경할 수 있습니다.

   기억된 디바이스는 MFA가 활성화된 사용자 풀에서만 MFA를 재정의할 수 있습니다.

사용자가 기억된 디바이스로 로그인하면 인증 흐름에서 추가 디바이스 인증을 수행해야 합니다. 자세한 내용은 디바이스로 로그인 섹션을 참조하세요.

디바이스 추적 내 사용자 풀의 로그인 경험 탭에서 사용자 풀이 디바이스를 기억하도록 구성합니다. Amazon Cognito 콘솔을 통해 기억된 디바이스 기능을 설정할 때, [항상(Always)], [사용자 옵트인(User Opt-In)] 및 [아니요(No)]라는 세 가지 옵션이 제공됩니다.

기억하지 않음

사용자 풀이 사용자가 로그인할 때 디바이스를 기억하라는 메시지를 표시하지 않습니다.

항상 기억

앱이 사용자의 디바이스를 확인하면 사용자 풀이 항상 디바이스를 기억하고 향후 디바이스 로그인에 성공할 때 MFA 챌린지를 반환하지 않습니다.

사용자 옵트인

앱이 사용자의 디바이스를 확인하더라도 사용자 풀이 MFA 챌린지를 자동으로 억제하지 않습니다. 사용자에게 디바이스를 기억할지 여부를 선택하라는 메시지를 표시해야 합니다.

항상 기억 또는 사용자 옵트인을 선택하는 경우, Amazon Cognito는 사용자가 미확인 디바이스에서 로그인할 때마다 디바이스 식별자 키와 암호를 생성합니다. 디바이스 키는 사용자가 디바이스 인증을 수행할 때 앱이 사용자 풀에 보내는 초기 식별자입니다.

자동으로 기억되든 옵트인되든 관계없이 확인된 각 사용자 디바이스에서 디바이스 식별자 키와 비밀을 사용하여 사용자가 로그인할 때마다 디바이스를 인증할 수 있습니다.

또한 CreateUserPool 또는 UpdateUserPool API 요청에서 사용자 풀의 기억된 디바이스 설정을 구성할 수 있습니다. 자세한 내용은 DeviceConfiguration 속성을 참조하세요.

Amazon Cognito 사용자 풀 API에는 기억된 디바이스에 대한 추가 작업이 있습니다.

1. ListDevices 및 AdminListDevices는 사용자에 대한 디바이스 키 및 메타데이터 목록을 반환합니다.
2. GetDevice 및 AdminGetDevice는 단일 디바이스에 대한 디바이스 키와 메타데이터를 반환합니다.
3. UpdateDeviceStatus 및 AdminUpdateDeviceStatus는 사용자의 디바이스를 기억됨 또는 기억되지 않음으로 설정합니다.
4. ForgetDevice 및 AdminForgetDevice는 프로필에서 사용자의 확인된 디바이스를 제거합니다.

이름이 `Admin`으로 시작하는 API 작업은 서버측 앱에서 사용하기 위한 것이며 IAM 보안 인증 정보로 승인되어야 합니다. 자세한 내용은 Amazon Cognito 사용자 풀 API 및 사용자 풀 엔드포인트 사용 섹션을 참조하세요.

디바이스 키 가져오기

사용자가 사용자 풀 API를 사용하여 로그인하고 인증 파라미터에 `DEVICE_KEY`로 디바이스 키를 포함하지 않을 때마다 Amazon Cognito는 응답으로 새 디바이스 키를 반환합니다. 퍼블릭 클라이언트측 앱에서는 향후 요청에 포함할 수 있도록 앱 스토리지에 디바이스 키를 저장합니다. 기밀 서버측 앱에서는 사용자의 디바이스 키로 브라우저 쿠키 또는 다른 클라이언트측 토큰을 설정합니다.

사용자가 신뢰할 수 있는 디바이스로 로그인하려면 먼저 앱에서 디바이스 키를 확인하고 추가 정보를 제공해야 합니다. Amazon Cognito에 ConfirmDevice 요청을 생성하여 디바이스 키, 친숙한 이름, 암호 확인 도구 및 솔트를 사용해 사용자의 디바이스를 확인합니다. 사용자 풀에 옵트인 디바이스 인증을 구성한 경우, Amazon Cognito는 `ConfirmDevice` 요청에 응답하여 사용자가 현재 디바이스를 기억할지 여부를 선택해야 한다는 메시지를 표시합니다. UpdateDeviceStatus 요청에서 사용자의 선택 사항으로 응답합니다.

사용자의 디바이스를 확인하지만 기억하도록 설정하지 않은 경우 Amazon Cognito는 연결을 저장하지만 디바이스 키를 제공할 때 디바이스 이외 로그인을 진행합니다. 디바이스는 사용자 보안 및 문제 해결에 유용한 로그를 생성할 수 있습니다. 확인되었지만 기억되지 않은 디바이스는 로그인 기능을 활용

하지 않지만 보안 모니터링 로그 기능은 활용합니다. 앱 클라이언트의 고급 보안 기능을 활성화하고 요청에 디바이스 풋프린트를 인코딩하면 Amazon Cognito가 사용자 이벤트를 확인된 디바이스와 연결합니다.

새 디바이스 키를 얻으려면

1. InitiateAuth API 요청으로 사용자의 로그인 세션을 시작합니다.
2. 사용자의 로그인 세션이 완료되었음을 표시하는 JSON 웹 토큰(JWT)을 받을 때까지 RespondToAuthChallenge를 사용하여 모든 인증 챌린지에 응답합니다.
3. 앱에서 Amazon Cognito가 `RespondToAuthChallenge` 또는 `InitiateAuth` 응답의 `NewDeviceMetadata`에 반환하는 값(`DeviceGroupKey` 및 `DeviceKey`)을 기록합니다.
4. 사용자를 위한 새 SRP 비밀(솔트 및 암호 검증자)을 생성합니다. 이 기능은 SRP 라이브러리를 제공하는 SDK에서 사용할 수 있습니다.
5. 사용자에게 디바이스 이름을 입력하라는 메시지를 표시하거나 사용자의 디바이스 특성에서 이름을 생성합니다.
6. ConfirmDevice API 요청에서 사용자의 액세스 토큰, 디바이스 키, 디바이스 이름 및 SRP 비밀을 제공합니다. 사용자 풀이 디바이스를 항상 기억하도록 설정된 경우 사용자 등록이 완료됩니다.
7. Amazon Cognito가 `"UserConfirmationNecessary": true`로 `ConfirmDevice`에 응답한 경우, 사용자에게 디바이스를 기억할지 선택하라는 메시지를 표시합니다. 사용자가 디바이스를 기억하겠다고 확인하면 사용자의 액세스 토큰, 디바이스 키, `"DeviceRememberedStatus": "remembered"` 등을 사용하여 UpdateDeviceStatus API 요청을 생성합니다.
8. Amazon Cognito에 디바이스를 기억하도록 지시한 경우, 다음에 로그인할 때 MFA 챌린지 대신 `DEVICE_SRP_AUTH` 챌린지가 표시됩니다.

디바이스로 로그인

사용자의 디바이스를 기억하도록 구성한 후에는 사용자가 동일한 디바이스 키로 로그인할 때 Amazon Cognito에서 더 이상 MFA 코드 제출을 요구하지 않습니다. 디바이스 인증은 MFA 인증 챌린지를 디바이스 인증 챌린지로 대체할 뿐입니다. 디바이스 인증만으로는 사용자가 로그인될 수 없습니다. 사용자는 먼저 암호 또는 사용자 지정 챌린지로 인증을 완료해야 합니다. 기억된 디바이스에서 사용자를 인증하는 프로세스는 다음과 같습니다.

사용자 지정 인증 챌린지 Lambda 트리거를 사용하는 흐름에서 디바이스 인증을 수행하려면 InitiateAuth API 요청에 `DEVICE_KEY` 파라미터를 전달합니다. 사용자가 모든 챌린지에 성공하고 `CUSTOM_CHALLENGE` 챌린지가 `true`인 `issueTokens` 값을 반환하면 Amazon Cognito는 하나의 최종 `DEVICE_SRP_AUTH` 챌린지를 반환합니다.

디바이스로 로그인하려면

1. 클라이언트 스토리지에서 사용자의 디바이스 키를 검색합니다.
2. InitiateAuth API 요청으로 사용자의 로그인 세션을 시작합니다. USER_SRP_AUTH, REFRESH_TOKEN_AUTH, USER_PASSWORD_AUTH 또는 CUSTOM_AUTH의 AuthFlow를 선택합니다. AuthParameters에서 사용자의 디바이스 키를 DEVICE_KEY 파라미터에 추가하고 선택한 로그인 흐름에 필요한 기타 파라미터를 포함합니다.
   a. 인증 챌린지에 대한 PASSWORD_VERIFIER 응답의 파라미터에 DEVICE_KEY를 전달할 수도 있습니다.
3. 응답으로 DEVICE_SRP_AUTH 챌린지를 받을 때까지 챌린지 응답을 완료합니다.
4. RespondToAuthChallenge API 요청에서 DEVICE_SRP_AUTH의 ChallengeName과 USERNAME, DEVICE_KEY 및 SRP_A에 대한 파라미터를 전송합니다.
5. Amazon Cognito가 DEVICE_PASSWORD_VERIFIER 챌린지로 응답합니다. 이 챌린지 응답에는 SECRET_BLOCK 및 SRP_B 값이 포함됩니다.
6. SRP 라이브러리를 사용하여 PASSWORD_CLAIM_SIGNATURE, PASSWORD_CLAIM_SECRET_BLOCK, TIMESTAMP, USERNAME 및 DEVICE_KEY 파라미터를 생성하고 제출합니다. 추가 RespondToAuthChallenge 요청으로 제출해야 합니다.
7. 사용자의 JWT를 받을 때까지 추가 챌린지를 완료합니다.

다음 유사 코드는 DEVICE_PASSWORD_VERIFIER 챌린지 응답의 값을 계산하는 방법을 보여줍니다.

```
PASSWORD_CLAIM_SECRET_BLOCK = SECRET_BLOCK
TIMESTAMP = Tue Sep 25 00:09:40 UTC 2018
PASSWORD_CLAIM_SIGNATURE = Base64(SHA256_HMAC(K_USER, DeviceGroupKey + DeviceKey +
 PASSWORD_CLAIM_SECRET_BLOCK + TIMESTAMP))
K_USER = SHA256_HASH(S_USER)
S_USER = (SRP_B - k * gx)(a + ux)
x = SHA256_HASH(salt + FULL_PASSWORD)
u = SHA256_HASH(SRP_A + SRP_B)
k = SHA256_HASH(N + g)
```

디바이스 보기, 업데이트 및 지우기

Amazon Cognito API를 사용하여 앱에서 다음 기능을 구현할 수 있습니다.

1. 사용자의 현재 디바이스에 대한 정보를 표시합니다.
2. 모든 사용자 디바이스 목록을 표시합니다.

3. 디바이스를 지웁니다.

4. 디바이스 기억 상태를 업데이트합니다.

다음 설명의 API 요청을 승인하는 액세스 토큰에는 `aws.cognito.signin.user.admin` 범위가 포함되어야 합니다. Amazon Cognito는 Amazon Cognito 사용자 풀 API로 생성하는 모든 액세스 토큰에 이 범위에 대한 클레임을 추가합니다. 서드 파티 IdP는 Amazon Cognito에 인증하는 사용자의 디바이스와 MFA를 별도로 관리해야 합니다. 호스팅 UI에서는 `aws.cognito.signin.user.admin` 범위를 요청할 수 있지만 호스팅 UI는 고급 보안 사용자 로그에 디바이스 정보를 자동으로 추가하며 디바이스 기억 기능을 제공하지 않습니다.

디바이스에 대한 정보 표시

사용자 디바이스에 대한 정보를 쿼리하여 디바이스가 현재 사용 중인지 확인할 수 있습니다. 예를 들어 90일 동안 로그인하지 않은 기억된 디바이스를 비활성화할 수 있습니다.

- 퍼블릭 클라이언트 앱에 사용자의 디바이스 정보를 표시하려면 GetDevice API 요청에서 사용자의 액세스 키와 디바이스 키를 제출합니다.
- 기밀 클라이언트 앱에 사용자의 디바이스 정보를 표시하려면 AWS 보안 인증 정보로 AdminGetDevice API 요청에 서명하고 사용자의 사용자 이름, 디바이스 키 및 사용자 풀을 제출합니다.

모든 사용자 디바이스 목록 표시

모든 사용자 디바이스 및 속성 목록을 표시할 수 있습니다. 예를 들어 현재 디바이스가 기억된 디바이스와 일치하는지 확인할 수 있습니다.

- 퍼블릭 클라이언트 앱에서는 ListDevices API 요청으로 사용자의 액세스 토큰을 제출합니다.
- 기밀 클라이언트 앱에서는 AWS 보안 인증 정보로 AdminListDevices API 요청에 서명하고 사용자의 사용자 이름 및 사용자 풀을 제출합니다.

디바이스 지우기

사용자의 디바이스 키를 삭제할 수 있습니다. 사용자가 더 이상 디바이스를 사용하지 않는 것으로 확인되거나 비정상적인 활동을 탐지하여 사용자에게 MFA를 다시 완료하도록 요구하려는 경우 이 방법을 사용하는 것이 좋습니다. 나중에 디바이스를 다시 등록하려면 새 디바이스 키를 생성하여 저장해야 합니다.

- 퍼블릭 클라이언트 앱에서는 ForgetDevice API 요청으로 사용자의 디바이스 키와 액세스 토큰을 제출합니다.
- 기밀 클라이언트 앱에서는 AdminForgetDevice API 요청으로 사용자의 디바이스 키와 액세스 토큰을 제출합니다.

Amazon Cognito 사용자 풀 API 및 사용자 풀 엔드포인트 사용

사용자 풀에 가입하고, 로그인하고, 사용자 풀에서 사용자를 관리할 때는 두 가지 옵션을 사용할 수 있습니다.

1. 사용자 풀 엔드포인트에는 호스팅 UI와 페더레이션 엔드포인트가 포함됩니다. 이 엔드포인트는 사용자 풀에 사용할 도메인을 선택할 때 Amazon Cognito가 활성화하는 퍼블릭 웹 페이지 패키지를 구성합니다. 가입, 로그인, 암호 관리 및 다중 인증(MFA) 페이지를 비롯한 Amazon Cognito 사용자 풀의 인증 및 권한 부여 기능을 빠르게 시작하려면, 호스팅된 UI의 내장 사용자 인터페이스를 사용하세요. 다른 사용자 풀 엔드포인트는 서드 파티 ID 제공업체(idP)를 사용한 인증을 용이하게 합니다. 이들이 수행하는 서비스에는 다음이 포함됩니다.

   a. `saml2/idpresponse` 및 `oauth2/idpresponse` 등 IdP의 인증된 클레임을 위한 서비스 공급자 콜백 엔드포인트. Amazon Cognito가 앱과 IdP 사이의 중간 서비스 공급자(SP)인 경우 콜백 엔드포인트는 서비스를 나타냅니다.

   b. `oauth2/userInfo` 및 `jwks.json` 등 환경에 대한 정보를 제공하는 엔드포인트. 앱은 AWS SDK 및 OAuth 2.0 라이브러리를 사용하여 토큰을 확인하거나 사용자 프로필 데이터를 검색할 때 이러한 엔드포인트를 사용합니다.

2. Amazon Cognito 사용자 풀 API는 웹 또는 모바일 앱이 자체 사용자 지정 프런트엔드에서 로그인 정보를 수집한 후 사용자를 인증하는 앱용 도구 집합입니다. 사용자 풀 API 인증은 다음 JSON 웹 토큰을 생성합니다.

   a. 사용자의 검증 가능한 속성 클레임이 포함된 자격 증명 토큰.

   b. AWS 서비스 엔드포인트에 대한 토큰 인증된 API 요청을 생성할 권한을 사용자에게 부여하는 액세스 토큰.

> **Note**
>
> 기본적으로 사용자 풀 API 인증의 액세스 토큰에는 `aws.cognito.signin.user.admin` 범위만 포함되어 있습니다. 추가 범위가 포함된 액세스 토큰을 생성하려면(예: 서드 파티 API에 대한 요청 승인을 위해) 사용자 풀 엔드포인트를 통한 인증 중에 범위를 요청하거나 사전 토큰 생성 Lambda 트리거에서 사용자 지

정 범위를 추가합니다. 액세스 토큰 사용자 지정 시에는 AWS 청구서에 비용이 추가됩니다.

일반적으로 사용자 풀 엔드포인트를 통해 로그인하는 페더레이션 사용자를 프로필이 사용자 풀에 로컬인 사용자와 연결할 수 있습니다. 로컬 사용자는 외부 IdP를 통한 페더레이션 없이 사용자 풀 디렉터리에만 존재합니다. AdminLinkProviderForUser API 요청에서 로컬 사용자에 페더레이션 자격 증명을 연결하면 해당 사용자가 사용자 풀 API로 로그인할 수 있습니다. 자세한 내용은 페더레이션 사용자를 기존 사용자 프로필에 연결 섹션을 참조하세요.

Amazon Cognito 사용자 풀 API는 두 가지 용도로 사용합니다. 이 API는 Amazon Cognito 사용자 풀 리소스를 생성하고 구성합니다. 예를 들어 사용자 풀을 만들고, AWS Lambda 트리거를 추가하고, 호스팅된 UI 도메인을 구성할 수 있습니다. 또한 사용자 풀 API는 로컬 및 연결된 사용자의 가입, 로그인 및 기타 사용자 작업을 수행합니다.

Amazon Cognito 사용자 풀 API를 사용한 예제 시나리오

1. 앱에서 생성한 “Create an account(계정 만들기)” 버튼을 사용자가 선택합니다. 사용자가 이메일 주소와 암호를 입력합니다.
2. 앱에서 SignUp API 요청을 보내고 사용자 풀에 새 사용자를 생성합니다.
3. 앱에서 사용자에게 이메일 확인 코드를 입력하라는 메시지를 표시합니다. 사용자가 이메일 메시지로 받은 코드를 입력합니다.
4. 앱은 ConfirmSignUp API 요청과 사용자의 확인 코드를 전송합니다.
5. 앱이 사용자 이름과 암호를 입력하라는 메시지를 표시하고, 사용자가 자신의 정보를 입력합니다.
6. 앱은 InitiateAuth API 요청을 보내고 ID 토큰, 액세스 토큰 및 새로 고침 토큰을 저장합니다. 앱은 OIDC 라이브러리를 호출하여 사용자의 토큰을 관리하고 해당 사용자의 영구 세션을 유지합니다.

Amazon Cognito 사용자 풀 API에서는 IdP를 통해 페더레이션하는 사용자를 로그인할 수 없습니다. 사용자 풀 엔드포인트를 통해 이러한 사용자를 인증해야 합니다. 호스팅 UI를 포함하는 사용자 풀 엔드포인트에 대한 자세한 내용은 사용자 풀 페더레이션 엔드포인트 및 호스팅 UI 참조 섹션을 참조하세요. 페더레이션 사용자는 호스팅 UI에서 시작하여 IdP를 선택하거나, 호스팅 UI를 건너뛰고 사용자를 IdP로 직접 보내 로그인하게 할 수 있습니다. 권한 부여 엔드포인트에 대한 API 요청에 IdP 파라미터가 포함되면, Amazon Cognito는 사용자를 자동으로 IdP 로그인 페이지로 리디렉션합니다.

사용자 풀 엔드포인트 예제 시나리오

1. 앱에서 생성한 “Create an account(계정 만들기)” 버튼을 사용자가 선택합니다.

2. 개발자 보안 인증 정보를 등록한 소셜 ID 공급자 목록을 사용자에게 제시합니다. 사용자가 Apple을 선택합니다.
3. 앱에서 공급자 이름 `SignInWithApple`로 권한 부여 엔드포인트에 대한 요청을 시작합니다.
4. 사용자의 브라우저에서 Apple OAuth 인증 페이지가 열립니다. 사용자가 Amazon Cognito가 자신의 프로필 정보를 읽도록 허용합니다.
5. Amazon Cognito가 Apple 액세스 토큰을 확인하고 사용자의 Apple 프로필을 쿼리합니다.
6. 사용자가 앱에 Amazon Cognito 인증 코드를 제시합니다.
7. 앱이 인증 코드를 Token 엔드포인트와 교환하고 ID 토큰, 액세스 토큰 및 새로 고침 토큰을 저장합니다. 앱은 OIDC 라이브러리를 호출하여 사용자의 토큰을 관리하고 해당 사용자의 영구 세션을 유지합니다.

사용자 풀 API와 사용자 풀 엔드포인트는 이 안내서에서 설명하는 다양한 시나리오를 지원합니다. 다음 섹션에서는 사용자 풀 API를 가입, 로그인, 리소스 관리 요구 사항을 지원하는 클래스로 구분하는 방법을 살펴봅니다.

Amazon Cognito 사용자 풀 인증 및 미인증 API 작업

리소스 관리 인터페이스이자 사용자 대면 인증 및 권한 부여 인터페이스인 Amazon Cognito 사용자 풀 API는 운영에 동반되는 권한 부여 모델을 결합합니다. API 작업에 따라 IAM 보안 인증 정보, 액세스 토큰, 세션 토큰, 클라이언트 암호 또는 이들의 조합을 통해 인증을 제공해야 할 수 있습니다. 대부분의 사용자 인증 및 권한 부여 작업의 경우, 요청의 인증 버전과 미인증 버전을 선택할 수 있습니다. 모바일 앱처럼 사용자에게 배포하는 앱의 경우 미인증 작업이 모범 사례입니다. 코드에 암호를 포함하지 않아도 됩니다.

IAM 인증 관리 작업 및 IAM 인증 사용자 작업에 대한 IAM 정책에서만 권한을 할당할 수 있습니다.

IAM 인증 관리 작업

IAM 인증 관리 작업은 AWS Management Console에서처럼 사용자 풀 및 앱 클라이언트 구성을 수정하고 확인합니다.

예를 들어 UpdateUserPool API 요청에서 사용자 풀을 수정하려면 AWS 보안 인증 정보와 IAM 권한을 제시하여 리소스를 업데이트해야 합니다.

AWS Command Line Interface(AWS CLI) 또는 AWS SDK에서 이러한 요청을 승인하려면, 요청에 IAM 자격 증명을 추가하는 환경 변수 또는 클라이언트 구성으로 환경을 구성합니다. 자세한 내용은 AWS 일반 참조의 AWS 보안 인증 정보를 사용하여 AWS에 액세스를 참조하세요. Amazon Cognito 사용자 풀 API의 서비스 엔드포인트로 바로 요청을 보낼 수도 있습니다. 요청의 헤더에 삽입한 AWS 보안 인

증 정보를 사용하여 이러한 요청을 승인하거나 서명해야 합니다. 자세한 내용은 AWS API 요청 서명을 참조하세요.

IAM 인증 관리 작업

AddCustomAttributes

CreateGroup

CreateIdentityProvider

CreateResourceServer

CreateUserImportJob

CreateUserPool

CreateUserPoolClient

CreateUserPoolDomain

DeleteGroup

DeleteIdentityProvider

DeleteResourceServer

DeleteUserPool

DeleteUserPoolClient

DeleteUserPoolDomain

DescribeIdentityProvider

DescribeResourceServer

DescribeRiskConfiguration

DescribeUserImportJob

DescribeUserPool

IAM 인증 관리 작업

DescribeUserPoolClient

DescribeUserPoolDomain

GetCSVHeader

GetGroup

GetIdentityProviderByIdentifier

GetSigningCertificate

GetUICustomization

GetUserPoolMfaConfig

ListGroups

ListIdentityProviders

ListResourceServers

ListTagsForResource

ListUserImportJobs

ListUserPoolClients

ListUserPools

ListUsers

ListUsersInGroup

SetRiskConfiguration

SetUICustomization

SetUserPoolMfaConfig

StartUserImportJob

IAM 인증 관리 작업

`StopUserImportJob`

`TagResource`

`UntagResource`

`UpdateGroup`

`UpdateIdentityProvider`

`UpdateResourceServer`

`UpdateUserPool`

`UpdateUserPoolClient`

`UpdateUserPoolDomain`

IAM 인증 사용자 작업

IAM 인증 사용자 작업은 사용자의 가입, 로그인, 보안 인증 정보 관리, 수정 및 보기를 수행합니다.

예를 들어 웹 프런트 엔드를 지원하는 서버 측 애플리케이션 계층을 보유할 수 있습니다. 서버 측 앱은 Amazon Cognito 리소스에 대한 액세스 권한이 있는 신뢰하는 OAuth 기밀 클라이언트입니다. 앱에서 사용자를 등록하려면 서버에서 AdminCreateUser API 요청에 AWS 보안 인증 정보를 포함하면 됩니다. OAuth 클라이언트 유형에 대한 자세한 내용은 OAuth 2.0 인증 프레임워크의 클라이언트 유형을 참조하세요.

AWS CLI 또는 AWS SDK에서 이러한 요청을 승인하려면, 요청에 IAM 자격 증명을 추가하는 환경 변수 또는 클라이언트 구성으로 서버 측 앱 환경을 구성합니다. 자세한 내용은 AWS 일반 참조의 AWS 보안 인증 정보를 사용하여 AWS에 액세스를 참조하세요. Amazon Cognito 사용자 풀 API의 서비스 엔드포인트로 바로 요청을 보낼 수도 있습니다. 요청의 헤더에 삽입한 AWS 보안 인증 정보를 사용하여 이러한 요청을 승인하거나 서명해야 합니다. 자세한 내용은 AWS API 요청 서명을 참조하세요.

앱 클라이언트에 클라이언트 암호가 있는 경우 `AuthParameters`에 IAM 보안 인증 정보와, 작업에 따라 `SecretHash` 파라미터 또는 `SECRET_HASH` 값을 입력해야 합니다. 자세한 내용은 암호 해시 값 컴퓨팅 섹션을 참조하세요.

IAM 인증 사용자 작업

AdminAddUserToGroup

AdminConfirmSignUp

AdminCreateUser

AdminDeleteUser

AdminDeleteUserAttributes

AdminDisableProviderForUser

AdminDisableUser

AdminEnableUser

AdminForgetDevice

AdminGetDevice

AdminGetUser

AdminInitiateAuth

AdminLinkProviderForUser

AdminListDevices

AdminListGroupsForUser

AdminListUserAuthEvents

AdminRemoveUserFromGroup

AdminResetUserPassword

AdminRespondToAuthChallenge

AdminSetUserMFAPreference

IAM 인증 사용자 작업

AdminSetUserPassword

AdminSetUserSettings

AdminUpdateAuthEventFeedback

AdminUpdateDeviceStatus

AdminUpdateUserAttributes

AdminUserGlobalSignOut

미인증 사용자 작업

미인증 사용자 작업은 사용자의 가입, 로그인 및 암호 재설정 시작 작업을 수행합니다. 인터넷에 있는 모든 사용자가 앱에 가입하고 로그인할 수 있게 하려면 미인증, 즉 공개 API 작업을 사용해야 합니다.

예를 들어 앱에 사용자를 등록하려면 암호에 대한 어떠한 권한 액세스도 제공하지 않는 OAuth 공개 클라이언트를 배포하면 됩니다. 미인증 API 작업 SignUp을 이용해 이 사용자를 등록할 수 있습니다.

AWS SDK로 개발한 공개 클라이언트에서 이러한 요청을 보내려는 경우에는 보안 인증 정보를 구성하지 않아도 됩니다. 추가 인증 없이 Amazon Cognito 사용자 풀 API의 서비스 엔드포인트로 바로 요청을 보낼 수도 있습니다.

앱 클라이언트에 클라이언트 암호가 있는 경우 작업에 따라 AuthParameters에 SecretHash 파라미터 또는 SECRET_HASH 값을 입력해야 합니다. 자세한 내용은 암호 해시 값 컴퓨팅 섹션을 참조하세요.

미인증 사용자 작업

SignUp

ConfirmSignUp

ResendConfirmationCode

ForgotPassword

미인증 사용자 작업

`ConfirmForgotPassword`

`InitiateAuth`

토큰 권한 부여 사용자 작업

토큰으로 권한이 부여된 사용자 작업은 사용자가 로그인하거나 로그인 프로세스를 시작한 후 사용자 로그아웃, 보안 인증 정보 관리, 수정 및 확인 작업을 수행합니다. 앱에서 암호를 배포하지 않고 사용자 자신의 보안 인증 정보로 요청을 승인하고 싶다면, 토큰으로 권한이 부여된 API 작업을 사용해야 합니다. 사용자가 로그인을 완료한 경우 액세스 토큰을 사용하여 토큰으로 권한이 부여된 API 요청을 승인해야 합니다. 사용자가 로그인 프로세스를 진행 중이라면, Amazon Cognito가 이전 요청에 대한 응답으로 반환한 세션 토큰을 사용하여 토큰으로 권한이 부여된 API 요청을 승인해야 합니다.

예를 들어 공개 클라이언트에서는 사용자 자신의 프로필에 대한 쓰기 권한만 제한하는 방식으로 사용자 프로필을 업데이트해야 할 수 있습니다. 이 업데이트를 수행하기 위해 클라이언트는 UpdateUserAttributes API 요청에 사용자의 액세스 토큰을 포함할 수 있습니다.

AWS SDK로 개발한 공개 클라이언트에서 이러한 요청을 보내려는 경우에는 보안 인증 정보를 구성하지 않아도 됩니다. 요청에 `AccessToken` 또는 `Session` 파라미터를 포함합니다. Amazon Cognito 사용자 풀 API의 서비스 엔드포인트로 바로 요청을 보낼 수도 있습니다. 서비스 엔드포인트에 대한 요청을 승인하려면 요청의 POST 본문에 액세스 또는 세션 토큰을 포함합니다.

토큰으로 권한이 부여된 작업을 위한 API 요청에 서명하려면 액세스 토큰을 `Bearer` *`<Base64-encoded access token>`* 형식의 `Authorization` 헤더로 요청에 포함하세요.

토큰 권한 부여 사용자 작업	AccessToken	세션
`RespondToAuthChallenge`		✓
`ChangePassword`	✓	
`GetUser`	✓	
`UpdateUserAttributes`	✓	

토큰 권한 부여 사용자 작업	AccessToken	세션
DeleteUserAttributes	✓	
DeleteUser	✓	
ConfirmDevice	✓	
ForgetDevice	✓	
GetDevice	✓	
ListDevices	✓	
UpdateDeviceStatus	✓	
GetUserAttributeVerificationCode	✓	
VerifyUserAttribute	✓	
SetUserSettings	✓	
SetUserMFAPreference	✓	
GlobalSignOut	✓	
AssociateSoftwareToken	✓	✓
UpdateAuthEventFeedback		✓
VerifySoftwareToken	✓	✓

토큰 권한 부여 사용자 작업	AccessToken	세션
RevokeToken [1]		

[1] RevokeToken은 새로 고침 토큰을 파라미터로 사용합니다. 새로 고침 토큰은 인증 토큰 및 대상 리소스 역할을 합니다.

사용자 풀 구성 업데이트

에서 Amazon Cognito 사용자 풀의 설정을 변경하려면 사용자 풀 설정의 기능 기반 탭을 탐색하고 이 가이드의 다른 영역에 설명된 대로 필드를 업데이트하십시오. AWS Management Console사용자 풀이 생성된 후에는 일부 설정을 변경할 수 없습니다. 다음 설정을 변경하려면 새 사용자 풀 또는 앱 클라이언트를 생성해야 합니다.

사용자 풀 이름

API 파라미터 이름: PoolName

사용자 풀에 할당한 표시 이름입니다. 사용자 풀의 이름을 변경하려면 새 사용자 풀을 생성합니다.

Amazon Cognito 사용자 풀 로그인 옵션

API 파라미터 이름: AliasAttributes및 UsernameAttributes

사용자가 로그인할 때 사용자 이름으로 전달할 수 있는 속성입니다. 사용자 풀을 생성할 때 사용자 이름, 이메일 주소, 전화번호 또는 기본 사용자 이름으로 로그인을 허용하도록 선택할 수 있습니다. 사용자 풀 로그인 옵션을 변경하려면 새 사용자 풀을 생성합니다.

사용자 이름의 대/소문자 구분(Make user name case sensitive)

API 파라미터 이름: UsernameConfiguration

대소문자를 제외하고 다른 사용자 이름과 일치하는 사용자 이름을 생성하면 Amazon Cognito가 동일한 사용자로도, 고유한 사용자로도 취급할 수 있습니다. 자세한 정보는 사용자 풀 대/소문자 구분을 참조하세요. 대소문자 구분을 변경하려면 새 사용자 풀을 생성합니다.

클라이언트 암호

API 파라미터 이름: GenerateSecret

앱 클라이언트를 만들 때 신뢰할 수 있는 소스만 사용자 풀에 요청하도록 클라이언트 암호를 생성할 수 있습니다. 자세한 정보는 사용자 풀 앱 클라이언트을 참조하세요. 클라이언트 암호를 변경하려면 동일한 사용자 풀에 새 앱 클라이언트를 생성합니다.

필수 속성

API 파라미터 이름: 스키마

사용자가 가입할 때 또는 사용자를 생성할 때 값을 제공해야 하는 속성입니다. 자세한 정보는 사용자 풀 속성을 참조하세요. 필수 속성을 변경하려면 새 사용자 풀을 생성합니다.

사용자 지정 속성

API 파라미터 이름: 스키마

사용자 지정 이름이 있는 속성입니다. 사용자의 사용자 지정 속성 값은 변경할 수 있지만, 사용자 풀에서 사용자 지정 속성을 삭제할 수는 없습니다. 자세한 정보는 사용자 풀 속성을 참조하세요. 최대 사용자 지정 속성 수에 도달하고 목록을 수정하려면 새 사용자 풀을 생성합니다.

SMS 컨피그레이션

사용자 풀에서 SMS 메시지를 활성화한 후에는 비활성화할 수 없습니다.

- 사용자 풀을 생성할 때 SMS 메시지를 구성하도록 선택한 경우 설정을 완료한 후에는 SMS를 비활성화할 수 없습니다.
- 생성한 사용자 풀에서 SMS 메시지를 활성화할 수 있지만 그 이후에는 SMS를 비활성화할 수 없습니다.
- Amazon Cognito는 사용자 계정 초대 및 복구, 속성 확인, 멀티 팩터 인증 (MFA) 에 SMS 메시지를 사용할 수 있습니다. SMS 메시지를 활성화한 후에는 언제든지 이러한 기능에 대해 SMS 메시지를 켜거나 끌 수 있습니다.
- SMS 메시지 구성에는 Amazon SNS로 메시지를 전송하기 위해 Amazon Cognito에 위임하는 IAM 역할이 포함됩니다. 할당된 역할은 언제든지 변경할 수 있습니다.

AWS SDK 또는 REST API로 사용자 풀 업데이트 AWS CDK

Amazon Cognito 콘솔에서는 사용자 풀 설정을 한 번에 하나의 파라미터씩 변경할 수 있습니다. 예를 들어, Lambda 트리거를 추가하려면 Add Lambda 트리거를 선택하고 함수와 트리거 유형을 선택합니다. Amazon Cognito 사용자 풀 API는 사용자 풀 및 앱 클라이언트의 업데이트 작업에서 사용자 풀에

대한 전체 파라미터 세트를 요구하는 방식으로 구성되어 있습니다. 하지만 콘솔은 다른 사용자 풀 설정과 함께 이 업데이트 작업을 투명하게 자동화합니다.

변경하려는 설정과 관련이 없는 업데이트에서 오류가 발생하는 경우가 종종 AWS 계정 있습니다. Amazon SES 자격 증명을 삭제하거나 IAM 권한을 변경한 경우를 예로 들 수 있습니다. AWS WAF현재 매개변수 중 하나가 더 이상 유효하지 않은 경우, 수정하기 전까지는 설정을 업데이트할 수 없습니다. 이러한 오류가 발생하면 오류 응답을 검사하고 해당 오류에 언급된 설정을 확인하십시오.

Amazon Cognito 사용자 풀 REST API 및 AWS SDK는 Amazon Cognito 리소스를 자동화하고 프로그래밍 방식으로 구성하기 위한 도구입니다. AWS Cloud Development Kit (AWS CDK) Amazon Cognito 콘솔과 마찬가지로 이러한 도구를 사용한 요청도 요청 본문에 전체 리소스 구성을 포함하여 설정을 업데이트해야 합니다. 상위 수준에서는 다음 프로세스를 수행해야 합니다.

1. 기존 리소스의 구성을 설명하는 작업의 출력을 캡처하십시오.
2. 설정 변경에 따라 출력을 수정하십시오.
3. 리소스를 업데이트하는 작업을 통해 수정된 구성을 전송하십시오.

다음 절차는 UpdateUserPoolAPI 작업을 통해 구성을 업데이트합니다. 입력 필드가 다른 동일한 접근 방식이 적용됩니다 UpdateUserPoolClient.

> **Important**
>
> 기존 파라미터에 대한 값을 입력하지 않으면 Amazon Cognito에 이 값이 기본값으로 설정됩니다. 예를 들어 기존 `LambdaConfig`가 있고 `LambdaConfig`를 비어 있는 상태로 `UpdateUserPool`을 제출하면 사용자 풀 트리거에 대한 모든 Lambda 함수 할당이 삭제됩니다. 사용자 풀 구성 변경을 자동화하려는 경우 적절하게 계획합니다.

1. 를 사용하여 사용자 풀의 기존 상태를 캡처하십시오 DescribeUserPool.
2. `DescribeUserPool`의 출력 형식을 `UpdateUserPool`의 요청 파라미터와 일치하도록 지정합니다. 출력 JSON에서 다음 최상위 필드 및 해당 하위 객체를 제거합니다.

   - `Arn`
   - `CreationDate`
   - `CustomDomain`
     - UpdateUserPoolDomainAPI 작업으로 이 필드를 업데이트하십시오.
   - `Domain`

    - UpdateUserPoolDomainAPI 작업으로 이 필드를 업데이트하십시오.
  - `EmailConfigurationFailure`
  - `EstimatedNumberOfUsers`
  - `Id`
  - `LastModifiedDate`
  - `Name`
  - `SchemaAttributes`
  - `SmsConfigurationFailure`
  - `Status`
3. 결과로 나타나는 JSON이 `UpdateUserPool`의 요청 파라미터와 일치하는지 확인합니다.
4. 결과로 나타나는 JSON에서 변경하려는 파라미터를 수정합니다.
5. 수정된 JSON을 요청 입력으로 하여 `UpdateUserPool` API 요청을 제출합니다.

수정된 출력은 AWS CLI에서 `update-user-pool`의 `--cli-input-json` 파라미터에도 사용할 수 있습니다.

또는 다음 AWS CLI 명령을 실행하여 허용된 입력 필드에 빈 값이 있는 JSON을 생성합니다. `update-user-pool` 그런 다음 이 필드를 사용자 풀의 기존 값으로 채울 수 있습니다.

```
aws cognito-idp update-user-pool --generate-cli-skeleton --output json
```

다음 명령을 실행하여 앱 클라이언트에 동일한 JSON 객체를 생성합니다.

```
aws cognito-idp update-user-pool-client --generate-cli-skeleton --output json
```

Amazon Cognito 호스팅 UI 및 페더레이션 엔드포인트 설정 및 사용

도메인이 있는 Amazon Cognito 사용자 풀은 OAuth-2.0 준수 권한 부여 서버이며 인증을 위한 ready-to-use 호스팅된 사용자 인터페이스 (UI) 입니다. 권한 부여 서버는 인증 요청을 라우팅하고, JSON 웹 토큰(JWT)을 발급 및 관리하고, 사용자 속성 정보를 제공합니다. 호스팅 UI는 사용자 풀의 기본적인 가입, 로그인, 다중 인증 및 암호 재설정 활동을 위한 웹 인터페이스 모음입니다. 또한 앱에 연결하는 타사 ID 공급자 (IdPs) 와의 인증을 위한 중앙 허브이기도 합니다. 사용자를 인증하고 권한을 부여하려는 경우 앱에서 호스팅 UI 및 권한 부여 엔드포인트를 호출할 수 있습니다. 자체 로고 및 CSS 사용자 지정을

통해 호스팅 UI 사용자 경험을 브랜드에 맞게 만들 수 있습니다. 호스팅 UI 및 권한 부여 서버의 구성 요소에 대한 자세한 내용은 사용자 풀 페더레이션 엔드포인트 및 호스팅 UI 참조 섹션을 참조하세요.

> **Note**
>
> Amazon Cognito 호스팅 UI는 사용자 지정 인증 문제 Lambda 트리거를 통한 사용자 지정 인증을 지원하지 않습니다.

주제

- 를 사용하여 호스팅된 UI를 설정합니다. AWS Amplify
- Amazon Cognito 콘솔을 사용하여 호스트된 UI 설정
- 로그인 페이지 보기
- Amazon Cognito 사용자 풀 호스팅 UI에 대해 알아야 할 사항
- 사용자 풀 도메인 구성
- 기본 제공 로그인 및 가입 웹 페이지 사용자 정의
- 호스팅 UI로 가입 및 로그인

를 사용하여 호스팅된 UI를 설정합니다. AWS Amplify

를 사용하여 웹 또는 모바일 앱에 인증을 AWS Amplify 추가하는 경우 프레임워크의 명령줄 인터페이스 (CLI) 및 라이브러리를 사용하여 호스팅된 UI를 설정할 수 있습니다. AWS Amplify 앱에 인증을 추가하려면 AWS Amplify CLI를 사용하여 프로젝트에 `Auth` 카테고리를 추가합니다. 그런 다음 클라이언트 코드에서 AWS Amplify 라이브러리를 사용하여 Amazon Cognito 사용자 풀로 사용자를 인증합니다.

미리 빌드된 호스팅 UI를 표시하거나 Facebook, Google, Amazon 또는 Apple과 같은 소셜 로그인 공급자로 리디렉션되는 OAuth 2.0 Endpoint를 통해 사용자를 연동할 수 있습니다. 사용자가 소셜 공급자를 통해 성공적으로 인증한 후 필요한 경우 AWS Amplify 가 사용자 풀에 새 사용자를 생성하고 사용자의 OIDC 토큰을 앱에 제공합니다.

다음 예는 앱에서 소셜 공급자를 통해 호스팅된 UI를 설정하는 AWS Amplify 데 사용하는 방법을 보여줍니다.

- AWS Amplify 인증 대상 JavaScript.
- AWS Amplify 스위프트에 대한 인증.
- AWS Amplify 플러터에 대한 인증.

- AWS Amplify 안드로이드용 인증.

Amazon Cognito 콘솔을 사용하여 호스트된 UI 설정

앱 클라이언트 생성

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 AWS 자격 증명을 입력합니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. [앱 통합(App integration)] 탭을 선택합니다.
5. [앱 클라이언트(App clients)]에서 [앱 클라이언트 생성(Create an app client)]을 선택합니다.
6. [앱 유형(App type)]에서 [퍼블릭 클라이언트(Public client)], [기밀 클라이언트(Confidential client)] 또는 [기타(Other)]를 선택합니다. [퍼블릭 클라이언트(Public client)]는 일반적으로 사용자 디바이스에서 작동하며, 인증되지 않은 API와 토큰 인증 API를 사용합니다. 기밀 클라이언트는 일반적으로 클라이언트 암호와 API 자격 증명으로 신뢰할 수 있는 중앙 서버의 앱에서 작동하며, 권한 부여 헤더와 자격 AWS Identity and Access Management 증명을 사용하여 요청에 서명합니다. 사용 사례가 [퍼블릭 클라이언트(Public client)] 또는 [기밀 클라이언트(Confidential client)]에 맞게 미리 구성된 앱 클라이언트 설정과 다른 경우 [기타(Other)]를 선택합니다.
7. [앱 클라이언트 이름(App client name)]을 입력합니다.
8. 앱 클라이언트에서 허용할 [인증 흐름(Authentication flows)]을 선택합니다.
9. Authentication flow session duration(인증 흐름 세션 기간)을 구성합니다. 세션 토큰이 만료되기 전에 사용자가 각 인증 챌린지를 완료해야 하는 시간입니다.
10. (선택 사항) 토큰 만료를 구성합니다.
    a. 앱 클라이언트의 [새로 고침 토큰 만료(Refresh token expiration)]를 지정합니다. 기본값은 30일입니다. 이 값을 1시간에서 10년 사이의 값으로 변경할 수 있습니다.
    b. 앱 클라이언트의 [액세스 토큰 만료(Access token expiration)]를 지정합니다. 기본값은 1시간입니다. 이 값을 5분에서 24시간 사이의 값으로 변경할 수 있습니다.
    c. 앱 클라이언트의 [ID 토큰 만료(ID token expiration)]를 지정합니다. 기본값은 1시간입니다. 이 값을 5분에서 24시간 사이의 값으로 변경할 수 있습니다.

> **Important**
>
> 호스트된 UI를 사용하여 1시간 미만의 토큰 수명을 구성할 경우 사용자는 현재 1시간으로 고정된 세션 쿠키 기간에 따라 토큰을 사용할 수 있습니다.

11. [클라이언트 암호 생성(Generate client secret)]을 선택하여 Amazon Cognito에서 클라이언트 암호를 자동으로 생성하도록 합니다. 클라이언트 암호는 일반적으로 기밀 클라이언트와 연결됩니다.
12. [토큰 철회 사용(Enable token revocation)]에서 이 앱 클라이언트에 토큰 철회를 사용할지 여부를 선택합니다. 사용할 경우 토큰의 크기가 늘어납니다. 자세한 내용은 토큰 철회를 참조하세요.
13. [사용자 존재를 나타내는 오류 메시지 방지(Prevent error messages that reveal user existence)]에서 이 앱 클라이언트에 사용자 존재를 나타내는 오류 메시지를 표시하지 않을지 여부를 선택합니다. Amazon Cognito는 존재하지 않는 사용자에 대한 로그인 요청에 사용자 이름이나 암호가 잘못되었다는 일반적인 메시지로 응답합니다.
14. (선택 사항) 이 앱 클라이언트에 대한 속성 읽기 및 쓰기 권한(Attribute read and write permissions)]을 구성합니다. 앱 클라이언트는 사용자 풀 속성 스키마의 제한된 하위 집합에 대한 읽기 및 쓰기 권한을 가질 수 있습니다.
15. 생성을 선택합니다.
16. [클라이언트 ID(Client id)]를 적어 둡니다. 클라이언트 ID는 가입 및 로그인 요청에서 앱 클라이언트를 식별합니다.

앱을 구성합니다.

1. [앱 통합(App integration)] 탭의 [앱 클라이언트(App clients)]에서 앱 클라이언트를 선택합니다. 현재 [호스트된 UI(Hosted UI)] 정보를 검토합니다.
2. 허용된 콜백 URL(Allowed callback URL(s))에서 콜백 URL 추가(Add a callback URL)를 수행합니다. 콜백 URL은 로그인 성공 후 사용자가 리디렉션되는 위치입니다.
3. [허용된 로그아웃 URL(Allowed sign-out URL(s))]에서 [로그아웃 URL 추가(Add a sign-out URL)]를 수행합니다. 로그아웃 URL은 로그아웃 후 사용자가 리디렉션되는 위치입니다.
4. [자격 증명 공급자(Identity providers)] 목록에서 나열된 옵션 중 하나 이상을 추가합니다.
5. [OAuth 2.0 권한 부여 유형(OAuth 2.0 grant types)]에서 인증 코드 권한 부여(Authorization code grant)를 선택하여 인증 코드를 반환합니다. 인증 코드는 사용자 풀 토큰으로 교환됩니다. 토큰은 최종 사용자에게 직접 노출되지 않기 때문에 침해될 가능성이 낮습니다. 하지만 사용자 풀 토큰에 대한 인증 코드를 교환하려면 백엔드에 사용자 지정 애플리케이션이 필요합니다. 보안상의 이유

로 모바일 앱에서는 PKCE(Proof Key for Code Exchange)와 함께 인증 코드 권한 부여 흐름을 사용하는 것이 좋습니다.

6. [OAuth 2.0 권한 부여 유형(OAuth 2.0 grant types)]에서 [암시적 권한 부여(Implicit grant)]를 선택하여 Amazon Cognito에서 사용자 풀 JSON 웹 토큰(JWT)이 반환되도록 합니다. 토큰에 대한 인증 코드를 교환할 수 있는 백엔드가 없을 때 이 흐름을 사용할 수 있습니다. 또한 토큰 디버깅에도 유용합니다.
7. [인증 코드(Authorization code)] 및 [암시적 코드(Implicit code)] 권한 부여를 둘 다 사용하도록 설정하고 필요에 따라 각 권한 부여를 사용할 수 있습니다. [인증 코드(Authorization code)] 또는 [암시적 코드(Implicit code)] 권한 부여를 선택하지 않았으며 앱 클라이언트에 클라이언트 암호가 있는 경우 [클라이언트 자격 증명(Client credentials)] 권한 부여를 사용하도록 설정할 수 있습니다. 앱이 사용자를 대신해서가 아니라 자체적으로 액세스 토큰을 요청해야 하는 경우에만 [클라이언트 자격 증명(Client credentials)]을 선택합니다.
8. 이 앱 클라이언트에 권한을 부여할 [OpenID Connect 범위(OpenID Connect scopes)]를 선택합니다.
9. 변경 사항 저장(Save changes)을 선택합니다.

도메인 구성

1. 사용자 풀의 [앱 통합(App integration)] 탭으로 이동합니다.
2. [도메인(Domain)] 옆의 [작업(Actions)]을 선택한 다음, [사용자 정의 도메인 생성(Create custom domain)] 또는 [Cognito 도메인 생성(Create Cognito domain)]을 선택합니다. 사용자 풀 도메인을 이미 구성한 경우 새 사용자 정의 도메인을 생성하기 전에 [Cognito 도메인 삭제(Delete Cognito domain)] 또는 [사용자 정의 도메인 삭제(Delete custom domain)]를 선택합니다.
3. [Cognito 도메인(Cognito domain)]에서 사용할 사용 가능한 도메인 접두사를 입력합니다. 사용자 정의 도메인을 설정하는 방법에 대한 자세한 내용은 호스트된 UI에 고유한 도메인 사용을 참조하세요.
4. 생성을 선택합니다.

로그인 페이지 보기

Amazon Cognito 콘솔에서 App integration(앱 통합) 탭의 App clients and analytics(앱 클라이언트 및 분석) 아래에 있는 앱 클라이언트 구성에서 View Hosted UI(호스팅 UI 보기) 버튼을 선택합니다. 이 버튼을 클릭하면 다음 기본 파라미터를 통해 호스팅 UI의 로그인 페이지로 이동합니다.

- 앱 클라이언트 ID

- 권한 부여 코드 부여 요청
- 현재 앱 클라이언트에 대해 활성화한 모든 범위에 대한 요청
- 현재 앱 클라이언트에 대한 목록의 첫 번째 콜백 URL

View hosted UI(호스팅 UI 보기) 버튼은 호스팅 UI의 기본 기능을 테스트하려는 경우에 유용합니다. 추가 및 수정된 파라미터를 사용하여 로그인 URL을 사용자 지정할 수 있습니다. 대부분의 경우 자동으로 생성된 파라미터가 View hosted UI(호스팅 UI 보기) 링크가 앱의 요구 사항과 완전히 일치하지 않습니다. 이러한 경우 앱에서 사용자가 로그인할 때 호출하는 URL을 사용자 지정해야 합니다. 파라미터 및 파라미터 값에 대한 자세한 내용은 사용자 풀 페더레이션 엔드포인트 및 호스팅 UI 참조 단원을 참조하십시오.

호스팅 UI 로그인 웹 페이지는 다음 URL 형식을 사용합니다. 이 예에서는 `response_type=code` 파라미터를 사용하여 인증 코드 권한 부여를 요청합니다.

```
https://<your domain>/oauth2/authorize?response_type=code&client_id=<your app client
 id>&redirect_uri=<your callback url>
```

앱 통합 탭에서 사용자 풀 도메인 문자열을 검색할 수 있습니다. 동일한 탭의 앱 클라이언트 및 분석에서 앱 클라이언트 ID, 콜백 URL, 허용 범위 및 기타 구성을 확인할 수 있습니다.

사용자 지정 파라미터를 사용하여 `/oauth2/authorize` 엔드포인트로 이동하면 Amazon Cognito가 `/oauth2/login` 엔드포인트로 리디렉션하거나 `identity_provider` 또는 `idp_identifier` 파라미터가 있는 경우 자동으로 IdP 로그인 페이지로 리디렉션합니다. 호스팅 UI를 우회하는 예제 URL은 Amazon Cognito 사용자 풀에서 SAML 세션 시작 단원을 참조하세요.

묵시적 부여를 위한 호스팅 UI 요청의 예

다음 URL을 사용하여 호스팅 UI 로그인 웹 페이지를 볼 수 있습니다. 여기서 `response_type=token`은 묵시적 코드 부여입니다. 로그인 성공 이후 Amazon Cognito는 사용자 풀 토큰을 웹 브라우저의 주소 표시줄로 반환합니다.

```
            https://mydomain.us-east-1.amazoncognito.com/authorize?
response_type=token&client_id=1example23456789&redirect_uri=https://
mydomain.example.com
```

자격 증명 및 액세스 토큰은 리디렉션 URL에 추가된 파라미터로 표시됩니다.

다음은 암시적 권한 부여 요청의 응답 예시입니다.

```
            https://mydomain.example.com/
#id_token=eyJraaBcDeF1234567890&access_token=eyJraGhIjKlM1112131415&expires_in=3600&token_type=
```

Amazon Cognito 사용자 풀 호스팅 UI에 대해 알아야 할 사항

호스팅 UI 및 사용자를 관리자로 확인

사용자 풀 로컬 사용자의 경우 Cognito가 검증 및 확인을 위한 메시지를 자동으로 보내도록 허용으로 사용자 풀을 구성할 때 호스팅 UI가 가장 잘 작동합니다. 이 설정을 활성화하면 Amazon Cognito는 가입한 사용자에게 확인 코드가 포함된 메시지를 보냅니다. 대신 사용자를 사용자 풀 관리자로 확인하면 가입 후 호스팅 UI에 오류 메시지가 표시됩니다. 이 상태에서 Amazon Cognito는 새 사용자를 생성했지만 검증 메시지를 보낼 수 없습니다. 여전히 사용자를 관리자로 확인할 수 있지만 오류가 발생할 경우 사용자가 지원 데스크에 문의할 수 있습니다. 관리 확인에 대한 자세한 내용은 사용자가 앱에 가입할 수 있도록 허용하지만 사용자 풀 관리자로 확인 섹션을 참조하세요.

호스팅 UI 구성 변경 사항 보기

호스트된 UI 페이지의 변경 사항이 즉시 표시되지 않는 경우 몇 분 동안 기다린 다음 페이지를 새로 고칩니다.

사용자 풀 토큰 디코딩

Amazon Cognito 사용자 풀 토큰은 RS256 알고리즘을 사용하여 서명됩니다. 를 사용하여 AWS Lambda사용자 풀 토큰을 디코딩하고 확인할 수 있습니다. 에서 Amazon Cognito JWT 토큰 디코딩 및 확인을 참조하십시오. GitHub

호스팅된 UI 및 TLS 버전

호스팅된 UI에는 전송 중 암호화가 필요합니다. Amazon Cognito에서 제공하는 사용자 풀 도메인에는 최소 1.2의 TLS 버전이 필요합니다. 사용자 지정 도메인은 TLS 버전 1.2를 지원하지만 필수는 아닙니다. Amazon Cognito는 호스팅된 UI 및 권한 부여 서버 엔드포인트의 구성을 관리하므로 사용자 풀 도메인의 TLS 요구 사항을 수정할 수 없습니다.

호스팅 UI 및 CORS 정책

Amazon Cognito 호스팅 UI는 사용자 지정 CORS(Cross-Origin Resource Sharing) 원본 정책을 지원하지 않습니다. 호스팅 UI의 CORS 정책은 사용자가 요청에 인증 파라미터를 전달하지 못하도록 합니다. 대신 앱의 웹 프론트엔드에서 CORS 정책을 구현하세요. Amazon Cognito는 다음 OAuth 엔드포인트에 대한 요청에 `Access-Control-Allow-Origin: *` 응답 헤더를 반환합니다.

1. Token 엔드포인트
2. 취소 엔드포인트
3. UserInfo 엔드포인트

호스팅된 UI 및 권한 부여 서버 쿠키

Amazon Cognito 사용자 풀 엔드포인트는 사용자 브라우저에 쿠키를 설정합니다. 쿠키는 사이트에서 타사 쿠키를 설정하지 않는 일부 브라우저의 요구 사항을 준수합니다. 쿠키의 범위는 사용자 풀 엔드포인트에만 적용되며 다음을 포함합니다.

- 각 `XSRF-TOKEN` 요청에 대한 쿠키.
- 사용자가 리디렉션될 때 세션 일관성을 유지하기 위한 `csrf-state` 쿠키입니다.
- 성공적인 로그인 시도를 한 시간 동안 보존하는 `cognito` 세션 쿠키입니다.

사용자 풀 도메인 구성

앱 클라이언트를 설정한 후 가입 및 로그인 웹 페이지의 주소를 구성할 수 있습니다. Amazon Cognito 호스트된 도메인을 사용하여 사용 가능한 도메인 접두사를 선택하거나 자체 웹 주소를 사용자 지정 도메인으로 사용할 수 있습니다.

AWS Management Console을 사용하여 앱 클라이언트와 Amazon Cognito 호스트된 도메인을 추가하려면 앱을 추가하여 호스트된 웹 UI 사용을 참조하세요.

> **Note**
>
> `aws`, `amazon` 또는 `cognito` 텍스트는 도메인 접두사에 사용할 수 없습니다.

주제

- 호스트된 UI에 Amazon Cognito 도메인 사용
- 호스트된 UI에 고유한 도메인 사용

호스트된 UI에 Amazon Cognito 도메인 사용

앱 클라이언트를 설정한 후 가입 및 로그인 웹 페이지의 주소를 구성할 수 있습니다. 호스트된 Amazon Cognito 도메인을 고유한 도메인 접두사와 함께 사용할 수 있습니다.

> **Note**
>
> Amazon Cognito 애플리케이션의 보안을 강화하기 위해 사용자 풀 엔드포인트의 상위 도메인은 PSL(Public Suffix List)에 등록됩니다. PSL은 사용자의 웹 브라우저가 사용자 풀 엔드포인트와 해당 엔드포인트가 설정하는 쿠키를 일관되게 이해하는 데 도움이 됩니다.
> 사용자 풀 엔드포인트 상위 도메인은 다음과 같은 형식을 갖습니다.
>
> ```
> auth.Region.amazoncognito.com
> auth-fips.Region.amazoncognito.com
> ```

를 사용하여 앱 클라이언트 및 Amazon Cognito 호스팅 도메인을 추가하려면 을 참조하십시오 AWS Management Console. 앱 클라이언트 생성

주제

- 필수 조건
- 1단계: 호스트된 사용자 풀 도메인 구성
- 2단계: 로그인 페이지 확인

필수 조건

시작하려면 다음이 필요합니다.

- 앱 클라이언트가 포함된 사용자 풀. 자세한 내용은 사용자 풀 시작하기 섹션을 참조하세요.

1단계: 호스트된 사용자 풀 도메인 구성

호스트된 사용자 풀 도메인을 구성하려면

OR AWS Management Console AWS CLI 또는 API를 사용하여 사용자 풀 도메인을 구성할 수 있습니다.

Amazon Cognito console

도메인 구성

1. 사용자 풀의 [앱 통합(App integration)] 탭으로 이동합니다.
2. 도메인 옆의 작업을 선택한 다음, 사용자 지정 도메인 생성 또는 Amazon Cognito 도메인 생성을 선택합니다. 사용자 풀 도메인을 이미 구성한 경우 새 사용자 지정 도메인을 생성하기 전에 Amazon Cognito 도메인 삭제 또는 사용자 지정 도메인 삭제를 선택합니다.
3. Amazon Cognito 도메인에서 사용할 사용 가능한 도메인 접두사를 입력합니다. 사용자 정의 도메인을 설정하는 방법에 대한 자세한 내용은 호스트된 UI에 고유한 도메인 사용을 참조하세요.
4. 생성을 선택합니다.

CLI/API

다음 명령을 사용하여 도메인 접두사를 생성하고 사용자 풀에 할당합니다.

사용자 풀 도메인을 구성하려면

- AWS CLI: `aws cognito-idp create-user-pool-domain`

  예: `aws cognito-idp create-user-pool-domain --user-pool-id <user_pool_id> --domain <domain_name>`
- AWS API: CreateUserPoolDomain

도메인 정보를 가져오려면

- AWS CLI: `aws cognito-idp describe-user-pool-domain`

  예: `aws cognito-idp describe-user-pool-domain --domain <domain_name>`
- AWS API: DescribeUserPoolDomain

도메인을 삭제하려면

- AWS CLI: `aws cognito-idp delete-user-pool-domain`

  예: `aws cognito-idp delete-user-pool-domain --domain <domain_name>`
- AWS API: DeleteUserPoolDomain

2단계: 로그인 페이지 확인

- Amazon Cognito 호스트된 도메인에서 로그인 페이지를 사용할 수 있는지 확인합니다.

```
https://<your_domain>/login?
response_type=code&client_id=<your_app_client_id>&redirect_uri=<your_callback_url>
```

도메인은 Amazon Cognito 콘솔의 [도메인 이름(Domain name)] 페이지에서 표시됩니다. 앱 클라이언트 설정(App client settings) 페이지에 앱 클라이언트 ID와 콜백 URL이 표시됩니다.

호스트된 UI에 고유한 도메인 사용

앱 클라이언트를 설정한 후 Amazon Cognito 호스팅 UI 및 인증 API 엔드포인트에 대한 사용자 지정 도메인으로 사용자 풀을 구성할 수 있습니다. 사용자 지정 도메인을 사용하면 사용자가 자체 웹 주소를 사용하여 애플리케이션에 로그인할 수 있습니다.

주제

- 사용자 풀에 사용자 정의 도메인 추가
- 사용자 정의 도메인의 SSL 인증서 변경

사용자 풀에 사용자 정의 도메인 추가

사용자 풀에 사용자 지정 도메인을 추가하려면 Amazon Cognito 콘솔에서 도메인 이름을 지정하고 AWS Certificate Manager(ACM)을 사용하여 관리하는 인증서를 제공합니다. 도메인을 추가하고 나면 Amazon Cognito는 사용자가 DNS 구성에 추가할 별칭 대상을 제공합니다.

필수 조건

시작하려면 다음이 필요합니다.

- 앱 클라이언트가 포함된 사용자 풀. 자세한 내용은 사용자 풀 시작하기 섹션을 참조하세요.
- 자신이 소유한 웹 도메인. 상위 도메인에 유효한 DNS A 레코드가 있어야 합니다. 이 레코드에 어떤 값이든 할당할 수 있습니다. 상위 도메인은 도메인의 루트 도메인이거나 도메인 계층 구조에서 한 단계 높은 자식 도메인일 수 있습니다. 예를 들어 사용자 지정 도메인이 auth.xyz.example.com인 경우 Amazon Cognito는 xyz.example.com을 IP 주소로 확인할 수 있어야 합니다. 고객 인프라에 대한 우발적인 영향을 방지하기 위해 Amazon Cognito는 사용자 지정 도메인에 대한 최상위 도메인(TLD) 사용을 지원하지 않습니다. 자세한 내용은 도메인 이름을 참조하세요.

- 사용자 지정 도메인의 하위 도메인을 생성하는 기능 auth를 하위 도메인으로 사용하는 것이 좋습니다. 예: *auth.example.com*.

> **Note**
>
> 와일드카드 인증서가 없는 경우 사용자 지정 도메인의 하위 도메인에 대한 새 인증서를 받아야 할 수 있습니다.

- ACM에서 관리하는 보안 소켓 계층(SSL) 인증서.

> **Note**
>
> 인증서를 요청하거나 가져오기 전에 ACM 콘솔에서 AWS 지역을 미국 동부 (버지니아 북부)로 변경해야 합니다.

- 사용자 풀 인증 서버가 사용자 세션에 쿠키를 추가할 수 있도록 허용하는 애플리케이션입니다. Amazon Cognito는 호스팅된 UI에 필요한 몇 가지 쿠키를 설정합니다. `cognito`, `cognito-fl` 및 `XSRF-TOKEN`가 포함됩니다. 각 개별 쿠키는 브라우저 크기 제한을 준수하지만 사용자 풀 구성을 변경하면 호스팅된 UI 쿠키의 크기가 커질 수 있습니다. 사용자 지정 도메인 앞에 있는 Application Load Balancer (ALB) 와 같은 중간 서비스는 최대 헤더 크기 또는 전체 쿠키 크기를 적용할 수 있습니다. 애플리케이션이 자체 쿠키도 설정하는 경우 사용자 세션이 이러한 제한을 초과할 수 있습니다. 크기 제한 충돌을 방지하려면 애플리케이션에서 호스팅된 UI 하위 도메인에 쿠키를 설정하지 않는 것이 좋습니다.
- Amazon CloudFront 배포판을 업데이트할 수 있는 권한. 다음 IAM 정책 문을 AWS 계정의 IAM 사용자, 그룹 또는 역할에 연결하면 이 작업을 할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Sid": "AllowCloudFrontUpdateDistribution",
            "Effect": "Allow",
            "Action": [
                "cloudfront:updateDistribution"
            ],
            "Resource": [
                "*"
            ]
        }
```

```
    ]
}
```

에서 CloudFront 작업을 승인하는 방법에 대한 자세한 내용은 ID 기반 정책 (IAM 정책) 사용을 참조하십시오. CloudFront

Amazon Cognito는 처음에 IAM 권한을 사용하여 배포를 구성하지만 CloudFront 배포는 에서 관리합니다. AWS Amazon Cognito가 사용자 풀에 연결한 CloudFront 배포의 구성은 변경할 수 없습니다. 예를 들어 보안 정책에서 지원되는 TLS 버전을 업데이트할 수 없습니다.

1단계: 사용자 정의 도메인 이름 입력

Amazon Cognito 콘솔 또는 API를 사용하여 도메인을 사용자 풀에 추가할 수 있습니다.

Amazon Cognito console

Amazon Cognito 콘솔에서 사용자 풀에 도메인을 추가하려면

1. Amazon Cognito 콘솔에 로그인합니다. 메시지가 표시되면 AWS 자격 증명을 입력합니다.
2. [사용자 풀(User pools)]을 선택합니다.
3. 도메인을 추가할 사용자 풀을 선택합니다.
4. [앱 통합(App integration)] 탭을 선택합니다.
5. [도메인(Domain)] 옆의 [작업(Actions)]을 선택한 다음, [사용자 정의 도메인 생성(Create custom domain)]을 선택합니다.

   > **Note**
   >
   > 사용자 풀 도메인을 이미 구성한 경우 새 사용자 정의 도메인을 생성하기 전에 [Cognito 도메인 삭제(Delete Cognito domain)] 또는 [사용자 정의 도메인 삭제(Delete custom domain)]를 선택하여 기존 도메인을 삭제합니다.

6. [사용자 정의 도메인(Custom domain)]에 Amazon Cognito에서 사용할 도메인의 URL을 입력합니다. 도메인 이름에는 소문자, 숫자, 하이픈(-)만 사용할 수 있습니다. 첫 번째나 마지막 자리에 하이픈을 사용해서는 안 됩니다. 마침표로 하위 도메인 이름을 구분하세요.
7. [ACM 인증서(ACM certificate)]에서 도메인에 사용할 SSL 인증서를 선택합니다. 사용자 풀과 상관없이 미국 동부 (버지니아 북부) 의 AWS 리전 ACM 인증서만 Amazon Cognito 사용자 지정 도메인과 함께 사용할 수 있습니다.

사용 가능한 인증서가 없는 경우 ACM을 사용하여 미국 동부(버지니아 북부)에서 인증서를 프로비저닝할 수 있습니다. 자세한 내용은 AWS Certificate Manager 사용 설명서에서 시작하기를 참조하세요.

8. 생성을 선택합니다.
9. Amazon Cognito가 [앱 통합(App integration)] 탭으로 돌아갑니다. [도메인의 DNS에 별칭 레코드 생성(Create an alias record in your domain's DNS)]이라는 제목의 메시지가 표시됩니다. 콘솔에 표시된 [도메인(Domain)]과 [별칭 대상(Alias target)]을 적어 둡니다. 이 정보는 다음 단계에서 트래픽을 사용자 정의 도메인으로 보내는 데 사용됩니다.

API

Amazon Cognito API를 사용하여 사용자 풀에 도메인을 추가하려면

- CreateUserPoolDomain 작업을 사용합니다.

2단계: 별칭 대상 및 하위 도메인 추가

이 단계에서는 도메인 이름 서버(DNS) 서비스 공급자를 통해, 이전 단계의 별칭 대상을 다시 가리키는 별칭을 설정합니다. DNS 주소 확인에 Amazon Route 53을 사용하는 경우 Route 53을 사용하여 별칭 대상 및 하위 도메인을 추가하려면 섹션을 선택하세요.

현재 DNS 구성에 별칭 대상 및 하위 도메인을 추가하려면

- DNS 주소 확인에 Route 53을 사용하지 않는 경우 DNS 서비스 공급자의 구성 도구를 사용하여 이전 단계의 별칭 대상을 도메인의 DNS 레코드에 추가해야 합니다. 또한 DNS 공급자가 사용자 지정 도메인에 대한 하위 도메인을 설정해야 합니다.

Route 53을 사용하여 별칭 대상 및 하위 도메인을 추가하려면

1. Route 53 콘솔에 로그인합니다. 메시지가 표시되면 AWS 자격 증명을 입력합니다.
2. Route 53에 호스팅 영역이 없는 경우, 사용자 지정 도메인의 상위인 루트를 사용하여 호스팅 영역을 생성하십시오. 자세한 내용을 알아보려면 다음 섹션을 참조하세요.

   a. Create Hosted Zone(호스팅 영역 생성)을 선택합니다.

   b. [도메인 이름(Domain Name)] 목록에서 사용자 정의 도메인(예: *myapp.auth.example.com*)의 상위 도메인(예: *auth.example.com*)을 입력합니다.

c. 호스팅 영역에 대한 설명을 입력합니다.

d. 퍼블릭 클라이언트가 사용자 정의 도메인을 확인할 수 있도록 호스팅 영역 [유형(Type)]으로 [퍼블릭 호스팅 영역(Public hosted zone)]을 선택합니다. [프라이빗 호스팅 영역(Private hosted zone)]은 선택할 수 없습니다.

e. 원하는 대로 [태그(Tags)]를 적용합니다.

f. 호스팅 영역 생성(Create hosted zone)을 선택합니다.

> **Note**
>
> 사용자 정의 도메인에 대한 새 호스팅 영역을 생성할 수도 있으며, 상위 호스팅 영역에 쿼리를 하위 도메인 호스팅 영역으로 보내는 위임 집합을 생성할 수 있습니다. 그렇지 않으면 A 레코드를 생성하십시오. 이 방법을 사용하면 호스팅 영역의 유연성과 보안이 강화됩니다. 자세한 내용은 Amazon Route 53을 통해 호스트되는 도메인의 하위 도메인 생성을 참조하세요.

3. 호스팅 영역(Hosted Zones) 페이지에서 호스팅 영역의 이름을 선택합니다.

4. 사용자 지정 도메인의 상위 도메인이 아직 없는 경우 해당 도메인의 상위 도메인에 대한 DNS 레코드를 추가하세요. 상위 도메인의 DNS A 레코드를 추가하고 레코드 생성을 선택합니다. 다음은 *auth.example.com* 도메인에 대한 예제 레코드입니다.

```
auth.example.com. 60 IN A 198.51.100.1
```

> **Note**
>
> Amazon Cognito는 프로덕션 도메인이 실수로 하이재킹되지 않도록 보호하기 위해 사용자 정의 도메인의 상위 도메인에 대한 DNS 레코드가 있는지 확인합니다. 상위 도메인에 대한 DNS 레코드가 없는 경우 사용자 정의 도메인을 설정하려고 하면 Amazon Cognito에서 오류를 반환합니다. SOA (권한 시작) 레코드는 부모 도메인 확인을 위한 충분한 DNS 레코드가 아닙니다.

5. 커스텀 도메인용 DNS 레코드를 추가하세요. 예를 `123example.cloudfront.net` 들어 레코드는 사용자 지정 도메인 별칭 대상을 가리켜야 합니다. [레코드 생성(Create record)]을 다시 선택합니다.

6. 사용자 정의 도메인과 일치하는 [레코드 이름(Record name)](예: *myapp*)을 입력하여 *myapp.auth.example.com*에 대한 레코드를 생성합니다.

7. [별칭(Alias)] 옵션을 사용하도록 설정합니다.

8. [다음으로 트래픽 라우팅(Route traffic to)]에서 [CloudFront 배포에 대한 별칭(Alias to Cloudfront distribution)]을 선택합니다. 사용자 정의 도메인을 생성할 때 Amazon Cognito에서 제공한 [별칭 대상(Alias target)]을 입력합니다.
9. [레코드 생성(Create Records)]을 선택합니다.

> **Note**
>
> 새 레코드가 모든 Route 53 DNS 서버에 전파되는 데 약 60초가 걸릴 수 있습니다. Route 53 GetChangeAPI 방법을 사용하여 변경 내용이 전파되었는지 확인할 수 있습니다.

3단계: 로그인 페이지 확인

- 사용자 지정 도메인에서 로그인 페이지를 사용할 수 있는지 확인합니다.

  브라우저에 이 주소를 입력하여 사용자 지정 도메인과 하위 도메인으로 로그인합니다. 다음은 사용자 지정 도메인이 *example.com*이고 하위 도메인이 *auth*인 URL의 예입니다.

```
https://myapp.auth.example.com/login?
response_type=code&client_id=<your_app_client_id>&redirect_uri=<your_callback_url>
```

사용자 정의 도메인의 SSL 인증서 변경

필요한 경우 Amazon Cognito를 사용하여 사용자 지정 도메인에 적용한 인증서를 변경할 수 있습니다.

일반적으로 ACM을 사용한 일상적인 인증서 갱신 후에는 이 작업이 필요하지 않습니다. ACM에서 기존 인증서를 갱신할 때 인증서의 ARN은 동일하게 유지되며 사용자 지정 도메인은 새 인증서를 자동으로 사용합니다.

그러나 기존 인증서를 새 인증서로 교체하는 경우에는 ACM이 새 인증서에 새 ARN을 할당합니다. 사용자 지정 도메인에 새 인증서를 적용하려면 Amazon Cognito에 이 ARN을 제공해야 합니다.

새 인증서를 제공한 후, Amazon Cognito가 사용자 지정 도메인에 인증서를 배포하는 데 최대 1시간이 소요됩니다.

> **시작하기 전 준비 사항**
>
> Amazon Cognito에서 인증서를 변경하려면 먼저 ACM에 인증서를 추가해야 합니다. 자세한 내용은 AWS Certificate Manager 사용 설명서에서 시작하기를 참조하세요.
> 인증서를 ACM에 추가할 때 미국 동부(버지니아 북부)를 AWS 리전으로 선택해야 합니다.

Amazon Cognito 콘솔 또는 API를 사용하여 인증서를 변경할 수 있습니다.

AWS Management Console

Amazon Cognito 콘솔에서 인증서를 갱신하려면

1. 에서 Amazon Cognito 콘솔에 AWS Management Console 로그인하고 엽니다. https://console.aws.amazon.com/cognito/home
2. [사용자 풀(User Pools)]을 선택합니다.
3. 인증서를 업데이트할 사용자 풀을 선택합니다.
4. [앱 통합(App integration)] 탭을 선택합니다.
5. [작업(Actions)], [ACM 인증서 편집(Edit ACM certificate)]을 차례로 선택합니다.
6. 사용자 정의 도메인에 연결할 새 인증서를 선택합니다.
7. 변경 사항 저장(Save changes)을 선택합니다.

API

인증서를 갱신하려면(Amazon Cognito API)

- UpdateUserPoolDomain 작업을 사용합니다.

기본 제공 로그인 및 가입 웹 페이지 사용자 정의

AWS Management Console이나 AWS CLI 또는 API를 사용하여 내장 앱 UI 환경에 대한 사용자 지정 설정을 지정할 수 있습니다. 앱에 표시할 사용자 지정 로고 이미지를 업로드할 수 있습니다. 계단식 스타일 시트(CSS)를 사용하여 UI의 모양을 사용자 정의할 수도 있습니다.

단일 클라이언트(특정 `clientId`를 가진) 또는 모든 클라이언트(`clientId`를 `ALL`로 설정)에서 앱 UI 사용자 지정 설정을 지정할 수 있습니다. `ALL`을 지정하면 이전에 UI 사용자 지정이 설정되지 않은 모든

클라이언트에서 기본 구성이 사용됩니다. 특정 클라이언트에 대해 UI 사용자 지정 설정을 지정한 경우에는 더 이상 ALL 구성으로 돌아가지 않습니다.

UI 사용자 지정을 설정하는 요청은 크기가 135KB를 초과할 수 없습니다. 드물지만 요청 헤더, CSS 파일 및 로고의 합계가 135KB를 초과할 때도 있습니다. Amazon Cognito는 이미지 파일을 Base64로 인코딩합니다. 이렇게 하면 100KB 이미지의 크기가 130KB로 늘어나, 요청 헤더와 CSS에는 5KB가 유지됩니다. 요청이 너무 크면 AWS Management Console 또는`SetUICustomization` API 요청에서 `request parameters too large` 오류를 반환합니다. 로고 이미지를 100KB 이하로 조정하고 CSS 파일은 3KB 이하로 조정하세요. CSS와 로고 사용자 지정을 별도로 설정할 수는 없습니다.

> **Note**
>
> UI를 사용자 지정하려면 사용자 풀에 대한 도메인을 설정해야 합니다.

앱의 사용자 정의 로고 지정

Amazon Cognito는 사용자 지정 로고를 Login 엔드포인트의 입력 필드 위 중간에 배치합니다.

사용자 지정 호스팅 UI 로고용으로 350x178픽셀까지 확대할 수 있는 PNG, JPG 또는 JPEG 파일을 선택합니다. 로고 파일의 크기는 100KB를 초과할 수 없으며, Amazon Cognito가 Base64로 인코딩한 후에는 130KB를 초과할 수 없습니다. API의 SetUICustomization에서 `ImageFile`을 설정하려면 파일을 Base64로 인코딩된 텍스트 문자열로 변환하거나, AWS CLI에 파일 경로를 입력하고 Amazon Cognito가 사용자 대신 인코딩하게 합니다.

앱의 CSS 사용자 정의 지정

호스팅된 앱 페이지에서 CSS를 사용자 지정할 수 있지만, 다음과 같은 제한 사항이 있습니다.

- 다음 CSS 클래스 이름 중 하나를 사용할 수 있습니다.
  - `background-customizable`
  - `banner-customizable`
  - `errorMessage-customizable`
  - `idpButton-customizable`
  - `idpButton-customizable:hover`
  - `idpDescription-customizable`
  - `inputField-customizable`

- `inputField-customizable:focus`
- `label-customizable`
- `legalText-customizable`
- `logo-customizable`
- `passwordCheck-valid-customizable`
- `passwordCheck-notValid-customizable`
- `redirect-customizable`
- `socialButton-customizable`
- `submitButton-customizable`
- `submitButton-customizable:hover`
- `textDescription-customizable`

- 속성 값에는 `@import`, `@supports`, `@page` 또는 `@media` 문이나 Javascript 값을 제외한 HTML이 포함될 수 있습니다.

다음 CSS 속성을 사용자 지정할 수 있습니다.

Labels

- font-weight는 100에서 900 사이의 100의 배수입니다.

입력 필드

- width는 포함 블록의 백분율로 표시되는 너비입니다.
- height는 픽셀(px) 단위의 입력 필드 높이입니다.
- color는 텍스트 색상입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.
- background-color는 입력 필드의 배경색입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.
- border는 앱 창 경계의 너비, 투명도 및 색상을 지정하는 표준 CSS 경계 값입니다. 너비의 범위는 1~100px입니다. 투명도는 단색(solid) 또는 없음(none)입니다. 표준 색상 값이면 무엇이든 가능합니다.

텍스트 설명

- padding-top은 텍스트 설명 위의 패딩 양입니다.
- padding-bottom은 텍스트 설명 아래의 패딩 양입니다.
- display는 `block` 또는 `inline`일 수 있습니다.
- font-size는 텍스트 설명의 글꼴 크기 입니다.

제출 버튼

- font-size는 버튼 텍스트의 글꼴 크기 입니다.
- font-weight는 버튼 텍스트의 글꼴 두께로 `bold`, `italic` 또는 `normal`일 수 있습니다.
- margin은 버튼의 위쪽, 오른쪽, 아래쪽 및 왼쪽 마진 크기를 나타내는 4자 문자열입니다.
- font-size는 텍스트 설명의 글꼴 크기 입니다.
- width는 포함 블록의 비율로 표시되는 버튼 텍스트의 너비입니다.
- height는 픽셀(px) 단위의 버튼 높이입니다.
- color는 버튼 텍스트 색상입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.
- background-color는 버튼의 배경색입니다. 표준 색상 값이면 무엇이든 가능합니다.

배너

- padding은 배너의 위쪽, 오른쪽, 아래쪽 및 왼쪽 패딩 크기를 나타내는 4자 문자열입니다.
- background-color는 배너의 배경색입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.

제출 버튼 가리키기

- color는 커서를 올려 놓았을 때 버튼의 전경색입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.
- background-color는 커서를 올려 놓았을 때 버튼의 배경색입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.

자격 증명 공급자 버튼 가리키기

- color는 커서를 올려 놓았을 때 버튼의 전경색입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.
- background-color는 커서를 올려 놓았을 때 버튼의 배경색입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.

암호 확인이 유효하지 않음

- color는 `"Password check not valid"` 메시지의 텍스트 색상입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.

배경

- background-color는 앱 창의 배경색입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.

오류 메시지

- margin은 위쪽, 오른쪽, 아래쪽 및 왼쪽 마진 크기를 나타내는 4자 문자열입니다.
- padding은 패딩 크기입니다.
- font-size는 글꼴 크기 입니다.

- width는 포함 블록의 비율로 표시되는 오류 메시지의 너비입니다.
- background-color는 오류 메시지의 배경색입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.
- border는 경계의 너비, 투명도 및 색상을 지정하는 3자 문자열입니다.
- color는 오류 메시지 텍스트 색상입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.
- box-sizing은 크기 속성(너비 및 높이)에 포함시켜야 할 브라우저를 나타내는 데 사용됩니다.

자격 증명 공급자 버튼

- height는 픽셀(px) 단위의 버튼 높이입니다.
- width는 포함 블록의 비율로 표시되는 버튼 텍스트의 너비입니다.
- text-align은 텍스트 정렬 설정입니다. `left`, `right` 또는 `center`일 수 있습니다.
- margin-bottom은 아래쪽 마진 설정입니다.
- color는 버튼 텍스트 색상입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.
- background-color는 버튼의 배경색입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.
- border-color는 버튼 경계의 색상입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.

자격 증명 공급자 설명

- padding-top은 설명 위의 패딩 양입니다.
- padding-bottom은 설명 아래의 패딩 양입니다.
- display는 `block` 또는 `inline`일 수 있습니다.
- font-size는 설명의 글꼴 크기 입니다.

법적 고지 텍스트

- color는 텍스트 색상입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.
- font-size는 글꼴 크기 입니다.

> **Note**
>
> [법적 고지 텍스트(Legal text)]를 사용자 정의하는 경우 로그인 페이지에서 소셜 자격 증명 공급자 아래에 표시되는 [먼저 묻지 않고 계정에 게시하지 않습니다.(We won't post to any of your accounts without asking first)]라는 메시지를 사용자 정의하는 것입니다.

로고

- max-width는 포함 블록의 비율로 표시되는 최대 너비입니다.
- max-height는 포함 블록의 비율로 표시되는 최대 높이입니다.

입력 필드 포커스

- border-color는 입력 필드의 색상입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.
- outline은 픽셀 단위의 입력 필드 경계 너비입니다.

소셜 버튼

- height는 픽셀(px) 단위의 버튼 높이입니다.
- text-align은 텍스트 정렬 설정입니다. `left`, `right` 또는 `center`일 수 있습니다.
- width는 포함 블록의 비율로 표시되는 버튼 텍스트의 너비입니다.
- margin-bottom은 아래쪽 마진 설정입니다.

암호 확인이 유효함

- color는 `"Password check valid"` 메시지의 텍스트 색상입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.

사용자 풀에 대한 앱 UI 사용자 정의 설정 지정(AWS Management Console)

AWS Management Console을 사용하여 앱의 UI 사용자 지정 설정을 지정할 수 있습니다.

> **Note**
>
> 사용자 풀에 대한 세부 사항을 포함시켜 URL `https://<your_domain>/login?response_type=code&client_id=<your_app_client_id>&redirect_uri=<your_callbac` 구성하고 이를 브라우저에 입력하면 사용자가 지정한 호스팅 UI를 볼 수 있습니다. 브라우저를 새로고침하고 콘솔의 변경 내용이 나타나기까지 최대 1분 정도 기다려야 할 수도 있습니다. 도메인은 [앱 통합(App integration)] 탭의 [도메인(Domain)] 아래에 표시됩니다. 앱 클라이언트 ID와 콜백 URL은 [앱 클라이언트(App clients)] 아래에 표시됩니다.

앱 UI 사용자 지정 설정 지정 방법

1. Amazon Cognito 콘솔에 로그인합니다.
2. 탐색 창에서 [사용자 풀(User Pools)]을 선택한 다음 편집할 사용자 풀을 선택합니다.
3. [앱 통합(App integration)] 탭을 선택합니다.
4. 모든 앱 클라이언트에 대한 UI 설정을 사용자 정의하려면 [호스트된 UI 사용자 정의(Hosted UI customization)]를 찾아서 [편집(Edit)]을 선택합니다.
5. 단일 앱 클라이언트의 UI 설정을 사용자 지정하려면 앱 클라이언트를 찾고 수정할 앱 클라이언트를 선택한 다음 호스팅 UI 사용자 지정을 찾고 편집을 선택합니다. 앱 클라이언트를 사용자 풀

기본 사용자 정의에서 클라이언트별 사용자 정의로 전환하려면 [클라이언트 수준 설정 사용(Use client-level settings)]을 선택합니다.

6. 고유한 로고 이미지 파일을 업로드하려면 [파일 선택(Choose file)] 또는 [현재 파일 바꾸기(Replace current file)]를 선택합니다.
7. 호스팅 UI CSS를 사용자 지정하려면 CSS template.css를 다운로드한 다음, 사용자 지정하려는 값으로 템플릿을 수정합니다. 템플릿에 포함된 키만 호스트된 UI에 사용할 수 있습니다. 추가된 CSS 키는 UI에 반영되지 않습니다. CSS 파일을 사용자 정의한 후 [파일 선택(Choose file)] 또는 [현재 파일 바꾸기(Replace current file)]를 선택하여 사용자 정의 CSS 파일을 업로드합니다.

사용자 풀에 대한 앱 UI 사용자 정의 설정 지정(AWS CLI 및 AWS API)

다음 명령을 사용하여 사용자 풀에 대한 앱 UI 사용자 지정 설정을 지정합니다.

사용자 풀의 내장 앱 UI에 대한 UI 사용자 지정 설정을 가져오려면 다음 API 작업을 사용합니다.

- AWS CLI: `aws cognito-idp get-ui-customization`
- AWS API: GetUICustomization

사용자 풀의 내장 앱 UI에 대한 UI 사용자 지정 설정을 설정하려면 다음 API 작업을 사용합니다.

- 이미지 파일의 AWS CLI: `aws cognito-idp set-ui-customization --user-pool-id <your-user-pool-id> --client-id <your-app-client-id> --image-file fileb://"<path-to-logo-image-file>" --css ".label-customizable{ color: <color>;}"`
- Base64 이진 텍스트로 인코딩된 이미지가 포함된 AWS CLI: `aws cognito-idp set-ui-customization --user-pool-id <your-user-pool-id> --client-id <your-app-client-id> --image-file <base64-encoded-image-file> --css ".label-customizable{ color: <color>;}"`
- AWS API: SetUICustomization

호스팅 UI로 가입 및 로그인

사용자 풀 및 앱 클라이언트에 Amazon Cognito 호스팅 UI를 구성하고 사용자 지정하면 앱에서 사용자에게 이를 표시할 수 있습니다. 호스팅 UI는 다음 예를 비롯한 여러 Amazon Cognito 인증 작업을 지원합니다.

- 앱에 새로운 사용자로 로그인
- 이메일 주소 또는 전화번호 인증
- 다중 인증(MFA) 설정
- 로컬 사용자 이름과 암호로 로그인
- 서드 파티 ID 제공업체(idP)로 로그인
- 암호 재설정

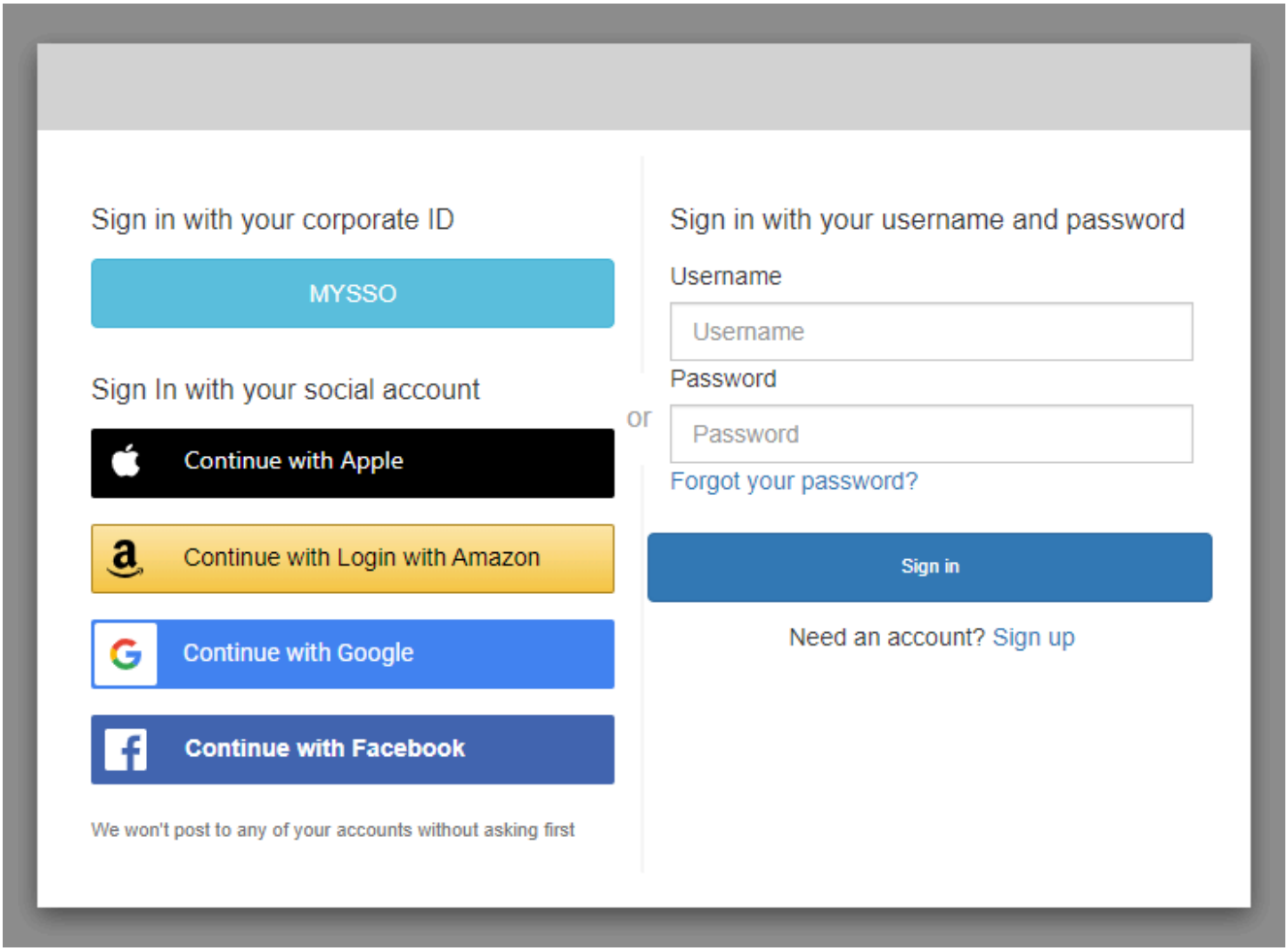
Amazon Cognito 호스팅 UI는 Login 엔드포인트에서 시작됩니다. 로그인 페이지의 URL은 사용자 풀에 대해 선택한 도메인과 발급하려는 OAuth 2.0 권한 부여, 앱 클라이언트, 앱 경로, 요청하려는 OpenID Connect(OIDC) 범위를 반영하는 파라미터의 조합입니다.

```
https://<your user pool domain>/authorize?client_id=<your app client
 ID>&response_type=<code/token>&scope=<scopes to request>&redirect_uri=<your callback
 URL>
```

다음 URL에서 위의 자리 표시자 필드를 예시 값으로 대체합니다.

```
https://auth.example.com/authorize? /
client_id=1example23456789 /
&response_type=code /
&scope=aws.cognito.signin.user.admin+email+openid+profile /
&redirect_uri=https%3A%2F%2Faws.amazon.com
```

Amazon Cognito 호스팅 UI의 로그인 페이지에는 사용자가 요청하는 앱 클라이언트에 할당한 모든 ID 제공업체(idP) 또는 사용자 풀을 통해 로그인할 수 있는 옵션이 있습니다. 여기에는 사용자 풀에서 새 사용자 계정에 등록하거나 잊어버린 암호를 재설정하는 링크도 포함됩니다.



주제

- Amazon Cognito 호스팅 UI에서 새 계정에 가입하는 방법
- Amazon Cognito 호스팅 UI에 로그인하는 방법
- Amazon Cognito 호스팅 UI에서 암호를 재설정하는 방법

Amazon Cognito 호스팅 UI에서 새 계정에 가입하는 방법

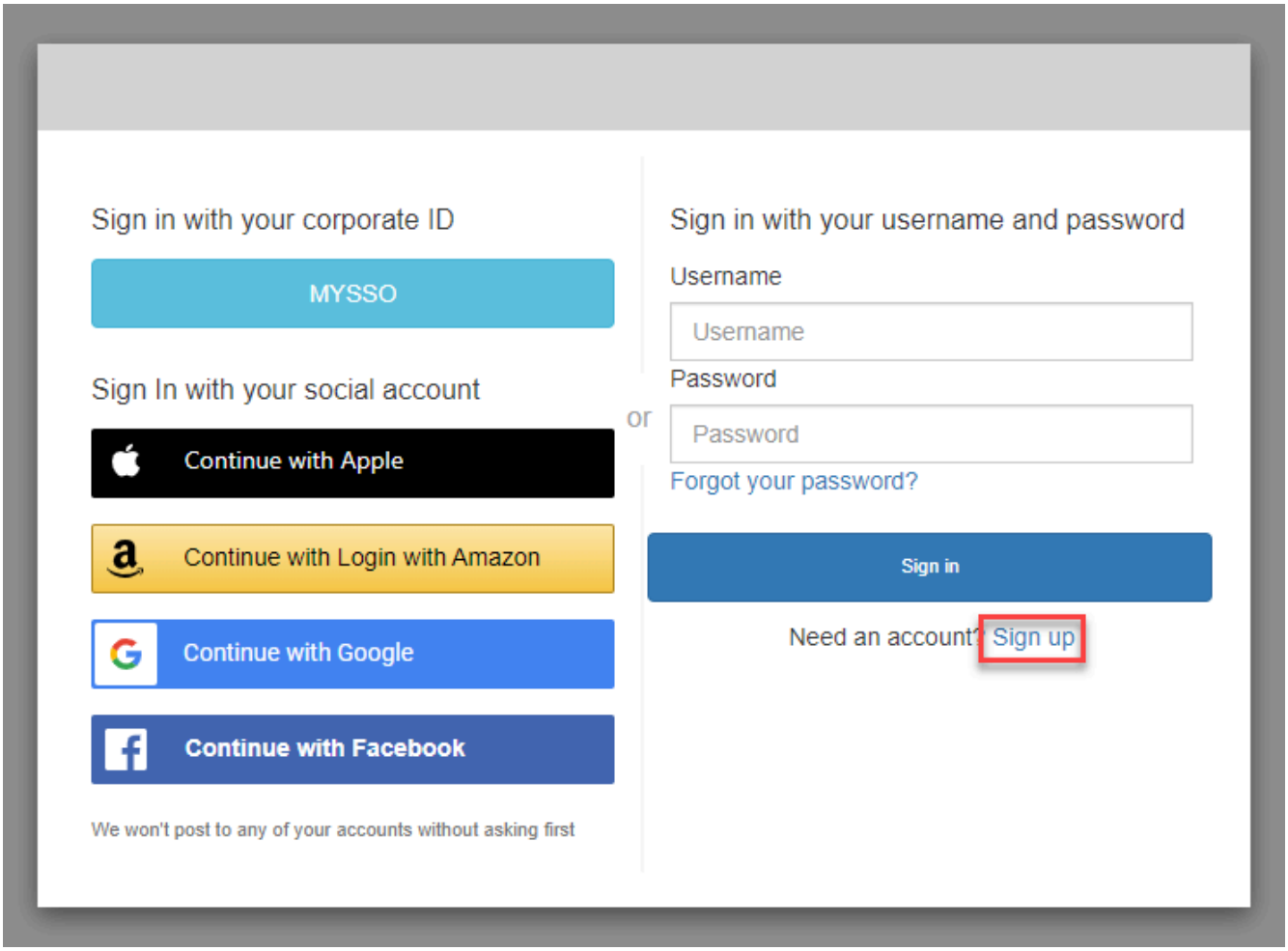
이 안내서에서는 Amazon Cognito를 사용하는 앱에서 사용자 계정에 가입하는 방법을 보여줍니다.

> **Note**
>
> Amazon Cognito 호스팅 사용자 인터페이스(UI)를 사용하는 앱에 로그인하면 이 안내서에 표시된 기본 구성 이상으로 앱 소유자가 사용자 지정한 페이지가 표시될 수 있습니다.

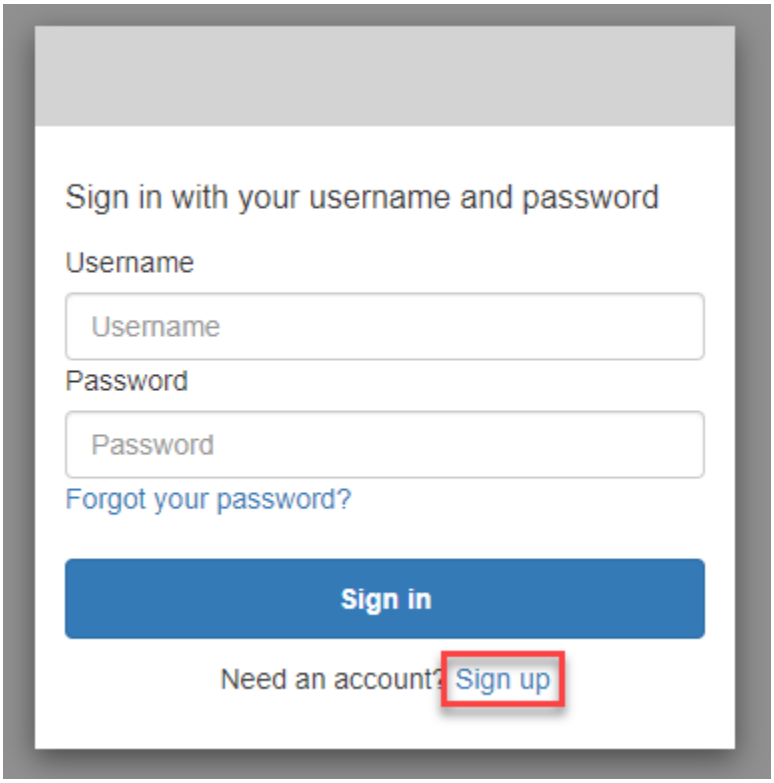
1. 앱 소유자가 나열한 서드 파티 로그인 서비스 제공업체 대신 사용자 이름과 암호를 사용하여 Amazon Cognito를 통해 로그인하려는 경우 로그인 페이지에서 Sign up(가입)을 선택합니다.

   Amazon Cognito가 아닌 경우 서드 파티 로그인 서비스 제공업체의 버튼을 선택하면 가입이 완료됩니다. 앱 소유자가 선택한 옵션에 따라 로그인할 제공업체를 선택하거나 사용자 이름과 암호를 묻는 메시지만 표시될 수 있습니다.

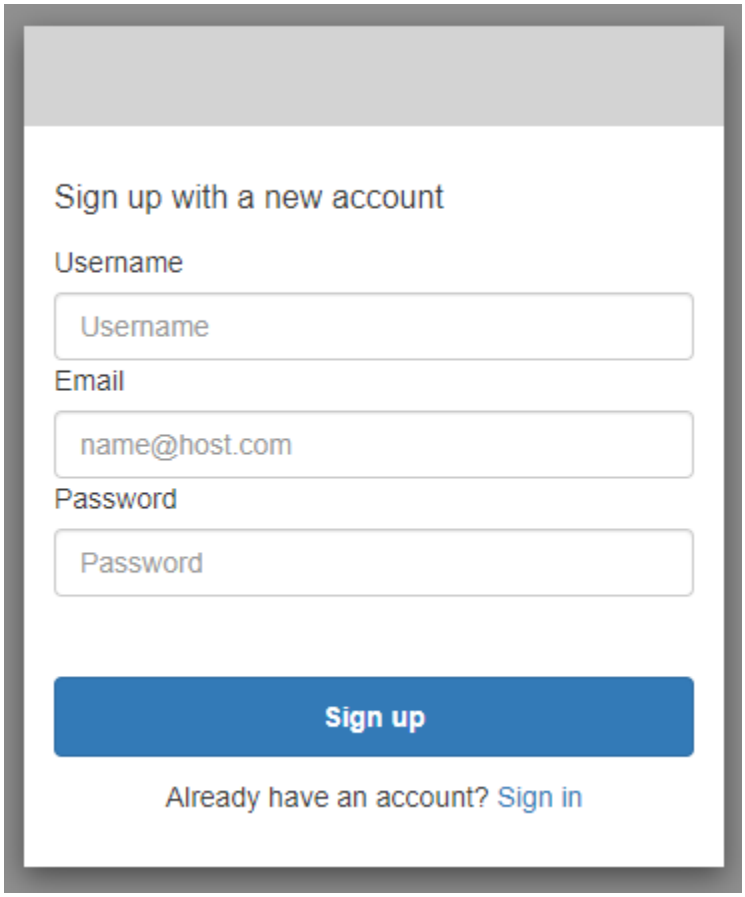
   With multiple sign-in providers



With only Amazon Cognito as a sign-in provider

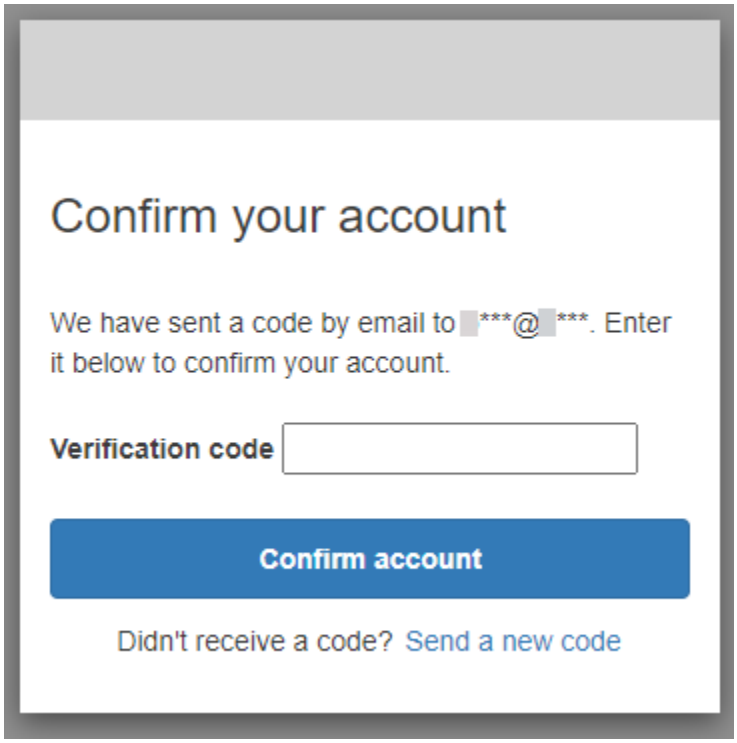


2. Sign up with a new account(새 계정으로 가입) 페이지에서 앱 소유자는 가입에 필요한 정보를 요청합니다. 사용자 이름, 이메일 주소 또는 전화번호를 요청할 수 있습니다. 필요한 정보를 입력한 후 암호를 선택합니다.



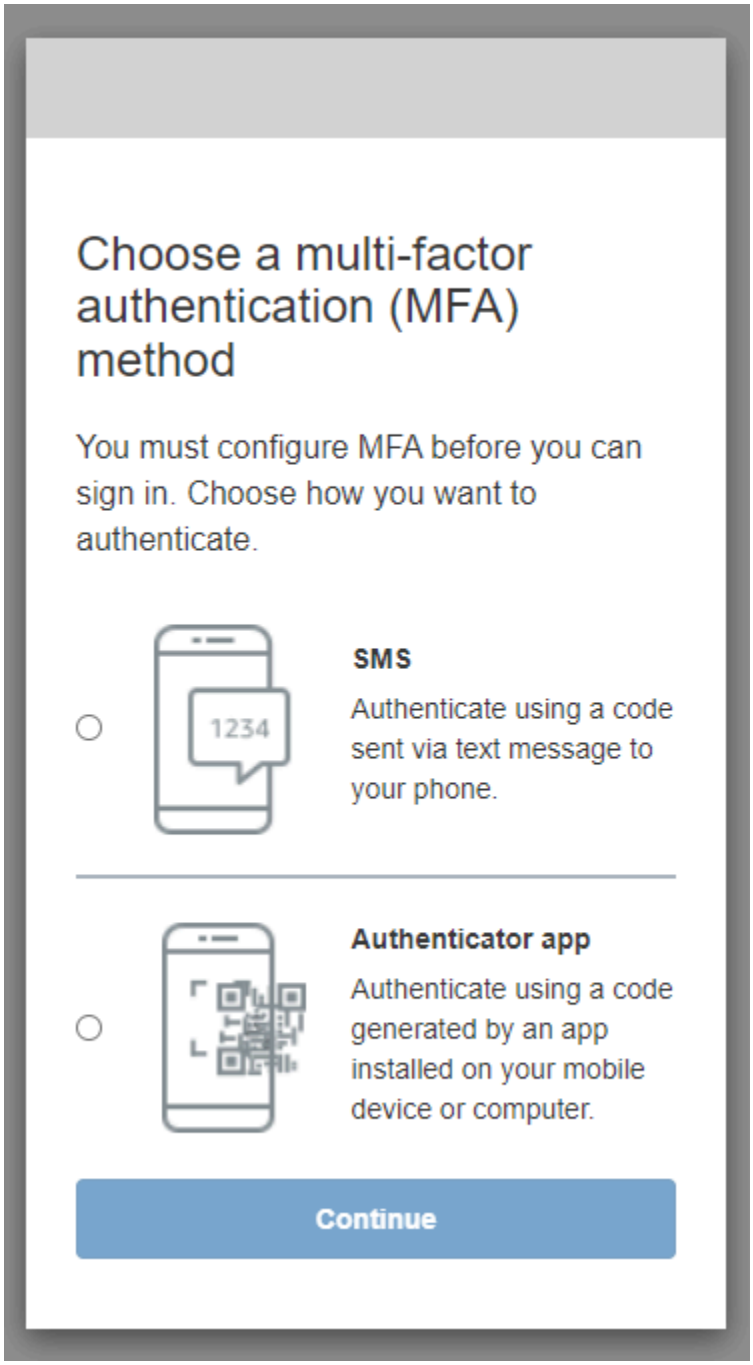
3. Confirm your account(계정 확인) 페이지에서 앱 소유자는 입력한 이메일 주소 또는 전화번호에서 메시지를 수신할 수 있는지 확인하기 위해 계정 확인을 요청할 수도 있습니다.

   이메일 또는 SMS 문자 메시지에 포함된 코드를 받게 됩니다. 양식에 코드를 입력하여 올바른 연락처 정보를 입력했는지 확인합니다.



4. 앱 소유자가 다중 인증(MFA)을 설정하도록 요구할 수 있습니다. MFA 방식을 선택하라는 메시지가 표시되거나 앱에서 다음 단계로 건너뛸 수 있습니다.

   Choose a multi-factor authentication (MFA) method(다중 인증(MFA) 방식 선택) 페이지에서 MFA 방식을 선택합니다. SMS를 선택하면 SMS 문자 메시지로 MFA 암호를 받게 됩니다. Authenticator app(인증 앱)을 선택하는 경우 시간 기반 MFA 암호를 생성하기 위해 디바이스에 앱을 설치해야 합니다. 3분 이내에 선택해야 합니다.



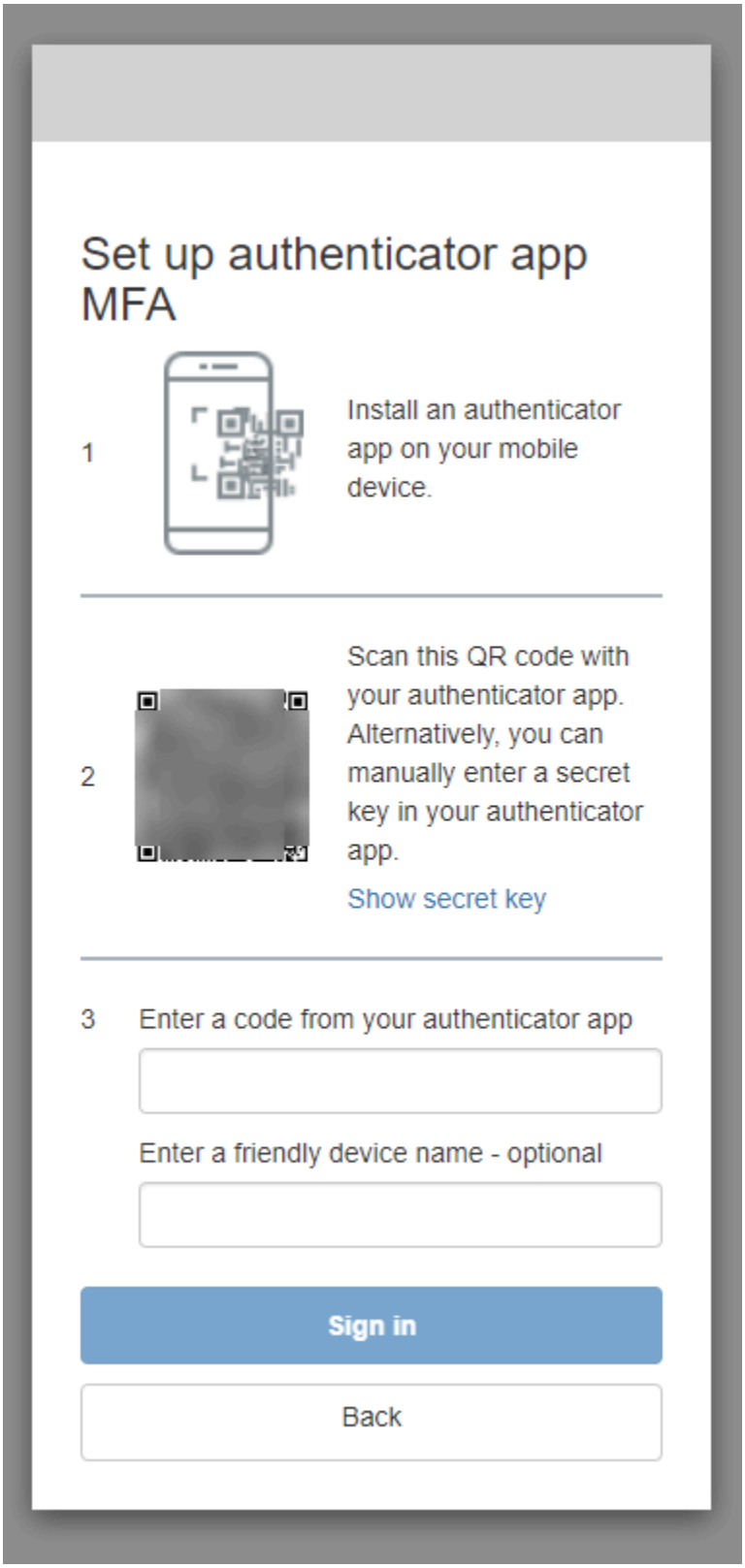
5. Amazon Cognito에서 인증 앱 또는 SMS 문자 메시지에서 코드를 입력하라는 메시지가 표시됩니다. 받은 코드를 3분 이내에 입력합니다.

   Authenticator app

   1. 다운로드한 인증 앱을 엽니다.
   2. 카메라로 페이지의 QR 코드를 스캔합니다. 카메라를 사용하려면 앱을 인증해야 할 수 있습니다.

QR 코드를 스캔할 수 없는 경우 인증 앱에 수동으로 입력할 수 있는 코드를 표시하려면 Show secret key(보안 키 표시)를 선택합니다.

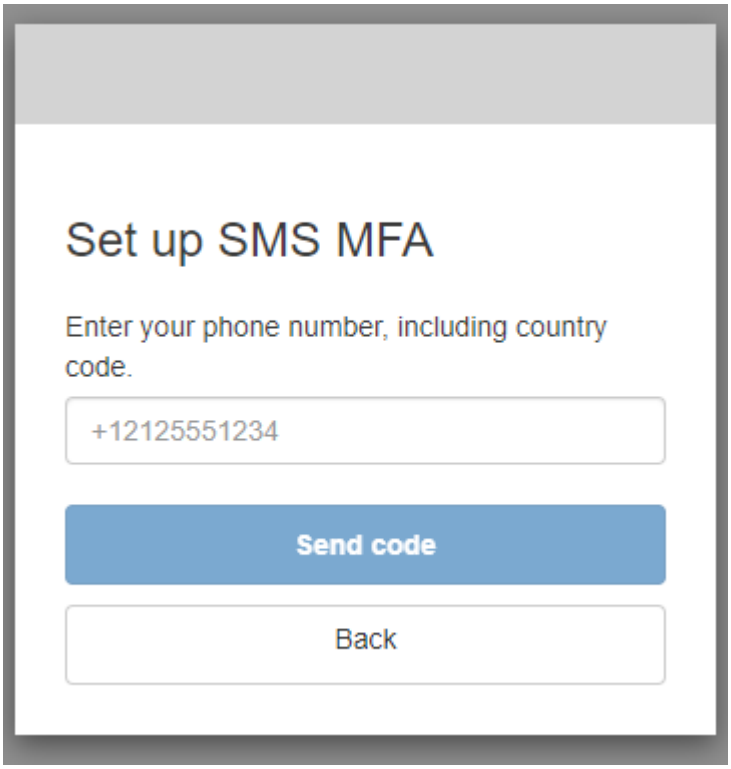
3. 인증 앱은 몇 초마다 변경되는 코드를 표시하기 시작합니다. 앱의 현재 코드를 입력합니다.
4. (선택 사항) Set up authenticator app MFA(인증 앱 MFA 설정) 페이지에서 디바이스 이름을 선택합니다. 로그인하면 Amazon Cognito에서 사용자가 여기에 지정한 이름으로 디바이스의 코드를 요청하게 됩니다.


Set up authenticator app MFA
1
Install an authenticator app on your mobile device.
2
Scan this QR code with your authenticator app. Alternatively, you can manually enter a secret key in your authenticator app.
Show secret key
3
Enter a code from your authenticator app
Enter a friendly device name - optional
Sign in
Back

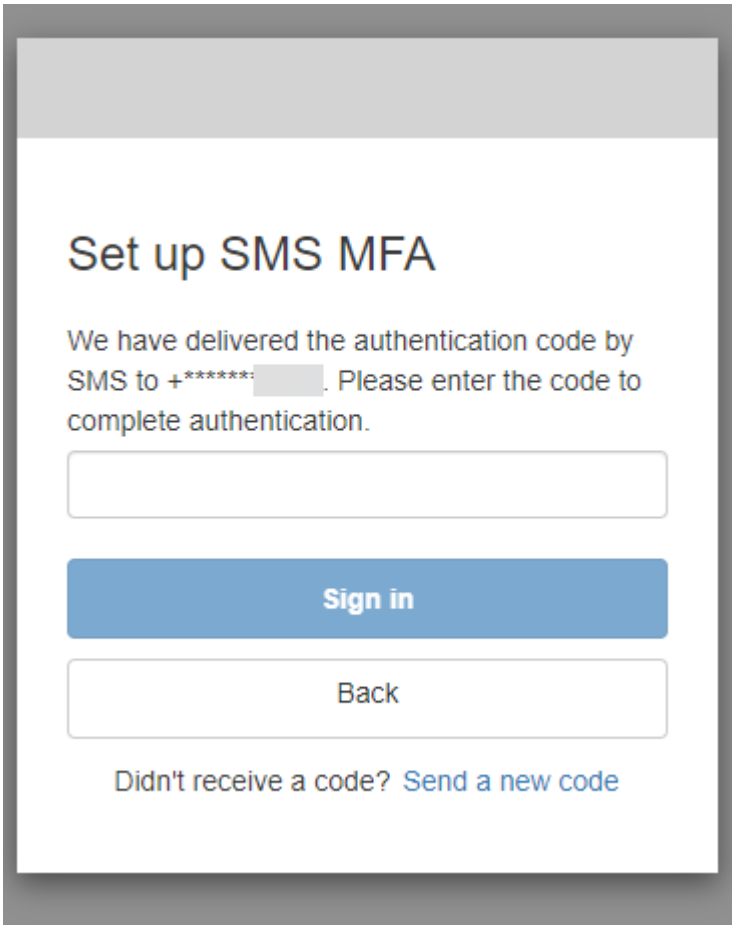
SMS text message

1. 앱 소유자가 아직 전화번호를 수집하지 않은 경우 Amazon Cognito에서 전화번호를 요청합니다.

   Set up SMS MFA(SMS MFA 설정) 페이지에서 + 기호와 국가 코드가 포함된 전화번호(예:+12125551234)를 입력합니다.



2. 코드가 포함된 SMS 메시지를 받게 됩니다. Set up SMS MFA(SMS MFA 설정) 페이지에서 코드를 입력합니다. 코드를 받지 못했는데 다시 시도하려면 Send a new code(새 코드 보내기)를 선택하세요. Back(뒤로)을 선택하여 새 전화번호를 입력합니다.



6. 처음 가입하고 세부 정보를 확인할 때 Amazon Cognito는 이 프로세스를 완료한 후 앱에 대한 액세스 권한을 부여합니다.

Amazon Cognito 호스팅 UI에 로그인하는 방법

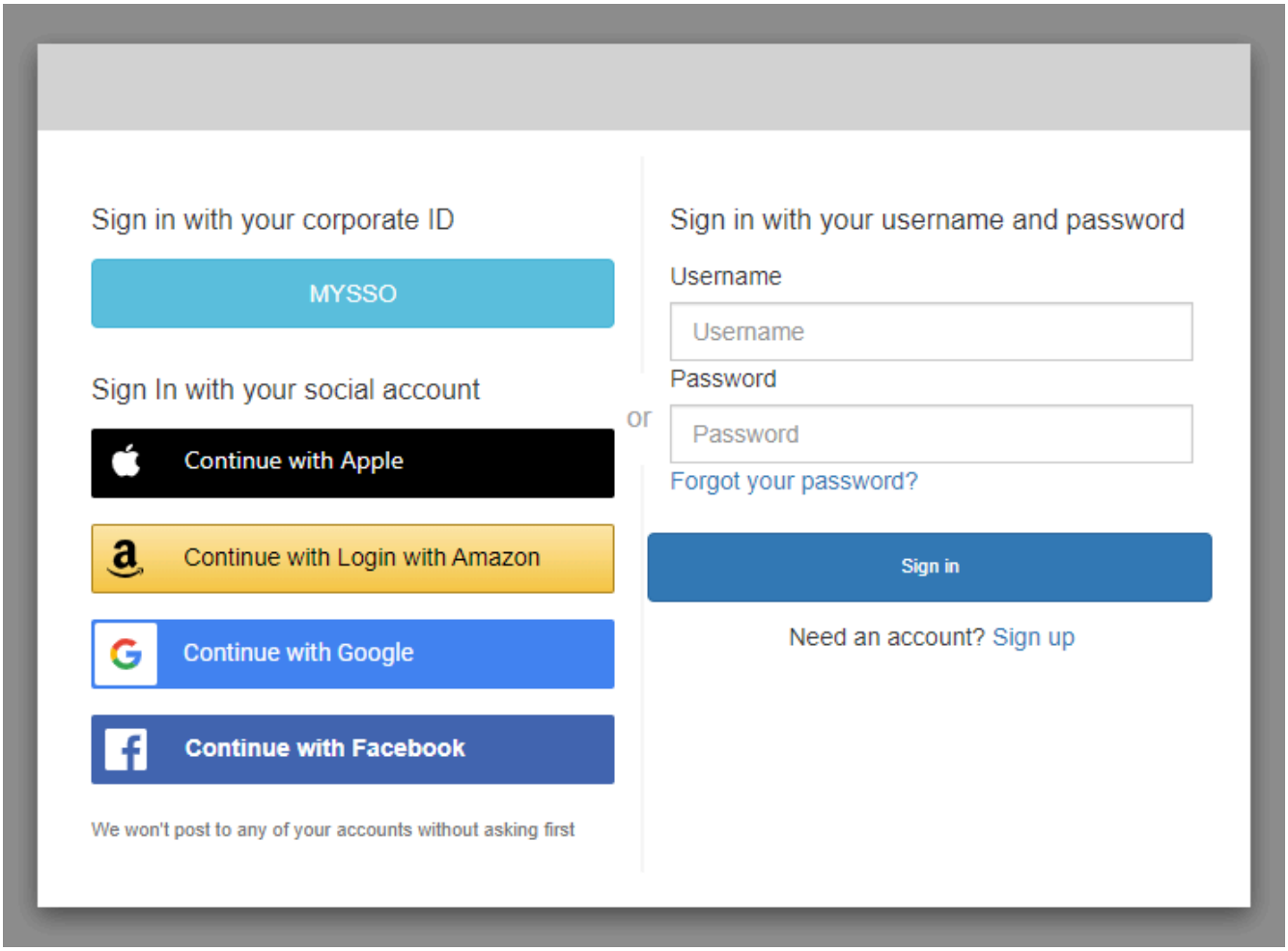
이 안내서에서는 Amazon Cognito를 사용하는 앱에 로그인하는 방법을 보여줍니다.

> **Note**
>
> Amazon Cognito 호스팅 사용자 인터페이스(UI)를 사용하는 앱에 로그인하면 이 안내서에 표시된 기본 구성 이상의 내용인 앱 소유자가 지정한 페이지에 표시될 수 있습니다.

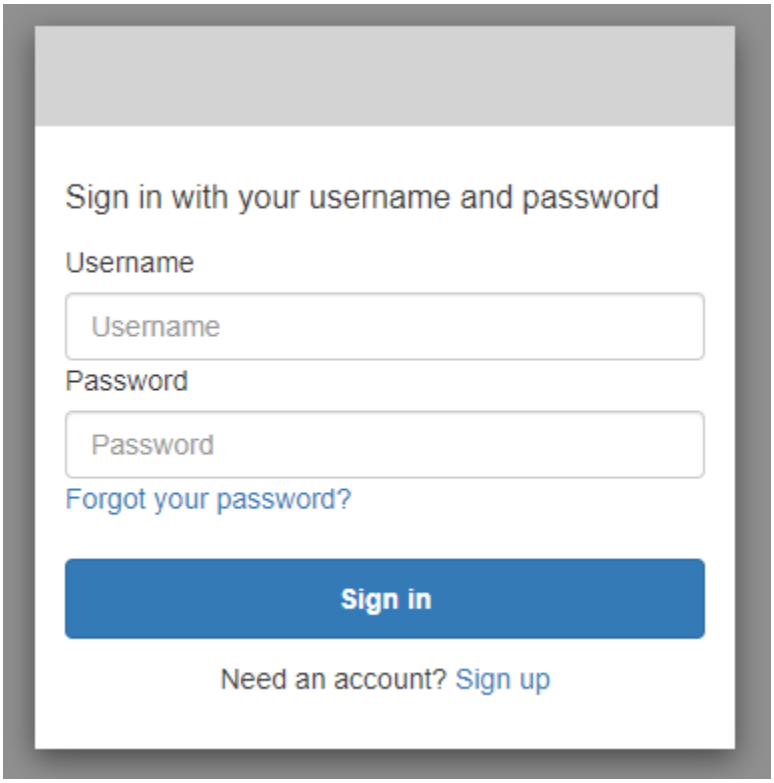
1. 앱 소유자가 선택한 옵션에 따라 로그인할 제공업체를 선택하거나 사용자 이름과 암호를 묻는 메시지만 표시될 수 있습니다. 이 페이지에서 사용자 이름과 암호로 로그인하면 Amazon Cognito가 로그인 서비스 제공업체입니다. 그렇지 않으면 선택한 버튼에 로그인 제공업체가 표시됩니다.

여기에서 제공업체를 선택하거나 사용자 이름과 암호를 입력하고 즉시 앱에 액세스할 수 있습니다. Amazon Cognito가 로그인 제공업체인 경우에는 앱 소유자에게도 다중 인증을 요구할 수 있습니다.

With multiple sign-in providers

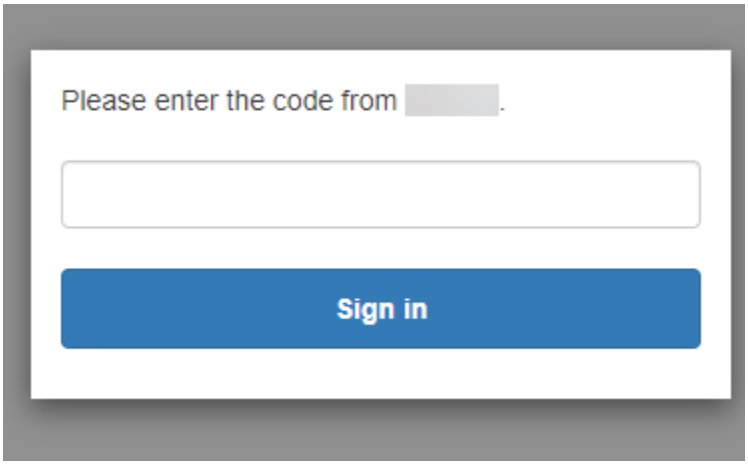


With only Amazon Cognito as a sign-in provider

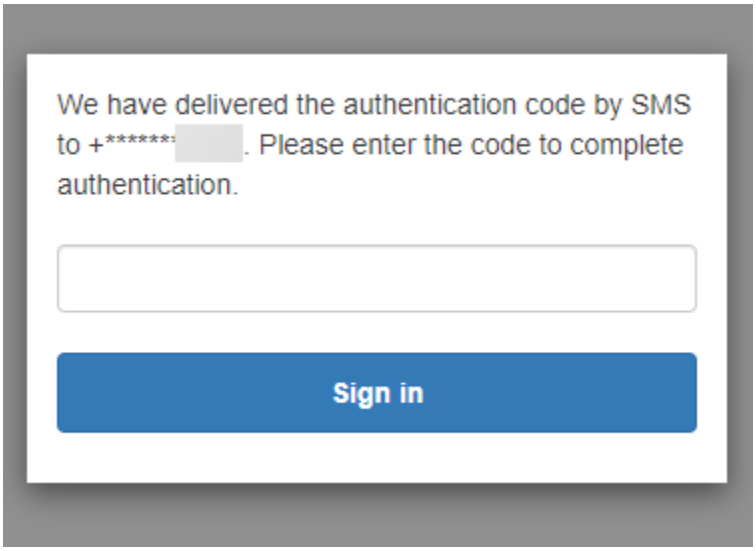


2. MFA는 앱에서 가입할 때 설정했을 수 있습니다. SMS 메시지로 받았거나 인증 앱에 표시된 MFA 코드를 입력합니다. 이 코드는 3분 이내에 입력해야 합니다.

With an authenticator app



With an SMS code



3. 로그인하고 MFA를 완료하면 Amazon Cognito에서 앱에 대한 액세스 권한을 부여합니다.

Amazon Cognito 호스팅 UI에서 암호를 재설정하는 방법

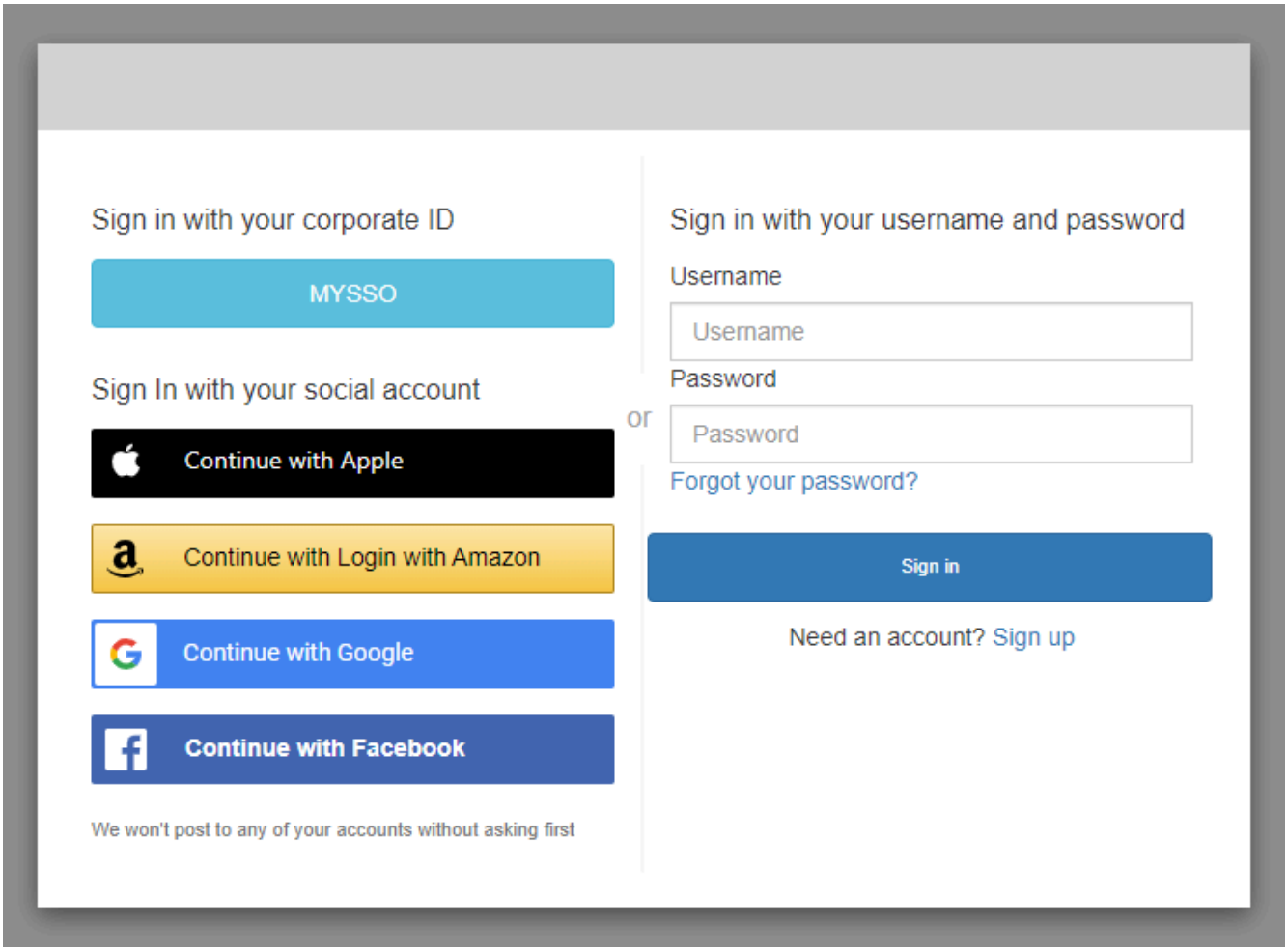
이 안내서에서는 Amazon Cognito를 사용하는 앱에서 암호를 재설정하는 방법을 보여줍니다.

> **Note**
>
> Amazon Cognito 호스팅 사용자 인터페이스(UI)를 사용하는 앱에 로그인하면 이 안내서에 표시된 기본 구성 이상의 내용인 앱 소유자가 지정한 페이지에 표시될 수 있습니다.

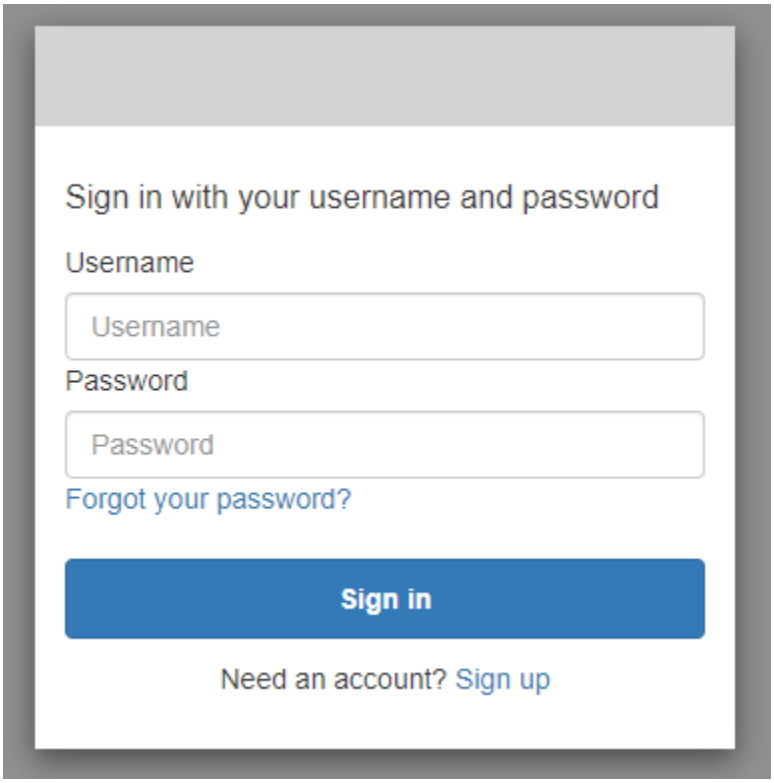
1. 앱 소유자가 선택한 옵션에 따라 로그인할 제공업체를 선택하거나 사용자 이름과 암호를 묻는 메시지만 표시될 수 있습니다. 이 페이지에서 사용자 이름과 암호로 로그인하면 Amazon Cognito가 로그인 서비스 제공업체입니다. 그렇지 않으면 선택한 버튼에 로그인 제공업체가 표시됩니다.

   일반적으로 로그인 페이지에서 제공업체를 선택하고 암호가 작동하지 않는 경우 해당 절차에 따라 제공업체를 통해 암호를 재설정합니다. Amazon Cognito가 로그인 제공업체인 경우 Forgot your password?(암호를 잊으셨습니까?)를 선택합니다.

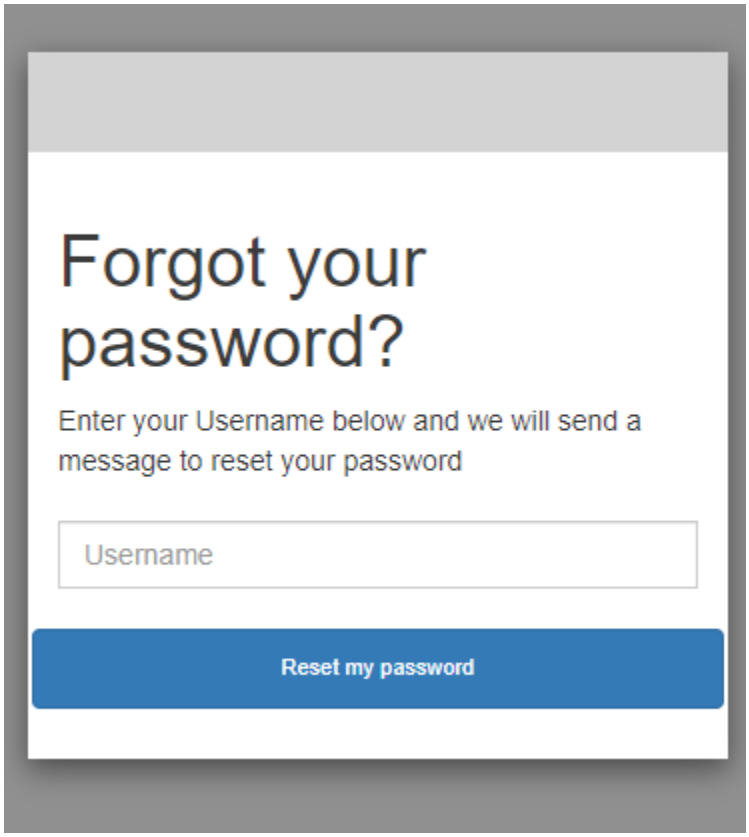
With multiple sign-in providers



With only Amazon Cognito as a sign-in provider

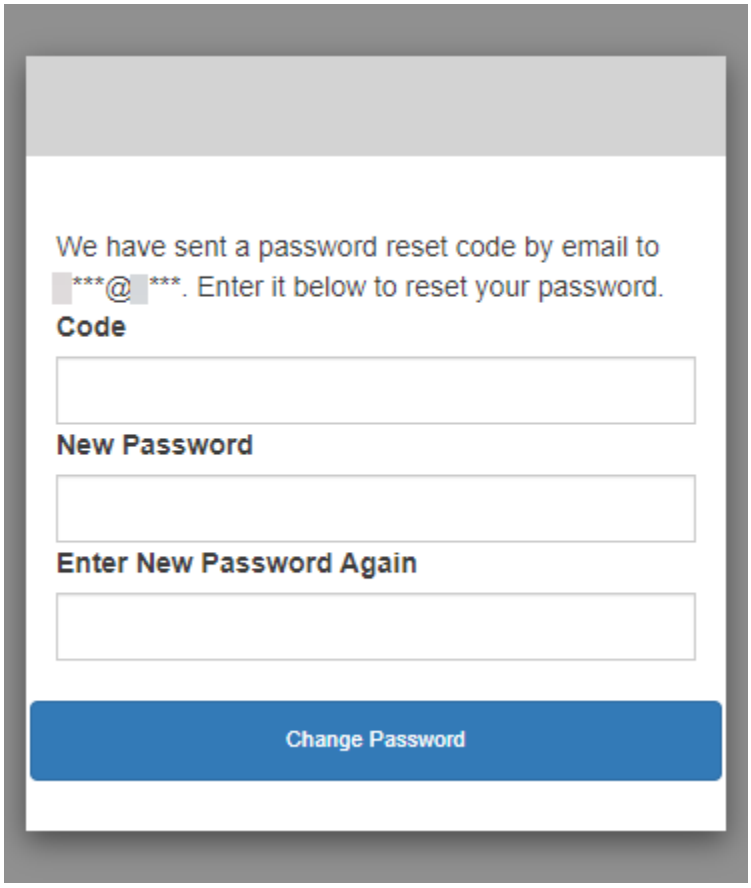


2. Forgot your password?(암호를 잊으셨습니까?) 페이지에서 Amazon Cognito는 로그인에 사용하는 정보를 묻는 메시지가 표시됩니다. 이러한 정보는 사용자 이름, 이메일 주소 또는 전화번호일 수 있습니다.



3. Amazon Cognito는 이메일 메시지 또는 SMS 문자 메시지로 코드를 전송합니다.

   받은 코드를 입력하고 제공된 필드에 새 암호를 두 번 입력합니다. 8분 이내에 재설정 코드를 입력해야 합니다.



4. 암호를 변경한 후 로그인 페이지로 돌아가 새 암호로 로그인합니다.

리소스 서버를 통한 범위, M2M 및 API 인증

사용자 풀에 대한 도메인을 구성한 후 Amazon Cognito는 OAuth 2.0 권한 부여 서버와 앱에서 사용자에게 표시할 수 있는 등록 및 로그인 페이지가 포함된 호스팅 웹 UI를 자동으로 프로비저닝합니다. 자세한 내용은 호스팅된 UI가 있는 앱 클라이언트 추가단원을 참조하세요. 권한 부여 서버가 액세스 토큰에 추가할 범위를 선택할 수 있습니다. 범위는 리소스 서버 및 사용자 데이터에 대한 액세스 권한을 부여합니다.

리소스 서버는 OAuth 2.0 API 서버입니다. 액세스 보호 리소스의 보안을 위해 이 서버는 보호하는 API에서 요청된 메서드와 경로를 승인하는 범위가 사용자 풀 액세스 토큰에 포함되어 있는지 확인합니다. 또한 토큰 서명, 토큰 만료 시간에 따른 유효성 및 토큰 클레임의 범위에 따라 액세스 수준을 기반으로 발급자를 확인합니다. 사용자 풀 범위는 액세스 토큰 클레임에 포함됩니다. `scope` Amazon Cognito 액세스 토큰의 클레임에 대한 자세한 내용은 액세스 토큰 사용 섹션을 참조하세요.

Amazon Cognito를 사용하면 액세스 토큰의 범위가 외부 API 또는 사용자 속성에 대한 액세스를 승인할 수 있습니다. 로컬 사용자, 연동 사용자 또는 시스템 ID에 액세스 토큰을 발행할 수 있습니다.

M achine-to-machine (M2M) 인증

Amazon Cognito는 시스템 ID로 API 데이터에 액세스하는 애플리케이션을 지원합니다. 사용자 풀의 시스템 ID는 애플리케이션 서버에서 실행되고 원격 API에 연결되는 기밀 클라이언트입니다. 예약된 작업, 데이터 스트림 또는 자산 업데이트 등 사용자 개입 없이 운영이 이루어집니다. 이러한 클라이언트가 액세스 토큰으로 요청을 승인하면 M2M (Machine to Machine) 권한 부여를 수행합니다. M2M 인증에서는 공유 암호가 액세스 제어의 사용자 자격 증명을 대체합니다.

M2M 인증을 사용하여 API에 액세스하는 애플리케이션에는 클라이언트 ID와 클라이언트 비밀번호가 있어야 합니다. 사용자 풀에서 클라이언트 자격 증명 부여를 지원하는 앱 클라이언트를 빌드해야 합니다. 클라이언트 자격 증명을 지원하려면 앱 클라이언트에 클라이언트 암호가 있어야 하고 사용자 풀 도메인이 있어야 합니다. 이 흐름에서는 컴퓨터 ID가 에서 직접 액세스 토큰을 Token 엔드포인트 요청합니다. 클라이언트 자격 증명 부여를 위한 액세스 토큰에서 리소스 서버의 사용자 지정 범위만 승인할 수 있습니다. 앱 클라이언트 설정에 대한 자세한 내용은 을 참조하십시오. 사용자 풀 앱 클라이언트

클라이언트 자격 증명 부여의 액세스 토큰은 시스템 ID가 API에서 요청하도록 허용하려는 작업에 대한 검증 가능한 설명입니다. 액세스 토큰이 API 요청을 승인하는 방법에 대해 자세히 알아보려면 계속 읽어보세요. 예제 애플리케이션은 CDK를 사용한 AWS Amazon Cognito 및 API Gateway 기반 머신 간 인증을 참조하십시오.

M2M 인증에는 월간 활성 사용자 (MAU) 에게 청구되는 방식과 다른 청구 모델이 있습니다. 사용자 인증에는 활성 사용자당 비용이 부과되는 반면, M2M 청구에는 활성 클라이언트 자격 증명, 앱 클라이언트 및 총 토큰 요청 양이 반영됩니다. 자세한 내용은 Amazon Cognito 요금을 참조하세요. M2M 인증 비용을 관리하려면 액세스 토큰 기간과 애플리케이션이 보내는 토큰 요청 수를 최적화하세요. API Gateway 캐싱을 사용하여 M2M 인증에서 새 토큰 요청을 줄이는 방법은 을 참조하십시오캐싱 토큰.

청구서에 비용을 추가하는 Amazon Cognito 작업을 최적화하는 방법에 대한 자세한 내용은 을 AWS 참조하십시오. 비용 관리

범위에 대한 정보

범위는 앱이 리소스에 요청할 수 있는 액세스의 수준입니다. Amazon Cognito 액세스 토큰에서 범위는 사용자 풀에 설정한 신뢰, 즉 알려진 디지털 서명이 있는 신뢰할 수 있는 액세스 토큰 발급자에 의해 뒷받침됩니다. 사용자 풀은 고객이 자신의 사용자 프로필 중 일부 또는 전부를 관리하거나 백엔드 API에서 데이터를 검색할 수 있음을 입증하는 범위를 사용하여 액세스 토큰을 생성할 수 있습니다. Amazon Cognito 사용자 풀은 사용자 풀에 예약된 API 범위, 사용자 지정 범위 및 표준 범위를 사용하여 액세스 토큰을 발급합니다.

사용자 풀의 예약된 API 범위

aws.cognito.signin.user.admin은 Amazon Cognito 사용자 풀 API에 권한을 부여합니다. 액세스 토큰 보유자에게 사용자 풀 사용자에 대한 모든 정보 (예: 및 API 작업) 를 쿼리하고 업데이트할 수 있는 권한을 부여합니다. GetUserUpdateUserAttributes 이 범위는 Amazon Cognito 사용자 풀 API로 사용자를 인증할 때 액세스 토큰에서 수신하는 유일한 범위입니다. 또한 앱 클라이언트에 읽기 및 쓰기 권한을 부여한 사용자 속성을 읽고 쓰는 데 필요한 유일한 범위이기도 합니다. 요청 시 이 범위를 권한 부여 엔드포인트에 요청할 수도 있습니다. 이 범위만으로는 UserInfo 엔드포인트에서 사용자 속성을 요청하기에 충분하지 않습니다. 사용자 풀 API 및 사용자에 대한 userInfo 요청을 모두 승인하는 액세스 토큰의 경우 /oauth2/authorize 요청 시 범위 openid 및 aws.cognito.signin.user.admin을 모두 요청해야 합니다.

사용자 지정 범위

사용자 지정 범위는 리소스 서버가 보호하는 외부 API에 대한 요청을 승인합니다. 다른 유형의 범위를 사용하여 사용자 지정 범위를 요청할 수 있습니다. 이 페이지 전반에서 사용자 지정 범위에 대한 자세한 정보를 찾을 수 있습니다.

표준 범위

호스팅 UI를 포함하여 사용자 풀 OAuth 2.0 권한 부여 서버로 사용자를 인증하는 경우 범위를 요청해야 합니다. Amazon Cognito 권한 부여 서버에서 사용자 풀 로컬 사용자와 서드 파티 페더레이션 사용자를 인증할 수 있습니다. 표준 OAuth 2.0 범위는 앱이 사용자 풀의 UserInfo 엔드포인트에서 사용자 정보를 읽을 수 있는 권한을 부여합니다. userInfo 엔드포인트에서 사용자 속성을 쿼리하는 OAuth 모델은 사용자 속성에 대한 대량의 요청에 맞게 앱을 최적화할 수 있습니다. userInfo 엔드포인트는 액세스 토큰의 범위에 따라 결정되는 권한 수준의 속성을 반환합니다. 앱 클라이언트가 다음과 같은 표준 OAuth 2.0 범위를 사용하여 액세스 토큰을 발급하도록 권한을 부여할 수 있습니다.

openid

OpenID Connect(OIDC) 쿼리의 필수 범위입니다. ID 토큰, 고유 식별자 클레임 sub 및 다른 범위를 요청하는 기능을 승인합니다.

> **Note**
>
> openid 범위를 요청하고 다른 범위를 요청하지 않은 경우 사용자 풀 ID 토큰과 userInfo 응답에는 앱 클라이언트가 읽을 수 있는 모든 사용자 속성에 대한 클레임이 포함됩니다. 요청할 때 openid 및 profile, email, phone 등 기타 표준 범위(예: ID 토큰 및 userInfo 응답)는 추가 범위의 제약 조건으로 제한됩니다.
> 예를 들어, 파라미터와 권한 부여 엔드포인트를 함께 요청하면 sub, email, email_verified 등이 포함된 ID 토큰이 scope=openid+email에서 반환됩니다.
> 이 요청의 액세스 토큰은 UserInfo 엔드포인트와 동일한 속성을 반환합니다. 파라미터

`scope=openid`가 있는 요청은 ID 토큰 및 원본 `userInfo`에서 클라이언트가 읽을 수 있는 모든 속성을 반환합니다.

profile

앱 클라이언트가 읽을 수 있는 모든 사용자 속성을 승인합니다.

이메일

사용자 속성 `email` 및 `email_verified`를 승인합니다. Amazon Cognito는 값이 명시적으로 설정된 경우 `email_verified`를 반환합니다.

phone

사용자 속성 `phone_number` 및 `phone_number_verified`를 승인합니다.

리소스 서버에 대한 정보

리소스 서버 API는 데이터베이스의 정보에 대한 액세스 권한을 부여하거나 IT 리소스를 제어할 수 있습니다. Amazon Cognito 액세스 토큰은 OAuth 2.0을 지원하는 API에 대한 액세스 권한을 부여할 수 있습니다. Amazon API Gateway REST API에는 Amazon Cognito 액세스 토큰을 통한 권한 부여 지원 기능이 내장되어 있습니다. 앱은 API 호출 시 액세스 토큰을 리소스 서버로 전달합니다. 리소스 서버는 액세스 토큰을 검사하여 액세스 권한을 부여할지 여부를 결정합니다.

Amazon Cognito는 향후 사용자 풀 액세스 토큰의 스키마를 업데이트할 수 있습니다. 앱이 액세스 토큰을 API로 전달하기 전에 콘텐츠를 분석하는 경우 스키마 업데이트를 수락하도록 코드를 엔지니어링해야 합니다.

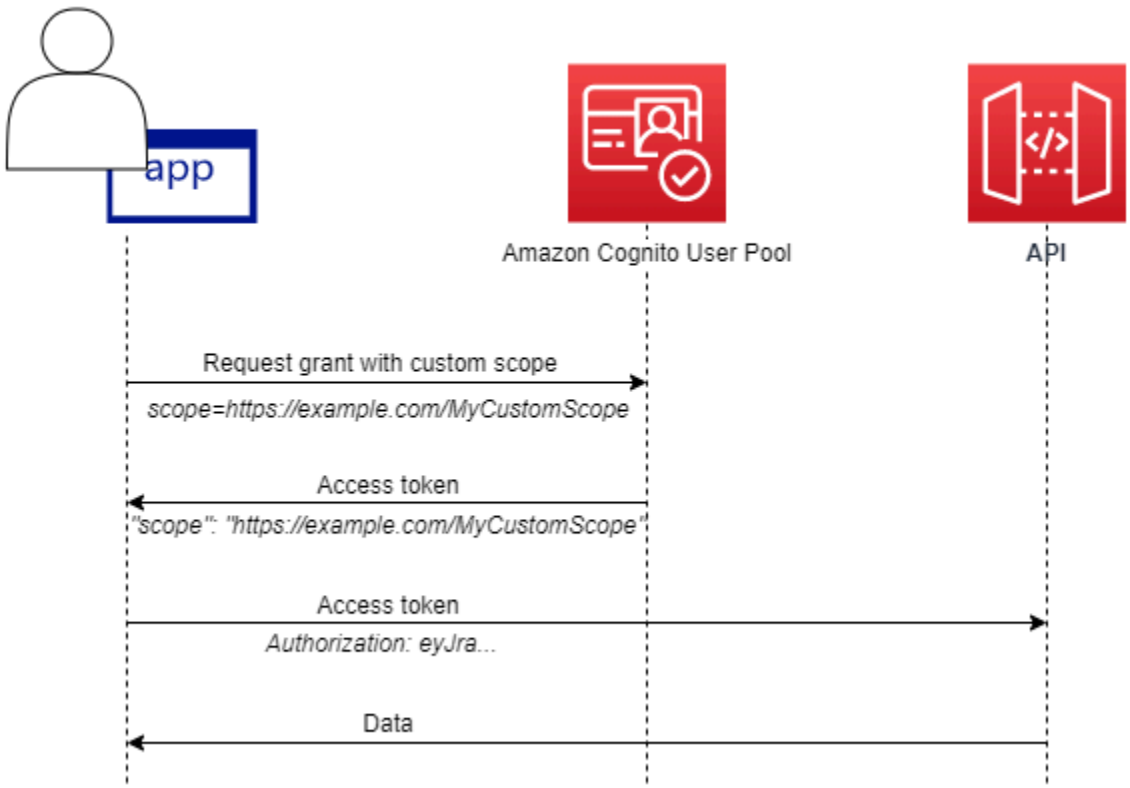
사용자 지정 범위는 사용자가 정의하며, 사용자 풀의 권한 부여 기능을 확대하여 사용자 및 해당 속성의 쿼리 및 수정과 관련이 없는 용도까지 포함합니다. 예를 들어 사진에 대한 리소스 서버를 가지고 있는 경우에는 두 개의 범위를 정의하게 되는데, `photos.read`는 사진에 대한 읽기 액세스를 위한 범위이고 `photos.write`는 쓰기/삭제 액세스를 위한 범위입니다. 권한 부여를 위해 액세스 토큰을 수락하도록 API를 구성하고, `scope` 클레임에 `photos.read`가 있는 액세스 토큰에 대한 `HTTP GET` 요청과 `photos.write`가 있는 토큰에 대한 `HTTP POST` 요청을 허용할 수 있습니다. 이들은 사용자 지정 범위입니다.

> **Note**
>
> 리소스 서버는 토큰 내에서 클레임을 처리하기 앞서 액세스 토큰 서명과 만료 날짜를 확인해야 합니다. 토큰 확인에 대한 자세한 내용은 JSON 웹 토큰 확인 섹션을 참조하세요. Amazon API

Gateway에서 사용자 풀 토큰을 확인하고 사용하는 방법에 대한 자세한 내용은 API Gateway와 Amazon Cognito 사용자 풀 통합을 참조하세요. API Gateway는 액세스 토큰을 검사하여 리소스를 보호하는 데 적합한 옵션입니다. API Gateway Lambda 권한 부여자에 대한 자세한 내용은 API Gateway Lambda 권한 부여자 사용을 참조하세요.

개요

Amazon Cognito를 사용하여 OAuth 2.0 리소스 서버를 생성하고 사용자 지정 범위를 리소스 서버와 연결할 수 있습니다. 액세스 토큰의 사용자 지정 범위는 API의 특정 작업을 승인합니다. 사용자 풀의 모든 앱 클라이언트가 모든 리소스 서버에서 사용자 지정 범위를 발급하도록 권한을 부여할 수 있습니다. 사용자 지정 범위를 앱 클라이언트와 연결하고 Token 엔드포인트에서 OAuth 2.0 인증 코드 권한 부여, 암시적 권한 부여 및 클라이언트 보안 인증 권한 부여에서 해당 범위를 요청할 수 있습니다. Amazon Cognito는 액세스 토큰의 `scope` 클레임에 사용자 지정 범위를 추가합니다. 클라이언트는 리소스 서버에 대해 액세스 토큰을 사용할 수 있습니다. 리소스 서버는 토큰에 있는 범위를 기준으로 권한 부여를 결정합니다. 액세스 토큰 범위에 대한 자세한 내용은 사용자 풀을 통해 토큰 사용을 참조하세요.



사용자 지정 범위가 있는 액세스 토큰을 얻으려면 앱에서 Token 엔드포인트에 요청하여 인증 코드를 사용하거나 클라이언트 자격 증명 권한 부여를 요청해야 합니다. 호스팅 UI에서 암시적 권한 부여에서 액세스 토큰의 사용자 지정 범위를 요청할 수도 있습니다.

> **Note**
>
> 사용자 풀을 IdP로 사용하는 사용자 대화형 인증을 위해 설계되었기 때문에 AdminInitiateAuth요청은 단일 값을 가진 액세스 토큰에 대한 `scope` 클레임만 생성합니다. InitiateAuth`aws.cognito.signin.user.admin`

리소스 서버 및 사용자 지정 범위 관리

리소스 서버를 생성할 때 반드시 리소스 서버 이름과 리소스 서버 식별자를 제공해야 합니다. 리소스 서버에서 생성한 각 범위에 대해 범위 이름과 설명을 입력해야 합니다.

- 리소스 서버 이름: `Solar system object tracker` 또는 `Photo API` 같은 기억하기 쉬운 리소스 서버 이름입니다.
- 리소스 서버 식별자: 리소스 서버의 고유 식별자입니다. 식별자는 API와 연결할 이름(예: `solar-system-data`)입니다. 예를 들어 API URI 경로에 대한 보다 직접적인 참조로 `https://solar-system-data-api.example.com` 같은 더 긴 식별자를 구성할 수 있지만 문자열이 길면 액세스 토큰의 크기가 커집니다.
- 범위 이름: `scope` 클레임에서 사용할 값입니다. 예를 들어 `sunproximity.read`입니다.
- 설명: 범위에 대한 알기 쉬운 설명입니다. 예를 들어 `Check current proximity to sun`입니다.

Amazon Cognito는 로컬 사용자 풀의 사용자이든 서드 파티 ID 제공업체와의 페더레이션 사용자이든, 모든 사용자의 액세스 토큰에 사용자 지정 범위를 포함할 수 있습니다. 호스팅 UI가 포함된 OAuth 2.0 권한 부여 서버를 사용하여 인증 흐름 중에 사용자 액세스 토큰의 범위를 선택할 수 있습니다. 사용자 인증은 권한 부여 엔드포인트에서 `scope`를 요청 파라미터 중 하나로 사용하여 시작해야 합니다. 다음은 리소스 서버의 권장 형식입니다. 식별자에는 API 친화적인 이름을 사용합니다. 그리고 사용자 지정 범위에는 해당 범위가 승인하는 작업을 사용합니다.

```
resourceServerIdentifier/scopeName
```

예를 들어, 카이퍼 벨트에서 새로운 소행성을 발견했고 이를 `solar-system-data` API를 통해 등록하려고 합니다. 소행성 데이터베이스에 대한 쓰기 작업을 승인하는 범위는 `asteroids.add`입니다.

검색을 등록할 수 있는 권한을 부여하는 액세스 토큰을 요청하는 경우 `scope` HTTPS 요청 파라미터의 형식을 `scope=solar-system-data/asteroids.add`로 지정합니다.

리소스 서버에서 범위를 삭제해도 모든 클라이언트와의 연결이 삭제되지는 않습니다. 대신, 범위를 삭제하면 범위가 inactive로 표시됩니다. Amazon Cognito는 액세스 토큰에 비활성 범위를 추가하지 않지만 앱에서 요청할 경우 정상적으로 진행합니다. 나중에 리소스 서버에 범위를 다시 추가하면 Amazon Cognito가 해당 범위를 액세스 토큰에 다시 기록합니다. 앱 클라이언트와 연결하지 않은 범위를 요청하면 사용자 풀 리소스 서버에서 해당 범위를 삭제했는지 여부에 관계없이 인증이 실패합니다.

AWS Management Console, API 또는 CLI를 사용하여 사용자 풀의 리소스 서버 및 범위를 정의할 수 있습니다.

사용자 풀의 리소스 서버 정의(AWS Management Console)

를 사용하여 사용자 풀의 AWS Management Console 리소스 서버를 정의할 수 있습니다.

리소스 서버를 정의하려면

1. Amazon Cognito 콘솔에 로그인합니다.
2. 탐색 창에서 [사용자 풀(User Pools)]을 선택한 다음 편집할 사용자 풀을 선택합니다.
3. [앱 통합(App integration) 탭을 선택하고 [리소스 서버(Resource servers)]를 찾습니다.
4. [리소스 서버 생성(Create a resource server)]을 선택합니다.
5. [리소스 서버 이름(Resource server name)]을 입력합니다. 예를 들어 `Photo Server`입니다.
6. [리소스 서버 식별자(Resource server identifier)]를 입력합니다. 예를 들어 `com.example.photos`입니다.
7. 리소스에 대한 [사용자 정의 범위(Custom scopes)]를 입력합니다(예: `read`, `write`).
8. 각 [범위 이름(Scope name)]에 대한 [설명(Description)]을 입력합니다(예: `view your photos`, `update your photos`).
9. 생성을 선택합니다.

사용자 정의 범위는 [앱 통합(App integration)] 탭의 [리소스 서버(Resource servers)] 아래에 있는 [사용자 정의 범위(Custom scopes)] 열에서 검토할 수 있습니다. [앱 통합(App integration)] 탭의 [앱 클라이언트(App clients)]에서 앱 클라이언트에 사용자 정의 범위를 사용하도록 설정할 수 있습니다. 앱 클라이언트를 선택하고 [호스트된 UI 설정(Hosted UI settings)]을 찾은 다음, [편집(Edit)]을 선택합니다. [사용자 정의 범위(Custom scopes)]를 추가하고 [변경 사항 저장(Save changes)]을 선택합니다.

사용자 풀 (AWS CLI 및 AWS API) 에 대한 리소스 서버 정의

다음 명령을 사용하여 사용자 풀에 대한 리소스 서버 설정을 지정합니다.

리소스 서버를 생성하려면

- AWS CLI: `aws cognito-idp create-resource-server`
- AWS API: CreateResourceServer

리소스 서버 설정에 대한 정보를 가져오려면

- AWS CLI: `aws cognito-idp describe-resource-server`
- AWS API: DescribeResourceServer

사용자 풀의 모든 리소스 서버에 대한 정보를 나열하려면

- AWS CLI: `aws cognito-idp list-resource-servers`
- AWS API: ListResourceServers

리소스 서버를 삭제하려면

- AWS CLI: `aws cognito-idp delete-resource-server`
- AWS API: DeleteResourceServer

리소스 서버의 설정을 업데이트하려면

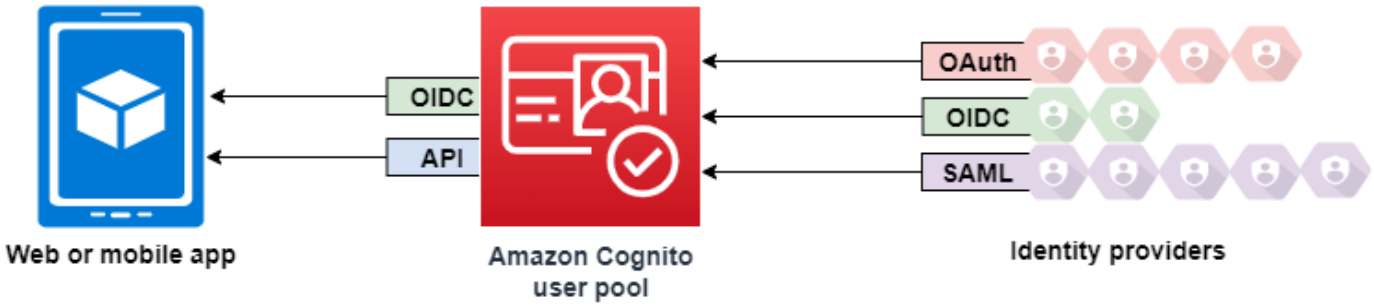
- AWS CLI: `aws cognito-idp update-resource-server`
- AWS API: UpdateResourceServer

서드 파티를 통한 사용자 풀 로그인 추가

앱 사용자는 사용자 풀을 통해 직접 로그인하거나 타사 ID 공급자 (IdP) 를 통해 페더레이션할 수 있습니다. 사용자 풀은 페이스북, 구글, 아마존, 애플을 통한 소셜 로그인과 OpenID Connect (OIDC) 및 SAML에서 반환되는 토큰을 처리하는 오버헤드를 관리합니다. IdPs 호스팅된 웹 UI가 내장되어 있는 Amazon Cognito는 모든 인증된 사용자를 위한 토큰 처리 및 관리 기능을 제공합니다. IdPs 이렇게 하면 백엔드 시스템을 한 세트의 사용자 풀 토큰에서 표준화할 수 있습니다.

Amazon Cognito 사용자 풀에서 페더레이션 로그인이 작동하는 방식

서드 파티(페더레이션)를 통한 로그인을 Amazon Cognito 사용자 풀에서 사용할 수 있습니다. 이 기능은 Amazon Cognito 자격 증명 풀(페더레이션 자격 증명)을 통한 페더레이션과 무관합니다.



Amazon Cognito는 사용자 디렉터리이자 OAuth 2.0 아이덴티티 제공업체(IdP)입니다. Amazon Cognito 디렉터리에 로컬 사용자를 로그인할 때 사용자 풀은 앱의 IdP입니다. 로컬 사용자는 외부 IdP를 통한 페더레이션 없이 사용자 풀 디렉터리에만 존재합니다.

Amazon Cognito를 소셜, SAML 또는 OpenID Connect (OIDC IdPs) 에 연결하면 사용자 풀이 여러 서비스 공급자와 앱을 연결하는 다리 역할을 합니다. IdP에 있어 Amazon Cognito는 서비스 제공업체(SP)입니다. OIDC ID 토큰 또는 SAML 어설션을 Amazon Cognito에 IdPs 전달합니다. Amazon Cognito는 토큰 또는 어설션에서 사용자에 대한 클레임을 읽고 이러한 클레임을 사용자 풀 디렉터리의 새 사용자 프로필에 매핑합니다.

그런 다음 Amazon Cognito는 페더레이션 사용자에 대한 사용자 프로필을 자체 디렉터리에 생성합니다. Amazon Cognito는 IdP, 그리고 OIDC와 소셜 아이덴티티 제공업체의 경우 IdP가 운영하는 퍼블릭 `userinfo` 엔드포인트의 클레임을 기반으로 사용자에게 속성을 추가합니다. 매핑된 IdP 속성이 변경되면 사용자 풀에서 사용자 속성이 변경됩니다. IdP와 별개로 속성을 더 추가할 수도 있습니다.

Amazon Cognito가 페더레이션 사용자 프로필을 생성한 후에는 해당 함수가 변경되고 앱의 IdP로 표시되어 SP가 됩니다. Amazon Cognito는 OIDC와 OAuth 2.0 IdP의 조합입니다. 액세스 토큰, ID 토큰 및 새로 고침 토큰을 생성합니다. 토큰에 대한 자세한 내용은 사용자 풀에 토큰 사용 섹션을 참조하세요.

Amazon Cognito와 통합되는 앱을 설계하여 페더레이션 사용자든 로컬 사용자든 상관없이 사용자를 인증하고 권한을 부여해야 합니다.

Amazon Cognito에서 서비스 공급자로서 앱의 책임

토큰의 정보 확인 및 처리

대부분의 시나리오에서 Amazon Cognito는 인증된 사용자를 인증 코드와 함께 추가한 앱 URL로 리디렉션합니다. 앱에서는 액세스, ID 및 새로 고침 토큰용 코드를 교환합니다. 그런 다음 토큰의 유효성을 확인하고 토큰의 클레임을 기반으로 사용자에게 정보를 제공해야 합니다.

Amazon Cognito API 요청으로 인증 이벤트에 응답

앱은 Amazon Cognito 사용자 풀 API 및 인증 API 엔드포인트와 통합되어야 합니다. 인증 API는 사용자를 로그인 및 로그아웃하고 토큰을 관리합니다. 사용자 풀 API에는 사용자 풀, 사용자 및 인증 환경의 보안을 관리하는 다양한 작업이 있습니다. 앱은 Amazon Cognito로부터 응답을 받을 때 다음에 수행할 작업을 알아야 합니다.

Amazon Cognito 사용자 풀 타사 로그인에 대해 알아야 할 사항

- 사용자가 페더레이션 공급자로 로그인하도록 하려면 도메인을 선택해야 합니다. 이렇게 하면 Amazon Cognito 호스팅 UI 및 호스팅된 UI와 OIDC 엔드포인트가 설정됩니다. 자세한 정보는 호스트된 UI에 고유한 도메인 사용을 참조하세요.
- 및 같은 API 작업으로는 페더레이션 사용자를 로그인할 수 없습니다. InitiateAuthAdminInitiateAuth 페더레이션 사용자는 Login 엔드포인트 또는 권한 부여 엔드포인트를 사용해서만 로그인할 수 있습니다.
- 권한 부여 엔드포인트는 리디렉션 엔드포인트입니다. 요청에 `idp_identifier` 또는 `identity_provider` 파라미터를 입력하면 호스팅된 UI를 우회하여 자동으로 IdP로 리디렉션됩니다. 그렇지 않으면 호스팅된 UI Login 엔드포인트로 리디렉션됩니다. 예시는 예제 시나리오: 엔터프라이즈 대시보드에서 Amazon Cognito 앱을 북마크하기단원을 참조하세요.
- 호스팅 UI가 세션을 페더레이션 IdP로 리디렉션하면 Amazon Cognito는 요청에 `user-agent` 헤더인 `Amazon/Cognito`를 포함합니다.
- Amazon Cognito는 고정 식별자와 IdP의 이름 조합에서 페더레이션 사용자 프로필의 `username` 속성을 추출합니다. 사용자 지정 요구 사항에 맞는 사용자 이름을 생성하려면 `preferred_username` 속성에 대한 매핑을 만듭니다. 자세한 정보는 매핑에 대해 알아야 할 사항을 참조하세요.

  예제: `MyIDP_bob@example.com`
- Amazon Cognito는 페더레이션 사용자의 ID 정보를 속성과 `identities`라는 ID 토큰의 클레임에 기록합니다. 이 클레임에는 사용자의 제공업체와 제공업체의 고유 ID가 포함됩니다. 사용자 프로필

에서 `identities` 속성을 직접 변경할 수 없습니다. 페더레이션 사용자를 연결하는 방법에 대한 자세한 내용은 페더레이션 사용자를 기존 사용자 프로필에 연결 섹션을 참조하세요.

- UpdateIdentityProvider API 요청에서 IdP를 업데이트하면, 변경 사항이 호스팅된 UI에 표시되기까지 최대 1분이 걸립니다.
- Amazon Cognito는 Amazon Cognito와 사용자 IdP 간의 HTTP 리디렉션을 20개까지 지원합니다.
- 사용자가 호스팅 UI로 로그인하면 브라우저는 로그인에 사용한 클라이언트 및 공급자를 기록하는 암호화된 로그인 세션 쿠키를 저장합니다. 동일한 파라미터로 다시 로그인을 시도하면 호스팅 UI는 만료되지 않은 기존 세션을 재사용하며, 사용자는 보안 인증을 다시 제공하지 않고 인증합니다. 사용자는 다른 IdP로 다시 로그인할 경우(로컬 사용자 풀 로그인으로 전환하거나 로컬 사용자 풀 로그인에서 전환하는 경우 포함) 보안 인증을 제공하고 새 로그인 세션을 생성해야 합니다.

  모든 사용자 풀을 IdPs 모든 앱 클라이언트에 할당할 수 있으며, 사용자는 앱 클라이언트에 할당한 IdP로만 로그인할 수 있습니다.

주제

- 사용자 풀의 자격 증명 공급자 구성
- 사용자 풀이 있는 소셜 ID 공급자 사용
- 사용자 풀과 함께 SAML ID 공급자 사용
- 사용자 풀과 함께 OIDC ID 공급자 사용
- 사용자 풀에 대한 자격 증명 공급자 속성 매핑 지정
- 페더레이션 사용자를 기존 사용자 프로필에 연결

사용자 풀의 자격 증명 공급자 구성

페더레이션 ID 공급자 로그인 아래의 로그인 경험 탭에서 사용자 풀에 ID 공급자 (IdPs) 를 추가할 수 있습니다. 자세한 내용은 서드 파티를 통한 사용자 풀 로그인 추가 섹션을 참조하세요.

주제

- 소셜 IdP를 사용하여 사용자 로그인 설정
- OIDC IdP를 사용하여 사용자 로그인 설정
- SAML IdP를 사용하여 사용자 로그인 설정

소셜 IdP를 사용하여 사용자 로그인 설정

페더레이션을 사용하여 Amazon Cognito 사용자 풀을 Facebook, Google, Login with Amazon 등의 소셜 자격 증명 공급자와 통합할 수 있습니다.

소셜 자격 증명 공급자를 추가하려면 먼저 자격 증명 공급자에서 개발자 계정을 생성합니다. 개발자 계정이 생성되면 자격 증명 공급자에 앱을 등록합니다. 자격 증명 공급자가 앱에 대한 ID와 암호를 생성하면 Amazon Cognito 사용자 풀에 이들 값을 구성합니다.

- Google 자격 증명 플랫폼
- 개발자용 Facebook
- Login with Amazon
- Apple로 로그인

사용자 로그인을 소셜 IdP와 통합하려면

1. Amazon Cognito 콘솔에 로그인합니다. 메시지가 표시되면 AWS 자격 증명을 입력합니다.
2. 탐색 창에서 [사용자 풀(User Pools)]을 선택한 다음 편집할 사용자 풀을 선택합니다.
3. 로그인 환경(Sign-in experience) 탭을 선택하고 페더레이션 로그인(Federated sign-in)을 찾습니다.
4. 자격 증명 공급자 추가(Add an identity provider)를 선택하거나, 구성한 Facebook, Google, Amazon 또는 Apple 자격 증명 공급자를 선택하고 자격 증명 공급자 정보(Identity provider information)를 찾은 다음 편집(Edit)을 선택합니다. 소셜 자격 증명 공급자를 추가하는 방법에 대한 자세한 내용은 사용자 풀이 있는 소셜 ID 공급자 사용 섹션을 참조하세요.
5. 선택한 IdP에 따라 다음 단계 중 하나를 완료하여 소셜 자격 증명 공급자의 정보를 입력합니다.

   Facebook, Google 및 Login with Amazon

   클라이언트 앱을 생성할 때 수신한 앱 ID 및 앱 암호를 입력합니다.

   Apple로 로그인

   Apple에 제공한 서비스 ID와 앱 클라이언트를 생성할 때 수신한 팀 ID, 키 ID, 프라이빗 키를 입력합니다.
6. 권한 있는 범위(Authorized scopes)에서 사용자 풀 속성에 매핑할 소셜 자격 증명 공급자 범위의 이름을 입력합니다. 범위는 앱에서 액세스하려는 사용자 속성(예: 이름, 이메일)을 정의합니다. 범위를 입력하는 경우 선택한 IdP에 따라 다음 지침을 따르세요.

- Facebook - 범위를 쉼표로 구분합니다. 예:

  `public_profile, email`

- Google, Login with Amazon, Sign In with Apple - 범위를 공백으로 구분합니다. 예:
  - Google: `profile email openid`
  - Login with Amazon: `profile postal_code`
  - Sign In with Apple: `name email`

> **Note**
>
> Sign In with Apple(콘솔)의 경우 확인란을 사용하여 범위를 선택합니다.

7. 변경 사항 저장(Save changes)을 선택합니다.
8. [앱 클라이언트 통합(App client integration)] 탭의 목록에서 [앱 클라이언트(App clients)] 중 하나를 선택한 다음, [호스트된 UI 설정 편집(Edit hosted UI settings)]을 선택합니다. [자격 증명 공급자(Identity providers)]에서 앱 클라이언트에 새 소셜 자격 증명 공급자를 추가합니다.
9. 변경 사항 저장(Save changes)을 선택합니다.

IdPs소셜에 대한 자세한 내용은 을 참조하십시오. 사용자 풀이 있는 소셜 ID 공급자 사용

OIDC IdP를 사용하여 사용자 로그인 설정

사용자 로그인을 Salesforce 또는 Ping Identity와 같은 OpenID Connect(OIDC) 자격 증명 공급자(IdP)와 통합할 수 있습니다.

사용자 풀에 OIDC 자격 증명 공급자 추가

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 AWS 자격 증명을 입력합니다.
2. 탐색 메뉴에서 [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. [로그인 환경(Sign-in experience)] 탭을 선택합니다. [페더레이션 로그인(Federated sign-in)]을 찾아서 [자격 증명 공급자 추가(Add an identity provider)]를 선택합니다.
5. OpenID Connect 자격 증명 공급자를 선택합니다.
6. 공급자 이름(Provider name)에 고유한 이름을 입력합니다.
7. 공급자로부터 수신한 클라이언트 ID를 클라이언트 ID(Client ID)에 입력합니다.

8. 공급자로부터 수신한 클라이언트 암호를 클라이언트 암호(Client secret)에 입력합니다.
9. 이 공급자의 권한 있는 범위(Authorized scopes)를 입력합니다. 범위는 애플리케이션이 공급자로부터 요청할 사용자 속성(예: `name`, `email`) 그룹을 정의합니다. OAuth 2.0 사양에 따라 범위를 공백으로 구분해야 합니다.

   사용자는 애플리케이션에 이러한 속성을 제공하는 것에 동의해야 합니다.
10. 속성 요청 메서드(Attribute request method)를 선택하여 Amazon Cognito가 공급자가 운영하는 userInfo 엔드포인트에서 사용자 세부 정보를 가져오는 데 사용하는 HTTP 메서드(GET 또는 POST)를 Amazon Cognito에 제공합니다.
11. 설정 메서드(Setup method)를 선택하여 발급자 URL을 통해 자동 채우기(Auto fill through issuer URL) 또는 수동 입력(Manual input)으로 OpenID Connect 엔드포인트를 검색합니다. 공급자에게 Amazon Cognito가 `authorization`, `token`, `userInfo` 및 `jwks_uri` 엔드포인트의 URL을 검색할 수 있는 퍼블릭 `.well-known/openid-configuration` 엔드포인트가 있는 경우 발급자 URL을 통해 자동 채우기(Auto fill through issuer URL)를 사용합니다.
12. 발급자 URL이나 IdP의 `authorization`, `token`, `userInfo`, `jwks_uri` 엔드포인트 URL을 입력합니다.

> **Note**
>
> 검색 URL, 자동으로 채워진 URL 및 수동으로 입력한 URL에는 포트 번호 443과 80만 사용할 수 있습니다. OIDC 공급자가 비표준 TCP 포트를 사용하는 경우 사용자 로그인이 실패합니다.
> 발급자 URL은 `https://`로 시작해야 하며 / 문자로 시작하지 않아야 합니다. 예를 들어, Salesforce는 다음 URL을 사용합니다.
> `https://login.salesforce.com`
> 발급자 URL과 연결된 `openid-configuration` 문서는 `authorization_endpoint`, `token_endpoint`, `userinfo_endpoint` 및 `jwks_uri` 값에 대해 HTTPS URL을 제공해야 합니다. 마찬가지로 수동 입력(Manual input)을 선택하면 HTTPS URL만 입력할 수 있습니다.

13. OIDC 클레임 sub는 기본값으로 사용자 풀 속성 사용자 이름(Username)에 매핑됩니다. 이 외의 OIDC 클레임도 사용자 풀 속성으로 매핑할 수 있습니다. OIDC 클레임을 입력하고, 드롭다운 목록에서 해당하는 사용자 풀 속성을 선택합니다. 예를 들어, 클레임 email은 주로 사용자 풀 속성 Email로 매핑됩니다.
14. 자격 증명 공급자의 추가 속성을 사용자 풀에 매핑합니다. 자세한 내용은 사용자 풀에 대한 자격 증명 공급자 속성 매핑 지정을 참조하세요.

15. 생성(Create)을 선택합니다.
16. [앱 클라이언트 통합(App client integration)] 탭의 목록에서 [앱 클라이언트(App clients)] 중 하나를 선택하고 [호스트된 UI 설정 편집(Edit hosted UI settings)]을 수행합니다. [자격 증명 공급자(Identity providers)]에서 앱 클라이언트에 새 OIDC 자격 증명 공급자를 추가합니다.
17. 변경 사항 저장(Save changes)을 선택합니다.

OIDC에 대한 자세한 내용은 IdPs 을 참조하십시오. 사용자 풀과 함께 OIDC ID 공급자 사용

SAML IdP를 사용하여 사용자 로그인 설정

Amazon Cognito 사용자 풀에 페더레이션을 사용하여 SAML IdP(자격 증명 공급자)와 통합할 수 있습니다. 파일을 업로드하거나 메타데이터 문서 엔드포인트 URL을 입력하여 메타데이터 문서를 제공해야 합니다. 타사 IdPs SAML용 메타데이터 문서 가져오기에 대한 자세한 내용은 을 참조하십시오. 타사 SAML ID 공급자 구성

사용자 풀에 SAML 2.0 자격 증명 공급자를 구성하려면

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 AWS 자격 증명을 입력합니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. [로그인 환경(Sign-in experience)] 탭을 선택합니다. [페더레이션 로그인(Federated sign-in)]을 찾아서 [자격 증명 공급자 추가(Add an identity provider)]를 선택합니다.
5. [SAML] 자격 증명 공급자를 선택합니다.
6. 식별자(Identifiers)를 쉼표로 구분하여 입력합니다. 식별자는 Amazon Cognito가 사용자 로그인 이메일 주소를 확인한 다음 해당 도메인에 해당하는 공급자로 사용자를 안내하도록 합니다.
7. 사용자가 로그아웃할 때 Amazon Cognito에서 서명된 로그아웃 요청을 공급자에게 보내도록 하려는 경우 로그아웃 흐름 추가(Add sign-out flow)를 선택합니다. 호스팅 UI를 구성할 때 Amazon Cognito에서 생성되는 `https://mydomain.us-east-1.amazoncognito.com/saml2/logout` 엔드포인트에 로그아웃 응답을 보내도록 SAML 2.0 자격 증명 공급자를 구성합니다. `saml2/logout` 엔드포인트는 POST 바인딩을 사용합니다.

> **Note**
>
> 이 옵션을 선택했고 SAML 자격 증명 공급자가 서명된 로그아웃 요청을 필요로 하는 경우 SAML IdP를 사용하여 Amazon Cognito에서 제공한 서명 인증서도 구성해야 합니다.

SAML IdP가 서명된 로그아웃 요청을 처리하고 사용자를 Amazon Cognito 세션에서 로그아웃합니다.

8. 메타데이터 문서 소스(Metadata document source)를 선택합니다. 자격 증명 공급자가 퍼블릭 URL에서 SAML 메타데이터를 제공하는 경우 메타데이터 문서 URL(Metadata document URL)을 선택하고 해당 퍼블릭 URL을 입력할 수 있습니다. 그렇지 않은 경우 메타데이터 문서 업로드(Upload metadata document)를 선택한 다음, 이전에 공급자로부터 다운로드한 메타데이터 파일을 선택합니다.

   Note

   공급자에게 퍼블릭 엔드포인트가 있는 경우 파일을 업로드하지 않고 메타데이터 문서 URL을 입력하는 것이 좋습니다. URL을 사용하는 경우 Amazon Cognito는 메타데이터를 자동으로 새로 고칩니다. 일반적으로 메타데이터 새로 고침은 6시간마다 또는 메타데이터가 만료되기 전 중 더 빠른 시간에 발생합니다.

9. [SAML 공급자와 앱 간에 속성 매핑(Map attributes between your SAML provider and your app)]을 선택하여 SAML 공급자 속성을 사용자 풀의 사용자 프로파일에 매핑합니다. 속성 맵에 사용자 풀 필수 속성을 포함합니다.

   예를 들어 [사용자 풀 속성(User pool attribute)] `email`을 선택한 경우 자격 증명 공급자의 SAML 어설션에 표시된 대로 SAML 속성 이름을 입력합니다. 자격 증명 공급자가 참조용으로 샘플 SAML 어설션을 제공할 수도 있습니다. `email`과 같은 간단한 이름을 사용하는 자격 증명 공급자도 있고, 다음과 같이 URL 포맷의 속성 이름을 사용하는 자격 증명 공급자도 있습니다.

   ```
   http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
   ```

10. 생성을 선택하세요.

Note

HTTPS 메타데이터 엔드포인트 URL을 사용하여 SAML IdP를 생성하는 동안 `InvalidParameterException`이 나타나는 경우 메타데이터 엔드포인트에 SSL이 올바르게 설정되어 있고 유효한 SSL 인증서가 연결되어 있는지 확인합니다. 이러한 예외의 한 예로는 "<메타데이터 엔드포인트>에서 메타데이터를 수신하는 동안 오류가 발생했습니다.(Error retrieving metadata from *<metadata endpoint>*)"가 있습니다.

서명 인증서를 추가하도록 SAML IdP를 설정하려면

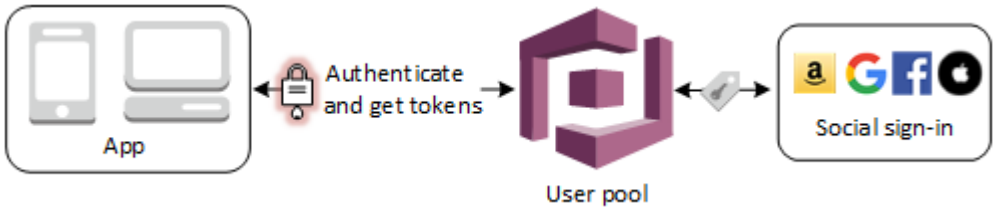
- IdP가 서명된 로그아웃 요청을 확인하는 데 사용할 퍼블릭 키가 포함된 인증서를 가져오려면 페더레이션 콘솔 페이지의 자격 증명 공급자에서 SAML 대화 상자의 활성 SAML 공급자 아래에 있는 서명 인증서 표시를 선택합니다.

IdPs SAML에 대한 자세한 내용은 을 참조하십시오. 사용자 풀과 함께 SAML ID 공급자 사용

사용자 풀이 있는 소셜 ID 공급자 사용

웹 및 모바일 앱 사용자들이 Facebook, Google, Amazon 및 Apple 같은 소셜 자격 증명 공급자(IdP)를 통해 로그인할 수 있습니다. Amazon Cognito는 기본 제공 호스트된 웹 UI를 사용하여 모든 인증된 사용자에 대한 토큰 처리 및 관리 기능을 제공합니다. 이렇게 하면 백엔드 시스템을 한 세트의 사용자 풀 토큰에서 표준화할 수 있습니다. 지원되는 소셜 자격 증명 공급자와 통합할 호스트된 UI를 사용하도록 설정해야 합니다. Amazon Cognito는 호스팅된 UI를 구축할 때 Amazon Cognito와 OIDC 및 소셜에서 정보를 교환하는 데 사용하는 OAuth 2.0 엔드포인트를 생성합니다. IdPs 자세한 내용은 Amazon Cognito 사용자 풀 인증 API 참조를 참조하세요.

에서 소셜 IdP를 추가하거나 AWS CLI 또는 Amazon Cognito API를 사용할 수 있습니다. AWS Management Console



> **Note**
>
> 서드 파티(페더레이션)를 통한 로그인을 Amazon Cognito 사용자 풀에서 사용할 수 있습니다. 이 기능은 Amazon Cognito 자격 증명 풀(페더레이션 자격 증명)을 통한 페더레이션과 무관합니다.

주제

- 필수 조건
- 1단계: 소셜 IdP에 등록
- 2단계: 사용자 풀에 소셜 IdP 추가

- 3단계: 소셜 IdP 구성 테스트

필수 조건

시작하려면 다음이 필요합니다.

- 앱 클라이언트와 사용자 풀 도메인이 있는 사용자 풀. 자세한 내용은 사용자 풀 생성을 참조하세요.
- 소셜 IdP입니다.

1단계: 소셜 IdP에 등록

Amazon Cognito에서 소셜 IdP를 생성하려면 소셜 IdP에 애플리케이션을 등록하여 클라이언트 ID와 클라이언트 암호를 받아야 합니다.

Facebook으로 앱을 등록하려면

1. Facebook에서 개발자 계정을 생성합니다.
2. Facebook 자격 증명으로 로그인합니다.
3. 내 앱(My Apps) 메뉴에서 Create New App(새 앱 생성)을 선택합니다.
4. Facebook 앱의 이름을 입력하고 [앱 ID 생성(Create App ID)]을 선택합니다.
5. 왼쪽 탐색 모음에서 [설정(Settings)], [기본 사항(Basic)]을 차례로 선택합니다.
6. 앱 ID(App ID)와 앱 보안(App Secret)을 메모합니다. 다음 섹션에서 이 둘을 사용합니다.
7. 페이지 하단에서 + 플랫폼 추가(+ Add Platform)을 선택합니다.
8. 웹 사이트(Website)를 선택합니다.
9. [웹 사이트(Website)]의 [사이트 URL(Site URL)]에 앱 로그인 페이지의 경로를 입력합니다.

```
https://mydomain.us-east-1.amazoncognito.com/login?
response_type=code&client_id=1example23456789&redirect_uri=https://www.example.com
```

10. 변경 사항 저장(Save changes)을 선택합니다.
11. [앱 도메인(App Domains)]에 사용자 풀 도메인 루트의 경로를 입력합니다.

```
https://mydomain.us-east-1.amazoncognito.com
```

12. 변경 사항 저장(Save changes)을 선택합니다.

13. 탐색 모음에서 [제품 추가(Add Product)]을 선택한 다음 [Facebook 로그인(Facebook Login)] 제품의 [설정(Set up)]을 선택합니다.
14. 탐색 모음에서 Facebook 로그인(Facebook Login)과 설정(Settings)을 차례로 선택합니다.

    유효한 OAuth 리디렉션 URI(Valid OAuth Redirect URIs)에 사용자 풀 도메인 `/oauth2/idpresponse` 엔드포인트의 경로를 입력합니다.

    ```
    https://mydomain.us-east-1.amazoncognito.com/oauth2/idpresponse
    ```

15. 변경 사항 저장(Save changes)을 선택합니다.

Amazon으로 앱을 등록하려면

1. Amazon에서 개발자 계정을 생성합니다.
2. Amazon 자격 증명으로 로그인합니다.
3. Amazon 보안 프로파일을 생성하여 Amazon 클라이언트 ID와 클라이언트 암호를 받아야 합니다.

    페이지 상단의 탐색 모음에서 Apps and Services(앱 및 서비스)를 선택한 다음 Login with Amazon을 선택합니다.
4. 보안 프로필 생성(Create a Security Profile)을 선택합니다.
5. 보안 프로파일 이름(Security Profile Name), 보안 프로파일 설명(Security Profile Description), 개인 정보 보호 정책 동의 URL(Consent Privacy Notice URL)을 입력합니다.
6. 저장(Save)을 선택합니다.
7. 클라이언트 ID(Client ID) 및 클라이언트 암호(Client Secret)를 선택하여 클라이언트 ID 및 암호를 표시합니다. 다음 섹션에서 이 둘을 사용합니다.
8. 톱니 모양 아이콘을 마우스로 가리키고 [웹 설정(Web Settings)]을 선택한 다음, [편집(Edit)]을 선택합니다.
9. [허용된 원본(Allowed Origins)]에 사용자 풀 도메인을 입력합니다.

    ```
    https://mydomain.us-east-1.amazoncognito.com
    ```

10. [허용된 반환 URL(Allowed Return URLs)]에 `/oauth2/idpresponse` 엔드포인트가 있는 사용자 풀 도메인을 입력합니다.

    ```
    https://mydomain.us-east-1.amazoncognito.com/oauth2/idpresponse
    ```

11. 저장(Save)을 선택합니다.

Google로 앱을 등록하려면

Google Cloud Platform의 OAuth 2.0에 대한 자세한 내용은 개발자용 Google Workspace 설명서의 인증 및 권한 부여 알아보기를 참조하세요.

1. Google에서 개발자 계정을 생성합니다.
2. Google Cloud Platform 콘솔에 로그인합니다.
3. 상단 탐색 모음에서 프로젝트 선택(Select a project)을 선택합니다. Google 플랫폼에 프로젝트가 이미 있는 경우 이 메뉴에는 대신 기본 프로젝트가 표시됩니다.
4. 새 프로젝트(NEW PROJECT)를 선택합니다.
5. 프로젝트의 이름을 입력한 다음 생성(CREATE)을 선택합니다.
6. 왼쪽 탐색 모음에서 API 및 서비스(APIs and Services)와 Oauth 동의 화면(Oauth consent screen)을 차례로 선택합니다.
7. 앱 정보(App information), 앱 도메인(App domain), 공인 도메인(Authorized domains) 및 개발자 연락처 정보(Developer contact information)를 입력합니다. 공인 도메인(Authorized domains)은 `amazoncognito.com` 및 사용자 지정 도메인의 루트를 포함해야 합니다(예: `example.com`). 저장 후 계속(SAVE AND CONTINUE)을 선택합니다.
8. 1. 범위(Scopes)에서 범위 추가 또는 제거(Add or remove scopes)를 선택하고 최소한 다음 OAuth 범위를 선택합니다.

    1. `.../auth/userinfo.email`
    2. `.../auth/userinfo.profile`
    3. openid
9. 사용자 테스트(Test users)에서 사용자 추가(Add users)를 선택합니다. 이메일 주소와 기타 인증된 테스트 사용자를 입력한 다음 저장 후 계속(SAVE AND CONTINUE)을 선택합니다.
10. 왼쪽 탐색 모음을 다시 확장하고 API 및 서비스(APIs and Services)와 보안 인증 정보(Credentials)를 차례로 선택합니다.
11. 보안 인증 정보 생성(CREATE CREDENTIALS)과 OAuth 클라이언트 ID(OAuth client ID)를 차례로 선택합니다.
12. 애플리케이션 유형(Application type)을 선택하고 클라이언트 이름을 지정합니다.
13. 승인된 JavaScript 출처에서 URI 추가를 선택합니다. 사용자 풀 도메인을 입력합니다.

```
https://mydomain.us-east-1.amazoncognito.com
```

14. 권한 있는 리디렉션 URI(Authorized redirect URIs)에서 URI 추가(ADD URI)를 선택합니다. 사용자 풀 도메인의 `/oauth2/idpresponse` 엔드포인트에 대한 경로를 입력합니다.

```
https://mydomain.us-east-1.amazoncognito.com/oauth2/idpresponse
```

15. 생성(CREATE)을 선택합니다.
16. 클라이언트 ID(Your client ID)와 클라이언트 암호(Your client secret)에서 Google이 표시하는 값을 안전하게 저장합니다. Google IdP를 추가할 때 이러한 값을 Amazon Cognito에 입력합니다.

Apple으로 앱을 등록하려면

Apple로 로그인 설정에 대한 자세한 up-to-date 내용은 Apple 개발자 설명서에서 Apple로 로그인할 수 있는 환경 구성을 참조하십시오.

1. Apple에서 개발자 계정을 생성합니다.
2. Apple 자격 증명으로 로그인합니다.
3. 왼쪽 탐색 모음에서 인증서, 식별자 및 프로필(Certificates, Identifiers & Profiles)을 선택합니다.
4. 왼쪽 탐색 모음에서 식별자(Identifiers)를 선택합니다.
5. 식별자(Identifiers) 페이지에서 + 아이콘을 선택합니다.
6. 새 식별자 등록(Register a New Identifier) 페이지에서 앱 ID(App IDs)를 선택한 다음 계속(Continue)을 선택합니다.
7. 유형 선택(Select a type) 페이지에서 앱(App)을 선택하고 계속(Continue)을 선택합니다.
8. 앱 ID 등록(Register an App ID) 페이지에서 다음을 수행합니다.
    1. 설명(Description)에 설명을 입력합니다.
    2. 앱 ID 접두사(App ID Prefix)에서 번들 ID(Bundle ID)를 입력합니다. [앱 ID 접두사(App ID Prefix)] 아래의 값을 적어 둡니다. 2단계: 사용자 풀에 소셜 IdP 추가에서 자격 증명 공급자로 Apple을 선택한 후 이 값을 사용합니다.
    3. 기능(Capabilities)에서 Apple로 로그인(Sign In with Apple)을 선택한 다음 편집(Edit)을 선택합니다.
    4. Apple로 로그인: 앱 ID 구성(Sign in with Apple: App ID Configuration) 페이지에서 앱을 기본 앱으로 설정하거나 다른 앱 ID와 함께 그룹화하도록 선택한 다음, 저장(Save)을 선택합니다.

5. 계속을 선택합니다.

9. 앱 ID 확인(Confirm your App ID) 페이지에서 등록(Register)을 선택합니다.
10. 식별자(Identifiers) 페이지에서 + 아이콘을 선택합니다.
11. 새 식별자 등록(Register a New Identifier) 페이지에서 서비스 ID(Services IDs)를 선택한 다음 계속(Continue)을 선택합니다.
12. 서비스 ID 등록(Register a Services ID) 페이지에서 다음을 수행합니다.

    1. 설명에 설명을 입력합니다.
    2. Identifier(식별자)에 식별자를 입력합니다. 2단계: 사용자 풀에 소셜 IdP 추가에서 ID 제공업체로 Apple을 선택한 후 이 값이 필요하므로 이 서비스 ID를 기록해 두세요.
    3. 계속(Continue)과 등록(Register)을 차례로 선택합니다.
13. 식별자(Identifiers) 페이지에서 방금 생성한 서비스 ID를 선택합니다.

    1. Apple로 로그인(Sign In with Apple)을 선택한 다음 구성(Configure)을 선택합니다.
    2. 웹 인증 구성(Web Authentication Configuration) 페이지에서 앞서 생성한 앱 ID를 기본 앱 ID(Primary App ID)로 선택합니다.
    3. 웹 사이트 URL(Website URLs)에서 + 아이콘을 선택합니다.
    4. 도메인 및 하위 도메인(Domains and subdomains)에서 `https://` 접두사를 사용하지 않고 사용자 풀 도메인을 입력합니다.

       ```
       mydomain.us-east-1.amazoncognito.com
       ```

    5. 반환 URL(Return URLs)에 사용자 풀 도메인의 `/oauth2/idpresponse` 엔드포인트에 대한 경로를 입력합니다.

       ```
       https://mydomain.us-east-1.amazoncognito.com/oauth2/idpresponse
       ```

    6. 다음(Next)과 완료(Done)를 차례로 선택합니다. 도메인을 확인할 필요는 없습니다.
    7. 계속(Continue)과 저장(Save)을 차례로 선택합니다.
14. 왼쪽 탐색 모음에서 키(Keys)를 선택합니다.
15. 키(Keys) 페이지에 + 아이콘을 선택합니다.
16. 새 키 등록(Register a New Key) 페이지에서 다음을 수행합니다.

    1. [키 이름(Key Name)] 아래에 키 이름을 입력합니다.
    2. Apple로 로그인(Sign In with Apple)을 선택한 다음 구성(Configure)을 선택합니다.

3. 키 구성(Configure Key) 페이지에서 앞서 생성한 앱 ID를 기본 앱 ID(Primary App ID)로 선택합니다. 저장을 선택합니다.
4. 계속(Continue), 등록(Register)을 차례로 선택합니다.

17. [키 다운로드(Download Your Key)] 페이지에서 [다운로드(Download)]를 선택하여 프라이빗 키를 다운로드하고 표시된 [키 ID(Key ID)]를 적어 둔 다음 [완료(Done)]를 선택합니다. 이 프라이빗 키와 이 페이지에 표시되는 키 ID(Key ID) 값은 2단계: 사용자 풀에 소셜 IdP 추가에서 자격 증명 공급자로 Apple을 선택한 후에 필요합니다.

2단계: 사용자 풀에 소셜 IdP 추가

를 사용하여 사용자 풀 소셜 IdP를 구성하려면 AWS Management Console

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 AWS 자격 증명을 입력합니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. [로그인 환경(Sign-in experience)] 탭을 선택합니다. 페더레이션 로그인(Federated sign-in)을 찾아서 자격 증명 공급자 추가(Add an identity provider)를 선택합니다.
5. Login with Amazon 또는 Facebook, Google Apple로 로그인(Sign in with Apple) 소셜 IdP를 선택합니다.
6. 선택한 소셜 IdP에 따라 다음 단계 중에서 선택합니다.

   - Google 및 Login with Amazon - 이전 섹션에서 생성된 앱 클라이언트 ID(app client ID)와 앱 클라이언트 암호(app client secret)를 입력합니다.
   - Facebook - 이전 섹션에서 생성된 앱 클라이언트 ID(app client ID)와 앱 클라이언트 암호(app client secret)를 입력하고 API 버전(예: 버전 2.12)을 선택합니다. 각 Facebook API에는 수명 주기와 사용 중단 날짜가 있으므로 가능한 한 최신 버전을 선택하는 것이 좋습니다. Facebook 범위와 속성은 API 버전마다 다를 수 있습니다. Facebook에서 소셜 자격 증명 로그인을 테스트하여 페더레이션이 의도한 대로 작동하는지 확인하는 것이 좋습니다.
   - Apple로 로그인(Sign In with Apple) - 이전 섹션에서 생성된 서비스 ID(Services ID), 팀 ID(Team ID), 키 ID(Key ID), 프라이빗 키(private key)를 입력합니다.

7. 사용할 [권한 있는 범위(Authorized scopes)]의 이름을 입력합니다. 범위는 앱에서 액세스하고자 하는 사용자 속성(예: `name` 및 `email`)을 정의합니다. Facebook의 경우 쉼표로 구분해야 합니다. Google 및 Login with Amazon의 경우 공백으로 구분해야 합니다. Apple로 로그인의 경우, 액세스하려는 범위의 확인란을 선택합니다.

소셜 자격 증명 공급자	예제 범위
Facebook	`public_profile, email`
Google	`profile email openid`
Login with Amazon	`profile postal_code`
Apple로 로그인	`email name`

앱에 이러한 속성을 제공하는 데 동의하라는 메시지가 앱 사용자에게 표시됩니다. 소셜 공급자 범위에 대한 자세한 내용은 Google, Facebook, Login with Amazon 또는 Sign in with Apple의 설명서를 참조하세요.

Sign in with Apple의 경우 다음은 범위가 반환되지 않을 수 있는 사용자 시나리오입니다.

- 최종 사용자가 Apple 로그인 페이지를 벗어난 후 오류 발생(Amazon Cognito 내부 오류 또는 개발자가 작성한 항목에서 발생할 수 있음)
- 서비스 ID 식별자가 여러 사용자 풀 및/또는 다른 인증 서비스에서 사용됨
- 최종 사용자가 이전에 로그인한 후 개발자가 추가 범위를 추가함(새로운 정보가 검색되지 않음)
- 개발자가 사용자를 삭제한 후 해당 사용자가 Apple ID 프로필에서 앱을 제거하지 않고 다시 로그인함

8. IdP의 속성을 사용자 풀에 매핑합니다. 자세한 내용은 사용자 풀에 대한 자격 증명 공급자 속성 매핑 지정을 참조하세요.
9. 생성(Create)을 선택합니다.
10. 앱 클라이언트 통합(App client integration) 탭의 목록에서 앱 클라이언트(App clients) 중 하나를 선택한 다음, 호스트된 UI 설정 편집(Edit hosted UI settings)을 선택합니다. 자격 증명 공급자(Identity providers)에서 앱 클라이언트에 새 소셜 IdP를 추가합니다.
11. 변경 사항 저장(Save changes)을 선택합니다.

3단계: 소셜 IdP 구성 테스트

이전 두 섹션의 요소를 사용하여 로그인 URL을 생성할 수 있습니다. 이것을 사용하여 소셜 IdP 구성을 테스트합니다.

```
https://mydomain.us-east-1.amazoncognito.com/login?
response_type=code&client_id=1example23456789&redirect_uri=https://www.example.com
```

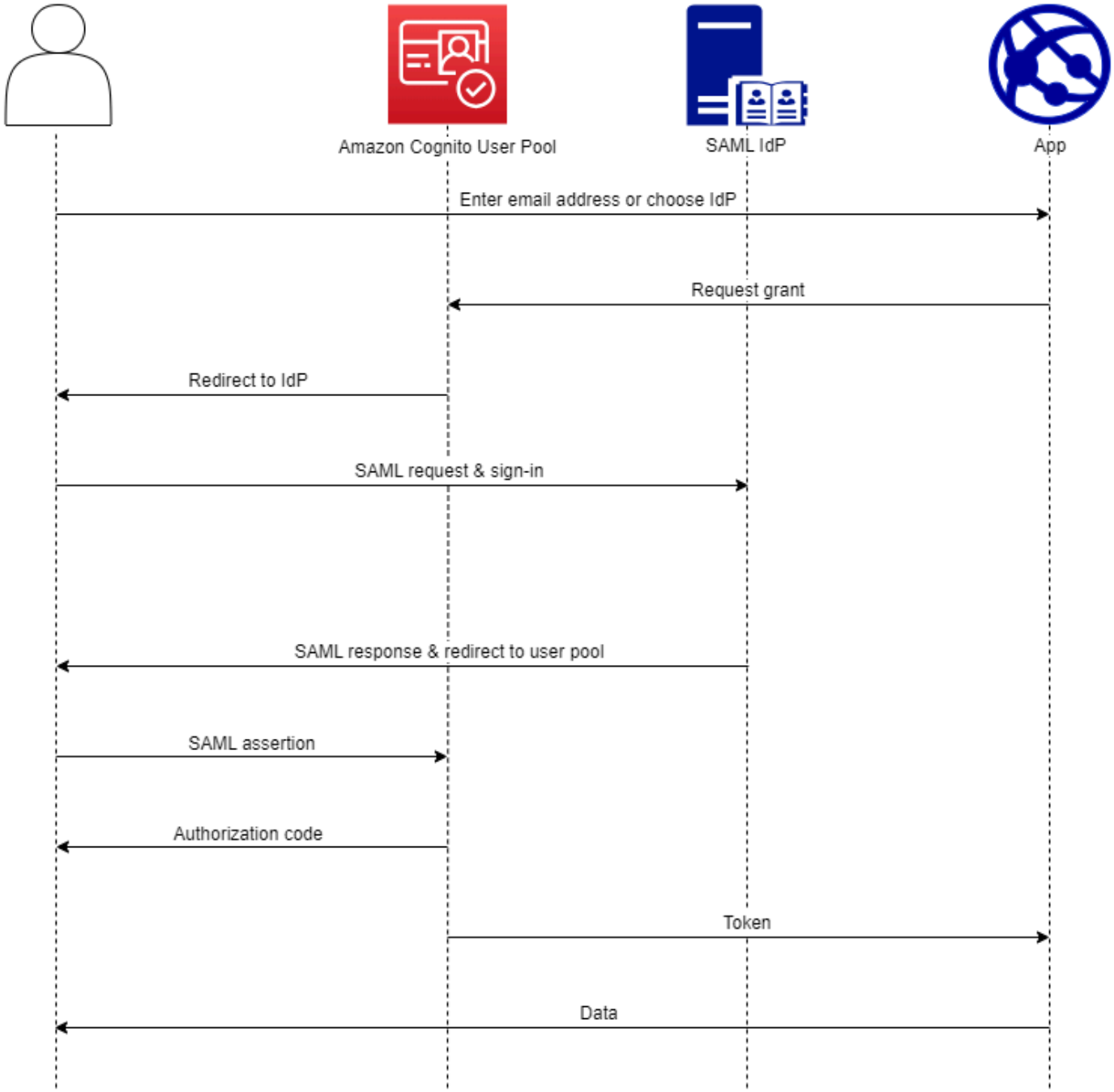
사용자 풀 도메인 이름(Domain name) 콘솔 페이지에서 사용자의 도메인을 찾을 수 있습니다. client_id는 앱 클라이언트 설정(App client settings) 페이지에 있습니다. redirect_uri 파라미터용 콜백 URL을 사용합니다. 이것은 인증 성공 이후에 사용자가 리디렉션되는 페이지의 URL입니다.

> **Note**
>
> Amazon Cognito는 5분 이내에 완료되지 않는 인증 요청을 취소하고 사용자를 호스팅 UI로 리디렉션합니다. 페이지에 `Something went wrong` 오류 메시지가 표시됩니다.

사용자 풀과 함께 SAML ID 공급자 사용

웹 및 모바일 앱 사용자가 Microsoft Active Directory 페더레이션 서비스 (ADFS) 또는 Shibboleth와 같은 SAML ID 공급자 (IdP) 를 통해 로그인하도록 선택할 수 있습니다. SAML 2.0 표준을 지원하는 SAML IdP를 선택해야 합니다.

호스팅된 UI 및 페더레이션 엔드포인트를 통해 Amazon Cognito는 로컬 및 타사 IdP 사용자를 인증하고 JSON 웹 토큰 (JWT) 을 발행합니다. Amazon Cognito에서 발급하는 토큰을 사용하면 모든 앱에서 여러 자격 증명 소스를 범용 OpenID Connect (OIDC) 표준으로 통합할 수 있습니다. Amazon Cognito는 타사 공급자의 SAML 어설션을 해당 SSO 표준으로 처리할 수 있습니다. Amazon Cognito 사용자 풀 API에서 AWS Management Console, API를 통해 AWS CLI또는 API를 사용하여 SAML IdP를 생성하고 관리할 수 있습니다. 에서 첫 번째 SAML IdP를 만들려면 AWS Management Console을 참조하십시오. 사용자 풀에 SAML 자격 증명 공급자 추가 및 관리



Note

타사 IdP를 통한 로그인과의 페더레이션은 Amazon Cognito 사용자 풀의 기능입니다. Amazon Cognito 페더레이션 자격 증명이라고도 하는 Amazon Cognito 자격 증명 풀은 각 자격 증명 풀

에서 개별적으로 설정해야 하는 페더레이션을 구현한 것입니다. 사용자 풀은 자격 증명 풀의 타사 IdP가 될 수 있습니다. 자세한 정보는 Amazon Cognito 자격 증명 풀을 참조하세요.

IdP 구성에 대한 빠른 참조

요청을 수락하고 사용자 풀에 응답을 보내도록 SAML IdP를 구성해야 합니다. SAML IdP 설명서에는 사용자 풀을 SAML 2.0 IdP의 신뢰 당사자 또는 애플리케이션으로 추가하는 방법에 대한 정보가 포함됩니다. 다음 설명서에는 SP 엔티티 ID 및 어설션 소비자 서비스 (ACS) URL에 제공해야 하는 값이 나와 있습니다.

사용자 풀 SAML 값 빠른 참조

SP 엔티티 ID

```
urn:amazon:cognito:sp:us-east-1_EXAMPLE
```

에이스 URL

```
https://Your user pool domain/saml2/idpresponse
```

ID 공급자를 지원하도록 사용자 풀을 구성해야 합니다. 외부 SAML IdP를 추가하는 상위 단계는 다음과 같습니다.

1. IdP에서 SAML 메타데이터를 다운로드하거나 메타데이터 엔드포인트의 URL을 검색합니다. 타사 SAML ID 공급자 구성를 참조하세요.
2. 사용자 풀에 새 IdP를 추가합니다. SAML 메타데이터를 업로드하거나 메타데이터 URL을 제공합니다. 사용자 풀에 SAML 자격 증명 공급자 추가 및 관리를 참조하세요.
3. 앱 클라이언트에 IdP를 할당합니다. 사용자 풀 앱 클라이언트 섹션 참조

주제

- Amazon Cognito 사용자 풀의 IdPs SAML에 대해 알아야 할 사항
- SAML 사용자 이름의 대/소문자 구분
- 사용자 풀에 SAML 자격 증명 공급자 추가 및 관리
- Amazon Cognito 사용자 풀에서 SAML 세션 시작

- SP에서 시작한 SAML 로그인 사용
- IDP에서 시작한 SAML 로그인 사용
- SAML 로그아웃 플로우
- SAML 서명 및 암호화
- SAML ID 공급자 이름 및 식별자
- 타사 SAML ID 공급자 구성

Amazon Cognito 사용자 풀의 IdPs SAML에 대해 알아야 할 사항

Amazon Cognito는 사용자를 대신하여 SAML 어설션을 처리합니다.

Amazon Cognito 사용자 풀은 사후 바인딩 엔드포인트와의 SAML 2.0 페더레이션을 지원합니다. 이렇게 하면 사용자 풀이 사용자 에이전트를 통해 IdP로부터 직접 SAML 응답을 받으므로 앱에서 SAML 어설션 응답을 검색하거나 구문 분석할 필요가 없습니다. 사용자 풀은 애플리케이션을 대신하여 서비스 공급자(SP) 역할을 합니다. Amazon Cognito는 SAML V2.0 기술 개요의 섹션 5.1.2 및 5.1.4에 설명된 대로 SP 개시 및 IdP 개시 싱글 사인온 (SSO) 을 지원합니다.

유효한 IdP 서명 인증서 제공

사용자 풀에서 SAML IdP를 구성할 때 SAML 공급자 메타데이터의 서명 인증서가 만료되어서는 안 됩니다.

사용자 풀은 다중 서명 인증서를 지원합니다.

SAML IdP가 SAML 메타데이터에 서명 인증서를 두 개 이상 포함하는 경우, 로그인 시 사용자 풀은 SAML 어설션이 SAML 메타데이터에 있는 인증서와 일치할 경우 유효하다고 판단합니다. 각 서명 인증서의 길이는 4,096자를 넘지 않아야 합니다.

릴레이 상태 매개 변수를 유지하십시오.

Amazon Cognito와 SAML IdP는 `relayState` 파라미터를 사용하여 세션 정보를 유지 관리합니다.

1. Amazon Cognito는 80바이트보다 큰 `relayState` 값을 지원합니다. SAML 사양에는 `relayState` 값의 "길이가 80바이트를 초과해서는 안 된다"고 명시되어 있지만, 현재 업계 관행은 이 동작에서 벗어나는 경우가 많습니다. 결과적으로 80바이트보다 큰 `relayState` 값을 거부하면 많은 표준 SAML 공급자 통합이 중단됩니다.
2. `relayState`토큰은 Amazon Cognito에서 유지 관리하는 상태 정보에 대한 불투명한 참조입니다. Amazon Cognito는 `relayState` 파라미터의 내용을 보장하지 않습니다. 앱이 결과에 따라 달라지도록 내용을 구문 분석하지 마세요. 자세한 내용은 SAML 2.0 사양을 참조하세요.

ACS 엔드포인트를 식별하십시오.

SAML 자격 증명 공급자에 어설션 소비자 엔드포인트를 설정해야 합니다. IdP는 SAML 어설션을 통해 사용자를 이 엔드포인트로 리디렉션합니다. SAML ID 제공업체의 SAML 2.0 POST 바인딩에 대해 사용자 풀 도메인에서 다음 엔드포인트 구성합니다.

```
https://Your user pool domain/saml2/idpresponse
With an Amazon Cognito domain:
https://mydomain.us-east-1.amazoncognito.com/saml2/idpresponse
With a custom domain:
https://auth.example.com/saml2/idpresponse
```

사용자 풀 도메인에 대한 자세한 정보는 사용자 풀 도메인 구성 단원을 참조하세요.

어설션이 다시 재생되지 않음

Amazon Cognito `saml2/idpresponse` 엔드포인트에 대한 SAML 어설션을 반복하거나 재생할 수 없습니다. 재생된 SAML 어설션에는 이전 IdP 응답의 ID와 중복되는 어설션 ID가 있습니다.

사용자 풀 ID는 SP 개체 ID입니다.

대상 URI 또는 SP 개체 ID라고도 하는 서비스 공급자 (SP) `urn` 의 사용자 풀 ID를 IdP에 제공해야 합니다. 사용자 풀에 대한 대상 URI는 다음 형식을 갖습니다.

```
urn:amazon:cognito:sp:us-east-1_EXAMPLE
```

Amazon Cognito 콘솔의 사용자 풀 개요에서 사용자 풀 ID를 찾을 수 있습니다.

모든 필수 속성을 매핑합니다.

SAML IdP에서 사용자 풀에 필요한 대로 설정한 속성에 대한 속성 값도 제공하도록 구성해야 합니다. 예를 들어 `email`은 사용자 풀의 일반적인 필수 속성입니다. 사용자가 로그인하려면 먼저 SAML IdP 어설션에 사용자 풀 속성 `email`에 매핑되는 클레임이 포함되어야 합니다. 속성 매핑에 대한 자세한 내용은 사용자 풀에 대한 자격 증명 공급자 속성 매핑 지정을 참조하십시오.

어설션 형식에는 특정 요구 사항이 있습니다.

SAML IdP는 SAML 어설션에 다음 클레임을 포함해야 합니다.

1. 클레임. `NameID` Amazon Cognito는 SAML 어설션을 대상 사용자와 연결합니다. `NameID` `NameID`변경되는 경우 Amazon Cognito는 어설션을 새 사용자를 위한 것으로 간주합니다. IdP `NameID` 구성에서 설정한 속성에는 영구 값이 있어야 합니다.

```
<saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:persistent">
  carlos
</saml2:NameID>
```

2. 사용자 풀 SP 엔터티 ID를 응답 대상으로 설정하는 Audience 값이 포함된 AudienceRestriction 클레임.

```
<saml:AudienceRestriction>
  <saml:Audience> urn:amazon:cognito:sp:us-east-1_EXAMPLE
</saml:AudienceRestriction>
```

3. SP에서 시작한 싱글 사인온의 경우 원래 SAML 요청 ID InResponseTo 값을 가진 Response 요소입니다.

```
<saml2p:Response Destination="https://mydomain.us-east-1.amazoncognito.com/
saml2/idpresponse" ID="id123" InResponseTo="_dd0a3436-bc64-4679-
a0c2-cb4454f04184" IssueInstant="Date-time stamp" Version="2.0"
 xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol" xmlns:xs="http://
www.w3.org/2001/XMLSchema">
```

> **Note**
>
> IdP에서 시작한 SAML 어설션은 값을 포함할 수 없습니다. InResponseTo

4. 사용자 풀 saml2/idpresponse 엔드포인트 Recipient 값이 있는 SubjectConfirmationData 요소 (SP에서 시작한 SAML의 경우 원래 SAML 요청 ID와 일치하는 값) InResponseTo

```
<saml2:SubjectConfirmationData InResponseTo="_dd0a3436-bc64-4679-a0c2-
cb4454f04184" NotOnOrAfter="Date-time stamp" Recipient="https://mydomain.us-
east-1.amazoncognito.com/saml2/idpresponse"/>
```

SP에서 시작한 로그인 요청

권한 부여 엔드포인트에서 사용자를 IdP 로그인 페이지로 리디렉션하면, Amazon Cognito는 SAML 요청을 HTTP GET 요청의 URL 파라미터에 포함합니다. SAML 요청에는 ACS 엔드포인트를 포함한 사용자 풀에 대한 정보가 포함됩니다. 선택적으로 이러한 요청에 암호화 서명을 적용할 수 있습니다.

요청에 서명하고 응답을 암호화합니다.

SAML 공급자가 있는 모든 사용자 풀은 Amazon Cognito가 SAML 요청에 할당하는 디지털 서명을 위한 비대칭 키 페어와 서명 인증서를 생성합니다. 암호화된 SAML 응답을 지원하도록 구성하는 모든 외부 SAML IdP는 Amazon Cognito에서 해당 공급자를 위한 새 키 페어와 암호화 인증서를 생성합니다. 공개 키가 있는 인증서를 보고 다운로드하려면 Amazon Cognito 콘솔의 로그인 환경 탭에서 IdP를 선택하십시오.

사용자 풀의 SAML 요청으로 신뢰를 구축하려면 IdP에 사용자 풀 SAML 2.0 서명 인증서 사본을 제공하십시오. 서명된 요청을 수락하도록 IdP를 구성하지 않으면 IdP가 사용자 풀이 서명한 SAML 요청을 무시할 수 있습니다.

1. Amazon Cognito는 사용자가 IdP로 전달하는 SAML 요청에 디지털 서명을 적용합니다. 사용자 풀은 모든 싱글 로그아웃 (SLO) 요청에 서명하며, 모든 SAML 외부 IdP에 대한 싱글 사인온 (SSO) 요청에 서명하도록 사용자 풀을 구성할 수 있습니다. 인증서 사본을 제공하면 IdP가 사용자 SAML 요청의 무결성을 확인할 수 있습니다.
2. SAML IdP는 암호화 인증서를 사용하여 SAML 응답을 암호화할 수 있습니다. SAML 암호화로 IdP를 구성하는 경우 IdP는 암호화된 응답만 전송해야 합니다.

영숫자가 아닌 문자를 인코딩하세요.

Amazon Cognito는 IdP가 속성 값으로 전달하는 # 또는 와 같은 4바이트 UTF-8 문자를 허용하지 않습니다. 문자를 Base64로 인코딩하여 텍스트로 전달한 다음 앱에서 디코딩할 수 있습니다.

다음 예제에서는 속성 클레임이 허용되지 않습니다.

```
<saml2:Attribute Name="Name" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:unspecified">
  <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">#</saml2:AttributeValue>
</saml2:Attribute>
```

앞의 예제와 달리 다음 속성 클레임은 허용됩니다.

```
<saml2:Attribute Name="Name" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:unspecified">
  <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">8J+YkA==</saml2:AttributeValue>
</saml2:Attribute>
```

메타데이터 엔드포인트에는 유효한 전송 계층 보안이 있어야 합니다.

HTTPS 메타데이터 엔드포인트 URL을 사용하여 SAML IdP를 생성하는 동안 `InvalidParameterException`(예: "*<metadata endpoint>*에서 메타데이터를 가져오는 동안 오류가 발생했습니다(Error retrieving metadata from <metadata endpoint>)")이 발생할 경우 메타데이터 엔드포인트에 SSL이 올바르게 설정되어 있고 유효한 SSL 인증서가 연결되어 있는지 확인합니다. 인증서 검증에 대한 자세한 내용은 SSL/TLS 인증서란? 을 참조하십시오. .

IdP에서 시작한 SAML을 사용하는 앱 클라이언트는 SAML로만 로그인할 수 있습니다.

앱 클라이언트에서 IdP 개시 로그인을 지원하는 SAML 2.0 IdP에 대한 지원을 활성화하면 해당 앱 클라이언트에 다른 IdPs SAML 2.0만 추가할 수 있습니다. 사용자 풀의 사용자 디렉토리와 SAML이 아닌 모든 외부 ID 공급자를 이러한 방식으로 구성된 앱 클라이언트에 추가할 수 없습니다.

로그아웃 응답은 POST 바인딩을 사용해야 합니다.

`/saml2/logout`엔드포인트는 `LogoutResponse` `HTTP POST` 요청으로 수락합니다. 사용자 풀은 `HTTP GET` 바인딩을 통한 로그아웃 응답을 수락하지 않습니다.

SAML 사용자 이름의 대/소문자 구분

연동 사용자가 로그인을 시도하면 SAML 자격 증명 공급자 (IdP) 는 사용자의 SAML 어설션에서 Amazon `NameId` Cognito에 고유한 데이터를 전달합니다. Amazon Cognito는 SAML 페더레이션 사용자를 해당 `NameId` 클레임으로 식별합니다. 사용자 풀의 대소문자 구분 설정과 관계없이 Amazon Cognito는 SAML IdP에서 돌아온 페더레이션 사용자가 고유하고 대소문자를 구분하는 클레임을 통과하면 이를 인식합니다. `NameId` `email`과 같은 속성을 `NameId`에 매핑하고 사용자가 자신의 이메일 주소를 변경하는 경우 해당 사용자는 앱에 로그인할 수 없습니다.

변경되지 않는 값을 가진 IdP 속성의 SAML 어설션에서 `NameId`를 매핑합니다.

예를 들어 Carlos라는 사용자가 `NameId` 값인 `Carlos@example.com`을 전달한 Active Directory Federation Services(ADFS) SAML 어설션에서 가져온 대/소문자를 구분하지 않는 사용자 풀에서 사용자 프로필을 갖고 있는 경우 Carlos가 다음에 로그인을 시도할 때 ADFS IdP에서 `NameId` 값인 `carlos@example.com`을 전달합니다. `NameId`는 대/소문자를 정확하게 구분해야 하기 때문에 로그인이 실패합니다.

사용자가 `NameID` 변경 이후에 로그인할 수 없는 경우 사용자 풀에서 해당 사용자 프로필을 삭제합니다. Amazon Cognito는 이러한 사용자가 다음에 로그인할 때 새 사용자 프로필을 생성합니다.

주제

- 사용자 풀에 SAML 자격 증명 공급자 추가 및 관리
- Amazon Cognito 사용자 풀에서 SAML 세션 시작
- SP에서 시작한 SAML 로그인 사용
- IDP에서 시작한 SAML 로그인 사용
- SAML 로그아웃 플로우
- SAML 서명 및 암호화
- SAML ID 공급자 이름 및 식별자
- 타사 SAML ID 공급자 구성

사용자 풀에 SAML 자격 증명 공급자 추가 및 관리

다음 절차는 Amazon Cognito 사용자 풀에서 SAML 공급자를 생성, 수정 및 삭제하는 방법을 보여줍니다.

AWS Management Console

를 사용하여 SAML 자격 증명 공급자 () 를 생성하고 삭제할 수 있습니다. AWS Management Console IdPs

SAML IdP를 생성하려면 먼저 타사 IdP에서 가져온 SAML 메타데이터 문서가 있어야 합니다. 필요한 SAML 메타데이터 문서를 가져오거나 생성하는 방법에 대한 지침은 타사 SAML ID 공급자 구성 섹션을 참조하세요.

사용자 풀에 SAML 2.0 IdP를 구성하려면

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 AWS 자격 증명을 입력합니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. [로그인 환경(Sign-in experience)] 탭을 선택합니다. 페더레이션 로그인(Federated sign-in)을 찾아서 자격 증명 공급자 추가(Add an identity provider)를 선택합니다.
5. SAML IdP를 선택합니다.
6. 제공자 이름을 입력합니다. `identity_provider`요청 매개변수에 이 친숙한 이름을 에 전달할 수 권한 부여 엔드포인트 있습니다.
7. 식별자(Identifiers)를 쉼표로 구분하여 입력합니다. 식별자는 사용자가 로그인할 때 입력한 이메일 주소를 확인한 다음, 사용자 도메인에 해당하는 공급자로 사용자를 보내도록 Amazon Cognito에 지시합니다.

8. 사용자가 로그아웃할 때 Amazon Cognito에서 서명된 로그아웃 요청을 공급자에게 보내도록 하려는 경우 로그아웃 흐름 추가(Add sign-out flow)를 선택합니다. 호스팅 UI를 구성할 때 생성되는 `https://mydomain.us-east-1.amazoncognito.com/saml2/logout` 엔드포인트에 로그아웃 응답을 보내도록 SAML 2.0 IdP를 구성해야 합니다. `saml2/logout` 엔드포인트는 POST 바인딩을 사용합니다.

   > **Note**
   >
   > 이 옵션을 선택하고 SAML IdP가 서명된 로그아웃 요청을 예상하는 경우 사용자 풀의 서명 인증서도 SAML IdP에 제공해야 합니다.
   > SAML IdP가 서명된 로그아웃 요청을 처리하고 사용자를 Amazon Cognito 세션에서 로그아웃합니다.

9. IdP에서 시작한 SAML 로그인 구성을 선택합니다. 보안 모범 사례로 SP에서 시작한 SAML 어설션만 허용을 선택하십시오. 요청하지 않은 SAML 로그인 세션을 안전하게 수락할 수 있도록 환경을 준비했다면 SP에서 시작한 SAML 어설션과 IdP에서 시작한 SAML 어설션 수락을 선택하십시오. 자세한 정보는 Amazon Cognito 사용자 풀에서 SAML 세션 시작을 참조하세요.

10. 메타데이터 문서 소스(Metadata document source)를 선택합니다. IdP가 퍼블릭 URL에서 SAML 메타데이터를 제공하는 경우 메타데이터 문서 URL(Metadata document URL)을 선택하고 해당 퍼블릭 URL을 입력할 수 있습니다. 그렇지 않은 경우 메타데이터 문서 업로드(Upload metadata document)를 선택한 다음, 이전에 공급자로부터 다운로드한 메타데이터 파일을 선택합니다.

    > **Note**
    >
    > 제공자가 파일을 업로드하는 대신 퍼블릭 엔드포인트를 사용하는 경우 메타데이터 문서 URL을 입력하는 것이 좋습니다. Amazon Cognito는 메타데이터 URL에서 메타데이터를 자동으로 새로 고칩니다. 일반적으로 메타데이터 새로 고침은 6시간마다 또는 메타데이터가 만료되기 전 중 더 빠른 시간에 발생합니다.

11. SAML 공급자와 사용자 풀 간의 속성을 매핑하여 SAML 공급자 속성을 사용자 풀의 사용자 프로필에 매핑합니다. 속성 맵에 사용자 풀 필수 속성을 포함합니다.

    예를 들어, 사용자 풀 속성(User pool attribute) `email`을 선택한 경우 IdP의 SAML 어설션에 표시된 대로 SAML 속성 이름을 입력합니다. IdP가 샘플 SAML 어설션을 제공하는 경우 이 샘플 어설션을 사용하여 이름을 찾을 수 있습니다. 일부는 다음과 같은 단순한 이름을 IdPs 사용하는 반면`email`, 다른 일부는 다음과 같은 이름을 사용합니다.

```
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
```

12. 생성을 선택합니다.

API/CLI

다음 명령을 사용하여 SAML 자격 증명 공급자(IdP)를 생성 및 관리합니다.

IdP를 생성하고 메타데이터 문서를 업로드하려면

- AWS CLI: `aws cognito-idp create-identity-provider`

  메타데이터 파일이 포함된 예제: `aws cognito-idp create-identity-provider --user-pool-id us-east-1_EXAMPLE --provider-name=SAML_provider_1 --provider-type SAML --provider-details file:///details.json --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

  여기서 `details.json`에 다음 사항이 포함됩니다.

```
"ProviderDetails": {
      "MetadataFile": "<SAML metadata XML>",
      "IDPSignout" : "true",
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

> **Note**
>
> 에 <SAML metadata XML>해당 " 문자의 인스턴스가 포함된 경우 이스케이프 \ 문자로 추가해야 합니다\".

  메타데이터 URL이 포함된 예제: `aws cognito-idp create-identity-provider --user-pool-id us-east-1_EXAMPLE --provider-name=SAML_provider_1 --provider-type SAML --provider-details MetadataURL=https://`

myidp.example.com/sso/saml/metadata --attribute-mapping *email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress*

- AWS API: CreateIdentityProvider

IdP에 대한 새 메타데이터 문서를 업로드하려면

- AWS CLI: `aws cognito-idp update-identity-provider`

  메타데이터 파일이 포함된 예제: `aws cognito-idp update-identity-provider --user-pool-id` *us-east-1_EXAMPLE* `--provider-name=`*SAML_provider_1* `--provider-details file:///details.json --attribute-mapping` *email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress*

  여기서 `details.json`에 다음 사항이 포함됩니다.

```
"ProviderDetails": {
      "MetadataFile": "<SAML metadata XML>",
      "IDPSignout" : "true",
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

> **Note**
>
> 에 <SAML metadata XML>"문자의 인스턴스가 포함되어 있는 경우 이스케이프 \ 문자로 추가해야 합니다\".

  메타데이터 URL이 포함된 예제: `aws cognito-idp update-identity-provider --user-pool-id` *us-east-1_EXAMPLE* `--provider-name=`*SAML_provider_1* `--provider-details MetadataURL=`*https://myidp.example.com/sso/saml/metadata* `--attribute-mapping` *email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress*

- AWS API: UpdateIdentityProvider

특정 IdP에 대한 정보를 가져오려면

- AWS CLI: `aws cognito-idp describe-identity-provider`

  ```
  aws cognito-idp describe-identity-provider --user-pool-id us-east-1_EXAMPLE --provider-name=SAML_provider_1
  ```

- AWS API: DescribeIdentityProvider

모두에 대한 정보를 나열하려면 IdPs

- AWS CLI: `aws cognito-idp list-identity-providers`

  예제: `aws cognito-idp list-identity-providers --user-pool-id us-east-1_EXAMPLE --max-results 3`

- AWS API: ListIdentityProviders

IdP를 삭제하려면

- AWS CLI: `aws cognito-idp delete-identity-provider`

  ```
  aws cognito-idp delete-identity-provider --user-pool-id us-east-1_EXAMPLE --provider-name=SAML_provider_1
  ```

- AWS API: DeleteIdentityProvider

사용자 풀을 신뢰 당사자로 추가하도록 SAML IdP를 설정하려면

- 사용자 풀 서비스 공급자 URN은 `urn:amazon:cognito:sp:us-east-1_EXAMPLE`입니다. Amazon Cognito는 SAML 응답에서 이 URN과 일치하는 사용자 제한 값을 요구합니다. IdP에서 SP 간 응답 메시지에 다음 POST 바인딩 엔드포인트를 사용하도록 IdP를 구성합니다.

  ```
  https://mydomain.us-east-1.amazoncognito.com/saml2/idpresponse
  ```

- SAML IdP는 SAML 어설션의 사용자 풀에 필요한 모든 속성을 `NameID` 채워야 합니다. `NameID`사용자 풀에서 SAML 연동 사용자를 고유하게 식별하는 데 사용됩니다. IdP는 각 사용자의 SAML 이름 ID를 대소문자를 구분하는 일관된 형식으로 전달해야 합니다. 사용자 이름 ID 값에 변동이 생기면 새 사용자 프로필이 생성됩니다.

SAML 2.0 IDP에 서명 인증서를 제공하려면

- IdP가 SAML 로그아웃 요청을 검증하는 데 사용할 수 있는 Amazon Cognito의 공개 키 사본을 다운로드하려면 사용자 풀의 로그인 환경 탭을 선택하고 IdP를 선택한 다음 서명 인증서 보기에서 Download as .crt를 선택합니다.

Amazon Cognito 콘솔을 사용하여 사용자 풀에 설정한 SAML 공급자를 삭제할 수 있습니다.

SAML 공급자 삭제

1. Amazon Cognito 콘솔에 로그인합니다.
2. 탐색 창에서 [사용자 풀(User Pools)]을 선택한 다음 편집할 사용자 풀을 선택합니다.
3. 로그인 경험 탭을 선택하고 페더레이션된 ID 공급자 로그인을 찾으십시오.
4. IdPs 삭제하려는 SAML 옆에 있는 라디오 버튼을 선택합니다.
5. 자격 증명 공급자 삭제(Delete identity provider) 메시지가 표시되면 SAML 공급자의 이름을 입력하여 삭제를 확인하고 삭제(Delete)를 선택합니다.

Amazon Cognito 사용자 풀에서 SAML 세션 시작

Amazon Cognito는 서비스 공급자 개시 (SP 시작) 싱글 사인온 (SSO) 및 IdP 개시 SSO를 지원합니다. 가장 좋은 보안 방법은 사용자 풀에 SP에서 시작한 SSO를 구현하는 것입니다. SAML V2.0 Technical Overview(SAML V2.0 기술 개요)의 섹션 5.1.2에서는 SP 시작 SSO에 대해 설명합니다. Amazon Cognito는 앱의 자격 증명 공급자(IdP)입니다. 이 앱은 인증된 사용자에 대한 토큰을 검색하는 서비스 공급자(SP)입니다. 그러나 서드 파티 IdP를 사용하여 사용자를 인증할 때 Amazon Cognito가 SP입니다. SAML 2.0 사용자가 SP에서 시작한 흐름으로 인증할 때는 항상 먼저 Amazon Cognito에 요청하고 인증을 위해 IdP로 리디렉션해야 합니다.

일부 엔터프라이즈 사용 사례의 경우 내부 애플리케이션에 대한 액세스는 엔터프라이즈 IdP가 호스팅하는 대시보드의 북마크에서 시작됩니다. 사용자가 북마크를 선택하면 IdP가 SAML 응답을 생성하고 SP로 전송하여 해당 애플리케이션을 사용하여 사용자를 인증합니다.

IdP에서 시작하는 SSO를 지원하도록 사용자 풀에서 SAML IdP를 구성할 수 있습니다. IdP 개시 인증을 지원하는 경우 Amazon Cognito는 SAML 요청으로 인증을 시작하지 않기 때문에 수신한 SAML 응답을 요청했는지 확인할 수 없습니다. SP가 시작한 SSO에서 Amazon Cognito는 원래 요청에 대해 SAML 응답을 검증하는 상태 파라미터를 설정합니다. SP에서 시작한 로그인을 사용하면 사이트 간 요청 위조 (CSRF) 를 방지할 수도 있습니다.

사용자가 사용자 풀 호스팅 UI와 상호 작용하지 못하게 하려는 환경에서 SP에서 시작하는 SAML을 구축하는 방법에 대한 예는 을 참조하십시오. 예제 시나리오: 엔터프라이즈 대시보드에서 Amazon Cognito 앱을 북마크하기

주제

- 예제 시나리오: 엔터프라이즈 대시보드에서 Amazon Cognito 앱을 북마크하기

예제 시나리오: 엔터프라이즈 대시보드에서 Amazon Cognito 앱을 북마크하기

Amazon Cognito 사용자 풀에 웹 애플리케이션에 대한 SSO 액세스를 제공하는 SAML 또는 OIDC IdP 대시보드에서 북마크를 생성할 수 있습니다. 사용자가 호스팅 UI로 로그인할 필요가 없는 방식으로 Amazon Cognito에 연결할 수 있습니다. 이렇게 하려면 권한 부여 엔드포인트 Amazon Cognito 사용자 풀로 리디렉션되는 로그인 북마크를 포털에 다음 형식으로 추가하십시오.

```
https://mydomain.us-east-1.amazoncognito.com/authorize?
response_type=code&identity_provider=MySAMLIdP&client_id=1example23456789&redire
www.example.com
```

> **Note**
>
> 권한 부여 엔드포인트에 대한 요청의 `identity_provider` 파라미터 대신 `idp_identifier` 파라미터를 사용할 수도 있습니다. IdP 식별자는 사용자 풀에서 ID 공급자를 생성할 때 구성할 수 있는 대체 이름 또는 이메일 도메인입니다. SAML ID 공급자 이름 및 식별자를 참조하세요.

`/authorize`에 대한 요청에서 적절한 파라미터를 사용하면 Amazon Cognito는 SP 시작 로그인 흐름을 자동으로 시작하고 사용자를 IdP로 로그인하도록 리디렉션합니다.

시작하려면 사용자 풀에 SAML IdP를 추가하세요. 로그인에 SAML IdP를 사용하고 앱의 URL을 권한 부여된 콜백 URL로 사용하는 앱 클라이언트를 생성합니다. 앱 클라이언트에 대한 자세한 내용은 사용자 풀 앱 클라이언트 섹션을 참조하세요.

이 인증된 액세스 권한을 포털에 배포하기 전에 호스팅된 UI에서 SP가 시작한 앱 로그인을 테스트하세요. Amazon Cognito에서 SAML IdP를 구성하는 방법에 대한 자세한 내용은 타사 SAML ID 공급자 구성 섹션을 참조하세요.

다음 다이어그램은 IdP에서 시작된 SSO를 에뮬레이트하는 인증 흐름을 보여줍니다. 사용자는 회사 포털의 링크를 통해 Amazon Cognito를 사용하여 인증할 수 있습니다.

요구 사항을 충족한 후에는 또는 매개변수를 권한 부여 엔드포인트 포함하는 북마크를 생성하십시오. `identity_provider` `idp_identifier` 사용자 인증은 다음과 같이 진행됩니다.

1. 사용자가 SSO IdP 대시보드에 로그인합니다. 사용자가 액세스할 수 있는 권한이 있는 엔터프라이즈 애플리케이션이 이 대시보드를 채웁니다.
2. 사용자가 Amazon Cognito를 사용하여 인증하는 애플리케이션에 대한 링크를 선택합니다. 많은 SSO 포털에서 사용자 지정 앱 링크를 추가할 수 있습니다. SSO 포털의 퍼블릭 URL에 대한 링크를 생성하는 데 사용할 수 있는 모든 기능이 작동합니다.
3. SSO 포털의 사용자 지정 앱 링크가 사용자를 사용자 풀 권한 부여 엔드포인트로 안내합니다. 이 링크에는 `response_type`, `client_id`, `redirect_uri` 및 `identity_provider`에 대한 파라미터가 포함됩니다. `identity_provider` 파라미터는 사용자 풀에서 IdP에 부여한 이름입니다. `identity_provider` 파라미터 대신 `idp_identifier` 파라미터를 사용할 수도 있습니다. 사용자는 `idp_identifier` 또는 `identity_provider` 매개변수가 포함된 링크를 통해 페더레이션 엔드포인트에 액세스합니다. 이 사용자는 로그인 페이지를 건너뛰고 IdP 인증으로 바로 이동합니다. SAML IdPs 이름 지정에 대한 자세한 내용은 을 참조하십시오. SAML ID 공급자 이름 및 식별자

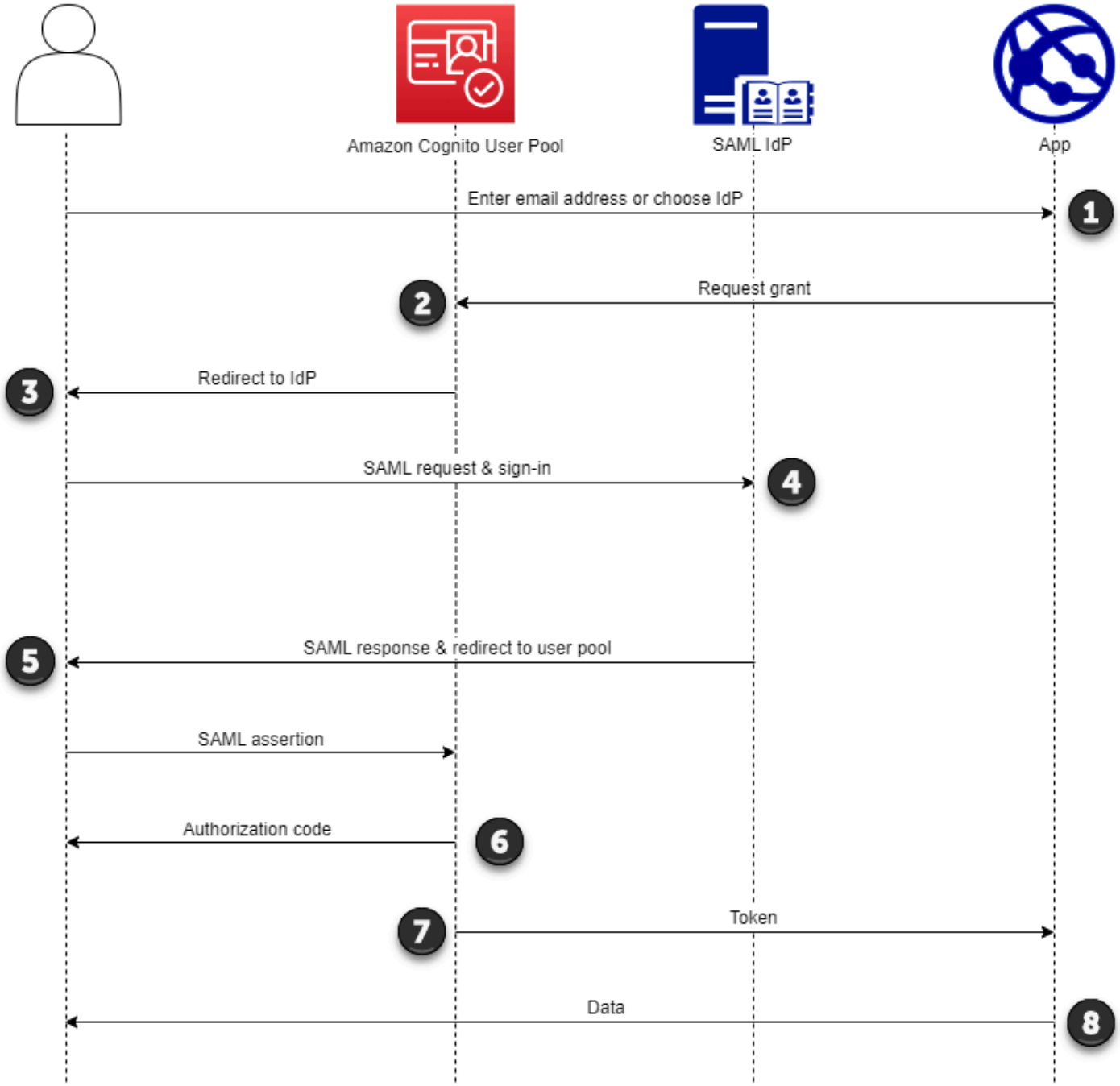
   예제 URL

   ```
   https://mydomain.us-east-1.amazoncognito.com/authorize?
   response_type=code&
   identity_provider=MySAMLIdP&
   client_id=1example23456789&
   redirect_uri=https://www.example.com
   ```
4. Amazon Cognito는 SAML 요청을 사용하여 사용자 세션을 IdP로 리디렉션합니다.
5. 사용자가 대시보드에 로그인할 때 IdP로부터 세션 쿠키를 수신했을 수도 있습니다. IdP는 이 쿠키를 사용하여 사용자를 자동으로 검증하고 해당 사용자를 SAML 응답과 함께 Amazon Cognito `idpresponse` 엔드포인트로 리디렉션합니다. 활성 세션이 없으면 IdP가 SAML 응답을 게시하기 전에 사용자를 다시 인증합니다.
6. Amazon Cognito는 SAML 응답을 검증하고 SAML 어설션을 기반으로 사용자 프로필을 생성하거나 업데이트합니다.
7. Amazon Cognito는 권한 부여 코드를 사용하여 사용자를 내부 앱으로 리디렉션합니다. 내부 앱 URL을 앱 클라이언트에 대해 권한 부여된 리디렉션 URL로 구성했습니다.
8. 앱이 권한 부여 코드를 Amazon Cognito 토큰과 교환합니다. 자세한 정보는 Token 엔드포인트을 참조하세요.

SP에서 시작한 SAML 로그인 사용

사용자 풀에 service-provider-initiated (SP에서 시작한) 로그인을 구현하는 것이 가장 좋습니다. Amazon Cognito는 사용자 세션을 시작하고 사용자를 IdP로 리디렉션합니다. 이 방법을 사용하면 로그인 요청을 제시하는 사람을 가장 효과적으로 제어할 수 있습니다. 특정 조건에서는 IdP 개시 로그인을 허용할 수도 있습니다. 자세한 정보는 Amazon Cognito 사용자 풀에서 SAML 세션 시작을 참조하세요.

다음 프로세스는 사용자가 SAML 공급자를 통해 사용자 풀에 로그인하는 방법을 보여줍니다.



1. 사용자는 로그인 페이지에 이메일 주소를 입력합니다. 사용자가 IdP로 리디렉션되는지 확인하려면 맞춤형 앱에서 이메일 주소를 수집하거나 웹 보기에서 호스팅된 UI를 호출할 수 있습니다. 이메일 주소 목록을 IdPs 표시하거나 이메일 주소를 입력하라는 메시지만 표시하도록 호스팅된 UI를 구성할 수 있습니다.
2. 앱은 사용자 풀 리디렉션 엔드포인트를 호출하고 앱에 해당하는 클라이언트 ID와 사용자에 해당하는 IdP ID를 사용하여 세션을 요청합니다.

3. Amazon Cognito는 요소에 SAML 요청 (선택적으로 서명됨) 을 통해 사용자를 IdP로 리디렉션합니다. `AuthnRequest`
4. IdP는 대화식으로 또는 브라우저 쿠키의 기억된 세션을 사용하여 사용자를 인증합니다.
5. IdP는 선택적으로 암호화된 SAML 어설션을 POST 페이로드에 포함하는 사용자 풀 SAML 응답 엔드포인트로 사용자를 리디렉션합니다.

   > **Note**
   >
   > Amazon Cognito는 5분 내에 응답을 받지 못한 세션을 취소하고 사용자를 호스팅된 UI로 리디렉션합니다. 사용자가 이러한 결과를 경험하면 오류 메시지를 받게 됩니다. `Something went wrong`

6. Amazon Cognito는 SAML 어설션을 확인하고 응답의 클레임으로부터 사용자 속성을 매핑한 후 사용자 풀에서 사용자 프로필을 내부적으로 생성하거나 업데이트합니다. 일반적으로 사용자 풀은 사용자의 브라우저 세션에 인증 코드를 반환합니다.
7. 사용자가 앱에 인증 코드를 제시하면 앱은 코드를 JSON 웹 토큰 (JWT) 으로 교환합니다.
8. 앱은 사용자의 ID 토큰을 인증으로 받아 처리하고, 액세스 토큰으로 리소스에 승인된 요청을 생성하고, 새로 고침 토큰을 저장합니다.

사용자가 인증을 받고 권한 부여 코드를 받으면 사용자 풀은 ID, 액세스 및 새로 고침 토큰을 반환합니다. ID 토큰은 OIDC 기반 ID 관리를 위한 인증 개체입니다. 액세스 토큰은 OAuth 2.0 범위의 권한 부여 객체입니다. 새로 고침 토큰은 사용자의 현재 토큰이 만료되었을 때 새 ID 및 액세스 토큰을 생성하는 객체입니다. 사용자 풀 앱 클라이언트에서 사용자 토큰의 기간을 구성할 수 있습니다.

새로 고침 토큰의 기간을 선택할 수도 있습니다. 사용자의 새로고침 토큰이 만료된 후에는 다시 로그인해야 합니다. SAML IdP를 통해 인증한 경우 사용자의 세션 기간은 IdP와의 세션 만료가 아니라 토큰 만료를 기준으로 설정됩니다. 앱은 각 사용자의 새로고침 토큰을 저장하고 만료되면 세션을 갱신해야 합니다. 호스팅된 UI는 1시간 동안 유효한 브라우저 쿠키에 사용자 세션을 유지합니다.

IDP에서 시작한 SAML 로그인 사용

ID 공급자가 IdP에서 시작한 SAML 2.0 로그인을 구성하면 에서 세션을 시작할 필요 없이 사용자 풀 도메인의 `saml2/idpresponse` 엔드포인트에 SAML 어설션을 제시할 수 있습니다. 권한 부여 엔드포인트 이 구성의 사용자 풀은 요청된 앱 클라이언트가 지원하는 사용자 풀 외부 ID 공급자로부터 IdP가 시작한 SAML 어설션을 수락합니다. 다음 단계는 IdP에서 시작한 SAML 2.0 공급자를 구성하고 이를 사용하여 로그인하는 전체 프로세스를 설명합니다.

1. 사용자 풀 및 앱 클라이언트를 만들거나 지정합니다.
2. 사용자 풀에 SAML 2.0 IdP를 생성하십시오.
3. IdP 시작을 지원하도록 IdP를 구성하십시오. IdP가 시작한 SAML은 다른 SSO 공급자에게 적용되지 않는 보안 고려 사항을 도입합니다. 따라서 IdP에서 시작한 로그인으로 SAML 공급자를 사용하는 앱 클라이언트에는 사용자 풀 자체를 포함하여 비 IdPs SAML을 추가할 수 없습니다.
4. IdP에서 시작한 SAML 공급자를 사용자 풀의 앱 클라이언트와 연결합니다.
5. 사용자를 SAML IdP의 로그인 페이지로 안내하고 SAML 어설션을 검색하십시오.
6. SAML 어설션을 통해 사용자를 사용자 풀 `saml2/idpresponse` 엔드포인트로 안내하세요.
7. JSON 웹 토큰 (JWT) 을 받으세요.

사용자 풀에서 요청하지 않은 SAML 어설션을 수락하려면 앱 보안에 미치는 영향을 고려해야 합니다. 요청 스푸핑 및 CSRF 시도는 IdP에서 시작한 요청을 수락할 때 발생할 가능성이 높습니다. 사용자 풀은 IdP가 시작한 로그인 세션을 확인할 수 없지만 Amazon Cognito는 요청 파라미터와 SAML 어설션을 검증합니다.

또한 SAML 어설션은 클레임을 포함하지 않아야 하며 이전 6분 이내에 발행되었어야 합니다. `InResponseTo`

IdP에서 시작한 SAML이 포함된 요청을 제출해야 합니다. `/saml2/idpresponse` SP에서 시작하고 호스팅된 UI 권한 부여 요청의 경우 요청된 앱 클라이언트를 식별하는 매개 변수, 범위, 리디렉션 URI 및 기타 세부 정보를 요청의 쿼리 문자열 매개 변수로 제공해야 합니다. `HTTP GET` 하지만 IdP에서 시작한 SAML 어설션의 경우 요청 세부 정보를 요청 본문의 `RelayState` 파라미터 형식으로 지정해야 합니다. `HTTP POST` 요청 본문에는 SAML 어설션도 파라미터로 포함되어야 합니다. `SAMLResponse`

다음은 IdP에서 시작한 SAML 공급자에 대한 요청 예시입니다.

```
POST /saml2/idpresponse HTTP/1.1
User-Agent: USER_AGENT
Accept: */*
Host: example.auth.us-east-1.amazoncognito.com
Content-Type: application/x-www-form-urlencoded

SAMLResponse=[Base64-encoded SAML assertion]&RelayState=identity_provider
%3DMySAMLIdP%26client_id%3D1example23456789%26redirect_uri%3Dhttps%3A%2F
%2Fwww.example.com%26response_type%3Dcode%26scope%3Demail%2Bopenid%2Bphone

HTTP/1.1 302 Found
```

```
Date: Wed, 06 Dec 2023 00:15:29 GMT
Content-Length: 0
x-amz-cognito-request-id: 8aba6eb5-fb54-4bc6-9368-c3878434f0fb
Location: https://www.example.com?code=[Authorization code]
```

AWS Management Console

IdP에서 시작하는 SAML을 위한 IdP를 구성하려면

1. 사용자 풀, 앱 클라이언트, SAML ID 공급자를 생성합니다.
2. 앱 클라이언트에서 모든 소셜 및 OIDC ID 제공자를 분리하십시오 (연결되어 있는 경우).
3. 사용자 풀의 로그인 경험 탭으로 이동합니다.
4. 페더레이션 ID 공급자 로그인에서 SAML 공급자를 편집하거나 추가합니다.
5. IdP에서 시작한 SAML 로그인에서 SP에서 시작한 SAML 어설션과 IdP에서 시작한 SAML 어설션 수락을 선택합니다.
6. 변경 사항 저장를 선택합니다.

API/CLI

IdP에서 시작하는 SAML을 위한 IdP를 구성하려면

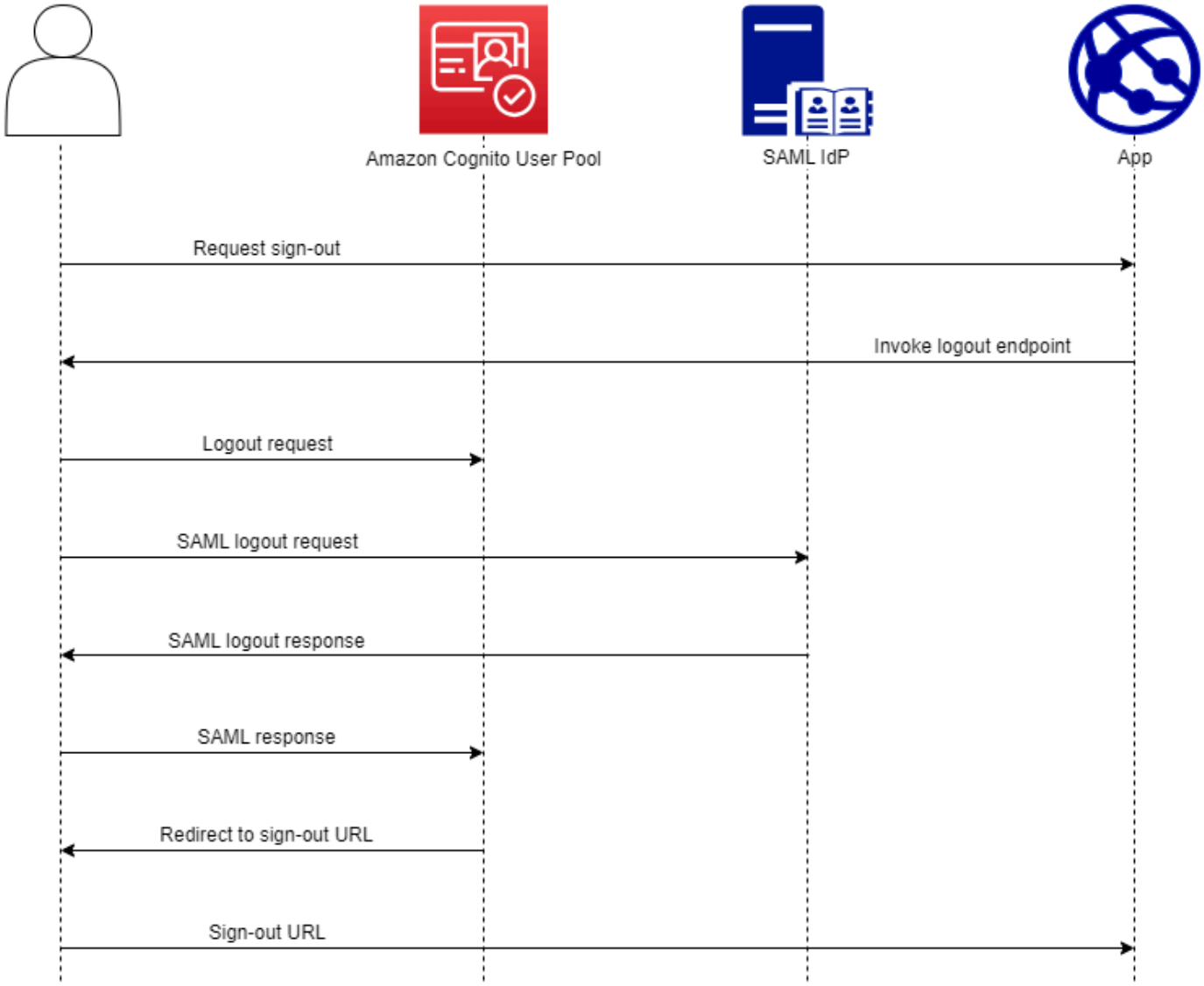
또는 API 요청의 `IDPInit` 파라미터를 사용하여 IdP에서 시작하는 SAML을 CreateIdentityProvider구성합니다. UpdateIdentityProvider 다음은 IdP에서 시작한 SAML을 지원하는 `ProviderDetails` IdP의 예입니다.

```
"ProviderDetails": {
      "MetadataURL" : "https://myidp.example.com/saml/metadata",
      "IDPSignout" : "true",
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

SAML 로그아웃 플로우

아마존 코그니토는 SAML 2.0 싱글 로그아웃을 지원합니다. 로그아웃 흐름을 지원하도록 SAML IdP를 구성하면 Amazon Cognito는 서명된 SAML 로그아웃 요청으로 사용자를 IdP로 리디렉션합니다.

Amazon Cognito는 IdP 메타데이터의 `SingleLogoutService` URL에서 리디렉션 위치를 결정합니다. Amazon Cognito는 사용자 풀 서명 인증서를 사용하여 로그아웃 요청에 서명합니다.



SAML 세션이 있는 사용자를 사용자 풀 `/logout` 엔드포인트로 리디렉션하면 Amazon Cognito는 다음 요청으로 SAML 사용자를 IdP 메타데이터에 지정된 SLO 엔드포인트로 리디렉션합니다.

```
https://[SingleLogoutService endpoint]?
SAMLRequest=[encoded SAML request]&
RelayState=[RelayState]&
SigAlg=http://www.w3.org/2001/04/xmldsig-more#rsa-sha256&
Signature=[User pool RSA signature]
```

그러면 사용자는 자신의 LogoutResponse IdP로부터 받은 ID를 가지고 saml2/logout 엔드포인트로 돌아옵니다. IdP가 LogoutResponse 요청을 보내야 합니다. HTTP POST 그러면 Amazon Cognito는 초기 로그아웃 요청에서 리디렉션 대상으로 리디렉션합니다.

SAML 공급자가 둘 이상이 포함된 a를 LogoutResponse 보낼 수 있습니다. AuthnStatement 이 유형의 AuthnStatement 응답에서 첫 번째 입력은 원래 사용자를 인증한 SAML 응답의 입력과 일치해야 합니다. sessionIndex sessionIndex 다른 세션에 있는 AuthnStatement 경우 Amazon sessionIndex Cognito는 세션을 인식하지 못하므로 사용자가 로그아웃하지 않습니다.

AWS Management Console

SAML 로그아웃을 구성하려면

1. 사용자 풀, 앱 클라이언트, SAML IdP를 생성합니다.
2. SAML ID 공급자를 만들거나 편집할 때는 ID 제공자 정보에서 로그아웃 흐름 추가라는 제목의 체크박스를 선택합니다.
3. 사용자 풀의 로그인 경험 탭에 있는 페더레이션 ID 공급자 로그인에서 IdP를 선택하고 서명 인증서를 찾습니다.
4. .crt로 다운로드를 선택합니다.
5. SAML 단일 로그아웃 및 요청 서명을 지원하도록 SAML 공급자를 구성하고 사용자 풀 서명 인증서를 업로드하십시오. IdP는 사용자 풀 /saml2/logout 도메인에서 리디렉션해야 합니다.

API/CLI

SAML 로그아웃을 구성하려면

또는 API 요청의 IDPSignout 파라미터를 사용하여 단일 로그아웃을 CreateIdentityProvider구성합니다. UpdateIdentityProvider 다음은 SAML 단일 로그아웃을 지원하는 ProviderDetails IdP의 예입니다.

```
"ProviderDetails": {
      "MetadataURL" : "https://myidp.example.com/saml/metadata",
      "IDPSignout" : "true",,
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

SAML 서명 및 암호화

Amazon Cognito는 로그인 및 로그아웃을 위한 서명된 SAML 요청과 암호화된 SAML 응답을 지원합니다. 사용자 풀 SAML 작업 중 모든 암호화 작업은 Amazon Cognito가 생성하는 user-pool-provided 키를 사용하여 서명과 암호문을 생성해야 합니다. 현재는 요청에 서명하거나 외부 키를 사용하여 암호화된 어설션을 수락하도록 사용자 풀을 구성할 수 없습니다.

> **Note**
>
> 사용자 풀 인증서는 10년 동안 유효합니다. Amazon Cognito는 일년에 한 번 사용자 풀에 대한 새로운 서명 및 암호화 인증서를 생성합니다. Amazon Cognito는 서명 인증서를 요청하면 가장 최신 인증서를 반환하고 가장 최근의 서명 인증서로 요청에 서명합니다. IdP는 만료되지 않은 모든 사용자 풀 암호화 인증서를 사용하여 SAML 어설션을 암호화할 수 있습니다. 이전 인증서는 전체 기간 동안 계속 유효합니다. 공급자 구성의 인증서를 매년 업데이트하는 것이 가장 좋습니다.

주제

IdP로부터 암호화된 SAML 응답 수락

Amazon Cognito와 IdP는 사용자가 로그인하고 로그아웃할 때 SAML 응답의 기밀을 설정할 수 있습니다. Amazon Cognito는 사용자 풀에서 구성한 각 외부 SAML 공급자에게 공개-사설 RSA 키 쌍과 인증서를 할당합니다. 사용자 풀 SAML 공급자에 대한 응답 암호화를 활성화하는 경우 암호화된 SAML 응답을 지원하는 IdP에 인증서를 업로드해야 합니다. IdP가 제공된 키로 모든 SAML 어설션을 암호화하기 시작하기 전에는 SAML IdP에 대한 사용자 풀 연결이 작동하지 않습니다.

다음은 암호화된 SAML 로그인의 흐름에 대한 개요입니다.

1. 사용자가 로그인을 시작하고 SAML IdP를 선택합니다.
2. 사용자 풀은 SAML 로그인 요청을 통해 사용자를 SAML IdP로 권한 부여 엔드포인트 리디렉션합니다. 사용자 풀은 선택적으로 이 요청과 함께 IdP의 무결성 검증을 가능하게 하는 서명을 첨부할 수 있습니다. SAML 요청에 서명하려면 사용자 풀이 서명 인증서의 공개 키로 서명한 요청을 수락하도록 IdP를 구성해야 합니다.

3. SAML IdP는 사용자를 로그인시키고 SAML 응답을 생성합니다. IdP는 공개 키로 응답을 암호화하고 사용자를 사용자 풀 엔드포인트로 리디렉션합니다. `/saml2/idpresponse` IdP는 SAML 2.0 사양에 정의된 대로 응답을 암호화해야 합니다. 자세한 내용은 OASIS 보안 어설션 마크업 언어 (SAML) V2.0용 어설션 및 프로토콜을 참조하십시오`Element <EncryptedAssertion>`.
4. 사용자 풀은 개인 키를 사용하여 SAML 응답의 암호문을 해독하고 사용자를 로그인시킵니다.

> **Important**
>
> 사용자 풀의 SAML IdP에 대한 응답 암호화를 활성화하는 경우 IdP는 공급자별 공개 키로 모든 응답을 암호화해야 합니다. Amazon Cognito는 암호화를 지원하도록 구성한 SAML 외부 IdP의 암호화되지 않은 SAML 응답을 수락하지 않습니다.

사용자 풀의 모든 외부 SAML IdP는 응답 암호화를 지원할 수 있으며 각 IdP는 고유한 키 쌍을 받습니다.

AWS Management Console

SAML 응답 암호화를 구성하려면

1. 사용자 풀, 앱 클라이언트, SAML IdP를 생성합니다.
2. SAML ID 공급자를 만들거나 편집할 때는 요청 서명 및 응답 암호화에서 이 공급자의 암호화된 SAML 어설션 필요라는 제목의 확인란을 선택합니다.
3. 사용자 풀의 로그인 경험 탭에 있는 페더레이션 ID 공급자 로그인에서 SAML IdP를 선택하고 암호화 인증서 보기를 선택합니다.
4. .crt로 다운로드를 선택하고 다운로드한 파일을 SAML IdP에 제공합니다. 인증서의 키를 사용하여 SAML 응답을 암호화하도록 SAML IdP를 구성합니다.

API/CLI

SAML 응답 암호화를 구성하려면

CreateIdentityProvider또는 UpdateIdentityProviderAPI 요청의 `EncryptedResponses` 파라미터를 사용하여 응답 암호화를 구성합니다. 다음은 요청 서명을 지원하는 `ProviderDetails` IdP의 예입니다.

```
"ProviderDetails": {
```

```
      "MetadataURL" : "https://myidp.example.com/saml/metadata",
      "IDPSignout" : "true",
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

SAML 요청에 서명

IdP에 대한 SAML 2.0 요청의 무결성을 입증할 수 있는 기능은 Amazon Cognito SP에서 시작한 SAML 로그인의 보안 이점입니다. 도메인이 있는 각 사용자 풀은 사용자 풀 X.509 서명 인증서를 받습니다. 이 인증서의 공개 키를 사용하면 사용자 풀은 사용자가 SAML IdP를 선택할 때 사용자 풀이 생성하는 로그아웃 요청에 암호화 서명을 적용합니다. 선택적으로 SAML 로그인 요청에 서명하도록 앱 클라이언트를 구성할 수 있습니다. SAML 요청에 서명하면 IdP는 요청의 XML 메타데이터에 있는 서명이 제공하는 사용자 풀 인증서의 공개 키와 일치하는지 확인할 수 있습니다.

AWS Management Console

SAML 요청 서명을 구성하려면

1. 사용자 풀, 앱 클라이언트, SAML IdP를 생성합니다.
2. SAML ID 공급자를 만들거나 편집할 때는 요청 서명 및 응답 암호화에서 이 공급자에 대한 SAML 요청 서명이라는 제목의 확인란을 선택합니다.
3. 사용자 풀의 로그인 환경 탭에 있는 페더레이션 ID 공급자 로그인에서 서명 인증서 보기를 선택합니다.
4. .crt로 다운로드를 선택하고 다운로드한 파일을 SAML IdP에 제공합니다. 들어오는 SAML 요청의 서명을 확인하도록 SAML IdP를 구성합니다.

API/CLI

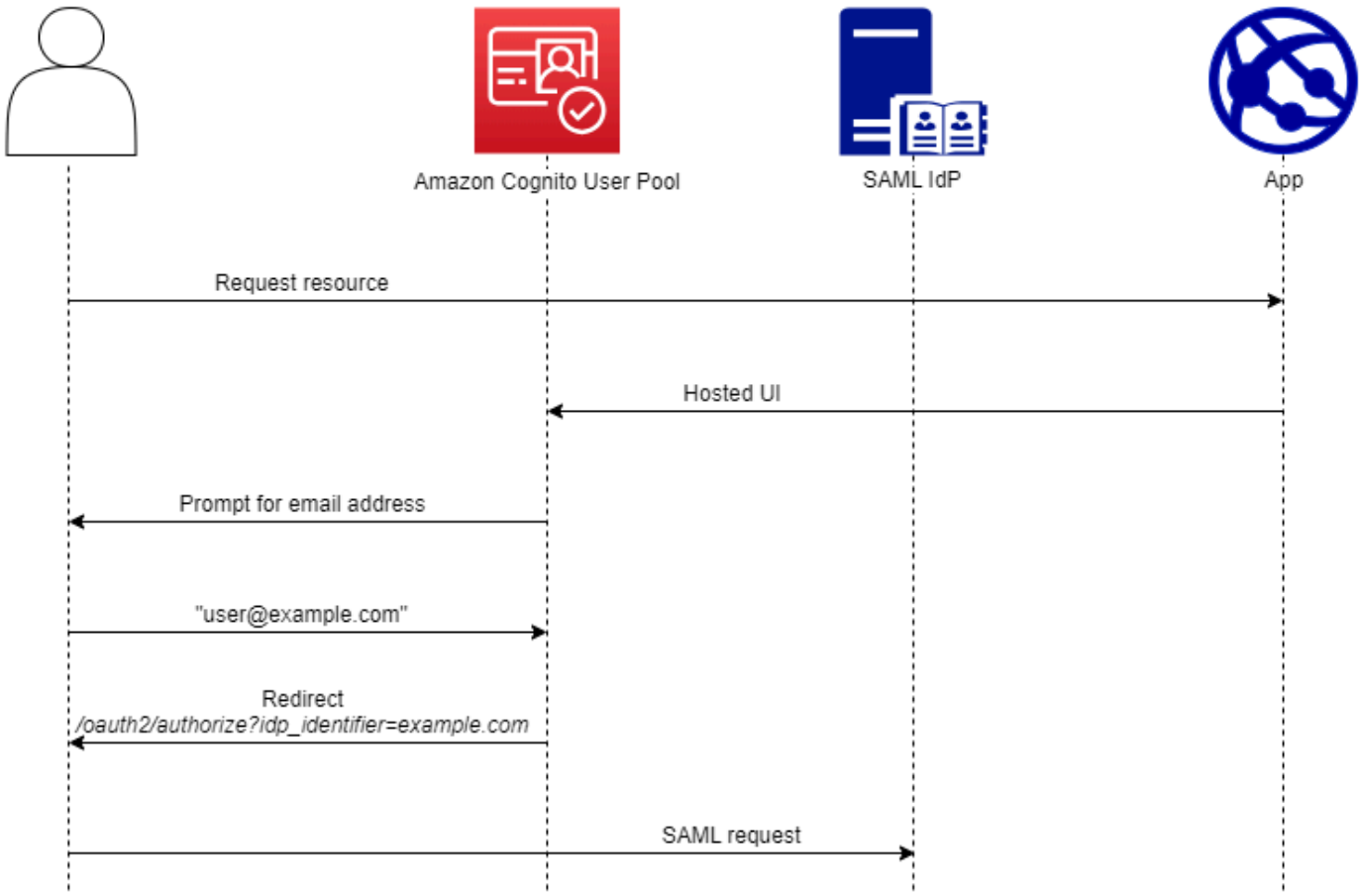
SAML 요청 서명을 구성하려면

CreateIdentityProvider또는 UpdateIdentityProviderAPI 요청의 `RequestSigningAlgorithm` 파라미터를 사용하여 요청 서명을 구성합니다. 다음은 요청 서명을 지원하는 `ProviderDetails` IdP의 예입니다.

```
"ProviderDetails": {
      "MetadataURL" : "https://myidp.example.com/saml/metadata",
```

```
      "IDPSignout" : "true",
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

SAML ID 공급자 이름 및 식별자



SAML ID 공급자 (IdPs) 의 이름을 지정하고 IdP 식별자를 할당하면 SP에서 시작한 로그인 및 로그아웃 요청이 해당 공급자에 전달되는 흐름을 자동화할 수 있습니다. 공급자 이름의 문자열 제약 조건에 대한 자세한 내용은 의 속성을 참조하십시오. `ProviderName` CreateIdentityProvider

또한 SAML 공급자의 식별자를 최대 50개까지 선택할 수 있습니다. 식별자는 사용자 풀의 IdP에 친숙한 이름이며 사용자 풀 내에서 고유해야 합니다. SAML 식별자가 사용자의 이메일 도메인과 일치하는 경우 Amazon Cognito 호스팅 UI는 각 사용자의 이메일 주소를 요청하고, 이메일 주소의 도메인을 평가하여 도메인에 해당하는 IdP로 리디렉션합니다. 동일한 조직이 여러 도메인을 소유할 수 있으므로 단일 IdP에 여러 식별자가 있을 수 있습니다.

이메일 도메인 식별자를 사용하든 사용하지 않든, 멀티 테넌트 앱의 식별자를 사용하여 사용자를 올바른 IdP로 리디렉션할 수 있습니다. 호스팅된 UI를 완전히 우회하려는 경우 사용자에게 제공하는 링크를 사용자 지정하여 사용자가 자신의 IdP로 권한 부여 엔드포인트 직접 리디렉션되도록 할 수 있습니다. 식별자로 사용자를 로그인하고 해당 IdP로 리디렉션하려면 초기 승인 요청의 요청 파라미터에 해당 `idp_identifier=myidp.example.com` 형식의 식별자를 포함하세요.

사용자를 IdP로 안내하는 또 다른 방법은 다음 URL 형식의 IdP 이름으로 매개변수를 `identity_provider` 채우는 것입니다.

```
https://mydomain.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&
identity_provider=MySAMLIdP&
client_id=1example23456789&
redirect_uri=https://www.example.com
```

사용자가 SAML IdP로 로그인하면 IdP는 본문에 SAML 응답을 포함하여 사용자를 엔드포인트로 리디렉션합니다. `HTTP POST /saml2/idpresponse` Amazon Cognito는 SAML 어설션을 처리하고, 응답의 클레임이 기대치를 충족하는 경우 앱 클라이언트 콜백 URL로 리디렉션합니다. 이러한 방식으로 인증을 완료한 사용자는 IdP 및 앱의 웹페이지에만 상호작용한 것입니다.

도메인 형식의 IdP 식별자를 사용하는 Amazon Cognito 호스팅 UI는 로그인 시 이메일 주소를 요청한 다음 이메일 도메인이 IdP 식별자와 일치하면 사용자를 IdP의 로그인 페이지로 리디렉션합니다. 예를 들어, 서로 다른 두 회사의 직원이 로그인해야 하는 앱을 빌드합니다. 첫 번째 회사인 AnyCompany A는 및 을 (를) 소유하고 있습니다. `exampleA.com` `exampleA.co.uk` 두 번째 회사인 AnyCompany B가 소유하고 `exampleB.com` 있습니다. 이 예시에서는 다음과 같이 회사별로 하나씩 두 개를 IdPs 설정했습니다.

- IdP A에는 `exampleA.com` 및 `exampleA.co.uk` 식별자를 정의합니다.
- IdP B에는 `exampleB.com` 식별자를 정의합니다.

앱에서 앱 클라이언트의 호스팅된 UI를 호출하여 각 사용자에게 이메일 주소를 입력하라는 메시지를 표시합니다. Amazon Cognito는 이메일 주소에서 도메인을 추출하고, 도메인 식별자를 사용하여 도메인과 IdP의 상관 관계를 분석하며, 요청 파라미터가 포함된 요청을 통해 사용자를 올바른 IdP로 리디렉션합니다. 권한 부여 엔드포인트 `idp_identifier` 예를 들어, 사용자가 들어오면 `bob@exampleA.co.uk` 그 다음 상호작용하는 페이지는 IdP 로그인 페이지입니다. `https://auth.exampleA.co.uk/sso/saml`

동일한 로직을 독립적으로 구현할 수도 있습니다. 앱에서 사용자 입력을 수집하고 자체 논리에 따라 올바른 IdP와 상호 연결하는 사용자 지정 양식을 만들 수 있습니다. 각 앱 테넌트에 대해 사용자 지정 앱

포털을 생성할 수 있으며, 각 포털은 요청 매개변수의 테넌트 식별자와 함께 권한 부여 엔드포인트로 연결됩니다.

이메일 주소를 수집하고 호스팅된 UI에서 도메인을 파싱하려면 앱 클라이언트에 할당한 각 SAML IdP에 식별자를 하나 이상 할당해야 합니다. 기본적으로 호스팅된 UI 로그인 화면에는 앱 클라이언트에 할당한 각 항목에 IdPs 대한 버튼이 표시됩니다. 하지만 식별자를 성공적으로 할당했다면 호스팅된 UI 로그인 페이지는 다음 이미지와 같이 표시됩니다.

호스팅된 UI에서 도메인을 파싱하려면 도메인을 IdP 식별자로 사용해야 합니다. 앱 클라이언트의 각 IdPs SAML에 모든 유형의 식별자를 할당하면 해당 앱의 호스팅된 UI에 더 이상 IdP 선택 버튼이 표시되지 않습니다. 이메일 파싱이나 사용자 지정 로직을 사용하여 리디렉션을 생성하려는 경우 SAML용 IdP 식별자를 추가하세요. 자동 리디렉션을 생성하고 호스팅된 UI에 목록을 표시하려면 식별자를 할당하지 말고 권한 부여 요청에서 요청 파라미터를 사용하세요. IdPs `identity_provider`

- 앱 클라이언트에 SAML IdP 하나만 할당하는 경우 호스팅 UI 로그인 페이지에 해당 IdP로 로그인하는 버튼이 표시됩니다.
- 앱 클라이언트용으로 활성화하는 모든 SAML IdP에 식별자를 할당하면 호스팅된 UI 로그인 페이지에 이메일 주소를 입력하라는 사용자 입력 프롬프트가 나타납니다.
- 여러 개의 IDP가 IdPs 있고 모든 IDP에 식별자를 할당하지 않은 경우 호스팅된 UI 로그인 페이지에 할당된 각 IdP로 로그인할 수 있는 버튼이 표시됩니다.
- 식별자를 IdPs 할당했고 호스팅된 UI에 IdP 버튼 선택이 표시되도록 하려면 식별자가 없는 새 IdP를 앱 클라이언트에 추가하거나 새 앱 클라이언트를 생성하십시오. 기존 IdP를 삭제하고 식별자 없이 다시 추가할 수도 있습니다. 새 IdP를 생성하면 SAML 사용자가 새 사용자 프로필을 생성합니다. 이렇게 활성 사용자가 중복되면 IdP 구성을 변경한 달의 청구에 영향을 미칠 수 있습니다.

IdP 설정에 대한 자세한 내용은 사용자 풀의 자격 증명 공급자 구성 섹션을 참조하세요.

타사 SAML ID 공급자 구성

Amazon Cognito 사용자 풀의 페더레이션과 함께 작동하도록 타사 SAML 2.0 ID 공급자 (IdP) 솔루션을 구성하려면 다음 어설션 소비자 서비스 (ACS) URL로 리디렉션되도록 SAML IdP를 구성해야 합니다. `https://mydomain.us-east-1.amazoncognito.com/saml2/idpresponse` 사용자 풀에 Amazon Cognito 도메인이 있는 경우 Amazon Cognito 콘솔(Amazon Cognito console)에 있는 사용자 풀의 앱 통합(App integration) 탭에서 사용자 풀 도메인 경로를 찾을 수 있습니다.

일부 IdPs SAML에서는 오디언스 URI 또는 SP 엔티티 `urn` ID라고도 하는 양식을 양식으로 제공해야 합니다. `urn:amazon:cognito:sp:us-east-1_EXAMPLE` Amazon Cognito 콘솔의 사용자 풀 개요에서 사용자 풀 ID를 찾을 수 있습니다.

또한 사용자 풀에서 필수 속성으로 지정한 모든 속성에 값을 제공하도록 SAML IdP를 구성해야 합니다. 일반적으로 `email` 는 사용자 풀의 필수 속성으로, 이 경우 SAML IdP는 SAML 어설션에 특정 형태의 `email` 클레임을 제공해야 하며 클레임을 해당 공급자의 속성에 매핑해야 합니다.

Amazon Cognito 사용자 풀과의 페더레이션 설정을 시작하려면 타사 SAML 2.0 IdP 솔루션에 대한 다음 구성 정보를 참조하십시오. 최신 정보는 공급자의 설명서를 직접 참조하십시오.

SAML 요청에 서명하려면 사용자 풀 서명 인증서로 서명한 요청을 신뢰하도록 IdP를 구성해야 합니다. 암호화된 SAML 응답을 수락하려면 사용자 풀에 대한 모든 SAML 응답을 암호화하도록 IdP를 구성해야 합니다. 제공자는 이러한 기능을 구성하는 방법에 대한 설명서를 제공합니다. Microsoft의 예를 보려면 Microsoft Entra SAML 토큰 암호화 구성을 참조하십시오.

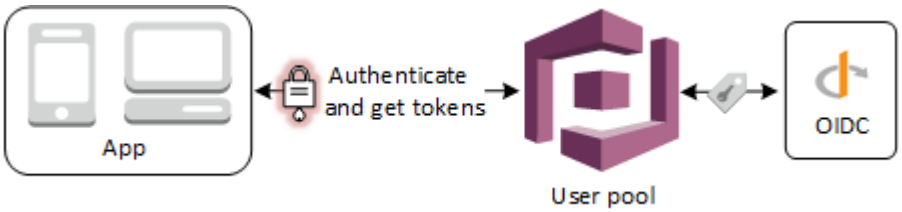
> **Note**
>
> Amazon Cognito에는 자격 증명 공급자 메타데이터 문서만 필요합니다. 공급자가 SAML 2.0과의 AWS 계정 페더레이션을 위한 구성 정보를 제공할 수 있습니다. 이 정보는 Amazon Cognito 통합과 관련이 없습니다.

Solution	추가 정보
Microsoft AD FS(Active Directory Federation Services)	페더레이션 메타데이터 탐색기
Okta	SAML 앱 통합을 위한 IdP 메타데이터 및 SAML 서명 인증서를 다운로드하는 방법
Auth0	Auth0을 SAML ID 공급자로 구성합니다.
핑 아이덴티티 () PingFederate	에서 SAML 메타데이터 내보내기 PingFederate
JumpCloud	SAML 컨피그레이션 노트
SecureAuth	SAML 애플리케이션 통합

사용자 풀과 함께 OIDC ID 공급자 사용

OpenID Connect (OIDC) ID 공급자 (IdPs) 계정을 이미 보유한 사용자가 가입 단계를 건너뛰고 기존 계정을 사용하여 애플리케이션에 로그인할 수 있도록 할 수 있습니다. Amazon Cognito는 기본 제공 호스

트된 웹 UI를 사용하여 모든 인증된 사용자에 대한 토큰 처리 및 관리 기능을 제공합니다. 이렇게 하면 백엔드 시스템을 한 세트의 사용자 풀 토큰에서 표준화할 수 있습니다.



> **Note**
>
> 서드 파티(페더레이션)를 통한 로그인을 Amazon Cognito 사용자 풀에서 사용할 수 있습니다. 이 기능은 Amazon Cognito 자격 증명 풀(페더레이션 자격 증명)을 통한 페더레이션과 무관합니다.

를 통해 또는 사용자 풀 API 방법을 사용하여 사용자 풀에 OIDC AWS Management Console IdP를 추가할 수 있습니다. AWS CLICreateIdentityProvider

주제

- 필수 조건
- 1단계: OIDC IdP로 등록
- 2단계: 사용자 풀에 OIDC IdP 추가
- 3단계: OIDC IdP 구성 테스트
- OIDC 사용자 풀 IdP 인증 흐름

필수 조건

시작하려면 다음이 필요합니다.

- 앱 클라이언트와 사용자 풀 도메인이 있는 사용자 풀. 자세한 내용은 사용자 풀 생성을 참조하세요.
- 다음과 같은 구성의 OIDC IdP:
  - `client_secret_post` 클라이언트 인증을 지원합니다. Amazon Cognito는 IdP에 대한 OIDC 검색 엔드포인트에서 `token_endpoint_auth_methods_supported` 신청을 확인하지 않습니다. Amazon Cognito는 `client_secret_basic` 클라이언트 인증을 지원하지 않습니다. 클라이언트 인증에 대한 자세한 내용은 OpenID Connect 설명서에서 클라이언트 인증을 참조하세요.

- `openid_configuration`, `userInfo` 및 `jwks_uri`와 같은 OIDC 엔드포인트에만 HTTPS를 사용합니다.
- OIDC 엔드포인트에만 TCP 포트 80 및 443을 사용합니다.
- HMAC-SHA 또는 RSA 알고리즘을 사용하여 ID 토큰에만 서명합니다.
- 키 ID `kid` 클레임을 `jwks_uri`에 게시하며 토큰에 `kid` 클레임을 포함합니다.

1단계: OIDC IdP로 등록

Amazon Cognito로 OIDC IdP를 생성하려면 먼저 애플리케이션을 OIDC IdP에 등록하여 클라이언트 ID와 클라이언트 암호를 받아야 합니다.

OIDC IdP로 등록하려면

1. OIDC IdP에서 개발자 계정을 생성합니다.

   OIDC로 연결되는 링크 IdPs

OIDC IdP	설치 방법	OIDC 검색 URL
Salesforce	Salesforce 자격 증명 공급자 설치	https://login.salesforce.com
Ping Identity	Ping Identity 자격 증명 공급자 설치	https://*Your Ping domain address*:9031/idp/userinfo.openid 예: `https://pf.company.com:9031/idp/userinfo.openid`
Okta	Okta 자격 증명 공급자 설치	https://*Your Okta subdomain*.oktapreview.com 또는 `https://`*Your Okta subdomain*`.okta.com`

OIDC IdP	설치 방법	OIDC 검색 URL
Microsoft Azure Active Directory (Azure AD)	Microsoft Azure AD 자격 증명 공급자 설치	`https://login.microsoftonline.com/ {tenant}/v2.0`
Google	Google 자격 증명 공급자 설치	https://accounts.google.com **Note** Amazon Cognito는 통합 소셜 로그인 IdP로 Google을 제공합니다. 통합 IdP 사용을 권장합니다. 사용자 풀이 있는 소셜 ID 공급자 사용를 참조하세요.

2. `/oauth2/idpresponse` 엔드포인트가 있는 사용자 풀 도메인 URL을 OIDC IdP에 등록합니다. 그래야만 사용자를 인증할 때 나중에 Amazon Cognito에서 OIDC IdP를 수락합니다.

```
https://mydomain.us-east-1.amazoncognito.com/oauth2/idpresponse
```

3. 콜백 URL을 Amazon Cognito 사용자 풀에 등록합니다. 이는 인증 성공 이후에 Amazon Cognito에서 사용자를 리디렉션하는 페이지의 URL입니다.

```
https://www.example.com
```

4. 사용자의 범위를 선택합니다. 범위 openid는 필수 항목입니다. 이메일 범위가 있어야만 이메일 및 email_verified 클레임에 대한 액세스 권한을 받을 수 있습니다.
5. OIDC IdP에서 클라이언트 ID와 클라이언트 암호를 제공합니다. 사용자 풀에서 OIDC IdP를 설정할 때 이 둘을 사용하게 됩니다.

예: 사용자 풀에서 Salesforce를 OIDC IdP로 사용

Salesforce 같은 OIDC 호환 IdP와 사용자 풀 사이에 신뢰를 설정해야 할 때 OIDC IdP를 사용합니다.

1. Salesforce 개발자 웹사이트에서 계정을 생성합니다.

2. 이전 단계에서 설정한 개발자 계정을 통해 로그인합니다.
3. Salesforce 페이지에서 다음 중 하나를 수행합니다.
   - Lightning Experience를 사용 중인 경우에는 설정 기어 아이콘을 선택하고 Setup Home(설정 홈)을 선택합니다.
   - Salesforce Classic을 사용 중인 경우에는 사용자 인터페이스 헤더에 Setup(설정)이 표시됩니다. 이를 선택합니다.
   - Salesforce Classic을 사용 중인 경우에는 헤더에 Setup(설정)이 표시됩니다. 상단의 탐색 모음에서 이름을 선택하고 드롭다운 목록에서 Setup(설정)을 선택합니다.
4. 왼쪽 탐색 모음에서 Company Settings(회사 설정)를 선택합니다.
5. 탐색 모음에서 [도메인(Domain)]을 선택하고 도메인을 입력한 다음, [생성(Create)]을 선택합니다.
6. 왼쪽 탐색 모음에서 [플랫폼 도구(Platform Tools)]에서 [앱(Apps)]을 선택합니다.
7. App Manager(앱 관리자)를 선택합니다.
8. a. [새 연결된 앱(New connected app)]을 선택합니다.
   b. 필수 필드를 작성합니다.

      시작 URL(Start URL) 아래에서 Salesforce IdP로 로그인하는 사용자 풀 도메인에 대한 `/authorize` 엔드포인트에 URL을 입력합니다. 사용자가 연결된 앱에 액세스하면 Salesforce가 사용자를 이 URL로 안내하여 로그인을 완료합니다. 그런 다음 Salesforce는 사용자를 앱 클라이언트와 연결된 콜백 URL로 리디렉션합니다.

      ```
      https://mydomain.us-east-1.amazoncognito.com/authorize?
      response_type=code&client_id=<your_client_id>&redirect_uri=https://
      www.example.com&identity_provider=CorpSalesforce
      ```

   c. OAuth 설정(OAuth settings)을 사용하고 콜백 URL(Callback URL)에 사용자 풀 도메인에 대한 `/oauth2/idpresponse` 엔드포인트의 URL을 입력합니다. 이는 Salesforce에서 Amazon Cognito가 OAuth 토큰과 교환하는 권한 부여 코드를 발행하는 URL입니다.

      ```
      https://mydomain.us-east-1.amazoncognito.com/oauth2/idpresponse
      ```

9. 사용자의 범위를 선택합니다. openid 범위를 포함해야 합니다. email 및 email_verified claims에 대한 액세스 권한을 부여하려면 email 범위를 추가합니다. 범위를 공백으로 구분합니다.
10. 생성을 선택합니다.

Salesforce에서는 클라이언트 ID를 사용자 키(Consumer Key)로 부르고, 클라이언트 암호를 사용자 암호(Consumer Secret)로 부릅니다. 클라이언트 ID와 클라이언트 암호를 메모합니다. 다음 섹션에서 이 둘을 사용합니다.

2단계: 사용자 풀에 OIDC IdP 추가

이 섹션에서는 OIDC IdP에서 OIDC 기반 인증 요청을 처리하도록 사용자 풀을 구성합니다.

OIDC IdP를 추가하려면(Amazon Cognito 콘솔)

OIDC IdP 추가

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 자격 증명을 입력합니다. AWS
2. 탐색 메뉴에서 [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. [로그인 환경(Sign-in experience)] 탭을 선택합니다. 페더레이션 로그인(Federated sign-in)을 찾아서 자격 증명 공급자 추가(Add an identity provider)를 선택합니다.
5. OpenID Connect IdP를 선택합니다.
6. 공급자 이름(Provider name)에 고유한 이름을 입력합니다.
7. [클라이언트 ID(Client ID)]에 공급자로부터 받은 클라이언트 ID를 입력합니다.
8. 클라이언트 암호(Client secret)에 공급자로부터 받은 클라이언트 암호를 입력합니다.
9. 이 공급자의 [권한 있는 범위(Authorized scopes)]를 입력합니다. 범위는 애플리케이션이 공급자로부터 요청할 사용자 속성(예: `name`, `email`) 그룹을 정의합니다. OAuth 2.0 사양에 따라 범위를 공백으로 구분해야 합니다.

   사용자는 애플리케이션에 이러한 속성을 제공하는 것에 동의하라는 메시지를 받게 됩니다.
10. [속성 요청 메서드(Attribute request method)]를 선택하여 공급자가 운영하는 userInfo 엔드포인트에서 사용자 세부 정보를 가져오는 데 사용해야 하는 HTTP 메서드(GET 또는 POST)를 Amazon Cognito에 제공합니다.
11. [설정 메서드(Setup method)]를 선택하여 [발급자 URL을 통해 자동 채우기(Auto fill through issuer URL)] 또는 [수동 입력(Manual input)]으로 OpenID Connect 엔드포인트를 검색합니다. 공급자에게 Amazon Cognito가 `authorization`, `token`, `userInfo` 및 `jwks_uri` 엔드포인트의 URL을 검색할 수 있는 퍼블릭 `.well-known/openid-configuration` 엔드포인트가 있는 경우 발급자 URL을 통해 자동 채우기(Auto fill through issuer URL)를 사용합니다.

12. 발급자 URL이나 IdP의 `authorization`, `token`, `userInfo`, `jwks_uri` 엔드포인트 URL을 입력합니다.

> **Note**
>
> URL은 `https://`로 시작해야 하며, 슬래시 `/`로 끝나면 안 됩니다. 포트 번호 443 및 80만 이 URL에 사용할 수 있습니다. 예를 들어, Salesforce는 다음 URL을 사용합니다. `https://login.salesforce.com`
> 자동 채우기를 선택하는 경우 검색 문서에서 `authorization_endpoint`, `token_endpoint`, `userinfo_endpoint`, `jwks_uri` 값에 HTTPS를 사용해야 합니다. 그렇지 않으면 로그인이 실패합니다.

13. OIDC 클레임 sub는 기본값으로 사용자 풀 속성 [사용자 이름(Username)]에 매핑됩니다. 이 외의 OIDC 클레임도 사용자 풀 속성으로 매핑할 수 있습니다. OIDC 클레임을 입력하고, 드롭다운 목록에서 해당하는 사용자 풀 속성을 선택합니다. 예를 들어, 클레임 email은 주로 사용자 풀 속성 Email로 매핑됩니다.
14. IdP의 속성을 사용자 풀에 매핑합니다. 자세한 내용은 사용자 풀에 대한 자격 증명 공급자 속성 매핑 지정을 참조하세요.
15. 생성(Create)을 선택합니다.
16. 앱 클라이언트 통합(App client integration) 탭의 목록에서 앱 클라이언트(App clients) 중 하나를 선택한 다음 호스트된 UI 설정 편집(Edit hosted UI settings)을 선택합니다. 자격 증명 공급자(Identity providers)에서 앱 클라이언트에 새 OIDC IdP를 추가합니다.
17. 변경 사항 저장(Save changes)을 선택합니다.

OIDC IdP를 추가하려면(AWS CLI)

- CreateIdentityProviderAPI 메서드의 매개변수 설명을 참조하십시오.

```
aws cognito-idp create-identity-provider
--user-pool-id string
--provider-name string
--provider-type OIDC
--provider-details map


--attribute-mapping string
```

```
--idp-identifiers (list)
--cli-input-json string
--generate-cli-skeleton string
```

공급자 세부 정보의 이 맵을 사용합니다.

```
{
  "client_id": "string",
  "client_secret": "string",
  "authorize_scopes": "string",
  "attributes_request_method": "string",
  "oidc_issuer": "string",

  "authorize_url": "string",
  "token_url": "string",
  "attributes_url": "string",
  "jwks_uri": "string"
}
```

3단계: OIDC IdP 구성 테스트

이전 두 섹션의 요소를 사용하여 인증 URL을 생성하고 OIDC IdP 구성을 테스트할 수 있습니다.

```
https://mydomain.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&client_id=1example23456789&redirect_uri=https://www.example.com
```

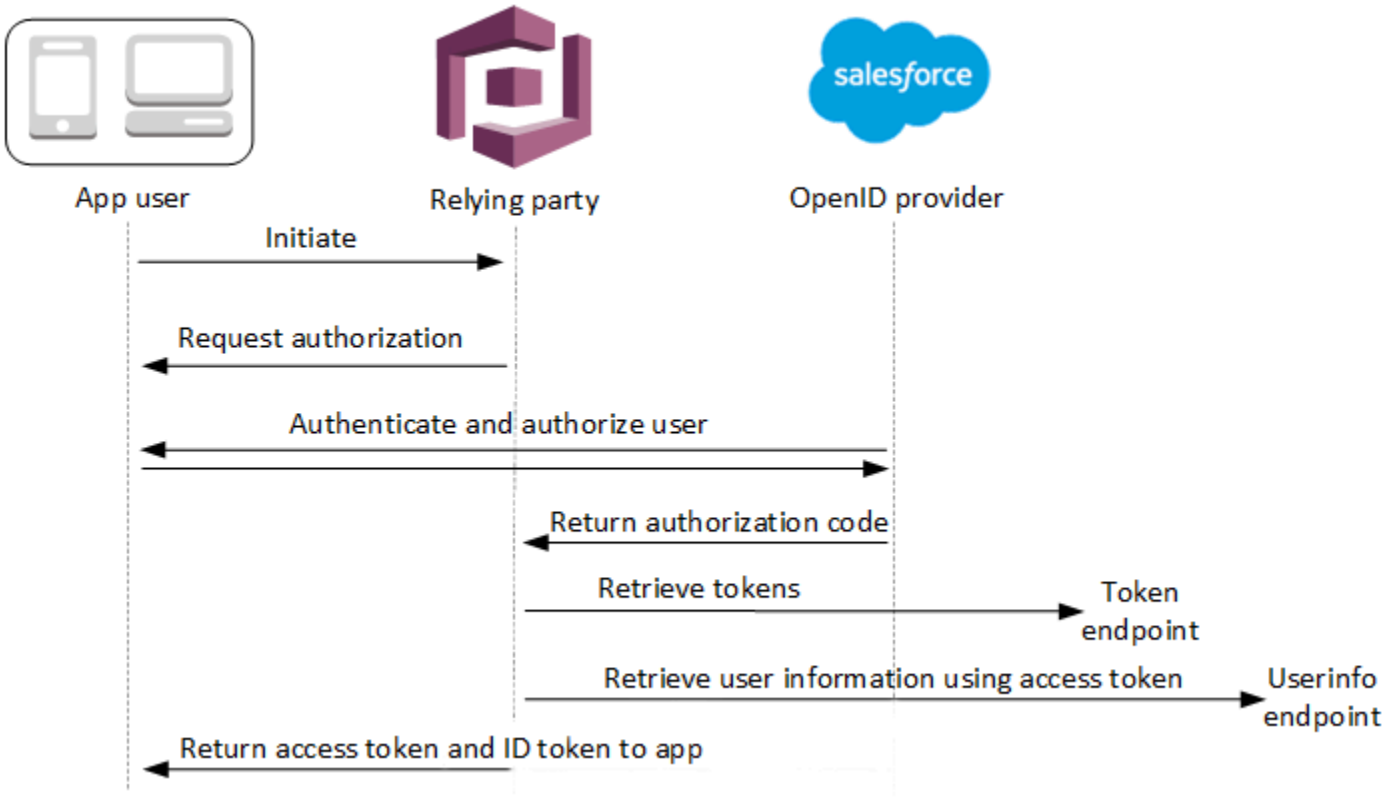
사용자 풀 도메인 이름(Domain name) 콘솔 페이지에서 사용자의 도메인을 찾을 수 있습니다. client_id는 일반 설정 페이지에 있습니다. redirect_uri 파라미터용 콜백 URL을 사용합니다. 이것은 인증 성공 이후에 사용자가 리디렉션되는 페이지의 URL입니다.

OIDC 사용자 풀 IdP 인증 흐름

사용자가 OIDC IdP를 사용하여 애플리케이션에 로그인하는 경우 다음 인증 흐름을 통과합니다.

1. Amazon Cognito 내장 로그인 페이지에 Salesforce 같은 OIDC IdP를 통해 로그인하는 옵션이 있습니다.

2. 사용자가 OIDC IdP의 `authorization` 엔드포인트로 리디렉션됩니다.
3. 사용자 인증 후 OIDC IdP에서 인증 코드를 사용하여 Amazon Cognito로 리디렉션합니다.
4. Amazon Cognito는 OIDC IdP의 인증 코드를 액세스 토큰으로 교환합니다.
5. Amazon Cognito는 사용자 풀에서 사용자 계정을 생성하거나 업데이트합니다.
6. Amazon Cognito는 자격 증명, 액세스 권한 및 새로 고침 토큰이 포함될 수 있는 애플리케이션 보유자 토큰을 발행합니다.



> **Note**
>
> Amazon Cognito는 5분 이내에 완료되지 않는 인증 요청을 취소하고 사용자를 호스팅 UI로 리디렉션합니다. 페이지에 `Something went wrong` 오류 메시지가 표시됩니다.

OIDC는 OAuth 2.0 위에 있는 ID 레이어로, OIDC 클라이언트 앱 (신뢰 당사자) 에서 발급하는 JSON 형식 (JWT) ID 토큰을 지정합니다. IdPs Amazon Cognito를 OIDC 신뢰 당사자로 추가하는 방법은 OIDC IdP 문서를 참조하세요.

사용자가 인증 코드 부여를 통해 인증할 때 사용자 풀이 ID, 액세스 및 새로 고침 토큰을 반환합니다. ID 토큰은 자격 증명 관리용 표준 OIDC 토큰이며, 액세스 토큰은 표준 OAuth 2.0 토큰입니다. 사용자 풀 앱 클라이언트가 지원할 수 있는 권한 부여 유형에 대한 자세한 내용은 권한 부여 엔드포인트 섹션을 참조하세요.

사용자 풀이 OIDC 제공업체의 클레임을 처리하는 방법

사용자가 서드 파티 OIDC 제공업체로 로그인을 완료하면 Amazon Cognito 호스팅 UI가 IdP에서 권한 부여 코드를 검색합니다. 사용자 풀은 액세스 및 ID 토큰의 권한 부여 코드를 IdP의 `token` 엔드포인트와 교환합니다. 사용자 풀은 이러한 토큰을 사용자 또는 앱에 전달하지 않지만, 이를 사용하여 자체 토큰의 클레임에 표시되는 데이터로 사용자 프로필을 구축합니다.

Amazon Cognito는 액세스 토큰을 독립적으로 검증하지 않습니다. 대신 제공업체 `userInfo` 엔드포인트에서 사용자 속성 정보를 요청하고 토큰이 유효하지 않을 경우 요청이 거부될 것으로 예상합니다.

Amazon Cognito는 다음 확인을 통해 제공업체 ID 토큰을 검증합니다.

1. 제공업체가 RSA, HMAC, Elliptic Curve 세트의 알고리즘을 사용하여 토큰에 서명했는지 확인합니다.
2. 제공업체가 비대칭 서명 알고리즘으로 토큰에 서명한 경우 토큰 `kid` 클레임의 서명 키 ID가 제공업체 `jwks_uri` 엔드포인트에 나열되어 있는지 확인합니다.
3. ID 토큰 서명을 제공업체 메타데이터를 기반으로 예상하는 서명과 비교합니다.
4. `iss` 클레임을 IdP용으로 구성된 OIDC 발급자와 비교합니다.
5. `aud` 클레임이 IdP에 구성된 클라이언트 ID와 일치하는지 또는 `aud` 클레임에 여러 값이 있는 경우 구성된 클라이언트 ID가 포함되어 있는지 비교합니다.
6. `exp` 클레임의 타임스탬프가 현재 시간 이전이 아닌지 확인합니다.

사용자 풀은 ID 토큰을 검증한 다음 제공업체 액세스 토큰으로 제공업체 `userInfo` 엔드포인트에 요청을 시도합니다. 액세스 토큰의 범위에 따라 읽기 권한이 부여된 모든 사용자 프로필 정보를 검색합니다. 그런 다음 사용자 풀은 사용자 풀에서 필요에 따라 설정한 사용자 속성을 검색합니다. 필수 속성에 대해서는 제공업체 구성에서 속성 매핑을 생성해야 합니다. 사용자 풀은 제공업체 ID 토큰과 `userInfo` 응답을 확인합니다. 사용자 풀은 매핑 규칙과 일치하는 모든 클레임을 사용자 풀 사용자 프로필의 사용자 속성에 기록합니다. 사용자 풀은 매핑 규칙과 일치하지만 필수는 아니며 제공업체의 클레임에도 없는 속성은 무시합니다.

사용자 풀에 대한 자격 증명 공급자 속성 매핑 지정

AWS CLI 또는 API를 사용하여 AWS Management Console사용자 풀의 ID 공급자 (IdP) 에 대한 속성 매핑을 지정할 수 있습니다.

매핑에 대해 알아야 할 사항

사용자 속성 매핑을 설정하기 전에 다음과 같은 중요한 세부 정보를 검토하십시오.

- 페더레이션 사용자가 애플리케이션에 로그인할 때 사용자 풀에 필요한 각 사용자 풀 속성의 매핑이 있어야 합니다. 예를 들어, 사용자 풀에 로그인용 `email` 속성이 필요한 경우 이 속성을 IdP의 해당 속성에 매핑합니다.
- 기본적으로 매핑된 이메일 주소는 확인되지 않습니다. 일회용 코드를 사용하여 매핑된 이메일 주소는 확인할 수 없습니다. 대신 IdP의 속성을 매핑하여 확인 상태를 가져옵니다. 예를 들어 Google과 대부분의 OIDC 공급자는 `email_verified` 속성을 포함합니다.
- ID 제공업체(IdP) 토큰을 사용자 풀의 사용자 지정 속성에 매핑할 수 있습니다. 소셜 제공업체는 액세스 토큰을 제시하고, OIDC 제공업체는 액세스 및 ID 토큰을 제공합니다. 토큰을 매핑하려면 최대 2,048자의 사용자 지정 속성을 추가하고, 앱 클라이언트에 속성에 대한 쓰기 권한을 부여하고, IdP에서 사용자 지정 속성으로 `access_token` 또는 `id_token`을 매핑합니다.
- 매핑된 각 사용자 풀 속성의 경우 최대 값 길이(2,048자)는 Amazon Cognito가 IdP에서 가져오는 값에 대해 충분히 커야 합니다. 그러지 않으면 사용자가 애플리케이션에 로그인할 때 Amazon Cognito에서 오류를 보고합니다. Amazon Cognito는 토큰 길이가 2,048자를 초과하는 경우 IdP 토큰을 사용자 지정 속성에 매핑하는 것을 지원하지 않습니다.
- Amazon Cognito는 다음 표에 나와 있는 것처럼 연동 IdP가 통과한 특정 클레임으로부터 연동 사용자 프로필의 `username` 속성을 도출합니다. 예를 들어 Amazon Cognito는 이 속성 값 앞에 IdP 이름을 추가합니다. `MyOIDCIdP_[sub]` 연동 사용자가 외부 사용자 디렉토리의 속성과 정확히 일치하는 속성을 갖도록 하려면 해당 속성을 와 같은 Amazon Cognito 로그인 속성에 매핑하십시오. `preferred_username`

ID 제공업체	**`username`** 소스 속성
Facebook	`id`
Google	`sub`
Login with Amazon	`user_id`

ID 제공업체	**`username`** 소스 속성
Apple로 로그인	`sub`
SAML 공급자	`NameID`
OpenID Connect(OIDC) 공급자	`sub`

- 사용자가 애플리케이션에 로그인할 때 Amazon Cognito는 매핑된 사용자 풀 속성을 업데이트할 수 있어야 합니다. 사용자가 IdP를 통해 로그인하면 Amazon Cognito가 매핑된 속성을 IdP의 최신 정보로 업데이트합니다. Amazon Cognito는 현재 값이 이미 최신 정보와 일치하더라도 매핑된 각 속성을 업데이트합니다. Amazon Cognito가 속성을 업데이트할 수 있는지 확인하려면 다음 요구 사항을 확인하세요.
  - IdP에서 매핑하는 모든 사용자 풀 사용자 지정 속성은 변경할 수 있어야 합니다. 변경 가능한 사용자 지정 속성은 언제든지 업데이트할 수 있습니다. 반대로 사용자 프로필을 처음 만들 때 사용자의 변경할 수 없는 사용자 지정 속성 값만 설정할 수 있습니다. Amazon Cognito 콘솔에서 변경 가능한 사용자 지정 속성을 생성하려면 가입 환경(Sign-up experience) 탭에서 사용자 지정 속성 추가(Add custom attributes)를 선택할 때 추가한 속성의 변경 가능(Mutable) 확인란을 활성화합니다. 또는 CreateUserPoolAPI 작업을 사용하여 사용자 풀을 생성하는 경우 각 속성의 `Mutable` 파라미터를 로 설정할 수 있습니다. `true` IdP가 매핑된 변경 불가 속성에 대한 값을 전송하는 경우 Amazon Cognito는 오류를 반환하고 로그인에 실패합니다.
  - 애플리케이션의 앱 클라이언트 설정에서 매핑된 속성은 쓰기 가능해야 합니다. Amazon Cognito 콘솔의 [앱 클라이언트(App clients)] 페이지에 쓸 수 있는 속성을 설정할 수 있습니다. 또는 `CreateUserPoolClient` API 작업을 사용하여 앱 클라이언트를 생성하는 경우 이러한 속성을 `WriteAttributes` 어레이에 추가할 수 있습니다. IdP가 매핑된 쓰기 불가능한 속성에 대한 값을 전송하는 경우 Amazon Cognito는 속성 값을 설정하지 않고 인증을 진행합니다.
- IdP 속성에 여러 값이 포함된 경우 Amazon Cognito는 모든 값을 쉼표로 구분된 단일 문자열로 병합하고 영숫자가 아닌 문자 (", ',', '및' " 문자 제외) 를 포함하는 값을 URL 형식으로 인코딩합니다. `.` `-` `*` `_` 이러한 개별 값은 앱에 사용하기 전에 디코딩 및 구문 분석해야 합니다.

사용자 풀에 대한 자격 증명 공급자 속성 매핑 지정(AWS Management Console)

를 사용하여 사용자 풀의 AWS Management Console IdP에 대한 속성 매핑을 지정할 수 있습니다.

> **Note**
>
> Amazon Cognito는 수신 토큰에 클레임이 있는 경우에만 수신 클레임을 사용자 풀 속성에 매핑합니다. 이전에 매핑된 클레임이 수신 토큰에 더 이상 존재하지 않으면 삭제되거나 변경되지 않습니다. 애플리케이션에 삭제된 클레임의 매핑이 필요한 경우 사전 인증 Lambda 트리거를 사용하여 인증 중에 사용자 지정 속성을 삭제하고 이러한 속성이 수신 토큰에서 다시 채워지도록 허용할 수 있습니다.

소셜 IdP 속성 매핑을 지정하려면

1. Amazon Cognito 콘솔에 로그인합니다. 메시지가 표시되면 자격 증명을 입력합니다. AWS
2. 탐색 창에서 [사용자 풀(User Pools)]을 선택한 다음 편집할 사용자 풀을 선택합니다.
3. 로그인 환경(Sign-in experience) 탭을 선택하고 페더레이션 로그인(Federated sign-in)을 찾습니다.
4. 자격 증명 공급자 추가(Add an identity provider)를 선택하거나 구성한 Facebook, Google, Amazon 또는 Apple IdP를 선택합니다. 속성 매핑(Attribute mapping)을 찾아서 편집(Edit)을 선택합니다.

   소셜 IdP 추가에 대한 자세한 내용은 사용자 풀이 있는 소셜 ID 공급자 사용 섹션을 참조하세요.
5. 매핑해야 하는 각 속성에 대해 다음 단계를 완료합니다.

   a. [사용자 풀 속성(User pool attribute)] 열에서 속성을 선택합니다. 사용자 풀의 사용자 프로파일에 할당되는 속성입니다. 사용자 정의 속성은 표준 속성 뒤에 나열됩니다.

   b. 속성(attribute) 열에서 속성을 선택합니다. 공급자 디렉터리에서 전달되는 속성입니다. 소셜 공급자의 알려진 속성이 드롭다운 목록에 제공됩니다.

   c. IdP와 Amazon Cognito 간에 추가 속성을 매핑하려면 [다른 속성 추가(Add another attribute)]를 선택합니다.
6. 변경 사항 저장(Save changes)을 선택합니다.

SAML 공급자 속성 매핑을 지정하려면

1. Amazon Cognito 콘솔에 로그인합니다. 메시지가 표시되면 AWS 자격 증명을 입력합니다.
2. 탐색 창에서 [사용자 풀(User Pools)]을 선택한 다음 편집할 사용자 풀을 선택합니다.
3. 로그인 환경(Sign-in experience) 탭을 선택하고 페더레이션 로그인(Federated sign-in)을 찾습니다.

4. 자격 증명 공급자 추가(Add an identity provider)를 선택하거나 구성한 SAML IdP를 선택합니다. 속성 매핑(Attribute mapping)을 찾아서 편집(Edit)을 선택합니다. SAML IdP 추가에 대한 자세한 내용은 사용자 풀과 함께 SAML ID 공급자 사용 섹션을 참조하세요.
5. 매핑해야 하는 각 속성에 대해 다음 단계를 완료합니다.

   a. [사용자 풀 속성(User pool attribute)] 열에서 속성을 선택합니다. 사용자 풀의 사용자 프로파일에 할당되는 속성입니다. 사용자 정의 속성은 표준 속성 뒤에 나열됩니다.

   b. [SAML 속성(SAML attribute)] 열에서 속성을 선택합니다. 공급자 디렉터리에서 전달되는 속성입니다.

      IdP가 샘플 SAML 어설션을 참조용으로 제공할 수도 있습니다. 일부는 다음과 같이 간단한 이름을 IdPs 사용하는 반면`email`, 다른 일부는 다음과 비슷한 URL 형식의 속성 이름을 사용합니다.

      ```
      http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
      ```

   c. IdP와 Amazon Cognito 간에 추가 속성을 매핑하려면 다른 속성 추가(Add another attribute)를 선택합니다.
6. 변경 사항 저장(Save changes)을 선택합니다.

사용자 풀 (및 API) 에 대한 ID 제공자 속성 매핑 지정AWS CLIAWS

다음 명령을 사용하여 사용자 풀에 대한 IdP 속성 매핑을 지정합니다.

공급자 생성 시 속성 매핑을 지정하려면

- AWS CLI: `aws cognito-idp create-identity-provider`

  메타데이터 파일이 포함된 예제: `aws cognito-idp create-identity-provider --user-pool-id <user_pool_id> --provider-name=SAML_provider_1 --provider-type SAML --provider-details file:///details.json --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

  여기서 `details.json`에 다음 사항이 포함됩니다.

  ```
  {
      "MetadataFile": "<SAML metadata XML>"
  }
  ```

> **Note**
>
> *<SAML metadata XML>*에 따옴표(")가 있으면 이스케이프해야 합니다(\").

메타데이터 URL을 사용한 예:

```
aws cognito-idp create-identity-provider \
--user-pool-id us-east-1_EXAMPLE \
--provider-name=SAML_provider_1 \
--provider-type SAML \
--provider-details MetadataURL=https://myidp.example.com/saml/metadata \
--attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/
emailaddress
```

- AWS API: CreateIdentityProvider

기존 IdP에 대한 속성 매핑을 지정하려면

- AWS CLI: `aws cognito-idp update-identity-provider`

  예제: `aws cognito-idp update-identity-provider --user-pool-id <user_pool_id> --provider-name <provider_name> --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`
- AWS API: UpdateIdentityProvider

특정 IdP에 대한 속성 매핑 정보를 가져오려면

- AWS CLI: `aws cognito-idp describe-identity-provider`

  예제: `aws cognito-idp describe-identity-provider --user-pool-id <user_pool_id> --provider-name <provider_name>`
- AWS API: DescribeIdentityProvider

페더레이션 사용자를 기존 사용자 프로필에 연결

사용자 풀에 연결한 여러 ID 공급자 (IdPs) 가 포함된 프로필을 동일한 사용자가 가지고 있는 경우가 많습니다. Amazon Cognito는 사용자의 각 발생을 디렉터리에 있는 동일한 사용자 프로필에 연결할 수 있습니다. 이렇게 하면 여러 IdP 사용자를 보유한 한 사람이 앱에서 일관된 경험을 할 수 있습니다. AdminLinkProviderForUserAmazon Cognito에 연동 디렉터리에 있는 사용자의 고유 ID를 사용자 풀의 사용자로서 인식하도록 지시합니다. 사용자 프로필과 연결된 페더레이션 ID가 0개 이상인 경우 사용자 풀의 사용자는 청구 목적으로 1명의 월별 활성 사용자(MAU)로 계산됩니다.

페더레이션 사용자가 처음으로 사용자 풀에 로그인하면 Amazon Cognito는 해당 사용자의 자격 증명에 연결된 로컬 프로필을 찾습니다. 연결된 프로필이 없는 경우 사용자 풀이 새 프로필을 생성합니다. 로컬 프로필을 생성하여 연동 사용자가 처음 로그인하기 전에 언제든지 `AdminLinkProviderForUser` API 요청 (계획된 프리스테이징 작업 또는 실행) 을 통해 페더레이션 사용자에게 연결할 수 있습니다. 사전 가입 Lambda 트리거 사용자가 로그인하고 Amazon Cognito가 연결된 로컬 프로필을 감지하면 사용자 풀이 사용자의 클레임을 읽고 이를 IdP의 매핑 규칙과 비교합니다. 그런 다음 사용자 풀은 로그인에서 매핑된 클레임으로 연결된 로컬 프로필을 업데이트합니다. 이렇게 하면 액세스 클레임이 포함된 로컬 프로필을 구성하고 제공자에게 ID up-to-date 클레임을 보관할 수 있습니다. Amazon Cognito가 페더레이션 사용자를 연결된 프로필과 매칭하면 페더레이션 사용자는 항상 해당 프로필에 로그인하게 됩니다. 그런 다음 더 많은 사용자 제공업체 자격 증명을 동일한 프로필에 연결하여 각 고객에게 일관된 앱 경험을 제공할 수 있습니다. 이전에 로그인한 페더레이션 사용자를 연결하려면 먼저 기존 프로필을 삭제해야 합니다. *`[Provider name]`*`_identifier` 형식으로 기존 프로필을 식별할 수 있습니다. 예를 들어 `LoginWithAmazon_amzn1.account.AFAEXAMPLE`입니다. 생성한 후 타사 사용자 ID에 연결한 사용자는 생성 시 사용한 사용자 이름과 연결된 ID의 세부 정보가 포함된 `identities` 속성을 가집니다.

> **Important**
>
> 외부 페더레이션 ID를 가진 사용자가 사용자 풀의 기존 사용자로 로그인할 수 있으므로 `AdminLinkProviderForUser` 응용 프로그램 소유자가 신뢰한 외부 IdPs 및 공급자 속성에만 사용하는 것이 중요합니다.

예를 들어, 여러 고객과 공유하는 앱을 사용하는 관리형 서비스 제공업체(MSP)라고 가정해 봅니다. 각 고객은 Active Directory Federation Services(ADFS)를 통해 앱에 로그인합니다. IT 관리자인 Carlos는 각 고객 도메인에 계정을 가지고 있습니다. IdP에 관계없이 Carlos가 로그인할 때마다 앱 관리자로 인식되기를 원합니다.

ADFS는 Amazon IdPs msp_carlos@example.com Cognito에 대한 카를로스의 SAML email 어설션을 청구할 때 카를로스의 이메일 주소를 제시합니다. 사용자 풀에 Carlos 사용자 이름으로 사용자를 생성합니다. 다음 AWS Command Line Interface (AWS CLI) 명령은 ADFS1, ADFS2 및 ADFS3 출신의 카를로스 ID를 연결합니다. IdPs

> **Note**
>
> 특정 속성 클레임을 기반으로 사용자를 연결할 수 있습니다. 이 기능은 OIDC와 SAML에서만 사용할 수 있습니다. IdPs 다른 공급자 유형의 경우 고정된 소스 속성을 기반으로 연결해야 합니다. 자세한 내용은 을 참조하십시오. AdminLinkProviderForUser 소셜 IdP를 사용자 프로필에 연결할 때 ProviderAttributeName을 Cognito_Subject로 설정해야 합니다. ProviderAttributeValue는 IdP와 함께 사용자의 고유 식별자여야 합니다.

```
aws cognito-idp admin-link-provider-for-user \
--user-pool-id us-east-1_EXAMPLE \
--destination-user ProviderAttributeValue=Carlos,ProviderName=Cognito \
--source-user
 ProviderName=ADFS1,ProviderAttributeName=email,ProviderAttributeValue=msp_carlos@example.com

aws cognito-idp admin-link-provider-for-user \
--user-pool-id us-east-1_EXAMPLE \
--destination-user ProviderAttributeValue=Carlos,ProviderName=Cognito \
--source-user
 ProviderName=ADFS2,ProviderAttributeName=email,ProviderAttributeValue=msp_carlos@example.com

aws cognito-idp admin-link-provider-for-user \
--user-pool-id us-east-1_EXAMPLE \
--destination-user ProviderAttributeValue=Carlos,ProviderName=Cognito \
--source-user
 ProviderName=ADFS3,ProviderAttributeName=email,ProviderAttributeValue=msp_carlos@example.com
```

이제 사용자 풀의 Carlos 사용자 프로필에 다음 identities 속성이 있습니다.

```
[{
    "userId": "msp_carlos@example.com",
    "providerName": "ADFS1",
    "providerType": "SAML",
    "issuer": "http://auth.example.com",
    "primary": false,
    "dateCreated": 111111111111111
```

```
}, {
    "userId": "msp_carlos@example.com",
    "providerName": "ADFS2",
    "providerType": "SAML",
    "issuer": "http://auth2.example.com",
    "primary": false,
    "dateCreated": 111111111111111
}, {
    "userId": "msp_carlos@example.com",
    "providerName": "ADFS3",
    "providerType": "SAML",
    "issuer": "http://auth3.example.com",
    "primary": false,
    "dateCreated": 111111111111111
}]
```

페더레이션 사용자 연결에 대해 알아야 할 사항

- 최대 5명의 페더레이션 사용자를 각 사용자 프로필에 연결할 수 있습니다.
- 페더레이션 사용자를 기존 페더레이션 사용자 프로필이나 로컬 사용자에 연결할 수 있습니다.
- 에서는 공급자를 사용자 프로필에 연결할 수 없습니다 AWS Management Console.
- 사용자의 ID 토큰에는 `identities` 클레임에 연결된 모든 공급자가 포함되어 있습니다.
- API 요청에서 자동으로 생성된 연동 사용자 프로필의 비밀번호를 설정할 수 있습니다. AdminSetUserPassword 그러면 해당 사용자의 상태가 `EXTERNAL_PROVIDER`에서 `CONFIRMED`로 변경됩니다. 이 상태의 사용자는 페더레이션 사용자로 로그인할 수 있으며, 연결된 로컬 사용자처럼 API에서 인증 흐름을 시작할 수 있습니다. 또한 및 와 같은 토큰 인증 API 요청에서 암호와 속성을 수정할 수 있습니다. ChangePasswordUpdateUserAttributes 보안을 유지하고 사용자를 외부 IdP와 동기화된 상태를 유지하기 위해 페더레이션 사용자에서 암호를 설정하지 않는 것이 좋습니다. 대신 `AdminLinkProviderForUser`를 사용하여 사용자를 로컬 프로필에 연결하세요.
- Amazon Cognito는 사용자가 IdP를 통해 로그인할 때 연결된 로컬 사용자 프로필에 사용자 속성을 채웁니다. Amazon Cognito는 OIDC IdP의 ID 토큰에 있는 자격 증명 클레임을 처리하고, OAuth 2.0 및 OIDC 공급자의 `userInfo` 엔드포인트도 확인합니다. Amazon Cognito는 `userInfo`의 정보보다 ID 토큰에 있는 정보를 우선시합니다.

프로필에 연결한 외부 사용자 계정을 사용자가 더 이상 사용하지 않는다는 사실을 알게 되면 해당 사용자 계정과 사용자 풀 사용자의 연결을 끊을 수 있습니다. 사용자를 연결할 때 요청에 사용자의 속성 이름, 속성 값 및 제공업체 이름을 제공했습니다. 사용자에게 더 이상 필요하지 않은 프로필을 삭제하려면 동등한 파라미터로 AdminDisableProviderForUserAPI 요청을 하세요.

AdminLinkProviderForUser AWS SDK의 추가 명령 구문 및 예제는 를 참조하십시오.

Lambda 트리거를 사용하여 사용자 풀 워크플로 사용자 정의

Amazon Cognito는 AWS Lambda 함수를 사용하여 사용자 풀의 인증 동작을 수정합니다. 처음 가입하기 전, 인증을 완료한 후, 그리고 그 사이의 여러 단계에서 Lambda 함수를 자동으로 호출하도록 사용자 풀을 구성할 수 있습니다. 함수는 인증 흐름의 기본 동작을 수정하고, 사용자 풀 또는 기타 AWS 리소스를 수정하기 위한 API 요청을 수행하고, 외부 시스템과 통신할 수 있습니다. Lambda 함수의 코드는 사용자 고유의 것입니다. Amazon Cognito는 이벤트 데이터를 함수로 전송하고, 함수가 데이터를 처리할 때까지 기다리며, 대부분의 경우 세션에 적용하려는 변경 사항이 반영된 응답 이벤트를 예상합니다.

요청 및 응답 이벤트 시스템 내에서 자체 인증 챌린지를 도입하고, 사용자 풀과 다른 자격 증명 스토어 간에 사용자를 마이그레이션하고, 메시지를 사용자 지정하고, JSON 웹 토큰(JWT)을 수정할 수 있습니다.

Lambda 트리거는 사용자가 사용자 풀에서 작업을 시작한 후 Amazon Cognito가 사용자에게 전달하는 응답을 사용자 지정할 수 있습니다. 예를 들어, 다른 방법으로는 성공할 수 있는 사용자의 로그인을 막을 수 있습니다. 또한 AWS 환경, 외부 API, 데이터베이스 또는 자격 증명 스토어에 대해 런타임 작업을 수행할 수도 있습니다. 예를 들어 사용자 마이그레이션 트리거는 외부 작업과 Amazon Cognito의 변경을 결합할 수 있습니다. 즉, 외부 디렉터리에서 사용자 정보를 조회한 다음 해당 외부 정보를 기반으로 새 사용자의 속성을 설정할 수 있습니다.

사용자 풀에 Lambda 트리거를 할당하면 Amazon Cognito는 기본 흐름을 중단하여 함수에서 정보를 요청합니다. Amazon Cognito는 JSON 이벤트를 생성하여 함수에 전달합니다. 해당 이벤트에는 사용자 계정 생성, 로그인, 암호 재설정 또는 속성 업데이트를 위한 사용자의 요청 관련 정보가 포함됩니다. 그러면 함수에서 조치를 취하거나 수정되지 않은 상태로 이벤트를 다시 보낼 수 있습니다.

다음 표에는 Lambda 트리거를 사용하여 사용자 풀 작업을 사용자 지정할 수 있는 방법 중 일부가 요약되어 있습니다.

사용자 풀 흐름	작업	설명
사용자 지정 인증 흐름	인증 문제 정의	사용자 지정 인증 흐름에서 다음 문제를 결정합니다.
	인증 문제 생성	사용자 지정 인증 흐름에서 다음 문제를 생성합니다.

사용자 풀 흐름	작업	설명
	인증 문제 응답 확인	사용자 지정 인증 흐름에서 응답이 올바른지 결정합니다.
인증 이벤트	the section called “사전 인증 Lambda 트리거”	로그인 요청 승인 및 거절에 대한 사용자 지정 검증입니다.
	the section called “사후 인증 Lambda 트리거”	사용자 지정 분석을 위한 이벤트 기록
	the section called “사전 토큰 생성 Lambda 트리거”	토큰 신청 증강 또는 억제
가입	the section called “사전 가입 Lambda 트리거”	가입 요청을 수락 또는 거부하는 사용자 지정 검증 수행
	the section called “사후 확인 Lambda 트리거”	사용자 지정 분석을 위한 사용자 지정 시작 메시지 또는 이벤트 로깅 추가
	the section called “사용자 마이그레이션 Lambda 트리거”	기존 사용자 디렉터리에 있는 사용자를 사용자 풀로 마이그레이션
메시지	the section called “사용자 정의 메시지 Lambda 트리거”	고급 사용자 지정 및 메시지 현지화 수행
토큰 생성	the section called “사전 토큰 생성 Lambda 트리거”	ID 토큰 속성 추가 또는 제거
이메일 및 SMS 서드 파티 공급자	the section called “사용자 지정 발신자 Lambda 트리거”	서드 파티 공급자를 사용하여 SMS 및 이메일 메시지 보내기

주제

- 중요 고려 사항
- 사용자 풀 Lambda 트리거 추가
- 사용자 풀 Lambda 트리거 이벤트

- 사용자 풀 Lambda 트리거 공통 파라미터
- API 작업을 Lambda 트리거에 연결
- Lambda 트리거를 사용자 풀 기능 작업에 연결
- 사전 가입 Lambda 트리거
- 사후 확인 Lambda 트리거
- 사전 인증 Lambda 트리거
- 사후 인증 Lambda 트리거
- 사용자 정의 인증 챌린지 Lambda 트리거
- 사전 토큰 생성 Lambda 트리거
- 사용자 마이그레이션 Lambda 트리거
- 사용자 정의 메시지 Lambda 트리거
- 사용자 지정 발신자 Lambda 트리거

중요 고려 사항

Lambda 함수를 위한 사용자 풀을 준비할 때는 다음을 고려하세요.

- Amazon Cognito가 Lambda 트리거로 전송하는 이벤트는 새 기능에 따라 변경될 수 있습니다. JSON 계층 구조에서 응답 및 요청 요소의 위치가 변경되거나 요소 이름이 추가될 수 있습니다. Lambda 함수에서 이 가이드에 설명된 입력 요소 키-값 페어를 수신할 것으로 예상할 수 있지만 입력 검증이 엄격해지면 함수가 실패할 수 있습니다.
- Amazon Cognito가 일부 트리거에 전송하는 여러 버전의 이벤트 중 하나를 선택할 수 있습니다. 일부 버전에서는 Amazon Cognito 요금 변경을 수락해야 할 수 있습니다. 요금에 대한 자세한 내용은 Amazon Cognito 요금 섹션을 참조하세요. 사전 토큰 생성 Lambda 트리거에서 액세스 토큰을 사용자 지정하려면 고급 보안 기능으로 사용자 풀을 구성하고 이벤트 버전 2를 사용하도록 Lambda 트리거 구성을 업데이트해야 합니다.
- 사용자 지정 발신자 Lambda 트리거를 제외하고, Amazon Cognito는 Lambda 함수를 동기적으로 간접 호출합니다. Amazon Cognito에서 Lambda 함수를 호출하면 5초 내에 응답해야 합니다. 그렇지 않고 직접적인 호출을 재시도할 수 있는 경우 Amazon Cognito는 호출을 재시도합니다. 3번 연속해서 실패하면 함수 제한 시간을 초과하게 됩니다. 이 5초 제한 시간 값은 변경할 수 없습니다. 자세한 내용은 AWS Lambda 개발자 안내서의 Lambda 프로그래밍 모델을 참조하세요.

Amazon Cognito는 HTTP 상태 코드가 500-599인 호출 오류를 반환하는 함수 호출을 재시도하지 않습니다. 이러한 코드는 Lambda가 함수를 시작할 수 없게 만드는 구성 문제를 나타냅니다. 자세한 내용은 AWS Lambda에서 오류 처리 및 자동 재시도를 참조하세요.

- Lambda 트리거 구성에서는 함수 버전을 선언할 수 없습니다. Amazon Cognito 사용자 풀은 기본적으로 최신 버전의 함수를 호출합니다. 하지만 CreateUserPool 또는 UpdateUserPool API 요청에서 함수 버전을 별칭과 연결하고 트리거 `LambdaArn`을 별칭 ARN으로 설정할 수 있습니다. AWS Management Console에서는 이 옵션을 사용할 수 없습니다. 별칭에 대한 자세한 내용은 AWS Lambda 개발자 안내서의 Lambda 함수 별칭을 참조하세요.
- Lambda 트리거를 삭제하는 경우 사용자 풀의 해당 트리거를 업데이트해야 합니다. 예를 들어 사후 인증 트리거를 삭제하면 해당 사용자 풀의 사후 인증 트리거를 없음으로 설정해야 합니다.
- Lambda 함수가 Amazon Cognito에 요청 및 응답 파라미터를 반환하지 않거나 오류를 반환하는 경우 인증 이벤트는 성공하지 못합니다. 함수에서 오류를 반환하여 사용자의 가입, 인증, 토큰 생성 또는 Lambda 트리거를 호출하는 인증 흐름의 다른 단계를 방지할 수 있습니다.

  Amazon Cognito 호스팅 UI는 Lambda 트리거가 생성한 오류를 로그인 프롬프트 위에 오류 텍스트로 반환합니다. Amazon Cognito 사용자 풀 API는 *`[trigger]`* `failed with error` *`[error text from response]`* 형식으로 트리거 오류를 반환합니다. Lambda 함수에서 사용자에게 표시하려는 오류만 생성하는 것이 좋습니다. `print()`와 같은 출력 방법을 사용하여 민감한 정보나 디버깅 정보를 CloudWatch Logs에 로깅합니다. 예시는 사전 가입 예: 사용자 이름이 5자 미만인 경우 가입 거부에서 확인하십시오.
- 다른 AWS 계정의 Lambda 함수를 사용자 풀의 트리거로 추가할 수 있습니다. AWS CloudFormation과 AWS CLI에서 CreateUserPool 및 UpdateUserPool API 작업 또는 이에 준하는 작업을 사용하여 교차 계정 트리거를 추가해야 합니다. AWS Management Console에서는 교차 계정 함수를 추가할 수 없습니다.
- Amazon Cognito 콘솔에서 Lambda 트리거를 추가하면 Amazon Cognito는 사용자 풀이 함수를 호출하도록 허용하는 리소스 기반 정책을 함수에 추가합니다. Amazon Cognito 콘솔 외부에서 교차 계정 함수를 비롯한 Lambda 트리거를 생성하는 경우, Lambda 함수의 리소스 기반 정책에 권한을 추가해야 합니다. 추가한 권한은 Amazon Cognito가 사용자 풀을 대신하여 함수를 호출하도록 허용해야 합니다. Lambda 콘솔에서 권한을 추가하거나 Lambda AddPermission API 작업을 사용할 수 있습니다.

  Lambda 리소스 기반 정책 예제

  다음 Lambda 리소스 기반 정책 예제는 Amazon Cognito에 Lambda 함수를 호출할 수 있는 제한적인 권한을 부여합니다. Amazon Cognito는 `aws:SourceArn` 조건의 사용자 풀과 `aws:SourceAccount` 조건의 계정을 둘 다 대신하는 경우에만 함수를 호출할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Id": "default",
    "Statement": [
        {
            "Sid": "lambda-allow-cognito",
            "Effect": "Allow",
            "Principal": {
                "Service": "cognito-idp.amazonaws.com"
            },
            "Action": "lambda:InvokeFunction",
            "Resource": "<your Lambda function ARN>",
            "Condition": {
                "StringEquals": {
                    "AWS:SourceAccount": "<your account number>"
                },
                "ArnLike": {
                    "AWS:SourceArn": "<your user pool ARN>"
                }
            }
        }
    ]
}
```

사용자 풀 Lambda 트리거 추가

콘솔을 사용하여 사용자 풀 Lambda 트리거를 추가하려면

1. Lambda 콘솔을 사용하여 Lambda 함수를 생성합니다. Lambda 함수에 대한 자세한 내용은 AWS Lambda 개발자 가이드를 참조하세요.
2. Amazon Cognito 콘솔로 이동한 다음 사용자 풀(User Pools)을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. [사용자 풀 속성(User pool properties)] 탭을 선택하고 [Lambda 트리거(Lambda triggers)]를 찾습니다.
5. Lambda 트리거 추가(Add a Lambda trigger)를 선택합니다.
6. 사용자 지정할 인증 스테이지에 따라 Lambda 트리거 범주(Category)를 선택합니다.
7. Lambda 함수 할당(Assign Lambda function)을 선택한 다음 사용자 풀과 동일한 AWS 리전의 함수를 선택합니다.

> **Note**
>
> AWS Identity and Access Management(IAM) 자격 증명에 Lambda 함수를 업데이트할 수 있는 권한이 있는 경우 Amazon Cognito는 Lambda 리소스 기반 정책을 추가합니다. 이 정책을 사용하면 선택한 함수를 Amazon Cognito에서 호출할 수 있습니다. 로그인한 자격 증명에 충분한 IAM 권한이 없는 경우 리소스 기반 정책을 별도로 업데이트해야 합니다. 자세한 내용은 the section called “중요 고려 사항” 섹션을 참조하세요.

8. 변경 사항 저장(Save changes)을 선택합니다.
9. Lambda 콘솔에서 CloudWatch를 사용하여 Lambda 함수를 로그할 수 있습니다. 자세한 내용은 Lambda에 대한 CloudWatch Logs 액세스를 참조하세요.

사용자 풀 Lambda 트리거 이벤트

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. Lambda 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. 이 이벤트에는 다음과 같은 Lambda 트리거 공통 파라미터가 표시됩니다.

JSON

```
{
    "version": "string",
    "triggerSource": "string",
    "region": AWSRegion,
    "userPoolId": "string",
    "userName": "string",
    "callerContext":
        {
            "awsSdkVersion": "string",
            "clientId": "string"
        },
    "request":
        {
            "userAttributes": {
                "string": "string",
                ....
            }
        },
    "response": {}
```

```
}
```

사용자 풀 Lambda 트리거 공통 파라미터

version

Lambda 함수의 버전 번호입니다.

triggerSource

Lambda 함수를 트리거한 이벤트의 이름입니다. 각 triggerSource의 설명은 Lambda 트리거를 사용자 풀 기능 작업에 연결 섹션을 참조하세요.

리전

`AWSRegion` 인스턴스로서의 AWS 리전입니다.

userPoolId

사용자 풀의 ID입니다.

사용자 이름

현재 사용자의 사용자 이름입니다.

callerContext

요청 및 코드 환경에 대한 메타데이터입니다. 여기에는 awsSdkVersion 및 clientId 필드가 포함됩니다.

awsSdkVersion

요청을 생성한 AWS SDK 버전입니다.

clientId

사용자 풀 앱 클라이언트의 ID입니다.

request

사용자의 API 요청 세부 정보입니다. 여기에는 다음 필드와 해당 트리거에 해당하는 모든 요청 파라미터가 포함됩니다. 예를 들어 Amazon Cognito가 사전 인증 트리거에 전송하는 이벤트에는 `userNotFound` 파라미터도 포함됩니다. 사용자가 등록되지 않은 사용자 이름으로 로그인하려고 할 때 이 파라미터의 값을 처리하여 사용자 지정 작업을 수행할 수 있습니다.

userAttributes

사용자 속성 이름 및 값의 키 값 페어(1개 이상)입니다. 예: `"email": "john@example.com"`.

응답

이 파라미터는 원래 요청의 정보를 포함하지 않습니다. Lambda 함수는 전체 이벤트를 Amazon Cognito에 반환하고 모든 반환 파라미터를 `response`에 추가해야 합니다. 함수에 포함할 수 있는 반환 파라미터를 보려면 사용하려는 트리거에 대한 설명서를 참조하세요.

API 작업을 Lambda 트리거에 연결

다음 섹션에서는 Amazon Cognito가 사용자 풀의 활동에서 호출하는 Lambda 트리거를 설명합니다.

앱이 Amazon Cognito 사용자 풀 API, 호스팅 UI 또는 사용자 풀 엔드포인트를 통해 사용자를 로그인하면 Amazon Cognito는 세션 컨텍스트를 기반으로 Lambda 함수를 호출합니다. Amazon Cognito 사용자 풀 API와 사용자 풀 엔드포인트에 대한 자세한 내용은 Amazon Cognito 사용자 풀 API 및 사용자 풀 엔드포인트 사용을 참조하세요. 다음 섹션의 표에서는 Amazon Cognito가 함수를 호출하도록 하는 이벤트와 Amazon Cognito가 요청에 포함하는 `triggerSource` 문자열을 설명합니다.

주제

- Amazon Cognito API의 Lambda 트리거
- 호스팅 UI에서 Amazon Cognito 로컬 사용자에 대한 Lambda 트리거
- 페더레이션 사용자를 위한 Lambda 트리거

Amazon Cognito API의 Lambda 트리거

다음 표에서는 앱이 사용자 풀 로컬 사용자를 생성, 로그인 또는 업데이트할 때 Amazon Cognito가 호출할 수 있는 Lambda 트리거의 소스 문자열을 설명합니다.

Amazon Cognito API의 로컬 사용자 트리거 소스

API 연산	Lambda 트리거	트리거 소스
AdminCreateUser	사전 가입	`PreSignUp_AdminCreateUser`

API 연산	Lambda 트리거	트리거 소스
	사전 토큰 생성	`TokenGeneration_NewPasswordChallenge`
	사용자 지정 메시지	`CustomMessage_AdminCreateUser`
	사용자 지정 이메일 발신자	`CustomEmailSender_AdminCreateUser`
	사용자 지정 SMS 발신자	`CustomSMSSender_AdminCreateUser`
SignUp	사전 가입	`PreSignUp_SignUp`
	사용자 지정 메시지	`CustomMessage_SignUp`
	사용자 지정 이메일 발신자	`CustomEmailSender_SignUp`
	사용자 지정 SMS 발신자	`CustomSMSSender_SignUp`
ConfirmSignUp AdminConfirmSignUp	게시 확인	`PostConfirmation_ConfirmSignUp`
InitiateAuth AdminInitiateAuth	사전 인증	`PreAuthentication_Authentication`
	인증 문제 정의	`DefineAuthChallenge_Authentication`
	인증 문제 생성	`CreateAuthChallenge_Authentication`

API 연산	Lambda 트리거	트리거 소스

	사전 토큰 생성	`TokenGeneration_Authentication`
		`TokenGeneration_AuthenticateDevice`
		`TokenGeneration_RefreshTokens`

	사용자 마이그레이션	`UserMigration_Authentication`
	사용자 지정 메시지	`CustomMessage_Authentication`
	사용자 지정 이메일 발신자	`CustomEmailSender_AccountTakeOverNotification`
	사용자 지정 SMS 발신자	`CustomSMSSender_Authentication`
ForgotPassword	사용자 마이그레이션	`UserMigration_ForgotPassword`
	사용자 지정 메시지	`CustomMessage_ForgotPassword`
	사용자 지정 이메일 발신자	`CustomEmailSender_ForgotPassword`
	사용자 지정 SMS 발신자	`CustomSMSSender_ForgotPassword`
ConfirmForgotPassword	게시 확인	`PostConfirmation_ConfirmForgotPassword`

API 연산	Lambda 트리거	트리거 소스
UpdateUserAttributes AdminUpdateUserAttributes	사용자 지정 메시지	`CustomMessage_UpdateUserAttribute`
	사용자 지정 이메일 발신자	`CustomEmailSender_UpdateUserAttribute`
	사용자 지정 SMS 발신자	`CustomSMSSender_UpdateUserAttribute`
VerifyUserAttributes	사용자 지정 메시지	`CustomMessage_VerifyUserAttribute`
	사용자 지정 이메일 발신자	`CustomEmailSender_VerifyUserAttribute`
	사용자 지정 SMS 발신자	`CustomSMSSender_VerifyUserAttribute`

호스팅 UI에서 Amazon Cognito 로컬 사용자에 대한 Lambda 트리거

다음 표에서는 로컬 사용자가 호스팅 UI를 사용하여 사용자 풀에 로그인할 때 Amazon Cognito가 호출할 수 있는 Lambda 트리거의 소스 문자열을 설명합니다.

호스팅 UI의 로컬 사용자 트리거 소스

호스팅 UI URI	Lambda 트리거	트리거 소스
`/signup`	사전 가입	`PreSignUp_SignUp`
	사용자 지정 메시지	`CustomMessage_SignUp`
	사용자 지정 이메일 발신자	`CustomEmailSender_SignUp`
	사용자 지정 SMS 발신자	`CustomSMSSender_SignUp`

호스팅 UI URI	Lambda 트리거	트리거 소스
/confirmuser	게시 확인	PostConfirmation_ConfirmSignUp
/login	사전 인증	PreAuthentication_Authentication
	인증 문제 정의	DefineAuthChallenge_Authentication
	인증 문제 생성	CreateAuthChallenge_Authentication
	사전 토큰 생성	TokenGeneration_Authentication TokenGeneration_AuthenticateDevice TokenGeneration_RefreshTokens
	사용자 마이그레이션	UserMigration_Authentication
	사용자 지정 메시지	CustomMessage_Authentication
	사용자 지정 이메일 발신자	CustomEmailSender_AccountTakeOverNotification
	사용자 지정 SMS 발신자	CustomSMSSender_Authentication
/forgotpassword	사용자 마이그레이션	UserMigration_ForgotPassword

호스팅 UI URI	Lambda 트리거	트리거 소스
	사용자 지정 메시지	`CustomMessage_ForgotPassword`
	사용자 지정 이메일 발신자	`CustomEmailSender_ForgotPassword`
	사용자 지정 SMS 발신자	`CustomSMSSender_ForgotPassword`
`/confirmforgotpassword`	게시 확인	`PostConfirmation_ConfirmForgotPassword`

페더레이션 사용자를 위한 Lambda 트리거

다음 Lambda 트리거를 사용하여 페더레이션 공급자로 로그인하는 사용자에 대한 사용자 풀 워크플로를 사용자 지정할 수 있습니다.

> **Note**
>
> 페더레이션 사용자는 Amazon Cognito 호스팅 UI를 사용하여 로그인할 수 있습니다. 또는 자격 증명 공급자 로그인 페이지로 해당 사용자를 자동 리디렉션하는 권한 부여 엔드포인트에 대한 요청을 귀하가 생성하는 방법도 있습니다. Amazon Cognito 사용자 풀 API를 사용하여 페더레이션 사용자를 로그인할 수 없습니다.

페더레이션 사용자 트리거 소스

로그인 이벤트	Lambda 트리거	트리거 소스
첫 로그인	사전 가입	`PreSignUp_ExternalProvider`
	게시 확인	`PostConfirmation_ConfirmSignUp`

로그인 이벤트	Lambda 트리거	트리거 소스
	사전 토큰 생성	`TokenGeneration_HostedAuth`
후속 로그인	사전 인증	`PreAuthentication_Authentication`
	사후 인증	`PostAuthentication_Authentication`
	사전 토큰 생성	`TokenGeneration_HostedAuth`

페더레이션 로그인은 사용자 풀에서 사용자 정의 인증 챌린지 Lambda 트리거, 사용자 마이그레이션 Lambda 트리거, 사용자 정의 메시지 Lambda 트리거 또는 사용자 지정 발신자 Lambda 트리거를 호출하지 않습니다.

Lambda 트리거를 사용자 풀 기능 작업에 연결

각 Lambda 트리거는 사용자 풀에서 기능적 역할을 수행합니다. 예를 들어 트리거는 가입 흐름을 수정하거나 사용자 지정 인증 문제를 추가할 수 있습니다. Amazon Cognito가 Lambda 함수로 보내는 이벤트는 해당 기능적 역할을 구성하는 여러 작업 중 하나를 반영할 수 있습니다. 예를 들어, Amazon Cognito는 사용자가 가입할 때와 귀하가 사용자를 생성할 때 가입 전 트리거를 호출합니다. 동일한 기능적 역할에 대해 서로 다른 이러한 사례는 각각 고유한 `triggerSource` 값을 지닙니다. Lambda 함수는 해당 함수를 호출한 작업에 따라 수신 이벤트를 다르게 처리할 수 있습니다.

또한 Amazon Cognito는 이벤트가 트리거 소스에 해당하는 경우 할당된 모든 함수를 호출합니다. 예를 들어, 마이그레이션 사용자 및 인증 전 트리거를 할당한 사용자 풀에 사용자가 로그인하면 두 가지가 모두 활성화됩니다.

가입, 확인 및 로그인(인증) 트리거

트리거	triggerSource 값	이벤트
사전 가입	`PreSignUp_SignUp`	사전 가입.
사전 가입	`PreSignUp_AdminCreateUser`	관리자가 새 사용자를 만들면 사전 가입.

트리거	triggerSource 값	이벤트
사전 가입	`PreSignUp_ExternalProvider`	외부 자격 증명 공급자에 대한 사전 가입.
게시 확인	`PostConfirmation_ConfirmSignUp`	사후 가입 확인.
게시 확인	`PostConfirmation_ConfirmForgotPassword`	사후 암호 분실 확인.
사전 인증	`PreAuthentication_Authentication`	사전 인증.
사후 인증	`PostAuthentication_Authentication`	사후 인증.

사용자 지정 인증 문제 트리거

트리거	triggerSource 값	이벤트
인증 문제 정의	`DefineAuthChallenge_Authentication`	인증 문제 정의.
인증 문제 생성	`CreateAuthChallenge_Authentication`	인증 문제 생성.
인증 문제 확인	`VerifyAuthChallengeResponse_Authentication`	인증 문제 응답 확인.

사전 토큰 생성 트리거

트리거	triggerSource 값	이벤트
사전 토큰 생성	`TokenGeneration_HostedAuth`	Amazon Cognito는 호스팅 UI 로그인 페이지에서 사용자를 인증합니다.
사전 토큰 생성	`TokenGeneration_Authentication`	사용자 인증 흐름이 완료되었습니다.
사전 토큰 생성	`TokenGeneration_NewPasswordChallenge`	관리자가 사용자를 생성합니다. Amazon Cognito는 사용자가 임시 암호를 변경해야 할 때 이를 호출합니다.
사전 토큰 생성	`TokenGeneration_AuthenticateDevice`	사용자 디바이스 인증의 끝입니다.
사전 토큰 생성	`TokenGeneration_RefreshTokens`	사용자가 자격 증명 및 액세스 토큰을 새로 고치려고 합니다.

사용자 트리거 마이그레이션

트리거	triggerSource 값	이벤트
사용자 마이그레이션	`UserMigration_Authentication`	로그인 시 사용자 마이그레이션입니다.
사용자 마이그레이션	`UserMigration_ForgotPassword`	암호 찾기 흐름 도중 사용자 마이그레이션.

사용자 지정 메시지 트리거

트리거	triggerSource 값	이벤트
사용자 지정 메시지	`CustomMessage_SignUp`	사용자가 사용자 풀에 가입할 때 사용자 지정 메시지입니다.

트리거	triggerSource 값	이벤트
사용자 지정 메시지	`CustomMessage_AdminCreateUser`	관리자로서의 사용자가 생성되고 Amazon Cognito에서 이 사용자에게 임시 암호를 보낼 때 사용자 지정 메시지입니다.
사용자 지정 메시지	`CustomMessage_ResendCode`	기존 사용자가 새 확인 코드를 요청할 때 사용자 지정 메시지입니다.
사용자 지정 메시지	`CustomMessage_ForgotPassword`	사용자가 암호 재설정을 요청할 때 사용자 지정 메시지입니다.
사용자 지정 메시지	`CustomMessage_UpdateUserAttribute`	사용자가 이메일 주소 또는 전화 번호를 변경하고 Amazon Cognito에서 확인 코드를 보낼 때 사용자 지정 메시지입니다.
사용자 지정 메시지	`CustomMessage_VerifyUserAttribute`	사용자가 이메일 주소 또는 전화 번호를 추가하고 Amazon Cognito에서 확인 코드를 보낼 때 사용자 지정 메시지입니다.
사용자 지정 메시지	`CustomMessage_Authentication`	SMS MFA를 구성한 사용자가 로그인할 때 사용자 지정 메시지입니다.

사전 가입 Lambda 트리거

Amazon Cognito는 새 사용자를 가입시키기 직전에 사전 가입 AWS Lambda 함수를 활성화합니다. 가입 프로세스의 일부로 이 함수를 사용하여 사용자 지정 검증을 수행하고 검증 결과에 따라 등록 요청을 수락 또는 거부할 수 있습니다.

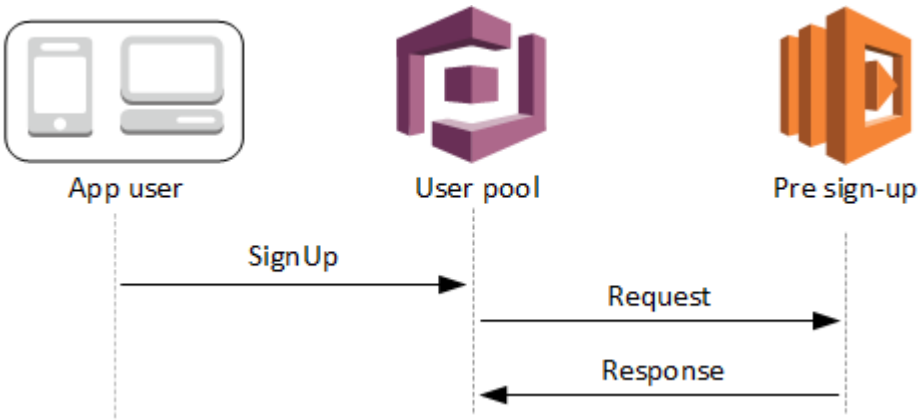
주제

- 사전 가입 Lambda 흐름

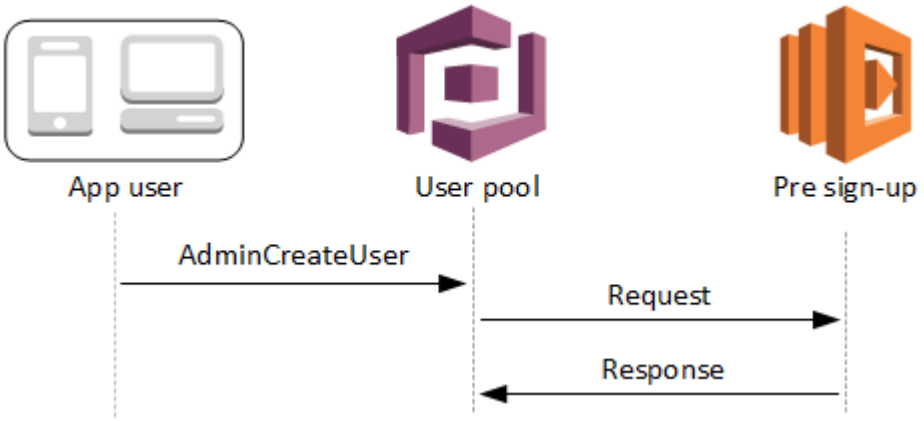
- 사전 가입 Lambda 트리거 파라미터
- 가입 자습서
- 사전 가입 예제: 등록된 도메인의 사용자 자동 확인
- 사전 가입 예제: 모든 사용자 자동 확인 및 자동 검증
- 사전 가입 예: 사용자 이름이 5자 미만인 경우 가입 거부

사전 가입 Lambda 흐름

클라이언트 가입 흐름



서버 가입 흐름



요청에 클라이언트의 검증 데이터가 포함됩니다. 이 데이터는 사용자 풀 SignUp 및 AdminCreateUser API 메서드에 전달된 `ValidationData` 값에서 가져옵니다.

사전 가입 Lambda 트리거 파라미터

Amazon Cognito가 이 Lambda 함수에 전달하는 요청은 아래 파라미터와 Amazon Cognito가 모든 요청에 추가하는 공통 파라미터의 조합입니다.

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
        },
        "validationData": {
            "string": "string",
            . . .
         },
        "clientMetadata": {
            "string": "string",
            . . .
         }
    },

    "response": {
        "autoConfirmUser": "boolean",
        "autoVerifyPhone": "boolean",
        "autoVerifyEmail": "boolean"
    }
}
```

사전 가입 요청 파라미터

userAttributes

사용자 속성을 나타내는 하나 이상의 이름-값 페어입니다. 속성 이름은 키입니다.

validationData

새 사용자 생성 요청 시 앱이 Amazon Cognito에 전달한 사용자 속성 데이터가 포함된 하나 이상의 키-값 페어입니다. 이 정보를 사용자 또는 API 요청의 파라미터로 Lambda 함수에 AdminCreateUser전송하십시오. ValidationData SignUp

Amazon Cognito는 ValidationData 데이터를 사용자가 생성한 사용자의 속성으로 설정하지 않습니다. ValidationData 사전 가입 Lambda 트리거를 위해 제공하는 임시 사용자 정보입니다.

clientMetadata

사전 가입 트리거에 지정하는 Lambda 함수에 사용자 지정 입력으로 제공할 수 있는 하나 이상의 키-값 페어입니다. , AdminRespondToAuthChallenge, ForgotPassword 및 API 작업에서 파라미터를 사용하여 ClientMetadata 이 데이터를 Lambda 함수에 전달할 수 있습니다 AdminCreateUser. SignUp

사전 가입 응답 파라미터

사용자를 자동으로 확인하려면 응답에서 `autoConfirmUser`를 `true`로 설정하면 됩니다. `autoVerifyEmail`을 `true`로 설정하여 사용자 이메일을 자동으로 확인할 수 있습니다. `autoVerifyPhone`을 `true`로 설정하여 사용자 전화 번호를 자동으로 확인할 수 있습니다.

> **Note**
>
> 응답 파라미터 `autoVerifyPhone`, `autoVerifyEmail` 및 `autoConfirmUser`는 `AdminCreateUser` API에 의해 사전 가입 Lambda 함수가 트리거될 때 Amazon Cognito에서 무시됩니다.

autoConfirmUser

사용자를 자동으로 확인하려면 `true`로 설정하고 그렇지 않으면 `false`로 설정합니다.

autoVerifyEmail

가입 중인 사용자의 이메일 주소를 확인한 것으로 설정하려면 `true`로 설정하고 그렇지 않으면 `false`로 설정합니다. `autoVerifyEmail`이 `true`로 설정되면 `email` 속성에 null이 아닌 유효한 값이 있어야 합니다. 그렇지 않으면 오류가 발생하고 사용자가 가입을 완료할 수 없습니다.

`email` 속성이 별칭으로 선택된 경우 `autoVerifyEmail`이 설정되면 사용자 이메일 주소의 별칭이 생성됩니다. 해당 이메일 주소의 별칭이 이미 있으면 별칭이 새 사용자로 이동하고 이전 사용자의 이메일 주소는 확인되지 않은 것으로 표시됩니다. 자세한 설명은 로그인 속성 사용자 지정 섹션을 참조하세요.

autoVerifyPhone

가입 중인 사용자의 전화 번호를 확인한 것으로 설정하려면 `true`로 설정하고 그렇지 않으면 `false`로 설정합니다. `autoVerifyPhone`이 `true`로 설정되면 `phone_number` 속성에 null이 아닌 유효한 값이 있어야 합니다. 그렇지 않으면 오류가 발생하고 사용자가 가입을 완료할 수 없습니다.

phone_number 속성이 별칭으로 선택될 경우 autoVerifyPhone이 설정되면 사용자 전화 번호의 별칭이 생성됩니다. 해당 전화 번호의 별칭이 이미 있으면 별칭이 새 사용자로 이동하고 이전 사용자의 전화번호는 확인되지 않은 것으로 표시됩니다. 자세한 내용은 로그인 속성 사용자 지정 섹션을 참조하세요.

가입 자습서

사전 가입 Lambda 함수는 사용자 가입 전에 트리거됩니다. 안드로이드 및 iOS용 Amazon Cognito 가입 튜토리얼을 JavaScript 참조하십시오.

플랫폼	튜토리얼
JavaScript 아이덴티티 SDK	다음을 사용하여 사용자를 등록합니다. JavaScript
Android 자격 증명 SDK	Android 사용자 가입
iOS 자격 증명 SDK	iOS 사용자 가입

사전 가입 예제: 등록된 도메인의 사용자 자동 확인

사전 가입 Lambda 트리거를 사용하여 사용자 풀에 가입하는 새 사용자를 검증하는 사용자 지정 로직을 추가할 수 있습니다. 새 사용자를 등록하는 방법을 보여주는 샘플 JavaScript 프로그램입니다. 인증 과정에서 사전 가입 Lambda 트리거를 호출합니다.

JavaScript

```
var attributeList = [];
var dataEmail = {
  Name: "email",
  Value: "...", // your email here
};
var dataPhoneNumber = {
  Name: "phone_number",
  Value: "...", // your phone number here with +country code and no delimiters in
 front
};

var dataEmailDomain = {
```

```
  Name: "custom:domain",
  Value: "example.com",
};
var attributeEmail = new AmazonCognitoIdentity.CognitoUserAttribute(dataEmail);
var attributePhoneNumber = new AmazonCognitoIdentity.CognitoUserAttribute(
  dataPhoneNumber
);
var attributeEmailDomain = new AmazonCognitoIdentity.CognitoUserAttribute(
  dataEmailDomain
);

attributeList.push(attributeEmail);
attributeList.push(attributePhoneNumber);
attributeList.push(attributeEmailDomain);

var cognitoUser;
userPool.signUp(
  "username",
  "password",
  attributeList,
  null,
  function (err, result) {
    if (err) {
      alert(err);
      return;
    }
    cognitoUser = result.user;
    console.log("user name is " + cognitoUser.getUsername());
  }
);
```

이것은 사용자 풀 사전 가입 Lambda 트리거로 가입하기 직전에 호출되는 샘플 Lambda 트리거입니다. 이 트리거는 사용자 지정 속성 custom:domain을 사용하여 특정 이메일 도메인의 신규 사용자를 자동으로 확인할 수 있습니다. 사용자 지정 도메인에 없는 모든 신규 사용자가 사용자 풀에 추가되지만 자동으로 확인되지는 않습니다.

Node.js

```
exports.handler = (event, context, callback) => {
  // Set the user pool autoConfirmUser flag after validating the email domain
  event.response.autoConfirmUser = false;
```

```
  // Split the email address so we can compare domains
  var address = event.request.userAttributes.email.split("@");

  // This example uses a custom attribute "custom:domain"
  if (event.request.userAttributes.hasOwnProperty("custom:domain")) {
    if (event.request.userAttributes["custom:domain"] === address[1]) {
      event.response.autoConfirmUser = true;
    }
  }

  // Return to Amazon Cognito
  callback(null, event);
};
```

Python

```
def lambda_handler(event, context):
    # It sets the user pool autoConfirmUser flag after validating the email domain
    event['response']['autoConfirmUser'] = False

    # Split the email address so we can compare domains
    address = event['request']['userAttributes']['email'].split('@')

    # This example uses a custom attribute 'custom:domain'
    if 'custom:domain' in event['request']['userAttributes']:
        if event['request']['userAttributes']['custom:domain'] == address[1]:
            event['response']['autoConfirmUser'] = True

    # Return to Amazon Cognito
    return event
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그런 다음 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. Lambda 콘솔에서 해당 Lambda 트리거와 관련 있는 데이터로 테스트 이벤트를 설정할 수 있습니다. 다음은 이 코드 샘플의 테스트 이벤트입니다.

JSON

```
{
    "request": {
        "userAttributes": {
            "email": "testuser@example.com",
            "custom:domain": "example.com"
```

```
        }
    },
    "response": {}
}
```

사전 가입 예제: 모든 사용자 자동 확인 및 자동 검증

이 예에서는 모든 사용자를 자동으로 컨펌하고, 속성이 있는 경우에 확인된 것으로 사용자의 email 및 phone_number 속성을 설정합니다. 그리고 별칭 기능이 활성화되어 있으면 자동 확인이 설정될 때 phone_number 및 email에 대한 별칭이 생성됩니다.

> **Note**
>
> 해당 전화 번호의 별칭이 이미 있으면 별칭이 새 사용자로 이동하고 이전 사용자의 phone_number는 확인되지 않은 것으로 표시됩니다. 이메일 주소도 마찬가지입니다. 이런 일이 발생하지 않도록 사용자 풀 ListUsers API를 사용하여 이미 새 사용자의 전화번호나 이메일 주소를 별칭으로 사용하고 있는 기존 사용자가 있는지 확인할 수 있습니다.

Node.js

```
const handler = async (event) => {
  // Confirm the user
  event.response.autoConfirmUser = true;
  // Set the email as verified if it is in the request
  if (event.request.userAttributes.hasOwnProperty("email")) {
    event.response.autoVerifyEmail = true;
  }

  // Set the phone number as verified if it is in the request
  if (event.request.userAttributes.hasOwnProperty("phone_number")) {
    event.response.autoVerifyPhone = true;
  }

  return event;
};

export { handler };
```

Python

```
def lambda_handler(event, context):
    # Confirm the user
    event['response']['autoConfirmUser'] = True

    # Set the email as verified if it is in the request
    if 'email' in event['request']['userAttributes']:
        event['response']['autoVerifyEmail'] = True

    # Set the phone number as verified if it is in the request
    if 'phone_number' in event['request']['userAttributes']:
        event['response']['autoVerifyPhone'] = True

    # Return to Amazon Cognito
    return event
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그런 다음 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. Lambda 콘솔에서 해당 Lambda 트리거와 관련 있는 데이터로 테스트 이벤트를 설정할 수 있습니다. 다음은 이 코드 샘플의 테스트 이벤트입니다.

JSON

```
{
  "request": {
    "userAttributes": {
      "email": "user@example.com",
      "phone_number": "+12065550100"
    }
  },
  "response": {}
}
```

사전 가입 예: 사용자 이름이 5자 미만인 경우 가입 거부

이 예에서는 가입 요청에서 사용자 이름의 길이를 확인합니다. 이 예에서는 사용자가 5자 미만의 이름을 입력한 경우 오류를 반환합니다.

Node.js

```
exports.handler = (event, context, callback) => {
    // Impose a condition that the minimum length of the username is 5 is imposed on
 all user pools.
    if (event.userName.length < 5) {
        var error = new Error("Cannot register users with username less than the
 minimum length of 5");
        // Return error to Amazon Cognito
        callback(error, event);
    }
    // Return to Amazon Cognito
    callback(null, event);
};
```

Python

```
def lambda_handler(event, context):
    if len(event['userName']) < 5:
        raise Exception("Cannot register users with username less than the minimum
 length of 5")
    # Return to Amazon Cognito
    return event
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그런 다음 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. Lambda 콘솔에서 해당 Lambda 트리거와 관련 있는 데이터로 테스트 이벤트를 설정할 수 있습니다. 다음은 이 코드 샘플의 테스트 이벤트입니다.

JSON

```
{
  "userName": "rroe",
  "response": {}
}
```

사후 확인 Lambda 트리거

Amazon Cognito는 가입한 사용자가 사용자 계정을 확인한 후 이 트리거를 호출합니다. 사후 확인 Lambda 함수에서 사용자 지정 메시지를 보내거나 사용자 지정 API 요청을 추가할 수 있습니다. 예를

들어 외부 시스템을 쿼리하고 사용자에게 추가 속성을 채울 수 있습니다. Amazon Cognito는 사용자 풀에 등록한 사용자에 대해서만 이 트리거를 호출하고 관리자 보안 인증 정보로 생성한 사용자 계정에 대해서는 호출하지 않습니다.

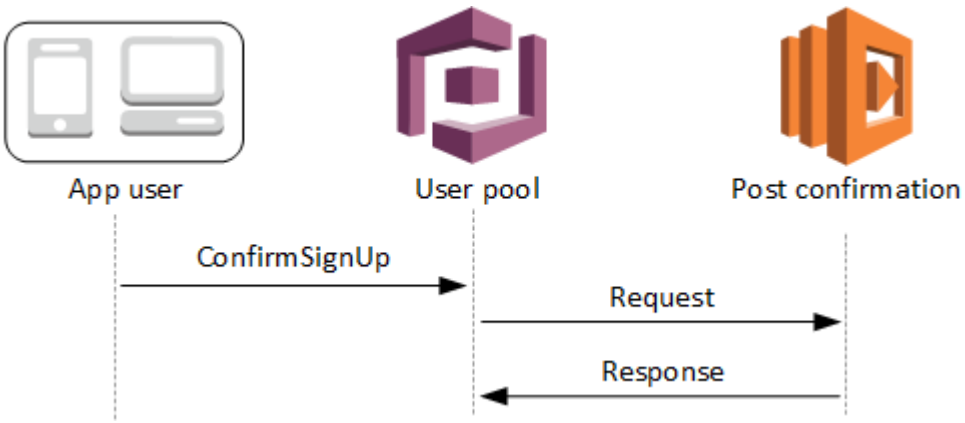
요청에는 확인된 사용자의 현재 속성이 포함됩니다.

주제

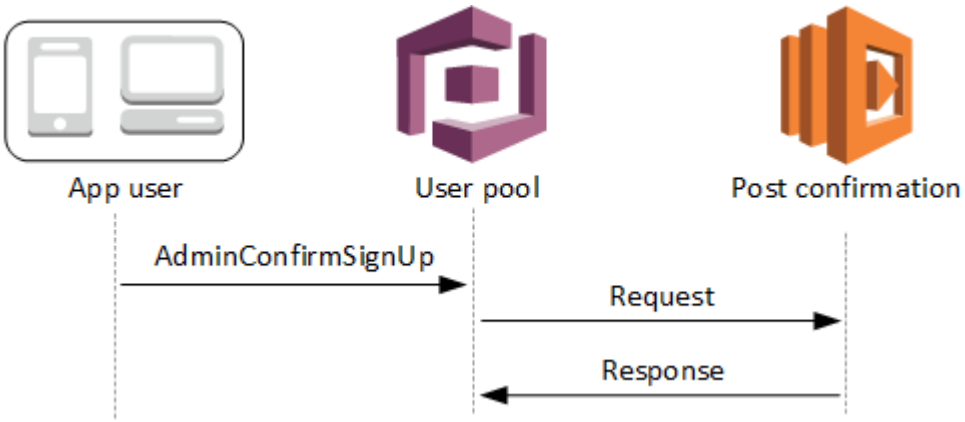
- 사후 확인 Lambda 흐름
- 사후 확인 Lambda 트리거 파라미터
- 사용자 확인 자습서
- 사후 확인 예제

사후 확인 Lambda 흐름

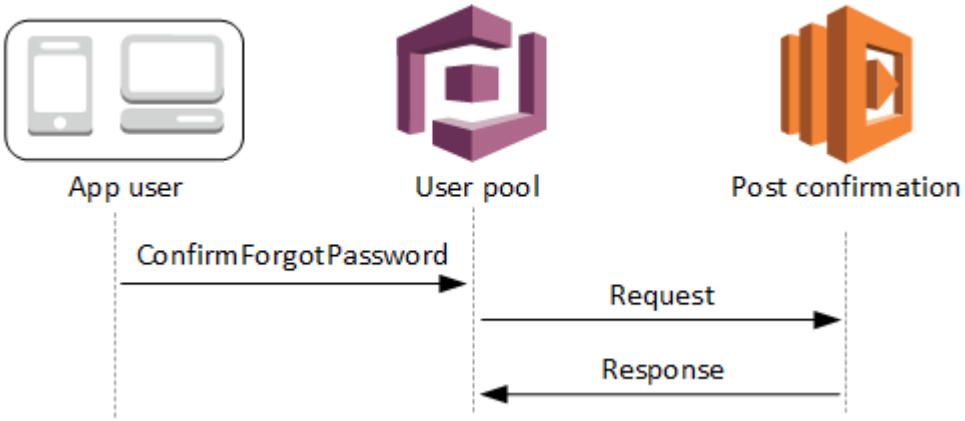
클라이언트 확인 가입 흐름



서버 확인 가입 흐름



암호 분실 확인 흐름



사후 확인 Lambda 트리거 파라미터

Amazon Cognito가 이 Lambda 함수에 전달하는 요청은 아래 파라미터와 Amazon Cognito가 모든 요청에 추가하는 공통 파라미터의 조합입니다.

JSON

```
{
    "request": {
            "userAttributes": {
                "string": "string",
                . . .
            },
            "clientMetadata": {
             "string": "string",
             . . .
            }
        },
    "response": {}
}
```

사후 확인 요청 파라미터

userAttributes

사용자 속성을 나타내는 하나 이상의 키-값 페어입니다.

clientMetadata

사후 확인 트리거에 지정하는 Lambda 함수에 사용자 지정 입력으로 제공할 수 있는 하나 이상의 키-값 페어입니다. 다음 API 작업에서 ClientMetadata 파라미터를 사용하여 이 데이터를 Lambda 함수에 전달할 수 있습니다. AdminConfirmSignUp, ConfirmForgotPassword, ConfirmSignUp 및 SignUp.

사후 확인 응답 파라미터

응답에는 추가적인 반환 정보가 없습니다.

사용자 확인 자습서

사후 확인 Lambda 함수는 Amazon Cognito가 새로운 사용자를 확인한 직후에 시작됩니다. JavaScript, Android 및 iOS용 사용자 확인 자습서를 참조하세요.

플랫폼	자습서
JavaScript 자격 증명 SDK	JavaScript 사용자 확인
Android 자격 증명 SDK	Android 사용자 확인
iOS 자격 증명 SDK	iOS 사용자 확인

사후 확인 예제

이 예에서 Lambda 함수는 Amazon SES를 사용하여 사용자에게 확인 이메일 메시지를 전송합니다. 자세한 내용은 Amazon Simple Storage Service 개발자 가이드를 참조하세요.

Node.js

```
// Import required AWS SDK clients and commands for Node.js. Note that this requires
// the `@aws-sdk/client-ses` module to be either bundled with this code or included
// as a Lambda layer.
import { SES, SendEmailCommand } from "@aws-sdk/client-ses";
const ses = new SES();

const handler = async (event) => {
  if (event.request.userAttributes.email) {
```

```
    await sendTheEmail(
      event.request.userAttributes.email,
      `Congratulations ${event.userName}, you have been confirmed.`
    );
  }
  return event;
};

const sendTheEmail = async (to, body) => {
  const eParams = {
    Destination: {
      ToAddresses: [to],
    },
    Message: {
      Body: {
        Text: {
          Data: body,
        },
      },
      Subject: {
        Data: "Cognito Identity Provider registration completed",
      },
    },
    // Replace source_email with your SES validated email address
    Source: "<source_email>",
  };
  try {
    await ses.send(new SendEmailCommand(eParams));
  } catch (err) {
    console.log(err);
  }
};

export { handler };
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그런 다음 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. Lambda 콘솔에서 해당 Lambda 트리거와 관련 있는 데이터로 테스트 이벤트를 설정할 수 있습니다. 다음은 이 코드 샘플의 테스트 이벤트입니다.

JSON

```
{
    "request": {
        "userAttributes": {
            "email": "user@example.com",
            "email_verified": true
        }
    },
    "response": {}
}
```

사전 인증 Lambda 트리거

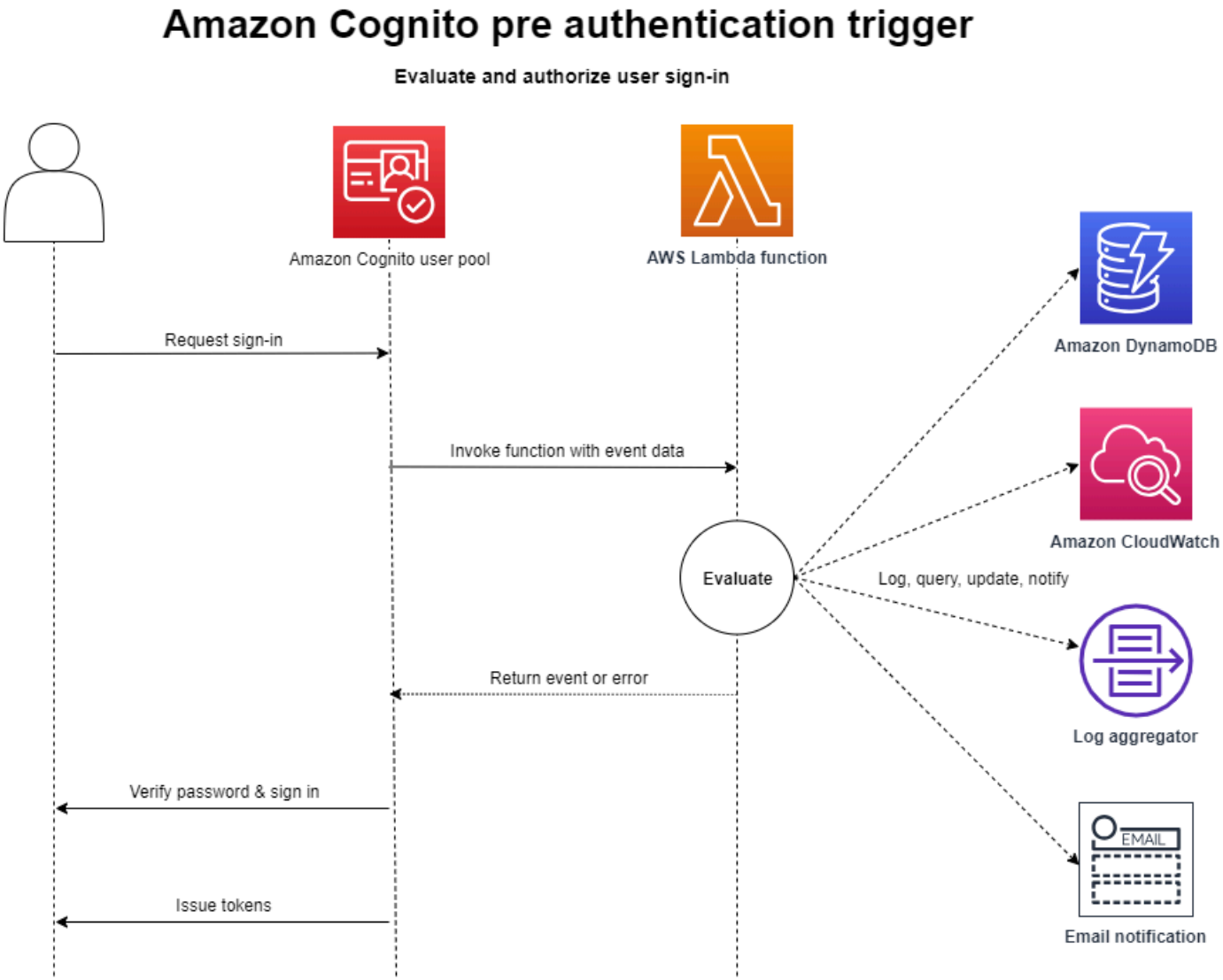
예비 작업을 수행하는 사용자 지정 검증을 생성할 수 있도록 사용자가 로그인을 시도할 때 Amazon Cognito가 이 트리거를 호출합니다. 예를 들어 인증 요청을 거부하거나 세션 데이터를 외부 시스템에 기록할 수 있습니다.

> **Note**
>
> 이 Lambda 트리거는 사용자가 존재하지 않거나 사용자 풀에 이미 기존 세션이 있는 경우에는 활성화되지 않습니다. 사용자 풀 앱 클라이언트의 `PreventUserExistenceErrors` 설정이 `ENABLED`로 설정된 경우 Lambda 트리거가 활성화됩니다.

주제

- 인증 흐름 개요
- 사전 인증 Lambda 트리거 파라미터
- 사전 인증 예제

인증 흐름 개요



요청에는 앱이 사용자 풀 `InitiateAuth` 및 `AdminInitiateAuth` API 작업에 전달하는 `ClientMetadata` 값에 있는 클라이언트 검증 데이터가 포함됩니다.

자세한 내용은 사용자 풀 인증 흐름 섹션을 참조하세요.

사전 인증 Lambda 트리거 파라미터

Amazon Cognito가 이 Lambda 함수에 전달하는 요청은 아래 파라미터와 Amazon Cognito가 모든 요청에 추가하는 공통 파라미터의 조합입니다.

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
        },
        "validationData": {
            "string": "string",
            . . .
        },
        "userNotFound": boolean
    },
    "response": {}
}
```

사전 인증 요청 파라미터

userAttributes

사용자 속성을 나타내는 하나 이상의 이름-값 페어입니다.

userNotFound

사용자 풀 클라이언트에 대해 `PreventUserExistenceErrors`를 `ENABLED`로 설정한 경우 Amazon Cognito에서 이 부울을 채웁니다.

validationData

사용자 로그인 요청에 검증 데이터를 포함하는 하나 이상의 키-값 페어입니다. 이 데이터를 Lambda 함수에 전달하려면 InitiateAuth 및 AdminInitiateAuth API 작업에서 ClientMetadata 파라미터를 사용합니다.

사전 인증 응답 파라미터

Amazon Cognito는 응답에서 추가 반환 정보를 기대하지 않습니다. 함수가 오류를 반환하여 로그인 시도를 거부하거나 API 작업을 사용하여 리소스를 쿼리하고 수정할 수 있습니다.

사전 인증 예제

이 예제 함수는 사용자가 특정 앱 클라이언트로 사용자 풀에 로그인하지 못하도록 합니다. 사전 인증 Lambda 함수는 사용자에게 기존 세션이 있을 때는 호출하지 않기 때문에 이 함수는 차단하려는 앱 클라이언트 ID가 있는 새 세션만 방지합니다.

Node.js

```
const handler = async (event) => {
  if (
    event.callerContext.clientId === "user-pool-app-client-id-to-be-blocked"
  ) {
    throw new Error("Cannot authenticate users from this user pool app client");
  }

  return event;
};

export { handler };
```

Python

```
def lambda_handler(event, context):
    if event['callerContext']['clientId'] == "<user pool app client id to be
 blocked>":
        raise Exception("Cannot authenticate users from this user pool app client")

    # Return to Amazon Cognito
    return event
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그런 다음 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. Lambda 콘솔에서 해당 Lambda 트리거와 관련 있는 데이터로 테스트 이벤트를 설정할 수 있습니다. 다음은 이 코드 샘플의 테스트 이벤트입니다.

JSON

```
{
    "callerContext": {
        "clientId": "<user pool app client id to be blocked>"
    },
    "response": {}
```

```
}
```

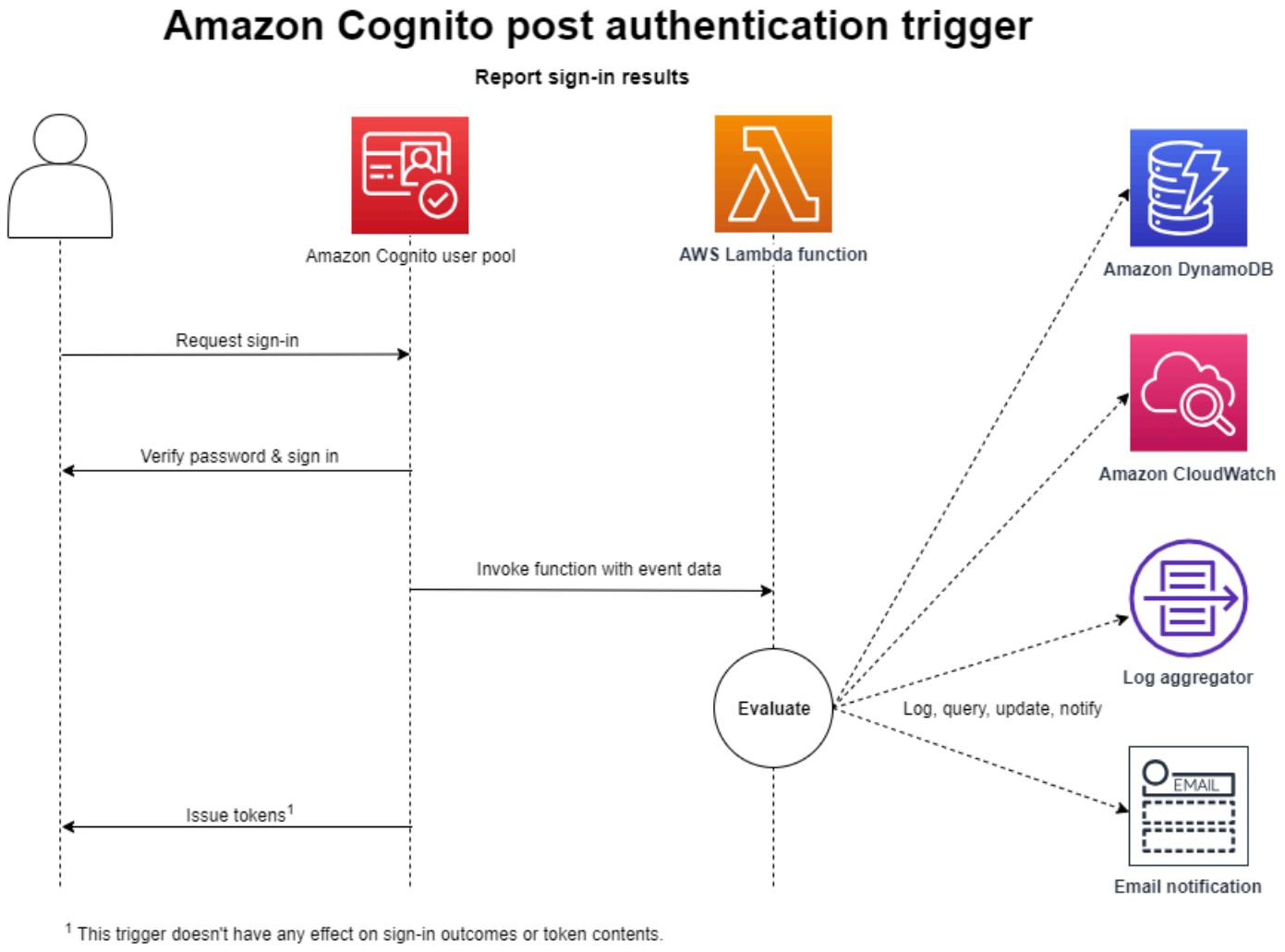
사후 인증 Lambda 트리거

Amazon Cognito는 사용자를 로그인한 후 이 트리거를 호출하기 때문에 Amazon Cognito에서 사용자를 인증한 후 사용자 지정 로직을 추가할 수 있습니다.

주제

- 인증 흐름 개요
- 사후 인증 Lambda 트리거 파라미터
- 인증 자습서
- 사후 인증 예제

인증 흐름 개요



자세한 내용은 사용자 풀 인증 흐름 섹션을 참조하세요.

사후 인증 Lambda 트리거 파라미터

Amazon Cognito가 이 Lambda 함수에 전달하는 요청은 아래 파라미터와 Amazon Cognito가 모든 요청에 추가하는 공통 파라미터의 조합입니다.

JSON

```
{
    "request": {
        "userAttributes": {
```

```
            "string": "string",
            . . .
        },
        "newDeviceUsed": boolean,
        "clientMetadata": {
            "string": "string",
            . . .
           }
       },
    "response": {}
}
```

사후 인증 요청 파라미터

newDeviceUsed

이 플래그는 사용자가 새로운 디바이스에 로그인했는지 여부를 표시합니다. 기억된 사용자 풀의 디바이스 값이 `Always` 또는 `User Opt-In`으로 설정된 경우에만 Amazon Cognito에서 이 플래그를 설정합니다.

userAttributes

사용자 속성을 나타내는 하나 이상의 이름-값 페어입니다.

clientMetadata

사후 인증 트리거에 지정하는 Lambda 함수에 사용자 지정 입력으로 제공할 수 있는 하나 이상의 키-값 페어입니다. 이 데이터를 Lambda 함수에 전달하려면 AdminRespondToAuthChallenge 및 RespondToAuthChallenge API 작업에서 ClientMetadata 파라미터를 사용합니다. Amazon Cognito는 사후 승인 함수에 전달하는 요청에 있는 AdminInitiateAuth 및 InitiateAuth API 작업의 ClientMetadata 파라미터에서 전달된 데이터를 포함하지 않습니다.

사후 인증 응답 파라미터

Amazon Cognito는 응답에서 추가 반환 정보를 기대하지 않습니다. 함수는 API 작업을 사용하여 리소스를 쿼리 및 수정하거나 이벤트 메타데이터를 외부 시스템에 기록할 수 있습니다.

인증 자습서

Amazon Cognito가 사용자를 로그인한 직후에 사후 인증 Lambda 함수를 활성화합니다. JavaScript, Android 및 iOS용 로그인 자습서를 참조하세요.

플랫폼	자습서
JavaScript 자격 증명 SDK	JavaScript 사용자 로그인
Android 자격 증명 SDK	Android 사용자 로그인
iOS 자격 증명 SDK	iOS 사용자 로그인

사후 인증 예제

이 사후 인증 Lambda 함수 샘플은 CloudWatch Logs에 성공적으로 로그인하여 데이터를 전송합니다.

Node.js

```
const handler = async (event) => {
  // Send post authentication data to Amazon CloudWatch logs
  console.log("Authentication successful");
  console.log("Trigger function =", event.triggerSource);
  console.log("User pool = ", event.userPoolId);
  console.log("App client ID = ", event.callerContext.clientId);
  console.log("User ID = ", event.userName);

  return event;
};

export { handler }
```

Python

```
import os
def lambda_handler(event, context):

    # Send post authentication data to Cloudwatch logs
    print ("Authentication successful")
    print ("Trigger function =", event['triggerSource'])
    print ("User pool = ", event['userPoolId'])
    print ("App client ID = ", event['callerContext']['clientId'])
    print ("User ID = ", event['userName'])
```

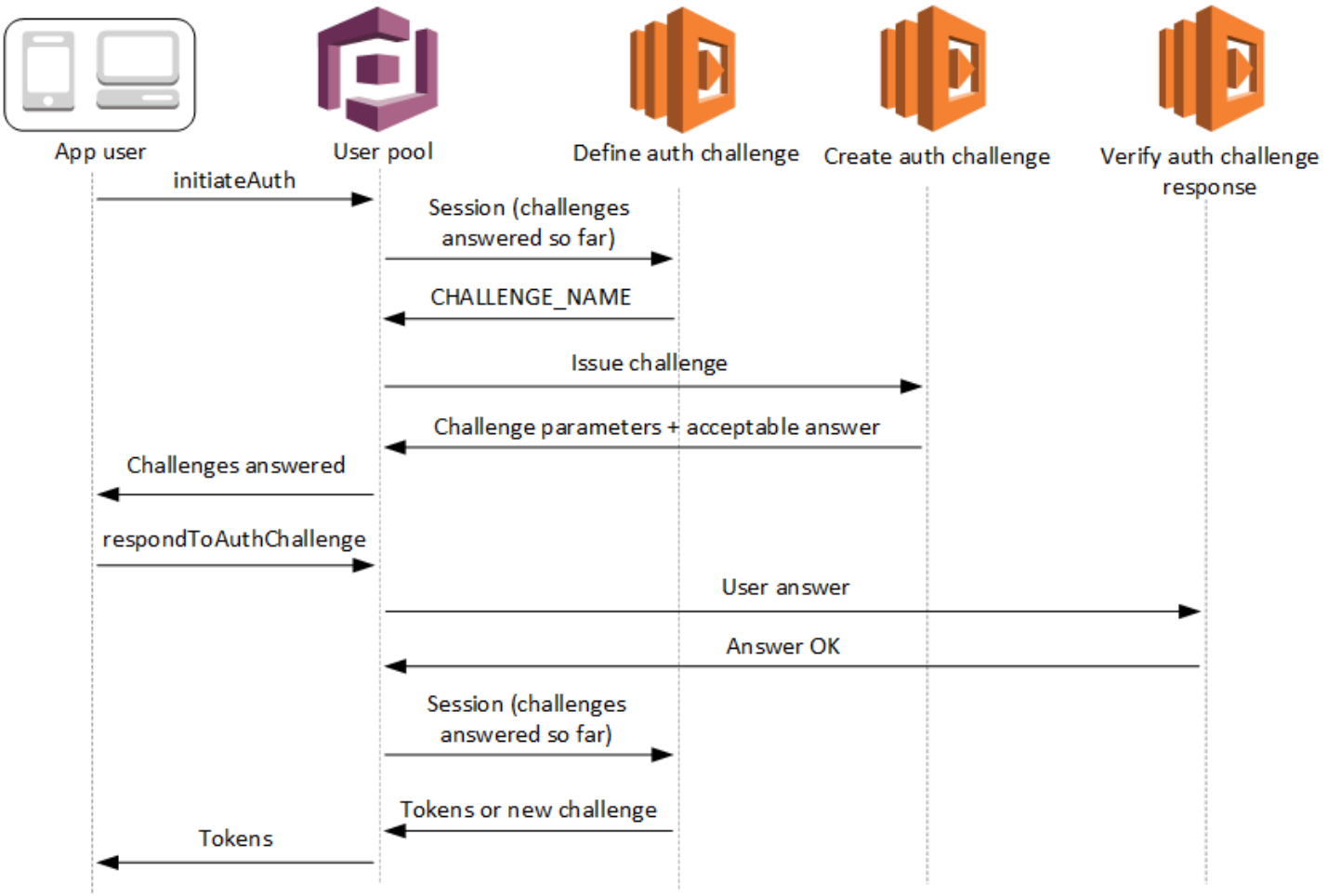
```
    # Return to Amazon Cognito
    return event
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그런 다음 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. Lambda 콘솔에서 해당 Lambda 트리거와 관련 있는 데이터로 테스트 이벤트를 설정할 수 있습니다. 다음은 이 코드 샘플의 테스트 이벤트입니다.

JSON

```
{
  "triggerSource": "testTrigger",
  "userPoolId": "testPool",
  "userName": "testName",
  "callerContext": {
      "clientId": "12345"
  },
  "response": {}
}
```

사용자 정의 인증 챌린지 Lambda 트리거



이러한 Lambda 트리거는 자체 문제를 사용자 풀 사용자 지정 인증 흐름의 일부로 표시하고 확인합니다.

인증 문제 정의

Amazon Cognito가 이 트리거를 호출하여 사용자 지정 인증 흐름을 시작합니다.

인증 문제 생성

Amazon Cognito는 이 트리거를 인증 문제 정의 다음에 호출하여 사용자 지정 문제를 생성합니다.

인증 문제 응답 확인

Amazon Cognito는 이 트리거를 호출하여 사용자 지정 문제에 대한 최종 사용자의 응답이 유효한지 여부를 확인합니다.

새로운 문제 유형을 이러한 문제 Lambda 트리거와 통합할 수 있습니다. 예를 들어, 이러한 문제 유형에 CAPTCHA 또는 동적 문제 질문이 포함될 수 있습니다.

사용자 풀 InitiateAuth와 RespondToAuthChallenge API 메서드가 있는 두 개의 공통 단계로 인증을 일반화할 수 있습니다.

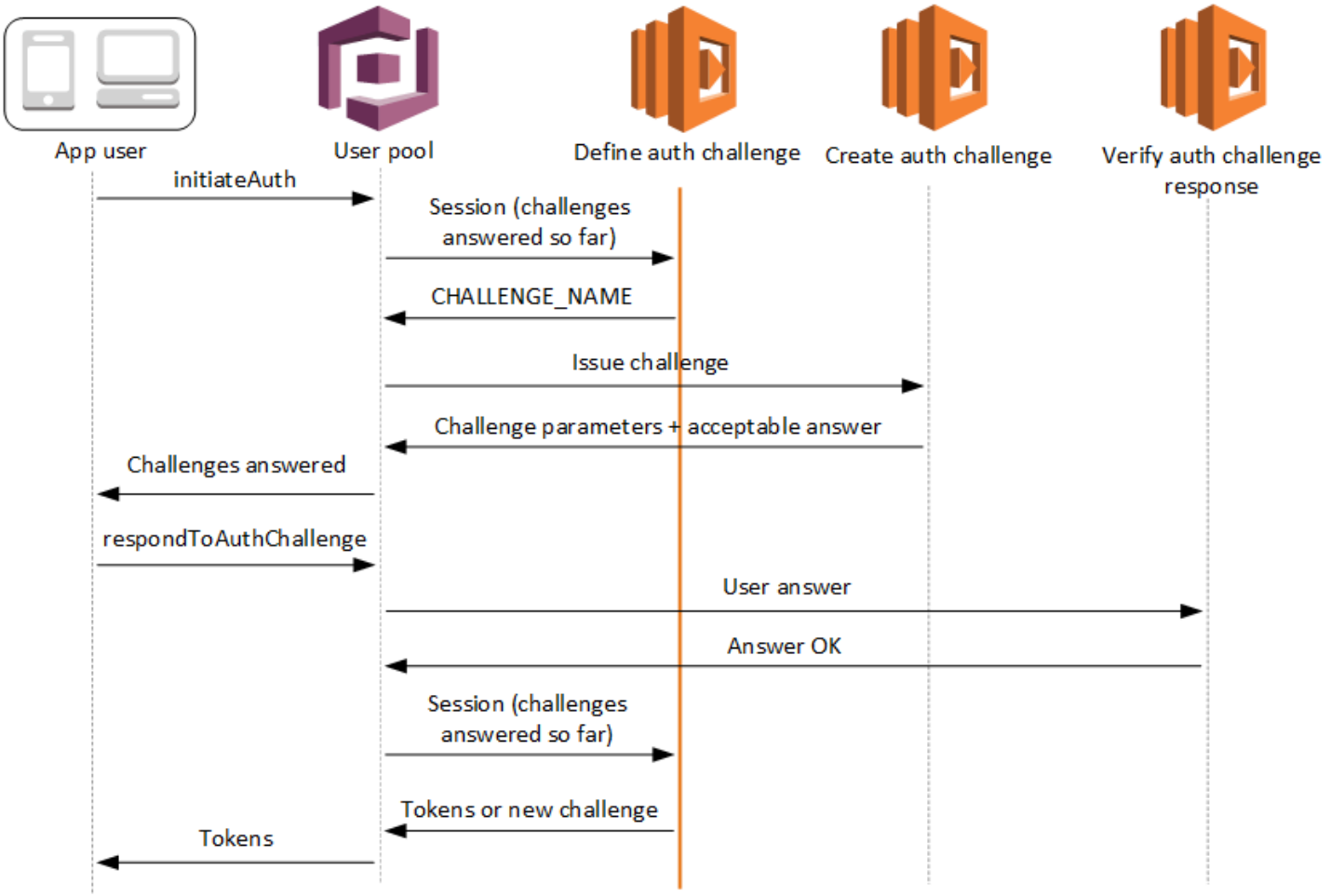
이 흐름에서 인증이 실패하거나 사용자에게 토큰이 발행될 때까지 사용자는 연속 문제에 응답하여 인증합니다. 이 두 API 호출을 반복하여 서로 다른 문제를 포함시킬 수 있습니다.

> **Note**
>
> Amazon Cognito 호스팅 UI는 사용자 지정 인증 문제 Lambda 트리거를 통한 사용자 지정 인증을 지원하지 않습니다.

주제

- 인증 챌린지 정의 Lambda 트리거
- 인증 챌린지 생성 Lambda 트리거
- 인증 챌린지 확인 응답 Lambda 트리거

인증 챌린지 정의 Lambda 트리거



인증 문제 정의

Amazon Cognito가 이 트리거를 호출하여 사용자 지정 인증 흐름을 시작합니다.

이 Lambda 트리거에 대한 요청에는 `session`이 포함됩니다. `session` 파라미터는 현재 인증 프로세스에서 사용자에게 표시되는 모든 문제가 포함된 배열입니다. 해당하는 결과를 포함하는 어레이입니다. `session` 배열에는 문제 세부 정보(`ChallengeResult`)가 시간순으로 저장됩니다. 문제 `session[0]`은 사용자가 수신하는 첫 번째 문제를 나타냅니다.

Amazon Cognito가 사용자 지정 문제를 표시하기 전에 사용자 암호를 확인하도록 할 수 있습니다. request-rate 할당량의 인증 카테고리에 연결된 Lambda 트리거는 사용자 지정 문제 흐름에서 SRP 인증을 수행할 때 실행됩니다. 다음은 이 프로세스의 개요입니다.

1. 앱이 AuthParameters 맵으로 InitiateAuth 또는 AdminInitiateAuth를 호출하여 로그인을 시작합니다. 파라미터에는 CHALLENGE_NAME: SRP_A,와, SRP_A 및 USERNAME에 대한 값이 포함되어야 합니다.
2. Amazon Cognito는 challengeName: SRP_A 및 challengeResult: true를 포함하는 초기 세션과 함께 인증 문제 정의 Lambda 트리거를 호출합니다.
3. 이러한 입력을 수신한 후 Lambda 함수는 challengeName: PASSWORD_VERIFIER, issueTokens: false, failAuthentication: false로 응답합니다.
4. 암호 확인이 성공하면 Amazon Cognito가 challengeName: PASSWORD_VERIFIER 및 challengeResult: true가 포함된 새 세션을 사용하여 Lambda 함수를 다시 호출합니다.
5. 사용자 지정 문제를 시작하려면 Lambda 함수가 challengeName: CUSTOM_CHALLENGE, issueTokens: false 및 failAuthentication: false로 응답합니다. 암호 확인을 사용하여 사용자 지정 인증 흐름을 시작하지 않으려면 CHALLENGE_NAME: CUSTOM_CHALLENGE를 포함한 AuthParameters 맵으로 로그인을 시작하면 됩니다.
6. 모든 챌린지에 응답할 때까지 챌린지 루프가 반복됩니다.

주제

- 인증 챌린지 정의 Lambda 트리거 파라미터
- 인증 챌린지 정의 예제

인증 챌린지 정의 Lambda 트리거 파라미터

Amazon Cognito가 이 Lambda 함수에 전달하는 요청은 아래 파라미터와 Amazon Cognito가 모든 요청에 추가하는 공통 파라미터의 조합입니다.

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
                . . .
        },
        "session": [
            ChallengeResult,
            . . .
        ],
        "clientMetadata": {
```

```
            "string": "string",
            . . .
        },
        "userNotFound": boolean
    },
    "response": {
        "challengeName": "string",
        "issueTokens": boolean,
        "failAuthentication": boolean
    }
}
```

인증 챌린지 정의 요청 파라미터

Amazon Cognito에서 Lambda 함수를 호출하면 Amazon Cognito가 다음과 같은 파라미터를 제공합니다.

userAttributes

사용자 속성을 나타내는 하나 이상의 이름-값 페어입니다.

userNotFound

사용자 풀 클라이언트에 대해 `PreventUserExistenceErrors`가 `ENABLED`로 설정된 경우 Amazon Cognito에서 채우는 부울입니다. 값이 `true`이면 해당 사용자 ID(사용자 이름, 이메일 주소, 기타 세부 정보)와 일치하는 기존 사용자가 없는 것입니다. `PreventUserExistenceErrors`가 `ENABLED`로 설정된 경우 서비스는 존재하지 않는 사용자를 앱에 알리지 않습니다. Lambda 함수에서 동일한 사용자 경험을 유지하고 대기 시간을 고려하는 것이 좋습니다. 이렇게 하면 호출자가 사용자의 존재 여부에 상관없이 동일한 동작을 감지할 수 있습니다.

세션

`ChallengeResult` 요소 배열입니다. 각각 다음 요소가 포함됩니다.

challengeName

문제 유형 `CUSTOM_CHALLENGE`, `SRP_A`, `PASSWORD_VERIFIER`, `SMS_MFA`, `DEVICE_SRP_AUTH`, `DEVICE_PASSWORD_VERIFIER`, `ADMIN_NO_SRP_AUTH` 중 하나입니다.

인증 챌린지 정의 함수가 멀티 팩터 인증을 설정한 사용자에게 `PASSWORD_VERIFIER` 챌린지를 발급하면 Amazon Cognito는 `SMS_MFA` 챌린지로 후속 작업을 실행합니다. 함수에 `SMS_MFA` 챌

린지의 입력 이벤트 처리를 포함하세요. 인증 챌린지 정의 함수에서 SMS_MFA 챌린지를 호출할 필요는 없습니다.

> **Important**
>
> 사용자가 성공적으로 인증되었고 토큰을 발급받아야 하는지 여부를 함수가 결정할 때 인증 챌린지 정의 함수에서 항상 challengeName을 확인하여 예상 값과 일치하는지 확인해야 합니다.

challengeResult

사용자가 챌린지를 성공적으로 완료하면 true로 설정하고 그렇지 않으면 false로 설정합니다.

challengeMetadata

사용자 지정 챌린지의 이름입니다. challengeName이 CUSTOM_CHALLENGE인 경우에만 사용됩니다.

clientMetadata

인증 문제 정의 트리거에 지정하는 Lambda 함수에 사용자 지정 입력으로 제공할 수 있는 하나 이상의 키-값 쌍입니다. 이 데이터를 Lambda 함수에 전달하려면 AdminRespondToAuthChallenge 및 RespondToAuthChallenge API 작업에서 ClientMetadata 파라미터를 사용합니다. 인증 문제 정의 함수를 호출하는 요청에는 AdminInitiateAuth 및 InitiateAuth API 작업의 ClientMetadata 파라미터에 전달된 데이터가 포함되지 않습니다.

인증 챌린지 정의 응답 파라미터

응답에서 인증 프로세스의 다음 단계를 반환할 수 있습니다.

challengeName

다음 문제의 이름이 포함된 문자열입니다. 사용자에게 새로운 챌린지를 표시하려면 여기에 챌린지 이름을 지정하세요.

issueTokens

사용자가 인증 문제를 충분히 완료했다고 확인한 경우 true로 설정합니다. 사용자가 문제를 충분히 충족하지 못한 경우 false로 설정합니다.

failAuthentication

현재 인증 프로세스를 종료하려는 경우 `true`로 설정합니다. 현재 인증 프로세스를 계속하려면 `false`로 설정합니다.

인증 챌린지 정의 예제

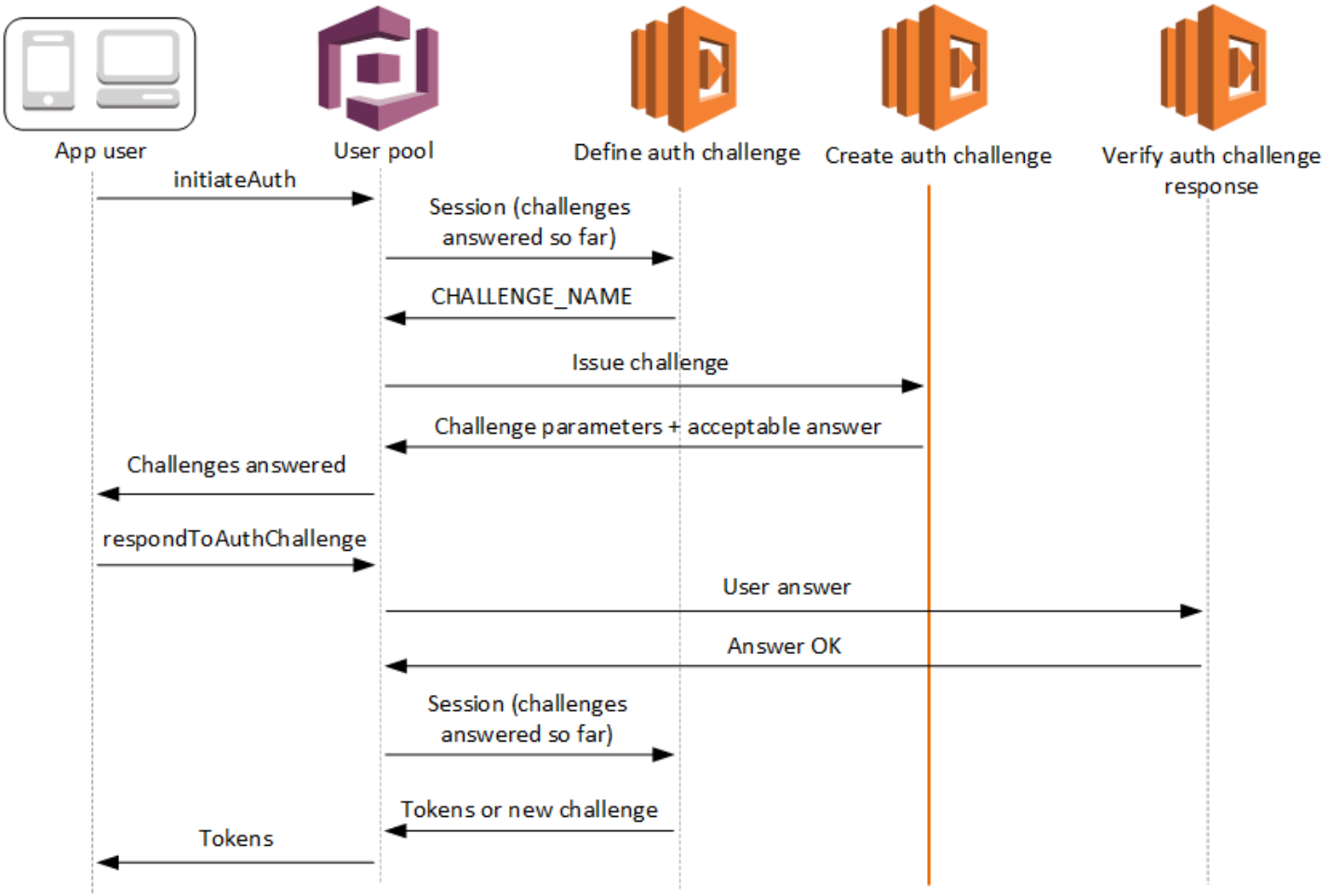
이 예에서는 인증을 위한 일련의 문제를 정의하고 사용자가 모든 문제를 성공적으로 완료한 경우에만 토큰을 발행합니다.

Node.js

```
const handler = async (event) => {
  if (
    event.request.session.length == 1 &&
    event.request.session[0].challengeName == "SRP_A"
  ) {
    event.response.issueTokens = false;
    event.response.failAuthentication = false;
    event.response.challengeName = "PASSWORD_VERIFIER";
  } else if (
    event.request.session.length == 2 &&
    event.request.session[1].challengeName == "PASSWORD_VERIFIER" &&
    event.request.session[1].challengeResult == true
  ) {
    event.response.issueTokens = false;
    event.response.failAuthentication = false;
    event.response.challengeName = "CUSTOM_CHALLENGE";
  } else if (
    event.request.session.length == 3 &&
    event.request.session[2].challengeName == "CUSTOM_CHALLENGE" &&
    event.request.session[2].challengeResult == true
  ) {
    event.response.issueTokens = false;
    event.response.failAuthentication = false;
    event.response.challengeName = "CUSTOM_CHALLENGE";
  } else if (
    event.request.session.length == 4 &&
    event.request.session[3].challengeName == "CUSTOM_CHALLENGE" &&
    event.request.session[3].challengeResult == true
  ) {
    event.response.issueTokens = true;
    event.response.failAuthentication = false;
```

```
  } else {
    event.response.issueTokens = false;
    event.response.failAuthentication = true;
  }

  return event;
};

export { handler }
```

인증 챌린지 생성 Lambda 트리거



인증 문제 생성

인증 문제 정의 트리거의 일부로 사용자 지정 문제를 지정한 경우 Amazon Cognito가 인증 문제 정의 이후에 이 트리거를 호출합니다. 사용자 지정 인증 흐름을 생성합니다.

사용자에게 표시할 문제를 만들기 위해 이 Lambda 트리거가 호출됩니다. 이 Lambda 트리거에 대한 요청에는 `challengeName` 및 `session`이 포함됩니다. `challengeName`은 문자열이며 사용자에게 표시할 다음 챌린지의 이름입니다. 인증 문제 정의 Lambda 트리거에서 이 속성의 값이 설정됩니다.

모든 챌린지에 응답할 때까지 챌린지 루프가 반복됩니다.

주제

- 인증 챌린지 생성 Lambda 트리거 파라미터
- 인증 챌린지 생성 예제

인증 챌린지 생성 Lambda 트리거 파라미터

Amazon Cognito가 이 Lambda 함수에 전달하는 요청은 아래 파라미터와 Amazon Cognito가 모든 요청에 추가하는 공통 파라미터의 조합입니다.

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
        },
        "challengeName": "string",
        "session": [
            ChallengeResult,
            . . .
        ],
        "clientMetadata": {
            "string": "string",
            . . .
        },
        "userNotFound": boolean
    },
    "response": {
        "publicChallengeParameters": {
            "string": "string",
            . . .
        },
        "privateChallengeParameters": {
            "string": "string",
            . . .
```

```
        },
        "challengeMetadata": "string"
    }
}
```

인증 챌린지 생성 요청 파라미터

userAttributes

사용자 속성을 나타내는 하나 이상의 이름-값 페어입니다.

userNotFound

이 부울은 `PreventUserExistenceErrors`가 사용자 풀 클라이언트에 대해 `ENABLED`로 설정된 경우에 채워집니다.

challengeName

새로운 챌린지의 이름입니다.

세션

세션 요소는 각각 다음 요소를 포함하는 `ChallengeResult` 요소의 어레이입니다.

challengeName

챌린지 유형입니다. `"CUSTOM_CHALLENGE"`, `"PASSWORD_VERIFIER"`, `"SMS_MFA"`, `"DEVICE_SRP_AUTH"`, `"DEVICE_PASSWORD_VERIFIER"` 또는 `"ADMIN_NO_SRP_AUTH"` 중 하나입니다.

challengeResult

사용자가 챌린지를 성공적으로 완료하면 `true`로 설정하고 그렇지 않으면 `false`로 설정합니다.

challengeMetadata

사용자 지정 챌린지의 이름입니다. `challengeName`이 `"CUSTOM_CHALLENGE"`인 경우에만 사용됩니다.

clientMetadata

인증 문제 생성 트리거에 지정하는 Lambda 함수에 사용자 지정 입력으로 제공할 수 있는 하나 이상의 키-값 페어입니다. AdminRespondToAuthChallenge 및 RespondToAuthChallenge API 작업에서

ClientMetadata 파라미터를 사용하여 이 데이터를 Lambda 함수에 전달할 수 있습니다. 인증 문제 생성 함수를 호출하는 요청에는 AdminInitiateAuth 및 InitiateAuth API 작업의 ClientMetadata 파라미터에 전달된 데이터가 포함되지 않습니다.

인증 챌린지 생성 응답 파라미터

publicChallengeParameters

클라이언트 앱이 사용자에게 표시할 챌린지에 사용하기 위한 하나 이상의 키-값 페어입니다. 사용자에게 문제를 정확히 표시하는 데 필요한 모든 정보를 이 파라미터에 포함해야 합니다.

privateChallengeParameters

이 파라미터는 인증 문제 응답 확인 Lambda 트리거에만 사용됩니다. 챌린지에 대한 사용자의 응답을 확인하는 데 필요한 모든 정보를 이 파라미터에 포함해야 합니다. 즉, `publicChallengeParameters` 파라미터는 사용자에게 표시할 질문을 포함하고 `privateChallengeParameters`는 질문의 유효한 대답을 포함합니다.

challengeMetadata

이 항목이 사용자 지정 챌린지일 경우 사용자 지정 챌린지의 이름입니다.

인증 챌린지 생성 예제

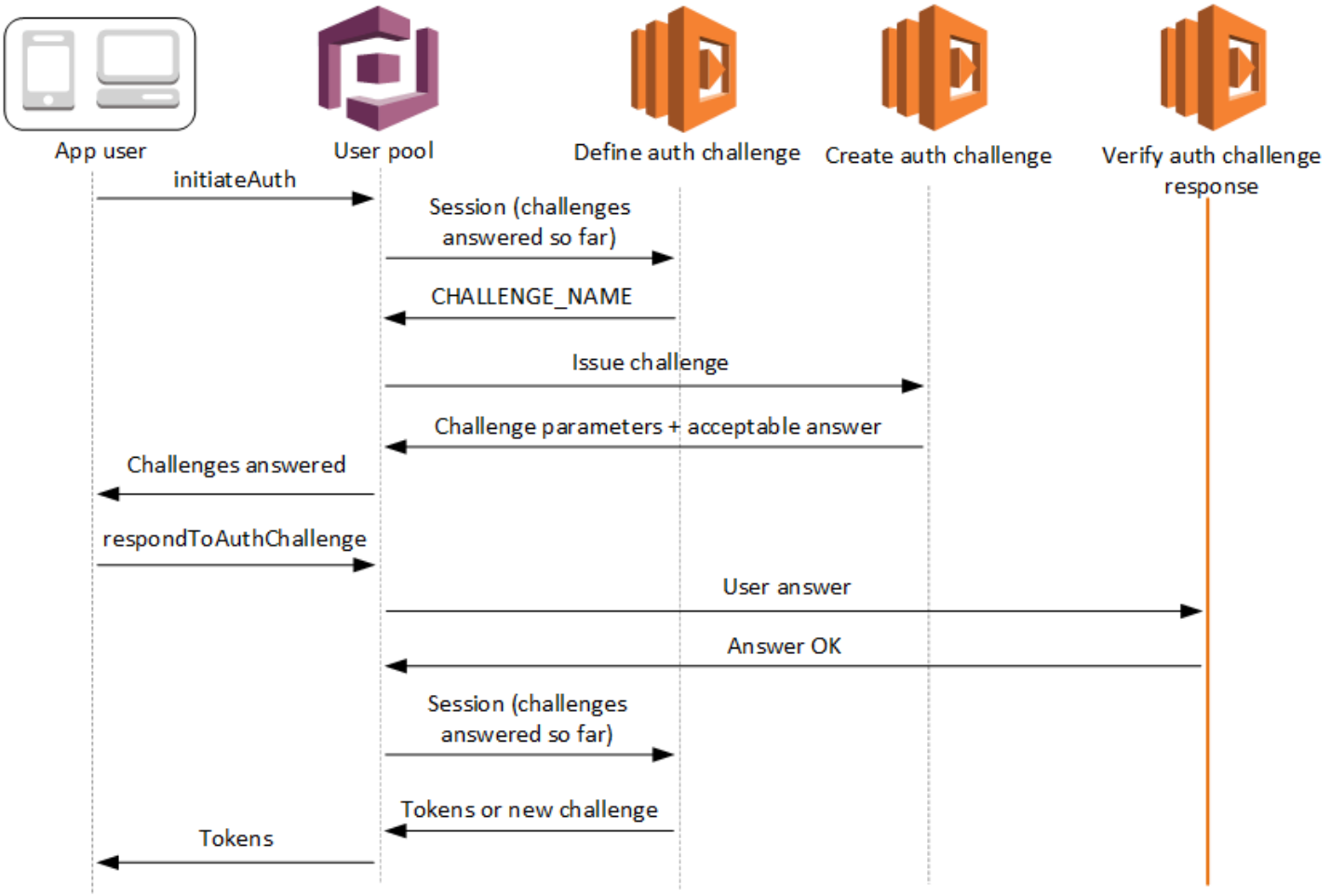
CAPTCHA가 사용자에 대한 챌린지로 생성됩니다. CAPTCHA 이미지 URL이 퍼블릭 챌린지 파라미터에 "`captchaUrl`"로 추가되고 예상되는 답이 프라이빗 챌린지 파라미터에 추가됩니다.

Node.js

```
const handler = async (event) => {
  if (event.request.challengeName !== "CUSTOM_CHALLENGE") {
    return event;
  }

  if (event.request.session.length === 2) {
    event.response.publicChallengeParameters = {};
    event.response.privateChallengeParameters = {};
    event.response.publicChallengeParameters.captchaUrl = "url/123.jpg";
    event.response.privateChallengeParameters.answer = "5";
  }

  if (event.request.session.length === 3) {
```

```
    event.response.publicChallengeParameters = {};
    event.response.privateChallengeParameters = {};
    event.response.publicChallengeParameters.securityQuestion =
      "Who is your favorite team mascot?";
    event.response.privateChallengeParameters.answer = "Peccy";
  }

  return event;
};

export { handler }
```

인증 챌린지 확인 응답 Lambda 트리거



인증 문제 응답 확인

Amazon Cognito는 이 트리거를 호출하여 사용자 지정 인증 문제에 대한 사용자의 응답이 유효한지 여부를 확인합니다. 이것은 사용자 풀 사용자 지정 인증 흐름에 속합니다.

이 트리거에 대한 요청에는 `privateChallengeParameters` 및 `challengeAnswer` 파라미터가 포함됩니다. 인증 문제 생성 Lambda 트리거는 `privateChallengeParameters` 값을 반환하고 사용자의 예상 응답을 포함합니다. `challengeAnswer` 파라미터에는 챌린지에 대한 사용자의 응답이 포함됩니다.

응답에는 `answerCorrect` 속성이 포함됩니다. 사용자가 문제를 성공적으로 완료한 경우 Amazon Cognito는 속성 값을 `true`로 설정합니다. 사용자가 문제를 성공적으로 완료하지 못한 경우 Amazon Cognito는 속성 값을 `false`로 설정합니다.

사용자가 모든 문제에 응답할 때까지 문제 루프가 반복됩니다.

주제

- 인증 챌린지 확인 Lambda 트리거 파라미터
- 인증 챌린지 확인 응답 예제

인증 챌린지 확인 Lambda 트리거 파라미터

Amazon Cognito가 이 Lambda 함수에 전달하는 요청은 아래 파라미터와 Amazon Cognito가 모든 요청에 추가하는 공통 파라미터의 조합입니다.

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
        },
        "privateChallengeParameters": {
            "string": "string",
            . . .
        },
        "challengeAnswer": "string",
        "clientMetadata": {
            "string": "string",
            . . .
        },
        "userNotFound": boolean
    },
    "response": {
        "answerCorrect": boolean
```

```
    }
}
```

인증 챌린지 확인 요청 파라미터

userAttributes

이 파라미터에는 사용자 속성을 나타내는 하나 이상의 이름-값 페어가 포함됩니다.

userNotFound

Amazon Cognito에서 사용자 풀 클라이언트에 대해 `PreventUserExistenceErrors`를 `ENABLED`로 설정한 경우 Amazon Cognito는 이 부울을 채웁니다.

privateChallengeParameters

이 파라미터는 인증 문제 생성 트리거에서 가져옵니다. 사용자가 문제를 통과했는지 여부를 확인하려면 Amazon Cognito에서 이 파라미터를 사용자의 challengeAnswer과 비교합니다.

이 파라미터는 사용자의 문제에 대한 응답을 검증하는 데 필요한 모든 정보를 포함합니다. 이 정보에는 Amazon Cognito가 사용자에게 제시하는 질문(`publicChallengeParameters`) 및 질문에 대한 유효한 대답(`privateChallengeParameters`)이 포함됩니다. 인증 문제 응답 확인 Lambda 트리거만 이 파라미터를 사용합니다.

challengeAnswer

이 파라미터 값은 사용자의 문제에 대한 응답에 있는 대답입니다.

clientMetadata

이 파라미터는 인증 문제 확인 트리거에 대한 Lambda 함수에 사용자 지정 입력으로 제공될 수 있는 하나 이상의 키-값 페어를 포함합니다. 이 데이터를 Lambda 함수에 전달하려면 AdminRespondToAuthChallenge 및 RespondToAuthChallenge API 작업에서 ClientMetadata 파라미터를 사용합니다. Amazon Cognito는 승인 문제 확인 함수에 전달하는 요청에 있는 AdminInitiateAuth 및 InitiateAuth API 작업의 ClientMetadata 파라미터에서 전달된 데이터는 포함하지 않습니다.

인증 챌린지 확인 응답 파라미터

answerCorrect

사용자가 문제를 성공적으로 완료한 경우 Amazon Cognito는 이 파라미터를 `true`로 설정합니다. 사용자가 문제를 성공적으로 완료하지 못한 경우 Amazon Cognito는 이 파라미터를 `false`로 설정합니다.

인증 챌린지 확인 응답 예제

이 예제에서 Lambda 함수가 문제에 대한 사용자의 응답이 예상 응답과 일치하는지 확인합니다. 사용자의 응답과 예상 응답이 일치하는 경우 Amazon Cognito는 `answerCorrect` 파라미터를 `true`로 설정합니다.

Node.js

```
const handler = async (event) => {
  if (
    event.request.privateChallengeParameters.answer ==
    event.request.challengeAnswer
  ) {
    event.response.answerCorrect = true;
  } else {
    event.response.answerCorrect = false;
  }

  return event;
};

export { handler };
```

사전 토큰 생성 Lambda 트리거

토큰 생성 전에 Amazon Cognito가 이 트리거를 간접 호출하기 때문에 사용자 풀 토큰 신청을 사용자 지정할 수 있습니다. 버전 1 또는 `V1_0` 사전 토큰 생성 트리거 이벤트의 기본 기능을 사용하여 ID 토큰을 사용자 지정할 수 있습니다. 고급 보안 기능이 활성화된 사용자 풀에서는 액세스 토큰 사용자 지정을 통해 버전 2 또는 `V2_0` 트리거 이벤트를 생성할 수 있습니다.

Amazon Cognito는 ID 토큰에 쓸 데이터와 함께 `V1_0` 이벤트를 요청으로서 함수에 보냅니다. `V2_0` 이벤트는 Amazon Cognito가 ID 및 액세스 토큰 모두에 쓸 데이터가 포함된 단일 요청입니다. 두 토큰을

모두 사용자 지정하려면 최신 트리거 버전을 사용하도록 함수를 업데이트하고 동일한 응답으로 두 토큰의 데이터를 전송해야 합니다.

이 Lambda 트리거는 Amazon Cognito가 앱에 클레임을 발행하기 전에 ID 및 액세스 토큰의 일부 클레임을 추가, 제거 및 수정할 수 있습니다. 이 기능을 사용하려면 Amazon Cognito 사용자 풀 콘솔에서 Lambda 함수를 연결하거나 AWS Command Line Interface (AWS CLI)를 통해 사용자 풀 `LambdaConfig`를 업데이트합니다.

이벤트 버전

사용자 풀은 다양한 버전의 사전 토큰 생성 트리거 이벤트를 Lambda 함수에 전달할 수 있습니다. `V1_0`트리거는 ID 토큰 수정을 위한 파라미터를 전달합니다. `V2_0`트리거는 다음에 대한 매개변수를 전달합니다.

1. `V1_0`트리거의 기능.
2. 액세스 토큰을 사용자 지정하는 기능.
3. 복잡한 데이터 유형을 ID 및 액세스 토큰 클레임 값에 전달하는 기능:
   - String
   - 숫자
   - 불
   - 문자열, 숫자, 불리언 배열 또는 이들 중 하나의 조합
   - JSON

> **Note**
>
> ID 토큰에서,, 및 를 제외한 복잡한 객체를 클레임 값으로 채울 수 있습니다.
> `phone_number_verified` `email_verified` `updated_at` `address`

사용자 풀은 기본적으로 `V1_0` 이벤트를 전달합니다. `V2_0`이벤트를 전송하도록 사용자 풀을 구성하려면 Amazon Cognito 콘솔에서 트리거를 구성할 때 기본 기능+액세스 토큰 사용자 지정의 트리거 이벤트 버전을 선택하십시오. UpdateUserPool 또는 CreateUserPool API 요청의 LambdaConfig파라미터에 `LambdaVersion` 의 값을 설정할 수도 있습니다. `V2_0`이벤트를 통한 액세스 토큰 사용자 지정에는 추가 비용이 적용됩니다. 자세한 내용은 Amazon Cognito 요금을 참조하세요.

제외된 클레임 및 범위

Amazon Cognito는 액세스 및 자격 증명 토큰에서 추가, 수정 또는 억제할 수 있는 클레임 및 범위를 제한합니다. Lambda 함수가 이러한 클레임 값을 설정하려고 시도하면 Amazon Cognito는 원래 클레임 값이 있는 토큰을 요청에서 발행합니다(요청에 클레임 값이 있는 경우).

공유 클레임

- `acr`
- `amr`
- `at_hash`
- `auth_time`
- `azp`
- `exp`
- `iat`
- `iss`
- `jti`
- `nbf`
- `nonce`
- `origin_jti`
- `sub`
- `token_use`

ID 토큰 클레임

- `identities`
- `aud`
- `cognito:username`

액세스 토큰 클레임

- `username`
- `client_id`
- `scope`

> **Note**
>
> `scopesToAdd` 및 `scopesToSuppress` 응답 값을 사용하여 액세스 토큰의 범위를 변경할 수 있지만, `scope` 클레임을 직접 수정할 수는 없습니다. 사용자 풀의 예약된 범위 `aws.cognito.signin.user.admin` 등 `aws.cognito`로 시작하는 범위는 추가할 수 없습니다.

- `device_key`
- `event_id`
- `version`

다음과 같은 접두사를 사용하여 클레임을 추가하거나 덮어쓸 수는 없지만 클레임을 숨기거나 토큰에 나타나지 않게 할 수는 있습니다.

- `dev:`
- `cognito:`

액세스 토큰에 `aud` 클레임을 추가할 수 있지만 해당 값은 현재 세션의 앱 클라이언트 ID와 일치해야 합니다. `event.callerContext.clientId`의 요청 이벤트에서 클라이언트 ID를 파생할 수 있습니다.

자격 증명 토큰 사용자 지정

사전 토큰 생성 Lambda 트리거를 사용하면 사용자 풀에서 자격 증명(ID) 토큰의 콘텐츠를 사용자 지정할 수 있습니다. ID 토큰은 웹 또는 모바일 앱에 로그인하기 위한 신뢰할 수 있는 ID 소스의 사용자 속성을 제공합니다. ID 토큰에 대한 자세한 내용은 ID 토큰 사용 섹션을 참조하세요.

사전 토큰 생성 Lambda 트리거를 ID 토큰과 함께 사용하는 방법은 다음과 같습니다.

- 사용자가 자격 증명 풀에서 요청하는 IAM 역할을 런타임에서 변경합니다.
- 외부 소스의 사용자 속성을 추가합니다.
- 기존 사용자 속성 값을 추가하거나 바꿉니다.
- 사용자의 승인된 범위와 앱 클라이언트에 부여한 속성에 대한 읽기 액세스 권한으로 인해 앱에 달리 전달될 수 있는 사용자 속성의 공개를 차단합니다.

액세스 토큰 사용자 지정

사전 토큰 생성 Lambda 트리거를 사용하면 사용자 풀에서 액세스 토큰의 콘텐츠를 사용자 지정할 수 있습니다. 액세스 토큰은 Amazon Cognito 토큰 인증 API 작업 및 서드 파티 API와 같은 액세스 보호 리소스에서 정보를 검색할 수 있는 권한을 사용자에게 부여합니다. 클라이언트 자격 증명 부여로 Amazon Cognito에서 machine-to-machine (M2M) 인증을 위한 액세스 토큰을 생성할 수 있지만, M2M 요청은 사전 토큰 생성 트리거 함수를 호출하지 않으며 사용자 지정 액세스 토큰을 발행할 수 없습니다. 액세스 토큰에 대한 자세한 내용은 액세스 토큰 사용 섹션을 참조하세요.

사전 토큰 생성 Lambda 트리거를 액세스 토큰과 함께 사용하는 방법은 다음과 같습니다.

- `scope` 클레임에 OAuth 2.0 범위를 추가하거나 숨깁니다. 예를 들어 범위 `aws.cognito.signin.user.admin`만 할당하는 Amazon Cognito 사용자 풀 API 인증을 통해 생성된 액세스 토큰에 범위를 추가할 수 있습니다.
- 사용자 풀 그룹의 사용자 멤버십을 변경합니다.
- Amazon Cognito 액세스 토큰에 아직 존재하지 않는 클레임을 추가합니다.
- 앱으로 달리 전달될 수 있는 클레임의 공개를 차단합니다.

사용자 풀의 액세스 사용자 지정을 지원하려면 트리거 요청의 업데이트된 버전을 생성하도록 사용자 풀을 구성해야 합니다. 다음 절차에 나오는 대로 사용자 풀을 업데이트합니다.

AWS Management Console

사전 토큰 생성 Lambda 트리거에서 액세스 토큰 사용자 지정의 지원

1. Amazon Cognito 콘솔로 이동한 다음 사용자 풀(User Pools)을 선택합니다.
2. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
3. 아직 활성화하지 않았다면 앱 통합 탭에서 고급 보안 기능을 활성화하십시오.
4. [사용자 풀 속성(User pool properties)] 탭을 선택하고 [Lambda 트리거(Lambda triggers)]를 찾습니다.
5. 사전 토큰 생성 트리거를 추가 또는 편집합니다.
6. Lambda 함수 할당에서 Lambda 함수를 선택합니다.
7. 기본 기능+액세스 토큰 사용자 지정의 트리거 이벤트 버전을 선택하십시오. 이 설정은 Amazon Cognito가 함수에 보내는 요청 파라미터를 업데이트하여 액세스 토큰 사용자 지정을 위한 필드를 포함합니다.

User pools API

사전 토큰 생성 Lambda 트리거에서 액세스 토큰 사용자 지정의 지원

또는 API 요청을 UpdateUserPool생성하십시오. CreateUserPool 기본값으로 설정하지 않으려는 모든 파라미터의 값을 지정해야 합니다. 자세한 정보는 사용자 풀 구성 업데이트을 참조하세요.

요청의 `LambdaVersion` 파라미터에 다음 콘텐츠를 포함합니다. `V2_0`의 `LambdaVersion` 값을 지정하면 사용자 풀이 액세스 토큰 사용자 지정을 위한 파라미터를 추가합니다. 특정 함수 버전을 간접 호출하려면 함수 버전을 `LambdaArn`의 값으로 포함하는 Lambda 함수 ARN을 사용하세요.

```
"PreTokenGenerationConfig": {
   "LambdaArn": "arn:aws:lambda:us-west-2:123456789012:function:MyFunction",
   "LambdaVersion": "V2_0"
},
```

주제

- 사전 토큰 생성 Lambda 트리거 소스
- 사전 토큰 생성 Lambda 트리거 파라미터
- 사전 토큰 트리거 이벤트 버전 2 예시: 클레임, 범위 및 그룹 추가 및 억제
- 사전 토큰 생성 이벤트 버전 2 예제: 복잡한 객체가 포함된 클레임 추가
- 사전 토큰 생성 이벤트 버전 1 예시: 새 클레임 추가 및 기존 클레임 억제
- 사전 토큰 생성 이벤트 버전 1 예시: 사용자의 그룹 멤버십 수정

사전 토큰 생성 Lambda 트리거 소스

triggerSource 값	Event
`TokenGeneration_HostedAuth`	인증하는 동안 Amazon Cognito 호스트된 UI 로그인 페이지에서 호출됩니다.
`TokenGeneration_Authentication`	사용자 인증 흐름이 완료된 이후 호출됩니다.
`TokenGeneration_NewPasswordChallenge`	사용자가 관리자에 의해 생성된 후 호출됩니다. 이 흐름은 사용자가 임시 비밀번호를 변경해야 할 때 호출됩니다.

triggerSource 값	Event
TokenGeneration_AuthenticateDevice	사용자 디바이스 인증 마지막에 호출됩니다.
TokenGeneration_RefreshTokens	사용자가 자격 증명 및 액세스 토큰을 새로 고치려 할 때 호출됩니다.

사전 토큰 생성 Lambda 트리거 파라미터

Amazon Cognito가 이 Lambda 함수에 전달하는 요청은 아래 파라미터와 Amazon Cognito가 모든 요청에 추가하는 공통 파라미터의 조합입니다. 사전 토큰 생성 Lambda 트리거를 사용자 풀에 추가할 때 트리거 버전을 선택할 수 있습니다. 이 버전은 Amazon Cognito가 액세스 토큰 사용자 지정을 위한 추가 파라미터와 함께 Lambda 함수에 요청을 전달할지 여부를 결정합니다.

Version 1

버전 1 토큰은 ID 토큰에 그룹 멤버십, IAM 역할 및 새 클레임을 설정할 수 있습니다.

```
{
    "request": {
        "userAttributes": {"string": "string"},
        "groupConfiguration": {
                "groupsToOverride": [
                    "string",
                    "string"
                ],
                "iamRolesToOverride": [
                    "string",
                    "string"
                ],
                "preferredRole": "string"
        },
        "clientMetadata": {"string": "string"}
    },
    "response": {
        "claimsOverrideDetails": {
            "claimsToAddOrOverride": {"string": "string"},
            "claimsToSuppress": [
                "string",
                "string"
```

```
            ],
            "groupOverrideDetails": {
                "groupsToOverride": [
                    "string",
                    "string"
                ],
                "iamRolesToOverride": [
                    "string",
                    "string"
                ],
                "preferredRole": "string"
            }
        }
    }
}
```

Version 2

버전 2 요청 이벤트는 액세스 토큰을 사용자 지정하는 필드를 추가합니다. 또한 응답 개체의 복잡한 `claimsToOverride` 데이터 유형에 대한 지원을 추가합니다. Lambda 함수는 다음 유형의 데이터를 다음 값으로 반환할 수 있습니다. `claimsToOverride`

- String
- 숫자
- 불
- 문자열, 숫자, 불리언 배열 또는 이들 중 하나의 조합
- JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string"
        },
        "scopes": ["string", "string"],
        "groupConfiguration": {
            "groupsToOverride": ["string", "string"],
            "iamRolesToOverride": ["string", "string"],
            "preferredRole": "string"
        },
        "clientMetadata": {
```

```
            "string": "string"
        }
    },
    "response": {
        "claimsAndScopeOverrideDetails": {
            "idTokenGeneration": {
                "claimsToAddOrOverride": {
                    "string": [accepted datatype]
                },
                "claimsToSuppress": ["string", "string"]
            },
            "accessTokenGeneration": {
                "claimsToAddOrOverride": {
                    "string": [accepted datatype]
                },
                "claimsToSuppress": ["string", "string"],
                "scopesToAdd": ["string", "string"],
                "scopesToSuppress": ["string", "string"]
            },
            "groupOverrideDetails": {
                "groupsToOverride": ["string", "string"],
                "iamRolesToOverride": ["string", "string"],
                "preferredRole": "string"
            }
        }
    }
}
```

사전 토큰 생성 요청 파라미터

명칭	설명	최소 트리거 이벤트 버전
userAttributes	사용자 풀에 있는 사용자의 사용자 프로필 속성입니다.	1
groupConfiguration	현재 그룹 구성을 포함하는 입력 객체입니다. 이 객체에는 `groupsToOverride` , `iamRolesToOverride` 및 `preferredRole` 이 포함됩니다.	1
groupsToOverride	사용자가 속한 사용자 풀 그룹입니다.	1

명칭	설명	최소 트리거 이벤트 버전
iamRolesTo오버라이드	사용자 풀 그룹을 AWS Identity and Access Management (IAM) 역할과 연결할 수 있습니다. 이 요소는 사용자가 속한 그룹의 모든 IAM 역할 목록입니다.	1
preferredRole	사용자 풀 그룹의 우선 순위를 설정할 수 있습니다. 이 요소에는 `groupsToOverride` 요소에서 우선 순위가 가장 높은 그룹의 IAM 역할 이름이 포함됩니다.	1
clientMetadata	사용자가 지정하고 사전 토큰 생성 트리거에 Lambda 함수에 대한 사용자 지정 입력으로 제공할 수 있는 하나 이상의 키-값 페어입니다. 이 데이터를 Lambda 함수로 전달하려면 및 API 작업에서 AdminRespondToAuthChallenge파라미터를 사용하십시오 ClientMetadata. RespondToAuthChallenge Amazon Cognito는 사전 토큰 생성 함수로 전달하는 요청에 `ClientMetadata` 파라미터 입력 AdminInitiateAuth및 InitiateAuthAPI 작업의 데이터를 포함하지 않습니다.	1
범위	사용자의 OAuth 2.0 범위입니다. 액세스 토큰에 표시되는 범위는 사용자가 요청하고 앱 클라이언트에 발급을 승인한 사용자 풀 표준 및 사용자 지정 범위입니다.	2

사전 토큰 생성 응답 파라미터

명칭	설명	최소 트리거 이벤트 버전
claimsOverrideDetails	`V1_0` 트리거 이벤트의 모든 요소를 담는 컨테이너입니다.	1

명칭	설명	최소 트리거 이벤트 버전
claimsAndScopeOverrideDetails	`V2_0` 트리거 이벤트의 모든 요소를 담는 컨테이너입니다.	2
idTokenGeneration	사용자 ID 토큰에서 재정의, 추가 또는 차단하려는 클레임입니다. 이 상위 ID 토큰 사용자 지정 값은 버전 2 이벤트에만 나타나지만 하위 요소는 버전 1 이벤트에 나타납니다.	2
accessTokenGeneration	사용자 액세스 토큰에서 재정의, 추가 또는 차단하려는 클레임 및 범위입니다. 이 상위 액세스 토큰 사용자 지정 값은 버전 2 이벤트에서만 나타납니다.	2
claimsToAddOrOverride	추가 또는 수정하려는 하나 이상의 클레임 및 해당 값이 표시된 맵입니다. 그룹 관련 클레임의 경우 대신 `groupOverrideDetails` 를 사용합니다. 버전 2 이벤트에서 이 요소는 `accessTokenGeneration` 및 `idTokenGeneration` 에 모두 나타납니다.	1*
claimsToSuppress	Amazon Cognito가 차단하길 원하는 클레임의 목록입니다. 함수가 신청 값을 억제하고 대체하는 경우 Amazon Cognito는 신청을 억제합니다. 버전 2 이벤트에서 이 요소는 `accessTokenGeneration` 및 `idTokenGeneration` 에 모두 나타납니다.	1

명칭	설명	최소 트리거 이벤트 버전
groupOverrideDetails	현재 그룹 구성을 포함하는 입력 객체입니다. 이 객체에는 `groupsToOverride` , `iamRolesToOverride` 및 `preferredRole` 이 포함됩니다. 함수가 `groupOverrideDetails` 객체를 사용자가 제공하는 객체로 바꿉니다. 응답에서 비어 있는 객체 또는 Null 객체를 제공하면 Amazon Cognito가 그룹을 억제합니다. 기존 그룹 구성을 동일하게 유지하려면 요청의 `groupConfiguration` 객체 값을 응답의 `groupOverrideDetails` 객체로 복사합니다. 그런 다음 해당 객체를 서비스로 다시 전달합니다. Amazon Cognito ID 및 액세스 토큰 둘 다에 `cognito:groups` 클레임이 포함되어 있습니다. `groupOverrideDetails` 객체는 액세스 토큰과 ID 토큰의 `cognito:groups` 클레임을 대체합니다.	1
scopesToAdd	사용자의 액세스 토큰에서 `scope` 클레임에 추가할 OAuth 2.0 범위 목록입니다. 하나 이상의 공백 문자가 포함된 범위 값은 추가할 수 없습니다.	2
scopesToSuppress	사용자의 액세스 토큰에서 `scope` 클레임을 제거할 OAuth 2.0 범위 목록입니다.	2

* 버전 1 이벤트에 대한 응답 객체는 문자열을 반환할 수 있습니다. 버전 2 이벤트에 대한 응답 객체는 복잡한 객체를 반환할 수 있습니다.

사전 토큰 트리거 이벤트 버전 2 예시: 클레임, 범위 및 그룹 추가 및 억제

이 예시에서는 사용자 토큰을 다음과 같이 수정합니다.

1. ID 토큰에서 `family_name`을 `Doe`와 동일하게 설정합니다.
2. ID 토큰에 `email` 및 `phone_number` 클레임이 나타나지 않도록 합니다.

3. ID 토큰 `cognito:roles` 클레임을 `"arn:aws:iam::123456789012:role\/sns_callerA","arn:aws:iam::123456789012:role\/sns_callerC","arn:aws:iam::123456789012:role\/sns_callerB"`로 설정합니다.
4. ID 토큰 `cognito:preferred_role` 클레임을 `arn:aws:iam::123456789012:role/sns_caller`로 설정합니다.
5. 액세스 토큰에 범위 `openid`, `email`, `solar-system-data/asteroids.add`를 추가합니다.
6. 액세스 토큰에서 범위 `phone_number` 및 `aws.cognito.signin.user.admin`을 억제합니다. `phone_number`를 제거하면 `userInfo`에서 사용자 전화 번호를 검색할 수 없습니다. `aws.cognito.signin.user.admin`를 제거하면 Amazon Cognito 사용자 풀 API로 자신의 프로필을 읽고 수정하려는 사용자의 API 요청이 방지됩니다.

> **Note**
>
> 액세스 토큰의 나머지 범위에 `openid` 및 하나 이상의 표준 범위가 포함된 경우에만 범위에서 `phone_number`를 제거하면 사용자 전화 번호를 검색할 수 없습니다. 자세한 정보는 범위에 대한 정보을 참조하세요.

7. ID 및 액세스 토큰 `cognito:groups` 클레임을 `"new-group-A","new-group-B","new-group-C"`로 설정합니다.

JavaScript

```
export const handler = function(event, context) {
  event.response = {
    "claimsAndScopeOverrideDetails": {
      "idTokenGeneration": {
        "claimsToAddOrOverride": {
          "family_name": "Doe"
    },
        "claimsToSuppress": [
          "email",
          "phone_number"
    ]
      },
      "accessTokenGeneration": {
        "scopesToAdd": [
          "openid",
          "email",
          "solar-system-data/asteroids.add"
```

```
        ],
        "scopesToSuppress": [
          "phone_number",
          "aws.cognito.signin.user.admin"
        ]
      },
      "groupOverrideDetails": {
        "groupsToOverride": [
    "new-group-A",
    "new-group-B",
    "new-group-C"
  ],
        "iamRolesToOverride": [
          "arn:aws:iam::123456789012:role/new_roleA",
          "arn:aws:iam::123456789012:role/new_roleB",
          "arn:aws:iam::123456789012:role/new_roleC"
        ],
        "preferredRole": "arn:aws:iam::123456789012:role/new_role",
      }
    }
  };
  // Return to Amazon Cognito
  context.done(null, event);
};
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그런 다음 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. Lambda 콘솔에서 해당 Lambda 트리거와 관련 있는 데이터로 테스트 이벤트를 설정할 수 있습니다. 다음은 이 코드 샘플의 테스트 이벤트입니다.

JSON

```
{
    "version": "2",
    "triggerSource": "TokenGeneration_Authentication",
    "region": "us-east-1",
    "userPoolId": "us-east-1_EXAMPLE",
    "userName": "JaneDoe",
    "callerContext": {
        "awsSdkVersion": "aws-sdk-unknown-unknown",
        "clientId": "1example23456789"
    },
    "request": {
```

```
        "userAttributes": {
            "sub": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "cognito:user_status": "CONFIRMED",
            "email_verified": "true",
            "phone_number_verified": "true",
            "phone_number": "+12065551212",
            "family_name": "Zoe",
            "email": "Jane.Doe@example.com"
        },
        "groupConfiguration": {
            "groupsToOverride": ["group-1", "group-2", "group-3"],
            "iamRolesToOverride": ["arn:aws:iam::123456789012:role/sns_caller1",
 "arn:aws:iam::123456789012:role/sns_caller2", "arn:aws:iam::123456789012:role/
sns_caller3"],
            "preferredRole": ["arn:aws:iam::123456789012:role/sns_caller"]
        },
        "scopes": [
            "aws.cognito.signin.user.admin", "openid", "email", "phone"
        ]
    },
    "response": {
        "claimsAndScopeOverrideDetails": []
    }
}
```

사전 토큰 생성 이벤트 버전 2 예제: 복잡한 객체가 포함된 클레임 추가

이 예시에서는 사용자 토큰을 다음과 같이 수정합니다.

1. 숫자, 문자열, 부울 및 JSON 유형의 클레임을 ID 토큰에 추가합니다. 버전 2 트리거 이벤트가 ID 토큰에 제공하는 유일한 변경 사항입니다.
2. 액세스 토큰에 숫자, 문자열, 부울 및 JSON 유형의 클레임을 추가합니다.
3. 액세스 토큰에 세 개의 범위를 추가합니다.
4. ID 및 액세스 토큰에서 `email` 및 `sub` 클레임을 억제합니다.
5. 액세스 토큰의 `aws.cognito.signin.user.admin` 범위를 숨깁니다.

JavaScript

```
export const handler = function(event, context) {
```

```
    var scopes = ["MyAPI.read", "MyAPI.write", "MyAPI.admin"]
    var claims = {}
    claims["aud"]= event.callerContext.clientId;
    claims["booleanTest"] = false;
    claims["longTest"] = 9223372036854775807;
    claims["exponentTest"] = 1.7976931348623157E308;
    claims["ArrayTest"] = ["test", 9223372036854775807, 1.7976931348623157E308,
true];
    claims["longStringTest"] = "\{\
        \"first_json_block\": \{\
            \"key_A\": \"value_A\",\
            \"key_B\": \"value_B\"\
        \},\
        \"second_json_block\": \{\
            \"key_C\": \{\
                \"subkey_D\": [\
                    \"value_D\",\
                    \"value_E\"\
                ],\
                \"subkey_F\": \"value_F\"\
            \},\
            \"key_G\": \"value_G\"\
        \}\
    \}";
    claims["jsonTest"] = {
     "first_json_block": {
      "key_A": "value_A",
      "key_B": "value_B"
     },
     "second_json_block": {
      "key_C": {
       "subkey_D": [
        "value_D",
        "value_E"
       ],
       "subkey_F": "value_F"
      },
      "key_G": "value_G"
     }
    };
    event.response = {
        "claimsAndScopeOverrideDetails": {
            "idTokenGeneration": {
                "claimsToAddOrOverride": claims,
```

```
                "claimsToSuppress": ["email","sub"]
            },
            "accessTokenGeneration": {
                "claimsToAddOrOverride": claims,
                "claimsToSuppress": ["email","sub"],
                "scopesToAdd": scopes,
                "scopesToSuppress": ["aws.cognito.signin.user.admin"]
            }
        }
    };
    console.info("EVENT response\n" + JSON.stringify(event, (_, v) => typeof v === 
 'bigint' ? v.toString() : v, 2))
    console.info("EVENT response size\n" + JSON.stringify(event, (_, v) => typeof v 
 === 'bigint' ? v.toString() : v).length)
    // Return to Amazon Cognito
    context.done(null, event);
};
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그런 다음 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. Lambda 콘솔에서 해당 Lambda 트리거와 관련 있는 데이터로 테스트 이벤트를 설정할 수 있습니다. 다음은 이 코드 샘플의 테스트 이벤트입니다.

JSON

```
{
    "version": "2",
    "triggerSource": "TokenGeneration_HostedAuth",
    "region": "us-west-2",
    "userPoolId": "us-west-2_EXAMPLE",
    "userName": "JaneDoe",
    "callerContext": {
        "awsSdkVersion": "aws-sdk-unknown-unknown",
        "clientId": "1example23456789"
    },
    "request": {
        "userAttributes": {
            "sub": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "cognito:user_status": "CONFIRMED"
            "email_verified": "true",
            "phone_number_verified": "true",
            "phone_number": "+12065551212",
            "email": "Jane.Doe@example.com"
```

```
        },
        "groupConfiguration": {
            "groupsToOverride": ["group-1", "group-2", "group-3"],
            "iamRolesToOverride": ["arn:aws:iam::123456789012:role/sns_caller1"],
            "preferredRole": ["arn:aws:iam::123456789012:role/sns_caller1"]
        },
        "scopes": [
            "aws.cognito.signin.user.admin",
            "phone",
            "openid",
            "profile",
            "email"
        ]
    },
    "response": {
        "claimsAndScopeOverrideDetails": []
    }
}
```

사전 토큰 생성 이벤트 버전 1 예시: 새 클레임 추가 및 기존 클레임 억제

이 예시에서는 사전 토큰 생성 Lambda 함수와 버전 1 트리거 이벤트를 사용하여 새 클레임을 추가하고 기존 클레임을 차단합니다.

Node.js

```
const handler = async (event) => {
  event.response = {
    claimsOverrideDetails: {
      claimsToAddOrOverride: {
        my_first_attribute: "first_value",
        my_second_attribute: "second_value",
      },
      claimsToSuppress: ["email"],
    },
  };

  return event;
};

export { handler };
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그런 다음 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. Lambda 콘솔에서 해당 Lambda 트리거와 관련 있는 데이터로 테스트 이벤트를 설정할 수 있습니다. 다음은 이 코드 샘플에 대한 테스트 이벤트입니다. 이 코드 예에서는 요청 파라미터를 처리하지 않기 때문에 빈 요청에 테스트 이벤트를 사용할 수 있습니다. 공통 요청 파라미터에 대한 자세한 내용은 사용자 풀 Lambda 트리거 이벤트 섹션을 참조하세요.

JSON

```
{
  "request": {},
  "response": {}
}
```

사전 토큰 생성 이벤트 버전 1 예시: 사용자의 그룹 멤버십 수정

이 예시에서는 사전 토큰 생성 Lambda 함수와 버전 1 트리거 이벤트를 사용하여 사용자의 그룹 멤버십을 수정합니다.

Node.js

```
const handler = async (event) => {
  event.response = {
    claimsOverrideDetails: {
      groupOverrideDetails: {
        groupsToOverride: ["group-A", "group-B", "group-C"],
        iamRolesToOverride: [
          "arn:aws:iam::XXXXXXXXXXX:role/sns_callerA",
          "arn:aws:iam::XXXXXXXXX:role/sns_callerB",
          "arn:aws:iam::XXXXXXXXXX:role/sns_callerC",
        ],
        preferredRole: "arn:aws:iam::XXXXXXXXXXX:role/sns_caller",
      },
    },
  };

  return event;
};

export { handler };
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그런 다음 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. Lambda 콘솔에서 해당 Lambda 트리거와 관련 있는 데이터로 테스트 이벤트를 설정할 수 있습니다. 다음은 이 코드 샘플의 테스트 이벤트입니다.

JSON

```
{
  "request": {},
  "response": {}
}
```

사용자 마이그레이션 Lambda 트리거

사용자가 암호로 로그인할 때 사용자 풀에 존재하지 않는 경우 또는 암호 찾기 흐름에 있는 경우 Amazon Cognito에서 이 트리거를 호출합니다. Lambda 함수가 성공적으로 반환된 이후 Amazon Cognito는 사용자를 사용자 풀에 추가합니다. 사용자 마이그레이션 Lambda 트리거를 사용하는 인증 흐름에 대한 자세한 내용은 사용자 마이그레이션 Lambda 트리거를 사용하여 사용자 풀로 사용자 가져오기를 참조하세요.

로그인 시 또는 암호 찾기 흐름 동안 기존 사용자 디렉터리에 있는 사용자를 Amazon Cognito 사용자 풀로 마이그레이션하려면 이 Lambda 트리거를 사용합니다.

주제

- 사용자 마이그레이션 Lambda 트리거 소스
- 사용자 마이그레이션 Lambda 트리거 파라미터
- 예제: 기존 암호가 있는 사용자 마이그레이션

사용자 마이그레이션 Lambda 트리거 소스

triggerSource 값	이벤트
`UserMigration_Authentication`	로그인 시 사용자 마이그레이션입니다.
`UserMigration_ForgotPassword`	암호 찾기 흐름 도중 사용자 마이그레이션.

사용자 마이그레이션 Lambda 트리거 파라미터

Amazon Cognito가 이 Lambda 함수에 전달하는 요청은 아래 파라미터와 Amazon Cognito가 모든 요청에 추가하는 공통 파라미터의 조합입니다.

JSON

```
{
    "userName": "string",
    "request": {
        "password": "string",
        "validationData": {
            "string": "string",
            . . .
        },
        "clientMetadata": {
            "string": "string",
       . . .
        }
    },
    "response": {
        "userAttributes": {
            "string": "string",
            . . .
        },
        "finalUserStatus": "string",
        "messageAction": "string",
        "desiredDeliveryMediums": [ "string", . . .],
        "forceAliasCreation": boolean,
        "enableSMSMFA": boolean
    }
}
```

사용자 마이그레이션 요청 파라미터

사용자 이름

로그인 시 사용자가 입력하는 사용자 이름입니다.

password

로그인 시 사용자가 입력하는 암호입니다. Amazon Cognito는 암호 찾기 흐름에 의해 시작된 요청에서 이 값을 보내지 않습니다.

validationData

사용자 로그인 요청에 검증 데이터를 포함하는 하나 이상의 키-값 페어입니다. 이 데이터를 Lambda 함수에 전달하려면 InitiateAuth 및 AdminInitiateAuth API 작업에서 ClientMetadata 파라미터를 사용합니다.

clientMetadata

사용자 마이그레이션 트리거를 위한 Lambda 함수에 사용자 지정 입력으로 제공할 수 있는 하나 이상의 키-값 페어입니다. 이 데이터를 Lambda 함수에 전달하려면 AdminRespondToAuthChallenge 및 ForgotPassword API 작업에서 ClientMetadata 파라미터를 사용합니다.

사용자 마이그레이션 응답 파라미터

userAttributes

이 필드는 필수 항목입니다.

이 필드는 Amazon Cognito가 사용자 풀의 사용자 프로필에 저장하고 사용자 속성으로 사용하는 이름-값 페어를 하나 이상 포함해야 합니다. 표준 속성과 사용자 지정 속성을 모두 포함할 수 있습니다. 표준 속성과 구별하기 위해 사용자 지정 속성에 `custom:` 접두사가 필요합니다. 자세한 내용은 사용자 지정 속성을 참조하세요.

> **Note**
>
> 암호 찾기 흐름에서 사용자가 암호를 재설정하려면 확인된 이메일 주소 또는 확인된 전화 번호를 입력해야 합니다. Amazon Cognito는 암호 재설정 코드가 포함된 메시지를 사용자 속성에 있는 이메일 주소 또는 전화 번호로 보냅니다.

Attributes	요구 사항
사용자 풀을 만들 때 필수로 표시한 모든 속성	마이그레이션 중에 필수 속성이 누락된 경우 Amazon Cognito는 기본값을 사용합니다.
`username`	로그인을 위한 사용자 이름 외에 별칭 속성을 사용하여 사용자 풀을 구성하고, 사용자가 유효한 별칭 값을 사용자 이름으로 입력한 경우 필요합니다. 이 별칭 값은 이메일 주소, 선호하는 사용자 이름 또는 전화 번호일 수 있습니다.

Attributes	요구 사항
	요청 및 사용자 풀이 별칭 요구 사항을 충족하는 경우 함수로부터의 응답은 해당 함수가 수신한 `username` 파라미터를 별칭 속성에 할당해야 하며 응답은 사용자의 값을 `username` 속성에 할당해야 합니다. 사용자 풀이 수신된 `username`을 별칭에 매핑하는 데 필요한 조건을 충족하지 않는 경우 응답의 `username` 파라미터는 요청과 정확하게 일치하거나 생략되어야 합니다. **Note** `username`은 사용자 풀에서 고유해야 합니다.

finalUserStatus

사용자가 이전 암호로 로그인할 수 있도록 사용자를 자동 확인하기 위해 이 파라미터를 `CONFIRMED`로 설정할 수 있습니다. 사용자를 `CONFIRMED`로 설정하면 해당 사용자는 사전에 추가 작업을 수행하지 않고도 로그인할 수 있습니다. 이 속성을 `CONFIRMED`로 설정하지 않은 경우 해당 속성이 `RESET_REQUIRED`로 설정됩니다.

`RESET_REQUIRED`의 `finalUserStatus`는 사용자가 로그인 시 마이그레이션 직후에 자신의 암호를 변경해야 하며 클라이언트 앱은 인증 흐름 동안 `PasswordResetRequiredException`을 처리해야 함을 의미합니다.

Note

Amazon Cognito는 Lambda 트리거를 사용하여 마이그레이션하는 동안 사용자 풀에 대해 구성된 암호 강도 정책을 적용하지 않습니다. 암호가 구성된 암호 정책에 부합하지 않는 경우에도 Amazon Cognito는 여전히 암호를 수락하여 사용자를 계속 마이그레이션할 수 있습니다. 암호 강도 정책을 적용하고 정책에 부합하지 않는 암호를 거부하려면 코드에서 암호 강도를 검증합니다. 그런 다음 암호가 정책에 부합하지 않는 경우 finalUserStatus를 `RESET_REQUIRED`로 설정합니다.

messageAction

Amazon Cognito가 일반적으로 새 사용자에게 보내는 시작 메시지 보내기를 거부하려면 이 파라미터를 SUPPRESS로 설정합니다. 함수가 이 파라미터를 반환하지 않으면 Amazon Cognito가 시작 메시지를 보냅니다.

desiredDeliveryMediums

시작 메시지를 이메일로 보내려면 이 파라미터를 EMAIL로 설정하고, 시작 메시지를 SMS로 보내려면 이 파라미터를 SMS로 설정합니다. 함수가 이 파라미터를 반환하지 않으면 Amazon Cognito가 시작 메시지를 SMS로 보냅니다.

forceAliasCreation

이 파라미터가 TRUE로 설정되고 UserAttributes 파라미터의 전화 번호 또는 이메일 주소가 다른 사용자의 별칭으로 이미 존재하는 경우 API 호출이 이전 사용자의 별칭을 새로 생성된 사용자로 마이그레이션합니다. 이전 사용자는 더 이상 해당 별칭을 사용하여 로그인할 수 없습니다.

이 파라미터를 FALSE로 설정하고 별칭이 존재하는 경우 Amazon Cognito가 사용자를 마이그레이션하지 않고 클라이언트 앱에 오류를 반환합니다.

이 파라미터를 반환하지 않는 경우 Amazon Cognito는 해당 값이 'false'라고 가정합니다.

enableSMSMFA

이 파라미터를 true로 설정하는 경우 마이그레이션된 사용자가 로그인하려면 SMS 문자 메시지 다중 인증(MFA)을 완료해야 합니다. 사용자 풀에 MFA가 활성화되어 있어야 합니다. 요청 파라미터의 사용자 속성에 전화 번호가 포함되어야 합니다. 그렇지 않으면 해당 사용자의 마이그레이션이 실패합니다.

예제: 기존 암호가 있는 사용자 마이그레이션

이 Lambda 함수 예는 기존 암호를 가진 사용자를 마이그레이션하고 Amazon Cognito의 환영 인사 메시지를 억제합니다.

Node.js

```
const validUsers = {
  belladonna: { password: "Test123", emailAddress: "bella@example.com" },
};

// Replace this mock with a call to a real authentication service.
const authenticateUser = (username, password) => {
```

```
  if (validUsers[username] && validUsers[username].password === password) {
    return validUsers[username];
  } else {
    return null;
  }
};

const lookupUser = (username) => {
  const user = validUsers[username];

  if (user) {
    return { emailAddress: user.emailAddress };
  } else {
    return null;
  }
};

const handler = async (event) => {
  if (event.triggerSource == "UserMigration_Authentication") {
    // Authenticate the user with your existing user directory service
    const user = authenticateUser(event.userName, event.request.password);
    if (user) {
      event.response.userAttributes = {
        email: user.emailAddress,
        email_verified: "true",
      };
      event.response.finalUserStatus = "CONFIRMED";
      event.response.messageAction = "SUPPRESS";
    }
  } else if (event.triggerSource == "UserMigration_ForgotPassword") {
    // Look up the user in your existing user directory service
    const user = lookupUser(event.userName);
    if (user) {
      event.response.userAttributes = {
        email: user.emailAddress,
        // Required to enable password-reset code to be sent to user
        email_verified: "true",
      };
      event.response.messageAction = "SUPPRESS";
    }
  }

  return event;
};
```

```
export { handler };
```

사용자 정의 메시지 Lambda 트리거

Amazon Cognito는 이메일이나 전화 확인 메시지 또는 멀티 팩터 인증(MFA) 코드를 보내기 전에 이 트리거를 호출합니다. 사용자 지정 메시지 트리거를 사용하여 메시지를 동적으로 사용자 지정할 수 있습니다. 원래 Amazon Cognito 콘솔의 메시지 사용자 지정(Message customizations) 탭에서 정적 사용자 지정 메시지를 편집할 수 있습니다.

요청에는 `codeParameter`가 포함됩니다. 이는 Amazon Cognito가 사용자에게 전달하는 코드의 자리 표시자 역할을 하는 문자열입니다. `codeParameter` 문자열을 확인 코드가 나타날 메시지 본문에 삽입합니다. Amazon Cognito에서 응답을 수신하면 Amazon Cognito에서 `codeParameter` 문자열을 실제 확인 코드로 바꿉니다.

> **Note**
>
> `CustomMessage_AdminCreateUser` 트리거 소스가 있는 사용자 지정 메시지 Lambda 함수는 사용자 이름 및 확인 코드를 반환합니다. 관리자가 생성한 사용자는 사용자 이름과 코드를 모두 수신해야 하므로 함수로부터의 응답에는 `request.usernameParameter`와 `request.codeParameter`가 둘 다 포함되어야 합니다.

주제

- 사용자 정의 메시지 Lambda 트리거 소스
- 사용자 정의 메시지 Lambda 트리거 파라미터
- 가입용 사용자 지정 메시지의 예
- 관리자로 사용자 생성용 사용자 정의 메시지 예제

사용자 정의 메시지 Lambda 트리거 소스

triggerSource 값	Event
`CustomMessage_SignUp`	사용자 지정 메시지 – 가입 후 확인 코드 전송.

triggerSource 값	Event
CustomMessage_AdminCreateUser	사용자 지정 메시지 – 새 사용자에게 임시 암호 전송.
CustomMessage_ResendCode	사용자 지정 메시지 – 기존 사용자에게 확인 코드 재전송.
CustomMessage_ForgotPassword	사용자 지정 메시지 – 암호 분실 요청을 위한 확인 코드 전송.
CustomMessage_UpdateUserAttribute	사용자 지정 메시지 – 사용자의 이메일 또는 전화 번호가 변경되면 이 트리거가 사용자에게 자동으로 확인 코드를 전송합니다. 다른 속성에는 사용할 수 없습니다.
CustomMessage_VerifyUserAttribute	사용자 지정 메시지 – 사용자가 새 이메일 또는 전화 번호에 대해 수동으로 요청하면 이 트리거가 사용자에게 확인 코드를 전송합니다.
CustomMessage_Authentication	사용자 지정 메시지 – 인증하는 동안 MFA 코드 전송.

사용자 정의 메시지 Lambda 트리거 파라미터

Amazon Cognito가 이 Lambda 함수에 전달하는 요청은 아래 파라미터와 Amazon Cognito가 모든 요청에 추가하는 공통 파라미터의 조합입니다.

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
        }
        "codeParameter": "####",
        "usernameParameter": "string",
        "clientMetadata": {
            "string": "string",
```

```
            . . .
        }
    },
    "response": {
        "smsMessage": "string",
        "emailMessage": "string",
        "emailSubject": "string"
    }
}
```

사용자 정의 메시지 요청 파라미터

userAttributes

사용자 속성을 나타내는 하나 이상의 이름-값 페어입니다.

codeParameter

사용자 지정 메시지에서 확인 코드의 자리 표시자로 사용할 문자열입니다.

usernameParameter

사용자 이름. Amazon Cognito는 관리자가 생성한 사용자의 요청에 이 파라미터를 포함합니다.

clientMetadata

사용자 지정 메시지 트리거에 지정하는 Lambda 함수에 사용자 지정 입력으로 제공할 수 있는 하나 이상의 키-값 쌍입니다. 사용자 지정 메시지 함수를 호출하는 요청에는 ClientMetadata 매개변수로 전달된 AdminInitiateAuth데이터와 InitiateAuthAPI 작업이 포함되지 않습니다. 이 데이터를 Lambda 함수로 전달하려면 다음 API 작업에서 파라미터를 사용할 ClientMetadata 수 있습니다.

- AdminResetUserPassword
- AdminRespondToAuthChallenge
- AdminUpdateUserAttributes
- ForgotPassword
- GetUserAttributeVerificationCode
- ResendConfirmationCode
- SignUp
- UpdateUserAttributes

사용자 정의 메시지 응답 파라미터

응답에서는 사용자에게 보낼 메시지에 사용할 사용자 지정 텍스트를 지정합니다. Amazon Cognito가 이러한 파라미터에 적용하는 문자열 제약 조건은 을 참조하십시오. MessageTemplateType

smsMessage

사용자에게 보낼 사용자 지정 SMS 메시지입니다. 요청에서 수신한 `codeParameter` 값을 포함해야 합니다.

emailMessage

사용자에게 보낼 사용자 지정 이메일 메시지입니다. `emailMessage` 파라미터에서 HTML 서식을 사용할 수 있습니다. 요청에서 수신한 `codeParameter` 값을 변수 `{####}`으로 포함해야 합니다. Amazon Cognito는 사용자 풀의 `EmailSendingAccount` 속성이 `DEVELOPER`인 경우에만 `emailMessage` 파라미터를 사용할 수 있습니다. 사용자 풀의 `EmailSendingAccount` 속성이 `DEVELOPER`가 아니고 `emailMessage` 파라미터가 반환된 경우 Amazon Cognito에서 400오류 코드 `com.amazonaws.cognito.identity.idp.model.InvalidLambdaResponseException`을 생성합니다. Amazon Simple Email Service(Amazon SES)를 선택하여 이메일 메시지를 보내면 사용자 풀의 `EmailSendingAccount` 속성이 `DEVELOPER`입니다. 그렇지 않으면 이 값은 `COGNITO_DEFAULT`입니다.

emailSubject

사용자 지정 메시지의 제목 줄입니다. 사용자 풀의 EmailSendingAccount 속성이 다음과 같은 경우에만 `emailSubject` 파라미터를 사용할 수 있습니다. `DEVELOPER` 사용자 풀의 `EmailSendingAccount` 속성이 `DEVELOPER`가 아니고 Amazon Cognito에서 `emailSubject` 파라미터를 반환하는 경우 Amazon Cognito에서 400 오류 코드 `com.amazonaws.cognito.identity.idp.model.InvalidLambdaResponseException`을 생성합니다. Amazon Simple Email Service(Amazon SES)를 선택하여 이메일 메시지를 보내도록 선택하면 사용자 풀의 `EmailSendingAccount` 속성이 `DEVELOPER`입니다. 그렇지 않으면 이 값은 `COGNITO_DEFAULT`입니다.

가입용 사용자 지정 메시지의 예

이 Lambda 함수 예는 서비스에서 앱이 사용자에게 확인 코드를 보내야 할 때 이메일 또는 SMS 메시지를 사용자 지정합니다.

Amazon Cognito는 사후 등록, 확인 코드 재전송, 잊어버린 암호 복구 또는 사용자 속성 확인과 같은 여러 이벤트에서 Lambda 트리거를 호출할 수 있습니다. SMS와 이메일 모두의 메시지가 응답에 포함됩

니다. 메시지에는 코드 파라미터 "####"이 포함되어야 합니다. 이 파라미터는 사용자가 수신하는 확인 코드의 자리 표시자입니다.

이메일 메시지의 최대 길이는 UTF-8 20,000자입니다. 이 길이에는 확인 코드가 포함됩니다. 이러한 이메일 메시지에 HTML 태그를 사용할 수 있습니다.

SMS 메시지의 최대 길이는 UTF-8 140자입니다. 이 길이에는 확인 코드가 포함됩니다.

Node.js

```
const handler = async (event) => {
  if (event.triggerSource === "CustomMessage_SignUp") {
    const message = `Thank you for signing up. Your confirmation code is
 ${event.request.codeParameter}.`;
    event.response.smsMessage = message;
    event.response.emailMessage = message;
    event.response.emailSubject = "Welcome to the service.";
  }
  return event;
};

export { handler };
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그런 다음 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. Lambda 콘솔에서 해당 Lambda 트리거와 관련 있는 데이터로 테스트 이벤트를 설정할 수 있습니다. 다음은 이 코드 샘플의 테스트 이벤트입니다.

JSON

```
{
  "version": 1,
  "triggerSource": "CustomMessage_SignUp/CustomMessage_ResendCode/
CustomMessage_ForgotPassword/CustomMessage_VerifyUserAttribute",
  "region": "<region>",
  "userPoolId": "<userPoolId>",
  "userName": "<userName>",
  "callerContext": {
      "awsSdk": "<calling aws sdk with version>",
      "clientId": "<apps client id>",
      ...
  },
  "request": {
```

```
      "userAttributes": {
          "phone_number_verified": false,
          "email_verified": true,
           ...
      },
      "codeParameter": "####"
  },
  "response": {
      "smsMessage": "<custom message to be sent in the message with code parameter>"
      "emailMessage": "<custom message to be sent in the message with code
 parameter>"
      "emailSubject": "<custom email subject>"
  }
}
```

관리자로 사용자 생성용 사용자 정의 메시지 예제

Amazon Cognito가 이 예제 사용자 지정 메시지 Lambda 함수에 전송한 요청에는 `triggerSource` 값, 사용자 이름 `CustomMessage_AdminCreateUser` 및 임시 비밀번호가 있습니다. 함수는 `${event.request.codeParameter}` 요청의 임시 비밀번호와 요청의 사용자 이름으로 채워집니다. `${event.request.usernameParameter}`

사용자 지정 메시지는 응답 개체에 `codeParameter` 및 `usernameParameter` 응답 개체에 값을 삽입해야 합니다. `smsMessage` `emailMessage` 이 예제에서 함수는 응답 필드 `event.response.smsMessage` 및 에 동일한 메시지를 씁니다`event.response.emailMessage`.

이메일 메시지의 최대 길이는 UTF-8 20,000자입니다. 이 길이에는 확인 코드가 포함됩니다. 이러한 이메일에 HTML 태그를 사용할 수 있습니다. SMS 메시지의 최대 길이는 UTF-8 140자입니다. 이 길이에는 확인 코드가 포함됩니다.

SMS와 이메일 모두의 메시지가 응답에 포함됩니다.

Node.js

```
const handler = async (event) => {
  if (event.triggerSource === "CustomMessage_AdminCreateUser") {
    const message = `Welcome to the service. Your user name is
 ${event.request.usernameParameter}. Your temporary password is
 ${event.request.codeParameter}`;
    event.response.smsMessage = message;
    event.response.emailMessage = message;
```

```
    event.response.emailSubject = "Welcome to the service";
  }
  return event;
};

export { handler }
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그런 다음 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. Lambda 콘솔에서 해당 Lambda 트리거와 관련 있는 데이터로 테스트 이벤트를 설정할 수 있습니다. 다음은 이 코드 샘플의 테스트 이벤트입니다.

JSON

```
{
  "version": 1,
  "triggerSource": "CustomMessage_AdminCreateUser",
  "region": "<region>",
  "userPoolId": "<userPoolId>",
  "userName": "<userName>",
  "callerContext": {
      "awsSdk": "<calling aws sdk with version>",
      "clientId": "<apps client id>",
      ...
  },
  "request": {
      "userAttributes": {
          "phone_number_verified": false,
          "email_verified": true,
           ...
      },
      "codeParameter": "####",
      "usernameParameter": "username"
  },
  "response": {
      "smsMessage": "<custom message to be sent in the message with code parameter
 and username parameter>"
      "emailMessage": "<custom message to be sent in the message with code parameter
 and username parameter>"
      "emailSubject": "<custom email subject>"
  }
}
```

사용자 지정 발신자 Lambda 트리거

Amazon Cognito 사용자 풀은 서드 파티 이메일 및 SMS 알림을 활성화하는 Lambda 트리거 `CustomEmailSender` 및 `CustomSMSSender`를 제공합니다. SMS 및 이메일 공급자를 선택하여 Lambda 함수 코드 내에서 사용자에게 알림을 보낼 수 있습니다. Amazon Cognito가 확인 코드, 인증 코드 또는 임시 암호와 같은 알림을 사용자에게 보내야 하는 경우 이벤트는 구성된 Lambda 함수를 활성화합니다. Amazon Cognito는 코드 및 임시 암호(보안 암호)를 활성화된 Lambda에 보냅니다. Amazon Cognito는 이러한 보안 암호를 AWS KMS 고객 관리형 키 및 AWS Encryption SDK로 암호화합니다. AWS Encryption SDK는 일반 데이터를 암호화 및 해독할 수 있도록 도와주는 클라이언트 측 암호화 라이브러리입니다.

> **Note**
>
> 사용자 풀에서 이러한 Lambda 트리거를 사용하도록 구성하려면 AWS CLI 또는 SDK를 사용합니다. Amazon Cognito 콘솔에서는 이러한 구성을 사용할 수 없습니다.

CustomEmailSender

Amazon Cognito는 이 트리거를 호출하여 사용자에게 이메일 알림을 전송합니다.

CustomSMSSender

Amazon Cognito는 이 트리거를 호출하여 사용자에게 SMS 알림을 전송합니다.

리소스

다음 리소스는 `CustomEmailSender` 및 `CustomSMSSender` 트리거를 사용하는 데 도움이 됩니다.

AWS KMS

AWS KMS는 AWS KMS 키를 생성하고 제어하는 관리형 서비스입니다. 이러한 키는 데이터를 암호화합니다. 자세한 내용은 AWS Key Management Service란 무엇입니까?를 참조하세요.

KMS 키

KMS 키는 암호화 키를 논리적으로 표현한 것입니다. KMS 키에는 키 ID, 생성 날짜, 설명 및 키 상태와 같은 메타데이터가 포함됩니다. 또한 KMS 키에는 데이터를 암호화 및 복호화하는 데 사용되는 키 재료도 포함됩니다. 자세한 내용은 AWS KMS 키를 참조하세요.

대칭 KMS 키

대칭 KMS 키는 암호화되지 않은 상태로 AWS KMS를 종료하지 않는 256비트 암호화 키입니다. 대칭 KMS 키를 사용하려면 AWS KMS를 호출해야 합니다. Amazon Cognito는 대칭 키를 사용합니다. 동일한 키가 암호화하고 해독합니다. 자세한 내용은 대칭 KMS 키를 참조하세요.

사용자 정의 이메일 발신자 Lambda 트리거

사용자 지정 이메일 발신자 트리거를 사용자 풀에 할당하면 Amazon Cognito가 사용자 이벤트로 인해 이메일 메시지를 전송해야 하는 경우 기본 동작 대신 Lambda 함수를 호출합니다. 사용자 지정 발신자 트리거를 사용하는 경우 선택한 방법 및 제공업체를 통해 AWS Lambda 함수가 사용자에게 이메일 알림을 보낼 수 있습니다. 함수의 사용자 지정 코드는 사용자 풀의 모든 이메일 메시지를 처리하고 전달해야 합니다.

> **Note**
>
> 현재 Amazon Cognito 콘솔에서 사용자 지정 발신자 트리거를 할당할 수 없습니다. `CreateUserPool` 또는 `UpdateUserPool` API 요청에서 `LambdaConfig` 파라미터를 사용하여 트리거를 할당할 수 있습니다.

이 트리거를 설정하려면 다음 단계를 수행합니다.

1. AWS Key Management Service(AWS KMS)에서 대칭 암호화 키를 생성합니다. Amazon Cognito가 암호(임시 암호, 검증 코드 및 확인 코드)를 생성한 다음 이 KMS 키를 사용하여 암호화합니다. 그런 다음 Lambda 함수에서 Decrypt API를 사용하여 암호를 해독하고 사용자에게 일반 텍스트로 보낼 수 있습니다. AWS Encryption SDK는 함수에서 AWS KMS 작업을 수행하는 데 유용한 도구입니다.
2. 사용자 지정 발신자 트리거로 할당할 Lambda 함수를 생성합니다. KMS 키에 대한 `kms:Decrypt` 권한을 Lambda 함수 역할에 부여합니다.
3. Amazon Cognito 서비스 보안 주체 `cognito-idp.amazonaws.com`에 Lambda 함수를 호출할 수 있는 권한을 부여합니다.
4. 사용자 지정 전송 방법 또는 서드 파티 공급자에게 메시지를 전달하는 Lambda 함수 코드를 작성합니다. 사용자의 검증 또는 확인 코드를 전달하기 위해 Base64는 요청의 `code` 파라미터 값을 디코딩하고 해독합니다. 이 작업을 수행하면 메시지에 포함해야 하는 일반 텍스트 코드 또는 암호가 생성됩니다.
5. 사용자 지정 발신자 Lambda 트리거를 사용하도록 사용자 풀을 업데이트합니다. 사용자 지정 발신자 트리거를 사용하여 사용자 풀을 업데이트하거나 생성하는 IAM 보안 주체는 KMS 키에 대한 권한

부여를 생성할 권한을 가져야 합니다. 다음 LambdaConfig 코드 조각에서는 사용자 지정 SMS 및 이메일 발신자 함수를 할당합니다.

```
"LambdaConfig": {
   "KMSKeyID": "arn:aws:kms:us-
east-1:123456789012:key/a6c4f8e2-0c45-47db-925f-87854bc9e357",
   "CustomEmailSender": {
      "LambdaArn": "arn:aws:lambda:us-east-1:123456789012:function:MyFunction",
      "LambdaVersion": "V1_0"
   },
   "CustomSMSSender": {
      "LambdaArn": "arn:aws:lambda:us-east-1:123456789012:function:MyFunction",
      "LambdaVersion": "V1_0"
   }
```

사용자 지정 이메일 발신자 Lambda 트리거 파라미터

Amazon Cognito가 이 Lambda 함수에 전달하는 요청은 아래 파라미터와 Amazon Cognito가 모든 요청에 추가하는 공통 파라미터의 조합입니다.

JSON

```
{
    "request": {
        "type": "customEmailSenderRequestV1",
        "code": "string",
        "clientMetadata": {
            "string": "string",
             . . .
            },
        "userAttributes": {
            "string": "string",
            . . .
         }
}
```

사용자 지정 이메일 발신자 요청 파라미터

유형

요청 버전입니다. 사용자 지정 이메일 발신자 이벤트의 경우 이 문자열의 값은 항상 `customEmailSenderRequestV1`입니다.

code

함수가 해독하여 사용자에게 보낼 수 있는 암호화된 코드입니다.

clientMetadata

사용자 지정 이메일 발신자 Lambda 함수 트리거에 대한 사용자 지정 입력으로 제공할 수 있는 하나 이상의 키-값 페어입니다. 이 데이터를 Lambda 함수에 전달하려면 AdminRespondToAuthChallenge 및 RespondToAuthChallenge API 작업에서 ClientMetadata 파라미터를 사용합니다. Amazon Cognito는 사후 승인 함수에 전달하는 요청에 있는 AdminInitiateAuth 및 InitiateAuth API 작업의 ClientMetadata 파라미터에서 전달된 데이터를 포함하지 않습니다.

userAttributes

사용자 속성을 나타내는 하나 이상의 키-값 페어입니다.

사용자 지정 이메일 발신자 응답 파라미터

Amazon Cognito는 사용자 지정 이메일 발신자 응답에서 추가 반환 정보를 기대하지 않습니다. 함수는 API 작업을 사용하여 리소스를 쿼리 및 수정하거나 이벤트 메타데이터를 외부 시스템에 기록할 수 있습니다.

사용자 지정 이메일 발신자 Lambda 트리거 활성화

사용자 지정 로직을 사용하여 사용자 풀에 대한 이메일 메시지를 보내는 사용자 지정 이메일 발신자 트리거를 설정하려면 다음과 같이 트리거를 활성화합니다. 다음 절차에서는 사용자 풀에 사용자 지정 이메일 트리거, 사용자 지정 SMS 트리거 또는 둘 다를 할당합니다. 사용자 지정 이메일 발신자 트리거를 추가하면 Amazon Cognito는 Amazon Simple Email Service를 통해 이메일 메시지를 전송했을 때처럼 항상 사용자 속성(예: 이메일 주소) 및 일회성 코드를 Lambda 함수로 전송합니다.

> **Important**
>
> Amazon Cognito HTML은 사용자의 임시 암호에 있는 <(`<`) 및>(`>`) 같은 예약 문자를 이스케이프 처리합니다. 이러한 문자는 Amazon Cognito가 사용자 지정 이메일 발신자 함수로 보

내는 임시 암호에 표시될 수 있지만, 임시 확인 코드에는 표시되지 않습니다. 임시 암호를 전송하려면 Lambda 함수가 암호를 해독하고 이러한 문자를 이스케이프 해제한 다음 사용자에게 메시지를 전송해야 합니다.

1. AWS KMS에서 암호화 키를 생성합니다. 이 키는 Amazon Cognito가 생성하는 임시 암호 및 권한 부여 코드를 암호화합니다. 그런 다음 사용자 지정 발신자 Lambda 함수에서 이러한 보안 암호를 해독하여 사용자에게 일반 텍스트로 보낼 수 있습니다.
2. Amazon Cognito 서비스 주체에 KMS 키로 코드를 암호화할 수 있는`cognito-idp.amazonaws.com` 액세스 권한을 부여합니다.

   다음 리소스 기반 정책을 KMS 키에 적용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": "cognito-idp.amazonaws.com"
        },
        "Action": "kms:CreateGrant",
        "Resource": "arn:aws:kms:us-
west-2:111222333444:key/1example-2222-3333-4444-999example",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "111222333444"
            },
            "ArnLike": {
                "aws:SourceArn": "arn:aws:cognito-idp:us-
west-2:111222333444:userpool/us-east-1_EXAMPLE"
            }
        }
    }]
}
```

3. 사용자 지정 SMS 발신자 트리거에 대한 Lambda 함수를 생성합니다. Amazon Cognito는 AWS 암호화 SDK를 사용하여 암호, 임시 암호 및 사용자의 API 요청을 인증하는 코드를 암호화합니다.
   - 최소한 KMS 키에 대한 `kms:Decrypt` 권한이 있는 Lambda 함수에 IAM 역할을 할당합니다.
4. Amazon Cognito 서비스 보안 주체 `cognito-idp.amazonaws.com`에 Lambda 함수를 호출할 수 있는 권한을 부여합니다.

다음 AWS CLI 명령은 Lambda 함수를 호출할 수 있는 Amazon Cognito 권한을 부여합니다.

```
       aws lambda add-permission --function-name lambda_arn --statement-id
 "CognitoLambdaInvokeAccess" --action lambda:InvokeFunction --principal cognito-
idp.amazonaws.com
```

5. Lambda 함수 코드를 작성하여 메시지를 전송합니다. Amazon Cognito는 AWS Encryption SDK를 사용하여 암호를 암호화한 다음 사용자 지정 발신자 Lambda 함수로 암호를 보냅니다. 함수에서 암호를 해독하고 관련 메타데이터를 처리합니다. 그런 다음 코드, 사용자 지정 메시지 및 대상 전화번호를 메시지를 전달하는 사용자 지정 API에 전송합니다.
6. AWS Encryption SDK를 Lambda 함수에 추가합니다. 자세한 내용은 AWS암호화 SDK 프로그래밍 언어를 참조하세요. Lambda 패키지를 업데이트하려면 다음 단계를 완료하세요.
   a. Lambda 함수를 AWS Management Console에 .zip 파일로 내보냅니다.
   b. 함수를 열고 AWS Encryption SDK를 추가합니다. 자세한 내용과 다운로드 링크는 AWS Encryption SDK개발자 안내서의 AWS Encryption SDK 프로그래밍 언어를 참조하세요.
   c. SDK 종속성과 함께 함수를 압축하고 함수를 Lambda에 업로드합니다. 자세한 내용은 AWS Lambda 개발자 안내서의 Lambda 함수를 .zip 파일 아카이브로 배포를 참조하세요.
7. 사용자 풀을 업데이트하여 사용자 지정 발신자 Lambda 트리거를 추가합니다. `UpdateUserPool` API 요청에 `CustomSMSSender` 또는`CustomEmailSender` 파라미터를 포함합니다. `UpdateUserPool` API 작업에서는 사용자 풀의 모든 파라미터와 변경할 파라미터가 모두 필요합니다. 관련 파라미터를 모두 제공하지 않으면 Amazon Cognito에서 누락된 파라미터 값을 기본값으로 설정합니다. 다음 예시처럼 사용자 풀에 추가하거나 사용자 풀에서 유지할 모든 Lambda 함수의 항목을 포함합니다. 자세한 내용은 사용자 풀 구성 업데이트 섹션을 참조하세요.

```
   #Send this parameter in an 'aws cognito-idp update-user-pool' CLI command,
including any existing
   #user pool configurations.

     --lambda-config "PreSignUp=lambda-arn, \
                      CustomSMSSender={LambdaVersion=V1_0,LambdaArn=lambda-arn}, \
                      CustomEmailSender={LambdaVersion=V1_0,LambdaArn=lambda-arn},
\
                      KMSKeyID=key-id"
```

update-user-pool AWS CLI를 이용해 사용자 지정 발신자 Lambda 트리거를 제거하려면, --lambda-config에서 CustomSMSSender 또는 CustomEmailSender 파라미터를 생략하고 사용자 풀과 함께 사용할 다른 트리거를 모두 포함합니다.

UpdateUserPool API 요청이 포함된 사용자 지정 발신자 Lambda 트리거를 제거하려면 나머지 사용자 풀 구성을 포함하는 요청 본문에서 CustomSMSSender 또는 CustomEmailSender 파라미터를 생략합니다.

코드 예제

다음 Node.js 예제에서는 사용자 지정 이메일 발신자 Lambda 함수에서 이메일 메시지 이벤트를 처리합니다. 이 예에서는 함수에 두 개의 환경 변수가 정의되어 있다고 가정합니다.

KEY_ALIAS

사용자 코드를 암호화하고 복호화하는 데 사용하려는 KMS 키의 별칭입니다.

KEY_ARN

사용자 코드를 암호화하고 복호화하는 데 사용하려는 KMS 키의 Amazon 리소스 이름(ARN)입니다.

```
const AWS = require('aws-sdk');
const b64 = require('base64-js');
const encryptionSdk = require('@aws-crypto/client-node');
//Configure the encryption SDK client with the KMS key from the environment variables.
const { encrypt, decrypt } =
 encryptionSdk.buildClient(encryptionSdk.CommitmentPolicy.REQUIRE_ENCRYPT_ALLOW_DECRYPT);
const generatorKeyId = process.env.KEY_ALIAS;
const keyIds = [ process.env.KEY_ARN ];
const keyring = new encryptionSdk.KmsKeyringNode({ generatorKeyId, keyIds })
exports.handler = async (event) => {
 //Decrypt the secret code using encryption SDK.
 let plainTextCode;
 if(event.request.code){
  const { plaintext, messageHeader } = await decrypt(keyring,
 b64.toByteArray(event.request.code));
  plainTextCode = plaintext
 }
 //PlainTextCode now contains the decrypted secret.
 if(event.triggerSource == 'CustomEmailSender_SignUp'){
  //Send an email message to your user via a custom provider.
```

```
  //Include the temporary password in the message.
 }
 else if(event.triggerSource == 'CustomEmailSender_ResendCode'){
 }
 else if(event.triggerSource == 'CustomEmailSender_ForgotPassword'){
 }
 else if(event.triggerSource == 'CustomEmailSender_UpdateUserAttribute'){
 }
 else if(event.triggerSource == 'CustomEmailSender_VerifyUserAttribute'){
 }
 else if(event.triggerSource == 'CustomEmailSender_AdminCreateUser'){
 }
 else if(event.triggerSource == 'CustomEmailSender_AccountTakeOverNotification'){
 }
 return;
};
```

사용자 정의 이메일 발신자 Lambda 트리거 소스

다음 표에는 Lambda 코드의 사용자 지정 이메일 트리거 소스에 대한 트리거 이벤트가 나와 있습니다.

TriggerSource value	이벤트
CustomEmailSender_SignUp	사용자가 가입하면 Amazon Cognito에서 시작 메시지를 보냅니다.
CustomEmailSender_ForgotPassword	사용자가 암호를 재설정하는 코드를 요청합니다.
CustomEmailSender_ResendCode	사용자가 암호를 재설정하는 대체 코드를 요청합니다.
CustomEmailSender_UpdateUserAttribute	사용자가 이메일 주소 또는 전화 번호 속성을 업데이트하면 Amazon Cognito에서 속성을 확인하는 코드를 보냅니다.
CustomEmailSender_VerifyUserAttribute	사용자가 새 이메일 주소 또는 전화 번호 속성을 생성하면 Amazon Cognito에서 속성을 확인하는 코드를 보냅니다.

TriggerSource value	이벤트
CustomEmailSender_AdminCreateUser	사용자 풀에 새 사용자를 생성하면 Amazon Cognito에서 해당 사용자에게 임시 암호를 보냅니다.
CustomEmailSender_AccountTakeOverNotification	Amazon Cognito는 사용자 계정을 획득하려는 시도를 감지하고 사용자에게 알림을 보냅니다.

사용자 지정 SMS 발신자 Lambda 트리거

사용자 지정 SMS 발신자 트리거를 사용자 풀에 할당하면 Amazon Cognito가 사용자 이벤트로 인해 SMS 메시지를 전송해야 하는 경우 기본 동작 대신 Lambda 함수를 호출합니다. 사용자 지정 발신자 트리거를 사용하면 AWS Lambda 함수에서 선택한 방법 및 공급자를 통해 사용자에게 SMS 알림을 보낼 수 있습니다. 함수의 사용자 지정 코드는 사용자 풀의 모든 SMS 메시지를 처리하고 전달해야 합니다.

> **Note**
>
> 현재 Amazon Cognito 콘솔에서 사용자 지정 발신자 트리거를 할당할 수 없습니다. `CreateUserPool` 또는 `UpdateUserPool` API 요청에서 `LambdaConfig` 파라미터를 사용하여 트리거를 할당할 수 있습니다.

이 트리거를 설정하려면 다음 단계를 수행합니다.

1. AWS Key Management Service ()AWS KMS에서 대칭 암호화 키를 생성합니다. Amazon Cognito가 암호(임시 암호, 검증 코드 및 확인 코드)를 생성한 다음 이 KMS 키를 사용하여 암호화합니다. 그런 다음 Lambda 함수에서 Decrypt API를 사용하여 암호를 해독하고 사용자에게 일반 텍스트로 보낼 수 있습니다. 함수 AWS KMS 작업에 유용한 도구입니다. AWS Encryption SDK
2. 사용자 지정 발신자 트리거로 할당할 Lambda 함수를 생성합니다. KMS 키에 대한 `kms:Decrypt` 권한을 Lambda 함수 역할에 부여합니다.
3. Amazon Cognito 서비스 보안 주체 `cognito-idp.amazonaws.com`에 Lambda 함수를 호출할 수 있는 권한을 부여합니다.
4. 사용자 지정 전송 방법 또는 서드 파티 공급자에게 메시지를 전달하는 Lambda 함수 코드를 작성합니다. 사용자의 검증 또는 확인 코드를 전달하기 위해 Base64는 요청의 `code` 파라미터 값을 디코딩하고 해독합니다. 이 작업을 수행하면 메시지에 포함해야 하는 일반 텍스트 코드 또는 암호가 생성됩니다.

5. 사용자 지정 발신자 Lambda 트리거를 사용하도록 사용자 풀을 업데이트합니다. 사용자 지정 발신자 트리거를 사용하여 사용자 풀을 업데이트하거나 생성하는 IAM 보안 주체는 KMS 키에 대한 권한 부여를 생성할 권한을 가져야 합니다. 다음 `LambdaConfig` 코드 조각에서는 사용자 지정 SMS 및 이메일 발신자 함수를 할당합니다.

```
"LambdaConfig": {
   "KMSKeyID": "arn:aws:kms:us-
east-1:123456789012:key/a6c4f8e2-0c45-47db-925f-87854bc9e357",
   "CustomEmailSender": {
      "LambdaArn": "arn:aws:lambda:us-east-1:123456789012:function:MyFunction",
      "LambdaVersion": "V1_0"
   },
   "CustomSMSSender": {
      "LambdaArn": "arn:aws:lambda:us-east-1:123456789012:function:MyFunction",
      "LambdaVersion": "V1_0"
   }
```

사용자 지정 SMS 발신자 Lambda 트리거 파라미터

Amazon Cognito가 이 Lambda 함수에 전달하는 요청은 아래 파라미터와 Amazon Cognito가 모든 요청에 추가하는 공통 파라미터의 조합입니다.

JSON

```
{
    "request": {
        "type": "customSMSSenderRequestV1",
        "code": "string",
        "clientMetadata": {
            "string": "string",
             . . .
            },
        "userAttributes": {
            "string": "string",
            . . .
         }
}
```

사용자 지정 SMS 발신자 요청 파라미터

유형

요청 버전입니다. 사용자 지정 SMS 발신자 이벤트의 경우 이 문자열의 값은 항상 `customSMSSenderRequestV1`입니다.

code

함수가 해독하여 사용자에게 보낼 수 있는 암호화된 코드입니다.

clientMetadata

사용자 지정 SMS 발신자 Lambda 함수 트리거에 대한 사용자 지정 입력으로 제공할 수 있는 하나 이상의 키-값 페어입니다. 이 데이터를 Lambda 함수로 전달하기 위해 및 API 작업에서 AdminRespondToAuthChallenge파라미터를 사용할 ClientMetadata 수 있습니다. RespondToAuthChallenge Amazon Cognito는 사후 인증 함수로 전달하는 요청에 ClientMetadata 파라미터 입력 AdminInitiateAuth및 InitiateAuthAPI 작업의 데이터를 포함하지 않습니다.

userAttributes

사용자 속성을 나타내는 하나 이상의 키-값 페어입니다.

사용자 지정 SMS 발신자 응답 파라미터

Amazon Cognito는 응답에서 추가 반환 정보를 기대하지 않습니다. 함수는 API 작업을 사용하여 리소스를 쿼리 및 수정하거나 이벤트 메타데이터를 외부 시스템에 기록할 수 있습니다.

사용자 지정 SMS 발신자 Lambda 트리거 활성화

사용자 지정 로직을 사용하여 사용자 풀에 대한 SMS 메시지를 보내는 사용자 지정 SMS 발신자 트리거를 설정할 수 있습니다. 다음 절차에서는 사용자 풀에 사용자 지정 SMS 트리거, 사용자 지정 이메일 트리거 또는 둘 다를 할당합니다. 사용자 지정 SMS 발신자 트리거를 추가하면 Amazon Cognito는 Amazon Simple Notification Service를 통해 SMS 메시지를 보내는 기본 동작을 하는 대신, 전화번호를 비롯한 사용자 속성과 일회용 코드를 항상 Lambda 함수로 전송합니다.

> **Important**
>
> Amazon Cognito HTML은 사용자의 임시 암호에 있는 <(`<`) 및>(`>`) 같은 예약 문자를 이스케이프 처리합니다. 이러한 문자는 Amazon Cognito가 사용자 지정 이메일 발신자 함수로 보내는 임시 암호에 표시될 수 있지만, 임시 확인 코드에는 표시되지 않습니다. 임시 암호를 전송

하려면 Lambda 함수가 암호를 해독하고 이러한 문자를 이스케이프 해제한 다음 사용자에게 메시지를 전송해야 합니다.

1. AWS KMS에서 암호화 키를 생성합니다. 이 키는 Amazon Cognito가 생성하는 임시 암호 및 권한 부여 코드를 암호화합니다. 그런 다음 사용자 지정 발신자 Lambda 함수에서 이러한 보안 암호를 해독하여 사용자에게 일반 텍스트로 보낼 수 있습니다.
2. Amazon Cognito 서비스 주체에 KMS 키로 코드를 암호화할 수 있는`cognito-idp.amazonaws.com` 액세스 권한을 부여합니다.

   다음 리소스 기반 정책을 KMS 키에 적용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": "cognito-idp.amazonaws.com"
        },
        "Action": "kms:CreateGrant",
        "Resource": "arn:aws:kms:us-
west-2:111222333444:key/1example-2222-3333-4444-999example",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "111222333444"
            },
            "ArnLike": {
                "aws:SourceArn": "arn:aws:cognito-idp:us-
west-2:111222333444:userpool/us-east-1_EXAMPLE"
            }
        }
    }]
}
```

3. 사용자 지정 SMS 발신자 트리거에 대한 Lambda 함수를 생성합니다. Amazon Cognito는 AWS 암호화 SDK를 사용하여 암호, 임시 암호 및 사용자의 API 요청을 인증하는 코드를 암호화합니다.
   - 최소한 KMS 키에 대한 `kms:Decrypt` 권한이 있는 Lambda 함수에 IAM 역할을 할당합니다.
4. Amazon Cognito 서비스 보안 주체 `cognito-idp.amazonaws.com`에 Lambda 함수를 호출할 수 있는 권한을 부여합니다.

다음 AWS CLI 명령은 Amazon Cognito에 Lambda 함수를 호출할 수 있는 권한을 부여합니다.

```
       aws lambda add-permission --function-name lambda_arn --statement-id
 "CognitoLambdaInvokeAccess" --action lambda:InvokeFunction --principal cognito-
idp.amazonaws.com
```

5. Lambda 함수 코드를 작성하여 메시지를 전송합니다. Amazon Cognito는 Amazon AWS Encryption SDK Cognito가 사용자 지정 발신자인 Lambda 함수에 암호를 보내기 전에 암호를 암호화하는 데 사용합니다. 함수에서 암호를 해독하고 관련 메타데이터를 처리합니다. 그런 다음 코드, 사용자 지정 메시지 및 대상 전화번호를 메시지를 전달하는 사용자 지정 API에 전송합니다.
6. Lambda 함수에 AWS Encryption SDK 추가합니다. 자세한 내용은 AWS 암호화 SDK 프로그래밍 언어를 참조하세요. Lambda 패키지를 업데이트하려면 다음 단계를 완료하세요.

   a. Lambda 함수를 AWS Management Console에 .zip 파일로 내보냅니다.

   b. 함수를 열고 추가합니다. AWS Encryption SDK자세한 내용과 다운로드 링크는 AWS Encryption SDK 개발자 안내서의 AWS Encryption SDK 프로그래밍 언어를 참조하세요.

   c. SDK 종속성과 함께 함수를 압축하고 함수를 Lambda에 업로드합니다. 자세한 내용은 AWS Lambda 개발자 안내서의 Lambda 함수를 .zip 파일 아카이브로 배포를 참조하세요.
7. 사용자 풀을 업데이트하여 사용자 지정 발신자 Lambda 트리거를 추가합니다. `UpdateUserPool` API 요청에 `CustomSMSSender` 또는`CustomEmailSender` 파라미터를 포함합니다. `UpdateUserPool` API 작업에서는 사용자 풀의 모든 파라미터와 변경할 파라미터가 모두 필요합니다. 관련 파라미터를 모두 제공하지 않으면 Amazon Cognito에서 누락된 파라미터 값을 기본값으로 설정합니다. 다음 예시처럼 사용자 풀에 추가하거나 사용자 풀에서 유지할 모든 Lambda 함수의 항목을 포함합니다. 자세한 설명은 사용자 풀 구성 업데이트 섹션을 참조하세요.

```
   #Send this parameter in an 'aws cognito-idp update-user-pool' CLI command,
including any existing
   #user pool configurations.

     --lambda-config "PreSignUp=lambda-arn, \
                      CustomSMSSender={LambdaVersion=V1_0,LambdaArn=lambda-arn}, \
                      CustomEmailSender={LambdaVersion=V1_0,LambdaArn=lambda-arn},
\
                      KMSKeyID=key-id"
```

를 사용하여 사용자 지정 발신자 Lambda 트리거를 제거하려면 `--lambda-config` 에서 `CustomEmailSender` 또는 파라미터를 `CustomSMSSender` 생략하고 사용자 풀에 사용하려는 다른 모든 트리거를 포함하십시오. `update-user-pool` AWS CLI

`UpdateUserPool` API 요청이 포함된 사용자 지정 발신자 Lambda 트리거를 제거하려면 나머지 사용자 풀 구성을 포함하는 요청 본문에서 `CustomSMSSender` 또는 `CustomEmailSender` 파라미터를 생략합니다.

코드 예제

다음 Node.js 예제에서는 사용자 지정 SMS 발신자 Lambda 함수에서 SMS 메시지 이벤트를 처리합니다. 이 예에서는 함수에 두 개의 환경 변수가 정의되어 있다고 가정합니다.

KEY_ALIAS

사용자 코드를 암호화하고 복호화하는 데 사용하려는 KMS 키의 별칭입니다.

KEY_ARN

사용자 코드를 암호화하고 복호화하는 데 사용하려는 KMS 키의 Amazon 리소스 이름(ARN)입니다.

```
const AWS = require('aws-sdk');
const b64 = require('base64-js');
const encryptionSdk = require('@aws-crypto/client-node');
//Configure the encryption SDK client with the KMS key from the environment variables.

const { encrypt, decrypt } =
 encryptionSdk.buildClient(encryptionSdk.CommitmentPolicy.REQUIRE_ENCRYPT_ALLOW_DECRYPT);
const generatorKeyId = process.env.KEY_ALIAS;
const keyIds = [ process.env.KEY_ARN ];
const keyring = new encryptionSdk.KmsKeyringNode({ generatorKeyId, keyIds })
exports.handler = async (event) => {
 //Decrypt the secret code using encryption SDK.
 let plainTextCode;
 if(event.request.code){
  const { plaintext, messageHeader } = await decrypt(keyring,
 b64.toByteArray(event.request.code));
  plainTextCode = plaintext
 }
 //PlainTextCode now contains the decrypted secret.
 if(event.triggerSource == 'CustomSMSSender_SignUp'){
  //Send an SMS message to your user via a custom provider.
```

```
  //Include the temporary password in the message.
 }
 else if(event.triggerSource == 'CustomSMSSender_ResendCode'){
 }
 else if(event.triggerSource == 'CustomSMSSender_ForgotPassword'){
 }
 else if(event.triggerSource == 'CustomSMSSender_UpdateUserAttribute'){
 }
 else if(event.triggerSource == 'CustomSMSSender_VerifyUserAttribute'){
 }
 else if(event.triggerSource == 'CustomSMSSender_AdminCreateUser'){
 }
 else if(event.triggerSource == 'CustomSMSSender_AccountTakeOverNotification'){
 }
 return;
};
```

주제

- 사용자 지정 SMS 발신자 함수로 SMS 메시지 기능 평가
- 사용자 지정 SMS 발신자 Lambda 트리거 소스

사용자 지정 SMS 발신자 함수로 SMS 메시지 기능 평가

사용자 지정 SMS 발신자 Lambda 함수는 사용자 풀이 보낼 SMS 메시지를 수락하고, 함수는 사용자 지정 논리를 기반으로 콘텐츠를 전달합니다. Amazon Cognito는 사용자 지정 SMS 발신자 Lambda 트리거 파라미터를 사용자의 함수에 보냅니다. 함수는 이 정보로 원하는 것을 할 수 있습니다. 예를 들어, 코드를 Amazon SNS Simple Notification Service(Amazon SNS) 주제에 보낼 수 있습니다. Amazon SNS 주제 구독자는 SMS 메시지, HTTPS 엔드포인트 또는 이메일 주소일 수 있습니다.

사용자 지정 SMS 발신자 Lambda 함수를 사용하여 Amazon Cognito SMS 메시징을 위한 테스트 환경을 만들려면 의 aws-sample 라이브러리에서 development-and-testing-with- sms-redirected-to-email를 amazon-cognito-user-pool 참조하십시오. GitHub 리포지토리에는 새 사용자 풀을 생성하거나 이미 보유한 사용자 풀을 사용할 수 있는 AWS CloudFormation 템플릿이 포함되어 있습니다. 이러한 템플릿은 Lambda 함수와 Amazon SNS 주제를 생성합니다. 템플릿이 사용자 지정 SMS 발신자 트리거로 할당하는 Lambda 함수는 SMS 메시지를 Amazon SNS 주제의 구독자로 리디렉션합니다.

이 솔루션을 사용자 풀에 배포하면 Amazon Cognito가 일반적으로 SMS 메시징을 통해 보내는 모든 메시지, 즉 Lambda 함수는 대신 중앙 이메일 주소로 보냅니다. 이 솔루션을 사용하여 SMS 메시지를 사용자 지정하고 미리 보고 Amazon Cognito에서 SMS 메시지를 보내도록 하는 사용자 풀 이벤트를 테스

트할 수 있습니다. 테스트를 완료한 후에는 CloudFormation 스택을 롤백하거나 사용자 풀에서 사용자 지정 SMS 발신자 함수 할당을 제거하십시오.

> **Important**
>
> amazon-cognito-user-pooldevelopment-and-testing-withsms-redirected-to-email--의 템플릿을 사용하여 프로덕션 환경을 구축하지 마세요. 솔루션의 사용자 지정 SMS 발신자 Lambda 함수에서 SMS 메시지를 시뮬레이션하지만 Lambda 함수는 이 모든 메시지를 하나의 중앙 이메일 주소로 보냅니다. 프로덕션 Amazon Cognito 사용자 풀에서 SMS 메시지를 보내려면 먼저 Amazon Cognito 사용자 풀의 SMS 메시지 설정에 표시된 요구 사항을 완료해야 합니다.

사용자 지정 SMS 발신자 Lambda 트리거 소스

다음 표에는 Lambda 코드의 사용자 지정 SMS 트리거 소스에 대한 트리거 이벤트가 나와 있습니다.

`TriggerSource value`	Event
`CustomSMSSender_SignUp`	사용자가 가입하면 Amazon Cognito에서 시작 메시지를 보냅니다.
`CustomSMSSender_ForgotPassword`	사용자가 암호를 재설정하는 코드를 요청합니다.
`CustomSMSSender_ResendCode`	사용자가 등록을 확인하기 위해 새 코드를 요청합니다.
`CustomSMSSender_VerifyUserAttribute`	사용자가 새 이메일 주소 또는 전화 번호 속성을 생성하면 Amazon Cognito에서 속성을 확인하는 코드를 보냅니다.
`CustomSMSSender_UpdateUserAttribute`	사용자가 이메일 주소 또는 전화 번호 속성을 업데이트하면 Amazon Cognito에서 속성을 확인하는 코드를 보냅니다.
`CustomSMSSender_Authentication`	SMS 멀티 팩터 인증(MFA)으로 구성된 사용자가 로그인합니다.

TriggerSource value	Event
CustomSMSSender_AdminCreateUser	사용자 풀에 새 사용자를 생성하면 Amazon Cognito에서 해당 사용자에게 임시 암호를 보냅니다.

Amazon Cognito 사용자 풀에서 Amazon Pinpoint 분석 사용

Amazon Cognito 사용자 풀은 Amazon Pinpoint와 통합되어 Amazon Cognito 사용자 풀에 대한 분석을 제공하고 Amazon Pinpoint 캠페인을 위한 사용자 데이터를 보강합니다. Amazon Pinpoint는 분석 및 표적화된 캠페인을 제공하여 푸시 알림을 사용하는 모바일 앱에서 사용자 참여를 유도합니다. Amazon Cognito 사용자 풀의 Amazon Pinpoint 분석 지원을 통해 Amazon Pinpoint 콘솔에서 사용자 풀 가입, 로그인, 인증 실패, 일 실사용자(DAU) 및 월 실사용자(MAU)를 추적할 수 있습니다. 디바이스 플랫폼, 디바이스 로캘 및 앱 버전과 같은 다양한 날짜 범위 또는 속성에 대한 데이터를 자세히 확인할 수 있습니다.

앱의 사용자 지정 속성을 설정할 수도 있습니다. 그런 다음 Amazon Pinpoint에서 사용자를 세분화하고 대상 푸시 알림에 전송하는데 이를 사용할 수 있습니다. Amazon Cognito 콘솔의 Analytics(분석) 탭에서 Share user attribute data with Amazon Pinpoint(Amazon Pinpoint와 사용자 속성 데이터 공유)를 선택하면, Amazon Pinpoint는 사용자 이메일 주소와 전화 번호에 대한 추가 엔드포인트를 생성합니다.

Amazon Cognito 콘솔을 사용하여 사용자 풀에서 Amazon Pinpoint 분석을 활성화하면 Amazon Cognito가 사용자 풀을 위해 Amazon Pinpoint에 API 요청을 보낼 때 Amazon Cognito가 맡는 서비스 연결 역할도 생성됩니다. 분석 구성을 추가하는 IAM 보안 주체에게는 CreateServiceLinkedRole 권한이 있어야 합니다. 이 서비스 연결 역할은 AWSServiceRoleForAmazonCognitoIdp입니다. 자세한 내용은 Amazon Cognito에 서비스 연결 역할 사용 섹션을 참조하세요.

Amazon Cognito API에서 앱 클라이언트에 `AnalyticsConfiguration`을 적용하면 Amazon Pinpoint의 사용자 지정 IAM 역할과 이 역할을 맡을 외부 ID를 할당할 수 있습니다. 역할은 `cognito-idp` 서비스 보안 주체를 신뢰해야 하며, 역할 신뢰 정책에 외부 ID가 필요한 경우, 해당 ID가 `AnalyticsConfiguration`과 일치해야 합니다. Amazon Pinpoint 프로젝트를 위한 역할 `cognito-idp:Describe*` 권한 및 다음 권한을 부여해야 합니다.

- `mobiletargeting:UpdateEndpoint`
- `mobiletargeting:PutEvents`

Amazon Cognito 및 Amazon Pinpoint 리전 가용성

다음 표에는 다음 조건 중 하나를 충족하는, Amazon Cognito Cognito와 Amazon Pinpoint 간의 AWS 리전 매핑이 나옵니다.

- Amazon Pinpoint 프로젝트는 미국 동부(버지니아 북부)(us-east-1) 리전에서만 사용할 수 있습니다.
- Amazon Pinpoint 프로젝트는 동일한 리전이나 미국 동부(버지니아 북부)(us-east-1) 리전에서만 사용할 수 있습니다.

기본적으로 Amazon Cognito는 동일한 AWS 리전에 있는 Amazon Pinpoint 프로젝트에만 분석을 전송할 수 있습니다. 다음 표에 나오는 지역과 Amazon Pinpoint를 사용할 수 없는 지역에는 이 규칙이 적용되지 않습니다.

다음 리전에서 Amazon Pinpoint를 사용할 수 없습니다. 이러한 리전에 있는 Amazon Cognito 사용자 풀은 분석을 지원하지 않습니다.

- 유럽(밀라노)
- 중동(바레인)
- 아시아 태평양(오사카)
- 이스라엘(텔아비브)
- 아프리카(케이프타운)
- 아시아 태평양(자카르타)

이 표에서는 Amazon Cognito 사용자 풀을 구축한 리전과 Amazon Pinpoint에 있는 대응하는 리전 간의 관계를 확인할 수 있습니다. Amazon Pinpoint 프로젝트를 Amazon Cognito와 통합하려면 사용 가능한 리전에서 Amazon Pinpoint 프로젝트를 구성해야 합니다.

Amazon Cognito 사용자 풀 리전	Amazon Pinpoint 프로젝트용 리전
ap-northeast-1	us-east-1
ap-northeast-2	us-east-1
ap-south-1	us-east-1, ap-south-1
ap-southeast-1	us-east-1

Amazon Cognito 사용자 풀 리전	Amazon Pinpoint 프로젝트용 리전
ap-southeast-2	us-east-1, ap-southeast-2
ca-central-1	us-east-1
eu-central-1	us-east-1, eu-central-1
eu-west-1	us-east-1, eu-west-1
eu-west-2	us-east-1
us-east-1	us-east-1
us-east-2	us-east-1
us-west-2	us-east-1, us-west-2

리전 매핑 예

- ap-northeast-1에 사용자 풀을 생성하는 경우 us-east-1에 Amazon Pinpoint 프로젝트를 생성할 수 있습니다.
- ap-south-1에 사용자 풀을 생성하는 경우 us-east-1 또는 ap-south-1에 Amazon Pinpoint 프로젝트를 생성할 수 있습니다.

> **Note**
>
> 위 표에 나와 있는 리전을 제외한 모든 AWS 리전에서, Amazon Cognito는 사용자 풀과 동일한 리전에 있는 Amazon Pinpoint 프로젝트만 사용할 수 있습니다. 사용자 풀을 구축한 리전에서 Amazon Pinpoint를 사용할 수 없다면, Amazon Cognito는 해당 리전에서 Amazon Pinpoint 분석을 지원하지 않습니다. 자세한 AWS 리전 정보는 Amazon Pinpoint 엔드포인트 및 할당량을 참조하세요.

Amazon Pinpoint 분석 설정 지정(AWS Management Console)

분석 데이터를 Amazon Pinpoint로 보내도록 Amazon Cognito 사용자 풀을 구성할 수 있습니다. Amazon Cognito는 로컬 사용자에 대한 분석 데이터만 Amazon Pinpoint로 전송합니다. Amazon

Pinpoint 프로젝트와 연결되도록 사용자 풀을 구성한 후에는 `AnalyticsMetadata`를 API 요청에 포함해야 합니다. 자세한 내용은 앱을 Amazon Pinpoint와 통합 섹션을 참조하세요.

분석 설정을 지정하려면

1. Amazon Cognito 콘솔로 이동합니다. AWS 자격 증명을 입력하라는 메시지가 나타날 수 있습니다.
2. User Pools(사용자 풀)을 선택하고 목록에서 기존 사용자 풀을 선택합니다.
3. [앱 통합(App integration)] 탭을 선택합니다.
4. App clients and analytics(앱 클라이언트 및 분석)의 목록에서 기존 App client name(앱 클라이언트 이름)을 선택합니다.
5. Pinpoint analytics(Pinpoint 분석)에서 Enable(활성화)을 선택합니다.
6. Pinpoint Region(Pinpoint 리전)을 선택합니다.
7. Amazon Pinpoint project(Amazon Pinpoint 프로젝트)를 선택하거나 Create Amazon Pinpoint project(Amazon Pinpoint 프로젝트 생성)를 선택합니다.

   > **Note**
   >
   > Amazon Pinpoint 프로젝트 ID는 Amazon Pinpoint 프로젝트에 대해 고유한 32자 문자열입니다. 이 ID는 Amazon Pinpoint 콘솔에 나열됩니다.
   > Amazon Pinpoint 프로젝트에 여러 Amazon Cognito 앱을 매핑할 수 있습니다. 그러나 각 Amazon Cognito 앱은 하나의 Amazon Pinpoint 프로젝트에만 매핑될 수 있습니다.
   > Amazon Pinpoint에서 각 프로젝트는 단일 앱이어야 합니다. 예를 들어, 게임 개발자에게 두 개의 게임이 있고 두 게임 모두 동일한 Amazon Cognito 사용자 풀을 사용한다 하더라도 각 게임은 개별 Amazon Pinpoint 프로젝트입니다. Amazon Pinpoint 프로젝트에 대한 자세한 내용은 Amazon Pinpoint에서 프로젝트 생성을 참조하세요.

8. Amazon Cognito에서 이메일 주소와 전화 번호를 Amazon Pinpoint로 보내 사용자를 위한 추가 엔드포인트를 생성하게 하고 싶다면, User data sharing(사용자 데이터 공유)에서 Share user data with Amazon Pinpoint(Amazon Pinpoint와 사용자 데이터 공유)를 선택합니다. 사용자가 자신의 이메일 주소와 전화 번호를 확인하면, Amazon Cognito는 사용자 계정에 사용할 수 있는 경우에만 이를 Amazon Pinpoint와 공유합니다.

> **Note**
>
> 엔드포인트는 Amazon Pinpoint를 사용하여 푸시 알림이 전송될 수 있는 사용자 디바이스를 고유하게 식별합니다. 엔드포인트에 대한 자세한 내용은 Amazon Pinpoint 개발자 가이드에서 엔드포인트 추가를 참조하세요.

9. 변경 사항 저장을 선택합니다.

Amazon Pinpoint 분석 설정 지정(AWS CLI 및 AWS API)

다음 명령을 사용하여 사용자 풀에 대한 Amazon Pinpoint 분석 설정을 지정합니다.

앱 생성 시간에 사용자 풀의 기존 클라이언트 앱에 대한 분석 설정을 지정하는 방법

- AWS CLI: `aws cognito-idp create-user-pool-client`
- AWS API: CreateUserPoolClient

사용자 풀의 기존 클라이언트 앱에 대한 분석 설정을 업데이트하는 방법

- AWS CLI: `aws cognito-idp update-user-pool-client`
- AWS API: UpdateUserPoolClient

> **Note**
>
> Amazon Cognito는 `ApplicationArn`을 사용하는 경우 리전 내 통합을 지원합니다.

앱을 Amazon Pinpoint와 통합

사용자 풀 API에서 Amazon Cognito 로컬 사용자를 위한 분석 메타데이터를 Amazon Pinpoint에 게시할 수 있습니다.

로컬 사용자

계정에 가입했거나 서드 파티 ID 제공업체(idP)를 통해 로그인하는 대신 사용자 풀에서 생성된 사용자입니다.

사용자 풀 API

사용자 지정 사용자 인터페이스(UI)가 있는 앱을 사용하여 AWS SDK와 통합할 수 있는 작업입니다. 호스팅 UI를 통해 로그인하는 페더레이션 또는 로컬 사용자에 대한 분석 메타데이터를 전달할 수 없습니다. 사용자 풀 API 작업 목록은 Amazon Cognito API 참조에서 확인하세요.

캠페인에 게시하도록 사용자 풀을 구성한 후 Amazon Cognito는 다음 API 작업을 위해 메타데이터를 Amazon Pinpoint에 전달합니다.

- `AdminInitiateAuth`
- `AdminRespondToAuthChallenge`
- `ConfirmForgotPassword`
- `ConfirmSignUp`
- `ForgotPassword`
- `InitiateAuth`
- `ResendConfirmationCode`
- `RespondToAuthChallenge`
- `SignUp`

사용자 세션에 대한 메타데이터를 Amazon Pinpoint 캠페인에 전달하려면 API 요청의 `AnalyticsMetadata` 파라미터에 `AnalyticsEndpointId` 값을 포함합니다. JavaScript 예제는 AWS 지식 센터에서 Amazon Cognito 사용자 풀 분석이 Amazon Pinpoint 대시보드에 나타나지 않는 이유는 무엇입니까?를 참조하세요.

사용자 풀 분석 구성

Amazon Pinpoint 분석을 이용하여 Amazon Cognito 사용자 풀 가입, 로그인, 인증 실패, 일 실사용자(DAU) 및 월 실사용자(MAU)를 추적할 수 있습니다. 또한 AWS Mobile SDK for Android 또는 AWS Mobile SDK for iOS를 사용하여 앱에 고유한 사용자 속성을 설정할 수 있습니다. 그런 다음 Amazon Pinpoint에서 사용자를 세분화하고 대상 푸시 알림에 전송하는데 이를 사용할 수 있습니다.

앱 클라이언트 및 분석 아래의 앱 통합 탭에서 기존 앱 클라이언트로 이동하거나 새 앱 클라이언트를 만들 수 있습니다. 앱 클라이언트 구성 내에서 앱과 함께 사용할 Amazon Pinpoint 프로젝트를 지정할 수 있습니다. 자세한 내용은 Amazon Cognito 사용자 풀과 함께 Amazon Pinpoint 분석 사용을 참조하세요.

> **Note**
>
> Amazon Pinpoint는 북미, 유럽, 아시아 및 오세아니아의 여러 AWS 리전에서 사용할 수 있습니다. Amazon Pinpoint 리전에는 Amazon Pinpoint API가 포함됩니다. Amazon Cognito에서 Amazon Pinpoint 리전을 지원하는 경우 Amazon Cognito는 이벤트를 동일한 Amazon Pinpoint 리전 내의 Amazon Pinpoint 프로젝트에 보냅니다. Amazon Pinpoint에서 해당 리전을 지원하지 않는 경우 Amazon Cognito는 us-east-1에서만 이벤트 전송을 지원합니다. 자세한 Amazon Pinpoint 리전 정보는 Amazon Pinpoint 엔드포인트 및 할당량과 Amazon Cognito 사용자 풀로 Amazon Pinpoint 분석 사용을 참조하세요.

분석 및 캠페인을 추가하려면

1. Add analytics and campaigns(분석 및 캠페인 추가)를 선택합니다.
2. 목록에서 Cognito app client(Cognito 앱 클라이언트)를 선택합니다.
3. Amazon Cognito 앱을 Amazon Pinpoint 프로젝트에 매핑하려면 목록에서 Amazon Pinpoint 프로젝트를 선택합니다.

   > **Note**
   >
   > Amazon Pinpoint 프로젝트 ID는 Amazon Pinpoint 프로젝트에 대해 고유한 32자 문자열입니다. 이 ID는 Amazon Pinpoint 콘솔에 나열됩니다.
   > Amazon Pinpoint 프로젝트에 여러 Amazon Cognito 앱을 매핑할 수 있습니다. 그러나 각 Amazon Cognito 앱은 하나의 Amazon Pinpoint 프로젝트에만 매핑될 수 있습니다.
   > Amazon Pinpoint에서 각 프로젝트는 단일 앱이어야 합니다. 예를 들어, 게임 개발자에게 두 개의 게임이 있고 두 게임 모두 동일한 Amazon Cognito 사용자 풀을 사용한다 하더라도 각 게임은 개별 Amazon Pinpoint 프로젝트입니다.

4. 사용자를 위한 추가 엔드포인트를 생성하기 위해 Amazon Cognito가 이메일 주소와 전화번호를 Amazon Pinpoint로 전송하도록 하려는 경우 [Amazon Pinpoint와 사용자 속성 데이터 공유(Share user attribute data with Amazon Pinpoint)]를 선택합니다.

   > **Note**
   >
   > 엔드포인트는 Amazon Pinpoint를 사용하여 푸시 알림이 전송될 수 있는 사용자 디바이스를 고유하게 식별합니다. 엔드포인트에 대한 자세한 내용은 Amazon Pinpoint 개발자 가이드에서 Amazon Pinpoint에 엔드포인트 추가를 참조하세요.

5. 이미 생성한 IAM 역할을 입력하거나 [새 역할 생성(Create new role)]을 선택하여 IAM 콘솔에서 새 역할을 생성합니다.
6. 변경 사항 저장을 선택합니다.
7. 추가 앱 매핑을 지정하려면 Add app mapping(앱 매핑 추가)을 선택합니다.
8. 변경 사항 저장을 선택합니다.

사용자 풀의 사용자 관리

사용자 풀을 생성한 후 사용자 계정을 생성, 확인 및 관리할 수 있습니다. Amazon Cognito 사용자 풀 그룹을 사용하면 IAM 역할을 그룹에 매핑하여 사용자와 리소스에 대한 액세스를 관리할 수 있습니다.

사용자 마이그레이션 Lambda 트리거를 사용하여 사용자를 사용자 풀로 가져올 수 있습니다. 이 방법을 사용하면 사용자가 사용자 풀에 처음 로그인할 때 기존 사용자 디렉터리에서 사용자 풀로 원활하게 마이그레이션할 수 있습니다.

주제

- 사용자 생성을 위한 정책 구성
- 사용자 계정 가입 및 확인
- 관리자로 사용자 계정 생성
- 사용자 풀에 그룹 추가
- 사용자 계정 관리 및 검색
- 사용자 계정 복구
- 사용자 풀로 사용자 가져오기
- 사용자 풀 속성
- 사용자 풀 암호 요구 사항 추가

사용자 생성을 위한 정책 구성

사용자 풀에서 사용자가 가입하도록 허용하거나 사용자를 관리자로 생성할 수 있습니다. 또한 가입 후 인증 및 확인 프로세스를 사용자에게 맡기는 정도를 제어할 수 있습니다. 예를 들어, 가입을 검토하고 외부 검증 프로세스에 따라 가입을 수락하고 싶을 수 있습니다. 이 구성 또는 관리자로 사용자 생성 정책은 또한 사용자가 더 이상 사용자 계정을 확인할 수 없을 때까지의 시간을 설정합니다.

Amazon Cognito는 소프트웨어를 위한 고객 ID 및 액세스 관리(CIAM) 플랫폼으로서 일반 고객의 요구를 충족할 수 있습니다. 호스팅된 UI의 유무에 관계없이 가입을 수락하고 앱 클라이언트가 있는 사

용자 풀은 공개적으로 검색 가능한 앱 클라이언트 ID와 가입 요청을 아는 인터넷 상의 모든 사용자를 위한 사용자 프로필을 생성합니다. 가입한 사용자 프로필은 액세스 및 ID 토큰을 받을 수 있으며 앱에 대해 승인된 리소스에 액세스할 수 있습니다. 사용자 풀에서 가입을 활성화하기 전에 옵션을 검토하고 구성이 보안 표준을 준수하는지 확인하십시오. 다음 절차에 설명된 대로 자체 등록 활성화 및 `AllowAdminCreateUserOnly`를 주의해서 설정하십시오.

AWS Management Console

사용자 풀의 가입 경험 탭과 사용자 풀 생성 마법사의 가입 경험 구성 단계에는 사용자 풀 내 사용자의 가입 및 관리자 생성을 위한 몇 가지 설정이 포함되어 있습니다.

가입 환경 구성

1. Cognito 지원 검증 및 확인에서 Cognito가 검증 및 확인을 위한 메시지를 자동으로 보내도록 허용 여부를 선택합니다. 이 설정을 활성화하면 Amazon Cognito는 사용자 풀에 제시해야 하는 코드가 포함된 이메일 또는 SMS 메시지를 새 사용자에게 보냅니다. 이렇게 하면 이메일 주소 또는 전화 번호의 소유권을 확인하고, 동등한 속성을 확인된 것으로 설정하고, 로그인할 사용자 계정을 확인합니다. 선택한 확인할 속성에 따라 확인 메시지의 전달 방법 및 대상이 결정됩니다.
2. 사용자를 생성할 때 속성 변경 확인은 중요하지 않지만, 이는 속성 확인과 관련이 있습니다. 로그인 속성을 변경했지만 아직 확인하지 않은 사용자가 새 속성 값이나 원래 속성으로 계속 로그인하도록 허용할 수 있습니다. 자세한 내용은 사용자가 이메일 또는 전화 번호를 변경하는 경우 확인 섹션을 참조하세요.
3. 필수 속성에는 사용자가 가입하거나 사용자를 생성하기 전에 값을 제공해야 하는 속성이 표시됩니다. 사용자 풀 생성 마법사에서는 필수 속성만 설정할 수 있습니다.
4. 사용자 지정 속성은 사용자를 처음 만들 때만 변경할 수 없는 사용자 지정 속성의 값을 설정할 수 있기 때문에 사용자 생성 및 가입 프로세스에서 중요합니다. 사용자 지정 속성에 대한 자세한 내용은 사용자 지정 속성 섹션을 참조하십시오.
5. 사용자가 인증되지 않은 `SignUp` API로 새 계정을 생성할 수 있도록 하려면 셀프 서비스 가입에서 자체 등록 활성화를 선택하십시오. 자체 등록을 비활성화하면 Amazon Cognito 콘솔에서 또는 AdminCreateUser API 요청으로 새 사용자만 관리자로 생성할 수 있습니다. 자체 등록이 비활성화된 사용자 풀에서는 SignUp API 요청이 `NotAuthorizedException`을 반환하고 호스팅된 UI에 가입 링크가 표시되지 않습니다.

사용자를 관리자로 생성하려는 사용자 풀의 경우 관리자가 설정한 임시 암호 만료 기간의 로그인 환경 탭에서 임시 암호의 사용 기간을 구성할 수 있습니다.

사용자를 관리자로서 생성할 때 고려해야 할 또 다른 중요한 요소는 초대 메시지입니다. 새 사용자를 생성하면 Amazon Cognito에서 앱 링크가 포함된 메시지를 전송하여 사용자가 처음으로 로그인할 수 있도록 합니다. 메시지 템플릿의 메시징 탭에서 이 메시지 템플릿을 사용자 지정합니다.

앱 클라이언트 암호가 없으면 가입을 방지하는 클라이언트 암호를 사용하여 기밀 앱 클라이언트(일반적으로 웹 애플리케이션)를 구성할 수 있습니다. 보안 모범 사례에 따라 공용 앱 클라이언트(일반적으로 모바일 앱)에서 앱 클라이언트 암호를 배포하지 마십시오. Amazon Cognito 콘솔의 앱 통합 탭에서 클라이언트 암호를 사용하여 앱 클라이언트를 생성할 수 있습니다.

Amazon Cognito user pools API

CreateUserPool 또는 UpdateUserPool API 요청에서 사용자 풀의 사용자 생성을 위한 파라미터를 프로그래밍 방식으로 설정할 수 있습니다.

AdminCreateUserConfig 요소는 사용자 풀의 다음 속성에 대한 값을 설정합니다.

1. 셀프 서비스 가입 활성화
2. 새 관리자 생성 사용자에게 보내는 초대 메시지

다음 예시를 전체 API 요청 본문에 추가하면 셀프 서비스 가입이 비활성화되고 기본 초대 이메일이 포함된 사용자 풀을 설정합니다.

```
"AdminCreateUserConfig": {
      "AllowAdminCreateUserOnly": true,
      "InviteMessageTemplate": {
         "EmailMessage": "Your username is {username} and temporary password is
 {####}.",
         "EmailSubject": "Welcome to ExampleApp",
         "SMSMessage": "Your username is {username} and temporary password is
 {####}."
      }
   }
```

CreateUserPool 또는 UpdateUserPool API 요청의 다음 추가 파라미터는 새 사용자 생성을 제어합니다.

AutoVerifiedAttributes

새 사용자를 등록할 때 메시지를 자동으로 전송하려는 속성, 이메일 주소 또는 전화번호입니다.

정책

사용자 풀 암호 정책입니다.

스키마

사용자 풀 사용자 지정 특성입니다. 이들은 사용자를 처음 만들 때만 변경할 수 없는 사용자 지정 속성의 값을 설정할 수 있기 때문에 사용자 생성 및 가입 프로세스에서 중요합니다.

또한 이 파라미터는 사용자 풀에 필요한 속성을 설정합니다. 다음 텍스트를 전체 API 요청 본문의 `Schema` 요소에 삽입한 후 필요에 따라 `email` 속성을 설정합니다.

```
{
            "Name": "email",
            "Required": true
}
```

사용자 계정 가입 및 확인

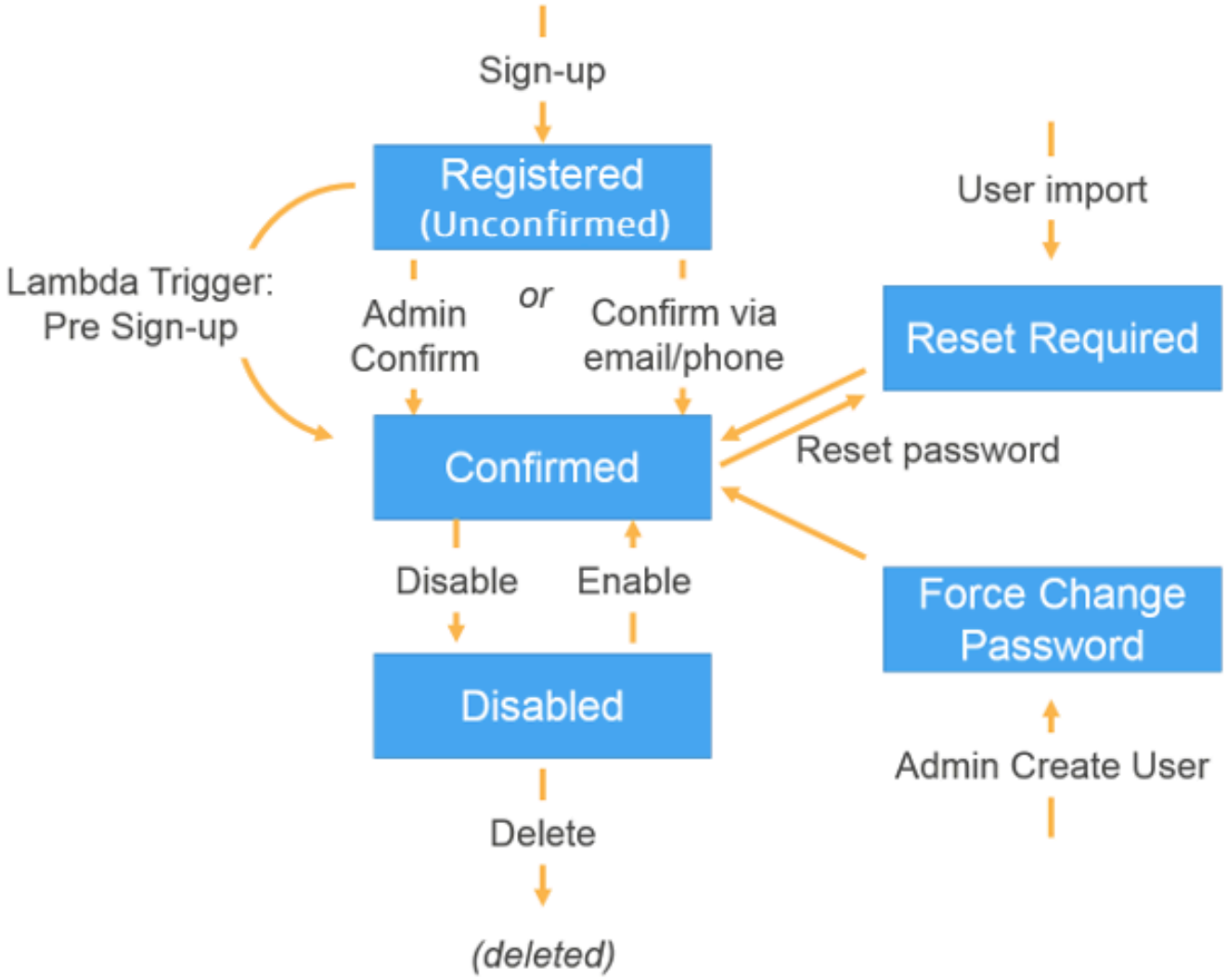
사용자 계정은 다음 방법 중 하나로 사용자 풀에 추가됩니다.

- 사용자가 사용자 풀의 클라이언트 앱에 가입합니다. 모바일 또는 웹 앱일 수 있습니다.
- 사용자 계정을 사용자 풀로 가져올 수 있습니다. 자세한 내용은 CSV 파일에서 사용자 풀로 사용자 가져오기 섹션을 참조하세요.
- 사용자 풀에 사용자 계정을 만들고 로그인하도록 사용자를 초대할 수 있습니다. 자세한 정보는 관리자로 사용자 계정 생성을 참조하세요.

가입한 사용자가 로그인하려면 먼저 확인을 받아야 합니다. 가져온 사용자 및 생성된 사용자는 이미 확인되었지만 처음 로그인할 때 자신의 암호를 생성해야 합니다. 다음 섹션에서는 확인 프로세스와 이메일 및 전화 확인에 대해 설명합니다.

사용자 계정 확인 개요

다음 다이어그램은 확인 프로세스를 보여 줍니다.



사용자 계정은 다음 상태 중 하나일 수 있습니다.

등록됨(확인되지 않음)

사용자가 성공적으로 가입했지만 사용자 계정이 확인될 때까지 로그인할 수 없습니다. 이 상태에서는 사용자가 활성화되었지만 확인되지 않았습니다.

자신을 등록하는 새 사용자는 이 상태에서 시작됩니다.

확인됨

사용자 계정이 확인되었으며 사용자가 로그인할 수 있습니다. 사용자가 코드를 입력하거나 이메일 링크를 따라 사용자 계정을 확인하면 해당 이메일 또는 전화번호가 자동으로 확인됩니다. 코드나 링크는 24시간 동안 유효합니다.

관리자 또는 사전 가입 Lambda 트리거에서 사용자 계정을 확인한 경우 계정에 연결되어 있는 확인된 이메일이나 전화번호가 없을 수 있습니다.

암호 재설정 필요

사용자 계정이 확인되었지만, 사용자가 로그인하기 전에 코드를 요청하고 암호를 재설정해야 합니다.

관리자 또는 개발자가 가져온 사용자 계정은 이 상태에서 시작합니다.

암호 강제 변경

사용자 계정이 확인되고 사용자가 임시 암호를 사용하여 로그인할 수 있지만, 처음 로그인할 때 다른 작업을 수행하기 전에 암호를 새 값으로 변경해야 합니다.

관리자 또는 개발자가 만든 사용자 계정은 이 상태에서 시작합니다.

Disabled(비활성)

사용자 계정을 삭제하려면 먼저 해당 사용자에 대한 로그인 액세스를 비활성화해야 합니다.

가입 시 연락처 정보 확인

새 사용자가 앱에 가입할 때 적어도 하나의 연락 방법을 제공하도록 하고 싶을 것입니다. 예를 들어 사용자의 연락처 정보를 사용하여 다음을 수행할 수 있습니다.

- 사용자가 암호를 재설정하기로 선택한 경우 임시 암호를 보냅니다.
- 개인 정보 또는 재무 정보가 업데이트되면 사용자에게 알립니다.
- 특별 제품 및 서비스 또는 할인과 같은 프로모션 메시지를 보냅니다.
- 계정 요약 또는 결제 미리 알림을 보냅니다.

이와 같은 사용 사례의 경우 확인된 대상에 메시지를 보내는 것이 중요합니다. 그렇지 않으면 잘못 입력한 이메일 주소 또는 전화 번호로 메시지를 보내게 될 수 있습니다. 또는 최악의 경우, 사용자를 사칭하는 악의적인 행위자에게 민감한 정보를 보내게 될 수도 있습니다.

올바른 사람에게만 메시지를 보내려면 사용자가 가입할 때 다음을 반드시 제공하도록 Amazon Cognito 사용자 풀을 구성합니다.

a. 이메일 주소 또는 전화 번호.

b. Amazon Cognito가 이메일 주소나 전화 번호로 전송하는 인증 코드. 24시간이 지났는데 사용자 코드나 링크가 더 이상 유효하지 않은 경우 ResendConfirmationCodeAPI 작업을 호출하여 새 코드나 링크를 생성하고 전송하십시오.

확인 코드를 제공하면 코드를 받은 메일함이나 휴대폰에 액세스할 수 있음을 사용자가 증명합니다. 사용자가 코드를 제공하고 나면 Amazon Cognito는 다음과 같은 방법으로 사용자 풀에서 사용자에 대한 정보를 업데이트합니다.

- 사용자의 상태를 `CONFIRMED`로 설정합니다.
- 이메일 주소나 전화 번호가 확인되었음을 나타내도록 사용자 속성을 업데이트합니다.

이 정보를 보려면 Amazon Cognito 콘솔을 사용하면 됩니다. 또는 AWS SDK 중 하나에서 `AdminGetUser` API 작업, 과 함께 `admin-get-user` 명령 또는 해당 작업을 사용할 수 있습니다. AWS CLI

사용자에게 확인된 연락 방법이 있는 경우 사용자가 암호 재설정을 요청할 때 Amazon Cognito가 자동으로 사용자에게 메시지를 보냅니다.

이메일 또는 전화 확인을 요구하도록 사용자 풀을 구성하려면

사용자의 이메일 주소와 전화번호를 확인할 때 사용자에게 연락할 수 있는지 확인합니다. 의 다음 단계를 AWS Management Console 완료하여 사용자가 이메일 주소 또는 전화번호를 확인하도록 사용자 풀을 구성하십시오.

> **Note**
>
> 아직 계정에 사용자 풀을 만들지 않았으면 사용자 풀 시작하기 섹션을 참조하세요.

사용자 풀을 구성하려면

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 AWS 자격 증명을 입력합니다.
2. 탐색 창에서 사용자 풀(User Pools)을 선택합니다. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
3. 가입 경험(Sign-up experience) 탭을 선택하고 속성 확인 및 사용자 계정 확인(Attribute verification and user account confirmation)을 찾습니다. 편집을 선택합니다.
4. Cognito 지원 검증 및 확인에서 Cognito가 검증 및 확인을 위한 메시지를 자동으로 보내도록 허용 여부를 선택합니다. 이 설정을 활성화하면 Amazon Cognito는 사용자가 가입하거나 사용자 프로

필을 생성할 때 선택한 사용자 연락처 속성으로 메시지를 보냅니다. 속성을 확인하고 로그인을 위한 사용자 프로필을 확인하기 위해 Amazon Cognito는 코드나 링크를 메시지로 사용자에게 보냅니다. 이후 사용자가 UI에 코드를 입력해야 앱이 `ConfirmSignUp` 또는 `AdminConfirmSignUp` API 요청에서 이를 확인할 수 있습니다.

> **Note**
>
> [Cognito 지원 검증 및 확인(Cognito-assisted verification and confirmation)]을 사용하지 않도록 설정하고 인증된 API 작업이나 Lambda 트리거를 사용하여 속성을 검증하고 사용자를 확인할 수도 있습니다.
> 이 옵션을 선택하면 사용자가 가입할 때 Amazon Cognito가 확인 코드를 보내지 않습니다. Amazon Cognito의 확인 코드를 사용하지 않고 하나 이상의 연락 방법을 확인하는 사용자 지정 인증 흐름을 사용하는 경우 이 옵션을 선택합니다. 예를 들어 특정 도메인에 속하는 이메일 주소를 자동으로 확인하는 사전 가입 Lambda 트리거를 사용할 수 있습니다.
> 사용자의 연락처 정보를 확인하지 않으면 사용자가 앱을 사용하지 못할 수 있습니다. 다음을 위해 사용자로부터 확인된 연락처 정보를 요구하는 것입니다.
>
> - [암호 재설정(Reset their passwords)] - 사용자가 앱에서 `ForgotPassword` API 작업을 호출하는 옵션을 선택하면 Amazon Cognito는 사용자의 이메일 주소나 전화 번호로 임시 암호를 전송합니다. Amazon Cognito는 사용자에게 확인된 연락 방법이 하나 이상 있는 경우에만 이 암호를 전송합니다.
> - [이메일 주소나 전화 번호를 별칭으로 사용하여 로그인(Sign in by using an email address or phone number as an alias)] - 이러한 별칭을 허용하도록 사용자 풀을 구성하면 사용자는 별칭이 확인된 경우에만 별칭으로 로그인할 수 있습니다. 자세한 내용은 로그인 속성 사용자 지정 섹션을 참조하세요.

5. [확인할 속성(Attributes to verify)]을 선택합니다.

   [SMS 메시지 전송, 전화 번호 확인(Send SMS message, verify phone number)]

   Amazon Cognito에서 사용자가 가입할 때 SMS 메시지로 확인 코드를 전송합니다. 일반적으로 SMS 메시지를 통해 사용자와 커뮤니케이션하는 경우 이 옵션을 선택합니다. 예를 들어 배송 알림, 약속 확인 또는 알림을 보내는 경우 확인된 전화 번호를 사용할 수 있습니다. 사용자 전화 번호는 계정을 확인할 때 검증되는 속성으로, 사용자 이메일 주소를 확인하고 커뮤니케이션하려면 추가 조치를 취해야 합니다.

[이메일 메시지 전송, 이메일 주소 확인(Send email message, verify email address)]

Amazon Cognito에서 사용자가 가입할 때 이메일 메시지를 통해 확인 코드를 전송합니다. 일반적으로 이메일을 통해 사용자와 커뮤니케이션하는 경우 이 옵션을 선택합니다. 예를 들어 결제 청구서, 주문 요약 또는 특별 제품 및 서비스를 보내는 경우 확인된 이메일 주소를 사용할 수 있습니다. 사용자 이메일 주소는 계정을 확인할 때 검증되는 속성으로, 사용자 전화 번호를 확인하고 커뮤니케이션하려면 추가 조치를 취해야 합니다.

[전화 번호를 사용할 수 있는 경우 SMS 메시지 전송, 그렇지 않으면 이메일 메시지 전송(Send SMS message if phone number is available, otherwise send email message)]

모든 사용자에게 동일한 확인된 연락 방법을 요구하지 않는 경우 이 옵션을 선택합니다. 이 경우 앱의 가입 페이지에서 사용자에게 선호하는 연락 방법만 확인하도록 요청할 수 있습니다. 확인 코드를 전송할 때 Amazon Cognito는 앱에서 `SignUp` 요청 시에 제공한 연락 수단으로 코드를 보냅니다. 사용자가 이메일 주소와 전화 번호를 모두 제공하고 앱이 `SignUp` 요청에서 두 가지 연락 수단을 모두 제공할 경우 Amazon Cognito는 전화 번호로만 확인 코드를 전송합니다.

사용자가 이메일 주소와 전화 번호를 모두 확인하도록 요구하는 경우 이 옵션을 선택합니다. Amazon Cognito는 사용자가 등록할 때 하나의 연락 방법을 확인하며, 사용자가 로그인한 후 앱이 다른 연락 방법을 확인해야 합니다. 자세한 내용은 사용자가 이메일 주소와 전화 번호를 둘 다 확인하도록 요구하는 경우 섹션을 참조하세요.

6. 변경 사항 저장(Save changes)을 선택합니다.

이메일 또는 전화 확인을 사용하는 인증 흐름

사용자 풀에서 사용자가 연락처 정보를 확인해야 하는 경우 사용자가 가입할 때 앱은 다음과 같은 흐름을 지원해야 합니다.

1. 사용자가 사용자 이름, 전화 번호 및/또는 이메일 주소와 기타 속성을 입력하여 앱에 가입합니다.
2. Amazon Cognito 서비스가 앱으로부터 가입 요청을 받습니다. 요청에 가입에 필요한 모든 속성이 포함되어 있는지 확인한 후, 이 서비스는 가입 프로세스를 완료하고 사용자의 휴대폰(SMS 메시지) 또는 이메일로 확인 코드를 보냅니다. 이 코드는 24시간 동안 유효합니다.
3. 가입이 완료되었으며 사용자 계정이 확인 대기 중이라는 메시지를 서비스가 앱에 반환합니다. 해당 응답에는 확인 코드가 전송된 위치에 대한 정보가 포함됩니다. 이 시점에 사용자의 계정은 확인되지 않은 상태이며 사용자의 이메일 주소와 전화 번호가 확인되지 않았습니다.
4. 이제 앱에서 확인 코드를 입력하라는 메시지를 표시할 수 있습니다. 사용자가 코드를 즉시 입력할 필요는 없습니다. 단, 사용자는 확인 코드를 입력할 때까지 로그인할 수 없습니다.

5. 사용자가 앱에서 확인 코드를 입력합니다.
6. 앱이 ConfirmSignUp을 호출하여 코드를 Amazon Cognito 서비스로 전송합니다. 이 서비스는 코드를 확인하고 코드가 올바르면 사용자 계정을 확인 상태로 설정합니다. 사용자 계정을 성공적으로 확인한 후 Amazon Cognito 서비스는 확인하는 데 사용된 속성(이메일 또는 전화번호)을 자동으로 확인된 것으로 표시합니다. 이 속성의 값이 변경되지 않는 한, 사용자가 다시 확인할 필요가 없습니다.
7. 이 시점에 사용자 계정은 확인된 상태이며 사용자가 로그인할 수 있습니다.

사용자가 이메일 주소와 전화 번호를 둘 다 확인하도록 요구하는 경우

Amazon Cognito는 사용자가 가입할 때 하나의 연락 방법만 확인합니다. 이메일 주소 또는 전화번호 확인 중에서 선택해야 하는 경우 Amazon Cognito는 SMS 메시지를 통해 확인 코드를 전송하여 전화번호를 확인하도록 선택합니다. 예를 들어 사용자가 이메일 주소 또는 전화 번호를 확인할 수 있도록 사용자 풀을 구성하고 가입 시 앱이 이러한 속성을 모두 제공하는 경우 Amazon Cognito는 전화 번호만 확인합니다. 사용자가 전화번호를 확인하고 나면 Amazon Cognito가 사용자의 상태를 `CONFIRMED`로 설정하며, 사용자는 앱에 로그인할 수 있습니다.

사용자가 로그인하면 가입 중에 확인되지 않은 연락 방법을 확인하는 옵션을 앱에서 제공할 수 있습니다. 이 두 번째 수단을 확인하기 위해 앱은 `VerifyUserAttribute` API 작업을 호출합니다. 이 작업을 수행하려면 `AccessToken` 파라미터가 필요하며, Amazon Cognito는 인증된 사용자에 대한 액세스 토큰만 제공합니다. 따라서 사용자가 로그인한 후에만 두 번째 연락 방법을 확인할 수 있습니다.

사용자가 이메일 주소와 전화 번호를 모두 확인하도록 요구하는 경우 다음을 수행합니다.

1. 사용자가 이메일 주소 또는 전화 번호를 확인할 수 있도록 사용자 풀을 구성합니다.
2. 앱의 가입 흐름에서 사용자에게 이메일 주소와 전화 번호를 모두 제공하도록 요구합니다. SignUp API 작업을 호출하고 `UserAttributes` 파라미터에서 이메일 주소와 전화 번호를 제공합니다. 이때 Amazon Cognito는 사용자의 휴대폰으로 확인 코드를 전송합니다.
3. 앱 인터페이스에서 사용자가 확인 코드를 입력하는 확인 페이지를 표시합니다. ConfirmSignUp API 작업을 호출하여 사용자를 확인합니다. 이 시점에 사용자의 상태는 `CONFIRMED`이며 사용자의 전화 번호는 확인되었지만 이메일 주소는 확인되지 않았습니다.
4. 로그인 페이지를 표시하고 InitiateAuth API 작업을 호출하여 사용자를 인증합니다. 사용자가 인증되면 Amazon Cognito는 앱에 액세스 토큰을 반환합니다.
5. GetUserAttributeVerificationCode API 작업을 호출합니다. 요청에 다음 파라미터를 지정합니다.

   - `AccessToken` - 사용자가 로그인할 때 Amazon Cognito가 반환하는 액세스 토큰입니다.

- `AttributeName` - `"email"`을 속성 값으로 지정합니다.

Amazon Cognito가 사용자의 이메일 주소로 확인 코드를 전송합니다.

6. 사용자가 확인 코드를 입력하는 확인 페이지를 표시합니다. 사용자가 코드를 제출하면 `VerifyUserAttribute` API 작업을 호출합니다. 요청에 다음 파라미터를 지정합니다.

   - `AccessToken` - 사용자가 로그인할 때 Amazon Cognito가 반환하는 액세스 토큰입니다.
   - `AttributeName` - `"email"`을 속성 값으로 지정합니다.
   - `Code` - 사용자가 제공한 확인 코드입니다.

   이 시점에 이메일 주소는 확인된 상태입니다.

사용자가 앱에 가입할 수 있도록 허용하지만 사용자 풀 관리자로 확인

사용자 풀이 사용자 풀에 자동으로 확인 메시지를 보내는 것을 원하지 않지만 누구나 계정에 가입하도록 허용하려는 경우가 있을 수 있습니다. 이 모델에서는 예를 들어 신규 가입 요청을 사람이 직접 검토하고 가입을 일괄적으로 검증하고 처리할 수 있는 여지가 남아 있습니다. Amazon Cognito 콘솔에서 또는 IAM 인증 API 작업을 통해 새 사용자 계정을 확인할 수 있습니다. AdminConfirmSignUp 사용자 풀의 확인 메시지 전송 여부에 관계없이 관리자로서 사용자 계정을 확인할 수 있습니다.

이 방법을 통해서만 사용자 셀프 서비스 가입을 확인할 수 있습니다. 관리자로 생성한 사용자를 확인하려면 set으로 설정된 AdminSetUserPasswordAPI 요청을 생성하십시오. `Permanent True`

1. 사용자가 사용자 이름, 전화 번호 및/또는 이메일 주소와 기타 속성을 입력하여 앱에 가입합니다.
2. Amazon Cognito 서비스가 앱으로부터 가입 요청을 받습니다. 요청에 가입에 필요한 모든 속성이 포함되어 있는지 확인한 후, 이 서비스는 가입 프로세스를 완료하고 등록이 완료되었으며 확인 대기 중이라는 메시지를 앱에 반환합니다. 이 시점에 사용자의 계정은 확인되지 않은 상태입니다. 사용자는 계정이 확인될 때까지 로그인할 수 없습니다.
3. 사용자의 계정을 확인합니다. 계정을 확인하려면 AWS 자격 증명으로 API 요청에 AWS Management Console 로그인하거나 서명해야 합니다.

   a. Amazon Cognito 콘솔에서 사용자를 확인하려면 사용자 탭으로 이동하여 확인하려는 사용자를 선택한 다음 작업 메뉴에서 확인을 선택합니다.

   b. AWS API 또는 CLI에서 사용자를 확인하려면 AdminConfirmSignUpAPI 요청을 생성하거나 admin-confirm-sign-up에서 생성하십시오. AWS CLI
4. 이 시점에 사용자 계정은 확인된 상태이며 사용자가 로그인할 수 있습니다.

암호 해시 값 컴퓨팅

모범 사례에 따라 기밀 앱 클라이언트에 클라이언트 암호를 할당합니다. 앱 클라이언트에 클라이언트 암호를 할당할 때, Amazon Cognito 사용자 풀 API 요청에는 클라이언트 암호가 포함된 해시가 요청 본문에 포함되어야 합니다. 다음 목록의 API 작업에 대한 클라이언트 암호를 알고 있는지 확인하려면, 클라이언트 암호를 앱 클라이언트 ID 및 사용자의 사용자 이름과 연결한 다음 문자열을 base64로 인코딩합니다.

앱이 암호 해시가 있는 클라이언트에 사용자를 로그인시키는 경우 모든 사용자 풀 로그인 속성의 값을 암호 해시의 사용자 이름 요소로 사용할 수 있습니다. 앱이 `REFRESH_TOKEN_AUTH`가 있는 인증 작업에서 새 토큰을 요청하는 경우 사용자 이름 요소의 값은 로그인 속성에 따라 달라집니다. 사용자 풀에 로그인 속성으로 `username`이 없는 경우 액세스 또는 ID 토큰의 사용자 `sub` 클레임에서 암호 해시 사용자 이름 값을 설정하십시오. `username`가 로그인 속성인 경우에는 `username` 클레임에서 암호 해시 사용자 이름 값을 설정합니다.

다음 Amazon Cognito 사용자 풀 API는 `SecretHash` 파라미터의 클라이언트 암호 해시 값을 허용합니다.

- ConfirmForgotPassword
- ConfirmSignUp
- ForgotPassword
- ResendConfirmationCode
- SignUp

또한 다음 API는 인증 파라미터 또는 챌린지 응답의`SECRET_HASH` 파라미터에 클라이언트 암호 해시 값을 허용합니다.

API 작업	SECRET_HASH의 상위 파라미터
InitiateAuth	AuthParameters
AdminInitiateAuth	AuthParameters
RespondToAuthChallenge	ChallengeResponses
AdminRespondToAuthChallenge	ChallengeResponses

암호 해시 값은 사용자 풀 클라이언트 및 사용자 이름의 보안 암호 키와 메시지의 클라이언트 ID를 사용하여 계산된 Base 64 인코딩 키 추가 해시 메시지 인증 코드(HMAC)입니다. 다음 유사 코드는 이 값이 계산되는 방식을 보여줍니다. 이 유사 코드에서 +는 연결을 나타내고, HMAC_SHA256은 HmacSHA256을 사용해 HMAC 값을 생성하는 기능을 나타내며, Base64는 해시 출력의 Base-64 인코딩 버전을 생성하는 기능을 나타냅니다.

```
Base64 ( HMAC_SHA256 ( "Client Secret Key", "Username" + "Client Id" ) )
```

SecretHash파라미터를 계산하고 사용하는 방법에 대한 자세한 개요는 Amazon Cognito 사용자 풀 API에서 “클라이언트의 보안 해시를 확인할 수 없음” 오류 문제를 해결하려면 어떻게 해야 합니까? <client-id>를 참조하십시오. 지식 센터에서. AWS

서버 측 앱 코드에서 다음 코드 예를 사용할 수 있습니다.

Shell

```
echo -n "[username][app client ID]" | openssl dgst -sha256 -hmac [app client secret]
 -binary | openssl enc -base64
```

Java

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

public static String calculateSecretHash(String userPoolClientId, String
 userPoolClientSecret, String userName) {
    final String HMAC_SHA256_ALGORITHM = "HmacSHA256";

    SecretKeySpec signingKey = new SecretKeySpec(
            userPoolClientSecret.getBytes(StandardCharsets.UTF_8),
            HMAC_SHA256_ALGORITHM);
    try {
        Mac mac = Mac.getInstance(HMAC_SHA256_ALGORITHM);
        mac.init(signingKey);
        mac.update(userName.getBytes(StandardCharsets.UTF_8));
        byte[] rawHmac =
 mac.doFinal(userPoolClientId.getBytes(StandardCharsets.UTF_8));
        return Base64.getEncoder().encodeToString(rawHmac);
    } catch (Exception e) {
        throw new RuntimeException("Error while calculating ");
    }
```

```
}
```

Python

```
import sys
import hmac, hashlib, base64
username = sys.argv[1]
app_client_id = sys.argv[2]
key = sys.argv[3]
message = bytes(sys.argv[1]+sys.argv[2],'utf-8')
key = bytes(sys.argv[3],'utf-8')
secret_hash = base64.b64encode(hmac.new(key, message,
 digestmod=hashlib.sha256).digest()).decode()
print("SECRET HASH:",secret_hash)
```

이메일 또는 전화 번호를 확인하지 않고 사용자 계정 확인

Lambda 사전 가입 트리거를 사용하면 확인 코드를 요구하거나 이메일 또는 전화번호를 확인하지 않고도 가입 시 사용자 계정을 자동으로 확인할 수 있습니다. 이 방법으로 확인된 사용자는 코드를 받지 않고도 즉시 로그인할 수 있습니다.

이 트리거를 통해 확인된 사용자의 이메일 또는 전화 번호를 표시할 수도 있습니다.

> **Note**
>
> 이 방법은 사용자가 시작할 때 편리하지만 이메일 또는 전화 번호 중 하나 이상을 자동 확인하는 것이 좋습니다. 그렇지 않으면 사용자가 암호를 잊어 버린 경우 복구할 수 없게 될 수 있습니다.

사용자가 가입 시 확인 코드를 받아 입력하도록 요구하지 않고 Lambda 사전 가입 트리거에서 이메일 및 전화번호를 자동으로 확인하지 않으면, 해당 사용자 계정에 대해 확인된 이메일 주소 또는 전화번호가 없을 위험이 있습니다. 이 경우 사용자가 나중에 이메일 주소 또는 전화 번호를 확인할 수 있습니다. 그러나 사용자가 암호를 잊어버렸는데 확인된 이메일 주소 또는 전화번호가 없는 경우 암호 찾기 흐름에서 사용자에게 확인 코드를 보내는 데 확인된 이메일 또는 전화번호가 필요하기 때문에 사용자의 계정이 잠깁니다.

사용자가 이메일 또는 전화 번호를 변경하는 경우 확인

사용자가 앱에서 이메일 주소 또는 전화번호를 업데이트한 경우 해당 속성을 자동으로 확인하도록 사용자 풀을 구성했다면 Amazon Cognito에서 확인 코드가 포함된 메시지를 즉시 사용자에게 보냅니다. 그러면 사용자는 인증 메시지의 코드를 앱에 입력해야 합니다. 그러면 앱이 VerifyUserAttributeAPI 요청으로 코드를 제출하여 새 속성 값의 검증을 완료합니다.

사용자 풀에서 사용자가 업데이트된 이메일 주소나 전화번호를 확인할 필요가 없는 경우 Amazon Cognito는 업데이트된 `email` 또는 `phone_number` 속성 값을 즉시 변경하고 속성을 확인되지 않은 것으로 표시합니다. 사용자는 확인되지 않은 이메일이나 전화번호로 로그인할 수 없습니다. 해당 속성을 로그인 별칭으로 사용하려면 업데이트된 값의 확인을 완료해야 합니다.

사용자 풀에서 사용자가 업데이트된 이메일 주소 또는 전화번호를 확인해야 하는 경우 Amazon Cognito는 사용자가 새 속성 값을 확인할 때까지 속성을 검증하고 원래 값으로 설정합니다. 속성이 로그인에 대한 별칭인 경우 사용자는 확인을 통해 해당 속성이 새 값으로 변경될 때까지 원래 속성 값으로 로그인할 수 있습니다. 사용자가 업데이트된 속성을 확인하도록 사용자 풀을 구성하는 방법에 대한 자세한 내용은 이메일 또는 전화 확인 구성을 참조하세요.

사용자 지정 메시지 Lambda 트리거를 사용하여 확인 메시지를 사용자 지정할 수 있습니다. 자세한 정보는 사용자 정의 메시지 Lambda 트리거을 참조하세요. 사용자의 이메일 주소나 전화번호가 확인되지 않으면 앱에서 속성을 확인해야 함을 사용자에게 알리고, 사용자가 새 이메일 주소나 전화번호를 확인할 수 있는 버튼이나 링크가 제공됩니다.

관리자 또는 개발자가 생성한 사용자 계정에 대한 확인 및 검증 프로세스

관리자 또는 개발자가 만든 사용자 계정은 이미 확인 상태이므로 사용자가 확인 코드를 입력할 필요가 없습니다. Amazon Cognito 서비스가 이러한 사용자에게 보내는 초대 메시지에는 사용자 이름과 임시 암호가 포함됩니다. 사용자는 로그인하기 전에 이 암호를 변경해야 합니다. 자세한 내용은 관리자로 사용자 계정 생성의 이메일 및 SMS 메시지 사용자 정의 섹션과 Lambda 트리거를 사용하여 사용자 풀 워크플로 사용자 정의의 사용자 지정 메시지 트리거 섹션을 참조하세요.

가져온 사용자 계정에 대한 확인 및 검증 프로세스

AWS Management Console, CLI 또는 API (참조CSV 파일에서 사용자 풀로 사용자 가져오기) 의 사용자 가져오기 기능을 사용하여 생성한 사용자 계정은 이미 확인된 상태이므로 사용자가 확인 코드를 입력할 필요가 없습니다. 초대 메시지가 전송되지 않습니다. 그러나 가져온 사용자 계정은 사용자가 먼저 `ForgotPassword` API를 호출하여 코드를 요청한 다음, `ConfirmForgotPassword` API를 호출하고 전달된 코드를 사용하여 암호를 만들어야 로그인할 수 있습니다. 자세한 내용은 가져온 사용자에게 암호 재설정 요구 섹션을 참조하세요.

사용자 계정을 가져올 때 사용자의 이메일 또는 전화 번호가 확인된 상태로 표시되어야 하므로 사용자가 로그인 시에는 확인이 필요하지 않습니다.

앱을 테스트하는 동안 이메일 전송

Amazon Cognito는 클라이언트 앱에서 사용자 풀을 위해 계정을 생성하고 관리할 때 사용자에게 이메일 메시지를 보냅니다. 이메일 확인을 요구하도록 사용자 풀을 구성하는 경우 Amazon Cognito는 다음과 같은 경우 이메일을 보냅니다.

- 사용자가 가입할 경우.
- 사용자가 이메일 주소를 업데이트할 경우.
- 사용자가 `ForgotPassword` API 작업을 호출하는 작업을 수행할 경우.
- 사용자 계정을 관리자로 생성할 경우.

이메일을 시작한 작업이 무엇인지에 따라 이메일에는 확인 코드가 들어가거나 임시 암호가 들어갑니다. 사용자는 이러한 이메일을 수신하고 메시지를 이해해야 합니다. 그렇지 않으면 로그인하여 앱을 사용하지 못할 수 있습니다.

이메일이 성공적으로 전송되고 메시지가 올바른지 확인하려면 앱에서 Amazon Cognito로부터의 이메일 전송을 시작하는 작업을 테스트합니다. 예를 들어 앱에서 가입 페이지를 사용하거나 `SignUp` API 작업을 사용하면, 테스트 이메일 주소로 등록하여 이메일을 시작할 수 있습니다. 이 방법으로 테스트하려면 다음을 기억해야 합니다.

> **중요**
>
> 이메일 주소를 사용하여 Amazon Cognito에서 이메일을 시작하는 작업을 테스트하는 경우, 가짜 이메일 주소(메일박스가 없는 이메일 주소)를 사용하지 마세요. Amazon Cognito로부터 이메일을 수신할 실제 이메일 주소를 사용하여 하드 바운스를 유발하지 않도록 합니다.
> 하드 바운스 메일은 Amazon Cognito가 수신자의 메일박스로 이메일을 전송하지 못할 때 발생하며, 메일박스가 없는 경우 항상 발생합니다.
> Amazon Cognito는 하드 바운스가 지속적으로 발생하는 AWS 계정에서 보낼 수 있는 이메일 수를 제한합니다.

이메일을 시작하는 작업을 테스트할 때 다음 이메일 주소 중 하나를 사용하여 하드 바운스를 방지합니다.

- 자신이 소유하고 테스트에 사용하는 이메일 계정의 주소. 자신의 이메일 주소를 사용하면 Amazon Cognito에서 보내는 이메일을 받게 됩니다. 이 이메일을 통해 인증 코드를 사용하여 앱의 가입 환경을 테스트할 수 있습니다. 사용자 풀에 대한 이메일 메시지를 사용자 지정한 경우 사용자 지정 내용이 올바른지 확인할 수 있습니다.
- 메일박스 시뮬레이터 주소, success@simulator.amazonses.com. 시뮬레이터 주소를 사용하는 경우 Amazon Cognito가 이메일을 성공적으로 전송하지만 사용자는 이메일을 볼 수 없습니다. 이 옵션은 인증 코드를 사용할 필요가 없고 이메일 메시지를 확인할 필요가 없을 때 유용합니다.
- 임의의 레이블이 추가된 메일박스 시뮬레이터 주소, 예: success+user1@simulator.amazonses.com 또는 success+user2@simulator.amazonses.com. Amazon Cognito가 이들 주소로 이메일을 성공적으로 전송하지만 사용자는 전송된 이메일을 볼 수 없습니다. 이 옵션은 사용자 풀에 여러 테스트 사용자를 추가하여 가입 프로세스를 테스트하며, 각 테스트 사용자에게 고유한 이메일 주소가 있는 경우에 유용합니다.

이메일 또는 전화 확인 구성

메시징 탭에서 이메일 또는 전화 확인 설정을 선택할 수 있습니다. 멀티 팩터 인증(MFA)에 대한 자세한 내용은 SMS Text Message MFA(SMS 문자 메시지 MFA)를 참조하세요.

Amazon Cognito는 Amazon SNS를 사용하여 SMS 메시지를 전송합니다. AWS 서비스 이전에 Amazon Cognito 또는 다른 곳에서 SMS 메시지를 보낸 적이 없는 경우 Amazon SNS에서 계정을 SMS 샌드박스에 배치할 수 있습니다. 샌드박스에서 프로덕션으로 계정을 제거하기 전에 확인된 전화번호로 테스트 메시지를 보내는 것이 좋습니다. 또한 SMS 메시지를 미국 대상 전화 번호로 보내려는 경우 Amazon Pinpoint에서 발신 또는 발신자 ID를 받아야 합니다. SMS 메시지를 사용하도록 Amazon Cognito 사용자 풀을 구성하려면 Amazon Cognito 사용자 풀의 SMS 메시지 설정 섹션을 참조하세요.

Amazon Cognito는 이메일 주소 또는 전화번호를 자동으로 확인할 수 있습니다. 이 확인 작업을 수행하기 위해 Amazon Cognito에서는 확인 코드를 보내거나 확인 링크를 보냅니다. 이메일 주소의 경우 Amazon Cognito에서 코드 또는 링크를 이메일 메시지에 넣어 보냅니다. Amazon Cognito 콘솔의 메시징 탭에서 인증 메시지 템플릿을 편집할 때 인증 유형의 코드 또는 링크를 선택할 수 있습니다. 자세한 정보는 이메일 확인 메시지 사용자 정의을 참조하세요.

전화번호의 경우 Amazon Cognito에서 코드를 SMS 문자 메시지에 넣어 보냅니다.

Amazon Cognito는 사용자를 확인하고 사용자가 잊어버린 암호를 복구할 수 있도록 전화 번호 또는 이메일 주소를 확인해야 합니다. 또는 사전 가입 Lambda 트리거를 사용하여 사용자를 자동으로 확인하거나 API 작업을 사용할 수 있습니다. AdminConfirmSignUp 자세한 정보는 사용자 계정 가입 및 확인을 참조하세요.

확인 코드나 링크는 24시간 동안 유효합니다.

이메일 주소 또는 전화 번호를 확인하도록 선택하면 사용자가 가입할 때 Amazon Cognito에서 확인 코드 또는 링크를 자동으로 보냅니다. 사용자 풀에 사용자 지정 SMS 발신자 Lambda 트리거 또는 사용자 정의 이메일 발신자 Lambda 트리거가 구성된 경우 해당 함수가 대신 호출됩니다.

참고

- Amazon SNS는 전화 번호 확인을 위해 사용하는 SMS 문자 메시지 사용 요금을 별도로 청구합니다. 이메일 메시지는 무료로 보낼 수 있습니다. Amazon SNS 요금에 대한 자세한 내용은 전 세계 SMS 요금을 참조하세요. 현재 SMS 메시징을 사용할 수 있는 국가의 목록은 지원되는 리전 및 국가를 참조하세요.
- Amazon Cognito에서 이메일 메시지를 생성하는 작업을 앱에서 테스트하는 경우 Amazon Cognito에서 하드 바운스 없이 연결할 수 있는 실제 이메일 주소를 사용합니다. 자세한 내용은 the section called “앱을 테스트하는 동안 이메일 전송” 섹션을 참조하세요.
- 암호 찾기 흐름에서는 사용자를 확인하기 위해 사용자의 이메일 또는 전화 번호를 요구합니다.

Important

사용자가 전화번호 및 이메일 주소 둘 다를 사용하여 가입했으며 사용자 풀 설정에서 이 두 속성의 확인을 요구하는 경우 Amazon Cognito에서 확인 코드를 SMS 메시지를 통해 전화번호로 보냅니다. Amazon Cognito는 아직 이메일 주소를 확인하지 않았으므로 앱에서 전화를 걸어 GetUser이메일 주소가 확인을 기다리고 있는지 확인해야 합니다. 확인이 필요한 경우 앱이 전화를 걸어 이메일 확인 GetUserAttributeVerificationCode흐름을 시작해야 합니다. 그런 다음 전화를 걸어 VerifyUserAttribute확인 코드를 제출해야 합니다.

개별 메시지의 SMS 메시지 지출 할당량을 조정할 수 있습니다. AWS 계정 이러한 한도는 SMS 메시지 전송 비용에만 적용됩니다. 자세한 내용은 Amazon SNS FAQ의 계정 수준 및 메시지 수준의 지출 할당량이란 무엇이며 어떻게 작동합니까?를 참조하세요.

Amazon Cognito는 사용자 풀을 생성한 AWS 리전 위치 또는 다음 표의 기존 Amazon SNS 대체 지역 중 하나에 있는 Amazon SNS 리소스를 사용하여 SMS 메시지를 전송합니다. 아시아 태평양(서울) 리전에 있는 Amazon Cognito 사용자 풀은 예외입니다. 이러한 사용자 풀은 Amazon SNS 구성을 아시아

태평양(도쿄) 리전에서 사용합니다. 자세한 내용은 Amazon SNS SMS AWS 리전 메시지용 선택 섹션을 참조하세요.

Amazon Cognito 리전	레거시 Amazon SNS 대체 리전
미국 동부(오하이오)	미국 동부(버지니아 북부)
아시아 태평양(뭄바이)	아시아 태평양(싱가포르)
아시아 태평양(서울)	아시아 태평양(도쿄)
캐나다(중부)	미국 동부(버지니아 북부)
유럽(프랑크푸르트)	유럽(아일랜드)
유럽(런던)	유럽(아일랜드)

예: Amazon Cognito 사용자 풀이 아시아 태평양(뭄바이)에 있고 ap-southeast-1의 지출 한도가 늘어난 경우 ap-southeast-1의 별도 증가를 요청하지 않을 수 있습니다. 대신 Amazon SNS 리소스를 아시아 태평양(싱가포르)에서 사용할 수 있습니다.

이메일 주소 및 전화번호에 대한 업데이트 확인

이메일 주소 또는 전화번호 속성은 사용자가 값을 변경한 직후에 미확인 상태로 활성화됩니다. 또한 Amazon Cognito에서 속성을 업데이트하기 전에 사용자가 새 값을 확인하도록 요구할 수도 있습니다. 사용자가 먼저 새 값을 확인하도록 요청하는 경우 원래 값을 사용하여 로그인하고 새 값을 확인할 때까지 메시지를 수신할 수 있습니다.

사용자가 이메일 주소 또는 전화번호를 사용자 풀의 로그인 별칭으로 사용할 수 있는 경우 업데이트된 속성의 로그인 이름은 업데이트된 속성의 확인이 필요한지에 따라 달라집니다. 업데이트된 속성을 확인해야 하는 경우 사용자는 새 값을 확인할 때까지 원래 속성 값으로 로그인할 수 있습니다. 사용자가 업데이트된 속성을 확인하지 않아도 되는 경우 사용자는 새 값을 확인할 때까지 새 속성 값이나 원래 속성 값으로 로그인하거나 메시지를 받을 수 없습니다.

예를 들어, 사용자 풀은 이메일 주소 별칭으로 로그인을 허용하고 사용자가 업데이트할 때 이메일 주소를 확인해야 합니다. `sue@example.com`으로 로그인하는 Sue는 이메일 주소를 `sue2@example.com`으로 바꾸기를 원했지만, 실수로 **s**`sue2@example.com`을 입력했습니다. Sue는 확인 이메일을 받지 못하니 `ssue2@example.com`을 확인할 수가 없습니다. Sue는 `sue@example.com`으로 로그인한 후 앱에서 양식을 다시 제출하여 이메일 주소를

sue2@example.com으로 업데이트합니다. 확인 이메일을 받고 앱에 확인 코드를 입력한 그녀는 이제 sue2@example.com으로 로그인하기 시작합니다.

사용자가 속성을 업데이트하고 사용자 풀이 새 속성 값을 검증하는 경우

- 새 값을 검증하기 위한 코드를 확인하기 전에 원래 속성 값으로 로그인할 수 있습니다.
- 새 값을 검증하기 위한 코드를 확인한 후에만 새 속성 값으로 로그인할 수 있습니다.
- AdminUpdateUserAttributesAPI 요청에 email_verified 또는 phone_number_verified 를 true 설정하면 Amazon Cognito가 보낸 코드를 확인하기 전에 로그인할 수 있습니다.

사용자가 속성을 업데이트하고 사용자 풀이 새 속성 값을 검증하지 않는 경우

- 원래 속성 값으로 로그인하거나 메시지를 수신할 수 없습니다.
- 새 값을 검증하기 위한 코드를 확인하기 전에는 새 속성 값으로 로그인하거나 확인 코드 이외의 메시지를 받을 수 없습니다.
- AdminUpdateUserAttributesAPI 요청에 email_verified 또는 phone_number_verified 를 true 설정하면 Amazon Cognito가 보낸 코드를 확인하기 전에 로그인할 수 있습니다.

사용자가 이메일 주소 또는 전화번호를 업데이트할 때 속성 확인을 요청하려면

1. Amazon Cognito 콘솔에 로그인합니다. 메시지가 표시되면 AWS 자격 증명을 입력합니다.
2. 탐색 창에서 [사용자 풀(User Pools)]을 선택한 다음 편집할 사용자 풀을 선택합니다.
3. 가입 환경(Sign-up experience) 탭의 속성 확인 및 사용자 계정 확인(Attribute verification and user account confirmation)에서 편집(Edit)을 선택합니다.
4. 업데이트가 보류 중일 때 원래 속성 값을 활성 상태로 유지(Keep original attribute value active when an update is pending)를 선택합니다.
5. 업데이트가 보류 중일 때 활성 속성 값(Active attribute values when an update is pending)에서 Amazon Cognito가 값을 업데이트하기 전에 사용자가 확인해야 할 속성을 선택합니다.
6. 변경 사항 저장를 선택합니다.

Amazon Cognito API를 사용하여 속성 업데이트 확인을 요구하려면 요청에서 AttributesRequireVerificationBeforeUpdate 파라미터를 설정할 수 있습니다. UpdateUserPool

Amazon Cognito에 자동으로 SMS 메시지를 전송할 수 있는 권한 부여

Amazon Cognito가 자동으로 사용자에게 SMS 메시지를 보내려면 권한이 필요합니다. AWS Identity and Access Management (IAM) 역할을 생성하여 해당 권한을 부여할 수 있습니다. Amazon Cognito 콘솔의 메시징 탭에서 SMS에서 편집을 선택하여 역할을 설정합니다.

SMS 및 이메일 확인 메시지와 사용자 초대 메시지 구성

Amazon Cognito를 사용하면 SMS 및 이메일 확인 메시지와 사용자 초대 메시지를 사용자 지정하여 애플리케이션의 보안 및 사용자 환경을 개선할 수 있습니다. Amazon Cognito를 사용하면 애플리케이션 요구 사항에 맞게 코드 기반 또는 원클릭 링크 인증 중에서 선택할 수 있습니다. 이 주제에서는 Amazon Cognito 콘솔에서 멀티 팩터 인증 (MFA) 및 검증 통신을 개인화하는 방법을 설명합니다.

메시지 템플릿 아래의 메시지 탭에서 다음을 사용자 지정할 수 있습니다.

- SMS 문자 메시지 멀티 팩터 인증(MFA) 메시지
- SMS 및 이메일 확인 메시지
- 이메일의 확인 유형(코드 또는 링크)
- 사용자 초대 메시지
- 사용자 풀을 통해 전송되는 이메일의 FROM 및 REPLY-TO 이메일 주소

> **Note**
>
> SMS 및 이메일 확인 메시지 템플릿은 [확인(Verifications)] 탭에서 전화 번호 및 이메일 확인을 요구하도록 선택한 경우에만 표시됩니다. 마찬가지로, SMS MFA 메시지 템플릿은 MFA 설정이 [필수(required)] 또는 [선택 사항(optional)]인 경우에만 표시됩니다.

주제

메시지 템플릿

메시지 템플릿을 사용하여 해당 값이 대체하는 자리 표시자를 사용하여 메시지에 필드를 삽입할 수 있습니다.

템플릿 자리 표시자

설명	토큰
확인 코드	`{####}`
임시 암호	`{####}`
사용자 이름	`{username}`

> **Note**
>
> `{username}` 자리 표시자는 확인 이메일 메시지에 사용할 수 없습니다. 작업과 함께 생성하는 초대 이메일 메시지에 `{username}` 자리 표시자를 사용할 수 있습니다. AdminCreateUser 이러한 초대 이메일 메시지에는 두 개의 자리 표시자인 사용자 이름(`{username}`) 및 임시 암호(`{####}`)를 사용합니다.

고급 보안 템플릿 자리 표시자를 사용하여 다음을 수행할 수 있습니다.

- IP 주소, 도시, 국가, 로그인 시간 및 디바이스 이름과 같은 이벤트에 대한 특정 세부 정보를 포함합니다. Amazon Cognito 고급 보안 기능은 이러한 세부 정보를 분석할 수 있습니다.
- 원클릭 링크가 유효한지 확인할 수 있습니다.
- 이벤트 ID, 피드백 토큰, 사용자 이름을 사용하여 자체 원클릭 링크를 작성합니다.

> **Note**
>
> 원클릭 링크를 생성하고 고급 보안 이메일 템플릿에서 `{one-click-link-valid}` 및 `{one-click-link-invalid}` 자리 표시자를 사용하려면 이미 사용자 풀에 대한 도메인이 구성되어 있어야 합니다.

고급 보안 템플릿 자리 표시자

설명	토큰
IP 주소	`{ip-address}`
구/군/시	`{city}`
국가	`{country}`
로그인 시간	`{login-time}`
디바이스 이름	`{device-name}`
원클릭 링크가 유효합니다.	`{one-click-link-valid}`
원클릭 링크가 유효하지 않습니다.	`{one-click-link-invalid}`
이벤트 ID	`{event-id}`
피드백 토큰	`{feedback-token}`

SMS 메시지 사용자 정의

> **Note**
>
> 새 Amazon Cognito 콘솔 환경에서는 SMS 메시지를 사용자 지정할 수 있습니다.

메시징 탭의 메시지 템플릿 아래에서 다중 인증(MFA)에 대한 SMS 메시지를 사용자 지정할 수 있습니다.

> **Important**
>
> 사용자 정의 메시지에는 `{####}` 자리 표시자가 포함되어야 합니다. 이 자리 표시자는 메시지를 보내기 전에 인증 코드로 대체됩니다.

Amazon Cognito에서는 인증 코드를 포함한 SMS 메시지의 최대 길이가 UTF-8 140자로 제한됩니다.

SMS 확인 메시지 사용자 정의

[SMS 확인 메시지를 사용자 지정하시겠습니까?(Do you want to customize your SMS verification messages?)]라는 머리글 아래의 템플릿을 편집하여 전화 번호 확인을 위한 SMS 메시지를 사용자 지정할 수 있습니다.

> **Important**
>
> 사용자 정의 메시지에는 {####} 자리 표시자가 포함되어야 합니다. 이 자리 표시자는 메시지를 보내기 전에 확인 코드로 대체됩니다.

메시지의 최대 길이는 확인 코드를 포함하여 140자(UTF-8)입니다.

이메일 확인 메시지 사용자 정의

Amazon Cognito를 사용하여 사용자 풀에 있는 사용자의 이메일 주소를 확인하려면 사용자가 선택할 수 있는 링크가 포함된 이메일 메시지를 사용자에 보내거나 사용자가 입력할 수 있는 코드를 사용자에게 보냅니다.

이메일 주소 인증 메시지의 이메일 제목 및 메시지 콘텐츠를 사용자 지정하려면 사용자 풀의 메시지 탭에서 인증 메시지 템플릿을 편집합니다. 인증 메시지 템플릿을 편집할 때 인증 유형의 코드 또는 링크를 선택할 수 있습니다.

확인 유형으로 코드를 선택한 경우 사용자 지정 메시지에 {####} 자리 표시자가 포함되어야 합니다. 메시지를 전송할 때 이 자리 표시자는 확인 코드로 대체됩니다.

확인 유형으로 링크를 선택한 경우 사용자 지정 메시지에 {##Verify Your Email##} 형식의 자리 표시자가 포함되어야 합니다. 예를 들어 {##Click here##} 같이 자리 표시자 문자 사이의 텍스트 문자열을 변경할 수 있습니다. 이 자리 표시자는 이메일 확인(Verify Your Email)이라는 확인 링크로 대체됩니다.

이메일 확인 메시지에 대한 링크를 클릭하면 다음 예와 같은 URL로 이동합니다.

```
https://<your user pool domain>/confirmUser/?
client_id=abcdefg12345678&user_name=emailtest&confirmation_code=123456
```

메시지의 최대 길이는 확인 코드(있는 경우)를 포함하여 20,000자(UTF-8)입니다. 이 메시지에 HTML 태그를 사용하여 내용을 포맷할 수 있습니다.

사용자 초대 메시지 사용자 정의

메시지 탭의 초대 메시지 템플릿을 편집하여 Amazon Cognito가 SMS 또는 이메일 메시지를 통해 새 사용자에게 보내는 사용자 초대 메시지를 사용자 정의할 수 있습니다.

> **Important**
>
> 사용자 정의 메시지에는 `{username}` 및 `{####}` 자리 표시자가 포함되어야 합니다. Amazon Cognito는 초대 메시지를 보낼 때 이러한 자리 표시자를 사용자의 사용자 이름과 암호로 대체합니다.

SMS 메시지의 최대 길이는 확인 코드를 포함하여 140자(UTF-8)입니다. 이메일 메시지의 최대 길이는 확인 코드를 포함하여 20,000자(UTF-8)입니다. 이메일 메시지에 HTML 태그를 사용하여 내용을 포맷할 수 있습니다.

이메일 주소 사용자 정의

기본값으로 Amazon Cognito는 사용자 풀의 사용자에게 no-reply@verificationemail.com 주소에서 이메일 메시지를 전송합니다. no-reply@verificationemail.com 대신 사용자 정의 FROM 및 REPLY-TO 이메일 주소를 지정할 수 있습니다.

FROM 및 REPLY-TO 이메일 주소를 사용자 정의하려면

1. Amazon Cognito 콘솔로 이동하여, [사용자 풀(User Pools)]을 선택합니다.
2. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
3. [메시징(Messaging)] 탭을 선택합니다. [이메일(Email)]에서 [편집(Edit)]을 선택합니다.
4. SES 리전(SES Region)을 선택합니다.
5. 선택한 SES 리전(SES Region)의 Amazon SES로 확인한 이메일 주소 목록에서 발신 이메일 주소(FROM email address)를 선택합니다. 확인된 도메인의 이메일 주소를 사용하려면 AWS Command Line Interface 또는 AWS API에서 이메일 설정을 구성합니다. 자세한 내용은 Amazon Simple Email Service 개발자 가이드에서 Amazon SES에서 이메일 주소 및 도메인 확인을 참조하세요.
6. 선택한 [SES 리전(SES Region)]의 구성 집합 목록에서 [구성 집합(Configuration set)]을 선택합니다.
7. 이메일 메시지의 친숙한 [FROM 발신자 이름(FROM sender name)]을 `John Stiles <johnstiles@example.com>` 포맷으로 입력합니다.

8. 회신 이메일 주소를 사용자 지정하려면 회신 이메일 주소 필드에 유효한 이메일 주소를 입력합니다.

Amazon Cognito가 사용자를 대신하여 (사용자 정의 FROM 이메일 주소에서) Amazon SES 이메일을 전송하도록 권한 부여

기본 주소 대신 사용자 정의 FROM 이메일 주소에서 이메일을 전송하도록 Amazon Cognito를 구성할 수 있습니다. 사용자 정의 주소를 사용하려면 Amazon SES 확인 자격 증명에서 이메일 메시지를 전송할 수 있는 권한을 Amazon Cognito에 부여해야 합니다. 대부분의 경우에 전송 권한 부여 정책을 생성하여 권한을 부여할 수 있습니다. 자세한 내용은 Amazon Simple Email Service 개발자 가이드에서 Amazon SES에서 전송 권한 부여 사용을 참조하세요.

이메일 메시지에 Amazon SES를 사용하도록 사용자 풀을 구성하면 Amazon Cognito는 계정에 `AWSServiceRoleForAmazonCognitoIdpEmailService` 역할을 생성하여 Amazon SES에 대한 액세스 권한을 부여합니다. `AWSServiceRoleForAmazonCognitoIdpEmailService` 서비스 연결 역할을 사용하는 경우에는 전송 권한 부여 정책이 필요하지 않습니다. 사용자 풀의 기본 이메일 기능 및 FROM 주소로 확인된 Amazon SES 자격 증명을 둘 다 사용하는 경우에만 전송 권한 부여 정책을 추가해야 합니다.

Amazon Cognito가 생성하는 서비스 연결 역할에 대한 자세한 내용은 Amazon Cognito에 서비스 연결 역할 사용 섹션을 참조하세요.

다음 전송 권한 부여 정책 예제는 Amazon Cognito에 Amazon SES 확인 자격 증명을 사용할 수 있는 제한적인 권한을 부여합니다. Amazon Cognito는 `aws:SourceArn` 조건의 사용자 풀과 `aws:SourceAccount` 조건의 계정을 둘 다 대신하는 경우에만 이메일 메시지를 전송할 수 있습니다. 자세한 내용은 Amazon Simple Email Service 개발자 가이드에서 Amazon SES 전송 권한 부여 정책 예제를 참조하세요.

> **Note**
>
> 이 예제에서 "Sid" 값은 명령문을 고유하게 식별하는 임의 문자열입니다. 정책 구문에 대한 자세한 내용은 Amazon Simple Email Service 개발자 가이드에서 Amazon SES 전송 권한 부여 정책을 참조하세요.

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

```
            "Sid": "stmnt1234567891234",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "email.cognito-idp.amazonaws.com"
                ]
            },
            "Action": [
                "SES:SendEmail",
                "SES:SendRawEmail"
            ],
            "Resource": "<your SES identity ARN>",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "<your account number>"
                },
                "ArnLike": {
                    "aws:SourceArn": "<your user pool ARN>"
                }
            }
        }
    ]
}
```

드롭다운 메뉴에서 Amazon SES 자격 증명을 선택하면 Amazon Cognito 콘솔이 유사한 정책을 자동으로 추가합니다. CLI 또는 API를 사용하여 사용자 풀을 구성하는 경우 이전 예제와 같이 구성된 정책을 Amazon SES 자격 증명에 연결해야 합니다.

관리자로 사용자 계정 생성

사용자 풀을 생성한 후 AWS Command Line Interface 또는 Amazon Cognito API 뿐만 아니라 AWS Management Console을 사용하여 사용자를 생성할 수 있습니다. 사용자 풀에서 새 사용자에 대한 프로파일을 생성하고, SMS 또는 이메일을 통해 사용자에게 가입 지침이 포함된 환영 메시지를 전송할 수 있습니다.

개발자 및 관리자는 다음 작업을 수행할 수 있습니다.

- AWS Management Console을 사용하거나 `AdminCreateUser` API를 호출하여 새 사용자 프로파일을 생성합니다.
- 사용자 속성 값을 설정합니다.
- 사용자 지정 속성을 생성합니다.

- `AdminCreateUser` API 요청에서 변경할 수 없는 사용자 지정 속성의 값을 설정합니다. 이 기능은 Amazon Cognito 콘솔에서 사용할 수 없습니다.
- 임시 암호를 지정하거나 Amazon Cognito가 암호를 자동으로 생성하도록 합니다.
- 제공된 이메일 주소 및 전화 번호가 새 사용자에 대해 확인됨으로 표시되는지 여부를 지정합니다.
- AWS Management Console 또는 사용자 지정 메시지 Lambda 트리거를 통해 새 사용자에 대한 사용자 지정 SMS 및 이메일 초대 메시지를 지정합니다. 자세한 내용은 Lambda 트리거를 사용하여 사용자 풀 워크플로 사용자 정의 섹션을 참조하세요.
- 초대 메시지가 SMS, 이메일 또는 둘 다를 통해 전송되는지 여부를 지정합니다.
- `AdminCreateUser` API를 호출하고 `RESEND` 파라미터에 대해 `MessageAction`를 지정하여 기존 사용자에게 환영 메시지를 재전송합니다.

> **Note**
>
> 이 작업은 현재 AWS Management Console을 사용하여 수행할 수 없습니다.

- 사용자가 생성될 때 초대 메시지의 전송을 제한합니다.
- 사용자 계정에 대한 만료 시간 제한을 지정합니다(최대 90일).
- 사용자가 직접 가입하거나 관리자가 새 사용자만 추가하도록 합니다.

관리자 또는 개발자가 생성한 사용자에 대한 인증 흐름

이러한 사용자에 대한 인증 흐름에는 새 암호를 제출하고 필수 속성에 대해 누락된 값을 제공하는 추가 단계가 포함되어 있습니다. 이러한 단계는 다음에 설명하며 5, 6 및 7단계가 이러한 사용자에 적용됩니다.

1. 사용자가 사용자 이름 및 암호를 제출하여 첫 로그인을 시작합니다.
2. SDK는 `InitiateAuth(Username, USER_SRP_AUTH)`를 호출합니다.
3. Amazon Cognito는 솔트 및 암호 블록이 있는 `PASSWORD_VERIFIER` 문제를 반환합니다.
4. SDK는 SRP 계산을 수행하며 `RespondToAuthChallenge(Username, <SRP variables>, PASSWORD_VERIFIER)`를 호출합니다.
5. Amazon Cognito는 `NEW_PASSWORD_REQUIRED` 챌린지를 반환합니다. 이 챌린지의 본문에는 사용자의 현재 속성과 현재 사용자 프로파일에 값이 없는 사용자 풀 내 필수 속성이 포함되어 있습니다. 자세한 내용은 RespondToAuthChallenge를 참조하세요.
6. 메시지가 표시되면 사용자는 새 암호 및 필수 속성에 대해 누락된 값을 입력합니다.

7. SDK는 `RespondToAuthChallenge(Username, <New password>, <User attributes>)`를 호출합니다.
8. 사용자에게 MFA에 대한 두 번째 요소가 필요한 경우 Amazon Cognito는 SMS_MFA 문제를 반환하고 코드가 제출됩니다.
9. 사용자가 암호와 선택 항목으로 제공된 속성 값 또는 완료된 MFA를 성공적으로 변경하면 사용자가 로그인되고 토큰이 발급됩니다.

사용자가 모든 문제에 대해 만족하면 Amazon Cognito 서비스는 사용자를 확인됨으로 표시하고 해당 사용자에게 ID, 액세스 및 새로 고침 토큰을 발급합니다. 자세한 내용은 사용자 풀에 토큰 사용 섹션을 참조하세요.

AWS Management Console에서 새 사용자 생성

Amazon Cognito 콘솔을 사용하여 사용자 암호 요구 사항을 설정하고, 사용자에게 전송되는 초대 및 확인 메시지를 구성하고, 새 사용자를 추가할 수 있습니다.

암호 정책 설정 및 자체 등록 사용

최소 암호 복잡성에 대한 설정과 사용자가 퍼블릭 API를 사용하여 사용자 풀에 가입할 수 있는지 여부를 구성할 수 있습니다.

암호 정책 구성

1. Amazon Cognito 콘솔로 이동하여, [사용자 풀(User Pools)]을 선택합니다.
2. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
3. [로그인 환경(Sign-in experience)] 탭을 선택하고 [암호 정책(Password policy)]을 찾습니다. 편집을 선택합니다.
4. [암호 정책 모드(Password policy mode)]로 [사용자 정의(Custom)]를 선택합니다.
5. [암호 최소 길이(Password minimum length)]를 선택합니다. 암호 길이 요구 사항에 대한 제한은 사용자 풀 리소스 할당량을 참조하세요.
6. [암호 복잡성(Password complexity)] 요구 사항을 선택합니다.
7. 관리자가 설정한 암호가 유효한 기간을 선택합니다.
8. 변경 사항 저장(Save changes)을 선택합니다.

셀프 서비스 가입 허용

1. Amazon Cognito 콘솔로 이동하여, [사용자 풀(User Pools)]을 선택합니다.

2. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
3. [가입 환경(Sign-up experience)] 탭을 선택하고 [셀프 서비스 가입(Self-service sign-up)]을 찾습니다. [편집(Edit)]을 선택합니다.
4. 자체 등록 사용(Enable self-registration) 여부를 선택합니다. 자체 등록은 일반적으로 클라이언트 암호 또는 AWS Identity and Access Management(IAM) API 자격 증명을 배포하지 않고 사용자 풀에 새 사용자를 등록해야 하는 퍼블릭 앱 클라이언트에서 사용됩니다.

   > **자체 등록 사용 중지**
   >
   > 자체 등록을 사용하지 않는 경우 IAM API 자격 증명을 사용하는 관리 API 작업 또는 페더레이션 공급자를 사용한 로그인을 통해 새 사용자를 생성해야 합니다.

5. 변경 사항 저장(Save changes)을 선택합니다.

이메일 및 SMS 메시지 사용자 정의

사용자 메시지 사용자 정의

사용자를 로그인하도록 초대하거나, 사용자가 사용자 계정에 가입하거나, 사용자가 로그인하고 멀티 팩터 인증(MFA) 메시지가 표시될 때 Amazon Cognito가 사용자에게 보내는 메시지를 사용자 정의할 수 있습니다.

> **Note**
>
> [초대 메시지(Invitation message)]는 사용자 풀에 사용자를 생성하고 로그인하도록 초대할 때 전송됩니다. Amazon Cognito는 사용자의 이메일 주소나 전화 번호로 초기 로그인 정보를 전송합니다.
> [확인 메시지(Verification message)]는 사용자가 사용자 풀의 사용자 계정에 가입할 때 전송됩니다. Amazon Cognito가 사용자에게 코드를 전송합니다. 사용자는 Amazon Cognito에 코드를 제공할 때 연락처 정보를 검증하고 로그인 계정을 확인합니다. 확인 코드는 24시간 동안 유효합니다.
> [MFA 메시지(MFA message)]는 사용자 풀에서 SMS MFA를 사용하도록 설정하고, SMS MFA를 구성한 사용자가 로그인하고 MFA 메시지가 표시될 때 전송됩니다.

1. Amazon Cognito 콘솔로 이동하여, [사용자 풀(User Pools)]을 선택합니다.
2. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.

3. [메시징(Messaging)] 탭을 선택하고 [메시지 템플릿(Message templates)]을 찾습니다. [확인 메시지(Verification messages)], [초대 메시지(Invitation messages)] 또는 [MFA 메시지(MFA messages)]를 선택한 다음 [편집(Edit)]을 선택합니다.
4. 선택한 메시지 유형에 맞게 메시지를 사용자 정의합니다.

> **Note**
>
> 메시지를 사용자 정의할 때 메시지 템플릿의 모든 변수를 포함해야 합니다. 변수(예: {####})를 포함하지 않으면 사용자에게 메시지 작업을 완료하기에 충분한 정보가 없습니다.
> 자세한 내용은 메시지 템플릿을 참조하세요.

5. a. [확인 메시지(Verification messages)]
   i. [이메일(Email)] 메시지에 대한 [확인 유형(Verification type)]을 선택합니다. [코드(Code)] 확인은 사용자가 입력해야 하는 숫자 코드를 전송합니다. [링크(Link)] 확인은 사용자가 연락처 정보를 확인하기 위해 클릭할 수 있는 링크를 전송합니다. [링크(Link)] 메시지에 대한 변수의 텍스트는 하이퍼링크 텍스트로 표시됩니다. 예를 들어 {##Click here##} 변수를 사용하는 메시지 템플릿은 이메일 메시지에 Click here로 표시됩니다.
   ii. [이메일(Email)] 메시지의 [이메일 제목(Email subject)]을 입력합니다.
   iii. [이메일(Email)] 메시지용 사용자 정의 [이메일 메시지(Email message)] 템플릿을 입력합니다. HTML을 사용하여 이 템플릿을 사용자 정의할 수 있습니다.
   iv. [SMS] 메시지용 사용자 정의 [SMS 메시지(SMS message)] 템플릿을 입력합니다.
   v. 변경 사항 저장(Save changes)을 선택합니다.

   b. [초대 메시지(Invitation messages)]
   i. [이메일(Email)] 메시지의 [이메일 제목(Email subject)]을 입력합니다.
   ii. [이메일(Email)] 메시지용 사용자 정의 [이메일 메시지(Email message)] 템플릿을 입력합니다. HTML을 사용하여 이 템플릿을 사용자 정의할 수 있습니다.
   iii. [SMS] 메시지용 사용자 정의 [SMS 메시지(SMS message)] 템플릿을 입력합니다.
   iv. 변경 사항 저장(Save changes)을 선택합니다.

   c. [MFA 메시지(MFA messages)]
   i. [SMS] 메시지용 사용자 정의 [SMS 메시지(SMS message)] 템플릿을 입력합니다.
   ii. 변경 사항 저장(Save changes)을 선택합니다.

사용자 생성

사용자 생성

Amazon Cognito 콘솔에서 사용자 풀의 새 사용자를 생성할 수 있습니다. 일반적으로 사용자는 암호를 설정한 후에 로그인할 수 있습니다. 이메일 주소를 사용하여 로그인하려면 사용자가 `email` 속성을 확인해야 합니다. 전화 번호를 사용하여 로그인하려면 사용자가 `phone_number` 속성을 확인해야 합니다. 관리자로 계정을 확인하려면 AWS CLI 또는 API를 사용하거나 페더레이션 자격 증명 공급자를 사용하여 사용자 프로필을 생성합니다. 자세한 내용은 Amazon Cognito API 참조를 참조하세요.

1. Amazon Cognito 콘솔로 이동하여, [사용자 풀(User Pools)]을 선택합니다.
2. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
3. 사용자(Users) 탭을 선택한 다음 사용자 생성(Create a user)을 선택합니다.
4. 사용자 풀에 대한 암호 요구 사항, 사용 가능한 계정 복구 방법 및 별칭 속성에 대한 지침은 사용자 풀 로그인 및 보안 요구 사항(User pool sign-in and security requirements)을 검토합니다.
5. 초청 메시지(Invitation message)를 보내는 방식을 선택합니다. SMS 메시지와 이메일 메시지 중 하나 또는 둘 다를 선택합니다.

   > **Note**
   >
   > 초청 메시지를 보내려면 먼저 사용자 풀의 메시징(Messaging) 탭에서 Amazon Simple Notification Service 및 Amazon Simple Email Service를 사용하여 발신자 및 AWS 리전을 구성합니다. 수신자 메시지 및 데이터 요금이 적용됩니다. Amazon SES에서 이메일 메시지를 별도로 청구하며, Amazon SNS에서 SMS 메시지를 별도로 청구합니다.

6. 새 사용자의 사용자 이름(Username)을 선택합니다.
7. 사용자에 대해 암호 생성(Create a password)을 직접 수행할지 아니면 Amazon Cognito에서 암호 생성(Generate a password)을 수행하도록 할지를 선택합니다. 임시 암호는 사용자 풀 암호 정책을 준수해야 합니다.
8. 생성을 선택합니다.
9. 사용자(Users) 탭을 선택하고 사용자에 대한 사용자 이름(User name) 항목을 선택합니다. 사용자 속성(User attributes) 및 그룹 멤버십(Group memberships)을 추가하고 편집합니다. 사용자 이벤트 기록(User event history)을 검토합니다.

사용자 풀에 그룹 추가

Amazon Cognito 사용자 풀에서 그룹에 대한 지원을 통해 그룹을 생성하고 관리하고, 사용자를 그룹에 추가하고, 그룹에서 사용자를 제거할 수 있습니다. 그룹을 통해 사용자 모음을 생성하여 권한을 관리하거나 다른 유형의 사용자를 표시합니다. 그룹에 AWS Identity and Access Management (IAM) 역할을 할당하여 그룹 구성원의 권한을 정의할 수 있습니다.

그룹을 사용하여 사용자 풀에서 사용자 모음을 생성할 수 있습니다. 이 작업은 해당 사용자에 대한 권한을 설정하여 수행되는 경우가 많습니다. 예를 들어, 웹 사이트 및 앱의 독자, 기고자 및 편집자에 대해 별도의 사용자 그룹을 생성할 수 있습니다. 그룹과 연관된 IAM 역할을 사용하면 Amazon S3에 기고자만 콘텐츠를 넣을 수 있으며, 편집자만 Amazon API Gateway의 API를 통해 콘텐츠를 게시할 수 있도록 다른 그룹에 대해 서로 다른 권한을 설정할 수도 있습니다.

AWS Management Console, API 및 CLI에서 사용자 풀의 그룹을 생성하고 관리할 수 있습니다. 개발자는 AWS 자격 증명을 사용하여 사용자 풀의 그룹을 만들고, 읽고, 업데이트하고, 삭제하고, 나열할 수 있습니다. 그룹에서 사용자를 추가하고 제거할 수도 있습니다.

사용자 풀 내에서 그룹 사용에 대한 추가 비용은 없습니다. 자세한 내용은 Amazon Cognito 요금을 참조하세요.

그룹에 IAM 역할 할당

그룹을 사용하면 IAM 역할을 사용하는 리소스에 대한 권한을 제어할 수 있습니다. IAM 역할에는 신뢰 정책 및 권한 정책이 포함됩니다. 역할 신뢰 정책은 역할을 사용할 수 있는 사용자를 지정합니다. 권한 정책은 그룹 구성원이 액세스할 수 있는 작업과 리소스를 지정합니다. IAM 역할을 생성하는 경우 그룹 사용자가 역할을 수임할 수 있도록 역할 트러스트 정책을 설정합니다. 역할 권한 정책에서 그룹에 부여할 권한을 지정합니다.

Amazon Cognito에서 그룹을 생성할 때 역할의 ARN을 제공하여 IAM 역할을 지정합니다. 그룹 구성원이 Amazon Cognito를 사용하여 로그인하면 자격 증명 풀에서 임시 자격 증명을 받을 수 있습니다. 해당 권한은 연결된 IAM 역할에 따라 결정됩니다.

개별 사용자가 여러 그룹에 있을 수 있습니다. 사용자가 여러 그룹에 있는 경우 개발자는 다음과 같이 IAM 역할을 자동으로 선택하기 위한 옵션을 보유합니다.

- 각 그룹에 우선 순위 값을 할당할 수 있습니다. 우선 순위가 좋은(낮은) 그룹이 선택되고 이와 연관된 IAM 역할이 적용됩니다.
- 또한 앱은 자격 증명 풀을 통해 사용자의 AWS 자격 증명을 요청할 때 파라미터에 역할 ARN을 지정하여 사용 가능한 역할 중에서 선택할 수 있습니다. GetCredentialsForIdentity`CustomRoleARN` 지정된 IAM 역할은 사용자에 대해 사용 가능한 역할과 일치해야 합니다.

그룹에 우선 순위 값 할당

사용자는 둘 이상의 그룹에 속할 수 있습니다. 사용자의 액세스 및 ID 토큰에 있는 `cognito:groups` 클레임에는 사용자가 속한 모든 그룹의 목록이 포함되어 있습니다. `cognito:roles` 클레임에는 그룹에 해당하는 역할 목록이 포함되어 있습니다.

사용자가 둘 이상의 그룹에 속할 수 있으므로 각 그룹에 우선 순위가 할당될 수 있습니다. 우선 순위는 사용자 풀에서 사용자가 속한 다른 그룹을 기준으로 이 그룹의 우선 순위를 지정하는, 음수가 아닌 숫자입니다. 0은 가장 높은 우선 순위 값입니다. 우선 순위 값이 낮은 그룹은 우선 순위 값이 높거나 null인 그룹보다 우선합니다. 사용자가 둘 이상의 그룹에 속해 있는 경우 우선 순위 값이 가장 낮은 그룹의 IAM 역할이 사용자 ID 토큰의 `cognito:preferred_role` 클레임에 적용됩니다.

두 개의 그룹에 동일한 우선 순위 값이 있을 수 있습니다. 이러한 경우 두 그룹 모두 우선 적용되지 않습니다. 우선 순위 값이 동일한 두 그룹에 동일한 역할 ARN이 있는 경우 해당 역할은 각 그룹의 사용자에 대한 ID 토큰의 `cognito:preferred_role` 클레임에 사용됩니다. 두 그룹의 역할 ARN이 서로 다른 경우 `cognito:preferred_role` 클레임은 사용자의 ID 토큰에서 설정되지 않습니다.

그룹을 사용하여 Amazon API Gateway로 권한 제어

Amazon API Gateway를 통해 사용자 풀에서 그룹을 사용하여 권한을 제어할 수 있습니다. 사용자가 속한 그룹은 `cognito:groups` 클레임에 있는 사용자 풀의 ID 토큰과 액세스 토큰에 모두 포함됩니다. 요청과 함께 ID 또는 액세스 토큰을 Amazon API Gateway에 제출하고 REST API에 Amazon Cognito 사용자 풀 권한 부여자를 사용할 수 있습니다. 자세한 내용은 API Gateway 개발자 가이드에서 Amazon Cognito 사용자 풀을 권한 부여자로 사용하여 REST API에 대한 액세스 제어를 참조하세요.

사용자 정의 JWT 권한 부여자를 사용하여 Amazon API Gateway HTTP API에 대한 액세스 권한을 부여할 수도 있습니다. 자세한 내용은 API Gateway 개발자 가이드에서 JWT 권한 부여자를 사용하여 HTTP API에 대한 액세스 제어를 참조하세요.

그룹에 대한 제한 사항

사용자 그룹에 다음 제한이 적용됩니다.

- 생성할 수 있는 그룹 수는 Amazon Cognito 서비스 할당량에 따라 제한됩니다.
- 그룹은 중첩될 수 없습니다.
- 그룹에서 사용자를 검색할 수 없습니다.
- 이름으로 그룹을 검색할 수 없지만 그룹을 나열할 수 있습니다.

AWS Management Console에서 새 그룹 생성

새 그룹을 생성하려면 다음 절차를 따르세요.

새 그룹을 생성하려면

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 자격 증명을 입력합니다. AWS
2. 사용자 풀(User Pools)을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택합니다.
4. 그룹(Groups) 탭을 선택한 다음 그룹 생성(Create a group)을 선택합니다.
5. [그룹 생성(Create a group)] 페이지의 [그룹 이름(Group name)]에 새 그룹의 친숙한 이름을 입력합니다.
6. 다음 필드를 사용하여 이 그룹에 대한 추가 정보를 선택적으로 제공할 수 있습니다.
   - [설명(Description)] - 새 그룹의 용도에 대한 세부 정보를 입력합니다.
   - [우선 순위(Precedence)] - Amazon Cognito는 우선 순위가 더 낮은 소속 그룹을 기준으로 지정된 사용자의 모든 그룹 권한을 평가하고 적용합니다. 우선 순위가 더 낮은 그룹이 선택되고 연결된 IAM 역할이 적용됩니다. 자세한 설명은 그룹에 우선 순위 값 할당 섹션을 참조하세요.
   - [IAM 역할(IAM role)] - 리소스에 대한 권한을 제어해야 하는 경우 그룹에 IAM 역할을 할당할 수 있습니다. 사용자 풀을 자격 증명 풀과 통합한 경우 IAM 역할 설정은 자격 증명 풀이 토큰에서 역할을 선택하도록 구성된 경우 사용자의 ID 토큰에 할당될 역할을 결정합니다. 자세한 설명은 그룹에 IAM 역할 할당 섹션을 참조하세요.
   - [이 그룹에 사용자 추가(Add users to this group)] - 그룹이 생성된 후 기존 사용자를 이 그룹의 구성원으로 추가합니다.
7. [생성(Create)]을 선택하여 확인합니다.

사용자 계정 관리 및 검색

사용자 풀을 생성하면 AWS Command Line Interface 또는 Amazon Cognito API 뿐만 아니라 AWS Management Console을 사용하여 사용자를 보고 관리할 수 있습니다. 이 주제에서는 AWS Management Console을 사용하여 사용자를 보고 검색하는 방법에 대해 설명합니다.

사용자 속성 보기

Amazon Cognito 콘솔에서 사용자 속성을 보려면 다음 절차를 따르세요.

사용자 속성을 보려면

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 AWS 자격 증명을 입력합니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택합니다.
4. [사용자(Users)] 탭을 선택한 다음, 목록에서 사용자를 선택합니다.
5. 사용자 세부 정보 페이지의 [사용자 속성(User attributes)]에서 사용자와 연결된 속성을 볼 수 있습니다.

사용자 암호 재설정

Amazon Cognito 콘솔에서 사용자 암호를 재설정하려면 다음 절차를 따르세요.

사용자 암호를 재설정하려면

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 AWS 자격 증명을 입력합니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택합니다.
4. [사용자(Users)] 탭을 선택한 다음, 목록에서 사용자를 선택합니다.
5. 사용자 세부 정보 페이지에서 [작업(Actions)], [암호 재설정(Reset password)]을 차례로 선택합니다.
6. [암호 재설정(Reset password)] 대화 상자에서 정보를 검토한 다음, 준비가 되면 [재설정(Reset)]을 선택합니다.

   이 작업으로 인해 사용자에게 확인 코드가 즉시 전송되며 사용자 상태가 `RESET_REQUIRED`로 변경되어 사용자의 현재 암호를 사용할 수 없게 됩니다. [암호 재설정(Reset password)] 코드는 1시간 동안 유효합니다.

사용자 속성 검색

사용자 풀을 이미 생성한 경우 AWS Management Console의 [사용자(Users)] 패널에서 검색할 수 있습니다. Filter 파라미터를 받는 Amazon Cognito ListUsers API를 사용할 수도 있습니다.

다음 표준 속성 중 하나를 검색할 수 있습니다. 사용자 지정 속성은 검색할 수 없습니다.

- username(대/소문자 구분)
- email

- phone_number
- name
- given_name
- family_name
- preferred_username
- cognito:user_status(콘솔에서 상태라고 함)(대/소문자 구분)
- status(콘솔에서 활성이라고 함)(대/소문자 구분)
- sub

> **Note**
>
> 클라이언트 측 필터를 사용하여 사용자를 나열할 수도 있습니다. 서버 측 필터는 1개 이하의 속성을 일치시킵니다. 고급 검색의 경우 AWS Command Line Interface에서 `list-users` 작업의 `--query` 파라미터에 클라이언트 측 필터를 사용합니다. 클라이언트 측 필터를 사용하는 경우 ListUsers는 0명 이상의 사용자를 페이지가 매겨진 목록으로 반환합니다. 결과가 0인 여러 페이지를 연속으로 받을 수 있습니다. Null 페이지 매김 토큰 값을 받을 때까지 반환되는 각 페이지 매김 토큰으로 쿼리를 반복한 다음, 결합된 결과를 검토합니다.
> 서버 측 필터링과 클라이언트 측 필터링에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서에서 AWS CLI 출력 필터링을 참조하세요.

AWS Management Console을 사용하여 사용자 검색

사용자 풀을 이미 생성한 경우 AWS Management Console의 사용자(Users) 패널에서 검색할 수 있습니다.

AWS Management Console 검색은 항상 접두사("다음으로 시작") 검색입니다.

Amazon Cognito 콘솔에서 사용자를 검색하려면

1. Amazon Cognito 콘솔로 이동합니다. AWS 자격 증명을 입력하라는 메시지가 나타날 수 있습니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택합니다.
4. [사용자(Users)] 탭을 선택하고 검색 필드에 사용자 이름을 입력합니다. 일부 속성 값은 대/소문자를 구분합니다(예: [사용자 이름(User name)]).

검색 필터를 조정하여 다른 사용자 속성(예: [이메일(Email)], [전화 번호(Phone number)] 또는 [성(Last name)])으로 범위를 좁혀 사용자를 찾을 수도 있습니다.

`ListUsers` API를 사용하여 사용자 검색

앱에서 사용자를 검색하려면 Amazon Cognito ListUsers API를 사용합니다. 이 API는 다음 파라미터를 사용합니다.

- `AttributesToGet`: 문자열의 어레이입니다. 여기에서 각 문자열은 검색 결과에서 각 사용자에 대해 반환될 사용자 속성의 이름입니다. 모든 속성을 검색하려면 `AttributesToGet` 파라미터를 포함하거나 리터럴 문자열 `null`의 값으로 `AttributesToGet`을 요청하지 마세요.
- `Filter`: "`AttributeName Filter-Type "AttributeValue"`" 양식의 필터 문자열입니다. 필터 문자열 내의 인용 부호는 백슬래시(\) 문자를 사용하여 이스케이프되어야 합니다. 예: `"family_name = \"Reddy\""`. 필터 문자열이 비어 있는 경우 `ListUsers`는 사용자 풀에서 모든 사용자를 반환합니다.
  - `AttributeName`: 검색할 속성의 이름입니다. 한 번에 속성 하나만 검색할 수 있습니다.

    > **Note**
    >
    > 표준 속성만 검색할 수 있습니다. 사용자 지정 속성은 검색할 수 없습니다. 이는 인덱싱된 속성만 검색할 수 있으며 사용자 지정 속성은 인덱싱될 수 없기 때문입니다.

  - `Filter-Type`: 정확하게 일치해야 하는 경우 `given_name = "Jon"`과 같이 `=`을 사용하세요. 접두사("다음으로 시작")가 일치해야 하는 경우 `given_name ^= "Jon"`과 같이 `^=`를 사용하세요.
  - `AttributeValue`: 각 사용자에 대해 일치해야 하는 속성 값입니다.
- `Limit`: 반환할 최대 사용자 수입니다.
- `PaginationToken`: 이전 검색에서 더 많은 결과를 가져올 토큰입니다. Amazon Cognito는 1시간 후에 페이지 매김 토큰을 만료합니다.
- `UserPoolId`: 검색을 수행해야 할 사용자 풀에 대한 사용자 풀 ID입니다.

모든 검색은 대/소문자를 구분합니다. 검색 결과는 `AttributeName` 문자열로 명명되는 속성별로 오름차순으로 정렬됩니다.

ListUsers API 사용 예제

다음은 모든 사용자를 반환하고 모든 속성을 포함하는 예제입니다.

```
{
    "AttributesToGet": null,
    "Filter": "",
    "Limit": 10,
    "UserPoolId": "us-east-1_samplepool"
}
```

다음은 전화 번호가 "+1312"로 시작하는 모든 사용자를 반환하고 모든 속성을 포함하는 예제입니다.

```
{
    "AttributesToGet": null,
    "Filter": "phone_number ^= \"+1312\"",
    "Limit": 10,
    "UserPoolId": "us-east-1_samplepool"
}
```

다음은 성이 "Reddy"인 사용자 중 처음 10명의 사용자를 반환하는 예제입니다. 각 사용자에 대해 검색 결과에는 사용자의 지정된 이름, 전화 번호 및 이메일 주소가 포함됩니다. 사용자 풀에 일치하는 사용자가 10명을 초과하는 경우 응답에 페이지 매김 토큰이 포함됩니다.

```
{
    "AttributesToGet": [
        "given_name",
        "phone_number",
        "email"
    ],
    "Filter": "family_name = \"Reddy\"",
    "Limit": 10,
    "UserPoolId": "us-east-1_samplepool"
}
```

이전 예제가 페이지 매김 토큰을 반환하는 경우 다음 예제는 동일한 필터 문자열과 일치하는 다음 10명의 사용자를 반환합니다.

```
{
    "AttributesToGet": [
        "given_name",
        "phone_number",
        "email"
    ],
    "Filter": "family_name = \"Reddy\"",
    "Limit": 10,
    "PaginationToken": "pagination_token_from_previous_search",
    "UserPoolId": "us-east-1_samplepool"
}
```

사용자 계정 복구

`AccountRecoverySetting` 파라미터를 사용하면 사용자가 `ForgotPassword` API를 호출할 때 암호를 복구하는 데 사용할 수 있는 방법을 사용자 정의할 수 있습니다. `ForgotPassword`는 확인된 이메일 또는 확인된 전화 번호로 복구 코드를 보냅니다. 복구 코드는 1시간 동안 유효합니다. 사용자 풀에 대해 `AccountRecoverySetting`을 지정할 때 Amazon Cognito는 사용자가 설정한 우선 순위에 따라 코드 전달 대상을 선택합니다.

사용자가 `AccountRecoverySetting`를 정의하고 SMS MFA를 구성한 경우 SMS를 계정 복구 메커니즘으로 사용할 수 없습니다. 이 설정의 우선 순위는 1이 가장 높은 우선 순위로 결정됩니다. Cognito는 지정된 메서드 중 하나에만 확인을 보냅니다.

예를 들어 `admin_only`는 사용자가 자신의 계정을 직접 복구하지 않고 대신 관리자에게 연락하여 계정을 재설정하도록 관리자가 요구할 때 사용되는 값입니다. `admin_only`는 다른 계정 복구 메커니즘과 함께 사용할 수 없습니다.

`AccountRecoverySetting`을 지정하지 않으면 Amazon Cognito는 기존 메커니즘을 사용하여 암호 복구 방법을 결정합니다. 이 경우 Cognito는 확인된 휴대폰을 먼저 사용합니다. 사용자에 대해 확인된 휴대폰을 찾을 수 없는 경우 Cognito는 폴백하고 다음 번 확인된 이메일을 사용합니다.

`AccountRecoverySetting`에 대한 자세한 내용은 Amazon Cognito 자격 증명 공급자 API 참조에서 CreateUserPool 및 UpdateUserPool을 참조하세요.

암호 찾기 동작

암호 및 암호 확인 작업의 일부로서 지정된 시간 동안 사용자가 암호 재설정 코드를 요청 또는 입력하는 데 5~20회의 시도가 허용됩니다. 정확한 값은 요청과 관련된 위험 매개 변수에 따라 다릅니다. 이 동작은 변경될 수 있습니다.

사용자 풀로 사용자 가져오기

두 가지 방법으로 기존 사용자 디렉터리 또는 사용자 데이터베이스에서 Amazon Cognito 사용자 풀로 사용자를 가져오거나 마이그레이션할 수 있습니다. Amazon Cognito를 사용하여 처음 로그인할 때 사용자 마이그레이션 Lambda 트리거를 사용하여 사용자를 마이그레이션할 수 있습니다. 이 방법을 사용하면 사용자가 기존 암호를 계속 사용할 수 있으므로 사용자 풀로 마이그레이션한 후 재설정할 필요가 없습니다. 또는 모든 사용자의 사용자 프로파일 속성이 포함된 CSV 파일을 업로드하여 사용자를 대량으로 마이그레이션할 수 있습니다. 다음 섹션에서는 이 두 가지 방법을 설명합니다.

주제

사용자 마이그레이션 Lambda 트리거를 사용하여 사용자 풀로 사용자 가져오기

이 접근 방식을 사용하면 사용자가 앱으로 처음 로그인하거나 암호 재설정을 요청할 때 기존 사용자 디렉터리에 있는 사용자를 사용자 풀로 원활하게 마이그레이션할 수 있습니다. 사용자 마이그레이션 Lambda 트리거 함수를 사용자 풀에 추가합니다. 이 함수는 로그인하려고 하는 사용자에 대한 메타데이터를 수신하고 외부 자격 증명 소스에서 사용자 프로필 정보를 반환합니다. 요청 및 응답 파라미터를 포함한 이 Lambda 트리거에 대한 자세한 내용과 코드 예는 사용자 마이그레이션 Lambda 트리거 파라미터 섹션을 참조하세요.

사용자 마이그레이션을 시작하기 전에 AWS 계정에서 사용자 마이그레이션 Lambda 함수를 생성하고, 이 Lambda 함수를 사용자 풀의 사용자 마이그레이션 트리거로 설정합니다. Amazon Cognito 서비스 계정 보안 주체 `cognito-idp.amazonaws.com`만 사용자 자신의 사용자 풀 컨텍스트에서만 Lambda 함수를 호출하도록 허용하는 권한 부여 정책을 Lambda 함수에 추가합니다. 자세한 내용은 AWS Lambda에 대한 리소스 기반 정책(Lambda 함수 정책) 사용을 참조하세요.

로그인 프로세스

1. 사용자가 앱을 열고 Amazon Cognito 사용자 풀 API 또는 Amazon Cognito 호스팅 UI를 통해 로그인합니다. Amazon Cognito API로 쉽게 로그인하는 방법에 대한 자세한 내용은 웹 및 모바일 앱과 Amazon Cognito 인증 및 권한 부여 통합 섹션을 참조하세요.
2. 앱이 사용자 이름 및 암호를 Amazon Cognito로 보냅니다. 앱에 AWS SDK를 사용하여 빌드한 사용자 지정 로그인 UI가 있는 경우 해당 앱에서 InitiateAuth 또는 AdminInitiateAuth를 `USER_PASSWORD_AUTH` 또는 `ADMIN_USER_PASSWORD_AUTH` 흐름과 함께 사용해야 합니다. 앱에서 이러한 흐름 중 하나를 사용하면 SDK가 암호를 서버로 보냅니다.

> **Note**
>
> 사용자 마이그레이션 트리거를 추가하기 전에 앱 클라이언트 설정에서 `USER_PASSWORD_AUTH` 또는 `ADMIN_USER_PASSWORD_AUTH` 흐름을 활성화합니다. 기본 `USER_SRP_AUTH` 흐름 대신 이러한 흐름을 사용해야 합니다. Amazon Cognito는 다른 디렉터리에서 사용자의 인증을 확인할 수 있도록 Lambda 함수에 암호를 보내야 합니다. SRP는 Lambda 함수에서 사용자의 암호를 가립니다.

3. Amazon Cognito는 제출된 사용자 이름이 사용자 풀의 사용자 이름 또는 별칭과 일치하는지 확인합니다. 사용자의 이메일 주소, 전화 번호 또는 기본 설정 사용자 이름을 사용자 풀의 별칭으로 설정할 수 있습니다. 사용자가 존재하지 않는 경우 Amazon Cognito는 사용자 이름 및 암호를 포함한 파라미터를 사용자 마이그레이션 Lambda 트리거 함수에 보냅니다.
4. 사용자 마이그레이션 Lambda 트리거 함수는 기존 사용자 디렉터리 또는 사용자 데이터베이스를 사용하여 사용자를 확인하거나 인증합니다. 이 함수는 Amazon Cognito가 사용자 풀의 사용자 프로필에 저장하는 사용자 속성을 반환합니다. 제출된 사용자 이름이 별칭 속성과 일치하는 경우에만 `username` 파라미터를 반환할 수 있습니다. 사용자가 기존 암호를 계속 사용하도록 하려는 경우 함수가 Lambda 응답에서 `finalUserStatus` 속성을 `CONFIRMED`로 설정합니다. 앱은 사용자 마이그레이션 Lambda 트리거 파라미터에 표시된 모든 `"response"` 파라미터를 반환해야 합니다.

> **Important**
>
> 사용자 마이그레이션 Lambda 코드에 전체 요청 이벤트 객체를 기록하지 마세요. 이 요청 이벤트 객체에는 사용자의 암호가 포함됩니다. 로그를 폐기하지 않는 경우 암호가 CloudWatch 로그에 나타납니다.

5. Amazon Cognito가 사용자 풀에 사용자 프로파일을 생성하고 앱 클라이언트에 토큰을 반환합니다.
6. 앱이 토큰 수집을 수행하고, 사용자 인증을 수락하고, 요청된 콘텐츠로 진행합니다.

사용자를 마이그레이션한 후 `USER_SRP_AUTH`를 로그인에 사용합니다. SRP(보안 원격 암호) 프로토콜에서 네트워크를 통해 암호를 보내지 않고 마이그레이션 중에 사용하는 `USER_PASSWORD_AUTH` 흐름에서 보안 강화를 제공합니다.

마이그레이션 중에 클라이언트 디바이스 문제, 네트워크 문제 등의 오류가 발생하면 앱이 Amazon Cognito 사용자 풀 API로부터 오류 응답을 수신합니다. 이 경우 Amazon Cognito가 사용자 풀에서 사용자 계정을 생성하거나 생성하지 않을 수도 있습니다. 그러면 사용자가 다시 로그인을 시도해야 합니

다. 로그인이 반복해서 실패할 경우 앱에서 암호 찾기 흐름을 사용하여 사용자의 암호 재설정을 시도합니다.

암호 찾기 흐름도 `UserMigration_ForgotPassword` 이벤트 소스를 사용하여 사용자 마이그레이션 Lambda 트리거 암호를 호출합니다. 사용자가 암호 재설정을 요청할 때 암호를 제출하지 않으므로 Amazon Cognito는 Lambda 함수로 전송하는 경우 암호를 포함하지 않습니다. 함수는 기존 사용자 디렉터리에서 사용자를 조회하고 사용자 풀의 사용자 프로필에 추가할 속성을 반환할 수만 있습니다. 함수가 간접 호출을 완료하고 Amazon Cognito에 응답을 반환하면 사용자 풀이 이메일 또는 SMS로 암호 재설정 코드를 보냅니다. 앱에서 사용자에게 확인 코드와 새 암호를 입력하라는 메시지를 표시한 다음 ConfirmForgotPassword API 요청에서 해당 정보를 Amazon Cognito로 전송합니다. Amazon Cognito 호스팅 UI의 암호 찾기 흐름에 대한 기본 제공 페이지를 사용할 수도 있습니다.

CSV 파일에서 사용자 풀로 사용자 가져오기

사용자를 Amazon Cognito 사용자 풀로 가져올 수 있습니다. 사용자 정보는 특수한 형식의.csv 파일에서 가져옵니다. 이 가져오기 프로세스는 password를 제외한 모든 사용자 속성의 값을 설정합니다. 암호 가져오기는 지원되지 않습니다. 보안 모범 사례에서는 일반 텍스트는 암호로 사용할 수 없으며, 해시 가져오기는 지원되지 않기 때문입니다. 즉, 사용자가 처음 로그인할 때 암호를 변경해야 합니다. 따라서 이 방법을 사용하여 가져올 때 사용자는 RESET_REFACED 상태가 됩니다.

`Permanent` 파라미터를 `true`로 설정하는 AdminSetUserPassword API 요청을 이용해 사용자의 암호를 설정할 수 있습니다.

> **Note**
>
> 각 사용자의 생성 날짜는 해당 사용자를 사용자 풀로 가져온 시간입니다. 생성 날짜는 가져온 속성 중 하나가 아닙니다.

기본 단계는 다음과 같습니다.

1. AWS Identity and Access Management(IAM) 콘솔에서 Amazon CloudWatch Logs 역할을 생성합니다.
2. 사용자 가져오기.csv 파일을 만듭니다.
3. 사용자 가져오기 작업을 생성하고 실행합니다.
4. 사용자 가져오기.csv 파일을 업로드합니다.
5. 사용자 가져오기 작업을 시작하고 실행합니다.

6. CloudWatch를 사용하여 이벤트 로그를 확인합니다.
7. 가져온 사용자에게 암호를 재설정하도록 요구합니다.

주제

- CloudWatch Logs IAM 역할 생성
- 사용자 가져오기 CSV 파일 생성
- Amazon Cognito 사용자 풀 가져오기 작업 생성 및 실행
- CloudWatch 콘솔에서 사용자 풀 가져오기 결과 보기
- 가져온 사용자에게 암호 재설정 요구

CloudWatch Logs IAM 역할 생성

Amazon Cognito CLI 또는 API를 사용하는 경우 CloudWatch IAM 역할을 생성해야 합니다. 다음 절차에서는 Amazon Cognito가 가져오기 작업의 결과를 CloudWatch Logs Logs에 기록하는 데 사용할 수 있는 IAM 역할을 생성하는 방법을 설명합니다.

> **Note**
>
> Amazon Cognito 콘솔에서 가져오기 작업을 생성할 때 IAM 역할을 생성할 수 있습니다. Create a new IAM role(새 IAM 역할 생성)을 선택하면 Amazon Cognito는 해당 역할에 적절한 신뢰 정책과 IAM 정책을 자동으로 적용합니다.

사용자 풀 가져오기를 위한 CloudWatch Logs IAM 역할을 생성하려면(AWS CLI, API)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. AWS 서비스용 새 IAM 역할 생성 자세한 지침은 AWS Identity and Access Management 사용 설명서의 AWS 서비스에 대한 역할 생성을 참조하세요.
   a. Trusted entity type(신뢰할 수 있는 엔티티 유형)의 Use case(사용 사례)를 선택할 때 아무 서비스를 선택합니다. Amazon Cognito는 현재 서비스 사용 사례에 나열되지 않습니다.
   b. Add permissions(권한 추가) 화면에서 Create policy(정책 생성)을 선택하고 다음 정책 설명을 삽입합니다. *REGION*을 사용자 풀의 AWS 리전(예: `us-east-1`)으로 바꿉니다. *ACCOUNT*를 AWS 계정 ID(예: `111122223333`)로 바꿉니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:REGION:ACCOUNT:log-group:/aws/cognito/*"
            ]
        }
    ]
}
```

3. 역할을 만들 때 Amazon Cognito를 신뢰할 수 있는 개체로 선택하지 않았으므로, 이제 역할의 신뢰 관계를 수동으로 편집해야 합니다. IAM 콘솔의 탐색 창에서 Roles(역할)를 선택하고 생성한 새 역할을 선택합니다.
4. 신뢰 관계 탭을 선택합니다.
5. 신뢰 정책 편집(Edit trust policy)을 선택합니다.
6. 다음 정책 설명을 Edit trust policy(신뢰 정책 편집)에 붙여넣어 기존 텍스트를 대체합니다.

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

7. 정책 업데이트(Update policy)를 선택합니다.
8. 역할 ARN을 기록해 둡니다. 가져오기 작업을 생성할 때 ARN을 입력해야 합니다.

사용자 가져오기 CSV 파일 생성

기존 사용자를 사용자 풀로 가져오려면 먼저 가져올 사용자와 사용자의 속성이 포함된 쉼표로 구분된 값(CSV) 파일을 생성해야 합니다. 사용자 풀에서는 사용자 풀의 속성 스키마를 반영하는 헤더가 있는 사용자 가져오기 파일을 검색할 수 있습니다. 그런 다음 CSV 파일 형식 지정의 형식 지정 요구 사항과 일치하는 사용자 정보를 삽입할 수 있습니다.

CSV 파일 헤더 다운로드(콘솔)

다음 절차를 사용하여 CSV 헤더 파일을 다운로드합니다.

CSV 파일 헤더를 다운로드하려면

1. Amazon Cognito 콘솔로 이동합니다. AWS 자격 증명을 입력하라는 메시지가 나타날 수 있습니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택합니다.
4. 사용자(Users) 탭을 선택합니다.
5. Import users(사용자 가져오기) 섹션에서 Create an import job(가져오기 작업 생성)을 선택합니다.
6. Upload CSV(CSV 업로드)에서 template.csv 링크를 선택하고 CSV 파일을 다운로드합니다.

CSV 파일 헤더 다운로드(AWS CLI)

올바른 헤더 목록을 보려면 다음 CLI 명령을 실행합니다. 여기서 *USER_POOL_ID*는 사용자를 가져올 사용자 풀의 사용자 풀 식별자입니다.

```
aws cognito-idp get-csv-header --user-pool-id "USER_POOL_ID"
```

샘플 응답:

```
{
    "CSVHeader": [
        "name",
        "given_name",
        "family_name",
        "middle_name",
        "nickname",
        "preferred_username",
```

```
        "profile",
        "picture",
        "website",
        "email",
        "email_verified",
        "gender",
        "birthdate",
        "zoneinfo",
        "locale",
        "phone_number",
        "phone_number_verified",
        "address",
        "updated_at",
        "cognito:mfa_enabled",
        "cognito:username"
    ],
    "UserPoolId": "USER_POOL_ID"
}
```

CSV 파일 형식 지정

다운로드한 사용자 가져오기 CSV 헤더 파일은 다음 문자열처럼 보입니다. 사용자 풀에 추가한 사용자 지정 속성도 포함되어 있습니다.

```
cognito:username,name,given_name,family_name,middle_name,nickname,preferred_username,profile,pi
```

CSV 파일을 편집하여 이 헤더와 사용자의 속성 값이 포함되고 다음 규칙에 따라 형식이 지정되게 해야 합니다.

> **Note**
>
> 전화 번호의 올바른 형식 등, 속성 값에 대한 자세한 내용은 사용자 풀 속성 섹션을 참조하세요.

- 파일의 첫 번째 행은 사용자 속성 이름을 포함하는 다운로드된 헤더 행입니다.
- CSV 파일의 열 순서는 중요하지 않습니다.
- 첫 번째 행 뒤의 각 행에는 사용자의 속성 값이 포함됩니다.
- 헤더의 모든 열이 있어야 하지만 모든 열에 값을 제공할 필요는 없습니다.
- 다음과 같은 속성이 필요합니다.

  - cognito:username
  - cognito:mfa_enabled
  - email_verified 또는 phone_number_verified
    - 각 사용자의 자동 확인 속성 중 하나 이상이 `true`이어야 합니다. 자동 확인 속성은 새 사용자가 사용자 풀에 가입할 때 Amazon Cognito에서 자동으로 코드를 보내는 이메일 주소 또는 전화 번호입니다.
    - 사용자 풀에는 하나 이상의 자동 확인 속성(email_verified 또는 phone_number_verified)이 있어야 합니다. 사용자 풀에 자동 확인된 속성이 없으면 가져오기 작업이 시작되지 않습니다.
    - 사용자 풀에 자동 확인 속성이 하나만 있는 경우 각 사용자에 대해 해당 속성을 확인해야 합니다. 예를 들어 사용자 풀에 phone_number만 자동 검증 속성으로 설정되어 있는 경우 각 사용자의 phone_number_verified 값이 `true`이어야 합니다.

    > **Note**
    >
    > 사용자가 암호를 재설정하려면 확인된 이메일 또는 전화 번호가 있어야 합니다. Amazon Cognito는 CSV 파일에 지정된 이메일 또는 전화 번호로 재설정 암호 코드가 포함된 메시지를 보냅니다. 전화 번호로 메시지를 전송하는 경우 SMS 메시지를 통해 전송됩니다. 자세한 내용은 가입 시 연락처 정보 확인 섹션을 참조하세요.

  - email(email_verified가 `true`인 경우)
  - phone_number(phone_number_verified가 `true`인 경우)
  - 사용자 풀을 생성할 때 필수 속성으로 표시한 모든 속성
- 문자열 속성 값은 인용 부호로 묶여 있지 않아야 합니다.
- 속성 값에 쉼표가 포함된 경우 쉼표 앞에 백슬래시(\)를 넣어야 합니다. 이는 CSV 파일의 필드가 쉼표로 구분되어 있기 때문입니다.
- CSV 파일 내용은 바이트 순서 표시가 없는 UTF-8 형식이어야 합니다.
- cognito:username 필드는 필수이며 사용자 풀 내에서 고유해야 합니다. 유니 코드 문자열일 수 있습니다. 단, 공백 또는 탭을 포함할 수 없습니다.
- birthdate 값이 있는 경우 이 값은 *mm/dd/yyyy* 포맷이어야 합니다. 즉, 예를 들어 생년월일 1985년 2월 1일은 **02/01/1985**로 인코딩되어야 합니다.
- cognito:mfa_enabled 필드는 필수입니다. 사용자 풀에서 멀티 팩터 인증(MFA)을 필수로 적용하도록 설정한 경우 이 필드는 모든 사용자에 대해 `true`이어야 합니다. MFA를 해제하도록 설정한 경우 이 필드는 모든 사용자에 대해 `false`이어야 합니다. MFA를 선택 사항으로 설정한 경우 이 필드는 `true` 또는 `false`일 수 있지만 비워 둘 수는 없습니다.

- 최대 행 길이는 16,000자입니다.
- 최대 CSV 파일 크기는 100MB입니다.
- 파일의 최대 행(사용자) 수는 500,000개입니다. 헤더 행은 이 최대값에 포함되지 않습니다.
- updated_at 필드 값은 초 단위의 Epoch 시간이어야 합니다(예: **1471453471**).
- 속성 값의 선행 또는 후행 공백은 잘립니다.

다음 목록은 사용자 지정 속성이 없는 사용자 풀에 대한 CSV 가져오기 파일 예시입니다. 사용자 풀 스키마가 이 예시와 다를 수 있습니다. 이 경우 사용자 풀에서 다운로드한 CSV 템플릿에 테스트 값을 입력해야 합니다.

```
cognito:username,name,given_name,family_name,middle_name,nickname,preferred_username,profile,pi
John,,John,Doe,,,,,,,johndoe@example.com,TRUE,,02/01/1985,,,+12345550100,TRUE,123 Any
 Street,,FALSE
Jane,,Jane,Roe,,,,,,,janeroe@example.com,TRUE,,01/01/1985,,,+12345550199,TRUE,100 Main
 Street,,FALSE
```

Amazon Cognito 사용자 풀 가져오기 작업 생성 및 실행

이 섹션에서는 Amazon Cognito 콘솔과 AWS Command Line Interface(AWS CLI)를 사용하여 사용자 풀 가져오기 작업을 생성하고 실행하는 방법을 설명합니다.

주제

- CSV 파일에서 사용자 가져오기(콘솔)
- 사용자 가져오기(AWS CLI)

CSV 파일에서 사용자 가져오기(콘솔)

다음 절차에서는 CSV 파일에서 사용자를 가져오는 방법에 대해 설명합니다.

CSV 파일에서 사용자를 가져오려면(콘솔)

1. Amazon Cognito 콘솔로 이동합니다. AWS 자격 증명을 입력하라는 메시지가 나타날 수 있습니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택합니다.
4. 사용자(Users) 탭을 선택합니다.
5. Import users(사용자 가져오기) 섹션에서 Create an import job(가져오기 작업 생성)을 선택합니다.

6. Create import job(가져오기 작업 생성) 페이지에 Job name(작업 이름)을 입력합니다.
7. Create a new IAM role(새 IAM 역할 생성) 또는Use an existing IAM role(기존 IAM 역할 사용)을 선택합니다.

   a. Create a new IAM role(새 IAM 역할 생성)을 선택한 경우 새 역할의 이름을 입력합니다. Amazon Cognito는 올바른 권한과 신뢰 관계를 가진 역할을 자동으로 생성합니다. 가져오기 작업을 생성하는 IAM 보안 주체는 IAM 역할을 생성할 권한이 있어야 합니다.

   b. Use an existing IAM role(기존 IAM 역할 사용)을 선택했다면 IAM role selection(IAM 역할 선택) 목록에서 역할을 선택합니다. 이 역할에는 CloudWatch Logs IAM 역할 생성에서 설명하는 권한 및 신뢰 정책이 있어야 합니다.
8. Create job(작업 생성)을 선택하여 작업을 제출하되, 시작은 나중에 합니다. 작업을 제출하고 즉시 시작하려면 Create and start job(작업 생성 및 시작)을 선택합니다.
9. 작업을 생성했지만 시작하지 않았다면 나중에 작업을 시작할 수 있습니다. Import users(사용자 가져오기)의 Users(사용자) 탭에서 가져오기 작업을 선택하고 Start(시작)를 선택합니다. AWS SDK에서 StartUserImportJob API 요청을 제출해도 됩니다.
10. Import users(사용자 가져오기)의 Users(사용자) 탭에서 사용자 가져오기 작업의 진행 상황을 모니터링합니다. 작업이 실패할 경우 Status(상태) 값을 선택합니다. 자세한 내용을 보려면 View the CloudWatch logs for more details(CloudWatch 로그 상세 정보 보기)를 선택하고 CloudWatch Logs 콘솔에서 문제를 검토합니다.

사용자 가져오기(AWS CLI)

다음 CLI 명령을 사용하여 사용자를 사용자 풀로 가져올 수 있습니다.

- `create-user-import-job`
- `get-csv-header`
- `describe-user-import-job`
- `list-user-import-jobs`
- `start-user-import-job`
- `stop-user-import-job`

이러한 명령의 명령줄 옵션 목록을 표시하려면 `help` 명령줄 옵션을 사용합니다. 예:

```
aws cognito-idp get-csv-header help
```

사용자 가져오기 작업 생성

CSV 파일을 생성한 후 다음 CLI 명령을 실행하여 사용자 가져오기 작업을 생성합니다. 여기서 *JOB_NAME*은 선택한 작업 이름이고, *USER_POOL_ID*는 새 사용자가 추가되는 사용자 풀의 사용자 풀 ID이며, *ROLE_ARN*은 CloudWatch Logs IAM 역할 생성에서 받은 역할 ARN입니다.

```
aws cognito-idp create-user-import-job --job-name "JOB_NAME" --user-pool-id
 "USER_POOL_ID" --cloud-watch-logs-role-arn "ROLE_ARN"
```

응답에 반환되는 *PRE_SIGNED_URL*은 15 분 동안 유효합니다. 이 시간이 지나면 만료되며 새 URL을 받으려면 새 사용자 가져오기 작업을 만들어야 합니다.

Example 샘플 응답:

```
{
    "UserImportJob": {
        "Status": "Created",
        "SkippedUsers": 0,
        "UserPoolId": "USER_POOL_ID",
        "ImportedUsers": 0,
        "JobName": "JOB_NAME",
        "JobId": "JOB_ID",
        "PreSignedUrl": "PRE_SIGNED_URL",
        "CloudWatchLogsRoleArn": "ROLE_ARN",
        "FailedUsers": 0,
        "CreationDate": 1470957431.965
    }
}
```

사용자 가져오기 작업의 상태 값

사용자 가져오기 명령에 대한 응답에 다음 `Status` 값 중 하나가 표시됩니다.

- `Created` - 작업이 생성되었지만 시작되지 않았습니다.
- `Pending` - 전환 상태입니다. 작업을 시작했지만 아직 사용자 가져오기를 시작하지 않았습니다.
- `InProgress` - 작업이 시작되었으며 사용자를 가져오는 중입니다.
- `Stopping` - 작업을 중지했지만 작업에서 아직 사용자 가져오기가 중지되지 않았습니다.
- `Stopped` - 작업을 중지했으며 작업에서 사용자 가져오기가 중지되었습니다.
- `Succeeded` - 작업이 성공적으로 완료되었습니다.

- `Failed` - 오류로 인해 작업이 중지되었습니다.
- `Expired` - 작업을 생성했지만 24~48시간 이내에 작업을 시작하지 않았습니다. 작업과 연결된 모든 데이터가 삭제되었으며 작업을 시작할 수 없습니다.

CSV 파일 업로드

다음 `curl` 명령을 사용하여 사용자 데이터가 포함된 CSV 파일을 `create-user-import-job` 명령의 응답에서 얻은 미리 서명된 URL에 업로드합니다.

```
curl -v -T "PATH_TO_CSV_FILE" -H "x-amz-server-side-encryption:aws:kms"
 "PRE_SIGNED_URL"
```

이 명령의 출력에서 `"We are completely uploaded and fine"`이라는 문구를 찾습니다. 이 구문은 파일이 성공적으로 업로드되었음을 나타냅니다.

사용자 가져오기 작업 설명

사용자 가져오기 작업에 대한 설명을 보려면 다음 명령을 사용합니다. 여기서 *USER_POOL_ID*는 사용자 풀 ID이며, *JOB_ID*는 사용자 가져오기 작업을 생성할 때 반환된 작업 ID입니다.

```
aws cognito-idp describe-user-import-job --user-pool-id "USER_POOL_ID" --job-id
 "JOB_ID"
```

Example 샘플 응답:

```
{
    "UserImportJob": {
        "Status": "Created",
        "SkippedUsers": 0,
        "UserPoolId": "USER_POOL_ID",
        "ImportedUsers": 0,
        "JobName": "JOB_NAME",
        "JobId": "JOB_ID",
        "PreSignedUrl": "PRE_SIGNED_URL",
        "CloudWatchLogsRoleArn":"ROLE_ARN",
        "FailedUsers": 0,
        "CreationDate": 1470957431.965
    }
}
```

앞의 샘플 출력에서 *PRE_SIGNED_URL*은 CSV 파일을 업로드한 URL입니다. *ROLE_ARN*은 역할을 생성할 때 받은 CloudWatch Logs 역할 ARN입니다.

사용자 가져오기 작업 나열

사용자 가져오기 작업을 나열하려면 다음 명령을 사용합니다.

```
aws cognito-idp list-user-import-jobs --user-pool-id "USER_POOL_ID" --max-results 2
```

Example 샘플 응답:

```
{
    "UserImportJobs": [
        {
            "Status": "Created",
            "SkippedUsers": 0,
            "UserPoolId": "USER_POOL_ID",
            "ImportedUsers": 0,
            "JobName": "JOB_NAME",
            "JobId": "JOB_ID",
            "PreSignedUrl":"PRE_SIGNED_URL",
            "CloudWatchLogsRoleArn":"ROLE_ARN",
            "FailedUsers": 0,
            "CreationDate": 1470957431.965
        },
        {
            "CompletionDate": 1470954227.701,
            "StartDate": 1470954226.086,
            "Status": "Failed",
            "UserPoolId": "USER_POOL_ID",
            "ImportedUsers": 0,
            "SkippedUsers": 0,
            "JobName": "JOB_NAME",
            "CompletionMessage": "Too many users have failed or been skipped during the
 import.",
            "JobId": "JOB_ID",
            "PreSignedUrl":"PRE_SIGNED_URL",
            "CloudWatchLogsRoleArn":"ROLE_ARN",
            "FailedUsers": 5,
            "CreationDate": 1470953929.313
        }
    ],
    "PaginationToken": "PAGINATION_TOKEN"
```

```
}
```

작업은 마지막으로 생성한 것부터 처음으로 생성한 것까지 시간순으로 나열됩니다. 두 번째 작업 다음에 있는 *PAGINATION_TOKEN* 문자열은 이 list 명령에 대한 추가 결과가 있음을 나타냅니다. 추가 결과를 나열하려면 다음과 같이 `--pagination-token` 옵션을 사용합니다.

```
aws cognito-idp list-user-import-jobs --user-pool-id "USER_POOL_ID" --max-results 10 --
pagination-token "PAGINATION_TOKEN"
```

사용자 가져오기 작업 시작

사용자 가져오기 작업을 시작하려면 다음 명령을 사용합니다.

```
aws cognito-idp start-user-import-job --user-pool-id "USER_POOL_ID" --job-id "JOB_ID"
```

계정당 한 번에 하나의 가져오기 작업만 활성화할 수 있습니다.

Example 샘플 응답:

```
{
    "UserImportJob": {
        "Status": "Pending",
        "StartDate": 1470957851.483,
        "UserPoolId": "USER_POOL_ID",
        "ImportedUsers": 0,
        "SkippedUsers": 0,
        "JobName": "JOB_NAME",
        "JobId": "JOB_ID",
        "PreSignedUrl":"PRE_SIGNED_URL",
        "CloudWatchLogsRoleArn": "ROLE_ARN",
        "FailedUsers": 0,
        "CreationDate": 1470957431.965
    }
}
```

사용자 가져오기 작업 중지

진행 중인 사용자 가져오기 작업을 중지하려면 다음 명령을 사용합니다. 작업을 중지한 후에는 다시 시작할 수 없습니다.

```
aws cognito-idp stop-user-import-job --user-pool-id "USER_POOL_ID" --job-id "JOB_ID"
```

Example 샘플 응답:

```
{
    "UserImportJob": {
        "CompletionDate": 1470958050.571,
        "StartDate": 1470958047.797,
        "Status": "Stopped",
        "UserPoolId": "USER_POOL_ID",
        "ImportedUsers": 0,
        "SkippedUsers": 0,
        "JobName": "JOB_NAME",
        "CompletionMessage": "The Import Job was stopped by the developer.",
        "JobId": "JOB_ID",
        "PreSignedUrl":"PRE_SIGNED_URL",
        "CloudWatchLogsRoleArn": "ROLE_ARN",
        "FailedUsers": 0,
        "CreationDate": 1470957972.387
    }
}
```

CloudWatch 콘솔에서 사용자 풀 가져오기 결과 보기

Amazon CloudWatch 콘솔에서 가져오기 작업의 결과를 볼 수 있습니다.

주제

- 결과 보기
- 결과 해석

결과 보기

다음 단계에서는 사용자 풀 가져오기 결과를 보는 방법을 설명합니다.

사용자 풀 가져오기의 결과를 보려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. [Logs]를 선택합니다.
3. 사용자 풀 가져오기 작업에 대한 로그 그룹을 선택합니다. 로그 그룹 이름은 `/aws/cognito/userpools/USER_POOL_ID/USER_POOL_NAME` 형식입니다.

4. 방금 실행한 사용자 가져오기 작업에 대한 로그를 선택합니다. 로그 이름은 *JOB_ID*/*JOB_NAME* 형식입니다. 로그의 결과에서는 행 번호로 사용자를 나타냅니다. 사용자 데이터는 로그에 기록되지 않습니다. 각 사용자에 대해 다음과 유사한 행이 표시됩니다.

   - `[SUCCEEDED] Line Number 5956 - The import succeeded.`
   - `[SKIPPED] Line Number 5956 - The user already exists.`
   - `[FAILED] Line Number 5956 - The User Record does not set any of the auto verified attributes to true. (Example: email_verified to true).`

결과 해석

성공적으로 가져온 사용자는 상태가 'PasswordReset'으로 설정되어 있습니다.

다음과 같은 경우에는 사용자가 가져와지지 않지만 가져오기 작업은 계속됩니다.

- `true`로 설정된 자동 확인 속성이 없는 경우.
- 사용자 데이터가 스키마와 일치하지 않는 경우.
- 내부 오류로 인해 사용자를 가져올 수 없는 경우.

다음과 같은 경우 가져오기 작업이 실패합니다.

- Amazon CloudWatch Logs 역할을 수임할 수 없거나, 해당 역할에 올바른 액세스 정책이 없거나, 해당 ㅇ역할이 삭제된 경우.
- 사용자 풀이 삭제된 경우.
- Amazon Cognito가 .csv 파일을 구문 분석할 수 없는 경우.

가져온 사용자에게 암호 재설정 요구

가져온 사용자는 처음 로그인하고 암호를 입력할 때 새 암호를 입력해야 합니다. 다음 절차는 CSV 파일을 가져온 후 로컬 사용자가 있는 사용자 지정 앱에서의 사용자 경험을 설명합니다. 사용자가 호스팅된 UI로 로그인하면 Amazon Cognito는 사용자가 처음 로그인할 때 새 암호를 설정하라는 메시지를 표시합니다.

가져온 사용자에게 암호 재설정 요구

1. 앱에서 임의의 암호를 사용하여 `InitiateAuth`를 통해 현재 사용자의 로그인을 자동으로 시도합니다.

2. `PreventUserExistenceErrors`를 사용할 경우 Amazon Cognito가 `NotAuthorizedException`을 반환합니다. 그렇지 않은 경우에는 `PasswordResetRequiredException`를 반환합니다.
3. 앱에서 `ForgotPassword` API를 요청하고 사용자의 암호를 재설정합니다.
   a. 앱에서 `ForgotPassword` API 요청에 사용자 이름을 제출합니다.
   b. Amazon Cognito는 확인된 이메일 또는 전화 번호에 코드를 보냅니다. 대상은 CSV 파일의 `email_verified` 및 `phone_number_verified`에 입력한 값에 따라 달라집니다. `ForgotPassword` 요청에 대한 응답은 코드의 대상을 나타냅니다.

      > **Note**
      >
      > 이메일이나 전화 번호를 인증하도록 사용자 풀을 구성해야 합니다. 자세한 내용은 사용자 계정 가입 및 확인 섹션을 참조하세요.

   c. 앱은 코드가 전송된 위치를 확인하라는 메시지를 사용자에게 표시하며, 코드와 새 암호를 입력하라고 지시합니다.
   d. 사용자가 앱에 코드와 새 암호를 입력합니다.
   e. 앱은 `ConfirmForgotPassword` API 요청에 코드와 새 암호를 제출합니다.
   f. 앱이 사용자를 로그인으로 리디렉션합니다.

사용자 풀 속성

속성은 이름, 이메일 주소, 전화 번호 등 개별 사용자를 식별하는 데 도움이 되는 정보입니다. 새 사용자 풀에는 기본 표준 속성 집합이 있습니다. 에서 사용자 풀 정의에 사용자 지정 특성을 추가할 수도 AWS Management Console있습니다. 이 섹션에서는 이러한 속성을 자세히 설명하며 사용자 풀을 설정하는 방법에 대한 팁을 제공합니다.

사용자에 대한 일부 정보는 속성에 저장하지 마세요. 예를 들어, 사용 통계나 게임 점수와 같이 자주 변경되는 사용자 데이터는 Amazon Cognito Sync 또는 Amazon DynamoDB와 같은 별도의 데이터 스토어에 보관합니다.

> **Note**
>
> 일부 문서 및 표준에서는 속성을 멤버로 참조합니다.

주제

표준 속성

Amazon Cognito는 OpenID Connect 사양에 따라 모든 사용자에게 표준 속성 집합을 할당합니다. 기본적으로 표준 및 사용자 지정 속성 값은 최대 2,048자의 문자열일 수 있지만 일부 속성 값에는 형식 제한이 있습니다.

표준 속성은 다음과 같습니다.

- `address`
- `birthdate`
- `email`
- `family_name`
- `gender`
- `given_name`
- `locale`
- `middle_name`
- `name`
- `nickname`
- `phone_number`
- `picture`
- `preferred_username`
- `profile`
- `sub`
- `updated_at`
- `website`
- `zoneinfo`

sub를 제외하고 표준 속성은 기본적으로 모든 사용자에게 선택 사항입니다. 필수 속성으로 설정하려면 사용자 풀을 생성하는 동안 속성 옆의 필수(Required) 확인란을 선택합니다. Amazon Cognito는 각 사용자의 sub 속성에 고유한 사용자 식별자 값을 할당합니다. email 및 phone_number 속성만 확인할 수 있습니다.

> **Note**
>
> 표준 속성이 필수(Required)로 표시된 경우 사용자는 해당 속성의 값을 제공해야 등록할 수 있습니다. 관리자는 AdminCreateUserAPI를 사용하여 사용자를 생성하고 필수 속성에는 값을 제공하지 않도록 할 수 있습니다. 사용자 풀을 생성한 후에는 필수 속성과 선택적 속성 간을 전환할 수 없습니다.

표준 속성 세부 정보 및 형식 제한

birthdate

값은 YYYY-MM-DD 형식으로 유효한 10자 날짜여야 합니다.

이메일

사용자 및 관리자는 이메일 주소 값을 확인할 수 있습니다.

적절한 AWS 계정 권한이 있는 관리자는 사용자의 이메일 주소를 변경하고 확인된 것으로 표시할 수도 있습니다. AdminUpdateUserAttributesAPI 또는 admin-update-user-attributes AWS Command Line Interface (AWS CLI) 명령을 사용하여 이메일 주소를 확인된 것으로 표시합니다. 관리자는 이 명령을 사용하여 `email_verified` 속성을 `true`로 변경할 수 있습니다. 의 사용자 탭에서 사용자를 AWS Management Console 편집하여 이메일 주소를 확인된 것으로 표시할 수도 있습니다.

값은 @ 기호 및 도메인이 포함된 표준 이메일 형식을 따르는 유효한 이메일 주소 문자열이어야 하며 길이는 최대 2,048자여야 합니다.

phone_number

SMS 멀티 팩터 인증(MFA)이 활성화되어 있는 경우 사용자는 전화 번호를 제공해야 합니다. 자세한 내용은 사용자 풀에 MFA 추가 섹션을 참조하세요.

사용자 및 관리자는 전화 번호 값을 확인할 수 있습니다.

적절한 AWS 계정 권한이 있는 관리자는 사용자의 전화번호를 변경하고 확인된 것으로 표시할 수도 있습니다. AdminUpdateUserAttributesAPI 또는 admin-update-user-attributes AWS

CLI 명령을 사용하여 전화번호를 확인된 것으로 표시합니다. 관리자는 이 명령을 사용하여 phone_number_verified 속성을 true로 변경할 수 있습니다. 의 사용자 탭에서 사용자를 AWS Management Console 편집하여 전화번호를 확인된 것으로 표시할 수도 있습니다.

> **Important**
>
> 전화 번호는 다음과 같은 형식 규칙을 따라야 합니다. 전화 번호는 더하기(+) 기호로 시작하고 바로 뒤에 국가 코드가 와야 합니다. + 기호와 숫자만 전화 번호에 포함할 수 있습니다. 서비스에 값을 제출하기 전에 전화 번호에서 괄호, 공백 또는 대시(-)와 같은 다른 문자는 모두 제거합니다. 예를 들어 미국 기반 전화 번호는 **+14325551212** 형식을 따라야 합니다.

preferred_username

preferred_username을 필수 또는 별칭으로 선택할 수 있지만 둘 다로 선택할 수는 없습니다. preferred_username가 별칭인 경우 UpdateUserAttributesAPI 작업에 요청을 보내고 사용자를 확인한 후 속성 값을 추가할 수 있습니다.

sub

sub 속성을 기반으로 사용자를 인덱싱하고 검색합니다. sub 속성은 각 사용자 풀 내의 고유한 사용자 식별자입니다. 사용자는 phone_number 및 email와 같은 속성을 변경할 수 있습니다. sub 속성에는 고정 값이 있습니다. 사용자 검색에 대한 자세한 내용은 사용자 계정 관리 및 검색을 참조하십시오.

필수 속성 보기

지정된 사용자 풀의 필수 속성을 보려면 다음 절차를 따르세요.

> **Note**
>
> 사용자 풀을 생성한 후에는 필수 속성을 변경할 수 없습니다.

필수 속성을 보려면

1. 에서 Amazon Cognito로 이동합니다. AWS Management Console콘솔에 메시지가 표시되면 자격 증명을 입력합니다. AWS
2. 사용자 풀(User Pools)을 선택합니다.

3. 목록에서 기존 사용자 풀을 선택합니다.
4. 가입 환경(Sign-up experience) 탭을 선택합니다.
5. 필수 속성(Required attributes) 섹션에서 사용자 풀의 필수 속성을 봅니다.

사용자 이름 및 기본 설정 사용자 이름

`username` 값은 별도의 속성이며 `name` 속성과 동일하지 않습니다. 각 사용자는 `username` 속성이 있습니다. Amazon Cognito는 페더레이션 사용자의 사용자 이름을 자동으로 생성합니다. Amazon Cognito 디렉터리에서 로컬 사용자를 생성하려면 `username` 속성을 제공해야 합니다. 사용자를 생성한 후에는 `username` 속성의 값을 변경할 수 없습니다.

개발자는 `preferred_username` 속성을 사용하여 사용자에게 변경할 수 있는 사용자 이름을 제공할 수 있습니다. 자세한 정보는 로그인 속성 사용자 지정을 참조하세요.

애플리케이션에 사용자 이름이 필요하지 않은 경우, 사용자에게 사용자 이름 제공을 요청할 필요가 없습니다. 앱이 백그라운드에서 고유한 사용자 이름을 생성할 수 있습니다. 이는 사용자가 이메일 주소와 암호를 사용하여 등록 및 로그인하게 하려는 경우 유용할 수 있습니다. 자세한 내용은 로그인 속성 사용자 지정 섹션을 참조하세요.

`username`은 사용자 풀에서 고유해야 합니다. `username`은 삭제되고 더 이상 사용되지 않는 경우에만 재사용될 수 있습니다. 속성에 대한 문자열 제약 조건에 대한 자세한 내용은 SignUpAPI 요청의 username *`username`* 속성을 참조하십시오.

로그인 속성 사용자 지정

사용자 풀을 생성할 때 사용자 이름 속성을 설정하여 사용자가 이메일 주소 또는 전화번호를 사용자 이름으로 사용하여 가입하고 로그인하게 할 수 있습니다. 또는 별칭 속성을 설정하여 사용자에게 선택지를 제공할 수도 있습니다. 사용자는 가입할 때 여러 속성을 포함한 다음 사용자 이름, 기본 설정 사용자 이름, 이메일 주소 또는 전화번호로 로그인할 수 있습니다.

> **Important**
>
> 사용자 풀이 생성된 후에는 이 설정을 변경할 수 없습니다.

별칭 속성과 사용자 이름 속성 중에서 선택하는 방법

사용자의 요구 사항	별칭 속성	사용자 이름 속성
사용자는 여러 로그인 속성을 가집니다.	예[1]	넘버 [2]
사용자는 먼저 이메일 주소 또는 전화번호를 확인해야 로그인할 수 있습니다.	예	아니요
중복된 이메일 주소 또는 전화번호로 사용자를 등록하고 `UsernameExistsException` 오류를 방지하세요[3]	예	아니요
두 명 이상의 사용자에게 동일한 이메일 주소 또는 전화번호 속성 값을 할당할 수 있습니다.	예[4]	아니요

[1] 사용 가능한 로그인 속성은 사용자 이름, 이메일 주소, 전화번호 및 기본 설정 사용자 이름입니다.

[2] 이메일 주소나 전화번호로 로그인할 수 있습니다.

[3] 사용자가 중복될 가능성이 있는 이메일 주소 또는 전화번호로 등록하지만 사용자 이름은 등록하지 않는 경우 사용자 풀에서 `UsernameExistsException` 오류가 생성되지 않습니다. 이 동작은 사용자 이름 존재 오류 방지와는 별개로, 로그인 작업에는 적용되지만 가입 작업에는 적용되지 않습니다.

[3] 속성을 검증한 마지막 사용자만 해당 속성을 사용하여 로그인할 수 있습니다.

옵션 1: 다중 로그인 속성(별칭 속성)

사용자가 로그인할 때 사용자 이름 또는 기타 속성 값 입력 중 선택하도록 허용하려는 경우, 별칭을 활성화할 수 있습니다. 기본적으로 사용자는 사용자 이름과 암호를 사용하여 로그인합니다. 사용자 이름은 사용자가 변경할 수 없는 고정 값입니다. 속성을 별칭으로 표시하면 사용자가 사용자 이름 대신 해당 속성을 사용하여 로그인할 수 있습니다. 이메일 주소, 전화 번호 및 기본 설정 사용자 이름 속성을 별칭으로 표시할 수 있습니다. 예를 들어 이메일 주소 및 전화번호를 사용자 풀 별칭으로 선택하는 경우 해당 사용자 풀의 사용자는 자신의 사용자 이름, 이메일 주소 또는 전화번호를 암호와 함께 사용하여 로그인할 수 있습니다.

별칭 특성을 선택하려면 사용자 풀을 만들 때 User name(사용자 이름)을 선택하고 하나 이상의 추가 로그인 옵션을 선택합니다.

> **Note**
>
> 사용자 풀을 대/소문자를 구분하지 않도록 구성하면 사용자가 소문자 또는 대문자를 사용하여 자신의 별칭으로 가입하거나 로그인할 수 있습니다. 자세한 내용은 Amazon Cognito 사용자 풀 API 참조를 참조하십시오 CreateUserPool.

이메일 주소를 별칭으로 선택하는 경우 Amazon Cognito는 유효한 이메일 주소 형식과 일치하는 사용자 이름을 수락하지 않습니다. 마찬가지로 전화번호를 별칭으로 선택하는 경우 Amazon Cognito는 유효한 전화번호 형식과 일치하는 해당 사용자 풀에 대한 사용자 이름을 수락하지 않습니다.

> **Note**
>
> 별칭 값은 사용자 풀에서 고유해야 합니다. 별칭을 이메일 주소 또는 전화 번호용으로 구성하는 경우 제공하는 값은 하나의 계정에서만 확인됨 상태가 될 수 있습니다. 가입하는 동안 사용자가 이메일 주소 또는 전화 번호를 별칭 값으로 제공하고 다른 사용자가 이미 해당 별칭 값을 사용한 경우 등록이 성공합니다. 그러나 사용자가 이 이메일(또는 전화 번호)로 계정 확인을 시도하고 유효한 코드를 입력하면 Amazon Cognito에서 `AliasExistsException` 오류를 반환합니다. 이 오류는 이 이메일 주소(또는 전화 번호)의 계정이 이미 있음을 사용자에게 나타냅니다. 이때 사용자는 새 계정 생성 시도를 중단하고 대신 이전 계정의 암호 재설정을 시도할 수 있습니다. 사용자가 계속해서 새 계정을 생성하는 경우 앱은 `forceAliasCreation` 옵션을 사용하여 `ConfirmSignUp` API를 호출해야 합니다. `ConfirmSignUp`과 `forceAliasCreation`은 함께 사용하여 이전 계정의 별칭을 새로 생성된 계정으로 이동하고 속성을 이전 계정에서 확인되지 않음으로 표시합니다.

전화 번호 및 이메일 주소는 사용자가 전화 번호 및 이메일 주소를 확인한 후에만 사용자에 대한 활성 별칭이 됩니다. 이메일 주소 및 전화 번호를 별칭으로 사용하는 경우 해당 이메일 주소 및 전화 번호의 자동 확인을 선택하는 것이 좋습니다.

사용자가 가입할 때 이메일 주소 및 전화번호 속성에 `UsernameExistsException` 오류가 발생하지 않도록, 별칭 속성을 선택하세요.

사용자의 `username` 속성 값이 변경되지 않는 동안 사용자가 로그인할 때 사용하는 사용자 이름을 사용자가 변경할 수 있도록 `preferred_username` 속성을 활성화합니다. 이 사용자 경험을 설정

하려는 경우 새 username 값을 preferred_username으로 제출하고 preferred_username을 별칭으로 선택합니다. 그러면 사용자가 자신이 입력한 새 값으로 로그인할 수 있습니다. preferred_username을 별칭으로 선택하는 경우 사용자는 계정을 확인할 때에만 값을 제공할 수 있습니다. 등록 중에는 값을 제공할 수 없습니다.

사용자가 사용자 이름을 사용하여 가입할 때, 다음 별칭 중 하나 이상을 사용하여 로그인할 수 있는지 여부를 선택할 수 있습니다.

- 확인된 이메일 주소
- 확인된 전화 번호
- 기본 설정 사용자 이름

사용자가 가입한 후에 이러한 별칭을 변경할 수 있습니다.

> **Important**
>
> 사용자 풀에서 별칭을 사용한 로그인을 지원하고 사용자를 인증하거나 검색하려는 경우 로그인 속성으로 사용자를 식별하지 마십시오. 고정 값 사용자 식별자 sub은 사용자 ID를 나타내는 유일한 일관된 지표입니다.

사용자가 별칭을 사용하여 로그인할 수 있도록 사용자 풀을 생성할 때 다음 단계를 포함합니다.

사용자가 기본 설정 사용자 이름을 사용하여 로그인할 수 있도록 사용자 풀을 구성하려면

1. AWS Management Console의 Amazon Cognito로 이동합니다. 콘솔에 메시지가 표시되면 자격 증명을 입력합니다. AWS
2. 사용자 풀(User Pools)을 선택합니다.
3. 페이지 오른쪽 상단에서 [사용자 풀 생성(Create a user pool)]을 선택하여 사용자 풀 생성 마법사를 시작합니다.
4. [로그인 환경 구성(Configure sign-in experience)]에서 사용자 풀에 연결할 자격 증명 [공급자 유형(Provider types)]을 선택합니다.
5. Cognito 사용자 풀 로그인 옵션(Cognito user pool sign-in options)에서 사용자 이름(User name), 이메일(Email), 전화 번호(Phone number)를 원하는 대로 조합해서 선택합니다.
6. 사용자 이름 요구 사항에서 사용자가 로그인할 때 대체 사용자 이름을 설정할 수 있도록 기본 설정 사용자 이름을 사용한 사용자 로그인 허용을 선택합니다.

7. 다음(Next)을 선택한 다음, 마법사의 모든 단계를 완료합니다.

옵션 2: 로그인 속성으로 사용하는 이메일 주소 또는 전화번호(사용자 이름 속성)

사용자가 이메일 주소 또는 전화번호를 사용자 이름으로 사용하여 가입할 때 사용자가 이메일 주소와 전화번호 중 하나만 사용하여 가입할 수 있는지 여부를 선택할 수 있습니다.

사용자 이름 특성을 선택하려면 사용자 풀을 만들 때 사용자 이름을 로그인 옵션으로 선택하지 마세요.

이메일 주소 또는 전화 번호는 고유해야 하며 다른 사용자가 이미 사용하고 있어서는 안 됩니다. 확인을 받을 필요는 없습니다. 사용자가 이메일 주소 또는 전화 번호를 사용하여 가입한 후에는 동일한 이메일 주소 또는 전화 번호로 새 계정을 생성할 수 없습니다. 사용자는 기존 계정을 재사용하고 필요한 경우 계정 암호를 재설정할 수만 있습니다. 그러나 사용자는 이메일 주소 또는 전화 번호를 새 이메일 주소 또는 전화 번호로 변경할 수 있습니다. 이메일 주소 또는 전화번호가 아직 사용 중이 아닌 경우 새 사용자 이름이 됩니다.

> **Note**
>
> 사용자가 이메일 주소를 사용자 이름으로 사용하여 가입하는 경우 사용자 이름을 다른 이메일 주소로 변경할 수 있으나 전화번호로 변경할 수는 없습니다. 사용자가 전화번호를 사용하여 가입하는 경우 사용자 이름을 다른 전화번호로 변경할 수 있지만 이메일 주소로 변경할 수는 없습니다.

이메일 주소 또는 전화 번호를 사용하는 가입 및 로그인을 설정하려면 사용자 풀 생성 프로세스 동안 다음 단계를 사용합니다.

이메일 주소 또는 전화 번호를 사용하여 가입 및 로그인할 수 있도록 사용자 풀을 구성하려면

1. AWS Management Console의 Amazon Cognito로 이동합니다. 콘솔에 메시지가 표시되면 자격 증명을 입력합니다. AWS
2. 사용자 풀(User Pools)을 선택합니다.
3. 페이지 오른쪽 상단에서 사용자 풀 생성(Create a user pool)을 선택하여 사용자 풀 생성 마법사를 시작합니다.
4. Cognito user pool sign-in options(Cognito 사용자 풀 로그인 옵션)에서 사용자가 로그인하는 데 사용할 수 있는 속성을 나타내는 아무 Email(이메일) 및 Phone number(전화 번호) 조합을 선택합니다.
5. Next(다음)을 선택한 다음 마법사의 나머지 단계를 완료합니다.

> **Note**
>
> 이메일 주소 또는 전화 번호를 사용자 풀의 필수 속성으로 지정할 필요가 없습니다.

앱에서 옵션 2를 실행하려면

1. `CreateUserPool` API를 호출하여 사용자 풀을 생성합니다. `UserNameAttributes` 파라미터를 `phone_number`, `email` 또는 `phone_number | email`로 설정합니다.
2. `SignUp` API를 호출하고 API의 `username` 파라미터에서 이메일 주소나 전화 번호를 전달합니다. 이 API는 다음을 수행합니다.
   - `username` 문자열이 유효한 이메일 주소 형식인 경우 사용자 풀에서 사용자의 `email` 속성을 `username` 값으로 자동으로 채웁니다.
   - `username` 문자열이 유효한 전화 번호 형식이면 사용자 풀에서 사용자의 `phone_number` 속성이 `username` 값으로 자동으로 채워집니다.
   - `username` 문자열 형식이 이메일 주소 또는 전화 번호 형식이 아닌 경우 `SignUp` API에서 예외를 반환합니다.
   - `SignUp` API는 사용자에 대해 영구적인 UUID를 생성하고 내부적으로 이를 변경 불가능한 사용자 이름 속성으로 사용합니다. 이 UUID는 사용자 ID 토큰의 `sub` 클레임과 동일한 값을 갖습니다.
   - `username` 문자열에 이미 사용 중인 이메일 주소 또는 전화 번호가 포함되어 있는 경우 `SignUp` API에서 예외가 발생합니다.

`ListUsers` API를 제외하고 모든 API에서 사용자 이름 자리에 별칭으로 이메일 주소나 전화 번호를 사용할 수 있습니다. `ListUsers`에게 전화를 걸 때 `email` 또는 `phone_number` 속성으로 검색할 수 있습니다. `username`으로 검색하는 경우 별칭이 아닌 실제 사용자 이름을 제공해야 합니다.

사용자 지정 속성

사용자 풀에 사용자 정의 속성을 최대 50개까지 추가할 수 있습니다. 사용자 지정 속성의 최소 및/또는 최대 길이를 지정할 수 있습니다. 단, 사용자 지정 속성의 최대 길이는 2,048자를 넘을 수 없습니다.

각 사용자 지정 속성의 특성은 다음과 같습니다.

- 사용자 지정 속성을 문자열 또는 숫자로 정의할 수 있습니다. Amazon Cognito는 사용자 지정 속성 값을 ID 토큰에 문자열로만 씁니다.

- 사용자에게 이 속성의 값을 제공하도록 요구할 수 없습니다.
- 이 속성을 사용자 풀에 추가한 후에는 제거하거나 변경할 수 없습니다.
- 이 속성 이름의 문자 길이는 Amazon Cognito가 허용하는 한도 내에 있습니다. 자세한 정보는 Amazon Cognito의 할당량을 참조하세요.
- 이 속성은 변경 가능 또는 변경 불가능일 수 있습니다. 사용자를 생성할 때만 변경 불가능한 사용자 지정 속성에 값을 쓸 수 있습니다. 앱 클라이언트에 속성에 대한 쓰기 권한이 있는 경우 변경 가능한 속성의 값을 변경할 수 있습니다. 자세한 내용은 속성 권한 및 범위 섹션을 참조하세요.

> **Note**
>
> 코드에서 그리고 역할 기반 액세스 제어 사용에 대한 규칙 설정에서는 사용자 지정 속성을 표준 속성과 구별하기 위해 사용자 지정 속성에 `custom:` 접두사가 필요합니다.

사용자 풀을 생성할 때 의 CreateUserPool속성에 개발자 속성을 추가할 수도 `SchemaAttributes` 있습니다. 개발자 속성에는 `dev:` 접두사가 있습니다. AWS 자격 증명을 사용하여 사용자의 개발자 속성만 수정할 수 있습니다. 개발자 속성은 Amazon Cognito가 앱 클라이언트 읽기-쓰기 권한으로 대체한 레거시 특성입니다.

새 사용자 정의 속성을 생성하려면 다음 절차를 따르세요.

콘솔을 사용하여 사용자 지정 속성을 추가하려면

1. 에서 Amazon Cognito로 이동합니다. AWS Management Console콘솔에 메시지가 표시되면 자격 증명을 입력합니다. AWS
2. 사용자 풀(User Pools)을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택합니다.
4. [가입 환경(Sign-up experience)] 탭을 선택하고 [사용자 정의 속성(Custom attributes)] 섹션에서 [사용자 정의 속성 추가(Add custom attributes)]를 선택합니다.
5. [사용자 정의 속성 추가(Add custom attributes)] 페이지에서 새 속성에 대한 다음 세부 정보를 제공합니다.
   - 이름을 입력합니다.
   - 문자열 또는 숫자의 형식을 선택합니다.
   - 최소 문자열 길이 또는 숫자 값을 입력합니다.

- 최대 문자열 길이 또는 숫자 값을 입력합니다.
- 사용자에게 사용자 지정 속성의 초기 값을 설정한 후 해당 값을 변경할 수 있는 권한을 제공하려는 경우 변경 가능(Mutable)을 선택합니다.

6. 변경 사항 저장(Save changes)을 선택합니다.

속성 권한 및 범위

각 앱 클라이언트에 대해 각 사용자 속성의 읽기 및 쓰기 권한을 설정할 수 있습니다. 이렇게 하면 모든 앱이 사용자를 위해 저장되는 각 속성을 읽고 수정하기 위해 갖고 있어야 할 액세스 권한을 제어할 수 있습니다. 예를 들어, 사용자가 유료 고객인지 여부를 나타내는 사용자 지정 속성이 있을 수 있습니다. 앱에서 이 속성을 볼 수 있지만 직접 변경할 수는 없습니다. 대신 관리 도구 또는 백그라운드 프로세스를 사용하여 이 속성을 업데이트합니다. Amazon Cognito 콘솔, API 또는 AWS CLI에서 사용자 속성에 대한 권한을 설정할 수 있습니다. 기본적으로 새 사용자 지정 속성은 해당 속성에 대한 읽기 및 쓰기 권한이 설정된 후에만 사용할 수 있습니다. 기본적으로 새 앱 클라이언트를 만들면 앱에 모든 표준 및 사용자 지정 속성에 대한 읽기 및 쓰기 권한을 부여합니다. 앱이 필요한 양의 정보만 사용하도록 제한하려면 앱 클라이언트 구성의 속성에 특정 권한을 할당합니다.

가장 좋은 방법은 앱 클라이언트를 만들 때 속성 읽기 및 쓰기 권한을 지정하는 것입니다. 애플리케이션 운영에 필요한 최소 사용자 속성 세트에 대한 액세스 권한을 앱 클라이언트에 부여하십시오.

> **Note**
>
> DescribeUserPoolClient기본값이 아닌 앱 클라이언트 권한을 구성한 경우에만 값을 반환합니다. `ReadAttributes` `WriteAttributes`

속성 권한을 업데이트하려면(AWS Management Console)

1. 에서 Amazon Cognito로 이동합니다. AWS Management Console콘솔에 메시지가 표시되면 자격 증명을 입력합니다. AWS
2. 사용자 풀(User Pools)을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택합니다.
4. [앱 통합(App integration)] 탭을 선택한 다음, [앱 클라이언트(App clients)] 섹션의 목록에서 앱 클라이언트를 선택합니다.
5. [속성 읽기 및 쓰기 권한(Attribute read and write permissions)] 섹션에서 [편집(Edit)]을 선택합니다.

6. 속성 읽기 및 쓰기 권한 편집(Edit attribute read and write permissions) 페이지에서 읽기 및 쓰기 권한을 구성한 다음 변경 사항 저장(Save changes)을 선택합니다.

사용자 지정 속성을 사용하는 각 앱 클라이언트에 대해 이러한 단계를 반복합니다.

각 앱에 대해 속성을 읽기 가능 또는 쓰기 가능으로 표시할 수 있습니다. 표준 속성과 사용자 정의 속성에 모두 적용됩니다. 앱에서 읽기 가능으로 표시한 속성 값을 검색하고 쓰기 가능으로 표시한 속성 값을 설정하거나 수정할 수 있습니다. 앱이 쓰기 권한이 없는 속성에 값을 설정하려고 하면 Amazon Cognito가 `NotAuthorizedException`을 반환합니다. GetUser요청에는 앱 클라이언트 클레임이 있는 액세스 토큰이 포함됩니다. Amazon Cognito는 앱 클라이언트가 읽을 수 있는 속성 값만 반환합니다. 앱의 사용자 ID 토큰에는 읽기 가능한 속성에 해당하는 클레임만 포함됩니다. 모든 앱 클라이언트는 사용자 풀에 필요한 속성을 작성할 수 있습니다. 아직 값이 없는 필수 속성에도 값을 제공하는 경우에만 Amazon Cognito 사용자 풀 API 요청에서 속성 값을 설정할 수 있습니다.

사용자 지정 속성에는 읽기 및 쓰기 권한에 대한 고유한 기능이 있습니다. 사용자 풀에 대해 변경 가능 또는 변경 불가능으로 만들 수 있으며 모든 앱 클라이언트에 대해 읽기 또는 쓰기 속성으로 설정할 수 있습니다.

변경 불가능한 사용자 지정 속성은 사용자 생성 중에 한 번 업데이트할 수 있습니다. 다음 방법으로 변경 불가능한 속성을 채울 수 있습니다.

- `SignUp`: 사용자가 변경 불가능한 사용자 지정 속성에 대한 쓰기 권한이 있는 앱 클라이언트에 가입합니다. 해당 속성에 대한 값을 제공합니다.
- 서드 파티 IdP로 로그인: 사용자가 변경 불가능한 사용자 지정 속성에 대한 쓰기 권한이 있는 앱 클라이언트에 로그인합니다. 해당 IdP의 사용자 풀 구성에는 제공된 클레임을 변경 불가능한 속성에 매핑하는 규칙이 있습니다.
- `AdminCreateUser`: 변경 불가능한 속성에 값을 입력합니다.

앱 클라이언트에 할당할 수 있는 범위에 대한 자세한 내용은 리소스 서버를 통한 범위, M2M 및 API 인증 섹션을 참조하세요.

사용자 풀을 생성한 후 속성 권한과 범위를 변경할 수 있습니다.

사용자 풀 암호 요구 사항 추가

강력하고 복잡한 암호는 사용자 풀의 보안 모범 사례입니다. 특히 인터넷에 개방된 응용 프로그램에서 취약한 암호는 암호를 추측하여 데이터에 액세스하려는 시스템에 사용자의 자격 증명을 노출시킬 수

있습니다. 암호가 복잡할수록 추측하기가 더 어려워집니다. Amazon Cognito에는 고급 보안 기능 및 AWS WAF웹 ACL과 같이 보안에 민감한 관리자를 위한 추가 도구가 있지만 암호 정책은 사용자 디렉토리 보안의 핵심 요소입니다.

Amazon Cognito 사용자 풀의 로컬 사용자 암호는 자동으로 만료되지 않습니다. 가장 좋은 방법은 사용자 암호 재설정의 시간, 날짜 및 메타데이터를 외부 시스템에 기록하는 것입니다. 암호 사용 기간의 외부 로그를 사용하면 애플리케이션 또는 Lambda 트리거가 사용자의 암호 사용 기간을 조회하고 지정된 기간이 지나면 재설정을 요구할 수 있습니다.

보안 표준을 준수하는 최소 암호 복잡성을 요구하도록 사용자 풀을 구성할 수 있습니다. 복잡한 암호의 최소 길이는 8자 이상이어야 합니다. 또한 대문자, 숫자 및 특수 문자가 혼합되어 포함됩니다.

사용자 풀 암호 정책 설정

1. 사용자 풀을 생성하고 보안 요구 사항 구성 단계로 이동하거나 기존 사용자 풀에 액세스하여 로그인 경험 탭으로 이동합니다.
2. 암호 정책으로 이동합니다.
3. 암호 정책 모드를 선택합니다. Cognito 기본값은 권장 최소 설정으로 사용자 풀을 구성합니다. 사용자 지정 암호 정책을 선택할 수도 있습니다.
4. 암호 최소 길이를 설정합니다. 모든 사용자는 이 값보다 크거나 같은 길이의 암호를 등록하거나 생성해야 합니다. 이 최솟값을 99까지 설정할 수 있지만 사용자는 최대 256자의 암호를 설정할 수 있습니다.
5. 암호 요구 사항에서 암호 복잡성 규칙을 구성하십시오. 각 사용자 암호에 필요한 문자 유형(숫자, 특수 문자, 대문자, 소문자)을 선택합니다.

   암호에 다음 문자 중 하나 이상을 요구할 수 있습니다. Amazon Cognito에서 암호에 필요한 최소 문자가 포함되어 있는지 확인한 후 사용자 암호에는 최대 암호 길이까지 모든 유형의 추가 문자가 포함될 수 있습니다.

   - 기본 라틴 문자의 대문자와 소문자
   - 숫자
   - 다음은 특수 문자입니다.

     ```
     ^ $ * . [ ] { } ( ) ? " ! @ # % & / \ , > < ' : ; | _ ~ ` = + -
     ```

   - 비선행 비후행 공백 문자
6. 관리자가 설정한 임시 암호의 만료 기간을 설정합니다. 이 기간이 지나면 Amazon Cognito 콘솔에서 또는 `AdminCreateUser`를 통해 생성한 새 사용자는 로그인하거나 새 암호를 설정할

수 없습니다. 임시 암호로 로그인한 후에는 사용자 계정이 만료되지 않습니다. Amazon Cognito 사용자 풀 API에서 암호 기간을 업데이트하려면 CreateUserPool또는 UpdateUserPoolAPI TemporaryPasswordValidityDays 요청에서 값을 설정하십시오.

- 만료된 사용자 계정의 액세스 권한을 재설정하려면 다음 중 하나를 수행합니다.
  - 사용자 프로필을 삭제하고 새 프로필을 생성합니다.
  - AdminSetUserPasswordAPI 요청에서 새 영구 암호를 설정합니다.
  - AdminResetUserPasswordAPI 요청에서 새 확인 코드를 생성하십시오.

Amazon Cognito 사용자 풀에 대한 이메일 설정

사용자 풀에 대한 클라이언트 앱의 특정 이벤트로 인해 Amazon Cognito에서 사용자에게 이메일을 보낼 수 있습니다. 예를 들어 이메일 확인을 요구하도록 사용자 풀을 구성한 경우 사용자가 앱에서 새 계정을 등록하거나 암호를 다시 설정하면 Amazon Cognito에서 이메일을 보냅니다. 이메일을 시작한 작업이 무엇인지에 따라 이메일에는 확인 코드가 들어가거나 임시 암호가 들어갑니다.

이메일 전송을 처리하려면 다음 옵션 중 하나를 사용할 수 있습니다.

- Amazon Cognito 서비스에 내장된 기본 이메일 구성입니다.
- Amazon Simple Email Service(Amazon SES) 구성

사용자 풀을 생성한 후 전송 옵션을 변경할 수 있습니다.

Amazon Cognito는 사용자가 입력할 수 있는 코드나 선택할 수 있는 URL 링크가 포함된 이메일 메시지를 사용자에게 보냅니다. 다음 표에서는 이메일 메시지를 생성할 수 있는 이벤트를 보여줍니다.

메시지 옵션

활동	API 작업	전송 옵션	포맷 옵션	사용자 지정 가능	메시지 템플릿
암호 찾기	ForgotPassword	이메일, SMS	code	아니요	N/A
초대	AdminCreateUser	이메일, SMS	code	예	초대 메시지
셀프 등록	SignUp	이메일, SMS	코드, 링크	예	확인 메시지

활동	API 작업	전송 옵션	포맷 옵션	사용자 지정 가능	메시지 템플릿
이메일 주소 또는 전화번호 인증	UpdateUserAttributes	이메일, SMS	code	예	확인 메시지
멀티 팩터 인증(MFA)	AdminInitiateAuth, InitiateAuth	SMS	code	예[1]	MFA 메시지

[1] SMS 메시지의 경우.

Amazon SES는 이메일 메시지에 대해 요금을 부과합니다. 자세한 내용은 Amazon SES 요금을 참조하세요.

기본 이메일 구성

Amazon Cognito는 기본 이메일 구성을 사용하여 사용자를 대신하여 이메일 전송을 처리할 수 있습니다. 기본 옵션을 사용하면 Amazon Cognito는 사용자 풀에 대해 매일 제한된 수의 이메일 보내기만 허용합니다. 서비스 제한에 대한 자세한 내용은 Amazon Cognito의 할당량 섹션을 참조하세요. 일반적인 프로덕션 환경의 경우 기본 이메일 제한은 필수 전송 볼륨보다 적습니다. 더 많은 전송량을 활성화하려면 Amazon SES 이메일 구성을 사용할 수 있습니다.

기본 구성을 사용할 때는 에서 관리하는 Amazon SES 리소스를 사용하여 이메일 메시지를 전송합니다. AWS Amazon SES는 하드 바운스를 반환하는 이메일 주소를 계정 수준 억제 목록 또는 글로벌 억제 목록에 추가합니다. 전송할 수 없는 이메일 주소를 나중에 전송할 수 있게 되더라도 사용자 풀이 기본 구성을 사용하도록 구성된 동안에는 금지 목록에서 해당 주소가 제거되도록 제어할 수 없습니다. 이메일 주소는 -managed 금지 목록에 무기한 남아 있을 수 있습니다. AWS전달 불가능한 이메일 주소를 관리하려면 다음 섹션의 설명대로 Amazon SES 이메일 구성을 계정 수준 억제 목록과 함께 사용합니다.

기본 이메일 구성을 사용하면 다음 이메일 주소 중 하나를 발신 주소로 사용할 수 있습니다.

- 기본 이메일 주소 no-reply@verificationemail.com입니다.
- 사용자 지정 이메일 주소. 자신의 이메일 주소를 사용하려면 먼저 Amazon SES로 확인해야 하며 이 주소를 사용할 수 있는 권한을 Amazon Cognito에 부여해야 합니다.

Amazon SES 이메일 구성

애플리케이션에 기본 옵션으로 제공되는 것보다 더 많은 전송 볼륨이 필요할 수도 있습니다. 가능한 전송량을 늘리려면 Amazon SES 리소스를 사용자 풀과 함께 사용하여 사용자에게 이메일을 보냅니다. 자체 Amazon SES 구성을 사용하여 이메일 메시지를 보낼 때 이메일 보내기 활동을 모니터링할 수도 있습니다.

Amazon SES 구성을 사용하려면 먼저 Amazon SES로 하나 이상의 이메일 주소 또는 도메인을 확인해야 합니다. 확인된 이메일 주소 또는 확인된 도메인에 속한 주소를 사용자 풀에 할당한 발신 이메일 주소로 사용합니다. Amazon Cognito에서 사용자에게 이메일을 보내면 Amazon SES가 호출되고 이메일 주소를 사용합니다.

Amazon SES 구성을 사용할 때 다음 조건이 적용됩니다.

- 사용자 풀에 대한 이메일 전송 제한은 AWS 계정에서 Amazon SES 확인 이메일 주소에 적용되는 것과 동일한 제한입니다.
- 글로벌 억제 목록을 재정의하는 Amazon SES의 계정 수준 억제 목록을 사용하여 전달 불가능한 이메일 주소에 대한 메시지를 관리할 수 있습니다. 계정 수준 억제 목록을 사용할 때 이메일 메시지 반송은 발신자로서의 계정 평판에 영향을 미칩니다. 자세한 내용은 Amazon Simple Email Service 개발자 안내서의 Amazon SES 계정 수준 금지 목록 사용을 참조하세요.

Amazon SES 이메일 구성 리전

사용자 풀을 생성하는 AWS 리전 위치에는 Amazon SES를 통한 이메일 메시지 구성을 위한 세 가지 요구 사항 중 하나가 적용됩니다. 사용자 풀과 동일한 지역, 동일한 지역을 포함한 여러 지역 또는 하나 이상의 원격 지역에 있는 Amazon SES로부터 이메일 메시지를 보낼 수 있습니다. 최상의 성능을 위해 옵션이 있는 경우 사용자 풀과 동일한 지역에서 Amazon SES 인증 ID를 사용하여 이메일 메시지를 보내십시오.

Amazon SES 검증 자격 증명에 대한 지역 요구 사항 카테고리

지역 내 전용

사용자 풀은 사용자 풀과 동일하게 AWS 리전 확인된 ID로 이메일 메시지를 보낼 수 있습니다. 사용자 지정 이메일 주소가 없는 기본 `FROM` 이메일 구성에서 Amazon Cognito는 동일한 지역의 `no-reply@verificationemail.com` 검증된 ID를 사용합니다.

이전 버전과 호환됩니다.

사용자 풀은 동일한 지역 AWS 리전 또는 다음 대체 지역 중 하나에서 확인된 ID를 사용하여 이메일 메시지를 보낼 수 있습니다.

- 미국 동부(버지니아 북부)
- US West (Oregon)
- 유럽(아일랜드)

이 기능은 서비스 시작 시 Amazon Cognito 요구 사항에 맞게 생성했을 수 있는 사용자 풀 리소스의 연속성을 지원합니다. 해당 기간의 사용자 풀은 제한된 수의 검증된 자격 증명을 포함하는 이메일 메시지만 보낼 수 있었습니다. AWS 리전사용자 지정 이메일 주소가 없는 기본 `FROM` 이메일 구성에서 Amazon Cognito는 동일한 지역의 `no-reply@verificationemail.com` 검증된 ID를 사용합니다.

대체 지역

사용자 풀은 사용자 풀 지역 외부의 대체 AWS 리전 수단으로 확인된 ID가 포함된 이메일 메시지를 보낼 수 있습니다. 이 구성은 Amazon Cognito를 사용할 수 있는 지역에서 Amazon SES를 사용할 수 없을 때 발생합니다.

대체 지역의 확인된 자격 증명에 대한 Amazon SES 전송 권한 부여 정책은 시작 지역의 Amazon Cognito 서비스 보안 주체를 신뢰해야 합니다. 자세한 정보는 기본 이메일 구성을 사용할 수 있는 권한을 부여하려면을 참조하세요.

일부 지역에서는 Amazon Cognito가 이메일 메시지를 두 개의 대체 지역 간에 분할하여 기본 이메일 구성을 적용합니다. `COGNITO_DEFAULT` 이러한 경우 사용자 지정 `FROM` 이메일 주소를 사용하려면 각 대체 지역의 확인된 자격 증명에 대한 Amazon SES 전송 권한 부여 정책이 발신 지역의 Amazon Cognito 서비스 보안 주체를 신뢰해야 합니다. 자세한 정보는 기본 이메일 구성을 사용할 수 있는 권한을 부여하려면을 참조하세요. 이러한 지역에서 Amazon SES 이메일을 구성하려면 첫 번째로 나열된 지역에서 검증된 자격 증명을 사용하고 사용자 풀 지역의 Amazon Cognito 서비스 보안 주체를 신뢰하도록 구성해야 합니다. `DEVELOPER` 예를 들어 중동 (UAE) 의 사용자 풀에서 유럽 (프랑크푸르트) 의 검증된 자격 증명을 신뢰할 수 있도록 구성합니다. `cognito-idp.me-central-1.amazonaws.com` 사용자 지정 이메일 주소가 없는 기본 `FROM` 이메일 구성에서 Amazon Cognito는 각 지역의 `no-reply@verificationemail.com` 검증된 ID를 사용합니다.

> **Note**
>
> 다음과 같은 조건 조합에서는 Region 요소에 와일드카드를 EmailConfiguration사용하여 의 SourceArn 파라미터를 형식으로 지정해야 합니다. `arn:${Partition}:ses:*:${Account}:identity/${IdentityName}` 이렇게 하면 사용자 풀이 양쪽 모두에서 동일한 확인된 ID를 사용하여 이메일 메시지를 보낼 수 AWS 계정 있습니다. AWS 리전
>
> - 당신의 것입니다. EmailSendingAccount `COGNITO_DEFAULT`
> - 사용자 지정 `FROM` 주소를 사용하고 싶습니다.
> - 사용자 풀은 대체 지역에서 이메일을 보냅니다.
> - 사용자 풀에는 다음 Amazon SES 지원 지역 표에 지정된 두 번째 [1]대체 지역이 있습니다.

SDK, Amazon Cognito API 또는 CLI AWS AWS CDK, 또는 를 사용하여 프로그래밍 방식으로 사용자 풀을 생성하는 경우 사용자 풀은 의 AWS CloudFormation파라미터가 사용자 풀에 지정하는 Amazon `SourceArn` SES ID와 함께 이메일 메시지를 보냅니다. EmailConfiguration Amazon SES ID는 지원되는 AWS 리전 ID를 차지해야 합니다. `EmailSendingAccount`가 `COGNITO_DEFAULT`이고 `SourceArn` 파라미터가 지정되지 않은 경우 Amazon Cognito는 사용자 풀이 생성된 리전의 리소스를 사용하여 `no-reply@verificationemail.com`에서 이메일 메시지를 보냅니다.

다음 표는 Amazon Cognito에서 Amazon SES ID를 사용할 수 있는 AWS 리전 위치를 보여줍니다.

사용자 풀 지역	지역 옵션	아마존 SES 지원 지역
미국 동부(버지니아 북부)	이전 버전과 호환됩니다.	미국 서부(오레곤), 미국 동부(버지니아 북부), 유럽(아일랜드)
미국 동부(오하이오)	이전 버전과 호환	미국 서부(오하이오), 미국 동부(버지니아 북부), 미국 서부(오레곤), 유럽(아일랜드)
미국 서부(캘리포니아 북부)	지역 내 전용	미국 서부(캘리포니아 북부)

사용자 풀 지역	지역 옵션	아마존 SES 지원 지역
미국 서부(오레곤)	이전 버전과 호환됩니다.	미국 동부(버지니아 북부), 미국 서부(오레곤), 유럽(아일랜드)
캐나다(중부)	이전 버전과 호환	캐나다(중부), 미국 동부(버지니아 북부), 미국 서부(오레곤), 유럽(아일랜드)
아시아 태평양(도쿄)	이전 버전과 호환	아시아 태평양(도쿄), 미국 동부(버지니아 북부), 미국 서부(오레곤), 유럽(아일랜드)
아시아 태평양(서울)	이전 버전과 호환	아시아 태평양(서울), 미국 동부(버지니아 북부), 미국 서부(오레곤), 유럽(아일랜드)
아시아 태평양(뭄바이)	이전 버전과 호환	아시아 태평양(뭄바이), 미국 동부(버지니아 북부), 미국 서부(오레곤), 유럽(아일랜드)
아시아 태평양(하이데라바드)	대체 지역	아시아 태평양 (뭄바이), 아시아 태평양 (싱가포르) [1]
아시아 태평양(싱가포르)	이전 버전과 호환	아시아 태평양(싱가포르), 미국 동부(버지니아 북부), 미국 서부(오레곤), 유럽(아일랜드)
아시아 태평양(시드니)	이전 버전과 호환	아시아 태평양(시드니), 미국 동부(버지니아 북부), 미국 서부(오레곤), 유럽(아일랜드)
아시아 태평양(오사카)	지역 내 전용	아시아 태평양(오사카)
아시아 태평양(자카르타)	지역 내 전용	아시아 태평양(자카르타)
아시아 태평양(멜버른)	대체 지역	아시아 태평양 (시드니), 아시아 태평양 (싱가포르) [1]

사용자 풀 지역	지역 옵션	아마존 SES 지원 지역
유럽(아일랜드)	이전 버전과 호환	미국 서부(오레곤), 미국 동부(버지니아 북부), 유럽(아일랜드)
유럽(런던)	이전 버전과 호환	유럽(런던), 미국 동부(버니지아 북부), 미국 서부(오레곤), 유럽(아일랜드)
유럽(파리)	지역 내 전용	유럽(파리)
유럽(프랑크푸르트)	이전 버전과 호환됩니다.	유럽(프랑크푸르트), 미국 동부(버니지아 북부), 미국 서부(오레곤), 유럽(아일랜드)
유럽(취리히)	대체 지역	유럽 (프랑크푸르트), 유럽 (런던) [1]
유럽(스톡홀름)	지역 내 전용	유럽(스톡홀름)
유럽(밀라노)	지역 내 전용	유럽(밀라노)
유럽(스페인)	대체 지역	유럽 (파리), 유럽 (스톡홀름) [1]
중동(바레인)	지역 내 전용	중동(바레인)
중동(UAE)	대체 지역	유럽 (프랑크푸르트), 유럽 (런던) [1]
남아메리카(상파울루)	지역 내 전용	남아메리카(상파울루)
이스라엘(텔아비브)	지역 내 전용	이스라엘(텔아비브)
아프리카(케이프타운)	지역 내 전용	아프리카(케이프타운)

[1] 기본 이메일 구성의 사용자 풀에 사용됩니다. Amazon Cognito는 각 리전에서 동일한 이메일 주소를 사용하여 확인된 ID 간에 이메일 메시지를 배포합니다. 사용자 지정 FROM 주소

를 사용하려면 해당 형식의 EmailConfiguration SourceArn 파라미터로 구성하십시오. arn:*${Partition}*:ses:*:*${Account}*:identity/*${IdentityName}*

사용자 풀에 대한 이메일 구성

사용자 풀에 대한 이메일 설정을 구성하려면 다음 단계를 완료하세요. 사용하는 설정에 따라 Amazon SES, AWS Identity and Access Management (IAM) 및 Amazon Cognito에서 IAM 권한이 필요할 수도 있습니다.

> **Note**
>
> 이러한 단계에서 생성되는 리소스는 AWS 계정간에 공유할 수 없습니다. 예를 들어 사용자 풀을 구성한 한 계정을 다른 계정의 Amazon SES 이메일 주소로 사용할 수 없습니다. Amazon Cognito를 여러 계정에서 사용하는 경우 각 계정에 대해 이러한 단계를 반복합니다.

1단계: Amazon SES로 이메일 주소 또는 도메인 확인

사용자 풀을 구성하기 전에 다음 중 하나를 수행하려면 Amazon SES로 하나 이상의 도메인 또는 이메일 주소를 확인해야 합니다.

- 자신의 이메일 주소를 발신 주소로 사용합니다.
- Amazon SES 구성을 사용하여 이메일 전송을 처리합니다.

이메일 주소나 도메인을 확인하여 이메일 주소를 소유하고 있음을 확인하면 무단 사용을 방지하는 데 도움이 됩니다.

Amazon SES에서 이메일 확인에 관한 자세한 내용은 Amazon Simple Email Service 개발자 가이드에서 Amazon SES에서 이메일 주소 확인을 참조하세요. Amazon SES로 도메인을 확인하는 방법에 대한 자세한 내용은 도메인 확인을 참조하세요.

2단계: Amazon SES 샌드박스에서 계정 이동

기본 Amazon Cognito 이메일 구성을 사용하는 경우 이 단계를 생략하십시오.

어느 곳에서든 AWS 리전 Amazon SES를 처음 사용하면 해당 지역의 Amazon SES 샌드박스에 배치됩니다. AWS 계정 Amazon SES는 사기 및 부정 사용을 방지하기 위해 샌드박스를 사용합니다. Amazon SES 구성을 사용하여 이메일 전송을 처리하는 경우 먼저 AWS 계정 을 샌드박스 외부로 이동해야 Amazon Cognito가 사용자에게 이메일을 보낼 수 있습니다.

샌드박스에서 Amazon SES는 전송할 수 있는 이메일 수와 전송할 수 있는 위치를 제한합니다. Amazon SES로 확인한 주소 및 도메인에만 이메일을 보내거나 Amazon SES 메일박스 시뮬레이터 주소로 이메일을 보낼 수 있습니다. 샌드박스에 있는 동안에는 프로덕션 중인 애플리케이션에 Amazon SES 구성을 사용하지 마십시오. AWS 계정 이 경우 Amazon Cognito에서 사용자의 이메일 주소로 메시지를 보낼 수 없습니다.

AWS 계정 샌드박스에서 사용자를 제거하려면 Amazon Simple 이메일 서비스 개발자 안내서의 Amazon SES 샌드박스에서 벗어나기를 참조하십시오.

3단계: Amazon Cognito에 이메일 권한 부여

사용자에게 이메일을 보내기 전에 특정 권한을 Amazon Cognito에 부여해야 할 수도 있습니다. 부여하는 권한과 권한을 부여하는 데 사용하는 프로세스는 기본 이메일 구성을 사용하는지, Amazon SES 구성을 사용하는지에 따라 달라집니다.

기본 이메일 구성을 사용할 수 있는 권한을 부여하려면

Cognito로 이메일을 보내도록 사용자 풀을 구성하거나 로 설정한 `EmailSendingAccount` 경우에만 이 단계를 완료하십시오. `COGNITO_DEFAULT`

기본 이메일 구성을 사용하면 사용자 풀이 다음 주소 중 하나로 이메일 메시지를 보낼 수 있습니다.

- 기본 주소`no-reply@verificationemail.com`.
- Amazon SES에서 확인된 이메일 주소 또는 도메인의 사용자 지정 FROM 주소.

사용자 정의 주소를 사용하는 경우 Amazon Cognito에 추가 권한이 있어야 해당 주소에서 사용자에게 이메일을 보낼 수 있습니다. 이러한 권한은 Amazon SES의 주소 또는 도메인에 대한 전송 권한 부여 정책을 통해 부여됩니다. Amazon Cognito 콘솔을 사용하여 사용자 풀에 사용자 정의 주소를 추가하는 경우 정책이 Amazon SES 확인 이메일 주소에 자동으로 연결됩니다. 하지만 AWS CLI 또는 Amazon Cognito API를 사용하는 등 콘솔 외부에서 사용자 풀을 구성하는 경우 Amazon SES 콘솔 또는 API를 사용하여 정책을 연결해야 합니다. PutIdentityPolicy

> **Note**
>
> AWS CLI 또는 Amazon Cognito API를 사용하여 확인된 도메인의 FROM 주소만 구성할 수 있습니다.

전송 권한 부여 정책은 Amazon Cognito를 사용하여 Amazon SES를 호출하는 계정 리소스를 기준으로 액세스를 허용하거나 거부합니다. 리소스 기반 정책에 대한 자세한 내용은 IAM 사용 설명서를 참조하세요. Amazon SES 개발자 가이드에서 리소스 기반 정책의 예제도 찾아볼 수 있습니다.

Example 전송 권한 부여 정책

다음 전송 권한 부여 정책 예제는 Amazon Cognito에 Amazon SES 확인 자격 증명을 사용할 수 있는 제한적인 권한을 부여합니다. Amazon Cognito는 `aws:SourceArn` 조건의 사용자 풀과 `aws:SourceAccount` 조건의 계정을 둘 다 대신하는 경우에만 이메일 메시지를 전송할 수 있습니다.

Regions with Amazon SES

사용자 풀 지역 또는 대체 지역의 전송 권한 부여 정책은 Amazon Cognito 서비스 주체가 이메일 메시지를 보낼 수 있도록 허용해야 합니다. 자세한 내용은 지역 표를 참조하십시오. 사용자 풀 지역이 Amazon SES 지역의 하나 이상의 값과 일치하는 경우, 다음 예제에서 글로벌 서비스 보안 주체를 사용하여 전송 권한 부여 정책을 구성하십시오.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "stmnt1234567891234",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "email.cognito-idp.amazonaws.com"
                ]
            },
            "Action": [
                "SES:SendEmail",
                "SES:SendRawEmail"
            ],
            "Resource": "<your SES identity ARN>",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "<your account number>"
                },
                "ArnLike": {
                    "aws:SourceArn": "<your user pool ARN>"
                }
            }
        }
```

```
    ]
}
```

Opt-in Regions without Amazon SES

Amazon Cognito를 사용할 수 있는 경우 모든 옵트인에서 Amazon SES를 사용할 수 AWS 리전 있는 것은 아닙니다. 중동 (UAE) 을 예로 들 수 있으며, 유럽 (프랑크푸르트) () 에서는 확인된 ID 를 가진 이메일만 보낼 수 있습니다. `eu-central-1` 기본 이메일 구성을 사용하는 사용자 풀에서 Amazon Cognito는 두 지역 각각에서 확인된 자격 증명이 포함된 이메일 메시지를 전송하기도 합니다. 중동 (UAE) 의 경우 추가 지역은 유럽 (런던) 입니다. 두 지역의 전송 권한 부여 정책을 업데이트해야 합니다.

각 대체 지역의 전송 권한 부여 정책은 사용자 풀 옵트인 지역의 Amazon Cognito 서비스 보안 주체가 이메일 메시지를 보낼 수 있도록 허용해야 합니다. 자세한 내용은 지역 표를 참조하십시오. 지역이 대체 지역으로 표시된 경우 다음 예와 같이 지역 서비스 주체를 사용하여 전송 권한 부여 정책을 구성하십시오. 필요에 따라 예제 지역 식별자 *me-central-1# ##* 지역 ID로 바꾸십시오.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "cognito-idp.me-central-1.amazonaws.com"
                ]
            },
            "Action": [
                "SES:SendEmail",
                "SES:SendRawEmail"
            ],
            "Resource": "<your SES identity ARN>",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "<your account number>"
                },
                "ArnLike": {
                    "aws:SourceArn": "<your user pool ARN>"
                }
            }
        }
    ]
```

```
}
```

정책 구문에 대한 자세한 내용은 Amazon Simple Email Service 개발자 가이드에서 Amazon SES 전송 권한 부여 정책을 참조하세요.

자세한 내용은 Amazon Simple Email Service 개발자 가이드에서 Amazon SES 전송 권한 부여 정책 예제를 참조하세요.

Amazon SES 구성을 사용할 수 있는 권한을 부여하려면

Amazon SES 구성을 사용하도록 사용자 풀을 구성한 경우 Amazon Cognito는 사용자에게 이메일을 보낼 때 사용자 대신 Amazon SES를 호출할 수 있는 추가 권한이 필요합니다. 이 권한은 IAM 서비스를 사용하여 부여됩니다.

이 옵션으로 사용자 풀을 구성하면 Amazon Cognito가 AWS 계정에서 IAM 역할 유형인 서비스 연결 역할을 생성합니다. 이 역할에는 Amazon Cognito에서 Amazon SES에 액세스하여 귀하의 주소로 이메일 전송을 허용하는 권한이 포함되어 있습니다.

Amazon Cognito는 구성을 설정하는 사용자 세션의 AWS 자격 증명을 사용하여 서비스 연결 역할을 생성합니다. 이 세션의 IAM 권한에는 `iam:CreateServiceLinkedRole` 작업이 포함되어야 합니다. IAM의 권한에 대한 자세한 내용은 IAM 사용 설명서의 AWS 리소스 액세스 관리를 참조하십시오.

Amazon Cognito가 생성하는 서비스 연결 역할에 대한 자세한 내용은 Amazon Cognito에 서비스 연결 역할 사용 섹션을 참조하세요.

4단계: 사용자 풀 구성

다음 중 하나를 사용하여 사용자 풀을 구성하려면 다음 단계를 완료하세요.

- 이메일 발신자로 표시되는 사용자 지정 발신 주소
- 사용자가 발신 주소로 보낸 메시지를 받는 사용자 지정 회신 주소
- Amazon SES 구성

> **Note**
>
> 확인된 자격 증명이 이메일 주소인 경우 Amazon Cognito는 기본적으로 해당 이메일 주소를 FROM 및 REPLY-TO 이메일 주소로 설정합니다. 하지만 확인된 자격 증명이 도메인인 경우 FROM 및 REPLY-TO 이메일 주소의 값을 제공해야 합니다. 예를 들어 확인된 도메인이

example.com인 경우 no-reply@example.com 을 FROM 이메일 주소와 REPLY-TO 이메일 주소로 설정할 수 있습니다.

기본 Amazon Cognito 이메일 구성 및 주소를 사용하려면 이 절차를 생략하십시오.

사용자 지정 이메일 주소를 사용하도록 사용자 풀을 구성하려면

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 자격 증명을 입력합니다. AWS
2. 사용자 풀(User Pools)을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택합니다.
4. [메시징(Messaging)] 탭을 선택하고 [이메일 구성(Email configuration)]을 찾은 다음, [편집(Edit)]을 선택합니다.
5. [이메일 구성 편집(Edit email configuration)] 페이지에서 [Amazon SES SES에서 이메일 전송(Send email from Amazon SES)] 또는 [Amazon Cognito로 이메일 전송(Send email with Amazon Cognito)]을 선택합니다. [Amazon SES에서 이메일 전송(Send email from Amazon SES)]을 선택한 경우에만 SES 리전(SES Region), 구성 집합(Configuration Set), FROM 발신자 이름(FROM sender name)을 사용자 정의할 수 있습니다.
6. 사용자 정의 FROM 주소를 사용하려면 다음 단계를 완료합니다.
   a. [SES 리전(SES Region)]에서 확인된 이메일 주소가 포함된 리전을 선택합니다.
   b. 발신 이메일 주소(FROM email address)에서 이메일 주소를 선택합니다. Amazon SES에서 확인된 이메일 주소를 사용합니다.
   c. (선택 사항) 구성 세트(Configuration set)에서 Amazon SES에 사용할 구성 세트를 선택합니다. 이 변경 사항을 수행하고 저장하면 서비스 연결 역할이 생성됩니다.
   d. (선택 사항) 발신 발신자 주소(FROM sender address) 아래에 이메일 주소를 입력합니다. 이메일 주소만 제공하거나, 이메일 주소와 친숙한 이름을 `Jane Doe <janedoe@example.com>` 포맷으로 제공할 수 있습니다.
   e. (선택 사항) [REPLY-TO 이메일 주소(REPLY-TO email address)] 아래에 사용자가 FROM 주소로 보내는 메시지를 받으려는 이메일 주소를 입력합니다.
7. 변경 사항 저장(Save changes)을 선택합니다.

관련 주제

- 이메일 확인 메시지 사용자 정의

- 사용자 초대 메시지 사용자 정의

Amazon Cognito 사용자 풀의 SMS 메시지 설정

사용자 풀에 대한 일부 Amazon Cognito 이벤트로 인해 Amazon Cognito가 사용자에게 SMS 문자 메시지를 보낼 수 있습니다. 예를 들어 전화 확인을 요구하도록 사용자 풀을 구성한 경우 사용자가 앱에서 새 계정을 등록하거나 암호를 다시 설정하면 Amazon Cognito에서 SMS 문자 메시지를 보냅니다. SMS 문자 메시지를 유발하는 작업에 따라 메시지에는 확인 코드, 임시 암호 또는 시작 메시지가 포함됩니다.

Amazon Cognito는 SMS 문자 메시지 전송에 Amazon Simple Notification Service(Amazon SNS)를 사용합니다. Amazon Cognito 또는 Amazon SNS를 통해 문자 메시지를 처음 보내는 경우 Amazon SNS에서 해당 보내는 사람을 샌드박스 환경에 배치합니다. 샌드박스 환경에서는 애플리케이션에서 SMS 문자 메시지를 테스트할 수 있습니다. 샌드박스에서 메시지는 확인된 전화 번호로만 보낼 수 있습니다.

Amazon SNS는 SMS 텍스트 메시지에 대해 요금을 부과합니다. 자세한 내용은 Amazon SNS 요금을 참조하세요.

> **Note**
>
> 원치 않는 SMS 트래픽 양의 전 세계적 증가로 인해 일부 정부는 SMS 메시지 발신자와 수신자 사이에 장벽을 부과하고 있습니다. MFA와 사용자 업데이트에 SMS 메시지를 사용하는 경우 메시지가 전달되도록 추가 조치를 취해야 합니다. 또한 사용자가 거주할 수 있는 국가의 SMS 메시지 관련 규정을 모니터링하고 SMS 메시지 구성을 최신 상태로 유지해야 합니다. 자세한 내용은 Amazon Simple Notification Service 개발자 안내서의 모바일 문자 메시지(SMS)를 참조하세요.
>
> SMS 메시지를 사용한 사용자 인증과 확인은 보안 모범 사례가 아닙니다. 전화번호는 소유자가 바뀔 수 있으며, 사용자의 가지고 있는 것 MFA 요소를 확실히 대표하지 못할 수 있습니다. 대신 앱이나 서드 파티 IdP를 사용하여 TOTP MFA를 구현하세요. 사용자 정의 인증 챌린지 Lambda 트리거를 사용하여 추가 사용자 지정 인증 요소를 생성할 수도 있습니다.

Amazon Cognito는 사용자가 입력할 수 있는 코드와 함께 SMS 메시지를 사용자에게 보냅니다. 다음 표에서는 SMS 메시지를 생성할 수 있는 이벤트를 보여줍니다.

메시지 옵션

활동	API 작업	전송 옵션	포맷 옵션	사용자 지정 가능	메시지 템플릿
암호 찾기	ForgotPassword	이메일, SMS	code	아니요	N/A
초대	AdminCreateUser	이메일, SMS	code	예	초대 메시지
셀프 등록	SignUp	이메일, SMS	코드, 링크	예	확인 메시지
이메일 주소 또는 전화번호 인증	UpdateUserAttributes	이메일, SMS	code	예	확인 메시지
멀티 팩터 인증(MFA)	AdminInitiateAuth, InitiateAuth	SMS, 인증 앱	code	예[1]	MFA 메시지

[1] SMS 메시지의 경우.

Amazon Cognito 사용자 풀에서 처음으로 SMS 메시지 설정

Amazon Cognito는 Amazon SNS를 사용하여 사용자 풀로 SMS 메시지를 전송합니다. 자체 리소스를 사용하는 사용자 지정 SMS 발신자 Lambda 트리거를 사용하여 SMS 메시지를 보낼 수도 있습니다. 특정 AWS 리전지역에서 SMS 문자 메시지를 전송하도록 Amazon SNS를 처음 설정하면 Amazon SNS는 해당 지역의 SMS 샌드박스에 사용자를 배치합니다. Amazon SNS는 샌드박스를 사용하여 사기 및 남용을 방지하고 규정 준수 요구 사항을 AWS 계정 충족합니다. 샌드박스에 있는 경우 Amazon AWS 계정 SNS는 몇 가지 제한을 적용합니다. 예를 들어 Amazon SNS에서 확인한 최대 10개의 전화번호로 문자 메시지를 보낼 수 있습니다. 샌드박스에 있는 동안에는 프로덕션 중인 애플리케이션에 Amazon SNS 구성을 사용하지 마십시오. AWS 계정 샌드박스에서는 Amazon Cognito가 사용자의 전화 번호로 메시지를 보낼 수 없습니다.

사용자 풀 사용자에게 SMS 문자 메시지를 보내려면

1. Amazon Cognito가 Amazon SNS로 SMS 메시지를 보내는 데 사용할 수 있는 IAM 역할을 준비합니다.
2. Amazon SNS SMS AWS 리전 메시지용 선택
3. 미국 전화 번호로 SMS 메시지를 보내기 위한 발신 자격 증명 얻기

4. 환경이 SMS 샌드박스인지 확인합니다.
5. Amazon SNS 샌드박스 외부로 계정 이동
6. Amazon SNS에서 Amazon Cognito의 전화 번호 확인
7. Amazon Cognito에서 사용자 풀 설정 완료

Amazon Cognito가 Amazon SNS로 SMS 메시지를 보내는 데 사용할 수 있는 IAM 역할을 준비합니다.

사용자 풀에서 SMS 메시지를 보낼 때 Amazon Cognito는 계정에서 IAM 역할을 맡습니다. Amazon Cognito는 해당 역할에 할당된 `sns:Publish` 권한을 사용하여 사용자에게 SMS 메시지를 보냅니다. Amazon Cognito 콘솔의 SMS 아래에 있는 사용자 풀의 Messaging(메시징) 탭에서 IAM role selection(IAM 역할 선택)을 설정하거나 사용자 풀 생성 마법사 중에 선택할 수 있습니다.

다음 예제에서는 IAM 역할 신뢰 정책은 Amazon Cognito 사용자 풀에 역할을 맡을 수 있는 제한된 권한을 부여합니다. Amazon Cognito는 `aws:SourceArn` 조건의 사용자 풀과 AWS 계정 조건의 `aws:SourceAccount`을 대신하여 역할을 맡을 때만 역할을 맡을 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": "cognito-idp.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "<your account number>"
            },
            "ArnLike": {
                "aws:SourceArn": "<your user pool ARN>"
            }
        }
    }]
}
```

`aws:SourceArn` 조건 값에 정확한 사용자 풀 ARN 또는 와일드카드 ARN을 지정할 수 있습니다. DescribeUserPoolAPI 요청을 통해 OR에 있는 사용자 풀의 AWS Management Console ARN을 조회하십시오.

IAM 역할 및 신뢰 정책에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 역할 용어 및 개념을 참조하세요.

Amazon SNS SMS AWS 리전 메시지용 선택

일부 AWS 리전지역에서는 Amazon Cognito SMS 메시지에 사용할 Amazon SNS 리소스가 포함된 지역을 선택할 수 있습니다. 아시아 태평양 (서울) 을 제외하고 Amazon Cognito를 사용할 수 있는 모든 AWS 리전 AWS 리전 곳에서 사용자 풀을 생성한 곳의 Amazon SNS 리소스를 사용할 수 있습니다. 리전을 선택할 때 SMS 메시징을 더 빠르고 안정적으로 만들려면 해당 사용자 풀과 동일한 리전에서 Amazon SNS 리소스를 사용합니다.

> **Note**
>
> 에서는 새로운 Amazon Cognito 콘솔 환경으로 전환한 후에만 SMS 리소스의 지역을 변경할 수 있습니다. AWS Management Console

새 사용자 풀 마법사의 메시지 전송 구성(Configure message delivery) 단계에서 SMS 리소스에 대한 리전을 선택합니다. 또한 기존 사용자 풀의 메시징(Messaging) 탭의 SMS 아래에서 편집(Edit)을 선택할 수 있습니다.

Amazon Cognito는 출시 당시 대체 지역의 Amazon SNS 리소스와 함께 SMS 메시지를 전송하기도 AWS 리전했습니다. 선호 지역을 설정하려면 사용자 풀의 SmsConfigurationType객체 `SnsRegion` 파라미터를 사용하십시오. 다음 표의 Amazon Cognito 리전에서 Amazon Cognito 사용자 풀 리소스를 프로그래밍 방식으로 생성하고 `SnsRegion` 파라미터를 제공하지 않으면 사용자 풀이 레거시 Amazon SNS 리전에서 Amazon SNS 리소스와 함께 SMS 메시지를 보낼 수 있습니다.

아시아 태평양 (서울) 의 Amazon Cognito 사용자 풀은 아시아 태평양 (도쿄) 지역의 Amazon SNS 구성을 AWS 리전 사용해야 합니다.

Amazon SNS는 모든 새 계정의 지출 할당량을 매월 1.00 USD로 설정합니다. Amazon Cognito와 함께 사용하는 경우 지출 한도가 증가했을 수 있습니다. AWS 리전 Amazon SNS SMS 메시지를 변경하기 전에 AWS Support Center에서 할당량 증가 사례를 열어 새 지역의 한도를 늘리십시오. AWS 리전 자세한 내용은 Amazon Simple Notification Service 개발자 안내서의 Amazon SNS에 대한 월별 SMS 지출 할당량 증가 요청을 참조하세요.

다음 표의 모든 Amazon Cognito 리전에 대한 SMS 메시지를 해당되는 Amazon SNS 리전의 Amazon SNS 리소스와 함께 보낼 수 있습니다.

Amazon Cognito 리전	Amazon SNS 리전
미국 동부(오하이오)	미국 동부(오하이오), 미국 동부(버지니아 북부)
캐나다(중부)	캐나다(중부), 미국 동부(버지니아 북부)
유럽(프랑크푸르트)	유럽(프랑크푸르트), 유럽(아일랜드)
유럽(런던)	유럽(런던), 유럽(아일랜드)
아시아 태평양(서울)	아시아 태평양(도쿄)
미국 동부(버지니아 북부)	미국 동부(버지니아 북부)
미국 서부(캘리포니아 북부)	미국 서부(캘리포니아 북부)
미국 서부(오레곤)	미국 서부(오리건)
아시아 태평양(뭄바이)	아시아 태평양(뭄바이), 아시아 태평양(싱가포르)
아시아 태평양(하이데라바드)	아시아 태평양(하이데라바드)
아시아 태평양(싱가포르)	아시아 태평양(싱가포르)
아시아 태평양(시드니)	아시아 태평양(시드니)
아시아 태평양(도쿄)	아시아 태평양(도쿄)
아시아 태평양(자카르타)	아시아 태평양(자카르타)
아시아 태평양(오사카)	아시아 태평양(오사카)
아시아 태평양(멜버른)	아시아 태평양(멜버른)
유럽(아일랜드)	유럽(아일랜드)
유럽(파리)	유럽(파리)
유럽(스톡홀름)	유럽(스톡홀름)
유럽(밀라노)	유럽(밀라노)

Amazon Cognito 리전	Amazon SNS 리전
유럽(스페인)	유럽(스페인)
중동(바레인)	중동(바레인)
남아메리카(상파울루)	남아메리카(상파울루)
이스라엘(텔아비브)	이스라엘(텔아비브)
아프리카(케이프타운)	아프리카(케이프타운)
중동(UAE)	중동(UAE)
유럽(취리히)	유럽(취리히)

미국 전화 번호로 SMS 메시지를 보내기 위한 발신 자격 증명 얻기

미국 전화 번호로 SMS 문자 메시지를 보내려는 경우 SMS 샌드박스 테스트 환경을 구축하거나 프로덕션 환경을 구축하는 경우 모두 발신 자격 증명을 얻어야 합니다.

2021년 6월 1일부터 미국 이동 통신사는 미국 전화 번호로 메시지를 보낼 때 발신 자격 증명을 요구합니다. 발신 자격 증명이 아직 없는 경우 받아야 합니다. 소스 자격 증명을 얻는 방법을 알아보려면 Amazon Pinpoint 사용 설명서에서 번호 요청을 참조하세요.

다음과 같은 AWS 리전방식으로 사업을 운영하는 경우 AWS Support 티켓을 열어 원본 ID를 얻어야 합니다. 자세한 지침은 Amazon Simple Notification Service 개발자 가이드에서 SMS 메시징 지원 요청을 참조하세요.

- 미국 동부(오하이오)
- 유럽(스톡홀름)
- 유럽(파리)
- 유럽(밀라노)
- Middle East (Bahrain)
- 남아메리카(상파울루)
- 미국 서부(캘리포니아 북부)

동일한 오리진 ID에 둘 이상의 오리진 ID가 있는 경우 Amazon SNS는 단축 코드 AWS 리전, 10DLC, 무료 전화 번호와 같은 우선 순위에 따라 오리진 ID 유형을 선택합니다. 이 우선 순위는 변경할 수 없습니다. 자세한 내용은 Amazon SNS FAQ를 참조하세요.

환경이 SMS 샌드박스인지 확인합니다.

환경이 SMS 샌드박스인지 확인하려면 다음 절차를 따르세요. 프로덕션 Amazon Cognito 사용자 풀이 AWS 리전 있는 각 위치에 대해 반복합니다.

Amazon Cognito 콘솔에서 SMS 샌드박스 상태 검토

환경이 SMS 샌드박스인지 확인하려면

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 AWS 자격 증명을 입력합니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택합니다.
4. [메시징(Messaging)] 탭을 선택합니다.
5. [SMS 구성(SMS configuration)] 섹션에서 [Amazon SNS 프로덕션 환경으로 이동(Move to Amazon SNS production environment)]을 확장합니다. 계정이 SMS 샌드박스에 있는 경우 다음 메시지가 표시됩니다.

   ```
   You are currently in the SMS Sandbox and cannot send SMS messages to
   unverified numbers.
   ```

   이 메시지가 표시되지 않으면 해당 계정으로 이미 SMS 메시지를 설정한 다른 사용자가 있는 것입니다. Amazon Cognito에서 사용자 풀 설정 완료 섹션으로 이동하세요.
6. 메시지에서 Amazon SNS 링크를 선택합니다. 이렇게 하면 Amazon SNS 콘솔이 새 탭에서 열립니다.
7. 샌드박스 환경에 있는지 확인합니다. 콘솔 메시지는 샌드박스 상태를 나타내며 AWS 리전, 다음과 같습니다.

   ```
   This account is in the SMS sandbox in US East (N. Virginia).
   ```

Amazon SNS 샌드박스 외부로 계정 이동

앱을 테스트하고 관리자가 확인할 수 있는 전화 번호로만 SMS 메시지를 보내야 하는 경우 이 단계를 건너뜁니다.

프로덕션에서 앱을 사용하려면 계정을 SMS 샌드박스에서 프로덕션으로 이동합니다. Amazon Cognito에서 사용할 Amazon SNS 리소스가 AWS 리전 포함된 오리진 ID를 구성한 후에는 SMS 샌드박스에 남아 있는 동안 미국 전화번호를 확인할 수 AWS 계정 있습니다. Amazon SNS 환경이 프로덕션 상태인 경우 사용자에게 SMS 메시지를 보내기 위해 Amazon SNS에서 사용자 전화 번호를 확인할 필요가 없습니다.

자세한 지침은 Amazon Simple Notification Service 개발자 가이드에서 SNS 샌드박스 환경에서 나가기를 참조하세요.

Amazon SNS에서 Amazon Cognito의 전화 번호 확인

계정을 SMS 샌드박스 외부로 이동한 경우 이 단계를 건너뜁니다.

환경이 SMS 샌드박스인 경우 Amazon SNS에서 확인한 모든 전화 번호로 메시지를 보낼 수 있습니다.

전화 번호를 확인하려면 다음을 수행합니다.

1. Amazon SNS 콘솔의 문자 메시지(SMS)(Text messaging (SMS)) 섹션에 Sandbox 대상 전환 번호(Sandbox destination phone number)를 추가합니다.
2. 입력한 전화 번호로 코드가 포함된 SMS 메시지를 수신합니다.
3. Amazon SNS 콘솔에 SMS 메시지의 확인 코드(Verification code)를 입력합니다.

자세한 지침은 Amazon Simple Notification Service 개발자 가이드에서 SMS 샌드박스에서 전화 번호 추가 및 확인을 참조하세요.

> **Note**
>
> Amazon SNS는 사용자가 SMS 샌드박스에 있는 동안 확인할 수 있는 대상 전화 번호 수를 제한합니다. 자세한 내용은 Amazon Simple Notification Service 개발자 가이드에서 SMS 샌드박스를 참조하세요.

Amazon Cognito에서 사용자 풀 설정 완료

사용자 풀을 생성 또는 편집하던 브라우저 탭으로 돌아갑니다. 절차를 완료합니다. 사용자 풀에 SMS 구성을 성공적으로 추가하면 Amazon Cognito가 내부 전화번호에 테스트 메시지를 전송하여 구성이 제대로 작동하는지 확인합니다. Amazon SNS는 각 테스트 SMS 메시지에 대해 요금을 청구합니다.

사용자 풀에 토큰 사용

사용자를 인증하고 토큰으로 리소스에 대한 액세스 권한을 부여합니다. 토큰의 클레임은 사용자에 대한 정보입니다. ID 토큰에는 사용자 이름, 성, 이메일 주소와 같은 사용자의 신원에 대한 클레임이 포함되어 있습니다. 액세스 토큰에는 인증된 사용자가 서드 파티 API, Amazon Cognito 사용자 셀프 서비스 API 작업, UserInfo 엔드포인트에 액세스하는 데 사용할 수 있는 `scope`과 같은 클레임이 포함되어 있습니다. 액세스 토큰과 ID 토큰에는 모두 사용자 풀의 사용자 그룹 멤버십이 포함된 `cognito:groups` 클레임이 포함되어 있습니다. 사용자 풀 그룹에 대한 자세한 내용은 사용자 풀에 그룹 추가 섹션을 참조하세요.

Amazon Cognito에는 새 토큰을 얻거나 기존 토큰을 취소하는 데 사용할 수 있는 새로 고침 토큰도 있습니다. 토큰을 새로 고쳐 새 ID와 액세스 토큰을 가져옵니다. 토큰을 취소하여 새로 고침 토큰을 통해 부여되는 사용자 액세스 권한을 취소할 수 있습니다.

Amazon Cognito는 Base64로 인코딩된 문자열으로 토큰을 발급합니다. 모든 Amazon Cognito ID 또는 액세스 토큰을 base64에서 일반 텍스트 JSON으로 디코딩할 수 있습니다. Amazon Cognito 새로 고침 토큰은 암호화되므로 사용자 풀 사용자 및 관리자가 읽을 수 없으며 사용자 풀만 읽을 수 있습니다.

토큰으로 인증

사용자가 앱에 로그인하면 Amazon Cognito가 로그인 정보를 확인합니다. 성공적으로 로그인하면 Amazon Cognito가 세션을 생성하고 인증된 사용자를 위한 ID 토큰, 액세스 토큰 및 새로 고침 토큰을 반환합니다. 이 토큰을 사용하여 Amazon API Gateway와 같은 API 및 다운스트림 리소스에 대한 액세스 권한을 사용자에게 부여할 수 있습니다. 또는 임시 AWS 보안 인증 정보로 다른 AWS 서비스에 액세스할 수 있도록 이 토큰을 교환할 수 있습니다.

토큰 저장

앱이 다양한 크기의 토큰을 저장할 수 있어야 합니다. 토큰 크기는 추가 클레임, 인코딩 알고리즘의 변경 및 암호화 알고리즘의 변경을 포함하되 이에 국한되지 않는 이유로 변경될 수 있습니다. 사용자 풀에서 토큰 취소를 활성화하면 Amazon Cognito가 JSON 웹 토큰에 추가 클레임을 추가하여 크기를 늘립니다. 새 클레임 `origin_jti` 및 `jti`가 액세스 토큰과 ID 토큰에 추가됩니다. 토큰 취소에 대한 자세한 내용은 토큰 철회를 참조하세요.

> **Important**
>
> 모범 사례는 전송 중인 토큰 및 애플리케이션 컨텍스트의 스토리지에 있는 모든 토큰을 보호하는 것입니다. 토큰에는 사용자에 대한 개인 식별 정보와 사용자 풀에 사용하는 보안 모델에 대한 정보가 포함될 수 있습니다.

토큰 사용자 지정

Amazon Cognito가 앱에 전달하도록 액세스 및 ID 토큰을 사용자 지정할 수 있습니다. 사전 토큰 생성 Lambda 트리거에서는 토큰 클레임을 추가, 수정 및 차단할 수 있습니다. 사전 토큰 생성 트리거는 Amazon Cognito가 기본 클레임 세트를 보내는 Lambda 함수입니다. 클레임에는 OAuth 2.0 범위, 사용자 풀 그룹 멤버십, 사용자 속성 등이 포함됩니다. 그러면 함수를 런타임에서 변경하고 업데이트된 토큰 클레임을 Amazon Cognito에 반환할 수 있습니다.

버전 2 이벤트를 사용한 토큰 사용자 지정에 액세스하는 경우 추가 비용이 적용됩니다. 자세한 내용은 Amazon Cognito 요금을 참조하세요.

주제

- ID 토큰 사용
- 액세스 토큰 사용
- 새로 고침 토큰 사용
- 토큰 철회
- JSON 웹 토큰 확인
- 캐싱 토큰

ID 토큰 사용

ID 토큰은 `name`, `email`, `phone_number` 등 인증된 사용자의 자격 증명에 대한 클레임이 포함된 JSON 웹 토큰(JWT)입니다. 애플리케이션 내에서 이 자격 증명 정보를 사용할 수 있습니다. 리소스 서버 또는 서버 애플리케이션에 사용자를 인증하는 경우에도 ID 토큰을 사용할 수 있습니다. 애플리케이션 외부에서 웹 API 작업에 ID 토큰을 사용할 수도 있습니다. 이러한 경우 ID 토큰의 서명을 확인해야 ID 토큰 내의 모든 클레임을 신뢰할 수 있습니다. JSON 웹 토큰 확인를 참조하세요.

ID 토큰 만료는 5분에서 1일 사이로 설정할 수 있습니다. 앱 클라이언트별로 이 값을 설정할 수 있습니다.

> **Important**
>
> 사용자가 호스팅 UI 또는 페더레이션 ID 제공업체(IdP)로 로그인하면 Amazon Cognito는 1시간 동안 유효한 세션 쿠키를 설정합니다. 호스팅 UI 또는 페더레이션을 사용하고 액세스 및 ID 토큰에 대해 1시간 미만의 최소 기간을 지정하는 경우 사용자는 쿠키가 만료될 때까지 유효한 세션을 보유하게 됩니다. 사용자에게 1시간 세션 동안 만료되는 토큰이 있는 경우 사용자는 재인증 없이 토큰을 새로 고칠 수 있습니다.

ID 토큰 헤더

헤더에는 키 ID(`kid`) 및 알고리즘(`alg`) 등 두 가지 정보가 포함되어 있습니다.

```
{
"kid" : "1234example=",
"alg" : "RS256"
}
```

`kid`

키 ID입니다. 이 값은 토큰의 JSON 웹 서명(JWS)을 보호하는 데 어떤 키가 사용되었는지 나타냅니다. `jwks_uri` 엔드포인트에서 사용자 풀 서명 키 ID를 볼 수 있습니다.

`kid` 파라미터에 대한 자세한 내용은 키 식별자(kid) 헤더 파라미터를 참조하세요.

`alg`

Amazon Cognito가 액세스 토큰을 보호하는 데 사용한 암호화 알고리즘입니다. 사용자 풀은 SHA-256에서의 RSA 서명인 RS256 암호화 알고리즘을 사용합니다.

`alg` 파라미터에 대한 자세한 내용은 알고리즘(alg) 헤더 파라미터를 참조하세요.

ID 토큰 기본 페이로드

다음은 ID 토큰의 페이로드 예시입니다. 여기에는 인증된 사용자에 대한 클레임이 포함되어 있습니다. OpenID Connect (OIDC) 표준 클레임에 대한 자세한 내용은 OIDC 표준 클레임 목록을 참조하십시오. a를 사용하여 직접 설계한 클레임을 추가할 수 있습니다. 사전 토큰 생성 Lambda 트리거

```
<header>.{
    "sub": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
    "cognito:groups": [
```

```
        "test-group-a",
        "test-group-b",
        "test-group-c"
    ],
    "email_verified": true,
    "cognito:preferred_role": "arn:aws:iam::111122223333:role/my-test-role",
    "iss": "https://cognito-idp.us-west-2.amazonaws.com/us-west-2_example",
    "cognito:username": "my-test-user",
    "middle_name": "Jane",
    "nonce": "abcdefg",
    "origin_jti": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
    "cognito:roles": [
        "arn:aws:iam::111122223333:role/my-test-role"
    ],
    "aud": "xxxxxxxxxxxxexample",
    "identities": [
        {
            "userId": "amzn1.account.EXAMPLE",
            "providerName": "LoginWithAmazon",
            "providerType": "LoginWithAmazon",
            "issuer": null,
            "primary": "true",
            "dateCreated": "1642699117273"
        }
    ],
    "event_id": "64f513be-32db-42b0-b78e-b02127b4f463",
    "token_use": "id",
    "auth_time": 1676312777,
    "exp": 1676316377,
    "iat": 1676312777,
    "jti": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
    "email": "my-test-user@example.com"
}
.<token signature>
```

sub

인증된 사용자에 대한 고유 식별자(UUID) 또는 제목입니다. 사용자 풀에서 사용자 이름이 고유하지 않을 수 있습니다. sub 클레임은 특정 사용자를 식별하는 가장 좋은 방법입니다.

cognito:groups

사용자를 멤버로 가진 사용자 풀 그룹의 이름 배열입니다. 그룹은 앱에 표시하는 식별자가 될 수도 있고 자격 증명 풀에서 선호하는 IAM 역할에 대한 요청을 생성할 수도 있습니다.

cognito:preferred_role

사용자의 우선순위가 가장 높은 사용자 풀 그룹과 연결한 IAM 역할의 ARN입니다. 사용자 풀이 이 역할 클레임을 선택하는 방법에 대한 자세한 내용은 그룹에 우선 순위 값 할당 섹션을 참조하세요.

iss

토큰을 발행한 자격 증명 공급자입니다. 클레임의 형식은 다음과 같습니다.

`https://cognito-idp.<Region>.amazonaws.com/<your user pool ID>`

cognito:username

사용자 풀에 있는 사용자의 사용자 이름입니다.

nonce

nonce 클레임은 OAuth 2.0 `authorize` 엔드포인트에 대한 요청에 추가할 수 있는 동일한 이름의 파라미터에서 비롯됩니다. 파라미터를 추가하면 nonce 클레임은 Amazon Cognito가 발행하는 ID 토큰에 포함되어 있으며, 이를 사용하여 재생 공격으로부터 보호할 수 있습니다. 요청에 nonce 값을 제공하지 않을 경우, 사용자가 서드 파티 자격 증명 공급자를 통해 인증할 때 Amazon Cognito는 자동으로 논스를 생성 및 검증한 다음 ID 토큰에 대한 nonce 클레임으로 추가합니다. Amazon Cognito에서 nonce 클레임 구현은 OIDC 표준에 기반합니다.

origin_jti

사용자의 새로 고침 토큰과 연결된 토큰 취소 식별자입니다. Amazon Cognito는 사용자가 취소 엔드포인트 또는 API 작업을 통해 사용자 토큰을 취소했는지 확인할 때 `origin_jti` 클레임을 참조합니다. RevokeToken 토큰을 취소하면 Amazon Cognito는 동일한 `origin_jti` 값을 가진 모든 액세스 및 ID 토큰을 무효화합니다.

cognito:roles

사용자 그룹과 연결된 IAM 역할의 이름 배열입니다. 모든 사용자 풀 그룹에는 하나의 IAM 역할을 연결할 수 있습니다. 이 배열은 우선 순위에 관계없이 사용자 그룹의 모든 IAM 역할을 나타냅니다. 자세한 정보는 사용자 풀에 그룹 추가을 참조하세요.

aud

사용자를 인증한 사용자 풀 앱 클라이언트입니다. Amazon Cognito는 액세스 토큰 `client_id` 클레임에서 동일한 값을 렌더링합니다.

identities

사용자 identities 속성의 내용입니다. 속성에는 페더레이션 로그인을 수행하거나 페더레이션 사용자를 로컬 프로필에 연결하여 사용자에게 연결한 각 서드 파티 ID 제공업체 프로필에 대한 정보가 포함됩니다. 이 정보에는 제공업체 이름, 제공업체 고유 ID 및 기타 메타데이터가 포함됩니다.

token_use

토큰의 의도된 목적입니다. ID 토큰에서 값은 id입니다.

auth_time

사용자가 인증을 완료한 인증 시간(Unix 시간 형식)입니다.

exp

사용자의 토큰이 만료되는 만료 시간(Unix 시간 형식)입니다.

iat

Amazon Cognito가 사용자의 토큰을 발급한 발급 시간(Unix 시간 형식)입니다.

jti

JWT의 고유 식별자입니다.

ID 토큰에는 OIDC 표준 클레임에 정의된 OIDC 표준 클레임이 포함될 수 있습니다. 또한 ID 토큰에는 사용자 풀에 정의된 사용자 지정 속성도 포함될 수 있습니다. Amazon Cognito는 ID 토큰에 사용자 지정 속성 값을 속성 유형에 상관없이 문자열로 씁니다.

> **Note**
>
> 사용자 풀 사용자 지정 속성에는 항상 접두사가 붙습니다. custom:

ID 토큰 서명

ID 토큰의 서명은 JWT 토큰의 헤더와 페이로드에 따라 계산됩니다. 앱에서 받는 ID 토큰의 클레임을 수락하기 전에 토큰의 서명을 확인하세요. 자세한 내용은 JSON 웹 토큰 확인을 참조하세요. JSON 웹 토큰 확인.

액세스 토큰 사용

사용자 풀 액세스 토큰에는 인증된 사용자에 대한 클레임, 사용자 그룹 목록 및 범위 목록이 포함되어 있습니다. 액세스 토큰의 목적은 API 작업을 승인하는 것입니다. 사용자 풀은 액세스 토큰을 받아 사용자 셀프 서비스 작업을 승인합니다. 예를 들어 액세스 토큰을 사용하여 사용자 속성을 추가, 변경 또는 삭제하기 위한 액세스 권한을 사용자에게 부여할 수 있습니다.

사용자 풀에 추가한 사용자 지정 범위에서 파생된 액세스 토큰의 OAuth 2.0 범위를 사용하면 사용자에게 API에서 정보를 검색하도록 권한을 부여할 수 있습니다. 예를 들어, Amazon API Gateway는 Amazon Cognito 액세스 토큰을 통한 권한 부여를 지원합니다. 사용자 풀의 정보로 REST API 권한 부여자를 채우거나 Amazon Cognito를 HTTP API용 JSON 웹 토큰(JWT) 권한 부여자로 사용할 수 있습니다. 사용자 지정 범위가 있는 액세스 토큰을 생성하려면 사용자 풀 퍼블릭 엔드포인트를 통해 요청해야 합니다.

사용자의 액세스 토큰은 UserInfo 엔드포인트에서 사용자 속성에 대한 추가 정보를 요청할 수 있는 권한입니다. 사용자의 액세스 토큰은 사용자 속성을 읽고 쓸 수 있는 권한이기도 합니다. 액세스 토큰이 부여하는 속성에 대한 액세스 수준은 앱 클라이언트에 할당하는 권한과 토큰에서 부여하는 범위에 따라 달라집니다.

액세스 토큰은 JSON 웹 토큰(JWT)입니다. 액세스 토큰의 헤더는 ID 토큰과 구조가 동일합니다. Amazon Cognito는 ID 토큰을 서명하는 키와는 다른 키를 사용하여 액세스 토큰을 서명합니다. 액세스 키 ID(`kid`) 클레임의 값은 동일한 사용자 세션의 ID 토큰에 있는 `kid` 클레임 값과 일치하지 않습니다. 앱 코드에서 ID 토큰과 액세스 토큰을 독립적으로 확인하세요. 서명을 확인하기 전까지는 액세스 토큰의 클레임을 신뢰하지 마세요. 자세한 정보는 JSON 웹 토큰 확인을 참조하세요. 액세스 토큰 만료 시간은 5분~1일 사이로 설정할 수 있습니다. 앱 클라이언트별로 이 값을 설정할 수 있습니다.

> **Important**
>
> 액세스 및 ID 토큰의 경우 호스팅 UI를 사용한다면 최솟값을 1시간 미만으로 지정하지 마세요. Amazon Cognito 호스팅된 UI는 1시간 동안 유효한 쿠키를 사용합니다. 최소값을 1시간 미만으로 입력하더라도 만료 시간이 1시간 미만이 되지 않습니다.

액세스 토큰 헤더

헤더에는 키 ID(`kid`) 및 알고리즘(`alg`) 등 두 가지 정보가 포함되어 있습니다.

```
{
"kid" : "1234example="
```

```
"alg" : "RS256",
}
```

kid

키 ID입니다. 이 값은 토큰의 JSON 웹 서명(JWS)을 보호하는 데 어떤 키가 사용되었는지 나타냅니다. `jwks_uri` 엔드포인트에서 사용자 풀 서명 키 ID를 볼 수 있습니다.

`kid` 파라미터에 대한 자세한 내용은 키 식별자(kid) 헤더 파라미터를 참조하세요.

alg

Amazon Cognito가 액세스 토큰을 보호하는 데 사용한 암호화 알고리즘입니다. 사용자 풀은 SHA-256에서의 RSA 서명인 RS256 암호화 알고리즘을 사용합니다.

`alg` 파라미터에 대한 자세한 내용은 알고리즘(alg) 헤더 파라미터를 참조하세요.

액세스 토큰 기본 페이로드

액세스 토큰에서 나온 샘플 페이로드입니다. 자세한 내용은 JWT 클레임을 참조하세요. 사전 토큰 생성 Lambda 트리거a를 사용하여 직접 디자인한 클레임을 추가할 수 있습니다.

```
<header>.
{
   "sub":"aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
   "device_key": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
   "cognito:groups":[
      "testgroup"
   ],
   "iss":"https://cognito-idp.us-west-2.amazonaws.com/us-west-2_example",
   "version":2,
   "client_id":"xxxxxxxxxxxxexample",
   "origin_jti":"aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
   "event_id":"aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
   "token_use":"access",
   "scope":"phone openid profile resourceserver.1/appclient2 email",
   "auth_time":1676313851,
   "exp":1676317451,
   "iat":1676313851,
   "jti":"aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
   "username":"my-test-user"
}
```

```
.<token signature>
```

sub

인증된 사용자에 대한 고유 식별자(UUID) 또는 제목입니다. 사용자 풀에서 사용자 이름이 고유하지 않을 수 있습니다. sub 클레임은 특정 사용자를 식별하는 가장 좋은 방법입니다.

cognito:groups

사용자를 멤버로 가진 사용자 풀 그룹의 이름 배열입니다.

iss

토큰을 발행한 자격 증명 공급자입니다. 클레임의 형식은 다음과 같습니다.

`https://cognito-idp.<Region>.amazonaws.com/<your user pool ID>`

client_id

사용자를 인증한 사용자 풀 앱 클라이언트입니다. Amazon Cognito는 ID 토큰 aud 클레임에서 동일한 값을 렌더링합니다.

origin_jti

사용자의 새로 고침 토큰과 연결된 토큰 취소 식별자입니다. Amazon Cognito는 사용자가 취소 엔드포인트 또는 API 작업을 통해 사용자 토큰을 취소했는지 확인할 때 `origin_jti` 클레임을 참조합니다. RevokeToken 토큰을 취소하면 Amazon Cognito는 동일한 `origin_jti` 값을 가진 모든 액세스 및 ID 토큰을 무효화합니다.

token_use

토큰의 의도된 목적입니다. 액세스 토큰에서 값은 `access`입니다.

scope

토큰이 제공하는 액세스를 정의하는 OAuth 2.0 범위 목록입니다. Token 엔드포인트의 토큰에는 앱 클라이언트가 지원하는 범위가 포함될 수 있습니다. Amazon Cognito API 로그인의 토큰에는 `aws.cognito.signin.user.admin` 범위만 포함됩니다.

auth_time

사용자가 인증을 완료한 인증 시간(Unix 시간 형식)입니다.

exp

사용자의 토큰이 만료되는 만료 시간(Unix 시간 형식)입니다.

`iat`

Amazon Cognito가 사용자의 토큰을 발급한 발급 시간(Unix 시간 형식)입니다.

`jti`

JWT의 고유 식별자입니다.

`username`

사용자 풀에 있는 사용자의 사용자 이름입니다.

액세스 토큰 서명

액세스 토큰의 서명은 JWT 토큰의 헤더와 페이로드에 따라 계산됩니다. 웹 API에 있는 애플리케이션의 외부에서 사용될 경우 해당 토큰을 수락하기 전에 항상 이 서명을 확인해야 합니다. 자세한 정보는 JSON 웹 토큰 확인을 참조하세요.

새로 고침 토큰 사용

새로 고침 토큰을 사용하여 새 ID와 액세스 토큰을 검색할 수 있습니다. 기본적으로 새로 고침 토큰은 애플리케이션 사용자가 사용자 풀에 로그인하고 30일이 지나면 만료됩니다. 사용자 풀에 대한 애플리케이션을 생성할 경우 애플리케이션의 새로 고침 토큰 만료를 60분에서 10년 사이의 값으로 설정할 수 있습니다.

유효한(만료되지 않은) 새로 고침 토큰이 있는 경우 Mobile SDK for iOS, Mobile SDK for Android, Amplify for iOS, Android 및 Flutter는 ID 및 액세스 토큰을 자동으로 새로 고칩니다. ID 및 액세스 토큰의 최소 잔여 유효 기간은 2분입니다. 새로 고침 토큰의 기간이 만료되면 앱 사용자가 사용자 풀에 다시 로그인하여 재인증을 해야 합니다. 액세스 토큰과 ID 토큰의 최소값을 5분으로 설정하고 SDK를 사용하는 경우 계속해서 새로 고침 토큰을 사용하여 새 액세스 토큰과 ID 토큰이 검색됩니다. 최소값을 5분이 아니라 7분으로 설정하면 예상 동작을 확인할 수 있습니다.

새 계정의 `UnusedAccountValidityDays` 시간 제한 전에 사용자가 한 번 이상 로그인하는 한, 사용자 계정 자체는 만료되지 않습니다.

새로 고침 토큰으로 새 액세스 및 자격 증명 토큰 받기

API 또는 호스팅 UI를 사용하여 새로 고침 토큰에 대한 인증을 시작합니다.

새로 고침 토큰을 사용하여 사용자 풀 API로 새 ID 및 액세스 토큰을 가져오려면 AdminInitiateAuth또는 InitiateAuthAPI 작업을 사용하십시오. `AuthFlow` 파라미터에 `REFRESH_TOKEN_AUTH`를 전달합니

다. `AuthFlow`의 `AuthParameters` 속성에서 사용자의 새로 고침 토큰을 `"REFRESH_TOKEN"` 값으로 전달합니다. Amazon Cognito는 API 요청이 모든 챌린지를 전달한 후 새 ID 및 액세스 토큰을 반환합니다.

> **Note**
>
> Amazon Cognito 사용자 풀 API를 사용하여 호스팅 UI 사용자의 토큰을 새로 고치려면 `InitiateAuth` 요청을 생성합니다.

도메인을 구성한 사용자 풀의 Token 엔드포인트에 새로 고침 토큰을 제출할 수도 있습니다. 요청 본문에 `refresh_token`의 `grant_type` 값과 사용자 새로 고침 토큰의 `refresh_token` 값을 포함합니다.

새로 고침 토큰 철회

사용자에게 속한 새로 고침 토큰을 취소할 수 있습니다. 토큰 취소에 대한 자세한 내용은 토큰 철회 섹션을 참조하세요.

> **Note**
>
> 새로 고침 토큰을 취소하면 Amazon Cognito가 해당 토큰을 사용하여 새로 고침 요청에서 발급한 모든 ID 및 액세스 토큰이 취소됩니다.

사용자는 `GlobalSignOut` 및 `AdminUserGlobalSignOut` API 작업을 사용하여 사용자의 모든 토큰을 취소할 경우 현재 로그인되어 있는 모든 디바이스에서 로그아웃할 수 있습니다. 사용자가 로그아웃하면 다음과 같은 효과가 발생합니다.

- 사용자의 새로 고침 토큰을 사용하여 사용자에 대한 새 토큰을 가져올 수 없습니다.
- 사용자의 액세스 토큰을 사용하여 토큰으로 권한이 부여된 API 요청을 할 수 없습니다.
- 새 토큰을 가져오려면 사용자가 다시 인증해야 합니다. 호스팅 UI 세션 쿠키는 자동으로 만료되지 않으므로 보안 인증 정보를 입력하라는 추가 메시지 없이 사용자가 세션 쿠키를 사용하여 다시 인증할 수 있습니다. 호스팅 UI 사용자를 로그아웃한 후 Logout 엔드포인트로 리디렉션하면 Amazon Cognito가 세션 쿠키를 지웁니다.

새로 고침 토큰을 사용하면 앱에서 사용자의 세션을 오랫동안 유지할 수 있습니다. 시간이 지남에 따라 사용자는 로그인한 일부 디바이스의 권한 부여를 해제하여 세션을 계속 새로 고칠 수 있습니다. 단

일 디바이스에서 사용자를 로그아웃하려면 새로 고침 토큰을 취소합니다. 사용자가 인증된 모든 세션에서 스스로 로그아웃하려는 경우 GlobalSignOutAPI 요청을 생성하세요. 앱에서 사용자에게 모든 디바이스에서 로그아웃과 같은 선택 항목을 제공할 수 있습니다. `GlobalSignOut`은 변경되지 않고 만료되지 않고 취소되지 않은 사용자의 유효한 액세스 토큰을 허용합니다. 이 API는 토큰으로 권한이 부여되었기 때문에 한 사용자가 이 API를 사용하여 다른 사용자의 로그아웃을 시작할 수 없습니다.

하지만 AWS 자격 증명으로 승인하는 AdminUserGlobalSignOutAPI 요청을 생성하여 모든 디바이스에서 모든 사용자를 로그아웃시킬 수 있습니다. 관리자 애플리케이션은 AWS 개발자 자격 증명으로 이 API 작업을 호출하고 사용자 풀 ID와 사용자 이름을 매개변수로 전달해야 합니다. `AdminUserGlobalSignOut` API를 사용하면 사용자 풀의 모든 사용자를 로그아웃할 수 있습니다.

AWS 자격 증명 또는 사용자 액세스 토큰으로 승인할 수 있는 요청에 대한 자세한 내용은 을 참조하십시오Amazon Cognito 사용자 풀 인증 및 미인증 API 작업.

토큰 철회

API를 사용하여 사용자의 새로 고침 토큰을 취소할 수 있습니다. AWS 새로 고침 토큰을 취소하면 해당 새로 고침 토큰에서 이전에 발급한 모든 액세스 토큰이 무효화됩니다. 사용자에게 발급된 다른 새로 고침 토큰은 영향을 받지 않습니다.

> **Note**
>
> JWT 토큰은 토큰이 생성될 때 서명 및 만료 시간이 할당되는 독립형 토큰입니다. 취소된 토큰은 토큰이 필요한 Amazon Cognito API 호출에 사용할 수 없습니다. 그러나 취소된 토큰은 토큰의 서명 및 만료를 확인하는 JWT 라이브러리를 사용하여 확인되는 경우 여전히 유효합니다.

토큰 취소를 사용하는 사용자 풀 클라이언트의 새로 고침 토큰을 취소할 수 있습니다. 새 사용자 풀 클라이언트를 생성하면 토큰 취소가 기본적으로 사용됩니다.

토큰 취소 사용

기존 사용자 풀 클라이언트의 토큰을 취소하려면 먼저 토큰 취소를 사용하도록 설정해야 합니다. 또는 API를 사용하여 기존 사용자 풀 클라이언트에 대한 토큰 취소를 활성화할 수 있습니다. AWS CLI AWS 이렇게 하려면 `aws cognito-idp describe-user-pool-client` CLI 명령 또는 `DescribeUserPoolClient` API 작업을 호출하여 앱 클라이언트에서 현재 설정을 검색합니다. 그런 다음 `aws cognito-idp update-user-pool-client` CLI 명령 또는 `UpdateUserPoolClient` API 작업을 호출합니다. 앱 클라이언트의 현재 설정을 포함하고 `EnableTokenRevocation` 파라미터를 `true`로 설정합니다.

AWS Management Console AWS CLI, 또는 AWS API를 사용하여 새 사용자 풀 클라이언트를 생성하면 토큰 취소가 기본적으로 활성화됩니다.

토큰 취소를 활성화하면 Amazon Cognito JSON 웹 토큰에 새 클레임이 추가됩니다. `origin_jti` 및 `jti` 클레임이 액세스 토큰과 ID 토큰에 추가됩니다. 이러한 클레임은 애플리케이션 클라이언트 액세스 토큰 및 ID 토큰의 크기를 늘립니다.

토큰 취소가 활성화된 상태로 앱 클라이언트를 만들거나 수정하려면 또는 API 요청에 다음 매개변수를 포함하십시오 CreateUserPoolClient. UpdateUserPoolClient

```
"EnableTokenRevocation": true
```

토큰 취소

RevokeTokenAPI 요청 (예: CLI `aws cognito-idp revoke-token` 명령) 을 사용하여 새로 고침 토큰을 취소할 수 있습니다. 취소 엔드포인트을 사용하여 토큰을 취소할 수도 있습니다. 이 엔드포인트는 사용자 풀에 도메인을 추가한 후에 사용할 수 있습니다. Amazon Cognito 호스트된 도메인 또는 자체 사용자 지정 도메인의 취소 엔드포인트를 사용할 수 있습니다.

> **Note**
>
> 새로 고침 토큰을 취소하는 요청에는 토큰을 얻는 데 사용한 것과 동일한 클라이언트 ID가 포함되어야 합니다.

다음은 RevokeToken API 요청 예시의 본문입니다.

```
{
   "ClientId": "1example23456789",
   "ClientSecret": "abcdef123456789ghijklexample",
   "Token": "eyJjdHkiOiJKV1QiEXAMPLE"
}
```

다음은 사용자 지정 도메인이 있는 사용자 풀의 /oauth2/revoke 엔드포인트에 대한 cURL 요청의 예입니다.

```
curl --location 'auth.mydomain.com/oauth2/revoke' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--header 'Authorization: Basic Base64Encode(client_id:client_secret)' \
```

```
--data-urlencode 'token=abcdef123456789ghijklexample' \
--data-urlencode 'client_id=1example23456789'
```

앱 클라이언트에 클라이언트 비밀이 없는 한, RevokeToken 작업 및 /oauth2/revoke 엔드포인트에서 추가 권한 부여가 필요하지 않습니다.

JSON 웹 토큰 확인

이러한 단계들은 사용자 풀 JSON 웹 토큰(JWT)의 확인 방법을 설명합니다.

주제

- 필수 조건
- 다음을 사용하여 토큰을 검증합니다. aws-jwt-verify
- 토큰 이해 및 검사

필수 조건

라이브러리, SDK 또는 소프트웨어 프레임워크에서 이 섹션의 작업을 이미 처리했을 수 있습니다. AWS SDK는 앱에서 Amazon Cognito 사용자 풀 토큰을 처리하고 관리하기 위한 도구를 제공합니다. AWS Amplify Amazon Cognito 토큰을 검색하고 새로 고치는 함수를 포함합니다.

자세한 내용은 다음 페이지를 참조하십시오.

- 웹 및 모바일 앱과 Amazon Cognito 인증 및 권한 부여 통합
- SDK를 사용하는 Amazon Cognito 자격 증명 공급자의 코드 예제 AWS
- Amplify 개발 센터의 Advanced workflows(고급 워크플로)

JSON 웹 토큰(JWT)을 디코딩하고 확인하는 데 사용할 수 있는 유용한 라이브러리가 많습니다. 서버 측 API 처리를 위해 토큰을 수동으로 처리하고자 하는 경우 또는 다른 프로그래밍 언어를 사용 중인 경우 이러한 라이브러리가 유용할 수 있습니다. JWT 토큰 작업을 위한 OpenID Foundation 라이브러리 목록을 참조하세요.

다음을 사용하여 토큰을 검증합니다. aws-jwt-verify

Node.js 앱에서는 aws-jwt-verify라이브러리가 사용자가 앱에 전달하는 토큰의 매개변수를 검증하도록 AWS 권장합니다. `aws-jwt-verify`를 사용하면 하나 이상의 사용자 풀에 대해 확인하려는 클레임 값으로 `CognitoJwtVerifier`를 채울 수 있습니다. 확인할 수 있는 값에는 다음이 포함됩니다.

- 액세스 또는 ID 토큰의 형식이 잘못되었거나 만료되지 않았으며 유효한 서명이 있음
- 액세스 토큰이 올바른 사용자 풀과 앱 클라이언트에서 제공됨
- 액세스 토큰 클레임에 올바른 OAuth 2.0 범위가 포함됨
- 액세스 및 ID 토큰을 서명한 키가 사용자 풀의 JWKS URI에 있는 `kid` 서명 키와 일치함

  JWKS URI에는 사용자 토큰을 서명한 프라이빗 키에 대한 공개 정보가 포함됩니다. 사용자 풀의 JWKS URI는 `https://cognito-idp.<Region>.amazonaws.com/<userPoolId>/.well-known/jwks.json`에서 찾을 수 있습니다.

Node.js 앱 또는 AWS Lambda 권한 부여자에서 사용할 수 있는 자세한 내용 및 예제 코드는 aws-jwt-verify GitHubon을 참조하십시오.

토큰 이해 및 검사

토큰 검사를 앱에 통합하기 전에 Amazon Cognito가 JWT를 어떻게 조합하는지 생각해 보세요. 사용자 풀에서 예시 토큰을 검색하세요. 디코딩하고 자세히 검사하여 특성을 이해하고 언제 무엇을 검증할지 결정하세요. 예를 들어 한 시나리오에서는 그룹 멤버십을 검사하고 또 다른 시나리오에서는 범위를 검사할 수 있습니다.

다음 섹션에서는 앱을 준비하면서 Amazon Cognito JWT를 수동으로 검사하는 프로세스를 설명합니다.

JWT의 구조 확인

JSON 웹 토큰(JWT)에는 `.`(점)으로 구분된 세 개의 섹션이 있습니다.

헤더

Amazon Cognito가 토큰을 서명하는 데 사용한 키 ID(`kid`) 및 RSA 알고리즘(`alg`)입니다. Amazon Cognito는 `RS256`이라는 `alg`로 토큰을 서명합니다.

페이로드

토큰 클레임입니다. ID 토큰의 클레임에는 사용자 속성 및 사용자 풀(`iss`), 앱 클라이언트(`aud`)에 대한 정보가 포함됩니다. 액세스 토큰의 페이로드에는 범위, 그룹 멤버십과 포함되며 사용자 풀이 `iss`로, 앱 클라이언트가 `client_id`로 포함됩니다.

Signature

서명은 헤더 및 페이로드와 같이 디코딩할 수 있는 base64가 아닙니다. JWKS URI에서 관찰할 수 있는 서명 키와 파리미터에서 파생된 RSA256 식별자입니다.

헤더와 페이로드는 base64로 인코딩된 JSON입니다. 시작 문자 {로 디코딩되는 시작 문자 eyJ로 알 수 있습니다. 사용자가 base64로 인코딩된 JWT를 앱에 제시했는데 형식이 `[JSON Header].[JSON Payload].[Signature]`가 아닌 경우, 이는 유효한 Amazon Cognito 토큰이 아니므로 폐기해도 됩니다.

JWT 검증

JWT 서명은 헤더와 페이로드의 해시 조합입니다. Amazon Cognito는 각 사용자 풀에서 RSA 암호화 키를 두 쌍 생성합니다. 하나의 프라이빗 키는 액세스 토큰을 서명하고 다른 하나는 ID 토큰을 서명합니다.

JWT 토큰의 서명을 확인하는 방법

1. ID 토큰을 디코딩합니다.

   또한 OpenID Foundation에는 JWT 토큰 작업을 위한 라이브러리 목록이 포함되어 있습니다.

   사용자 풀 AWS Lambda JWT를 디코딩하는 데도 사용할 수 있습니다. 자세한 내용은 를 사용하여 Amazon Cognito JWT 토큰의 디코딩 및 확인을 참조하십시오. AWS Lambda
2. 로컬 키 ID(`kid`)를 퍼블릭 `kid`와 비교합니다.

   a. 사용자 풀에 해당되는 퍼블릭 JSON 웹 키(JWK)를 다운로드하고 저장합니다. 이 키는 JSON 웹 키 집합(JWKS)의 일부로 사용이 가능합니다. 사용자 환경에 다음 `jwks_uri` URL을 구성하여 찾을 수 있습니다.

   ```
   https://cognito-idp.<Region>.amazonaws.com/<userPoolId>/.well-known/jwks.json
   ```

   JWK 및 JWK 집합에 대한 자세한 내용은 JSON 웹 키(JWK)를 참조하십시오.

   > **Note**
   >
   > Amazon Cognito는 사용자 풀의 서명 키를 교체할 수 있습니다. 가장 좋은 방법은 `kid`를 캐시 키로 사용하여 앱에서 퍼블릭 키를 캐시하고 주기적으로 캐시를 새로 고치는 것입니다. 앱에서 받는 토큰의 `kid`를 캐시와 비교합니다.
   > 발급자는 올바르지만 `kid`가 다른 토큰을 받은 경우 Amazon Cognito가 서명 키를 교체했을 수 있습니다. 사용자 풀 `jwks_uri` 엔드포인트에서 캐시를 새로 고칩니다.

   다음은 샘플 `jwks.json` 파일입니다.

```
{
 "keys": [{
  "kid": "1234example=",
  "alg": "RS256",
  "kty": "RSA",
  "e": "AQAB",
  "n": "1234567890",
  "use": "sig"
 }, {
  "kid": "5678example=",
  "alg": "RS256",
  "kty": "RSA",
  "e": "AQAB",
  "n": "987654321",
  "use": "sig"
 }]
}
```

키 ID(**kid**)

kid는 토큰의 JSON 웹 서명(JWS)을 보호하는 데 어떤 키가 사용되었는지 나타내는 힌트입니다.

알고리즘(**alg**)

alg 헤더 파라미터는 ID 토큰을 보호하는 데 사용되는 암호화 알고리즘을 나타냅니다. 사용자 풀은 SHA-256에서의 RSA 서명인 RS256 암호화 알고리즘을 사용합니다. RSA에 대한 자세한 내용은 RSA 암호화를 참조하세요.

키 유형(**kty**)

kty 파라미터는 이 예제의 'RSA'와 같이 키에서 사용되는 암호화 알고리즘 그룹을 식별합니다.

RSA 지수(**e**)

e 파라미터는 RSA 퍼블릭 키의 지수 값을 포함합니다. 이 값은 Base64urlUInt 인코딩 값으로 표현됩니다.

RSA 모듈러스(**n**)

n 파라미터는 RSA 퍼블릭 키의 모듈러스 값을 포함합니다. 이 값은 Base64urlUInt 인코딩 값으로 표현됩니다.

사용(**use**)

use 파라미터는 퍼블릭 키의 용도를 설명합니다. 이 예제에서 use 값 sig는 서명을 나타냅니다.

b. JWT의 kid와 일치하는 kid에서 퍼블릭 JSON 웹 키를 검색합니다.

3. JWT 라이브러리를 사용하여 발급자 서명을 토큰의 서명과 비교합니다. 발급자 서명은 kid 토큰과 일치하는 jwks.json 내 kid의 퍼블릭 키(RSA 모듈러스 "n")에서 파생됩니다. 먼저 형식을 JWK에서 PEM로 변환해야 합니다. 다음 예시에서는 JWT와 JWK를 가져와서 Node.js 라이브러리인 jsonwebtoken을 사용하여 JWT 서명을 확인합니다.

Node.js

```
var jwt = require('jsonwebtoken');
var jwkToPem = require('jwk-to-pem');
var pem = jwkToPem(jwk);
jwt.verify(token, pem, { algorithms: ['RS256'] }, function(err, decodedToken) {
});
```

클레임 확인

JWT 클레임을 확인하는 방법

1. 다음 방법 중 하나를 사용하여 토큰이 만료되지 않았는지 확인합니다.

   a. 토큰을 디코딩하고 exp 클레임을 현재 시간과 비교합니다.

   b. 액세스 토큰에 aws.cognito.signin.user.admin 클레임이 포함된 경우 와 같은 API로 요청을 보내십시오. GetUser 액세스 토큰으로 권한 부여된 API 요청에서는 토큰이 만료되면 오류가 반환됩니다.

   c. UserInfo 엔드포인트 요청 시 액세스 토큰을 제시합니다. 토큰이 만료된 경우 요청에서 오류가 반환됩니다.

2. ID 토큰의 aud 클레임 및 액세스 토큰의 client_id 클레임은 Amazon Cognito 사용자 풀에서 생성된 앱 클라이언트 ID와 일치해야 합니다.

3. 발행자(iss) 클레임은 사용자 풀과 일치해야 합니다. 예를 들어 us-east-1 리전에서 생성된 사용자 풀은 다음과 같은 iss 값을 갖게 됩니다.

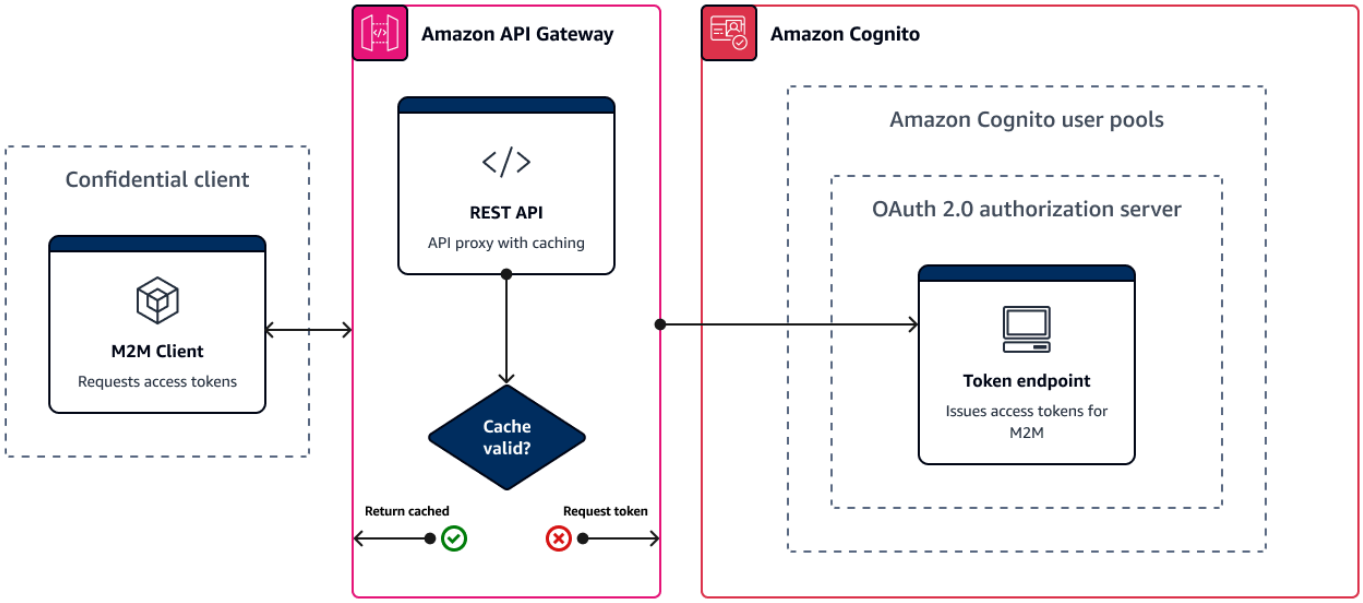
   https://cognito-idp.us-east-1.amazonaws.com/*<userpoolID>*.

4. `token_use` 클레임을 확인합니다.

   - 웹 API 작업에서 액세스 토큰만 허용하는 경우 해당 값은 `access`여야 합니다.
   - ID 토큰만 사용하고 있는 경우 해당 값은 `id`여야 합니다.
   - ID와 액세스 토큰을 모두 사용하고 있는 경우에는 `token_use` 클레임이 `id` 또는 `access`여야 합니다.

이제 토큰 내에 있는 클레임을 신뢰할 수 있습니다.

캐싱 토큰



앱은 새 JSON 웹 토큰(JWT)을 가져올 때마다 다음 요청 중 하나를 성공적으로 완료해야 합니다.

- Token 엔드포인트에서 클라이언트 자격 증명 또는 권한 부여 코드 부여를 요청합니다.
- 호스팅 UI에서 암시적 권한 부여를 요청하세요.
- 와 같은 Amazon Cognito API 요청에서 로컬 사용자를 인증합니다. InitiateAuth

사용자 풀을 구성하여 토큰이 분, 시간 또는 일 단위로 만료되도록 설정할 수 있습니다. 앱의 성능과 가용성을 보장하려면 만료될 때까지 Amazon Cognito 토큰을 사용하고 그 후에만 새 토큰을 검색해야 합니다. 앱용으로 구축한 캐시 솔루션은 토큰을 계속 사용할 수 있도록 하고 요청 비율이 너무 높을 때 Amazon Cognito의 요청 거부를 방지합니다. 클라이언트 측 앱은 메모리 캐시에 토큰을 저장해야 합니다. 서버 측 앱은 암호화된 캐시 메커니즘을 추가하여 토큰을 저장할 수 있습니다.

사용자 풀이 대량의 사용자 또는 machine-to-machine 활동을 생성하는 경우 Amazon Cognito에서 설정할 수 있는 토큰 요청 수에 대해 설정한 한도에 도달할 수 있습니다. Amazon Cognito 엔드포인트에 대한 요청 수를 줄이기 위해 인증 데이터를 안전하게 저장 및 재사용하거나 지수 백오프 및 재시도를 구현할 수 있습니다.

인증 데이터는 두 가지 종류의 클래스 엔드포인트에서 제공됩니다. Amazon Cognito OAuth 2.0 엔드포인트에는 클라이언트 자격 증명 및 호스팅 UI 인증 코드 요청 서비스를 제공하는 토큰 엔드포인트가 포함됩니다. 서비스 엔드포인트는 `InitiateAuth` 및 `RespondToAuthChallenge`와 같은 사용자 풀 API 요청에 응답합니다. 각 요청 유형에는 고유한 제한이 있습니다. 제한에 대한 자세한 내용은 Amazon Cognito의 할당량 섹션을 참조하세요.

Amazon API Gateway를 사용한 machine-to-machine 액세스 토큰 캐싱

API Gateway 토큰 캐싱을 사용하면 Amazon Cognito OAuth 엔드포인트의 기본 요청 속도 할당량보다 큰 이벤트에 대한 응답으로 앱을 확장할 수 있습니다.

캐시된 토큰이 만료된 경우에만 앱이 새 액세스 토큰을 요청하도록 액세스 토큰을 캐시할 수 있습니다. 그렇지 않으면 캐싱 엔드포인트가 캐시에서 토큰을 반환합니다. 이렇게 하면 Amazon Cognito API 엔드포인트에 대한 추가 호출이 방지됩니다. Amazon API Gateway를 Token 엔드포인트에 대한 프록시로 사용하면 API가 요청 할당량에 기여하는 대부분의 요청에 응답하여 속도 제한으로 인해 실패한 요청을 방지합니다.

다음 API 게이트웨이 기반 솔루션은 짧은 대기 시간, 낮은 코드/코드 없는 토큰 캐싱 구현을 제공합니다. API Gateway API는 전송 중 암호화되며 필요에 따라 미사용 시에도 암호화됩니다. API Gateway 캐시는 OAuth 2.0 클라이언트 자격 증명 부여에 적합합니다. OAuth 2.0 클라이언트 자격 증명 부여는 권한 부여 및 마이크로서비스 세션을 위한 액세스 토큰을 생성하는 대량의 권한 부여 machine-to-machine 유형입니다. 트래픽 급증과 같이 마이크로서비스가 수평적으로 확장되는 경우 많은 시스템에서 사용자 풀 또는 앱 클라이언트의 AWS 요청 속도 제한을 초과하는 볼륨으로 동일한 클라이언트 자격 증명을 사용하게 될 수 있습니다. 앱 가용성을 유지하고 지연 시간을 줄이려면 이러한 시나리오에서는 캐싱 솔루션이 가장 좋습니다.

이 솔루션에서는 API에 캐시를 정의하여 앱에서 요청하려는 OAuth 범위 및 앱 클라이언트의 각 조합에 대해 별도의 액세스 토큰을 저장합니다. 앱이 캐시 키와 일치하는 요청을 하면 API는 캐시 키와 일치하는 첫 번째 요청에 대해 Amazon Cognito가 발급한 액세스 토큰으로 응답합니다. 캐시 키 기간이 만료되면 API는 요청을 토큰 엔드포인트로 전달하고 새 액세스 토큰을 캐싱합니다.

> **Note**
>
> 캐시 키 지속 시간은 앱 클라이언트의 액세스 토큰 기간보다 짧아야 합니다.

캐시 키는 `scope` URL 파라미터에서 요청하는 OAuth 범위와 요청의 `Authorization` 헤더의 조합입니다. 이 `Authorization` 헤더에는 앱 클라이언트 ID와 클라이언트 암호가 포함됩니다. 이 솔루션을 구현하기 위해 앱에서 추가 로직을 구현할 필요가 없습니다. 사용자 풀 토큰 엔드포인트의 경로를 변경하려면 구성만 업데이트해야 합니다.

Redis용 토큰을 사용하여 토큰 캐싱을 구현할 수도 있습니다. ElastiCache AWS Identity and Access Management (IAM) 정책을 사용하여 세밀하게 제어하려면 Amazon DynamoDB 캐시를 사용해 보세요.

> **Note**
>
> API Gateway의 캐싱에는 추가 비용이 부과됩니다. 자세한 내용은 요금을 참조하세요.

API Gateway를 사용하여 캐싱 프록시를 설정하려면 다음과 같이 하세요.

1. API Gateway 콘솔을 열고 REST API를 생성합니다.
2. Resources(리소스)에서 POST 메서드를 생성합니다.
   a. HTTP Integration type(통합 유형)을 선택합니다.
   b. Use HTTP proxy integration(HTTP 프록시 통합 사용)을 선택합니다.
   c. `https://<your user pool domain>/oauth2/token`의 Endpoint URL(엔드포인트 URL)을 입력합니다.
3. Resources(리소스)에서 캐시 키를 구성합니다.
   a. POST 메서드의 Method request(메서드 요청)를 편집합니다.
   b. `scope` 파라미터와 `Authorization` 헤더를 캐싱 키로 설정합니다.
      i. URL query string parameters(URL 쿼리 문자열 파라미터)에 쿼리 문자열을 추가하고 `scope` 문자열에 대해 Caching(캐싱)을 선택합니다.
      ii. HTTP request headers(HTTP 요청 헤더)에 헤더를 추가하고 `Authorization` 헤더에 대해 Caching(캐싱)을 선택합니다.
4. Stages(단계)에서 캐싱을 구성합니다.
   a. 수정할 단계를 선택합니다.
   b. Settings(설정)에서 Enable API cache(API 캐시 활성화)를 선택합니다.
   c. Cache capacity(캐시 용량)를 선택합니다.

d. 3600초 이상의 캐시 time-to-live (TTL) 를 선택하십시오.

e. 권한 부여 필요 확인란의 선택을 취소합니다.

5. Stages(단계)에서 Invoke URL(URL 호출)을 기록해 둡니다.

6. 사용자 풀의 `/oauth2/token` 엔드포인트 대신 API의 Invoke URL(URL 호출)에 대한 POST 토큰 요청으로 앱을 업데이트합니다.

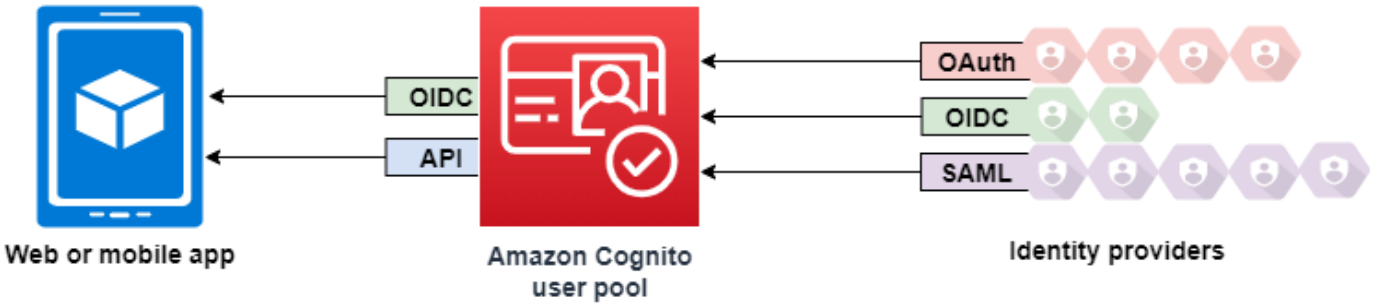
성공적인 사용자 풀 인증 후 리소스 액세스

앱 사용자는 사용자 풀을 통해 직접 로그인하거나 타사 ID 공급자 (IdP) 를 통해 페더레이션할 수 있습니다. 사용자 풀은 페이스북, 구글, 아마존, 애플을 통한 소셜 로그인과 OpenID Connect (OIDC) 및 SAML에서 반환되는 토큰을 처리하는 오버헤드를 관리합니다. IdPs 자세한 정보는 사용자 풀에 토큰 사용을 참조하세요.

인증 성공 이후 앱은 Amazon Cognito에서 사용자 풀 토큰을 받습니다. 사용자 풀 토큰을 사용하여 다음을 수행할 수 있습니다.

- Amazon DynamoDB 및 Amazon AWS 서비스 S3와 같은 애플리케이션 리소스에 대한 요청을 승인하는 AWS 자격 증명을 검색합니다.
- 취소 가능한 임시 인증 증명을 제공하십시오.
- 앱의 사용자 프로필에 ID 데이터를 입력합니다.
- 사용자 풀 디렉터리에서 로그인한 사용자의 프로필 변경을 승인합니다.
- 액세스 토큰으로 사용자 정보 요청을 승인합니다.
- 액세스 토큰을 사용하여 액세스가 보호되는 외부 API 뒤에 있는 데이터에 대한 요청을 승인합니다.
- Amazon 검증 권한으로 클라이언트 또는 서버에 저장된 애플리케이션 자산에 대한 액세스를 승인합니다.

자세한 내용은 사용자 풀 인증 흐름 및 사용자 풀에 토큰 사용 섹션을 참조하세요.



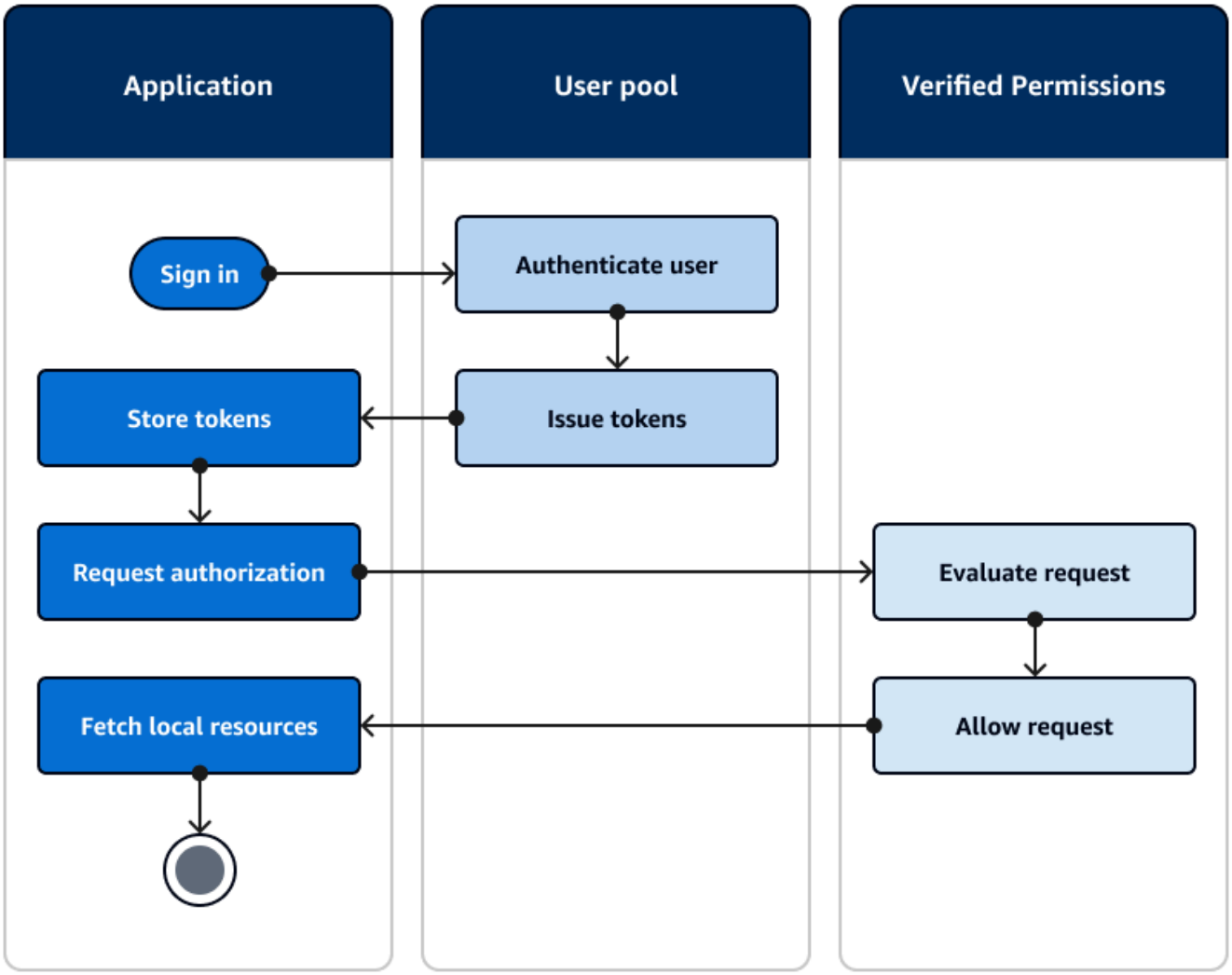
주제

- Amazon 검증 권한으로 클라이언트 또는 서버 리소스에 대한 액세스 권한 부여
- 로그인 후 API Gateway로 리소스에 액세스
- 로그인 후 자격 증명 풀을 AWS 서비스 사용하여 액세스

Amazon 검증 권한으로 클라이언트 또는 서버 리소스에 대한 액세스 권한 부여

앱은 로그인한 사용자의 토큰을 Amazon 검증 권한으로 전달할 수 있습니다. Verified Permissions는 사용자가 구축한 사용자 지정 애플리케이션을 위한 확장 가능하고 세분화된 권한 관리 및 권한 부여 서비스입니다. Amazon Cognito 사용자 풀은 검증된 권한 정책 스토어의 자격 증명 소스가 될 수 있습니다. 검증된 권한은 사용자 풀 토큰의 보안 주체 및 해당 속성과 같은 요청된 작업 및 리소스에 `GetPhoto` 대한 `premium_badge.png` 권한 부여 결정을 내립니다.

다음 다이어그램은 애플리케이션이 권한 부여 요청에서 사용자 토큰을 Verified Permissions에 전달하는 방법을 보여줍니다.



Amazon 검증 권한으로 시작하기

사용자 풀을 검증된 권한과 통합하면 모든 Amazon Cognito 앱에 대한 세분화된 권한 부여의 중앙 소스를 확보할 수 있습니다. 이렇게 하면 모든 앱 간에 코딩하고 복제해야 하는 세밀한 보안 로직이 필요하지 않습니다. 검증된 권한을 통한 권한 부여에 대한 자세한 내용은 을 참조하십시오. Amazon Verified Permissions를 통한 권한 부여

검증된 권한 권한 부여 요청에는 AWS 자격 증명이 필요합니다. 다음 기술 중 일부를 구현하여 인증 요청에 자격 증명을 안전하게 적용할 수 있습니다.

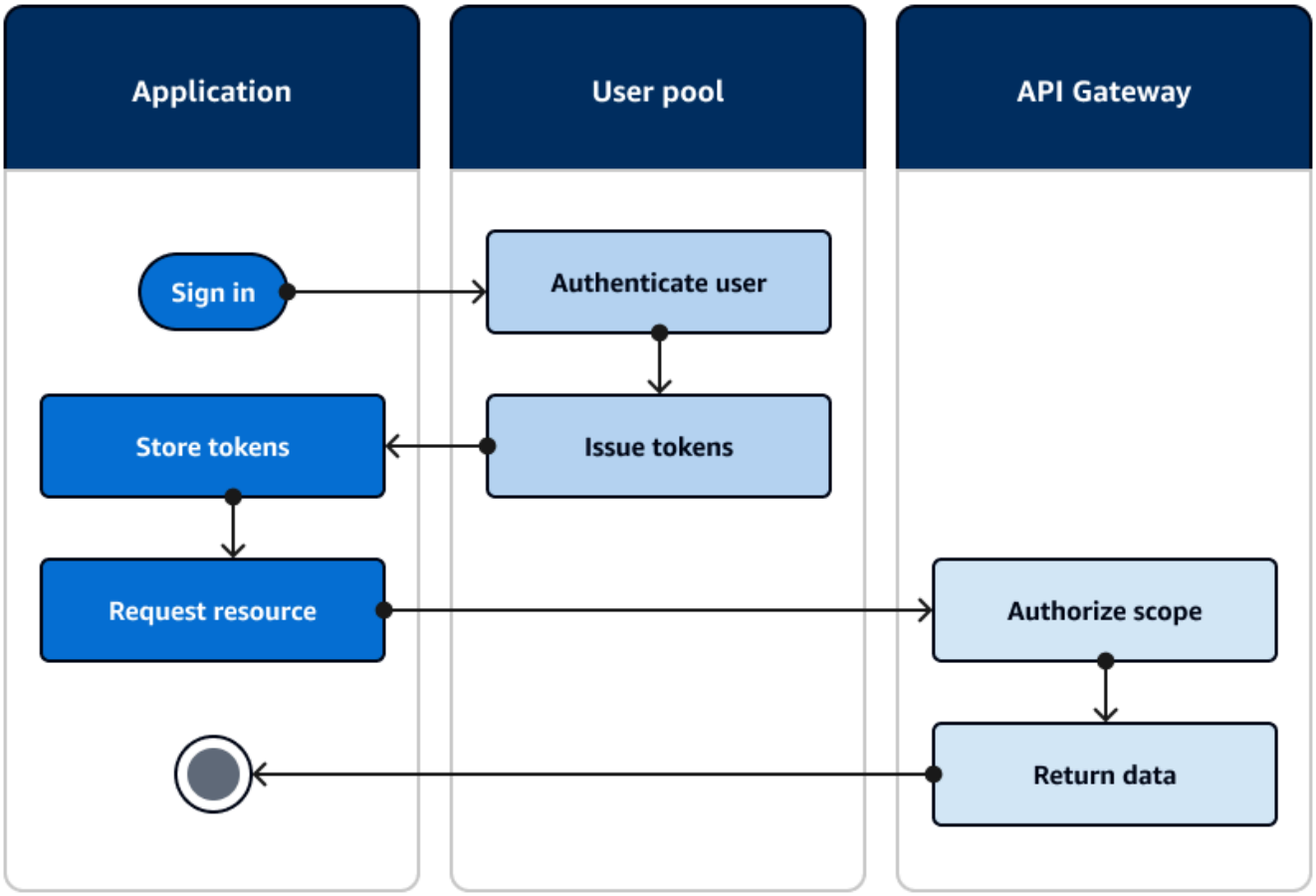
- 서버 백엔드에 비밀을 저장할 수 있는 웹 애플리케이션을 운영하십시오.
- 인증된 자격 증명 풀 자격 증명을 획득하십시오.

- access-token-authorized API를 통해 사용자 요청을 프록시하고 요청에 AWS 자격 증명을 추가합니다.

로그인 후 API Gateway로 리소스에 액세스

Amazon Cognito 사용자 풀 토큰은 일반적으로 API Gateway REST API에 대한 요청을 승인하는 데 사용됩니다. 액세스 토큰의 OAuth 2.0 범위는 for와 같은 메서드와 경로를 승인할 수 있습니다. HTTP `GET /app_assets` ID 토큰은 API에 대한 일반 인증 역할을 할 수 있으며 사용자 속성을 백엔드 서비스에 전달할 수 있습니다. API Gateway에는 HTTP API용 JWT 권한 부여자 및 보다 세분화된 로직을 적용할 수 있는 Lambda 권한 부여자와 같은 추가 사용자 지정 권한 부여 옵션이 있습니다.

다음 다이어그램은 액세스 토큰에 OAuth 2.0 범위를 사용하여 REST API에 대한 액세스 권한을 얻는 애플리케이션을 보여줍니다.



앱은 인증된 세션에서 토큰을 수집하여 요청의 헤더에 전달자 토큰으로 추가해야 합니다. `Authorization` API, 경로, 메서드에 맞게 구성한 권한 부여자를 구성하여 토큰 콘텐츠를 평가하세요. API Gateway는 요청이 권한 부여자에 대해 설정한 조건과 일치하는 경우에만 데이터를 반환합니다.

API Gateway API가 애플리케이션으로부터의 액세스를 승인할 수 있는 몇 가지 잠재적 방법은 다음과 같습니다.

- 액세스 토큰에는 올바른 OAuth 2.0 범위가 포함됩니다. REST API용 Amazon Cognito 사용자 풀 권한 부여자는 진입 장벽이 낮은 일반적인 구현입니다. 또한 이 유형의 권한 부여자에 대한 요청의 본문, 쿼리 문자열 파라미터 및 헤더를 평가할 수 있습니다.
- ID 토큰은 유효하며 만료되지 않았습니다. Amazon Cognito 권한 부여자에게 ID 토큰을 전달하면 애플리케이션 서버의 ID 토큰 콘텐츠에 대한 추가 검증을 수행할 수 있습니다.
- 액세스 또는 ID 토큰의 그룹, 클레임, 속성 또는 역할은 Lambda 함수에서 정의한 요구 사항을 충족합니다. Lambda 권한 부여자는 요청 헤더의 토큰을 파싱하고 이를 평가하여 승인 결정을 내립니다. 함수에서 사용자 지정 로직을 구성하거나 Amazon 검증 권한에 API 요청을 보낼 수 있습니다.

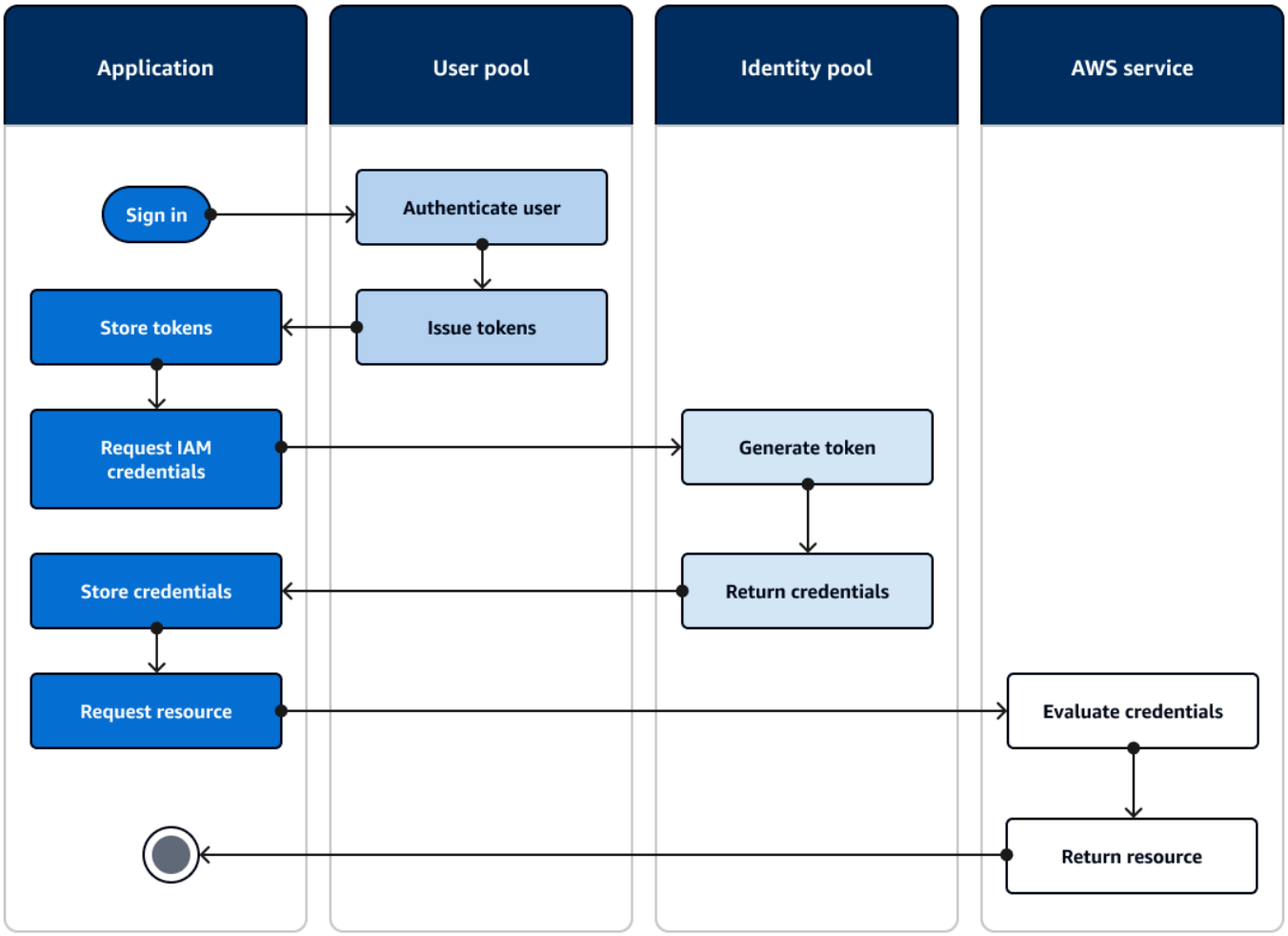
사용자 풀의 토큰을 사용하여 AWS AppSync GraphQL API에 대한 요청을 승인할 수도 있습니다.

로그인 후 자격 증명 풀을 AWS 서비스 사용하여 액세스

사용자는 사용자 풀로 로그인한 후 자격 증명 풀에서 발급한 임시 API 자격 AWS 서비스 증명으로 액세스할 수 있습니다.

웹 또는 모바일 앱은 사용자 풀로부터 토큰을 받습니다. 사용자 풀을 자격 증명 풀의 ID 공급자로 구성하면 자격 증명 풀이 토큰을 임시 AWS 자격 증명으로 교환합니다. 이러한 자격 증명의 범위는 제한된 리소스 세트에 대한 액세스를 사용자에게 부여하는 IAM 역할 및 해당 정책으로 제한될 수 있습니다. AWS 자세한 정보는 자격 증명 풀(페더레이션 자격 증명) 인증 흐름을 참조하세요.

다음 다이어그램은 애플리케이션이 사용자 풀로 로그인하고, 자격 증명 풀 자격 증명을 검색하고, 에서 자산을 요청하는 방법을 보여줍니다. AWS 서비스



자격 증명 풀 자격 증명을 사용하여 다음을 수행할 수 있습니다.

- 사용자의 자격 증명을 사용하여 Amazon Verified Permissions에 세분화된 권한 부여 요청을 하십시오.
- IAM과의 연결을 승인하는 Amazon API Gateway REST API AWS AppSync 또는 GraphQL API에 연결합니다.
- IAM과의 연결을 승인하는 Amazon DynamoDB 또는 Amazon RDS와 같은 데이터베이스 백엔드에 연결합니다.
- Amazon S3 버킷에서 애플리케이션 자산을 검색합니다.
- Amazon WorkSpaces 가상 데스크톱으로 세션을 시작합니다.

자격 증명 풀은 사용자 풀을 사용한 인증된 세션 내에서만 독점적으로 작동하지 않습니다. 또한 타사 ID 공급자로부터 직접 인증을 수락하고 인증되지 않은 게스트 사용자에 대한 자격 증명을 생성할 수 있습니다.

자격 증명 풀을 사용자 풀 그룹과 함께 사용하여 AWS 리소스에 대한 액세스를 제어하는 방법에 대한 자세한 내용은 및 을 참조하십시오사용자 풀에 그룹 추가. 역할 기반 액세스 제어 사용 또한 ID 풀에 대한 자세한 내용은 및 AWS Identity and Access Management을 참조하십시오자격 증명 풀 개념.

를 사용하여 사용자 풀을 설정합니다. AWS Management Console

Amazon Cognito 사용자 풀을 생성하고 사용자 풀 ID와 각 클라이언트 앱의 앱 클라이언트 ID를 기록해 둡니다. 사용자 풀 생성에 대한 자세한 내용은 사용자 풀 시작하기 섹션을 참조하세요.

를 사용하여 자격 증명 풀을 설정합니다. AWS Management Console

다음 절차는 를 사용하여 자격 증명 풀을 하나 이상의 사용자 풀 및 클라이언트 앱과 통합하는 방법을 설명합니다. AWS Management Console

Amazon Cognito 사용자 풀 ID 제공업체(idP)를 추가하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. ID 제공업체 추가를 선택합니다.
4. Amazon Cognito 사용자 풀을 선택합니다.
5. 사용자 풀 ID와 앱 클라이언트 ID를 입력합니다.
6. Amazon Cognito가 이 공급자를 통해 인증된 사용자에게 보안 인증을 발급할 때 요청하는 역할을 설정하려면 역할 설정을 구성하세요.

    a. 인증된 역할을 구성할 때 설정한 기본 역할을 해당 IdP의 사용자에게 제공하거나 규칙을 사용하여 역할을 선택할 수 있습니다. Amazon Cognito 사용자 풀 IdP를 사용하면 토큰에서 preferred_role 클레임을 포함한 역할을 선택할 수도 있습니다. `cognito:preferred_role` 클레임에 대한 자세한 내용은 그룹에 우선 순위 값 할당을 참조하세요.

        i. 규칙이 포함된 역할 선택을 선택한 경우 사용자 인증의 소스 클레임, 클레임을 규칙과 비교하는 데 사용할 운영자, 이 역할 선택과 일치하도록 하는 값, 역할 할당이 일치할 때 할당하려는 역할을 입력합니다. 다른 조건에 따라 추가 규칙을 생성하려면 다른 항목 추가를 선택합니다.

        ii. 토큰에서 preferred_role 클레임을 포함하는 역할 선택을 선택한 경우 Amazon Cognito는 사용자 클레임에서 해당 역할에 대한 자격 증명을 발급합니다.

`cognito:preferred_role` 기본 역할 클레임이 없는 경우 Amazon Cognito는 역할 해결을 기반으로 보안 인증 정보를 발급합니다.

b. 역할 해결을 선택합니다. 사용자의 클레임이 규칙과 일치하지 않는 경우 보안 인증을 거부하거나 인증된 역할의 보안 인증을 발급할 수 있습니다.

7. Amazon Cognito가 이 공급자를 통해 인증한 사용자에게 보안 인증을 발급할 때 할당하는 보안 주체 태그를 변경하려면 액세스 제어를 위한 속성을 구성합니다.

   - 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.
   - `sub` 및 `aud` 클레임 기반 보안 주체 태그를 적용하려면 기본 매핑 사용을 선택합니다.
   - 보안 주체 태그에 대한 속성의 자체 사용자 지정 스키마를 생성하려면 사용자 지정 매핑 사용을 선택합니다. 그런 다음 태그에 표시하려는 각 클레임에서 소싱하려는 태그 키를 입력합니다.

8. 변경 사항 저장(Save changes)을 선택합니다.

자격 증명 풀과 사용자 풀 통합

앱 사용자가 인증되면 자격 증명 공급자의 로그인 맵에 해당 사용자의 자격 증명 토큰을 추가합니다. 공급자 이름은 Amazon Cognito 사용자 풀 ID에 따라 다릅니다. 공급자 이름은 다음과 같은 구조를 갖습니다.

```
cognito-idp.<region>.amazonaws.com/<YOUR_USER_POOL_ID>
```

사용자 풀 ID에서 의 값을 도출할 수 있습니다. <region> 예를 들어, 사용자 풀 ID가 `us-east-1_EXAMPLE1` 이면 <region>입니다`us-east-1`. 사용자 풀 ID가 `us-west-2_EXAMPLE2` 이면 <region>입니다`us-west-2`.

JavaScript

```
var cognitoUser = userPool.getCurrentUser();

if (cognitoUser != null) {
 cognitoUser.getSession(function(err, result) {
  if (result) {
   console.log('You are now logged in.');

   // Add the User's Id Token to the Cognito credentials login map.
   AWS.config.credentials = new AWS.CognitoIdentityCredentials({
    IdentityPoolId: 'YOUR_IDENTITY_POOL_ID',
    Logins: {
```

```
     'cognito-idp.<region>.amazonaws.com/<YOUR_USER_POOL_ID>':
 result.getIdToken().getJwtToken()
    }
   });
  }
 });
}
```

Android

```
cognitoUser.getSessionInBackground(new AuthenticationHandler() {
 @Override
 public void onSuccess(CognitoUserSession session) {
  String idToken = session.getIdToken().getJWTToken();

  Map<String, String> logins = new HashMap<String, String>();
  logins.put("cognito-idp.<region>.amazonaws.com/<YOUR_USER_POOL_ID>",
 session.getIdToken().getJWTToken());
  credentialsProvider.setLogins(logins);
 }

});
```

iOS - objective-C

```
AWSServiceConfiguration *serviceConfiguration = [[AWSServiceConfiguration alloc]
 initWithRegion:AWSRegionUSEast1 credentialsProvider:nil];
AWSCognitoIdentityUserPoolConfiguration *userPoolConfiguration =
 [[AWSCognitoIdentityUserPoolConfiguration alloc] initWithClientId:@"YOUR_CLIENT_ID"
  clientSecret:@"YOUR_CLIENT_SECRET" poolId:@"YOUR_USER_POOL_ID"];
[AWSCognitoIdentityUserPool
 registerCognitoIdentityUserPoolWithConfiguration:serviceConfiguration
 userPoolConfiguration:userPoolConfiguration forKey:@"UserPool"];
AWSCognitoIdentityUserPool *pool = [AWSCognitoIdentityUserPool
 CognitoIdentityUserPoolForKey:@"UserPool"];
AWSCognitoCredentialsProvider *credentialsProvider = [[AWSCognitoCredentialsProvider
 alloc] initWithRegionType:AWSRegionUSEast1 identityPoolId:@"YOUR_IDENTITY_POOL_ID"
 identityProviderManager:pool];
```

iOS - swift

```
let serviceConfiguration = AWSServiceConfiguration(region: .USEast1,
 credentialsProvider: nil)
```

```
let userPoolConfiguration = AWSCognitoIdentityUserPoolConfiguration(clientId:
 "YOUR_CLIENT_ID", clientSecret: "YOUR_CLIENT_SECRET", poolId: "YOUR_USER_POOL_ID")
AWSCognitoIdentityUserPool.registerCognitoIdentityUserPoolWithConfiguration(serviceConfigura
 userPoolConfiguration: userPoolConfiguration, forKey: "UserPool")
let pool = AWSCognitoIdentityUserPool(forKey: "UserPool")
let credentialsProvider = AWSCognitoCredentialsProvider(regionType: .USEast1,
 identityPoolId: "YOUR_IDENTITY_POOL_ID", identityProviderManager:pool)
```

Amazon Cognito 사용자 풀의 보안 기능 사용

멀티 팩터 인증(MFA)을 사용자 풀에 추가하여 사용자의 자격 증명을 보호할 수 있습니다. MFA에서 두 번째 인증 요소를 추가하기 때문에 사용자 풀이 사용자 이름 및 암호에만 의존하지 않습니다. SMS 문자 메시지 또는 시간 동기화 방식 일회용 암호(TOTP)를 사용자 로그인에 대한 두 번째 요소로 사용할 수 있습니다. 또한 위험 기반 모델에서 조정 인증을 사용하여 다른 인증 팩터가 필요할 때를 예측할 수 있습니다. 사용자 풀 고급 보안 기능에는 조정 인증 및 손상된 자격 증명에 대한 보호 기능이 포함됩니다.

주제

- 사용자 풀에 MFA 추가
- 사용자 풀에 고급 보안 기능 추가
- 웹 ACL을 사용자 풀과 AWS WAF 연결
- 사용자 풀 대/소문자 구분
- 사용자 풀 삭제 방지
- 사용자 존재 오류 응답 관리

사용자 풀에 MFA 추가

멀티 팩터 인증(MFA)은 앱의 보안을 강화합니다. MFA는 사용자 이름 및 암호의 알고 있는 것 인증 요소에 가지고 있는 것 인증 요소를 추가합니다. SMS 문자 메시지 또는 시간 동기화 방식 일회용 암호(TOTP)를 사용자 로그인에 대한 두 번째 팩터로 사용하도록 선택할 수 있습니다.

> **Note**
>
> 새 사용자가 앱에 처음 로그인할 때 Amazon Cognito는 사용자 풀에 MFA가 필요한 경우에도 OAuth 2.0 토큰을 발급합니다. 사용자가 처음 로그인할 때 두 번째 인증 요소는 Amazon Cognito가 사용자에게 보내는 확인 메시지를 확인하는 것입니다. 사용자 풀에 MFA가 필요한

경우 Amazon Cognito는 첫 로그인 후 각 로그인 시도 중에 사용할 추가 로그인 요소를 등록하라는 메시지를 표시합니다.

조정 인증을 통해 위험 수준이 높아졌을 때 두 번째 팩터를 통한 인증을 요구하도록 사용자 풀을 구성할 수 있습니다. 사용자 풀에 조정 인증을 추가하는 방법은 사용자 풀에 고급 보안 기능 추가 섹션을 참조하세요.

MFA를 사용자 풀에 대해 `required`로 설정하면 모든 사용자가 로그인하려면 MFA를 완료해야 합니다. 로그인하려면 각 사용자가 SMS 또는 TOTP와 같은 MFA 팩터를 하나 이상 설정해야 합니다. MFA를 `required`로 설정하면 사용자 풀에서 사용자의 로그인을 허용하도록 사용자 온보딩에 MFA 설정을 포함해야 합니다.

SMS를 MFA 팩터로 활성화하는 경우 사용자가 가입할 때 전화 번호를 제공하고 해당 전화 번호를 확인하도록 요구할 수 있습니다. MFA를 `required`로 설정하고 SMS만 인증 요소로 지원하는 경우 사용자는 전화번호를 제공해야 합니다. 전화 번호가 없는 사용자는 관리자의 지원을 받아 프로필에 전화 번호를 추가해야 로그인할 수 있습니다. 확인되지 않은 전화 번호를 SMS MFA에 사용할 수 있습니다. 이 번호는 MFA가 성공한 후에 확인됨 상태를 받습니다.

MFA를 필수로 설정하고 SMS 및 TOTP를 지원되는 검증 방법으로 활성화한 경우 Amazon Cognito는 전화 번호가 없는 새 사용자에게 TOTP MFA를 설정하라는 메시지를 표시합니다. MFA를 필수로 설정하고 활성화된 유일한 MFA 방법이 TOTP인 경우 Amazon Cognito는 모든 새 사용자에게 두 번째로 로그인할 때 TOTP MFA를 설정하라는 메시지를 표시합니다. Amazon Cognito는 API 작업에 대한 InitiateAuth응답으로 TOTP MFA를 설정하는 데 어려움을 겪습니다. AdminInitiateAuth

MFA를 필수로 설정하면 호스팅 UI에 MFA를 설정하라는 메시지가 표시됩니다. 사용자 풀에서 MFA를 선택 사항으로 설정하면 호스팅된 UI에서 사용자에게 메시지를 표시하지 않습니다. 선택적 MFA를 사용하려면 사용자에게 MFA를 설정할 것인지 선택하라는 메시지를 표시한 다음 API 입력을 통해 추가 로그인 요소를 검증하도록 안내하는 인터페이스를 앱에 구축해야 합니다.

MFA 코드 제시 시도가 5번 실패하면 Amazon Cognito는 사용자 풀 인증 흐름에 설명된 기하급수적 타임아웃 잠금 프로세스를 시작합니다.

주제

- 필수 조건
- 멀티 팩터 인증 구성
- SMS 문자 메시지 MFA.
- TOTP 소프트웨어 토큰 MFA

필수 조건

MFA를 설정하기 전에 다음을 고려하세요.

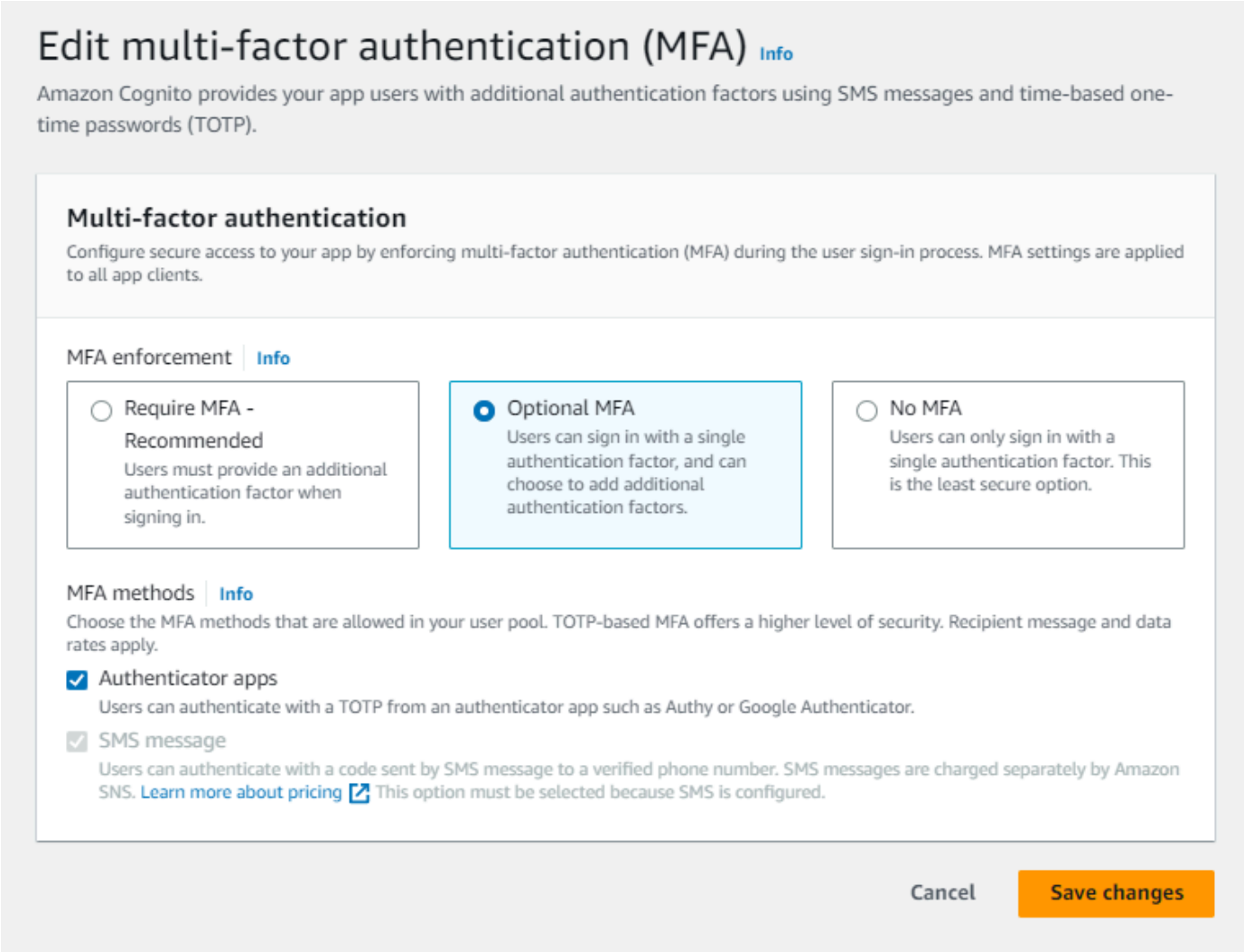
- 사용자 풀에서 MFA를 활성화하고 SMS 문자 메시지(SMS text message)를 두 번째 팩터로 선택하면 Amazon Cognito에서 확인하지 않은 전화 번호 속성에 SMS 메시지를 보낼 수 있습니다. 사용자가 SMS MFA를 완료하면 Amazon Cognito에서 `phone_number_verified` 속성을 `true`로 설정합니다.
- 사용자 풀에 대한 Amazon Simple Notification Service (Amazon SNS) 리소스가 포함된 SMS 샌드박스에 계정이 AWS 리전 있는 경우, SMS 메시지를 보내려면 먼저 Amazon SNS에서 전화번호를 확인해야 합니다. 자세한 내용은 Amazon Cognito 사용자 풀의 SMS 메시지 설정 섹션을 참조하세요.
- 고급 보안 기능을 사용하려면 Amazon Cognito 사용자 풀 콘솔에서 MFA를 활성화하고 선택 사항으로 설정해야 합니다. 자세한 내용은 사용자 풀에 고급 보안 기능 추가 섹션을 참조하세요.

멀티 팩터 인증 구성

Amazon Cognito 콘솔에서 MFA를 구성할 수 있습니다.

Amazon Cognito 콘솔에서 MFA를 구성하려면

1. Amazon Cognito 콘솔에 로그인합니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. [로그인 환경(Sign-in experience)] 탭을 선택합니다. 멀티 팩터 인증(Multi-factor authentication)을 찾아서 편집(Edit)을 선택합니다.
5. 사용자 풀에 사용하려는 MFA 시행(MFA enforcement) 방법을 선택합니다.



a. MFA 필수(Require MFA). 사용자 풀의 모든 사용자가 추가 SMS 코드 또는 시간 동기화 방식 일회용 암호(TOTP) 팩터로 로그인해야 합니다.

b. 선택적 MFA(Optional MFA) - 사용자에게 추가 로그인 팩터를 등록하는 옵션을 제공할 수 있지만 MFA를 구성하지 않은 사용자의 로그인도 허용할 수 있습니다. 조정 인증을 사용하는 경우 이 옵션을 선택합니다. 조정 인증에 대한 자세한 내용은 사용자 풀에 고급 보안 기능 추가 섹션을 참조하세요.

c. MFA 없음(No MFA). 사용자는 추가 로그인 요소를 등록할 수 없습니다.

6. 앱에서 지원하는 MFA 방법(MFA methods)을 선택합니다. SMS 메시지(SMS message) 또는 TOTP 생성 인증자 앱(Authenticator apps)을 두 번째 요소로 설정할 수 있습니다. 계정 복구에서 SMS 메시지를 사용할 수 있도록 TOTP 기반 MFA를 구현하는 것이 좋습니다.

7. SMS 문자 메시지를 두 번째 팩터로 사용하는데 SMS 메시지용 Amazon Simple Notification Service(Amazon SNS)와 함께 사용하도록 구성된 IAM 역할이 없는 경우 콘솔에서 이러한 역할을 생성합니다. 사용자 풀에 대한 메시징(Messaging) 탭에서 SMS를 찾아서 편집(Edit)을 선택합니

다. Amazon Cognito가 사용자에게 SMS 메시지를 전송하도록 허용하는 기존 역할을 사용할 수도 있습니다. 자세한 내용은 IAM 역할을 참조하세요.

8. 변경 사항 저장(Save changes)을 선택합니다.

SMS 문자 메시지 MFA.

사용자가 MFA를 사용하여 로그인할 때는 먼저 사용자 이름과 암호를 입력하고 제출합니다. 권한 부여 코드가 전송된 위치를 나타내는 `getMFA` 응답이 클라이언트 앱에 수신됩니다. 클라이언트 앱은 코드를 찾을 수 있는 위치(예: 코드가 전송된 전화 번호)를 사용자에게 표시해야 합니다. 다음으로 코드를 입력하기 위한 양식을 제공합니다. 마지막으로 클라이언트 앱이 로그인 프로세스를 완료하기 위해 코드를 제출합니다. 대상이 마스킹되고 전화번호의 마지막 네 자리를 제외한 모든 숫자가 숨겨집니다. 앱이 Amazon Cognito 호스팅 UI를 사용하는 경우 사용자에게 MFA 코드를 입력하는 페이지를 표시합니다.

SMS 문자 메시지 인증 코드는 앱 클라이언트에 대해 설정한 Authentication flow session duration(인증 흐름 세션 기간) 동안 유효합니다.

App clients and analytics(앱 클라이언트 및 분석)에서 앱 클라이언트를 수정할 때 App integration(앱 통합) 탭의 Amazon Cognito 콘솔에서 인증 흐름 세션의 기간을 설정합니다. `CreateUserPoolClient` 또는 `UpdateUserPoolClient` API 요청에서 인증 흐름 세션 기간을 설정할 수도 있습니다. 자세한 정보는 사용자 풀 인증 흐름을 참조하세요.

SMS 문자 메시지 MFA 코드가 전송된 장치에 액세스할 수 있는 권한이 없는 더 이상 없는 사용자는 고객 서비스 센터에 도움을 요청해야 합니다. 필요한 AWS 계정 권한이 있는 관리자는 AWS CLI 또는 API를 통해서만 사용자의 전화번호를 변경할 수 있습니다.

사용자가 SMS 문자 메시지 MFA 흐름을 성공적으로 통과하면 해당 사용자의 전화 번호도 확인됨으로 표시됩니다.

> **Note**
>
> MFA을 위한 SMS 요금은 따로 부과됩니다. 이메일 주소로 확인 코드를 전송하는 것은 무료입니다. Amazon SNS 요금에 대한 자세한 내용은 전 세계 SMS 요금을 참조하세요. 현재 SMS 메시징이 가능한 국가의 목록은 지원되는 리전 및 국가를 참조하세요.

> **Important**
>
> 전화 번호 및 SMS 문자 메시지 MFA를 확인하기 위해 SMS 메시지가 전송되도록 하려면 Amazon SNS의 증가된 지출 한도를 요청해야 합니다.

Amazon Cognito는 Amazon SNS를 사용하여 사용자에게 SMS 메시지를 전송합니다. Amazon SNS가 전달하는 SMS 메시지 수에는 지출 한도가 적용됩니다. AWS 계정 및 개별 메시지에 대해 지출 한도를 지정할 수 있으며, SMS 메시지 전송 비용에만 한도가 적용됩니다.
계정당 기본 지출 한도는(지정되지 않은 경우) 매월 1.00USD입니다. 한도를 높이려면 AWS Support 센터에서 SNS 한도 증가 사례를 제출하세요. New limit value(새 한도 값)의 경우 원하는 월 지출 한도를 입력합니다. Use Case Description(사용 사례 설명) 필드에서 SMS 월 지출 한도 인상을 요청하고 있음을 설명합니다.

사용자 풀에 MFA를 추가하는 방법은 사용자 풀에 MFA 추가 섹션을 참조하세요. 사용자 풀에서 Amazon SNS를 사용하는 SMS 메시지에 대한 자세한 내용은 을 참조하십시오Amazon Cognito 사용자 풀의 SMS 메시지 설정.

TOTP 소프트웨어 토큰 MFA

사용자 풀에서 TOTP 소프트웨어 토큰 MFA를 설정하면 사용자가 사용자 이름 및 암호로 로그인한 다음 TOTP를 사용하여 인증을 완료합니다. 사용자가 사용자 이름 및 암호를 설정하고 확인하면 MFA에서 TOTP 소프트웨어 토큰을 활성화할 수 있습니다. 앱이 Amazon Cognito 호스팅 UI를 사용하여 사용자를 로그인하는 경우 사용자는 자신의 사용자 이름 및 암호를 제출한 다음 추가 로그인 페이지에서 TOTP 암호를 제출합니다.

Amazon Cognito 콘솔에서 사용자 풀에 대해 TOTP MFA를 활성화하거나 Amazon Cognito API 작업을 사용할 수 있습니다. 사용자 풀 수준에서 SetUserPoolMfaConfig호출하여 MFA를 구성하고 TOTP MFA를 활성화할 수 있습니다.

> **Note**
>
> 사용자 풀에 대해 TOTP 소프트웨어 토큰 MFA를 활성화하지 않는 경우 Amazon Cognito는 토큰을 사용하여 사용자를 연결하거나 확인할 수 없습니다. 이 경우 사용자는 `SoftwareTokenMFANotFoundException` 설명과 함께 `Software Token MFA has not been enabled by the userPool` 예외를 수신합니다. 나중에 사용자 풀에 대해 소프트웨어 토큰 MFA를 비활성화하면 이전에 TOTP 토큰을 연결하고 확인한 사용자는 해당 토큰을 MFA에 계속 사용할 수 있습니다.

사용자에 대해 TOTP를 구성하는 것은 사용자가 일회용 암호를 입력하여 유효성을 검사하는 보안 코드를 받는 다단계 프로세스입니다. 그런 다음 사용자에 대해 TOTP MFA를 활성화하거나 TOTP를 사용자에 대한 기본 MFA 방법으로 설정할 수 있습니다.

TOTP MFA를 요구하도록 사용자 풀을 구성하고 사용자가 호스팅 UI에서 앱에 가입하면 Amazon Cognito가 사용자 프로세스를 자동화합니다. Amazon Cognito는 사용자에게 MFA 방식을 선택하라는 메시지를 표시하고 QR 코드를 표시하여 인증 앱을 설정하고 MFA 등록을 확인합니다. 사용자가 SMS와 TOTP MFA 중에서 선택할 수 있도록 허용한 사용자 풀에서 Amazon Cognito는 사용자에게 방식을 선택할 수 있는 기회도 제공합니다. 호스팅 UI 가입 경험에 대한 자세한 내용은 Amazon Cognito 호스팅 UI에서 새 계정에 가입하는 방법 단원을 참조하세요.

> **Important**
>
> 사용자 풀과 연결된 AWS WAF 웹 ACL이 있고 웹 ACL의 규칙에 CAPTCHA가 표시되는 경우 호스팅된 UI TOTP 등록에서 복구할 수 없는 오류가 발생할 수 있습니다. CAPTCHA 작업이 있고 호스팅 UI TOTP에 영향을 주지 않는 규칙을 생성하려면 호스팅된 UI TOTP MFA를 위한 AWS WAF 웹 ACL 구성 단원을 참조하세요. AWS WAF 웹 ACL 및 Amazon Cognito에 대한 자세한 내용은 을 참조하십시오. 웹 ACL을 사용자 풀과 AWS WAF 연결

Amazon Cognito API를 사용하는 사용자 지정 UI에서 TOTP MFA를 구현하려면 Amazon Cognito 사용자 풀 API에서 사용자 MFA 구성 단원을 참조하세요.

사용자 풀에 MFA를 추가하는 방법은 사용자 풀에 MFA 추가 섹션을 참조하세요.

TOTP MFA 고려 사항 및 제한 사항

1. Amazon Cognito는 TOTP 코드를 생성하는 인증자 앱을 통해 소프트웨어 토큰 MFA를 지원합니다. Amazon Cognito는 하드웨어 기반 MFA를 지원하지 않습니다.
2. 사용자 풀이 TOTP를 구성하지 않은 사용자에 대해 TOTP를 요구하는 경우 사용자는 앱이 사용자의 TOTP MFA를 활성화하는 데 사용할 수 있는 일회성 액세스 토큰을 수신합니다. 사용자가 추가 TOTP 로그인 팩터를 등록할 때까지 후속 로그인 시도는 실패합니다.
   - `SignUp` API 작업과 함께 또는 호스팅 UI를 통해 사용자 풀에 가입하는 사용자는 가입을 완료할 때 일회성 토큰을 수신합니다.
   - 사용자를 생성하고 해당 사용자가 자신의 초기 암호를 설정하면 Amazon Cognito가 호스팅 UI에서 해당 사용자에게 일회성 토큰을 발행합니다. 사용자에 대한 영구 암호를 설정하면 사용자가 처음 로그인할 때 Amazon Cognito가 일회성 토큰을 발급합니다.
   - Amazon Cognito는 또는 API 작업으로 로그인한 관리자가 생성한 사용자에게 일회성 토큰을 발행하지 않습니다. InitiateAuthAdminInitiateAuth 사용자가 자신의 초기 암호를 설정하는 문제에 성공한 후 또는 사용자에 대한 영구 암호가 설정된 경우 Amazon Cognito는 즉시 해당 사용자에게 MFA를 설정하는 문제를 냅니다.

3. MFA가 필요한 사용자 풀의 사용자가 이미 일회성 액세스 토큰을 수신했지만 TOTP MFA를 설정하지 않은 경우 사용자는 MFA를 설정할 때까지 호스팅 UI로 로그인할 수 없습니다. 액세스 토큰 대신 `MFA_SETUP` 챌린지 또는 `session` 요청에 대한 응답 값을 사용할 수 있습니다. InitiateAuthAdminInitiateAuthAssociateSoftwareToken
4. 사용자가 TOTP를 설정한 경우 나중에 사용자 풀에 대해 TOTP가 비활성화되어도 해당 TOTP를 MFA에 사용할 수 있습니다.
5. Amazon Cognito는 SHA-1 해시 함수를 사용하여 코드를 생성하는 인증 앱의 TOTP만 허용합니다. SHA-256 해싱으로 생성된 코드는 `Code mismatch` 오류를 반환합니다.

Amazon Cognito 사용자 풀 API에서 사용자 MFA 구성

사용자가 처음 로그인하면 앱은 일회성 액세스 토큰을 사용하여 TOTP 프라이빗 키를 생성하고 해당 키를 텍스트 또는 QR 코드 형식으로 사용자에게 제공합니다. 사용자가 인증자 앱을 구성하고 후속 로그인 시도에 TOTP를 제공합니다. 앱 또는 호스팅 UI는 MFA 문제 응답에서 Amazon Cognito에 TOTP를 제공합니다.

주제

- TOTP 소프트웨어 토큰 연결
- TOTP 토큰 확인
- TOTP MFA로 로그인
- TOTP 토큰 제거

TOTP 소프트웨어 토큰 연결

TOTP 토큰을 연결하려면 일회용 암호로 검증해야 하는 비밀 코드를 사용자에게 보냅니다. 토큰 연결에는 세 가지 단계가 필요합니다.

1. 사용자가 TOTP 소프트웨어 토큰 MFA를 선택할 경우 사용자 계정에 대해 생성된 고유한 공유 암호 키 코드를 AssociateSoftwareToken호출하여 반환합니다. 액세스 토큰 또는 세션 문자열을 AssociateSoftwareToken 사용하여 권한을 부여할 수 있습니다.
2. 앱은 사용자에게 프라이빗 키 또는 프라이빗 키에서 생성한 QR 코드를 제공합니다. 사용자는 이 키를 Google Authenticator와 같은 TOTP 생성 앱에 입력해야 합니다. libqrencode를 사용하여 QR 코드를 생성할 수 있습니다.
3. 사용자가 키를 입력하거나 Google Authenticator와 같은 인증자 앱에 QR 코드를 스캔하면 앱에서 코드 생성을 시작합니다.

TOTP 토큰 확인

다음으로 TOTP 토큰을 확인합니다. 다음과 같이 사용자의 샘플 코드를 요청하고 해당 코드를 Amazon Cognito 서비스에 제공하여 사용자가 TOTP 코드를 성공적으로 생성하고 있는지 확인합니다.

1. 앱에서 사용자에게 인증자 앱을 올바르게 설정했음을 보여주는 코드를 입력하라는 메시지를 표시합니다.
2. 사용자의 인증자 앱에서 임시 암호를 표시합니다. 인증자 앱은 이 암호를 사용자에게 제공된 비밀 키를 기반으로 생성합니다.
3. 사용자가 자신의 임시 암호를 입력합니다. `VerifySoftwareToken` API 요청에서 앱이 임시 암호를 Amazon Cognito에 전달합니다.
4. Amazon Cognito는 사용자와 연결된 비밀 키를 유지하고 TOTP를 생성하여 사용자가 제공한 TOTP와 비교합니다. 일치하는 경우 `VerifySoftwareToken`에서 `SUCCESS` 응답을 반환합니다.
5. Amazon Cognito에서 TOTP 팩터를 사용자와 연결합니다.
6. `VerifySoftwareToken` 작업에서 `ERROR` 응답을 반환하는 경우 사용자의 클록이 정확하고 최대 재시도 횟수를 초과하지 않았는지 확인합니다. Amazon Cognito는 낮은 클록 스큐를 고려해서 시도 전후 30초 이내 TOTP 토큰은 허용합니다. 문제를 해결했으면 VerifySoftwareToken 작업을 다시 시도하세요.

TOTP MFA로 로그인

이 시점에서 사용자는 시간 기반 일회용 암호를 사용하여 로그인합니다. 프로세스는 다음과 같습니다.

1. 사용자는 사용자 이름과 암호를 입력하여 클라이언트 앱에 로그인합니다.
2. TOTP MFA 문제가 호출되고, 앱은 사용자에게 임시 암호를 입력하라는 메시지를 표시합니다.
3. 사용자는 연결된 TOTP 생성 앱에서 임시 암호를 가져옵니다.
4. 사용자는 클라이언트 앱에 TOTP 코드를 입력합니다. 앱이 이를 확인하라고 Amazon Cognito 서비스에 알립니다. 새 TOTP 인증 챌린지에 대한 응답을 받으려면 로그인할 때마다 를 RespondToAuthChallenge호출해야 합니다.
5. Amazon Cognito에서 토큰을 확인하면 로그인이 성공하고 사용자가 인증 흐름을 계속합니다.

TOTP 토큰 제거

마지막으로, 앱에서 사용자가 TOTP 구성을 비활성화할 수 있도록 허용해야 합니다. 현재 사용자의 TOTP 소프트웨어 토큰을 삭제할 수 없습니다. 사용자의 소프트웨어 토큰을 교체하려면 새 소프트웨어

토큰을 연결하고 확인하세요. 사용자에 대해 TOTP MFA를 비활성화하려면 SetUserMFAReference를 호출하여 사용자가 MFA를 사용하지 않거나 SMS MFA만 사용하도록 수정하십시오.

1. MFA를 재설정하려는 사용자를 위한 인터페이스를 앱에 생성합니다. 이 인터페이스에서 사용자에게 암호를 입력하라는 메시지를 표시합니다.
2. Amazon Cognito에서 TOTP MFA 챌린지를 반환하는 경우 MFAReference를 사용하여 사용자의 MFA 기본 설정을 업데이트하십시오. SetUser
3. 앱에서 사용자에게 MFA를 비활성화했음을 알리고 다시 로그인하라는 메시지를 표시합니다.

호스팅된 UI TOTP MFA를 위한 AWS WAF 웹 ACL 구성

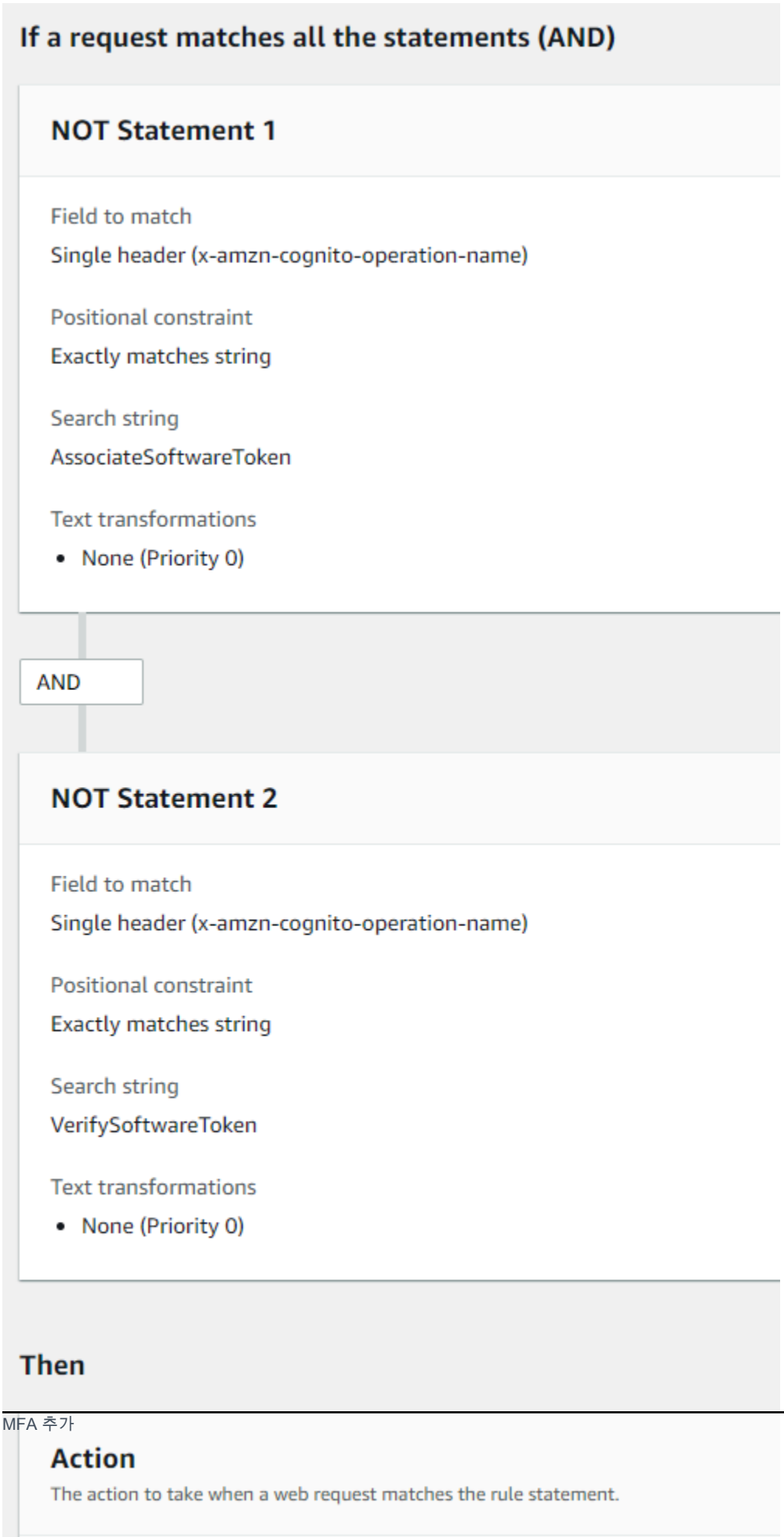
사용자 풀과 연결된 AWS WAF 웹 ACL이 있고 웹 ACL의 규칙에 CAPTCHA가 있는 경우 호스팅된 UI TOTP 등록에서 복구할 수 없는 오류가 발생할 수 있습니다. AWS WAF CAPTCHA 규칙은 이러한 방식으로 호스팅된 UI의 TOTP MFA에만 영향을 줍니다. SMS MFA는 영향을 받지 않습니다.

CAPTCHA 규칙으로 인해 사용자가 TOTP MFA 설정을 완료할 수 없으면 Amazon Cognito에서 다음 오류를 표시합니다.

Request not allowed due to WAF captcha.

이 오류는 사용자 풀이 백그라운드에서 생성하는 VerifySoftwareTokenAPI 요청에 대한 응답으로 CAPTCHA를 AWS WAF AssociateSoftwareToken입력하라는 메시지가 표시될 때 발생합니다. CAPTCHA 작업이 있고 호스팅 UI TOTP에 영향을 주지 않는 규칙을 만들려면 규칙의 CAPTCHA 작업에서 `AssociateSoftwareToken` 및 `VerifySoftwareToken`의 `x-amzn-cognito-operation-name` 헤더 값을 제외하세요.

다음 스크린샷은 헤더 값이 또는 이 아닌 모든 요청에 CAPTCHA 작업을 적용하는 예제 AWS WAF 규칙을 보여줍니다. `x-amzn-cognito-operation-name` `AssociateSoftwareToken` `VerifySoftwareToken`


If a request matches all the statements (AND)
NOT Statement 1
Field to match
Single header (x-amzn-cognito-operation-name)
Positional constraint
Exactly matches string
Search string
AssociateSoftwareToken
Text transformations
None (Priority 0)
AND
NOT Statement 2
Field to match
Single header (x-amzn-cognito-operation-name)
Positional constraint
Exactly matches string
Search string
VerifySoftwareToken
Text transformations
None (Priority 0)
Then
Action
The action to take when a web request matches the rule statement.

AWS WAF 웹 ACL 및 Amazon Cognito에 대한 자세한 내용은 을 참조하십시오. 웹 ACL을 사용자 풀과 AWS WAF 연결

사용자 풀에 고급 보안 기능 추가

사용자 풀을 생성한 후에는 Amazon Cognito 콘솔의 탐색 모음에서 [고급 보안(Advanced security)]에 액세스할 수 있습니다. 사용자 풀 고급 보안 기능을 켜고 여러 위험에 대응하여 취하는 조치를 사용자 지정할 수 있습니다. 또는 감사 모드를 사용하여 보안 완화를 적용하지 않고 탐지된 위험에 대한 지표를 수집할 수 있습니다. 감사 모드에서는 고급 보안 기능이 지표를 Amazon에 CloudWatch 게시합니다. Amazon Cognito가 첫 번째 고급 보안 이벤트를 생성한 후에 고급 보안 지표를 볼 수 있습니다. 고급 보안 지표 보기를 참조하세요.

고급 보안 기능에는 손상된 보안 인증 탐지 및 조정 인증이 포함됩니다.

손상된 보안 인증

사용자는 여러 사용자 계정에 암호를 재사용합니다. Amazon Cognito의 손상된 보안 인증 기능은 사용자 이름 및 암호의 공개 유출 데이터를 수집하여 사용자의 보안 인증을 유출된 보안 인증 목록과 비교합니다. 손상된 보안 인증 탐지는 일반적으로 추측되는 암호도 검사합니다.

손상된 보안 인증을 확인하도록 요청하는 사용자 작업과 이에 대한 응답으로 Amazon Cognito가 수행하기를 원하는 작업을 선택할 수 있습니다. 로그인, 가입, 암호 변경 이벤트의 경우, Amazon Cognito는 로그인을 차단하거나 로그인을 허용할 수 있습니다. 두 경우 모두 Amazon Cognito가 생성하는 사용자 활동 로그에서 이벤트에 대한 추가 정보를 찾을 수 있습니다.

조정 인증

Amazon Cognito는 사용자의 로그인 요청에서 위치 및 디바이스 정보를 검토하고 자동 응답을 적용하여 의심스러운 활동으로부터 사용자 풀의 사용자 계정을 보호할 수 있습니다.

고급 보안을 활성화하면 Amazon Cognito가 사용자 활동에 위험 점수를 할당합니다. 의심스러운 활동에 자동 응답을 할당할 수 있습니다. MFA를 요구하거나, 로그인을 차단하거나, 활동 세부 정보 및 위험 점수를 로깅할 수 있습니다. 또한 의심스러운 활동을 사용자에게 알리는 이메일 메시지를 자동으로 전송하여 사용자가 암호를 재설정하거나 기타 자체 안내 조치를 취할 수 있습니다.

액세스 토큰 사용자 지정

고급 보안 기능을 활성화하면 버전 2 Lambda 트리거 이벤트에 대한 응답을 수락하도록 사용자 풀을 구성할 수 있습니다. 버전 2에서는 액세스 토큰의 범위 및 기타 클레임을 사용자 지정할 수 있습니다. 이렇게 하면 사용자가 인증할 때 유연한 권한 부여 결과를 생성할 수 있는 능력이 향상됩니다. 자세한 정보는 액세스 토큰 사용자 지정을 참조하세요.

주제

고려 사항 및 제한

- Amazon Cognito 고급 보안 기능에는 추가 요금이 적용됩니다. Amazon Cognito 요금 페이지를 참조하세요.
- Amazon Cognito는 다음과 같은 표준 인증 흐름을 통해 적응형 인증 및 손상된 자격 증명 탐지를 지원합니다. `USER_PASSWORD_AUTH` `ADMIN_USER_PASSWORD_AUTH` `USER_SRP_AUTH` 고급 보안은 `CUSTOM_AUTH` 흐름과 사용자 정의 인증 챌린지 Lambda 트리거 또는 페더레이션 로그인과 함께 사용할 수 없습니다.
- 전체 기능 모드의 Amazon Cognito 고급 보안 기능을 사용하면 IP 주소 항상 차단 및 항상 허용 예외를 생성할 수 있습니다. 항상 차단(Always block) 예외 목록에 있는 IP 주소의 세션에는 조정 인증별 위험 수준이 할당되지 않으며 이러한 세션은 사용자 풀에 로그인할 수 없습니다.
- 사용자 풀의 항상 차단(Always block) 예외 목록에 있는 IP 주소의 차단된 요청은 사용자 풀에 대한 요청 비율 할당량에 기여합니다. Amazon Cognito 고급 보안 기능은 분산 서비스 거부(DDoS) 공격을 차단하지 않습니다. 사용자 풀의 대량 공격에 대한 방어를 구현하려면 웹 ACL을 추가하십시오. AWS WAF 자세한 정보는 웹 ACL을 사용자 풀과 AWS WAF 연결을 참조하세요.
- 클라이언트 자격 증명 부여는 사용자 계정과 연결되지 않은 상태에서 machine-to-machine (M2M) 인증을 위한 것입니다. 고급 보안 기능은 사용자 풀에서 사용자 계정 및 암호만 모니터링합니다. M2M 활동에 보안 기능을 구현하려면 요청 비율 및 콘텐츠 모니터링 기능을 고려해 보세요. AWS WAF 자세한 정보는 웹 ACL을 사용자 풀과 AWS WAF 연결을 참조하세요.

사전 조건

시작하려면 다음이 필요합니다.

- 앱 클라이언트가 포함된 사용자 풀. 자세한 내용은 사용자 풀 시작하기 섹션을 참조하세요.

- Amazon Cognito 콘솔에서 멀티 팩터 인증(MFA)을 [선택 사항(Optional)]으로 설정하여 위험 기반 조정 인증 기능을 사용합니다. 자세한 내용은 사용자 풀에 MFA 추가 섹션을 참조하세요.
- 이메일 알림을 사용하고 있는 경우에는 Amazon SES 콘솔로 이동하여 이메일 알림에서 사용할 이메일 주소 또는 도메인을 구성하고 확인합니다. Amazon SES에 대한 자세한 내용은 Amazon SES에서 자격 증명 확인을 참조하세요.

고급 보안 기능 구성

AWS Management Console에서 Amazon Cognito 고급 보안 기능을 구성할 수 있습니다.

사용자 풀에서 고급 보안 기능을 구성하는 방법

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 자격 증명을 입력합니다. AWS
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. [앱 통합(App integration)] 탭을 선택합니다. [고급 보안(Advanced security)]을 찾아서 [사용(Enable)]을 선택합니다. 이전에 고급 보안을 사용하도록 설정한 경우 [편집(Edit)]을 선택합니다.
5. 손상된 자격 증명 및 조정 인증에 대한 고급 보안 응답을 구성하려면 전체 기능(Full function)을 선택합니다. 정보를 수집하고 사용자 풀 데이터를 전송하려면 감사만을 선택합니다 CloudWatch. 고급 보안 요금은 감사 전용(Audit only) 및 전체 기능(Full function) 모드 둘 다에서 적용됩니다. 자세한 내용은 Amazon Cognito 요금을 참조하세요.

   작업을 활성화하기 전에 고급 보안 기능을 2주 동안 감사 모드로 유지하는 것이 좋습니다. 이 경우 Amazon Cognito가 앱 사용자의 사용 패턴을 학습할 수 있습니다.
6. 감사 전용(Audit only)을 선택한 경우 변경 사항 저장(Save changes)을 선택합니다. [전체 기능(Full function)]을 선택한 경우 다음을 수행합니다.

   a. [사용자 정의(Custom)] 작업을 수행할지 또는 [Cognito 기본값(Cognito defaults)]을 사용하여 의심스러운 [손상된 자격 증명(Compromised credentials)]에 대응할지 선택합니다. [Cognito 기본값(Cognito defaults)]은 다음과 같습니다.

      i. [로그인(Sign-in)], [가입(Sign-up)], 암호 변경(Password change)]에서 손상된 자격 증명을 탐지합니다.

      ii. [로그인 차단(Block sign-in)] 작업으로 손상된 자격 증명에 대응합니다.

   b. 손상된 자격 증명(Compromised credentials)에 대해 사용자 지정(Custom) 작업을 선택한 경우 Amazon Cognito가 이벤트 감지(Event detection)에 사용할 사용자 풀 작업 및 Amazon

Cognito가 수행하도록 할 손상된 자격 증명 응답(Compromised credentials responses)을 선택합니다. 의심스러운 손상된 자격 증명으로 [로그인 차단(Block sign-in)] 또는 [로그인 허용(Allow sign-in)]을 선택할 수 있습니다.

c. 조정 인증(Adaptive authentication)에서 악의적인 로그인 시도에 대응하는 방법을 선택합니다. 사용자 지정(Custom) 작업을 수행할지 또는 Cognito 기본값(Cognito defaults)을 사용하여 의심스러운 악의적인 활동에 대응할지 선택합니다. Cognito 기본값(Cognito defaults)을 선택한 경우 Amazon Cognito는 모든 위험 수준에서 로그인을 차단하고 사용자에게 알리지 않습니다.

d. 조정 인증(Adaptive authentication)에서 사용자 지정(Custom) 작업을 선택한 경우 Amazon Cognito에서 심각도 수준에 따라 탐지된 위험에 대응하여 수행할 자동 위험 대응(Automatic risk response)을 선택합니다. 위험 수준에 대응을 할당할 때 더 높은 수준의 위험에 덜 제한적인 대응을 할당할 수는 없습니다. 위험 수준에 할당할 수 있는 대응은 다음과 같습니다.

   i. [로그인 허용(Allow sign-in)] - 예방 조치를 수행하지 않습니다.

   ii. [선택적 MFA(Optional MFA)] - 사용자에 대해 MFA가 구성되어 있는 경우 Amazon Cognito는 사용자가 로그인할 때 항상 추가 SMS 또는 시간 기반 일회용 암호(TOTP) 팩터를 제공하도록 요구합니다. MFA가 구성되어 있지 않은 사용자는 정상적으로 로그인을 계속할 수 있습니다.

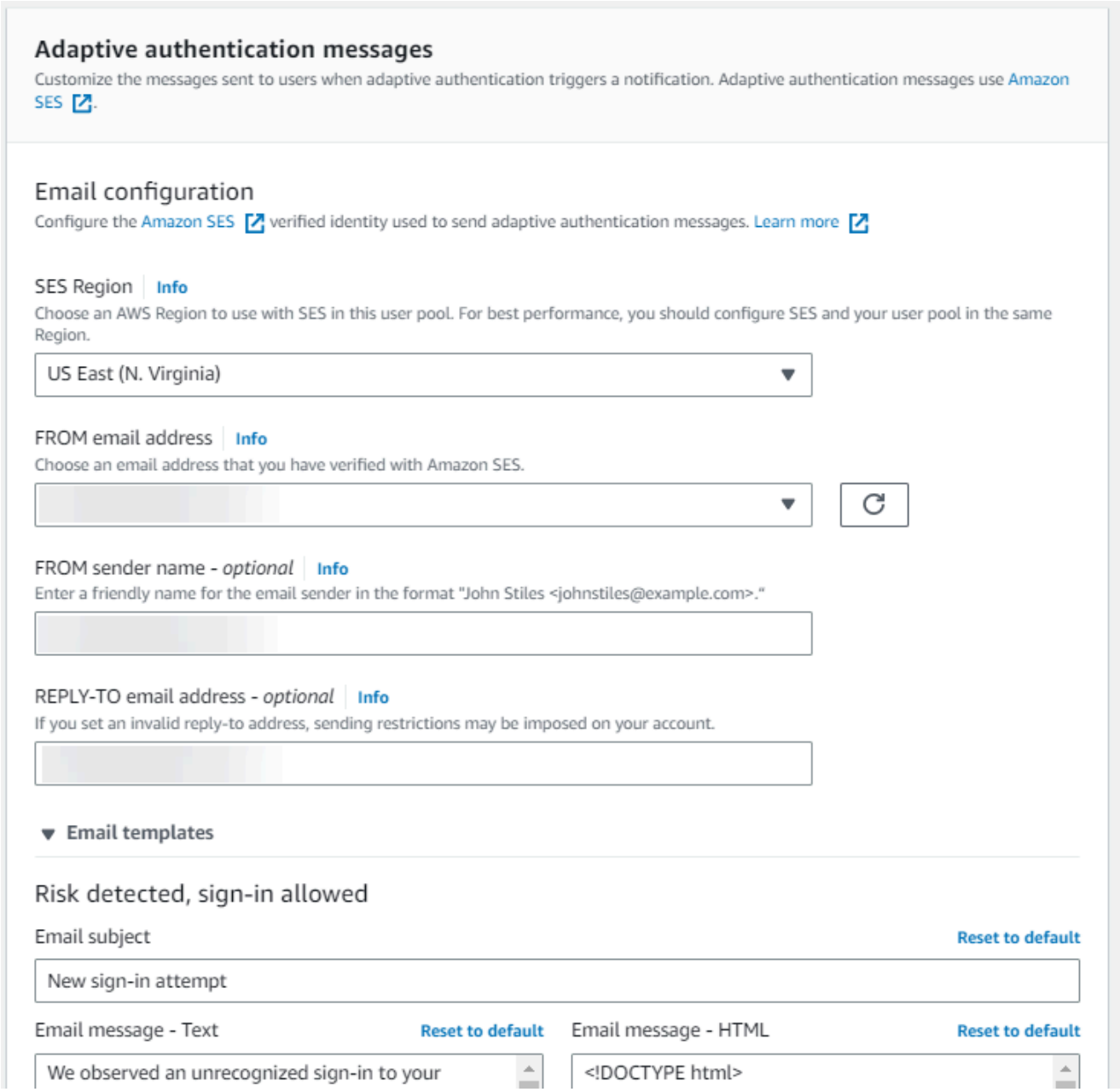
   iii. [MFA 필요(Require MFA)] - 사용자에 대해 MFA가 구성되어 있는 경우 Amazon Cognito는 사용자가 로그인할 때 항상 추가 SMS 또는 TOTP 팩터를 제공하도록 요구합니다. MFA가 구성되어 있지 않은 사용자의 경우 Amazon Cognito에서 MFA를 설정하라는 메시지를 표시합니다. 사용자에게 MFA를 자동으로 요구하기 전에 SMS MFA용 전화 번호를 캡처하거나 TOTP MFA용 인증 앱을 등록하는 메커니즘을 앱에 구성합니다.

   iv. 로그인 차단(Block sign-in) - 사용자가 로그인할 수 없도록 합니다.

   v. 사용자 알림(Notify user) - Amazon Cognito가 탐지한 위험과 수행된 대응 작업에 대한 정보가 포함된 이메일 메시지를 사용자에게 보냅니다. 보내는 메시지의 이메일 메시지 템플릿을 사용자 지정할 수 있습니다.

7. 이전 단계에서 사용자 알림(Notify user)을 선택한 경우 조정 인증용 이메일 전송 설정 및 이메일 템플릿을 사용자 지정할 수 있습니다.

   a. 이메일 구성(Email configuration)에서 조정 인증과 함께 사용할 SES 리전(SES Region), 발신 이메일 주소(FROM email address), 발신 발신자 이름(FROM sender name) 및 회신 이메일 주소(REPLY-TO email address)를 선택합니다. 사용자 풀 이메일 메시지를 Amazon Simple Email Service와 통합하는 방법에 대한 자세한 내용은 Amazon Cognito 사용자 풀에 대한 이메일 설정을 참조하세요.



b. 이메일 템플릿(Email templates)을 확장하여 HTML 버전의 이메일 메시지와 일반 텍스트 버전의 이메일 메시지 모두에서 조정 인증 알림을 사용자 지정합니다. 이메일 메시지 템플릿에 대한 자세한 내용은 메시지 템플릿 섹션을 참조하세요.

8. IP 주소 예외(IP address exceptions)를 확장하여 고급 보안 위험 평가와 관계없이 항상 허용 또는 차단되는 IPv4 또는 IPv6 주소 범위의 항상 허용(Always-allow) 또는 항상 차단(Always-block) 목록을 생성합니다. CIDR 표기법(예: 192.168.100.0/24)으로 IP 주소 범위를 지정합니다.

9. 변경 사항 저장(Save changes)을 선택합니다.

손상된 자격 증명 확인

Amazon Cognito는 사용자의 사용자 이름과 암호가 다른 곳에서 손상되었는지 탐지할 수 있습니다. 사용자가 자격 증명을 둘 이상의 사이트에서 재사용하거나 안전하지 않은 암호를 사용할 때 이러한 문제가 발생할 수 있습니다. Amazon Cognito는 사용자 이름과 암호로 호스팅 UI 및 Amazon Cognito API로 로그인하는 로컬 사용자를 확인합니다. 로컬 사용자는 외부 IdP를 통한 페더레이션 없이 사용자 풀 디렉터리에만 존재합니다.

Amazon Cognito 콘솔 내 앱 통합(App integration) 탭의 고급 보안(Advanced security)에서 손상된 보안 인증 정보(Compromised credentials)를 구성할 수 있습니다. 이벤트 감지(Event detection)를 구성하여 손상된 보안 인증 정보에 대해 모니터링할 사용자 이벤트를 선택합니다. 손상된 보안 인증 정보 응답(Compromised credentials responses)을 구성하여 손상된 보안 인증 정보가 감지된 경우 사용자를 허용할지 또는 차단할지 선택합니다. 로그인, 가입 및 암호 변경을 진행하는 동안 Amazon Cognito에서 손상된 보안 인증 정보를 확인할 수 있습니다.

로그인 허용을 선택하면 Amazon CloudWatch Logs를 검토하여 Amazon Cognito가 사용자 이벤트에 대해 수행하는 평가를 모니터링할 수 있습니다. 자세한 정보는 고급 보안 지표 보기을 참조하세요. 로그인 차단(Block sign-in)을 선택하면 Amazon Cognito는 손상된 보안 인증 정보를 이용하는 사용자의 로그인을 방지합니다. Amazon Cognito가 사용자의 로그인을 차단하면 사용자의 `UserStatus`가 `RESET_REQUIRED`로 설정됩니다. `RESET_REQUIRED` 상태의 사용자는 암호를 변경해야 다시 로그인할 수 있습니다.

> **Note**
>
> 현재 Amazon Cognito는 SRP(Secure Remote Password) 흐름을 사용한 로그인 작업에서 손상된 보안 인증을 확인하지 않습니다. SRP는 로그인 시 해시된 암호 증명을 전송합니다. Amazon Cognito는 내부적으로 암호에 액세스할 수 없으므로 클라이언트가 일반 텍스트로 전달하는 암호만 평가할 수 있습니다.
>
> Amazon Cognito는 `ADMIN_USER_PASSWORD_AUTH` 플로우가 포함된 AdminInitiateAuthAPI와 플로우가 있는 API를 사용하는 로그인에서 자격 증명이 InitiateAuth손상되었는지 확인합니다. `USER_PASSWORD_AUTH`

사용자 풀에 손상된 자격 증명 차단을 추가하는 방법은 사용자 풀에 고급 보안 기능 추가 섹션을 참조하세요.

조정 인증 사용

조정 인증을 사용하면 의심스러운 로그인을 차단하거나 위험 수준이 높아졌을 때 두 번째 인증 요소를 추가하도록 사용자 풀을 구성할 수 있습니다. Amazon Cognito는 각 로그인 시도에 대해 손상된 소스로부터 로그인 요청이 나오게 될 가능성에 대해 위험 점수를 계산합니다. 이 위험 점수는 디바이스 및 사용자 정보를 포함하는 요소를 기반으로 합니다. 적응형 인증은 Amazon Cognito가 사용자 세션에서 위험을 감지하고 사용자가 아직 MFA 방법을 선택하지 않은 경우 사용자 풀의 사용자에 대해 다중 인증(MFA)을 켜거나 요구할 수 있습니다. 사용자에 대해 MFA를 활성화하면 적응형 인증이 구성된 방식과 관계없이 인증 중에 두 번째 요소를 제공하거나 설정하라는 인증 문제가 항상 수신됩니다. 사용자의 관점에서 보면 앱이 MFA 설정을 지원하며, 선택적으로 Amazon Cognito는 추가 요소를 구성할 때까지 사용자가 다시 로그인하지 못하도록 합니다.

Amazon Cognito는 로그인 시도, 위험 수준, 실패한 챌린지를 Amazon에 게시합니다. CloudWatch 자세한 내용은 고급 보안 지표 보기 섹션을 참조하세요.

사용자 풀에 조정 인증을 추가하는 방법은 사용자 풀에 고급 보안 기능 추가 섹션을 참조하세요.

주제

- 조정 인증 개요
- API 요청에 사용자 디바이스 및 세션 데이터 추가
- 사용자 이벤트 기록 보기
- 이벤트 피드백 제공
- 알림 메시지 전송

조정 인증 개요

Amazon Cognito 콘솔의 앱 통합 탭에 있는 고급 보안에서 조정 인증에 대한 설정을 선택할 수 있습니다. 여기에는 다양한 위험 수준에서 취할 조치, 사용자에게 보내는 알림 메시지의 사용자 지정 등이 포함됩니다. 모든 앱 클라이언트에 글로벌 고급 보안 구성을 할당할 수 있지만 개별 앱 클라이언트에는 클라이언트 수준 구성을 적용할 수 있습니다.

Amazon Cognito 조정 인증은 각 사용자 세션에 높음, 중간, 낮음 또는 위험 없음 위험 수준 중 하나를 할당합니다.

Enforcement method(적용 방법)를 Audit-only(감사 전용)에서 Full-function(전체 기능)으로 변경할 때는 옵션을 신중하게 고려하세요. 위험 수준에 적용하는 자동 응답은 Amazon Cognito가 동일한 특성을 가진 후속 사용자 세션에 할당하는 위험 수준에 영향을 줍니다. 예를 들어, Amazon Cognito가 처음에

고위험으로 평가한 사용자 세션에 조치를 취하지 않거나 Allow(허용)를 선택하면 Amazon Cognito는 유사한 세션을 위험이 낮은 것으로 간주합니다.

위험 수준별로 다음 옵션 중에서 선택할 수 있습니다.

옵션	작업
허용	사용자는 추가 팩터 없이 로그인할 수 있습니다.
선택 사항 MFA	두 번째 팩터가 구성된 사용자는 로그인을 위해 두 번째 팩터 문제를 완료해야 합니다. SMS 전화 번호 및 TOTP 소프트웨어 토큰이 사용 가능한 두 번째 팩터입니다. 두 번째 인증 요소를 구성하지 않은 사용자는 하나의 자격 증명 세트로만 로그인할 수 있습니다.
MFA 필요	두 번째 팩터가 구성된 사용자는 로그인을 위해 두 번째 팩터 문제를 완료해야 합니다. Amazon Cognito는 두 번째 팩터가 구성되지 않은 사용자의 로그인을 차단합니다.
차단	Amazon Cognito는 지정된 위험 수준의 모든 로그인 시도를 차단합니다.

> **Note**
>
> 두 번째 인증 팩터인 SMS에 사용하기 위해 전화 번호를 인증할 필요가 없습니다.

API 요청에 사용자 디바이스 및 세션 데이터 추가

API를 사용하여 가입하고, 로그인하고, 암호를 재설정할 때 사용자의 세션 정보를 수집하여 Amazon Cognito 고급 보안으로 전달할 수 있습니다. 이 정보에는 사용자의 IP 주소와 고유 디바이스 식별자가 포함됩니다.

프록시 서비스나 애플리케이션 서버와 같이 사용자와 Amazon Cognito 간에 중간 네트워크 디바이스가 있을 수 있습니다. 적응형 인증이 서버나 프록시 대신 사용자 엔드포인트의 특성을 기반으로 위험을 계산하도록 사용자의 컨텍스트 데이터를 수집하여 Amazon Cognito에 전달할 수 있습니다. 클라이언트측 앱이 Amazon Cognito API 작업을 직접 호출하는 경우 적응형 인증은 소스 IP 주소를 자동으로 기

록합니다. 그러나 디바이스 지문을 수집하지 않는 한 `user-agent`와 같은 다른 디바이스 정보는 기록하지 않습니다.

Amazon Cognito 컨텍스트 데이터 수집 라이브러리를 사용하여 이 데이터를 생성하고 및 파라미터를 사용하여 Amazon Cognito 고급 보안에 제출하십시오 ContextData. UserContextData 컨텍스트 데이터 수집 라이브러리는 SDK에 포함되어 있습니다. AWS 자세한 내용은 Amazon Cognito와 웹 및 모바일 앱 통합을 참조하세요. 사용자 풀에서 고급 보안 기능을 활성화한 경우 `ContextData`를 제출할 수 있습니다. 자세한 내용은 고급 보안 기능 구성을 참조하세요.

애플리케이션 서버에서 다음 Amazon Cognito 인증 API 작업을 호출할 때 사용자 디바이스의 IP를 `ContextData` 파라미터에 전달합니다. 또한 서버 이름, 서버 경로 및 인코딩된 디바이스 지문 데이터를 전달합니다.

- AdminInitiateAuth
- AdminRespondToAuthChallenge

Amazon Cognito에서 인증하지 않은 API 작업을 호출하면 `UserContextData`를 Amazon Cognito 고급 보안 기능에 제출할 수 있습니다. 이 데이터에는 `EncodedData` 파라미터에 디바이스 지문이 포함되어 있습니다. 다음 조건을 충족하는 경우 `UserContextData`에 `IpAddress` 파라미터를 제출할 수도 있습니다.

- 사용자 풀에서 고급 보안 기능을 활성화했습니다. 자세한 내용은 고급 보안 기능 구성을 참조하세요.
- 앱 클라이언트에 클라이언트 암호가 있습니다. 자세한 내용은 사용자 풀 앱 클라이언트 구성을 참조하세요.
- 앱 클라이언트에서 추가 사용자 컨텍스트 데이터 허용(Accept additional user context data)을 활성화했습니다. 자세한 정보는 추가 사용자 컨텍스트 데이터 수락(AWS Management Console)을 참조하세요.

앱은 다음 Amazon Cognito에서 인증하지 않은 API 작업의 인코딩된 디바이스 지문 데이터와 사용자 디바이스 IP 주소로 `UserContextData` 파라미터를 채울 수 있습니다.

- InitiateAuth
- RespondToAuthChallenge
- SignUp
- ConfirmSignUp
- ForgotPassword

- ConfirmForgotPassword
- ResendConfirmationCode

추가 사용자 컨텍스트 데이터 수락(AWS Management Console)

추가 사용자 컨텍스트 데이터 허용(Accept additional user context data) 기능을 활성화하면 사용자 풀에서 `UserContextData` 파라미터의 IP 주소를 수락합니다. 다음과 같은 경우에는 이 기능을 활성화할 필요가 없습니다.

- 사용자는 와 같은 인증된 API 작업을 통해서만 로그인하고 AdminInitiateAuth , 사용자는 매개변수를 사용합니다. `ContextData`
- 인증되지 않은 API 작업에서 IP 주소가 아닌 디바이스 지문만 Amazon Cognito 고급 보안 기능으로 전송하기를 원합니다.

Amazon Cognito 콘솔에서 다음과 같이 앱 클라이언트를 업데이트하여 추가 사용자 컨텍스트 데이터에 대한 지원을 추가합니다.

1. Amazon Cognito 콘솔에 로그인합니다.
2. 탐색 창에서 사용자 풀 관리(Manage your User Pools)를 선택한 다음 편집할 사용자 풀을 선택합니다.
3. [앱 통합(App integration)] 탭을 선택합니다.
4. 앱 클라이언트 및 분석(App clients and analytics)에서 앱 클라이언트를 선택하거나 생성합니다. 자세한 내용은 사용자 풀 앱 클라이언트 구성을 참조하세요.
5. 앱 클라이언트 정보(App client information) 컨테이너에서 편집(Edit)을 선택합니다.
6. 앱 클라이언트의 고급 인증 설정(Advanced authentication settings)에서 추가 사용자 컨텍스트 데이터 허용(Accept additional user context data)을 선택합니다.
7. 변경 사항 저장를 선택합니다.

Amazon Cognito API에서 사용자 컨텍스트 데이터를 수락하도록 앱 클라이언트를 구성하려면 CreateUserPoolClientor `true` UpdateUserPoolClient요청에서 `EnablePropagateAdditionalUserContextData` 를 설정하십시오. 웹 또는 모바일 앱에서 고급 보안 기능을 활성화하는 방법은 앱에서 사용자 풀 고급 보안 기능 활성화를 참조하세요. 앱이 서버에서 Amazon Cognito를 호출하면 클라이언트 측에서 사용자 컨텍스트 데이터를 수집합니다. 다음은 JavaScript SDK 메서드를 사용하는 예제입니다. `getData`

```
var encodedData =
 AmazonCognitoAdvancedSecurityData.getData(username, userPoolId, clientId);
```

앱에서 조정 인증을 사용하도록 설계하는 경우 최신 Amazon Cognito SDK를 앱에 통합하는 것이 좋습니다. SDK의 최신 버전은 디바이스 ID, 모델 및 표준 시간대와 같은 디바이스 지문 정보를 수집합니다. Amazon Cognito SDK에 대한 자세한 내용은 사용자 풀 SDK 설치를 참조하세요. Amazon Cognito 고급 보안 기능은 앱이 올바른 형식으로 제출하는 이벤트에만 위험 점수를 저장하고 할당합니다. Amazon Cognito에서 오류 응답을 반환하는 경우 요청에 유효한 암호 해시가 포함되어 있는지, `IPaddress` 파라미터가 유효한 IPv4 또는 IPv6 주소인지 확인합니다.

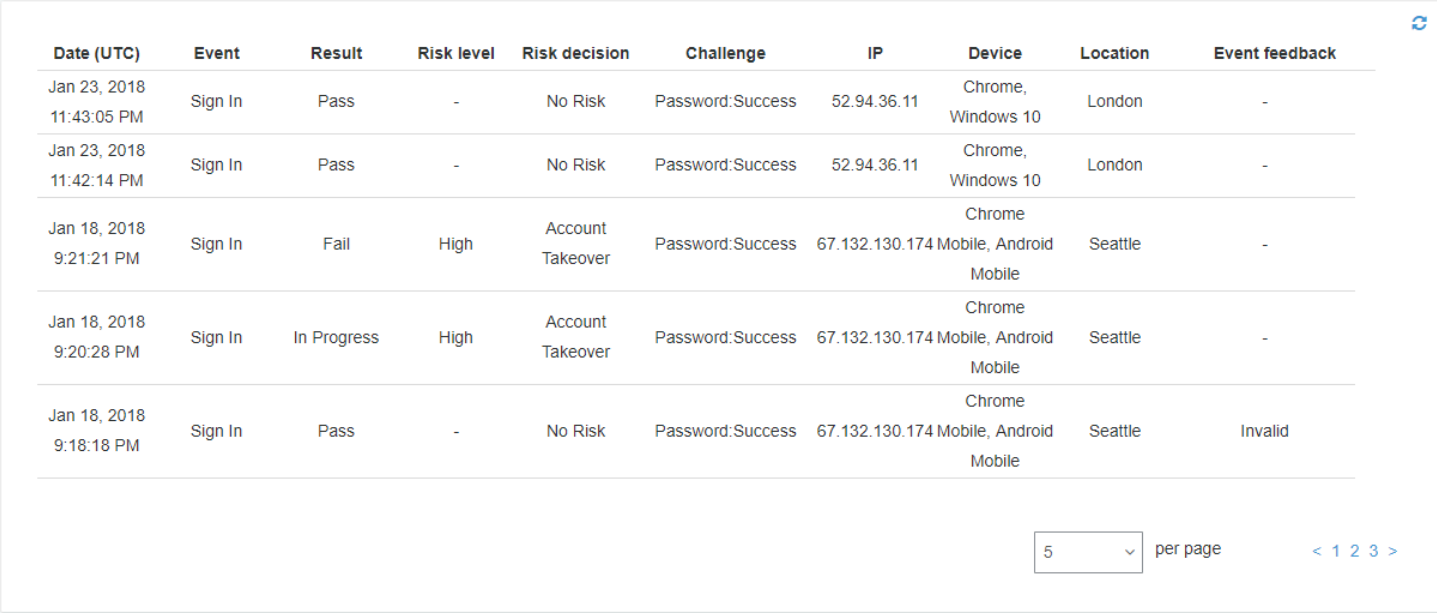
`ContextData` 및 `UserContextData` 리소스

- AWS Amplify 안드로이드용 SDK: GetUserContextData
- AWS Amplify iOS용 SDK: userContextData
- JavaScript: amazon-cognito-advanced-security -data.min.js

사용자 이벤트 기록 보기

> **Note**
>
> 새로운 Amazon Cognito 콘솔의 사용자(Users) 탭에서 사용자 이벤트 기록을 볼 수 있습니다.

사용자의 로그인 기록을 보려면 Amazon Cognito 콘솔의 Users(사용자) 탭에서 사용자를 선택합니다. Amazon Cognito는 사용자 이벤트 기록을 2년 동안 보관합니다.



Date (UTC)	Event	Result	Risk level	Risk decision	Challenge	IP	Device	Location	Event feedback
Jan 23, 2018 11:43:05 PM	Sign In	Pass	-	No Risk	Password:Success	52.94.36.11	Chrome, Windows 10	London	-
Jan 23, 2018 11:42:14 PM	Sign In	Pass	-	No Risk	Password:Success	52.94.36.11	Chrome, Windows 10	London	-
Jan 18, 2018 9:21:21 PM	Sign In	Fail	High	Account Takeover	Password:Success	67.132.130.174	Chrome Mobile, Android Mobile	Seattle	-
Jan 18, 2018 9:20:28 PM	Sign In	In Progress	High	Account Takeover	Password:Success	67.132.130.174	Chrome Mobile, Android Mobile	Seattle	-
Jan 18, 2018 9:18:18 PM	Sign In	Pass	-	No Risk	Password:Success	67.132.130.174	Chrome Mobile, Android Mobile	Seattle	Invalid

5 per page < 1 2 3 >

각 로그인 이벤트에는 이벤트 ID가 있습니다. 이벤트에는 위치, 디바이스 세부 정보 및 위험 탐지 결과와 같은 해당 컨텍스트 데이터도 있습니다. Amazon Cognito API 작업을 AdminListUserAuthEvents사용하거나 -events가 포함된 AWS Command Line Interface (AWS CLI) 를 사용하여 사용자 이벤트 기록을 쿼리할 수 있습니다. admin-list-user-auth

이벤트 ID를 이벤트를 기록한 시점에 Amazon Cognito가 발행한 토큰과 연관시킬 수도 있습니다. ID 및 액세스 토큰은 페이로드에 이 이벤트 ID를 포함합니다. Amazon Cognito는 새로 고침 토큰 사용을 원래 이벤트 ID와 연관시킵니다. 원래 이벤트 ID를 Amazon Cognito 토큰을 발급한 로그인 이벤트의 이벤트 ID로 다시 추적할 수 있습니다. 시스템 내에서 특정 인증 이벤트에 대한 토큰 사용을 추적할 수 있습니다. 자세한 정보는 사용자 풀에 토큰 사용을 참조하세요.

이벤트 피드백 제공

이벤트 피드백은 실시간으로 위험 평가에 영향을 미치고 시간이 지남에 따라 위험 평가 알고리즘을 개선합니다. 또한 Amazon Cognito 콘솔 및 API 작업을 통해 로그인 시도의 유효성에 대한 피드백을 제공할 수 있습니다.

> **Note**
>
> 이벤트 피드백은 Amazon Cognito가 동일한 특성을 가진 후속 사용자 세션에 할당하는 위험 수준에 영향을 미칩니다.

Amazon Cognito 콘솔의 Users(사용자) 탭에서 사용자를 선택하고 Provide event feedback(이벤트 피드백 제공)을 선택합니다. 이벤트 세부 정보를 검토하고 Set as valid(유효한 것으로 설정) 또는 Set as invalid(잘못된 것으로 설정)를 선택할 수 있습니다.

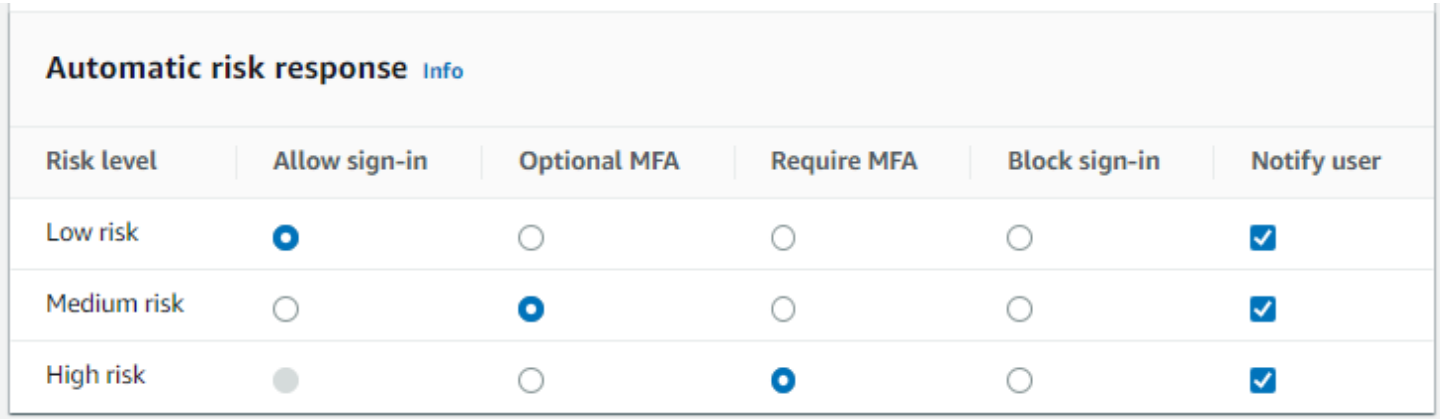
콘솔의 사용자 및 그룹(Users and groups) 탭에 로그인 기록이 나열됩니다. 항목을 선택한 경우 이벤트를 유효하거나 유효하지 않음으로 표시할 수 있습니다. 또한 사용자 풀 API AdminUpdateAuthEventFeedback작업과 -feedback AWS CLI 명령을 admin-update-auth-event 통해 피드백을 제공할 수 있습니다.

Amazon Cognito 콘솔에서 Set as valid(유효한 것으로 설정)를 선택하거나 API에서 `valid FeedbackValue` 값을 제공하면 Amazon Cognito가 일정 수준의 위험을 평가한 사용자 세션을 신뢰한다고 Amazon Cognito에 알리는 것입니다. Amazon Cognito 콘솔에서 Set as invalid(잘못된 것으로 설정)를 선택하거나 API에서 `invalid FeedbackValue` 값을 제공하면 사용자 세션을 신뢰하지 않거나 Amazon Cognito가 충분히 높은 수준의 위험을 평가했다고 믿지 않는다고 Amazon Cognito에 알리는 것입니다.

알림 메시지 전송

고급 보안 보호를 통해 Amazon Cognito는 사용자에게 위험한 로그인 시도를 알릴 수 있습니다. 또한 Amazon Cognito는 사용자에게 로그인이 유효한지 여부를 나타내는 링크를 선택하라는 메시지를 표시할 수 있습니다. Amazon Cognito는 이 피드백을 사용하여 사용자 풀의 위험 탐지 정확도를 개선합니다.

Automatic risk response(자동 위험 대응) 섹션에서 낮음, 중간 또는 높은 위험 사례에 대해 Notify Users(사용자 알림)를 선택합니다.



Amazon Cognito는 이메일 주소 확인 여부와 관계없이 사용자에게 이메일 알림을 보냅니다.

알림 이메일 메시지를 사용자 지정하고, 이러한 메시지의 일반 텍스트 버전과 HTML 버전을 모두 제공할 수 있습니다. 이메일 알림을 사용자 지정하려면 고급 보안 구성의 Adaptive authentication

messages(적응형 인증 메시지)에서 Email templates(이메일 템플릿)를 엽니다. 이메일 템플릿에 대한 자세한 내용은 메시지 템플릿 섹션을 참조하세요.

고급 보안 지표 보기

Amazon Cognito는 고급 보안 기능에 대한 메트릭을 Amazon 계정에 게시합니다. CloudWatch Amazon Cognito는 고급 보안 지표를 위험 수준별 및 요청 수준별로 함께 그룹화합니다.

콘솔에서 지표를 보려면 CloudWatch

1. https://console.aws.amazon.com/cloudwatch/ 에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표(Metrics)를 선택합니다.
3. Amazon Cognito를 선택합니다.
4. By Risk Classification(위험 분류별) 같이 집계된 지표 그룹을 선택합니다.
5. All metrics(모든 측정치) 탭에 선택한 모든 측정치가 표시됩니다. 다음을 수행할 수 있습니다.
   - 테이블을 정렬하려면 열 머리글을 사용합니다.
   - 측정치를 그래프로 표시하려면 측정치 옆에 있는 확인란을 선택합니다. 모든 지표를 선택하려면 테이블의 머리글 행에 있는 확인란을 선택합니다.
   - 리소스로 필터링하려면 리소스 ID를 선택한 후 검색에 추가를 선택합니다.
   - 지표로 필터링하려면 지표 이름을 선택한 후 검색에 추가를 선택합니다.

지표	설명	측정치 차원
CompromisedCredentialRisk	Amazon Cognito가 손상된 자격 증명을 탐지한 요청.	작업: 작업 유형 `PasswordChange` , `SignIn` 또는 `SignUp`이 유일한 차원입니다. UserPoolId: 사용자 풀의 식별자입니다. RiskLevel: 높음 (기본값), 중간 또는 낮음
AccountTakeoverRisk	Amazon Cognito가 계정 탈취 위험을 탐지한 요청.	작업: 작업 유형 `PasswordChange` , `SignIn` 또는 `SignUp`이 유일한 차원입니다.

지표	설명	측정치 차원
		UserPoolId: 사용자 풀의 식별자입니다. RiskLevel: 높음, 중간 또는 낮음
OverrideBlock	Amazon Cognito가 개발자가 제공한 구성의 결과로 차단한 요청.	작업: 작업 유형 `PasswordChange` , `SignIn` 또는 `SignUp`이 유일한 차원입니다. UserPoolId: 사용자 풀의 식별자입니다. RiskLevel: 높음, 중간 또는 낮음
Risk	Amazon Cognito가 위험한 것으로 표시한 요청.	작업: `PasswordChange` , `SignIn` 또는 `SignUp` 등의 작업 유형 UserPoolId: 사용자 풀의 식별자입니다.
NoRisk	Amazon Cognito가 어떤 위험도 식별하지 않은 요청.	작업: `PasswordChange` , `SignIn` 또는 `SignUp` 등의 작업 유형 UserPoolId: 사용자 풀의 식별자입니다.

Amazon Cognito는 바로 분석할 수 있도록 미리 정의된 두 개의 지표 그룹을 제공합니다. CloudWatch 위험 분류별(By Risk Classification)은 Amazon Cognito가 위험하다고 식별한 요청에 대한 위험의 세부 수준을 식별합니다. 요청 분류별(By Request Classification)은 요청 수준별로 집계된 지표를 반영합니다.

집계된 측정치 그룹	설명
위험 분류별	Amazon Cognito가 위험한 것으로 식별한 요청입니다.
요청 분류별	요청별로 집계된 측정치

앱에서 사용자 풀 고급 보안 기능 활성화

사용자 풀에 대한 고급 보안 기능을 구성한 후에는 웹 또는 모바일 앱에서 해당 기능을 활성화해야 합니다.

다음과 같은 고급 보안 사용 JavaScript

1. 앱에 Amazon Cognito 자격 증명 SDK를 JavaScript 추가합니다.
2. CognitoUserPool.js에서는 로 설정합니다. `AdvancedSecurityDataCollectionFlag` `true` `UserPoolId`를 사용자 풀 ID로 설정합니다.
3. 이 소스 참조를 앱 JavaScript 파일에 추가합니다. `us-east-1`,,, `us-east-2` `us-west-1` `eu-west-1` `eu-west-2`, 또는 목록에 `<region>` 있는 것으로 `eu-central-1` 바꾸십시오. AWS 리전

```
<script src="https://amazon-cognito-assets.<region>.amazoncognito.com/amazon-cognito-advanced-security-data.min.js"></script>
```

Android에서 고급 보안 기능 사용

1. AWS Amplify Android용 앱을 만들어 보세요. 자세한 내용은 AWS Amplify 개발자 센터의 프로젝트 설정을 참조하세요.
2. `userContextDataProvider`를 사용하여 인증 요청에 사용자 및 디바이스 정보를 포함합니다.

레거시 Android SDK에 사용자 컨텍스트 데이터를 추가하는 방법에 대한 자세한 내용은 aws-android-sdk-cognito identityprovider-asf를 참조하세요.

iOS에서 고급 보안 기능 사용

1. 스위프트 또는 AWS Amplify 플러터용으로 앱을 만드세요. 자세한 내용은 AWS Amplify 개발자 센터의 Swift 프로젝트 설정 및 Flutter 프로젝트 설정을 참조하세요.

2. 인증 요청에 사용자 및 디바이스 정보를 포함하세요. InitiateAuthAPI 작업과 함께 사용할 예제는 InitiateAuthInput+Amplify.Swift `userContextData` on을 참조하십시오. GitHub

레거시 iOS SDK에 사용자 컨텍스트 데이터를 추가하는 방법에 대한 자세한 내용은 AWSCognitoIdentityProviderASF를 참조하세요.

웹 ACL을 사용자 풀과 AWS WAF 연결

AWS WAF 웹 애플리케이션 방화벽입니다. AWS WAF 웹 액세스 제어 목록 (웹 ACL) 을 사용하면 호스팅된 UI 및 Amazon Cognito API 서비스 엔드포인트에 대한 원치 않는 요청으로부터 사용자 풀을 보호할 수 있습니다. 웹 ACL을 사용하면 사용자 풀이 응답하는 모든 HTTPS 웹 요청을 세부적으로 제어할 수 있습니다. AWS WAF 웹 ACL에 대한 자세한 내용은 개발자 안내서의 웹 액세스 제어 목록 (웹 ACL) 관리 및 사용을 참조하십시오.AWS WAF

사용자 풀과 연결된 AWS WAF 웹 ACL이 있는 경우 Amazon Cognito는 선택된 비기밀 헤더 및 사용자 요청 콘텐츠를 다음으로 전달합니다. AWS WAF AWS WAF 요청 내용을 검사하여 웹 ACL에 지정한 규칙과 비교한 다음 Amazon Cognito에 응답을 반환합니다.

AWS WAF 웹 ACL과 아마존 코그니토에 대해 알아야 할 사항

- 에서 차단한 요청은 모든 요청 유형의 요청 속도 할당량에 포함되지 AWS WAF 않습니다. AWS WAF 핸들러는 API 레벨 스로틀링 핸들러보다 먼저 호출됩니다.
- 웹 ACL을 생성할 때 웹 ACL이 완전히 전파되어 Amazon Cognito에서 사용할 수 있게 되기까지 약간의 시간이 걸립니다. 전파 시간은 몇 초에서 몇 분까지 걸릴 수 있습니다. AWS WAF 웹 ACL이 완전히 전파되기 전에 연결을 `WAFUnavailableEntityException`시도하면 a를 반환합니다.
- 각 사용자 풀에 하나의 웹 ACL을 연결할 수 있습니다.
- 요청으로 인해 AWS WAF 가 검사할 수 있는 한계보다 더 큰 페이로드가 발생할 수 있습니다. Amazon Cognito의 크기 초과 요청을 처리하는 방법을 구성하는 방법을 AWS WAF 알아보려면 AWS WAF 개발자 안내서의 크기 초과 요청 구성 요소 처리를 참조하십시오.
- AWS WAF 사기 방지 계정 도용 방지 (ATP) 를 사용하는 웹 ACL을 Amazon Cognito 사용자 풀과 연결할 수 없습니다. `AWS-AWSManagedRulesATPRuleSet` 관리형 규칙 그룹을 추가할 때 ATP 기능을 구현합니다. 사용자 풀과 연결하기 전에 웹 ACL에서 이 관리형 규칙 그룹을 사용하지 않는지 확인하세요.
- 사용자 풀과 연결된 AWS WAF 웹 ACL이 있고 웹 ACL의 규칙에 CAPTCHA가 표시되는 경우 호스팅된 UI TOTP 등록에서 복구할 수 없는 오류가 발생할 수 있습니다. CAPTCHA 작업이 있고 호스팅 UI TOTP에 영향을 주지 않는 규칙을 생성하려면 호스팅된 UI TOTP MFA를 위한 AWS WAF 웹 ACL 구성 단원을 참조하세요.

AWS WAF 다음 엔드포인트에 대한 요청을 검사합니다.

호스팅된 UI

사용자 풀 페더레이션 엔드포인트 및 호스팅 UI 참조의 모든 엔드포인트에 대한 요청입니다.

퍼블릭 API 작업

AWS 자격 증명을 사용하여 권한을 부여하지 않는 앱에서 Amazon Cognito API로 보내는 요청입니다. 여기에는 InitiateAuthRespondToAuthChallenge, 및 같은 API 작업이 포함됩니다. GetUser 범위 내에 있는 API 작업에는 AWS 자격 증명을 사용한 인증이 AWS WAF 필요하지 않습니다. 인증되지 않았거나 세션 문자열 또는 액세스 토큰으로 권한이 부여되었습니다. 자세한 정보는 Amazon Cognito 사용자 풀 인증 및 미인증 API 작업을 참조하세요.

Count(개수), Allow(허용), Block(차단) 또는 규칙과 일치하는 요청에 대한 응답으로 CAPTCHA를 표시하는 규칙 작업을 사용하여 웹 ACL에서 규칙을 구성할 수 있습니다. 자세한 내용은 AWS WAF 개발자 안내서의 AWS WAF 규칙을 참조하세요. 규칙 작업에 따라 Amazon Cognito가 사용자에게 반환하는 응답을 사용자 지정할 수 있습니다.

Important

오류 응답을 사용자 지정하는 옵션은 API 요청을 수행하는 방식에 따라 달라집니다.

- 호스팅된 UI 요청의 오류 코드와 응답 본문을 사용자 지정할 수 있습니다. 호스팅된 UI에서는 사용자가 해결할 수 있는 CAPTCHA만 제시할 수 있습니다.
- Amazon Cognito 사용자 풀 API로 요청하는 경우 차단 응답을 수신하는 요청의 응답 본문을 사용자 지정할 수 있습니다. 400~499 범위의 사용자 지정 오류 코드를 지정할 수도 있습니다.
- AWS Command Line Interface (AWS CLI) 및 AWS SDK는 차단 또는 CAPTCHA 응답을 생성하는 요청에 `ForbiddenException` 오류를 반환합니다.

웹 ACL을 사용자 풀과 연결

사용자 풀에서 웹 ACL을 사용하려면 AWS Identity and Access Management (IAM) 보안 주체에 다음과 같은 Amazon Cognito 권한이 있어야 합니다. AWS WAF 권한에 대한 자세한 내용은 개발자 AWS WAF 안내서의 API 권한을 참조하십시오.AWS WAF

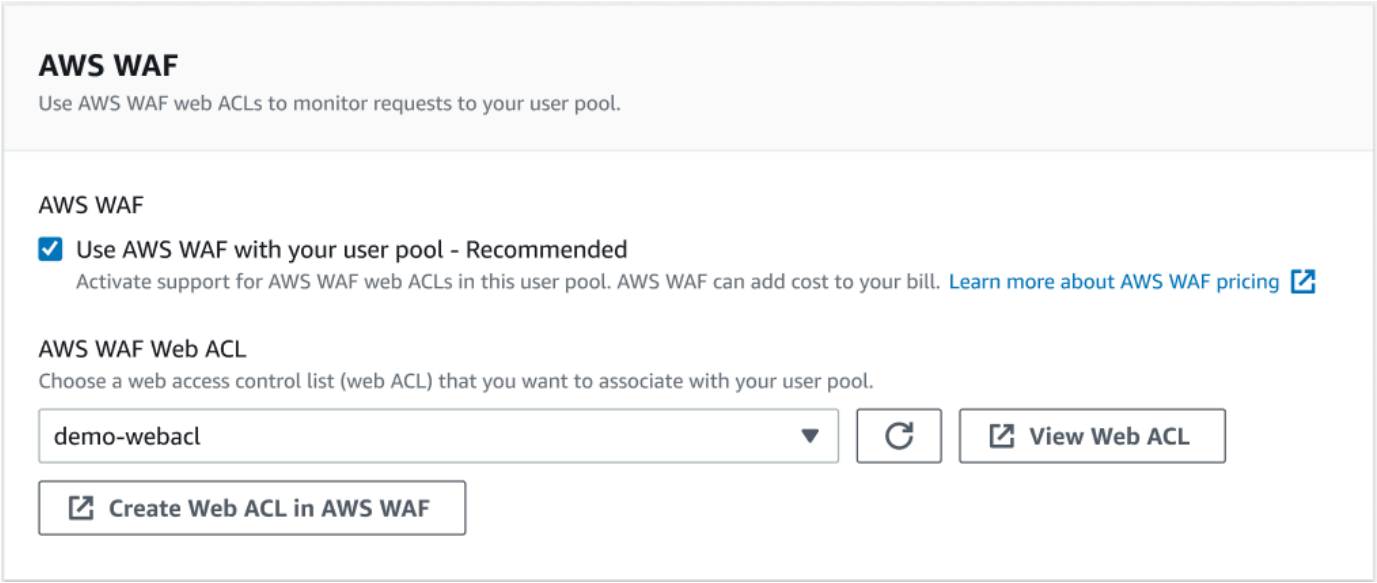
- `cognito-idp:AssociateWebACL`

- `cognito-idp:DisassociateWebACL`
- `cognito-idp:GetWebACLForResource`
- `cognito-idp:ListResourcesForWebACL`

IAM 권한을 부여해야 하지만 나열된 작업은 권한 전용이며 API 작업에 해당하지 않습니다.

사용자 풀을 AWS WAF 활성화하고 웹 ACL을 연결하려면

1. Amazon Cognito 콘솔에 로그인합니다.
2. 탐색 창에서 [사용자 풀(User Pools)]을 선택한 다음 편집할 사용자 풀을 선택합니다.
3. User pool properties(사용자 풀 속성) 탭을 선택합니다.
4. AWS WAF 옆에 있는 Edit(편집)을 선택합니다.
5. 아래에서 AWS WAF사용자 AWS WAF 풀과 함께 사용을 선택합니다.



6. 이미 생성한 AWS WAF 웹 ACL을 선택하거나, 에서 AWS WAF웹 ACL 생성을 선택하여 새 AWS WAF 세션에 웹 ACL을 생성합니다. AWS Management Console
7. 변경 사항 저장를 선택합니다.

프로그래밍 방식으로 웹 ACL을 AWS Command Line Interface 또는 SDK의 사용자 풀과 연결하려면 API의 ACL을 사용하십시오AssociateWeb. AWS WAF Amazon Cognito에는 웹 ACL을 연결하는 별도의 API 작업이 없습니다.

웹 ACL 테스트 및 로깅 AWS WAF

웹 ACL에서 규칙 작업을 카운트로 설정하면 규칙과 일치하는 요청 수에 요청이 AWS WAF 추가됩니다. 사용자 풀로 웹 ACL을 테스트하려면 규칙 작업을 Count(개수)로 설정하고 각 규칙과 일치하는 요청 볼륨을 고려합니다. 예를 들어 Block(차단) 작업으로 설정하려는 규칙이 일반 사용자 트래픽으로 판단되는 많은 수의 요청과 일치하는 경우 규칙을 재구성해야 할 수 있습니다. 자세한 내용은 AWS WAF 개발자 안내서의 AWS WAF 보호 기능 테스트 및 조정을 참조하십시오.

Amazon CloudWatch Logs 로그 그룹, Amazon Simple Storage Service (Amazon S3) 버킷 또는 Amazon 데이터 파이어호스에 요청 헤더를 AWS WAF 기록하도록 구성할 수도 있습니다. `x-amzn-cognito-client-id` 및 `x-amzn-cognito-operation-name`을 통해 사용자 풀 API로 수행한 Amazon Cognito 요청을 식별할 수 있습니다. 호스팅된 UI 요청에는 `x-amzn-cognito-client-id` 헤더만 포함됩니다. 자세한 내용은 AWS WAF 개발자 안내서의 웹 ACL 트래픽 로깅을 참조하세요.

AWS WAF 웹 ACL에는 Amazon Cognito의 고급 보안 기능 요금이 적용되지 않습니다. 의 보안 기능은 Amazon Cognito의 고급 보안 기능을 AWS WAF 보완합니다. 사용자 풀에서 두 기능을 모두 활성화할 수 있습니다. AWS WAF 사용자 풀 요청 검사에 대한 요금은 별도로 청구됩니다. 자세한 내용은 AWS WAF 요금을 참조하세요.

로깅 AWS WAF 요청 데이터에 대해서는 로그를 대상으로 하는 서비스에서 추가 요금을 청구해야 합니다. 자세한 내용은 AWS WAF 개발자 안내서의 웹 ACL 트래픽 정보 로깅 요금을 참조하세요.

사용자 풀 대/소문자 구분

에서 생성하는 Amazon Cognito 사용자 AWS Management Console 풀은 기본적으로 대소문자를 구분하지 않습니다. 사용자 풀이 대/소문자를 구분하지 않는 경우 user@example.com과 User@example.com은 동일한 사용자를 지칭합니다. 사용자 풀의 사용자 이름이 대/소문자를 구분하지 않는 경우 `preferred_username` 및 `email` 속성도 대/소문자를 구분하지 않습니다.

사용자 풀 대/소문자 구분 설정을 고려하여 앱 코드에서 대체 사용자 속성을 기반으로 사용자를 식별합니다. 사용자 이름, 기본 설정 사용자 이름 또는 이메일 주소 속성의 대/소문자는 사용자 프로필마다 다를 수 있으므로 `sub` 속성을 참조하세요. 사용자 풀에 변경 불가능한 사용자 지정 속성을 생성하고 각 새 사용자 프로필의 속성에 고유한 식별자 값을 할당할 수도 있습니다. 사용자를 처음 생성할 때 생성된 변경 불가능한 사용자 지정 속성에 값을 쓸 수 있습니다.

> **Note**
>
> 사용자 풀의 대/소문자 구분 설정에 관계없이 Amazon Cognito에서는 SAML 자격 증명 공급자(IdP)의 페더레이션 사용자가 고유하고 대/소문자를 구분하는 `NameId` 또는 `sub` 클레임을 전

달해야 합니다. 고유 식별자 대소문자 구분 및 IdPs SAML에 대한 자세한 내용은 을 참조하십시오. SP에서 시작한 SAML 로그인 사용

대/소문자를 구분하는 사용자 풀 생성

AWS Command Line Interface (AWS CLI) 및 API 작업 (예:) 을 사용하여 리소스를 생성하는 경우 Boolean `CaseSensitive` 파라미터를 로 CreateUserPool설정해야 합니다. `false` 이 설정은 대/소문자를 구분하지 않는 사용자 풀을 생성합니다. 값을 지정하지 않으면 `CaseSensitive`가 `true`를 기본값으로 사용합니다. 이 기본값은 AWS Management Console에서 생성되는 사용자 풀에 대한 기본 동작과 반대입니다. 2020년 2월 12일 이전에는 플랫폼과 관계없이 사용자 풀이 기본적으로 대/소문자를 구분하도록 설정되었습니다.

AWS Management Console 또는 DescribeUserPoolAPI 작업의 로그인 경험 탭을 사용하여 계정 내 각 사용자 풀의 대소문자 구분 설정을 검토할 수 있습니다.

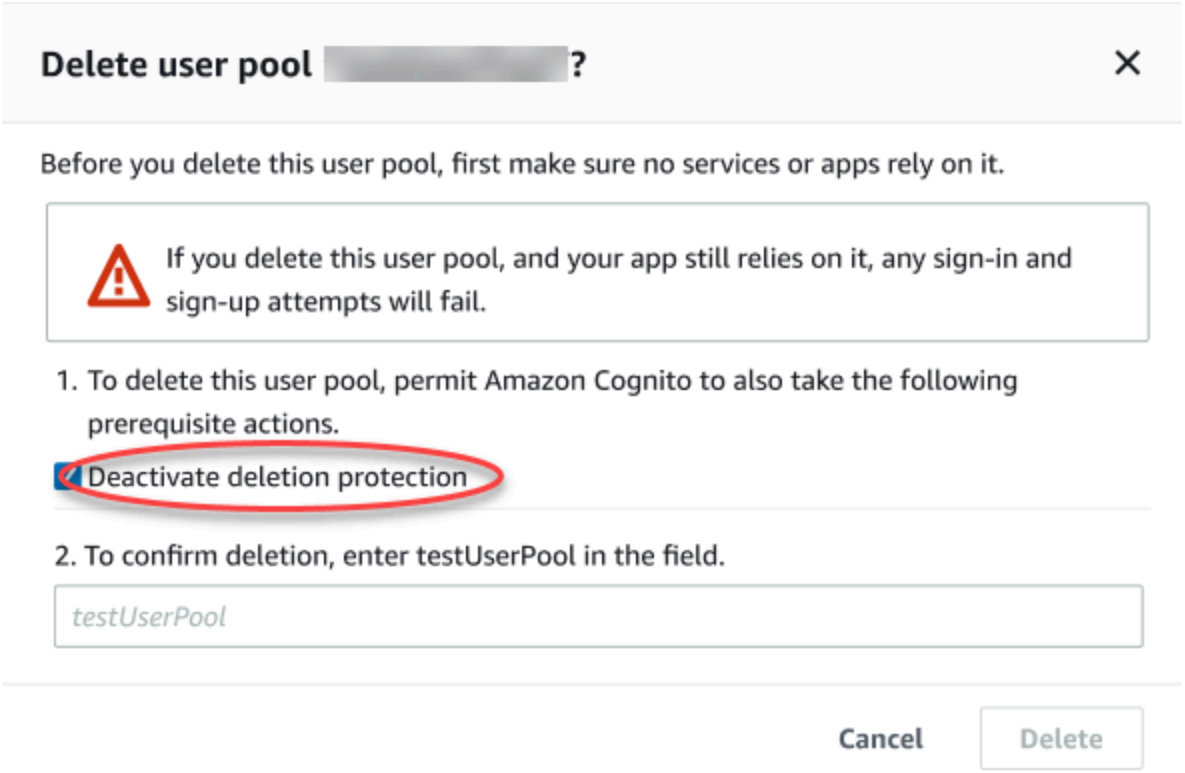
새 사용자 풀로 마이그레이션

사용자 프로필 간에 충돌이 발생할 수 있으므로 Amazon Cognito 사용자 풀을 대/소문자 구분에서 대/소문자 구분 안 함으로 변경할 수 없습니다. 대신 사용자를 새 사용자 풀로 마이그레이션할 수 있습니다. 대/소문자 관련 충돌을 해결하려면 마이그레이션 코드를 작성해야 합니다. 이 코드는 충돌이 감지될 때 고유한 새 사용자를 반환하거나 로그인 시도를 거부해야 합니다. 대/소문자를 구분하지 않는 새 사용자 풀에서 사용자 마이그레이션 Lambda 트리거를 할당합니다. AWS Lambda 함수를 사용하면 대소문자를 구분하지 않는 새 사용자 풀에서 사용자를 생성할 수 있습니다. 사용자가 대/소문자를 구분하지 않는 사용자 풀로 로그인에 실패하면 Lambda 함수가 대/소문자를 구분하는 사용자 풀에서 사용자를 찾아서 복제합니다. 또한 이벤트에서 마이그레이션 사용자 Lambda 트리거를 활성화할 수 있습니다. ForgotPassword Amazon Cognito는 로그인 또는 암호 복구 작업의 사용자 정보 및 이벤트 메타데이터를 Lambda 함수로 전달합니다. 함수가 대/소문자를 구분하지 않는 사용자 풀에서 새 사용자를 생성할 때 이벤트 데이터를 사용하여 사용자 이름과 이메일 주소 간의 충돌을 관리할 수 있습니다. 이러한 충돌은 대/소문자를 구분하지 않는 사용자 풀에서는 고유하지만 대/소문자를 구분하는 사용자 풀에서는 동일한 사용자 이름과 이메일 주소 간에 발생합니다.

Amazon Cognito 사용자 풀 간에 사용자 마이그레이션 Lambda 트리거를 사용하는 방법에 대한 자세한 내용은 블로그의 Amazon Cognito 사용자 풀로 사용자 마이그레이션을 참조하십시오. AWS

사용자 풀 삭제 방지

관리자가 실수로 사용자 풀을 삭제하지 않도록 하려면 삭제 방지를 활성화합니다. 삭제 방지가 활성화된 상태에서는 사용자 풀을 삭제하기 전에 사용자 풀을 삭제할 것임을 확인해야 합니다. AWS Management Console에서 사용자 풀을 삭제할 때 동시에 삭제 방지를 비활성화할 수 있습니다. 다음 이미지와 같이 삭제 방지를 비활성화하라는 메시지를 수락하고 삭제 의사를 확인하면 Amazon Cognito가 사용자 풀을 삭제합니다.



Amazon Cognito API 요청을 사용하여 사용자 풀을 삭제하려면 먼저 UpdateUserPool 요청에서 `DeletionProtection`을 `Inactive`로 변경해야 합니다. 삭제 방지를 비활성화하지 않으면 Amazon Cognito에서 `InvalidParameterException` 오류를 반환합니다. 삭제 방지를 비활성화한 후 DeleteUserPool 요청에서 사용자 풀을 삭제할 수 있습니다.

AWS Management Console에서 새 사용자 풀을 생성하면 기본적으로 Amazon Cognito가 Deletion protection(삭제 방지)을 활성화합니다. `CreateUserPool` API를 이용해 사용자 풀을 생성하면 기본적으로 삭제 방지가 비활성화됩니다. AWS CLI 또는AWS SDK로 만든 사용자 풀에서 이 기능을 사용하려면 `DeletionProtection` 파라미터를 `True`로 설정합니다.

Amazon Cognito 콘솔의 User pool settings(사용자 풀 설정) 탭에 있는 Deletion protection(삭제 방지) 컨테이너에서 삭제 방지 상태를 활성화하거나 비활성화할 수 있습니다.

삭제 방지를 구성하려면

1. Amazon Cognito 콘솔로 이동합니다. AWS 자격 증명을 입력하라는 메시지가 나타날 수 있습니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. User pool settings(사용자 풀 설정) 탭을 선택합니다. Deletion Protection(삭제 방지)을 찾아 Activate(활성화) 또는 Deactivate(비활성화)를 선택합니다.
5. 다음 대화 상자에서 선택을 확인합니다.

사용자 존재 오류 응답 관리

Amazon Cognito는 사용자 풀에서 반환하는 오류 응답을 사용자 지정하도록 지원합니다. 사용자 지정 오류 응답은 사용자 생성, 인증, 암호 복구 및 확인 작업에 사용할 수 있습니다.

사용자 풀 앱 클라이언트의 `PreventUserExistenceErrors` 설정을 사용하여 사용자 존재 여부와 관련한 오류를 사용하거나 사용하지 않도록 설정합니다. Amazon Cognito 사용자 풀 API로 새 앱 클라이언트를 생성하면 기본적으로 `PreventUserExistenceErrors` API가 `LEGACY` 활성화되거나 비활성화됩니다. Amazon Cognito 콘솔에서는 사용자 존재 오류 방지 옵션 (`ENABLED`PreventUserExistenceErrorsfor 설정) 이 기본적으로 선택됩니다. `PreventUserExistenceErrors`구성을 업데이트하려면 다음 중 하나를 수행하십시오.

- UpdateUserPoolClientAPI `LEGACY` 요청에서 `ENABLED` 과 `PreventUserExistenceErrors` 사이의 값을 변경합니다.
- Amazon Cognito 콘솔에서 앱 클라이언트를 편집하고 사용자 존재 오류 방지 상태를 선택 () 과 선택 취소 (`ENABLED`) 사이에서 변경합니다. `LEGACY`

이 속성의 값이 1인 경우`LEGACY`, 사용자가 사용자 풀에 없는 사용자 이름으로 로그인을 시도하면 앱 클라이언트가 `UserNotFoundException` 오류 응답을 반환합니다.

이 속성의 `ENABLED` 값이 1인 경우 앱 클라이언트는 `UserNotFoundException` 오류가 발생하여 사용자 풀에 사용자 계정이 존재하지 않는다고 알리지 않습니다. 의 `PreventUserExistenceErrors` `ENABLED` 구성은 다음과 같은 영향을 미칩니다.

- Amazon Cognito는 API 요청에 대해 비특이적 정보로 응답하며 그렇지 않으면 유효한 사용자의 존재가 드러날 수 있습니다.
- Amazon Cognito 로그인 및 비밀번호 분실 API는 일반적인 인증 실패 응답을 반환합니다. 이 오류 응답은 사용자 이름 또는 암호가 잘못되었음을 나타냅니다.

- Amazon Cognito 계정 확인 및 암호 복구 API는 사용자 연락처 정보를 부분적으로 표시하는 대신 시뮬레이션된 전송 매체로 코드가 전송되었음을 나타내는 응답을 반환합니다.

다음 정보는 로 설정된 PreventUserExistenceErrors 경우의 사용자 풀 작업 동작을 자세히 설명합니다. ENABLED

인증 및 사용자 생성 작업

사용자 이름-암호와 보안 원격 암호 (SRP) 인증 모두에서 오류 응답을 구성할 수 있습니다. 사용자 지정 인증을 통해 반환되는 오류를 사용자 지정할 수도 있습니다. 다음 API는 이러한 인증 작업을 수행합니다.

- AdminInitiateAuth
- AdminRespondToAuthChallenge
- InitiateAuth
- RespondToAuthChallenge

다음 목록은 사용자 인증 작업에서 오류 응답을 사용자 지정하는 방법을 보여줍니다.

사용자 이름 및 암호 인증

ADMIN_USER_PASSWORD_AUTH 및 USER_PASSWORD_AUTH를 사용하여 사용자를 로그인하려면 AdminInitiateAuth 또는 InitiateAuth API 요청에 사용자 이름과 암호를 포함해야 합니다. 사용자 이름 또는 암호가 올바르지 않으면 Amazon Cognito는 일반 NotAuthorizedException 오류를 반환합니다.

Secure Remote Password(SRP) 기반 인증

USER_SRP_AUTH를 사용하려 사용자를 로그인하려면 사용자 이름과 SRP_A 파라미터를 AdminInitiateAuth 또는 InitiateAuth API 요청에 포함해야 합니다. 이에 대한 응답으로 Amazon Cognito는 사용자를 대신하여 SRP_B 반환하고 솔트합니다. 사용자를 찾을 수 없는 경우 Amazon Cognito가 RFC 5054에 설명된 대로 첫 번째 단계에서 시뮬레이션된 응답을 반환합니다. Amazon Cognito가 동일한 사용자 이름 및 사용자 풀 조합에 대해 동일한 솔트와 내부 사용자 ID(Universally Unique Identifier(UUID) 형식)를 반환합니다. RespondToAuthChallenge 요청을 암호 증명과 함께 보내면, Amazon Cognito는 사용자 이름 또는 암호가 올바르지 않을 때 일반 NotAuthorizedException 오류를 반환합니다.

> **Note**
>
> 검증 기반 별칭 속성을 사용하고 있으며 변경 불가능한 사용자 이름의 형식이 UUID로 지정되지 않았다면, 사용자 이름 및 암호 인증을 사용하여 일반 응답을 시뮬레이션할 수 있습니다.

사용자 지정 인증 문제 Lambda 트리거

사용자 지정 인증 문제 Lambda 트리거를 사용하고 오류 응답을 사용하는 경우 `LambdaChallenge`는 `UserNotFound`라는 부울 파라미터를 반환합니다. 이 파라미터는 `DefineAuthChallenge`, `VerifyAuthChallenge` 및 `CreateAuthChallenge` Lambda 트리거의 요청에 전달됩니다. 이 트리거를 사용하여 존재하지 않는 사용자에 대해 사용자 지정 권한 부여 문제를 시뮬레이션할 수 있습니다. 존재하지 않는 사용자에 대해 사전 인증 Lambda 트리거를 호출하면 Amazon Cognito가 `UserNotFound`를 반환합니다.

다음 목록은 사용자 생성 작업에서 오류 응답을 사용자 지정하는 방법을 보여줍니다.

SignUp

사용자 이름을 이미 사용한 `UsernameExistsException` 경우 `SignUp` 작업이 항상 반환됩니다. 앱에서 사용자를 등록할 때 Amazon Cognito가 이메일 주소 및 전화번호에 대한 `UsernameExistsException` 오류를 반환하지 않게 하려면, 확인 기반 별칭 속성을 사용해야 합니다. 별칭에 대한 자세한 내용은 로그인 속성 사용자 지정을 참조하십시오.

Amazon Cognito가 `SignUp` API 요청을 사용하여 사용자 풀에서 사용자를 검색하지 못하게 하는 예시는 가입 시 이메일 주소 및 전화 번호 관련 `UsernameExistsException` 오류 방지에서 확인할 수 있습니다.

가져온 사용자

`PreventUserExistenceErrors`를 사용하는 경우 가져온 사용자를 인증하는 중에 `PasswordResetRequiredException`이 반환되지 않고 대신 사용자 이름 또는 암호가 잘못되었음을 나타내는 일반 `NotAuthorizedException` 오류가 반환됩니다. 자세한 내용은 가져온 사용자에게 암호 재설정 요구를 참조하세요.

사용자 마이그레이션 Lambda 트리거

Amazon Cognito는 Lambda 트리거에서 원래 이벤트 컨텍스트에 빈 응답이 설정된 경우 존재하지 않는 사용자에 대해 시뮬레이션된 응답을 반환합니다. 자세한 내용은 사용자 마이그레이션 Lambda 트리거를 참조하세요.

가입 시 이메일 주소 및 전화 번호 관련 **UsernameExistsException** 오류 방지

다음 예제는 사용자 풀에서 별칭 속성을 구성할 때 중복된 이메일 주소와 전화번호 때문에 SignUp API 요청에 대한 응답으로 UsernameExistsException 오류가 생성되지 않게 하는 방법을 보여줍니다. 이메일 주소나 전화 번호를 별칭 속성으로 사용하여 사용자 풀을 생성해야 합니다. 자세한 내용은 사용자 풀 속성의 로그인 특성 사용자 지정 섹션을 참조하세요.

1. Jie는 새 사용자 이름을 등록하고 이메일 주소(jie@example.com)도 입력합니다. Amazon Cognito가 이 이메일 주소로 확인 코드를 전송합니다.

   AWS CLI 명령 예시

   ```
   aws cognito-idp sign-up --client-id 1234567890abcdef0 --username jie --password
    PASSWORD --user-attributes Name="email",Value="jie@example.com"
   ```

   응답의 예

   ```
   {
       "UserConfirmed": false,
       "UserSub": "<subId>",
       "CodeDeliveryDetails": {
           "AttributeName": "email",
           "Destination": "j****@e*****",
           "DeliveryMedium": "EMAIL"
       }
   }
   ```

2. Jie는 전송된 코드를 입력하여 이메일 주소의 소유권을 확인합니다. 이로써 사용자 등록이 완료됩니다.

   예제 AWS CLI 명령

   ```
   aws cognito-idp confirm-sign-up --client-id 1234567890abcdef0 --username=jie --
   confirmation-code xxxxxx
   ```

3. Shirley는 새 사용자 계정을 등록하고 이메일 주소(`jie@example.com`)를 입력합니다. Amazon Cognito는 `UsernameExistsException` 오류를 반환하지 않고, Jie의 이메일 주소로 확인 코드를 보냅니다.

   예제 AWS CLI 명령

   ```
   aws cognito-idp sign-up --client-id 1234567890abcdef0 --username shirley --password
    PASSWORD --user-attributes Name="email",Value="jie@example.com"
   ```

   응답의 예

   ```
   {
       "UserConfirmed": false,
       "UserSub": "<new subId>",
       "CodeDeliveryDetails": {
           "AttributeName": "email",
           "Destination": "j****@e****",
           "DeliveryMedium": "EMAIL"
       }
   }
   ```

4. 다른 시나리오에서는 Shirley 가 `jie@example.com`에 대한 소유권을 가집니다. 셜리는 Amazon Cognito가 Jie의 이메일 주소로 보낸 코드를 검색하고 계정 확인을 시도합니다.

   예제 AWS CLI 명령

   ```
   aws cognito-idp confirm-sign-up --client-id 1234567890abcdef0 --username=shirley --
   confirmation-code xxxxxx
   ```

   응답의 예

   ```
   An error occurred (AliasExistsException) when calling the ConfirmSignUp operation: An
    account with the email already exists.
   ```

`jie@example.com`이 기존 사용자에게 할당되었지만 Amazon Cognito는 Shirley의 `aws cognito-idp sign-up` 요청에 대해 오류를 반환하지 않습니다. Shirley는 Amazon Cognito가 오류 응답을 반환하기 전에 이메일 주소의 소유권을 입증해야 합니다. 별칭 속성이 있는 사용자 풀에서는 이 동작 때문에, 지정된 이메일 주소나 전화 번호를 가진 사용자의 존재 여부를 공개 `SignUp` API로 확인할 수 없습니다.

이 동작은 다음 예에서처럼 Amazon Cognito가 기존 사용자 이름과 함께 SignUp 요청에 반환하는 응답과는 다릅니다. Shirley는 이 응답을 통해 사용자 이름이 jie인 사용자가 이미 존재함을 알게 되지만, 이 사용자와 관련된 이메일 주소나 전화번호는 알지 못합니다.

CLI 명령 예

```
aws cognito-idp sign-up --client-id 1example23456789 --username jie --password PASSWORD
      --user-attributes Name="email",Value="shirley@example.com"
```

응답의 예

```
An error occurred (UsernameExistsException) when calling the SignUp operation: User
 already exists
```

암호 재설정 작업

Amazon Cognito는 사용자 존재 오류를 방지하면 사용자 암호 재설정 작업에 다음과 같은 응답을 반환합니다.

ForgotPassword

사용자를 찾을 수 없거나, 사용자가 비활성화되었거나, 암호를 복구할 수 있는 확인된 전달 메커니즘이 없는 경우 Amazon Cognito는 사용자에 대해 시뮬레이션된 전송 미디어와 함께 CodeDeliveryDetails를 반환합니다. 시뮬레이션된 전송 미디어는 사용자 풀의 입력 사용자 이름 형식 및 검증 설정에 따라 결정됩니다.

ConfirmForgotPassword

Amazon Cognito는 존재하지 않거나 사용 중지된 사용자에 대해 CodeMismatchException 오류를 반환합니다. ForgotPassword를 사용할 때 코드가 요청되지 않은 경우 Amazon Cognito가 ExpiredCodeException 오류를 반환합니다.

확인 작업

Amazon Cognito는 사용자 존재 오류를 방지하는 경우 사용자 확인 및 검증 작업에 다음과 같은 응답을 반환합니다.

ResendConfirmationCode

Amazon Cognito는 사용 중지되거나 존재하지 않는 사용자에 대해 CodeDeliveryDetails 오류를 반환합니다. Amazon Cognito는 기존 사용자의 이메일 또는 전화 번호로 확인 코드를 보냅니다.

ConfirmSignUp

`ExpiredCodeException`은 코드가 만료된 경우 반환됩니다. Amazon Cognito는 사용자에게 권한이 부여되지 않은 경우 `NotAuthorizedException`을 반환합니다. 서버에서 예상하는 코드와 일치하지 않는 경우 Amazon Cognito는 `CodeMismatchException`을 반환합니다.

Amazon Cognito 자격 증명 풀

Amazon Cognito 자격 증명 풀은 AWS 보안 인증으로 교환할 수 있는 페더레이션 자격 증명의 디렉터리입니다. AWS 자격 증명 풀은 앱 사용자가 로그인했든 아직 식별하지 않았든 관계없이 앱 사용자를 위한 임시 자격 증명을 생성합니다. AWS Identity and Access Management (IAM) 역할 및 정책을 사용하여 사용자에게 부여할 권한 수준을 선택할 수 있습니다. 사용자는 게스트로 시작하고 AWS 서비스에 보관하는 자산을 검색할 수 있습니다. 그런 다음 사용자는 서드 파티 ID 제공업체를 통해 로그인하여 등록된 회원에게 제공되는 자산에 대한 액세스 권한을 잠금 해제할 수 있습니다. 서드 파티 ID 제공업체는 Apple이나 Google 같은 소비자(소셜) OAuth 2.0 공급자, 사용자 지정 SAML 또는 OIDC ID 제공업체 또는 자체 설계한 사용자 지정 인증 체계(개발자 공급자라고도 함)일 수 있습니다.

Amazon Cognito 자격 증명 풀의 기능

서명 요청: AWS 서비스

아마존 심플 스토리지 서비스 (Amazon S3) 와 아마존 DynamoDB와 AWS 서비스 같은 API 요청에 서명하십시오. Amazon Pinpoint 및 Amazon과 같은 서비스를 사용하여 사용자 활동을 분석할 수 있습니다. CloudWatch

리소스 기반 정책으로 요청 필터링

리소스에 대한 사용자 액세스에 대한 세분화된 제어를 실행합니다. 사용자 클레임을 IAM 세션 태그로 변환하고 개별 사용자 하위 집합에 리소스 액세스 권한을 부여하는 IAM 정책을 구축합니다.

게스트 액세스 할당

아직 로그인하지 않은 사용자의 경우 액세스 범위가 좁은 AWS 보안 인증을 생성하도록 자격 증명 풀을 구성합니다. SSO(Single Sign-On) 공급자를 통해 사용자를 인증하여 액세스 권한을 높입니다.

사용자 특성에 따라 IAM 역할 할당

인증된 모든 사용자에게 단일 IAM 역할을 할당하거나 각 사용자의 클레임을 기반으로 역할을 선택합니다.

다양한 ID 제공업체 수락

ID 또는 액세스 토큰, 사용자 풀 토큰, SAML 어설션 또는 소셜 공급업체 OAuth 토큰을 자격 증명으로 교환하십시오. AWS

자체 자격 증명 검증

자체 사용자 검증을 수행하고 개발자 AWS 자격 증명을 사용하여 사용자를 위한 자격 증명을 발급하세요.

인증 및 권한 부여 서비스를 앱에 제공하는 Amazon Cognito 사용자 풀이 이미 있을 수도 있습니다. 사용자 풀을 자격 증명 풀에 대한 ID 제공업체(idP)로 설정할 수 있습니다. 이렇게 하면 사용자가 사용자 풀을 IdPs 통해 인증하고 클레임을 공통 OIDC ID 토큰으로 통합한 다음 해당 토큰을 자격 증명으로 교환할 수 있습니다. AWS 그러면 사용자는 서명된 요청의 보안 인증을 AWS 서비스에 제시할 수 있습니다.

또한 어떤 ID 제공업체의 인증된 클레임도 자격 증명 풀에 직접 제시할 수 있습니다. Amazon Cognito는 SAML, OAuth 및 OIDC 공급자의 사용자 클레임을 단기 자격 증명을 위한 API 요청으로 사용자 지정합니다. AssumeRoleWithWebIdentity

Amazon Cognito 사용자 풀은 SSO 지원 앱에 대한 OIDC ID 제공업체와 같습니다. 자격 증명 풀은 IAM 권한 부여에서 가장 잘 작동하는 리소스 종속성이 있는 앱에 AWS ID 제공업체 역할을 합니다.

Amazon Cognito 자격 증명 풀은 다음 자격 증명 공급자를 지원합니다.

- 퍼블릭 공급자: Amazon으로 로그인을 자격 증명 풀 IdP로 설정, 페이스북을 아이덴티티 풀 IdP로 설정하기, 구글을 아이덴티티 풀 IdP로 설정하기, Apple을 자격 증명 풀 IdP로 사용하여 로그인 설정하기, Twitter.
- Amazon Cognito 사용자 풀
- OIDC 공급자를 자격 증명 풀 IdP로 설정
- SAML 공급자를 자격 증명 풀 IdP로 설정
- 개발자 인증 자격 증명(자격 증명 풀)

Amazon Cognito 자격 증명 풀 리전 가용성에 대한 자세한 내용은 AWS 서비스 리전 가용성을 참조하세요.

Amazon Cognito 자격 증명 풀에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- 자격 증명 풀(페더레이션 자격 증명) 사용
- 자격 증명 풀 개념
- Amazon Cognito 자격 증명 풀의 보안 모범 사례

- 액세스 제어에 속성 사용
- 역할 기반 액세스 제어 사용
- 자격 증명 얻기
- 서비스 액세스 AWS
- 자격 증명 풀 외부 자격 증명 공급자
- 개발자 인증 자격 증명(자격 증명 풀)
- 인증되지 않은 사용자를 인증된 사용자로 전환(자격 증명 풀)

자격 증명 풀(페더레이션 자격 증명) 사용

Amazon Cognito AWS 자격 증명 풀은 게스트 (미인증) 인 사용자와 인증을 받고 토큰을 받은 사용자에게 임시 자격 증명을 제공합니다. 자격 증명 풀은 계정에 관련된 사용자 자격 증명 데이터의 저장소입니다.

콘솔에서 새 자격 증명 풀을 만들려면

1. Amazon Cognito 콘솔에 로그인하고 자격 증명 풀을 선택합니다.
2. 자격 증명 풀 생성을 선택합니다.
3. 자격 증명 풀 신뢰 구성에서 인증된 액세스, 게스트 액세스 또는 둘 다에 대해 자격 증명 풀을 설정하도록 선택합니다.

    - 인증된 액세스를 선택한 경우 자격 증명 풀에서 인증된 자격 증명의 소스로 설정하려는 자격 증명 유형을 하나 이상 선택합니다. 사용자 지정 개발자 공급자를 구성하는 경우 자격 증명 풀을 생성한 후에는 이를 수정하거나 삭제할 수 없습니다.
4. 권한 구성에서 자격 증명 풀의 인증된 사용자 또는 게스트 사용자의 기본 IAM 역할을 선택합니다.

    a. Amazon Cognito가 기본 권한 및 자격 증명 풀과의 신뢰 관계가 있는 새 역할을 생성하도록 하려면 새 IAM 역할 생성을 선택합니다. 새 역할을 식별하는 IAM 역할 이름을 입력합니다 (예: `myidentitypool_authenticatedrole`). Amazon Cognito가 새 IAM 역할에 할당할 권한을 검토하려면 정책 문서 보기를 선택합니다.

    b. 사용하려는 역할이 이미 AWS 계정 있는 경우 기존 IAM 역할을 사용하도록 선택할 수 있습니다. `cognito-identity.amazonaws.com`을 포함하도록 IAM 역할 신뢰 정책을 구성해야 합니다. 특정 자격 증명 풀의 인증된 사용자로부터 요청이 시작되었다는 증거가 제시되는 경우에만 Amazon Cognito가 역할을 맡을 수 있도록 역할 신뢰 정책을 구성합니다. 자세한 정보는 역할 트러스트 및 권한을 참조하세요.

5. Connect ID 제공자에 ID 풀 신뢰 구성에서 선택한 ID 제공자 (IdPs) 의 세부 정보를 입력합니다. OAuth 앱 클라이언트 정보를 제공하거나, Amazon Cognito 사용자 풀을 선택하거나, IAM IdP를 선택하거나, 개발자 공급자의 사용자 지정 식별자를 입력하라는 메시지가 표시될 수 있습니다.

   a. 각 IdP의 역할 설정을 선택합니다. 인증된 역할을 구성할 때 설정한 기본 역할을 이 IdP의 사용자에게 할당하거나 규칙을 사용하여 역할 선택을 선택할 수 있습니다. Amazon Cognito 사용자 풀 IdP를 사용하면 토큰에서 preferred_role을 포함한 역할을 선택할 수도 있습니다. `cognito:preferred_role` 클레임에 대한 자세한 내용은 그룹에 우선 순위 값 할당을 참조하세요.

      i. 규칙을 사용하여 역할 선택을 선택한 경우 사용자 인증의 소스 클레임, 클레임 비교 기준으로 사용할 운영자, 이 역할 선택과 일치하도록 하는 값, 역할 할당이 일치할 때 할당할 역할을 입력합니다. 다른 조건에 따라 추가 규칙을 생성하려면 다른 항목 추가를 선택합니다.

      ii. 역할 해결을 선택합니다. 사용자의 클레임이 규칙과 일치하지 않는 경우 보안 인증을 거부하거나 인증된 역할의 보안 인증을 발급할 수 있습니다.

   b. 각 IdP별로 액세스 제어를 위한 속성을 구성합니다. 액세스 제어를 위한 속성은 사용자 클레임을 Amazon Cognito가 임시 세션에 적용하는 보안 주체 태그에 매핑합니다. 세션에 적용하는 태그를 기반으로 사용자 액세스를 필터링하는 IAM 정책을 구축할 수 있습니다.

      i. 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.

      ii. `sub` 및 `aud` 클레임 기반 보안 주체 태그를 적용하려면 기본 매핑 사용을 선택합니다.

      iii. 보안 주체 태그에 대한 속성의 자체 사용자 지정 스키마를 생성하려면 사용자 지정 매핑 사용을 선택합니다. 그런 다음 태그에 표시하려는 각 클레임에서 소싱하려는 태그 키를 입력합니다.

6. 속성 구성에서 자격 증명 풀 이름 아래에 이름을 입력합니다.

7. 기본(Classic) 인증에서 기본 흐름 활성화 여부를 선택합니다. 기본 흐름이 활성화되면 선택한 역할을 IdPs 우회하여 직접 AssumeRoleWithWebIdentity호출할 수 있습니다. 자세한 정보는 자격 증명 풀(페더레이션 자격 증명) 인증 흐름을 참조하세요.

8. 자격 증명 풀에 태그를 적용하려면 태그에서 태그 추가를 선택합니다.

9. 검토 및 생성에서 새 자격 증명 풀에 대한 선택 사항을 확인합니다. 편집을 선택하여 마법사로 돌아가서 설정을 변경합니다. 완료하면 자격 증명 풀 생성을 선택합니다.

사용자 IAM 역할

IAM 역할은 사용자가 AWS 리소스에 액세스할 수 있는 권한 (예:) 을 정의합니다. Amazon Cognito Sync 생성된 역할을 애플리케이션 사용자가 수임합니다. 인증된 사용자와 인증되지 않은 사용자에게 다른 역할을 지정할 수 있습니다. IAM 역할에 대한 자세한 내용은 IAM 역할 섹션을 참조하세요.

인증된 자격 증명 및 인증되지 않은 자격 증명

Amazon Cognito 자격 증명 풀은 인증된 자격 증명과 인증되지 않은 자격 증명을 모두 지원합니다. 인증된 자격 증명은 지원되는 자격 증명 공급자가 인증한 사용자를 위한 것이고, 인증되지 않은 자격 증명은 대개 게스트 사용자를 위한 것입니다.

- 퍼블릭 로그인 공급자를 통해 인증된 자격 증명을 구성하려면 자격 증명 풀 외부 자격 증명 공급자 섹션을 참조하세요.
- 자체 백엔드 인증 프로세스를 구성하려면 개발자 인증 자격 증명(자격 증명 풀) 섹션을 참조하세요.

게스트 액세스 활성화 또는 비활성화

Amazon Cognito 자격 증명 풀 게스트 액세스 (비인증 자격 증명) 는 자격 증명 공급자를 통해 인증하지 않은 사용자에게 고유한 식별자와 AWS 자격 증명을 제공합니다. 애플리케이션에서 로그인하지 않은 사용자를 허용하는 경우 인증하지 않은 자격 증명의 액세스를 활성화할 수 있습니다. 자세한 내용은 Amazon Cognito 자격 증명 풀 시작하기 섹션을 참조하세요.

자격 증명 풀에서 게스트 액세스를 업데이트하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. 게스트 액세스를 찾습니다. 현재 게스트 액세스를 지원하지 않는 자격 증명 풀에서는 상태가 비활성입니다.

    a. 게스트 액세스가 활성이고 이를 비활성화하려면 비활성화를 선택합니다.

    b. 게스트 액세스가 비활성이고 이를 활성화하려면 편집을 선택합니다.

        - 자격 증명 풀의 게스트 사용자의 기본 IAM 역할을 선택합니다.

            A. Amazon Cognito가 기본 권한 및 자격 증명 풀과의 신뢰 관계가 있는 새 역할을 생성하도록 하려면 새 IAM 역할 생성을 선택합니다. 새 역할을 식별하는 IAM 역할 이

름을 입력합니다(예: `myidentitypool_authenticatedrole`). Amazon Cognito가 새 IAM 역할에 할당할 권한을 검토하려면 정책 문서 보기를 선택합니다.

B. 사용하려는 역할이 이미 있는 경우 기존 IAM 역할을 사용하도록 선택할 수 있습니다 AWS 계정 . `cognito-identity.amazonaws.com`을 포함하도록 IAM 역할 신뢰 정책을 구성해야 합니다. 특정 자격 증명 풀의 인증된 사용자로부터 요청이 시작되었다는 증거가 제시되는 경우에만 Amazon Cognito가 역할을 맡을 수 있도록 역할 신뢰 정책을 구성합니다. 자세한 정보는 역할 트러스트 및 권한을 참조하세요.

C. 변경 사항 저장(Save changes)을 선택합니다.

D. 게스트 액세스를 활성화하려면 사용자 액세스 탭에서 활성화를 선택합니다.

자격 증명 유형과 연관된 역할 변경

자격 증명 풀의 모든 자격 증명은 인증 또는 미인증입니다. 인증 자격 증명은 퍼블릭 로그인 공급자(Amazon Cognito 사용자 풀, Login with Amazon, Sign in with Apple, Facebook, Google, SAML 또는 OpenID Connect 공급자) 또는 개발자 공급자(자체 백엔드 인증 프로세스)에 의해 인증된 사용자에 속합니다. 인증되지 않은 자격 증명은 대개 게스트 사용자를 위한 것입니다.

각 자격 증명 유형에 대해 할당된 역할이 있습니다. 이 역할에는 해당 역할이 액세스할 수 있는 대상을 지정하는 정책이 첨부되어 AWS 서비스 있습니다. Amazon Cognito가 요청을 받으면 이 서비스가 자격 증명 유형을 결정하고, 해당 자격 증명 유형에 할당된 역할을 결정하고, 해당 역할에 첨부된 정책을 사용하여 대응합니다. 정책을 수정하거나 ID 유형에 다른 역할을 할당하여 액세스할 수 있는 ID 유형을 제어할 수 있습니다. AWS 서비스 자격 증명 풀에서 역할과 연관된 정책을 보거나 수정하려면 AWS IAM 콘솔을 참조하세요.

자격 증명 풀 기본 인증 역할 또는 인증되지 않은 역할을 변경하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. 게스트 액세스 또는 인증된 액세스를 찾습니다. 현재 해당 액세스 유형에 대해 구성되지 않은 자격 증명 풀에서는 상태가 비활성입니다. [편집(Edit)]을 선택합니다.
4. 자격 증명 풀의 게스트 또는 인증된 사용자의 기본 IAM 역할을 선택합니다.

   a. Amazon Cognito가 기본 권한 및 자격 증명 풀과의 신뢰 관계가 있는 새 역할을 생성하도록 하려면 새 IAM 역할 생성을 선택합니다. 새 역할을 식별하는 IAM 역할 이름을 입력합니다(예: `myidentitypool_authenticatedrole`). Amazon Cognito가 새 IAM 역할에 할당할 권한을 검토하려면 정책 문서 보기를 선택합니다.

b. 사용하려는 역할이 이미 AWS 계정 있는 경우 기존 IAM 역할을 사용하도록 선택할 수 있습니다. `cognito-identity.amazonaws.com`을 포함하도록 IAM 역할 신뢰 정책을 구성해야 합니다. 특정 자격 증명 풀의 인증된 사용자로부터 요청이 시작되었다는 증거가 제시되는 경우에만 Amazon Cognito가 역할을 맡을 수 있도록 역할 신뢰 정책을 구성합니다. 자세한 정보는 역할 트러스트 및 권한을 참조하세요.

5. 변경 사항 저장(Save changes)을 선택합니다.

ID 제공업체 편집

사용자가 소비자 ID 제공업체(예: Amazon Cognito 사용자 풀, Login with Amazon, Sign in with Apple, Facebook, Google)를 사용하여 인증하는 것을 허용하는 경우 Amazon Cognito 자격 증명 풀(페더레이션 자격 증명) 콘솔에서 애플리케이션 식별자를 지정할 수 있습니다. 이 퍼블릭 자격 증명 공급자는 애플리케이션 ID(퍼블릭 로그인 공급자가 제공함)와 자격 증명 풀을 연결합니다.

이 페이지에서 각 공급자에 대한 인증 규칙을 구성할 수도 있습니다. 각 공급자에 대해 최대 25개의 규칙이 허용됩니다. 규칙은 저장한 순서대로 각 공급자에 적용됩니다. 자세한 내용은 역할 기반 액세스 제어 사용 섹션을 참조하세요.

> **Warning**
>
> 자격 증명 풀에서 연결된 IdP 애플리케이션 ID를 변경하면 기존 사용자가 해당 자격 증명 풀을 사용하여 인증할 수 없게 됩니다. 자세한 정보는 자격 증명 풀 외부 자격 증명 공급자을 참조하세요.

자격 증명 풀 ID 제공업체(idP)를 업데이트하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. ID 제공업체를 찾습니다. 편집할 ID 제공업체를 선택합니다. 새 IdP를 추가하려면 ID 제공업체 추가를 선택합니다.

   - ID 제공업체 추가를 선택한 경우 추가하려는 자격 증명 유형 중 하나를 선택합니다.

4. 애플리케이션 ID를 변경하려면 ID 제공업체 정보에서 편집을 선택합니다.
5. Amazon Cognito가 이 공급자를 통해 인증한 사용자에게 보안 인증을 발급할 때 요청하는 역할을 변경하려면 역할 설정에서 편집을 선택합니다.

- 인증된 역할을 구성할 때 설정한 기본 역할을 해당 IdP의 사용자에게 할당하거나 규칙을 사용하여 역할 선택을 선택할 수 있습니다. Amazon Cognito 사용자 풀 IdP를 사용하면 토큰에서 preferred_role을 포함한 역할을 선택할 수도 있습니다. `cognito:preferred_role` 클레임에 대한 자세한 내용은 그룹에 우선 순위 값 할당을 참조하세요.

  i. 규칙을 사용하여 역할 선택을 선택한 경우 사용자 인증의 소스 클레임, 클레임 비교 기준으로 사용할 운영자, 이 역할 선택과 일치하도록 하는 값, 역할 할당이 일치할 때 할당할 역할을 입력합니다. 다른 조건에 따라 추가 규칙을 생성하려면 다른 항목 추가를 선택합니다.

  ii. 역할 해결을 선택합니다. 사용자의 클레임이 규칙과 일치하지 않는 경우 보안 인증을 거부하거나 인증된 역할의 보안 인증을 발급할 수 있습니다.

6. Amazon Cognito가 이 공급자를 통해 인증한 사용자에게 보안 인증을 발급할 때 할당하는 보안 주체 태그를 변경하려면 액세스 제어를 위한 속성에서 편집을 선택합니다.

   a. 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.

   b. sub 및 aud 클레임 기반 보안 주체 태그를 적용하려면 기본 매핑 사용을 선택합니다.

   c. 보안 주체 태그에 대한 속성의 자체 사용자 지정 스키마를 생성하려면 사용자 지정 매핑 사용을 선택합니다. 그런 다음 태그에 표시하려는 각 클레임에서 소싱하려는 태그 키를 입력합니다.

7. 변경 사항 저장(Save changes)을 선택합니다.

자격 증명 풀 삭제

자격 증명 풀 삭제는 실행 취소할 수 없습니다. 자격 증명 풀을 삭제하면 해당 자격 증명 풀을 사용하는 모든 앱과 사용자의 작동이 중지됩니다.

자격 증명 풀을 삭제하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 삭제할 자격 증명 풀 옆의 라디오 버튼을 선택합니다.
2. 삭제를 선택합니다.
3. 자격 증명 풀의 이름을 입력하거나 붙여넣고 삭제를 선택합니다.

> **Warning**
>
> 삭제(Delete) 버튼을 선택하면 자격 증명 풀과 여기에 포함된 모든 사용자 데이터를 영구적으로 삭제합니다. 자격 증명 풀을 삭제할 경우 자격 증명 풀을 사용하는 애플리케이션과 기타 서비스의 작동이 중지됩니다.

자격 증명 풀에서 자격 증명 삭제

자격 증명 풀에서 자격 증명을 삭제하면 Amazon Cognito가 해당 페더레이션 사용자에 대해 저장한 식별 정보가 제거됩니다. 사용자가 보안 인증을 다시 요청하면 자격 증명 풀이 여전히 사용자의 ID 제공 업체를 신뢰하는 경우 새 자격 증명 ID를 받게 됩니다. 이 작업은 실행 취소할 수 없습니다.

자격 증명을 삭제하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 자격 증명 브라우저 탭을 선택합니다.
3. 삭제할 자격 증명 옆의 확인란을 선택하고 삭제를 선택합니다. 자격 증명을 정말로 삭제할지 다시 묻는 메시지가 나오면 확인 후 삭제를 선택합니다.

Amazon Cognito Sync를 자격 증명 풀과 함께 사용

Amazon Cognito Sync는 AWS 서비스 디바이스 간에 애플리케이션 관련 사용자 데이터를 동기화할 수 있게 해주는 클라이언트 라이브러리입니다. Amazon Cognito Sync를 사용하면 자체 백엔드를 사용하지 않고 모바일 디바이스 및 웹 간에 사용자 프로필 데이터를 동기화할 수 있습니다. 클라이언트 라이브러리는 디바이스 연결 상태와 관계없이 앱에서 데이터를 읽고 쓸 수 있도록 로컬로 데이터를 캐싱합니다. 디바이스가 온라인 상태일 때 데이터를 동기화할 수 있습니다. 푸시 동기화를 설정한 경우 업데이트가 있음을 다른 디바이스에 즉시 알릴 수 있습니다.

데이터 집합 관리

애플리케이션에서 Amazon Cognito Sync 기능을 구현한 경우 Amazon Cognito 자격 증명 풀 콘솔에서는 개별 자격 증명에 대한 데이터 집합과 레코드를 수동으로 생성하고 삭제할 수 있습니다. Amazon Cognito 자격 증명 풀 콘솔에서 자격 증명의 데이터 집합 또는 레코드에 대한 변경 사항은 사용자가 콘솔에서 동기화(Synchronize)를 선택할 때까지 저장되지 않습니다. 자격 증명이 동기화(Synchronize)를 호출할 때까지 최종 사용자에게 변경 사항이 표시되지 않습니다. 개별 자격 증명에 대해 다른 장치에서 동기화되고 있는 데이터는 특정 자격 증명에 대한 목록 데이터 집합 페이지를 새로 고칠 경우 표시됩니다.

자격 증명에 대한 데이터 집합 생성

Amazon Cognito Sync는 데이터 세트를 하나의 자격 증명과 연결합니다. 자격 증명이 나타내는 사용자에 대한 식별 정보로 데이터 세트를 채운 다음 해당 정보를 사용자의 모든 디바이스에 동기화할 수 있습니다.

자격 증명에 데이터 세트와 데이터 세트 레코드를 추가하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 자격 증명 브라우저 탭을 선택합니다.
3. 편집할 자격 증명을 선택합니다.
4. 데이터 세트에서 데이터 세트 생성을 선택합니다.
5. 데이터 세트 이름을 입력하고 데이터 세트 생성을 선택합니다.
6. 데이터 세트에 레코드를 추가하려면 자격 증명 세부 정보에서 데이터 세트를 선택합니다. 레코드에서 레코드 생성을 선택합니다.
7. 레코드의 키와 값을 입력합니다. 확인을 선택합니다. 레코드를 더 추가하려면 이 단계를 반복합니다.

자격 증명과 연결된 데이터 집합 삭제

자격 증명에서 데이터 세트와 해당 레코드를 삭제하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 자격 증명 브라우저 탭을 선택합니다.
3. 삭제할 데이터 세트가 포함된 자격 증명을 선택합니다.
4. 데이터 세트에서 삭제할 데이터 세트 옆에 있는 라디오 버튼을 선택합니다.
5. 삭제를 선택합니다. 선택 사항을 검토하고 삭제를 다시 선택합니다.

데이터 대량 게시

Amazon Cognito Sync 스토어에 이미 저장되어 있는 데이터를 Amazon Kinesis 스트림으로 내보내는 데 대량 게시를 사용할 수 있습니다. 모든 스트림을 대량 게시하는 방법에 대한 자세한 내용은 Amazon Cognito 스트림 섹션을 참조하세요.

푸시 동기화 활성화

Amazon Cognito는 자격 증명과 디바이스 간의 연결 관계를 자동으로 추적합니다. 푸시 동기화 기능을 사용하면 자격 증명 데이터가 변경될 경우 지정한 자격 증명의 모든 인스턴스가 통지되도록 할 수 있습니다. 푸시 동기화는 자격 증명의 데이터 세트가 변경될 때마다 해당 자격 증명과 연결된 모든 디바이스가 변경 사항에 대해 알리는 자동 푸시 알림을 받도록 합니다.

Amazon Cognito 콘솔에서 푸시 동기화를 활성화할 수 있습니다.

푸시 동기화를 활성화하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 자격 증명 풀 속성 탭을 선택합니다.
3. 푸시 동기화에서 편집을 선택합니다.
4. 자격 증명 풀과 푸시 동기화 활성화를 선택합니다.
5. 현재 AWS 리전에 생성한 Amazon Simple Notification Service(SNS) 플랫폼 애플리케이션 중 하나를 선택합니다. Amazon Cognito는 플랫폼 애플리케이션에 푸시 알림을 게시합니다. 플랫폼 애플리케이션 생성을 선택하여 Amazon SNS 콘솔로 이동하고 새 플랫폼 애플리케이션을 생성합니다.
6. Amazon Cognito는 플랫폼 애플리케이션에 게시하기 위해 AWS 계정의 IAM 역할을 맡습니다. Amazon Cognito가 기본 권한 및 자격 증명 풀과의 신뢰 관계가 있는 새 역할을 생성하도록 하려면 새 IAM 역할 생성을 선택합니다. 새 역할을 식별하는 IAM 역할 이름을 입력합니다(예: `myidentitypool_authenticatedrole`). Amazon Cognito가 새 IAM 역할에 할당할 권한을 검토하려면 정책 문서 보기를 선택합니다.
7. 사용하려는 역할이 이미 AWS 계정 있는 경우 기존 IAM 역할을 사용하도록 선택할 수 있습니다. `cognito-identity.amazonaws.com`을 포함하도록 IAM 역할 신뢰 정책을 구성해야 합니다. 특정 자격 증명 풀의 인증된 사용자로부터 요청이 시작되었다는 증거가 제시되는 경우에만 Amazon Cognito가 역할을 맡을 수 있도록 역할 신뢰 정책을 구성합니다. 자세한 정보는 역할 트러스트 및 권한을 참조하세요.
8. 변경 사항 저장(Save changes)을 선택합니다.

Amazon Cognito 스트림 설정

Amazon Cognito 스트림은 개발자에게 Amazon Cognito Sync에 저장된 데이터에 대한 제어와 인사이트를 제공합니다. 개발자는 이제 이벤트를 데이터로 수신하도록 Kinesis 스트림을 구성할 수 있습니다. Amazon Cognito는 각 데이터 집합 변경 사항을 사용자가 소유하는 Kinesis 스트림으로 실시간으

로 푸시합니다. Amazon Cognito 콘솔에서 Amazon Cognito 스트림을 설정하는 방법에 대한 지침은 Amazon Cognito 스트림 섹션을 참조하세요.

Amazon Cognito 이벤트 설정

Amazon Cognito 이벤트를 사용하면 Amazon Cognito Sync에서 중요한 이벤트에 대한 응답으로 AWS Lambda 함수를 실행할 수 있습니다. Amazon Cognito Sync는 데이터 집합이 동기화될 경우 동기화 트리거 이벤트를 유발합니다. 사용자가 데이터를 업데이트할 때 동기화 트리거 이벤트를 사용하여 작업을 수행할 수 있습니다. 콘솔에서 Amazon Cognito 이벤트를 설정하는 방법에 대한 자세한 내용은 Amazon Cognito 이벤트 섹션을 참조하세요.

에 대해 AWS Lambda자세히 알아보려면 을 참조하십시오. AWS Lambda

자격 증명 풀 개념

Amazon Cognito 자격 증명 풀을 사용하여 사용자를 위한 고유의 자격 증명을 생성하고 자격 증명 공급자를 통해 해당 자격 증명을 인증할 수 있습니다. ID가 있으면 권한이 제한된 임시 자격 증명을 얻어 다른 AWS 자격 증명에 액세스할 수 있습니다. AWS 서비스 Amazon Cognito 자격 증명 풀은 퍼블릭 자격 증명 공급자(Amazon, Apple, Facebook 및 Google)와 인증되지 않은 자격 증명을 지원합니다. 또한 자체의 백엔드 인증 프로세스를 통해 사용자를 등록하고 인증할 수 있는 개발자 인증 자격 증명도 지원합니다.

Amazon Cognito 자격 증명 풀 리전 가용성에 대한 자세한 내용은 AWS 서비스 리전 가용성을 참조하세요. Amazon Cognito 자격 증명 풀 개념에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- 자격 증명 풀(페더레이션 자격 증명) 인증 흐름
- IAM 역할
- 역할 트러스트 및 권한

자격 증명 풀(페더레이션 자격 증명) 인증 흐름

Amazon Cognito를 사용하면 디바이스와 플랫폼에서 일관성이 유지되는 최종 사용자의 고유한 식별자를 생성할 수 있습니다. 또한 Amazon Cognito는 리소스에 액세스할 수 있도록 애플리케이션에 제한된 권한의 임시 자격 증명을 제공합니다. AWS 이 페이지에서는 Amazon Cognito의 기본 인증 방법을 다루며, 자격 증명 풀 내에서 자격 증명의 수명 주기에 대해 설명합니다.

외부 공급자 인증 흐름

Amazon Cognito로 인증하는 사용자는 자격 증명을 부트스트랩하기 위해 다단계 프로세스를 진행합니다. Amazon Cognito는 퍼블릭 공급자로부터 인증을 받기 위한, 향상된 흐름과 기본 흐름의 두 가지 흐름을 제공합니다.

이러한 흐름 중 하나를 완료하면 역할의 액세스 정책에 정의된 AWS 서비스 대로 다른 흐름에도 액세스할 수 있습니다. 기본적으로 Amazon Cognito 콘솔은 Amazon Cognito Sync 스토어 및 Amazon Mobile Analytics에 액세스할 수 있는 역할을 생성합니다. 추가 액세스 권한을 부여하는 방법에 대한 자세한 내용은 IAM 역할을 참조하세요.

자격 증명 풀은 공급자의 다음 아티팩트를 수락합니다.

공급자	인증 아티팩트
Amazon Cognito 사용자 풀	ID 토큰
오픈ID 커넥트 (OIDC)	ID 토큰
SAML 2.0	SAML 어설션
소셜 프로바이더	액세스 토큰

향상된(간소화된) 인증 흐름

향상된 인증 흐름을 사용하는 경우 앱은 먼저 요청에서 승인된 Amazon Cognito 사용자 풀 또는 타사 자격 증명 공급자의 인증 증명을 제공합니다. GetId

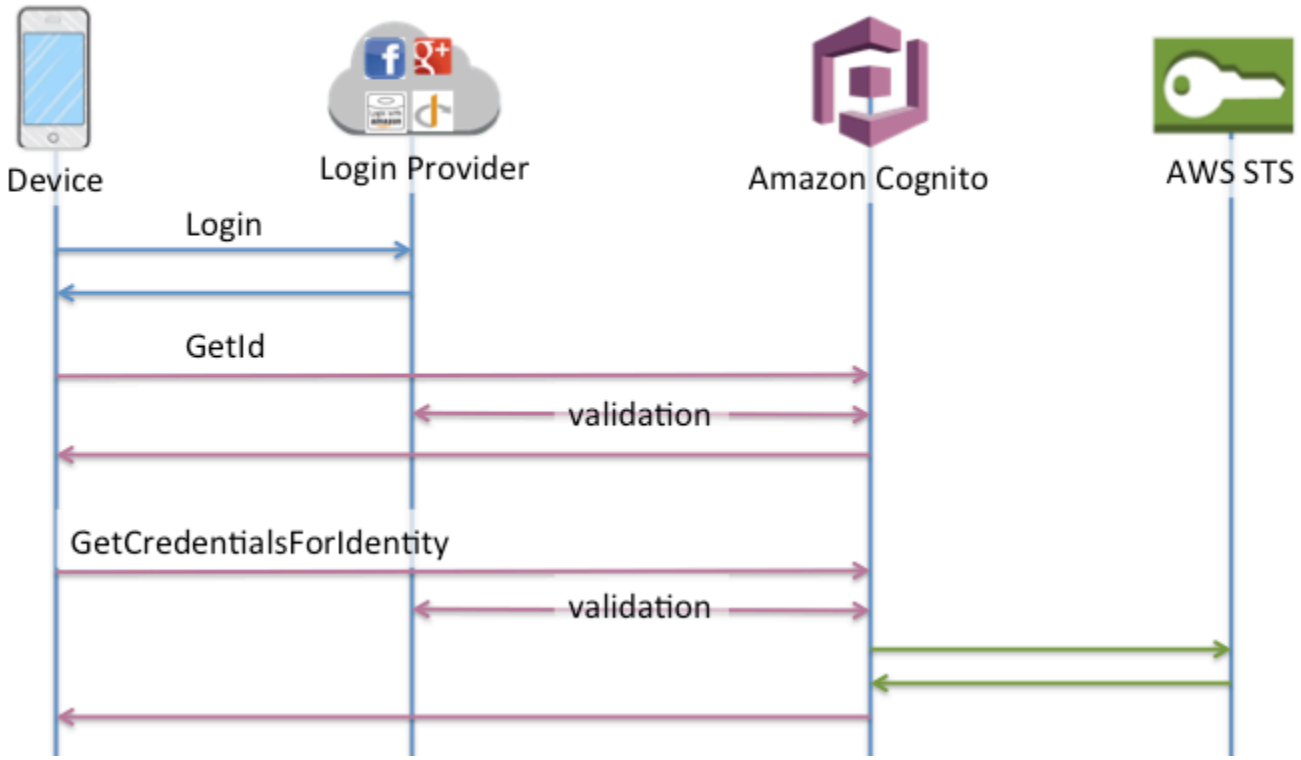
1. 애플리케이션은 GetID 요청에서 승인된 Amazon Cognito 사용자 풀 또는 타사 자격 증명 공급자로부터 받은 인증 증명 (JSON 웹 토큰 또는 SAML 어설션) 을 제공합니다.
2. 자격 증명 풀은 자격 증명 ID를 반환합니다.
3. 애플리케이션은 GetCredentialsForIdentity요청에서 ID ID를 동일한 인증 증명과 결합합니다.
4. 자격 증명 풀은 AWS 자격 증명을 반환합니다.
5. 애플리케이션은 임시 자격 증명으로 AWS API 요청에 서명합니다.

향상된 인증은 자격 증명 풀 구성에서 IAM 역할 선택 및 자격 증명 검색의 로직을 관리합니다. 기본 역할을 선택하고, 속성 기반 액세스 제어 (ABAC) 또는 역할 기반 액세스 제어 (RBAC) 원칙을 역할 선택

에 적용하도록 자격 증명 풀을 구성할 수 있습니다. 향상된 AWS 인증의 자격 증명은 1시간 동안 유효합니다.

향상된 인증의 작업 순서

1. `GetId`
2. `GetCredentialsForIdentity`



기본(클래식) 인증 흐름

기본 인증 흐름을 사용하는 경우

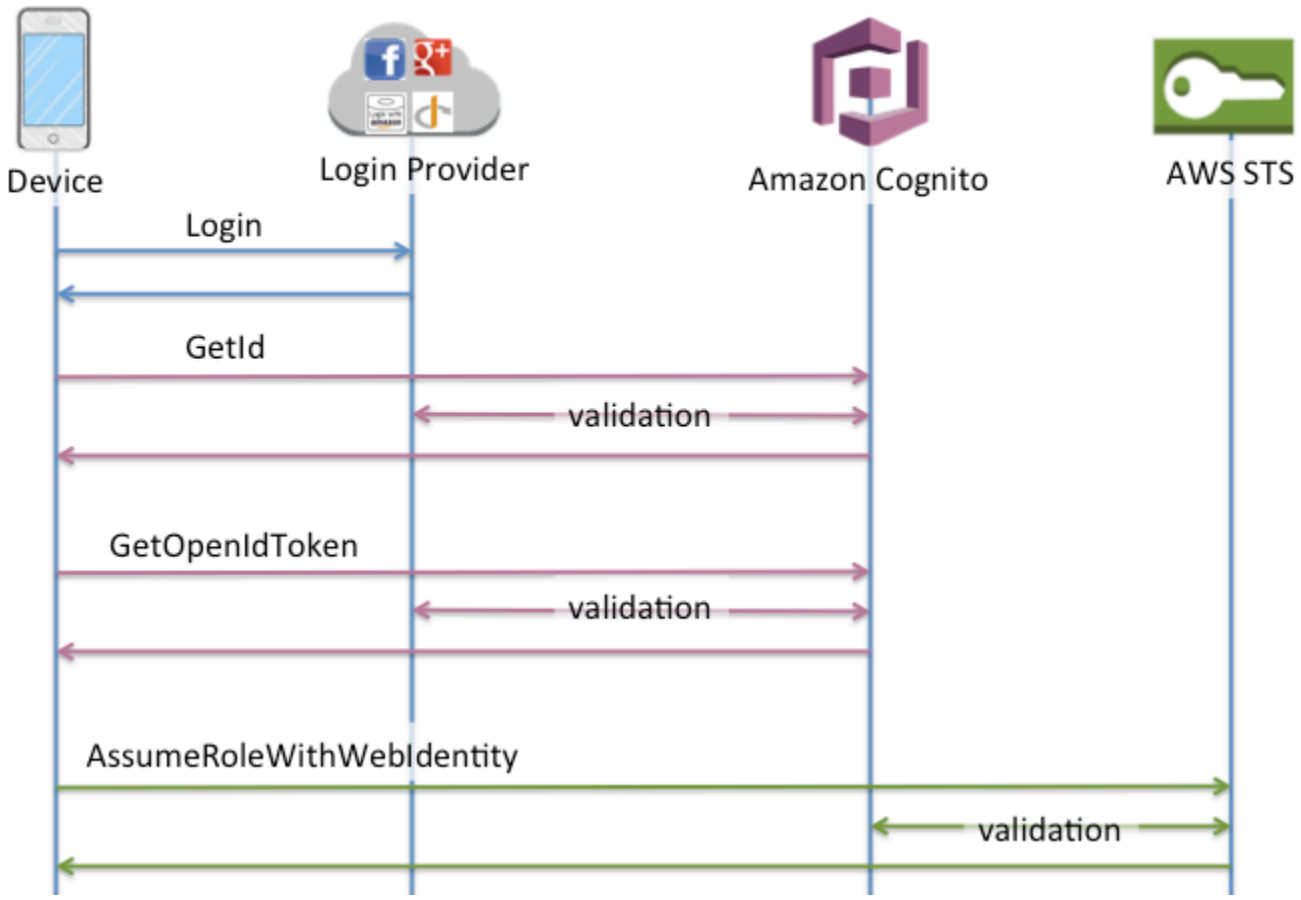
1. 애플리케이션은 GetID 요청에서 승인된 Amazon Cognito 사용자 풀 또는 타사 자격 증명 공급자로부터 받은 인증 증명 (JSON 웹 토큰 또는 SAML 어설션) 을 제공합니다.
2. 자격 증명 풀은 자격 증명 ID를 반환합니다.
3. 애플리케이션은 GetOpenIdToken요청에서 ID ID를 동일한 인증 증명과 결합합니다.
4. `GetOpenIdToken`자격 증명 풀에서 발급한 새 OAuth 2.0 토큰을 반환합니다.
5. 애플리케이션은 요청에서 새 토큰을 제공합니다. AssumeRoleWithWebIdentity
6. AWS Security Token Service AWS STS) 는 AWS 자격 증명을 반환합니다.

7. 애플리케이션은 임시 자격 증명으로 AWS API 요청에 서명합니다.

기본 워크플로를 통해 사용자에게 배포하는 자격 증명을 보다 세밀하게 제어할 수 있습니다. 향상된 인증 흐름의 `GetCredentialsForIdentity` 요청은 액세스 토큰의 내용을 기반으로 역할을 요청합니다. 클래식 워크플로의 `AssumeRoleWithWebIdentity` 요청은 충분한 신뢰 정책으로 구성한 모든 AWS Identity and Access Management 역할에 대한 자격 증명을 앱에 더 많이 요청할 수 있는 권한을 부여합니다. 사용자 지정 역할 세션 기간을 요청할 수도 있습니다.

기본 인증의 작업 순서

1. `GetId`
2. `GetOpenIdToken`
3. `AssumeRoleWithWebIdentity`



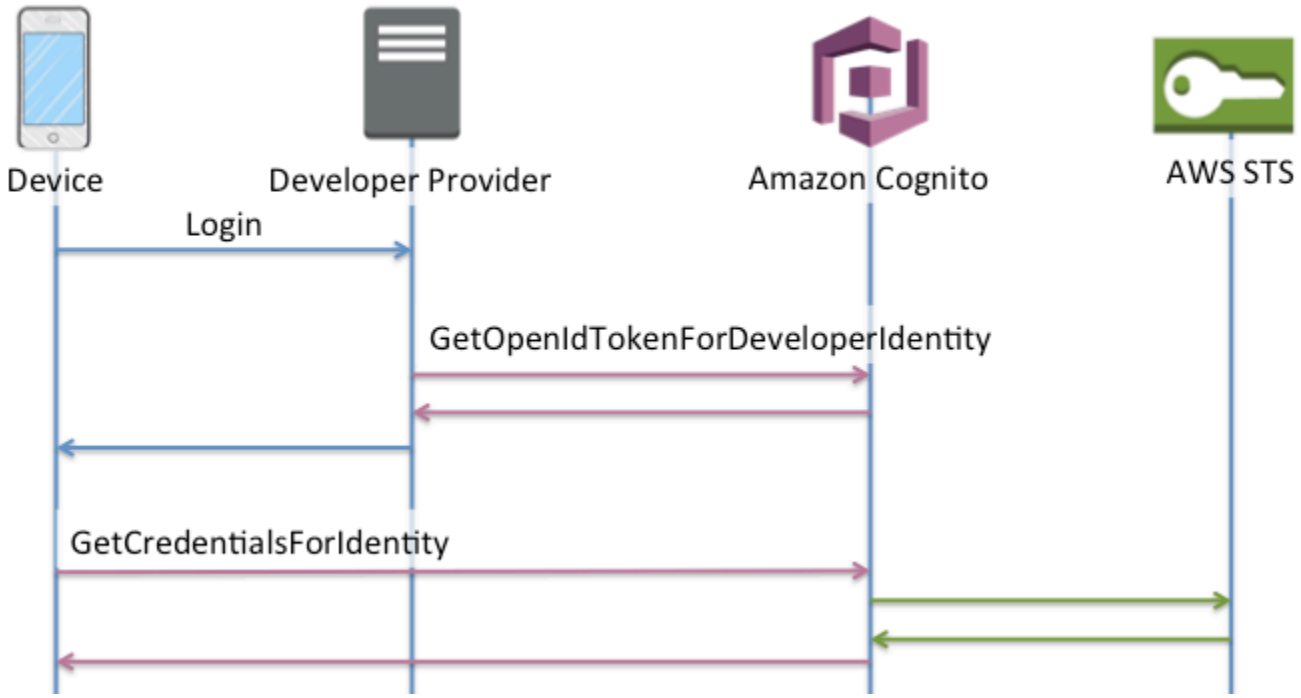
개발자 인증된 자격 증명 인증 흐름

개발자 인증 자격 증명(자격 증명 풀)을 사용할 때 클라이언트는 Amazon Cognito에 속하지 않는 코드가 포함된 다른 인증 흐름을 사용하여 자체 인증 시스템에서 사용자를 검증합니다. Amazon Cognito에 속하지 않는 코드는 다음과 같이 나타납니다.

향상된 인증 흐름

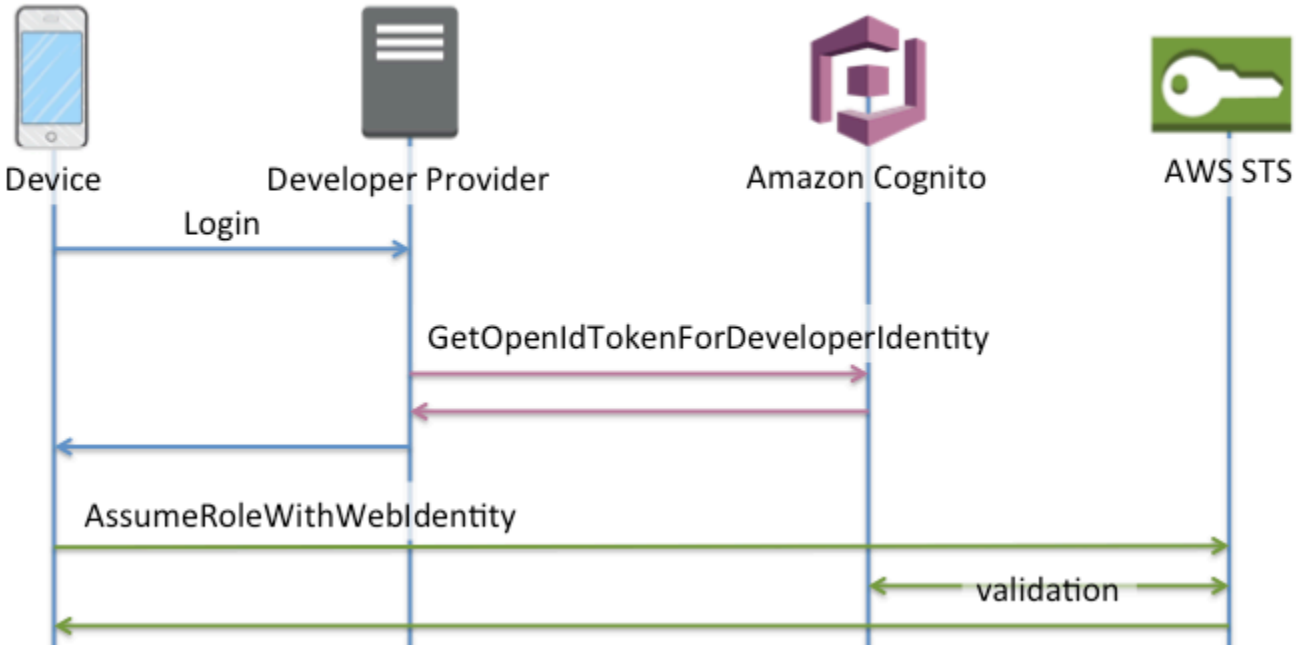
개발자 공급자를 통한 향상된 인증 작업 순서

1. 개발자 공급자를 통해 로그인(Amazon Cognito에 속하지 않는 코드)
2. 사용자 로그인 검증(Amazon Cognito에 속하지 않는 코드)
3. GetOpenIdTokenForDeveloperIdentity
4. GetCredentialsForIdentity



개발자 공급자와의 기본 인증 작업 순서

1. 로그인 및 개발자-제공자 식별자를 생성하기 위한 로직을 자격 증명 풀 외부에서 구현하십시오.
2. 저장된 서버측 AWS 자격 증명을 검색합니다.
3. 승인된 AWS 자격 증명으로 서명된 GetOpenIdTokenForDeveloperIdentityAPI 요청으로 개발자 제공자 식별자를 전송합니다.
4. 를 사용하여 애플리케이션 자격 증명을 요청하세요 AssumeRoleWithWebIdentity.



어떤 인증 흐름을 사용해야 합니까?

향상된 흐름은 개발자의 노력을 최소화하면서도 가장 안전한 선택입니다.

- 향상된 흐름은 API 요청의 복잡성, 크기 및 비율을 줄여줍니다.
- 애플리케이션은 추가 API 요청을 할 필요가 없습니다 AWS STS.
- 자격 증명 풀은 사용자가 받아야 하는 IAM 역할 자격 증명을 평가합니다. 클라이언트에 역할 선택을 위한 로직을 내장할 필요는 없습니다.

> **Important**
>
> 새 자격 증명 풀을 생성할 때는 기본 (클래식) 인증을 기본적으로 활성화하지 않는 것이 가장 좋습니다. 기본 인증을 구현하려면 먼저 웹 ID에 대한 IAM 역할의 신뢰 관계를 평가해야 합니다. 그런 다음 클라이언트에 역할 선택 로직을 구축하고 사용자가 수정하지 못하도록 클라이언트를 보호하십시오.

기본 인증 흐름은 IAM 역할 선택 로직을 애플리케이션에 위임합니다. 이 흐름에서 Amazon Cognito는 사용자의 인증된 세션 또는 인증되지 않은 세션을 검증하고 자격 증명으로 교환할 수 있는 토큰을 발급합니다. AWS STS사용자는 기본 인증의 토큰을 자격 증명 풀 및/또는 인증/비인증 상태를 신뢰하는 모든 IAM 역할과 교환할 수 있습니다. `amr`

마찬가지로 개발자 인증은 ID 제공자 인증 검증의 지름길이라는 점을 이해하세요. Amazon Cognito는 요청 내용에 대한 추가 검증 없이 GetOpenIdTokenForDeveloperIdentity요청을 승인하는 AWS 자격 증명을 신뢰합니다. 개발자 인증을 승인하는 암호를 사용자의 액세스로부터 보호하십시오.

API 요약

GetId

GetIdAPI 호출은 Amazon Cognito에서 새 자격 증명을 설정하는 데 필요한 첫 번째 호출입니다.

인증되지 않은 액세스

Amazon Cognito는 애플리케이션에서 인증되지 않은 게스트 액세스를 허용할 수 있습니다. 자격 증명 풀에서 이 기능이 활성화되면 사용자는 언제든지 `GetId` API를 통해 새 자격 증명 ID를 요청할 수 있습니다. 애플리케이션에서는 다음에 Amazon Cognito를 호출하기 위해 이 자격 증명 ID를 캐시해야 합니다. AWS 모바일 SDK와 브라우저용 AWS JavaScript SDK에는 이 캐싱을 처리하는 자격 증명 공급자가 있습니다.

인증된 액세스

퍼블릭 로그인 공급자(Facebook, Google+, Login with Amazon 또는 Sign in with Apple)의 지원을 통해 애플리케이션을 구성한 경우 사용자는 해당 공급자에서 자신을 식별하는 토큰(OAuth 또는 OpenID Connect)을 제공할 수도 있습니다. `GetId` 호출에 사용된 경우 Amazon Cognito는 새 인증 자격 증명을 생성하거나 이미 특정 로그인에 연결되어 있는 자격 증명을 반환합니다. Amazon Cognito는 공급자로 토큰을 검증하고 다음을 확인하여 이를 수행합니다.

- 토큰이 유효하며 구성된 공급자로부터 받음
- 토큰이 만료되지 않음
- 토큰이 해당 공급자를 통해 생성된 애플리케이션 식별자와 일치함(예: Facebook 앱 ID)
- 토큰이 사용자 식별자와 일치함

GetCredentialsForIdentity

ID ID를 설정한 후 GetCredentialsForIdentityAPI를 호출할 수 있습니다. 그렇다면 AssumeRoleWithWebIdentity이 작업은 함수를 GetOpenIdToken호출하는 것과 기능적으로 동일합니다.

사용자 대신 Amazon Cognito에서 `AssumeRoleWithWebIdentity`를 호출하기 위해 자격 증명 풀에 IAM 역할이 연결되어 있어야 합니다. Amazon Cognito 콘솔을 통해 이 작업을 수행하거나 작업을 통해 수동으로 수행할 수 있습니다. SetIdentityPoolRoles

GetOpenIdToken

자격 증명 ID를 설정한 후 GetOpenIdTokenAPI를 요청하십시오. 첫 번째 요청 후 자격 증명 ID를 캐시하고 GetOpenIdToken을 사용하여 해당 자격 증명에 대한 후속 기본(클래식) 세션을 시작합니다.

GetOpenIdToken API 요청에 대한 응답은 Amazon Cognito가 생성하는 토큰입니다. AssumeRoleWithWebIdentity요청의 WebIdentityToken 파라미터로 이 토큰을 제출할 수 있습니다.

OpenID 토큰을 제출하기 전에 앱에서 토큰을 확인하세요. SDK의 OIDC 라이브러리 또는 aws-jwt-verify와 같은 라이브러리를 사용하여 Amazon Cognito가 토큰을 발급했는지 확인할 수 있습니다. OpenID 토큰의 서명 키 ID, 즉 kid는 Amazon Cognito 자격 증명 jwks_uri 문서†에 나열된 것 중 하나입니다. 이러한 키는 변경될 수 있습니다. Amazon Cognito 자격 증명 토큰을 확인하는 함수는 jwks_uri 문서에서 키 목록을 정기적으로 업데이트해야 합니다. Amazon Cognito는 jwks_uri 캐시 제어 응답 헤더에 새로 고침 기간을 설정하며, 현재 max-age인 30일로 설정되어 있습니다.

인증되지 않은 액세스

미인증 자격 증명에 대한 토큰을 얻으려면 자격 증명 ID 자체만 필요합니다. 인증된 자격 증명이나 비활성화된 자격 증명에 대한 미인증 토큰은 가져올 수 없습니다.

인증된 액세스

인증 자격 증명이 있는 경우 이미 해당 자격 증명과 연관된 로그인에 대해 유효한 토큰을 하나 이상 전달해야 합니다. GetOpenIdToken을 호출하는 동안 전달된 모든 토큰은 이전에 언급한 동일한 검증을 통과해야 합니다. 토큰 중 하나가 실패한 경우 전체 호출이 실패합니다. GetOpenIdToken 호출의 응답에도 자격 증명 ID가 포함됩니다. 이는 전달된 자격 증명 ID가 반환된 자격 증명 ID가 아닐 수도 있기 때문입니다.

로그인 연결

아직 자격 증명과 연관되지 않은 로그인에 대한 토큰을 제출하면 해당 로그인이 연관된 자격 증명에 '연결된' 것으로 간주됩니다. 퍼블릭 공급자당 로그인 하나만 연결할 수 있습니다. 퍼블릭 공급자에 로그인을 둘 이상 연결하려고 시도하면 ResourceConflictException 오류 응답이 발생합니다. 로그인이 기존 자격 증명에 연결만 되어 있는 경우 GetOpenIdToken에서 반환된 자격 증명 ID는 전달된 자격 증명 ID와 동일합니다.

자격 증명 병합

현재 지정된 자격 증명과 연결되지 않았지만 다른 자격 증명에 연결된 로그인에 대한 토큰을 전달하면 자격 증명 두 개가 병합됩니다. 병합되면 자격 증명 하나가 연관된 모든 로그인의 상위/소유자가 되며 다른 자격 증명이 비활성화됩니다. 이 경우 상위/소유자의 자격 증명 ID가 반환됩

니다. 이 값이 다른 경우 로컬 캐시를 업데이트해야 합니다. 브라우저용 AWS 모바일 AWS SDK 또는 JavaScript SDK의 공급자가 이 작업을 대신 수행합니다.

GetOpenIdTokenForDeveloperIdentity

이 GetOpenIdTokenForDeveloperIdentity작업은 개발자 인증 ID를 사용할 때 디바이스를 사용하거나 GetOpenIdToken디바이스에서 사용하는 것을 대체합니다. GetId 애플리케이션이 AWS 자격 증명으로 이 API 작업에 대한 요청에 서명하므로 Amazon Cognito는 요청에 제공된 사용자 식별자가 유효하다고 신뢰합니다. 개발자 인증은 Amazon Cognito가 외부 공급자와 함께 수행하는 토큰 검증을 대체합니다.

이 API의 페이로드에는 맵이 포함되어 있습니다. `logins` 이 맵에는 개발자 제공자의 키와 시스템의 사용자 식별자 값이 포함되어야 합니다. 사용자 식별자가 기존 자격 증명에 아직 연결되지 않은 경우 Amazon Cognito는 새 자격 증명을 생성하고 새 자격 증명 ID와 해당 자격 증명에 대한 OpenID Connect 토큰을 반환합니다. 사용자 식별자가 이미 연결된 경우 Amazon Cognito는 기존 자격 증명 ID와 OpenID Connect 토큰을 반환합니다. 첫 번째 요청 후 개발자 자격 증명 ID를 캐시하고 `GetOpenIdTokenForDeveloperIdentity`을 사용하여 해당 자격 증명에 대한 후속 기본(클래식) 세션을 시작합니다.

`GetOpenIdTokenForDeveloperIdentity` API 요청에 대한 응답은 Amazon Cognito가 생성하는 토큰입니다. 이 토큰을 `AssumeRoleWithWebIdentity` 요청의 `WebIdentityToken` 파라미터로 제출할 수 있습니다.

OpenID Connect 토큰을 제출하기 전에 앱에서 토큰을 확인하세요. SDK의 OIDC 라이브러리 또는 aws-jwt-verify와 같은 라이브러리를 사용하여 Amazon Cognito가 토큰을 발급했는지 확인할 수 있습니다. OpenID 토큰의 서명 키 ID, 즉 `kid`는 Amazon Cognito 자격 증명 jwks_uri 문서†에 나열된 것 중 하나입니다. 이러한 키는 변경될 수 있습니다. Amazon Cognito 자격 증명 토큰을 확인하는 함수는 jwks_uri 문서에서 키 목록을 정기적으로 업데이트해야 합니다. Amazon Cognito는 jwks_uri `cache-control` 응답 헤더에 새로 고침 기간을 설정하며, 현재 `max-age`인 30일로 설정되어 있습니다.

로그인 연결

외부 공급자와 마찬가지로 아직 자격 증명에 연결되지 않은 추가 로그인을 제공하면 이러한 로그인과 해당 자격 증명이 암시적으로 연결됩니다. 외부 공급자 로그인을 자격 증명에 연결하면 사용자는 해당 공급자와 함께 외부 공급자 인증 흐름을 사용할 수 있습니다. 하지만 `GetId` 또는 `GetOpenIdToken`을 호출할 때는 로그인 맵에서 개발자 공급자 이름을 사용할 수 없습니다.

자격 증명 병합

개발자 인증 자격 증명을 사용하는 Amazon Cognito는 API를 통한 암시적 병합과 명시적 병합을 모두 지원합니다. MergeDeveloperIdentities 이러한 명시적 병합을 통해 시스템에서 사용자

식별자가 있는 두 개의 자격 증명을 단일 자격 증명으로 표시할 수 있습니다. 소스 및 대상 사용자 식별자를 제공하기만 하면 Amazon Cognito에서는 이를 병합합니다. 다음에 이 두 사용자 식별자에 대한 OpenId Connect 토큰을 요청하면 동일한 자격 증명 ID가 반환됩니다.

AssumeRoleWithWebIdentity

OpenID Connect 토큰이 있으면 AWS Security Token Service ()AWS STS에 대한 AssumeRoleWithWebIdentityAPI 요청을 통해 이 토큰을 임시 AWS 자격 증명으로 교환할 수 있습니다.

생성할 수 있는 자격 증명의 수에 대한 제한이 없으므로 사용자에게 부여되는 권한을 이해하는 것이 중요합니다. 애플리케이션에 대해 서로 다른 IAM 역할을 설정합니다. 하나는 인증되지 않은 사용자용이고 다른 하나는 인증된 사용자용입니다. Amazon Cognito 콘솔은 자격 증명 풀을 처음 설정할 때 기본 역할을 생성할 수 있습니다. 이러한 역할에는 사실상 권한이 부여되지 않습니다. 필요에 맞게 수정하십시오.

역할 트러스트 및 권한 단원에 대해 자세히 알아보세요.

† 기본 Amazon Cognito 자격 증명 jwks_uri 문서에는 대부분의 AWS 리전에 있는 자격 증명 풀의 토큰을 서명하는 키에 대한 정보가 포함되어 있습니다. 다음 리전에는 서로 다른 jwks_uri 문서가 있습니다.

Amazon Cognito Identity JSON web key URIs in other AWS 리전

AWS 리전	jwks_uri 문서의 경로
AWS GovCloud (미국 서부)	`https://cognito-identity.us-gov-west-1.amazonaws.com/.well-known/jwks_uri`
중국(베이징)	`https://cognito-identity.cn-north-1.amazonaws.com.cn/.well-known/jwks_uri`
유럽 (밀라노) 및 아프리카 (케이프타운) 와 같은 옵트인 지역	`https://cognito-identity.`*`Region`*`.amazonaws.com/.well-known/jwks_uri`

발급자(iss)가 제공한 jwks_uri를 외삽하거나 Amazon Cognito로부터 OpenID 토큰에서 jwks_uri를 외삽할 수도 있습니다. OIDC 표준 검색 엔드포인트 <issuer>/.well-known/openid-configuration는 토큰의 jwks_uri 경로를 나열합니다.

IAM 역할

자격 증명 풀을 생성하는 동안 사용자가 맡을 IAM 역할을 업데이트하라는 메시지가 표시됩니다. IAM 역할은 다음과 같이 작동합니다. 사용자가 앱에 로그인하면 Amazon Cognito는 사용자를 위한 AWS 임시 자격 증명을 생성합니다. 이러한 임시 자격 증명은 특정 IAM 역할에 연결됩니다. IAM 역할을 사용하면 리소스에 액세스할 수 있는 권한 세트를 정의할 수 있습니다. AWS

인증 및 미인증 사용자에 대해 기본 IAM 역할을 지정할 수 있습니다. 또한 사용자의 ID 토큰에 있는 클레임에 따라 각 사용자에 대한 역할을 선택하는 규칙을 정의할 수 있습니다. 자세한 내용은 역할 기반 액세스 제어 사용 섹션을 참조하세요.

기본적으로 Amazon Cognito 콘솔은 Amazon Mobile Analytics 및 Amazon Cognito Sync에 대한 액세스 권한을 제공하는 IAM 역할을 생성합니다. 또는 기존 IAM 역할을 사용하도록 선택할 수 있습니다.

다른 서비스에 대한 액세스를 허용하거나 제한하도록 IAM 역할을 수정합니다. 이렇게 하려면 IAM 콘솔에 로그인합니다. 그런 다음 역할(Roles)을 선택하고 역할을 선택합니다. 선택한 역할에 연결된 정책은 권한(Permissions) 탭에 나열됩니다. 해당하는 정책 관리(Manage Policy) 링크를 선택하여 액세스 정책을 사용자 지정할 수 있습니다. 정책 사용 및 정의에 대한 자세한 내용은 IAM 정책 개요를 참조하세요.

> **Note**
>
> 가장 좋은 방법은 최소 권한 부여 원칙을 따르는 정책을 정의하는 것입니다. 다시 말해, 해당 정책은 사용자가 작업을 수행하는 데 필요한 권한만을 포함합니다. 자세한 내용은 IAM 사용 설명서에서 최소 권한 부여를 참조하세요.
> 인증되지 않은 자격 증명은 앱에 로그인하지 않은 사용자에 의해 수임된다는 점을 유의하세요. 일반적으로, 인증되지 않은 자격 증명에 할당한 권한은 인증된 자격 증명의 권한보다 더 제한적이어야 합니다.

주제

- 트러스트 정책 설정
- 액세스 정책

트러스트 정책 설정

Amazon Cognito는 IAM 역할을 사용하여 애플리케이션의 사용자에 대한 임시 자격 증명을 생성합니다. 권한에 대한 액세스는 역할의 신뢰 관계에 의해 제어됩니다. 역할 트러스트 및 권한 섹션에 대해 자세히 알아봅니다.

제공되는 AWS STS 토큰은 자격 증명 풀에 의해 생성되며, 자격 증명 풀은 사용자 풀, 소셜 또는 OIDC 공급자 토큰 또는 SAML 어설션을 자체 토큰으로 변환합니다. 자격 증명 풀 토큰에는 자격 증명 풀 ID인 aud 클레임이 포함되어 있습니다.

다음 예제 역할 신뢰 정책은 페더레이션된 서비스 보안 주체가 API를 호출하도록 허용합니다. `cognito-identity.amazonaws.com` AWS STS `AssumeRoleWithWebIdentity` API 요청의 자격 증명 풀 토큰에 다음과 같은 클레임이 있는 경우에만 요청이 성공합니다.

1. 자격 증명 풀 ID `us-west-2:abcdefg-1234-5678-910a-0e8443553f95`에 대한 `aud` 클레임.
2. 사용자가 로그인했지만 게스트 사용자가 아닌 경우에 추가되는 `authenticated`에 대한 `amr` 클레임.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "cognito-identity.amazonaws.com"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "cognito-identity.amazonaws.com:aud": "us-
west-2:abcdefg-1234-5678-910a-0e8443553f95"
                },
                "ForAnyValue:StringLike": {
                    "cognito-identity.amazonaws.com:amr": "authenticated"
                }
            }
        }
    ]
}
```

기본 (클래식) 인증의 IAM 역할에 대한 신뢰 정책

자격 증명 풀과 함께 사용하는 역할에 대한 신뢰 정책을 제한하는 조건을 하나 이상 적용해야 합니다. 자격 증명 풀에 대한 역할 신뢰 정책을 생성하거나 업데이트할 때 원본 ID를 제한하는 조건 키가 하나 이상 없이 변경 내용을 저장하려고 하면 IAM에서 오류를 반환합니다. AWS STS 자격 증명 풀에서 이러한 유형의 조건이 없는 IAM 역할로의 계정 간 AssumeRoleWithWebIdentity작업은 허용되지 않습니다.

이 주제에는 자격 증명 풀의 원본 ID를 제한하는 몇 가지 조건이 포함되어 있습니다. 전체 목록은 AWS 웹 ID 페더레이션에 사용 가능한 키를 참조하십시오.

자격 증명 풀을 사용한 기본 또는 클래식 인증에서는 올바른 신뢰 정책이 있는 AWS STS 경우 모든 IAM 역할을 맡을 수 있습니다. Amazon Cognito 자격 증명 풀의 IAM 역할은 서비스 보안 주체 `cognito-identity.amazonaws.com`의 역할 수임을 신뢰합니다. 이 구성으로는 의도하지 않은 리소스 액세스로부터 IAM 역할을 보호하기에 충분하지 않습니다. 이 유형의 역할은 역할 신뢰 정책에 추가 조건을 적용해야 합니다. 다음 조건 중 하나 이상이 없으면 자격 증명 풀의 역할을 만들거나 수정할 수 없습니다.

`cognito-identity.amazonaws.com:aud`

역할을 하나 이상의 자격 증명 풀의 작업으로 제한합니다. Amazon Cognito는 `aud` 클레임의 소스 자격 증명 풀을 자격 증명 풀 토큰에 표시합니다.

`cognito-identity.amazonaws.com:amr`

역할을 사용자 `authenticated` 또는 `unauthenticated` (게스트) 사용자로 제한합니다. Amazon Cognito는 자격 증명 풀 토큰의 `amr` 클레임에 인증 상태를 표시합니다.

`cognito-identity.amazonaws.com:sub`

UUID를 기준으로 역할을 한 명 이상의 사용자로 제한합니다. 이 UUID는 자격 증명 풀에 있는 사용자의 ID ID입니다. 이 값은 사용자의 원래 ID 제공자의 `sub` 값이 아닙니다. Amazon Cognito는 자격 증명 풀 토큰의 `sub` 클레임에 이 UUID를 표시합니다.

향상된 흐름 인증을 사용하려면 IAM 역할이 자격 증명 AWS 계정 풀과 동일해야 하지만 기본 인증에서는 그렇지 않습니다.

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 특정 조건을 포함해야 합니다. `cognito-identity.amazonaws.com:aud` AWS 리소스에 의도

하지 않은 액세스를 방지하기 위해 aud 조건 키는 조건 값에 있는 자격 증명 풀의 사용자로 역할을 제한합니다.

자격 증명 풀이 자격 증명에 대해 발급하는 토큰에는 자격 증명 AWS 계정 풀의 출처에 대한 정보가 들어 있습니다. AssumeRoleWithWebIdentityAPI 요청에서 자격 증명 풀 토큰을 제시하면 원본 자격 증명 풀이 IAM AWS 계정 역할과 동일한지 AWS STS 확인합니다. 요청이 크로스 어카운트라고 AWS STS 판단되면 역할 신뢰 정책에 조건이 있는지 확인합니다. aud 역할 신뢰 정책에 해당 조건이 없는 경우 assume-role 호출은 실패합니다. 요청이 크로스 어카운트가 아닌 경우 이 제한을 적용하지 AWS STS 않습니다. 가장 좋은 방법은 항상 이 유형의 조건을 자격 증명 풀 역할의 신뢰 정책에 적용하는 것입니다.

추가 신뢰 정책 조건

자격 증명 풀에서 역할 재사용

여러 자격 증명 풀에서 역할을 재사용하려면 역할이 공통 권한 집합을 공유하므로 다음과 같이 여러 자격 증명 풀을 포함할 수 있습니다.

```
"StringEquals": {
    "cognito-identity.amazonaws.com:aud": [
        "us-east-1:12345678-abcd-abcd-abcd-123456790ab",
        "us-east-1:98765432-dcba-dcba-dcba-123456790ab"
    ]
}
```

특정 자격 증명으로 액세스 제한

특정 앱 사용자 집합에 제한된 정책을 생성하려면 cognito-identity.amazonaws.com:sub 값을 선택하세요.

```
"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"
    ]
}
```

특정 공급자로 액세스 제한

특정 공급자(고유한 로그인 공급자)를 통해 로그인한 사용자에 제한된 정책을 생성하려면 `cognito-identity.amazonaws.com:amr` 값을 선택하세요.

```
"ForAnyValue:StringLike": {
    "cognito-identity.amazonaws.com:amr": "login.myprovider.myapp"
}
```

예를 들어, Facebook만 신뢰하는 앱에는 다음 amr 절이 있습니다.

```
"ForAnyValue:StringLike": {
    "cognito-identity.amazonaws.com:amr": "graph.facebook.com"
}
```

액세스 정책

역할에 연결하는 권한은 해당 역할을 맡은 모든 사용자에게 적용됩니다. 사용자의 액세스를 분할하려면 정책 조건 및 변수를 사용해야 합니다. 자세한 내용은 IAM 정책 요소: 변수 및 태그 단원을 참조하세요. `sub` 조건을 사용하여 액세스 정책에서 Amazon Cognito ID에 대한 작업을 제한할 수 있습니다. 이 옵션은 주의해서 사용해야 하며, 일관된 사용자 ID가 부족한 미인증 ID의 경우에는 더욱 주의해야 합니다. Amazon Cognito와의 웹 페더레이션을 위한 IAM 정책 변수에 대한 자세한 내용은 사용 설명서의 IAM AWS STS 및 조건 컨텍스트 키를 참조하십시오.AWS Identity and Access Management

보안 강화를 위해, Amazon Cognito는 `GetCredentialsForIdentity`를 사용하여 향상된 흐름에서 미인증 사용자에 할당된 보안 인증 정보에 범위 축소 정책을 적용합니다. 범위 축소 정책은 미인증 역할에 적용하는 IAM 정책에 인라인 세션 정책 및 AWS 관리형 세션 정책을 추가합니다. 역할에 대한 IAM 정책과 세션 정책 모두에서 액세스 권한을 부여해야 하므로, 범위 축소 정책은 다음 목록에 있는 서비스 이외의 서비스에 대한 사용자의 액세스를 제한합니다.

> **Note**
>
> 기본(클래식) 흐름에서는 자체 AssumeRoleWithWebIdentity API 요청을 하고, 이러한 제한을 요청에 적용할 수 있습니다. 보안 모범 사례는 미인증 사용자에게 이 범위 제한 정책을 초과하는 권한을 할당하지 않는 것입니다.

또한 Amazon Cognito는 인증된 사용자와 인증되지 않은 사용자가 Amazon Cognito 자격 증명 풀 및 Amazon Cognito Sync에 API를 요청하는 것을 방지합니다. 웹 ID를 통한 서비스 액세스를 제한하는 곳도 AWS 서비스 있을 수 있습니다.

향상된 흐름으로 요청을 하는 데 성공하면, Amazon Cognito가 백그라운드에서 `AssumeRoleWithWebIdentity` API 요청을 합니다. 이 요청의 파라미터 중에서 Amazon Cognito에는 다음이 포함됩니다.

1. 사용자의 자격 증명 ID입니다.
2. 사용자가 수임할 IAM 역할의 ARN입니다.
3. 인라인 세션 정책을 추가하는`policy` 파라미터입니다.
4. Amazon에서 추가 권한을 부여하는 AWS 관리형 정책이 값을 가지는 `PolicyArns.member.N` CloudWatch 파라미터입니다.

미인증 사용자가 액세스할 수 있는 서비스

향상된 흐름을 사용하는 경우, Amazon Cognito가 사용자 세션에 적용하는 범위 축소 정책에 따라 사용자는 다음 표에 나열된 서비스 이외의 서비스를 사용할 수 없습니다. 일부 서비스의 경우, 특정 작업만 허용됩니다.

범주	Service
분석	Amazon Data Firehose Amazon Managed Service for Apache Flink
애플리케이션 통합	Amazon Simple Queue Service
AR 및 VR	Amazon Sumerian[1]
비즈니스 애플리케이션	Amazon Mobile Analytics Amazon Simple Email Service
컴퓨팅	AWS Lambda
암호화 및 PKI	AWS Key Management Service[1]
데이터베이스	Amazon DynamoDB Amazon SimpleDB
프런트 엔드 웹 및 모바일	AWS AppSync

범주	Service
	Amazon Location Service Amazon Simple Notification Service Amazon Pinpoint
게임 개발	아마존 GameLift
사물 인터넷(IoT)	AWS IoT
기계 학습	아마존 CodeWhisperer Amazon Comprehend Amazon Lex Amazon Machine Learning Personalize Amazon Polly Amazon Rekognition 아마존 SageMaker [1] Amazon Textract[1] Amazon Transcribe Amazon Translate
관리 및 거버넌스	아마존 CloudWatch 아마존 CloudWatch 로그
네트워킹 및 콘텐츠 전송	Amazon API Gateway
보안, 자격 증명 및 규정 준수	Amazon Cognito 사용자 풀
스토리지	Amazon Simple Storage Service(S3)

[1] 다음 AWS 서비스 표의 경우 인라인 정책은 작업의 하위 집합을 허용합니다. 표에는 각 항목에서 사용할 수 있는 작업이 표시됩니다.

AWS 서비스	미인증 향상된 흐름 사용자의 최대 권한

AWS Key Management Service	`Encrypt`
	`Decrypt`
	`ReEncrypt`
	`GenerateDataKey`

아마존 SageMaker	`InvokeEndpoint`
Amazon Textract	`DetectDocumentText` `AnalyzeDocument`
Amazon Sumerian	`View*`

이 목록 AWS 서비스 이외의 항목에 대한 액세스 권한을 부여하려면 자격 증명 풀에서 기본 (클래식) 인증 흐름을 활성화하십시오. 미인증 사용자의 IAM 역할에 할당된 정책에서 허용하는 AWS 서비스 의 `NotAuthorizedException` 오류가 사용자에게 표시된다면, 사용 사례에서 해당 서비스를 제거할 수 있는지 여부를 평가하세요. 제거할 수 없다면 기본 흐름으로 전환하세요.

인라인 세션 정책

인라인 세션 정책은 사용자의 유효 권한이 다음 목록에 있는 권한 AWS 서비스 이외의 모든 권한에 대한 액세스를 포함하는 것을 제한합니다. 또한 사용자의 IAM 역할에 적용하는 AWS 서비스 정책에서 이러한 사용자에게 권한을 부여해야 합니다. 수임하는 역할 세션에 대한 사용자의 유효 권한은 해당 역할에 할당된 정책과 관련 세션 정책의 교차점입니다. 자세한 정보는 AWS Identity and Access Management 사용 설명서의 세션 정책을 참조하세요.

Amazon Cognito는 기본적으로 활성화된 AWS 리전 의 사용자 세션에 다음과 같은 인라인 정책을 추가합니다.

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

```
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:*",
                "logs:*",
                "dynamodb:*",
                "kinesis:*",
                "mobileanalytics:*",
                "s3:*",
                "ses:*",
                "sns:*",
                "sqs:*",
                "lambda:*",
                "machinelearning:*",
                "execute-api:*",
                "iot:*",
                "gamelift:*",
                "scs:*",
                "cognito-identity:*",
                "cognito-idp:*",
                "lex:*",
                "polly:*",
                "comprehend:*",
                "translate:*",
                "transcribe:*",
                "rekognition:*",
                "mobiletargeting:*",
                "firehose:*",
                "appsync:*",
                "personalize:*",
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "sagemaker:InvokeEndpoint",
                "cognito-sync:*",
                "sumerian:View*",
                "codewhisperer:*",
                "textract:DetectDocumentText",
                "textract:AnalyzeDocument",
                "sdb:*"
            ],
            "Resource": [
                "*"
```

```
            ]
        }
    ]
}
```

다른 모든 리전의 경우, 인라인 범위 축소 정책에는 다음 Action 문을 제외하고 기본 리전에 나열된 모든 항목이 포함됩니다.

```
                "cognito-sync:*",
                "sumerian:View*",
                "codewhisperer:*",
                "textract:DetectDocumentText",
                "textract:AnalyzeDocument",
                "sdb:*"
```

AWS 관리형 세션 정책

또한 Amazon Cognito는 AWS 관리형 정책 AmazonCognitoUnAuthedIdentitiesSessionPolicy를 통해 인증되지 않은 사용자의 권한 범위를 향상된 흐름에서 인증되지 않은 사용자로 제한합니다. 인증되지 않은 IAM 역할에 연결하는 정책에서도 이 권한을 부여해야 합니다.

AmazonCognitoUnAuthedIdentitiesSessionPolicy 관리형 정책에는 다음 권한이 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "rum:PutRumEvents",
            "polly:*",
            "comprehend:*",
            "translate:*",
            "transcribe:*",
            "rekognition:*",
            "mobiletargeting:*",
            "firehose:*",
            "personalize:*",
            "sagemaker:InvokeEndpoint"
        ],
        "Resource": "*"
    }]
```

```
}
```

액세스 정책 예제

이 섹션에서는 특정 작업을 완료하는 데 필요한 최소 권한만 사용자에게 부여하는 Amazon Cognito 액세스 정책의 예시를 찾을 수 있습니다. 가능한 경우 정책 변수를 사용하여 지정된 자격 증명 ID에 대한 권한을 한층 더 제한할 수 있습니다. 예를 들면, ${cognito-identity.amazonaws.com:sub}를 사용합니다. 자세한 내용은 AWS 모바일 블로그에서 Amazon Cognito 인증 이해 3부: 역할 및 정책을 참조하세요.

> **Note**
>
> 보안 모범 사례로, 정책은 사용자가 작업을 수행하는 데 필요한 권한만을 포함해야 합니다. 다시 말해서 가능한 경우 항상 액세스 범위를 객체에 대한 개별 자격 증명으로 조정해야 합니다.

Amazon S3에서 단일 객체에 대한 읽기 권한을 자격 증명에 부여

다음 액세스 정책은 지정된 S3 버킷에서 단일 객체를 검색할 수 있는 읽기 권한을 자격 증명에게 부여합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::mybucket/assets/my_picture.jpg"]
    }
  ]
}
```

Amazon S3에서 자격 증명 관련 경로에 대한 읽기 및 쓰기 권한을 모두 자격 증명에 부여

다음 액세스 정책은 접두사를 `${cognito-identity.amazonaws.com:sub}` 변수에 매핑하여 S3 버킷에서 특정 접두사 “folder”에 액세스할 수 있는 읽기 및 쓰기 권한을 부여합니다.

이 정책을 사용하면 `${cognito-identity.amazonaws.com:sub}`를 통해 삽입된 `us-east-1:12345678-1234-1234-1234-123456790ab`와 같은 자격 증명은 객체를 `arn:aws:s3:::mybucket/us-east-1:12345678-1234-1234-1234-123456790ab`로 가져오

고 넣고 나열할 수 있습니다. 하지만 arn:aws:s3:::mybucket의 다른 객체에 대한 액세스는 이 자격 증명에게 부여되지 않습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::mybucket"],
      "Condition": {"StringLike": {"s3:prefix": ["${cognito-
identity.amazonaws.com:sub}/*"]}}
    },
    {
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::mybucket/${cognito-identity.amazonaws.com:sub}/*"]
    }
  ]
}
```

자격 증명에 Amazon DynamoDB에 대한 세분화된 액세스 권한 할당

다음 액세스 정책은 Amazon Cognito 환경 변수를 사용하여 Amazon DynamoDB 리소스에 대한 세분화된 액세스 제어를 제공합니다. 이러한 변수는 자격 증명 ID를 기준으로 DynamoDB의 항목에 대한 액세스 권한을 부여합니다. 자세한 내용은 Amazon DynamoDB 개발자 가이드에서 IAM 정책 조건을 사용하여 세부적인 액세스 제어 구현을 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:us-west-2:123456789012:table/MyTable"
      ],
      "Condition": {
        "ForAllValues:StringEquals": {
          "dynamodb:LeadingKeys":  ["${cognito-identity.amazonaws.com:sub}"]
        }
      }
    }
  ]
}
```

Lambda 함수를 호출할 수 있는 권한을 자격 증명에 부여

다음 액세스 정책은 Lambda 함수를 호출할 수 있는 권한을 자격 증명에 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
             "Effect": "Allow",
             "Action": "lambda:InvokeFunction",
             "Resource": [
                 "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
             ]
         }
     ]
 }
```

레코드를 Amazon Kinesis Data Streams에 게시할 수 있는 권한을 자격 증명에 부여

다음 액세스 정책은 자격 증명이 Kinesis Data Streams 중 하나에 대해 `PutRecord` 작업을 사용하도록 허용합니다. 계정의 모든 스트림에 데이터 레코드를 추가해야 하는 사용자에게 이 정책을 적용할 수 있습니다. 자세한 내용은 Amazon Kinesis Data Streams 개발자 가이드에서 IAM을 사용하여 Amazon Kinesis Data Streams 리소스에 대한 액세스 제어를 참조하세요.

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

```
        {
            "Effect": "Allow",
            "Action": "kinesis:PutRecord",
            "Resource": [
                "arn:aws:kinesis:us-east-1:111122223333:stream/stream1"
            ]
        }
    ]
}
```

Amazon Cognito Sync 스토어에 있는 자신의 데이터에 액세스할 수 있는 권한을 자격 증명에 부여

다음 액세스 정책은 Amazon Cognito Sync 스토어에 있는 자신의 데이터에만 액세스할 수 있는 권한을 자격 증명에 부여합니다.

```
{
  "Version": "2012-10-17",
  "Statement":[{
      "Effect":"Allow",
      "Action":"cognito-sync:*",
      "Resource":["arn:aws:cognito-sync:us-east-1:123456789012:identitypool/${cognito-identity.amazonaws.com:aud}/identity/${cognito-identity.amazonaws.com:sub}/*"]
      }]
  }
```

역할 트러스트 및 권한

이러한 역할이 서로 다른 방식은 신뢰 관계에 있습니다. 다음은 인증되지 않은 역할에 대한 신뢰 정책의 예입니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Federated": "cognito-identity.amazonaws.com"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
```

```
          "cognito-identity.amazonaws.com:aud": "us-east-1:12345678-corner-
cafe-123456790ab"
        },
        "ForAnyValue:StringLike": {
          "cognito-identity.amazonaws.com:amr": "unauthenticated"
        }
      }
    }
  ]
}
```

이 정책은 cognito-identity.amazonaws.com의 페더레이션 사용자(OpenID Connect 토큰의 발급자)에게 이 역할을 맡을 수 있는 권한을 부여합니다. 또한 이 정책은 토큰의 aud(이 경우 자격 증명 풀 ID)가 자격 증명 풀과 일치하도록 제한합니다. 마지막으로, 정책은 unauthenticated 값을 지닌 Amazon Cognito GetOpenIdToken API 작업에서 발행한 토큰의 다중 값 amr 클레임의 배열 멤버 중 하나를 지정합니다

Amazon Cognito에서 토큰을 생성하면 해당 토큰의 amr을 unauthenticated 또는 authenticated로 설정합니다. amr이 authenticated이면 토큰에 인증 중에 사용되는 모든 공급자가 포함됩니다. 즉, amr 조건을 다음과 같이 변경하여 Facebook을 통해 로그인한 사용자만 신뢰하는 역할을 생성할 수 있습니다.

```
"ForAnyValue:StringLike": {
  "cognito-identity.amazonaws.com:amr": "graph.facebook.com"
}
```

역할에 대한 신뢰 관계를 변경할 때 또는 자격 증명 풀에 대해 역할을 사용하려고 시도할 때 주의를 기울이세요. 역할이 자격 증명 풀을 신뢰하도록 올바르게 구성되지 않은 경우 다음과 같은 STS 예외가 발생합니다.

```
AccessDenied -- Not authorized to perform sts:AssumeRoleWithWebIdentity
```

이러한 메시지가 표시되면 자격 증명 풀과 인증 유형에서 적절한 역할을 사용하는지 확인합니다.

Amazon Cognito 자격 증명 풀의 보안 모범 사례

Amazon Cognito 자격 증명 풀은 애플리케이션에 대한 임시 AWS 자격 증명을 제공합니다. AWS 계정 애플리케이션 사용자에게 필요한 리소스와 프라이빗 백엔드 리소스를 모두 포함하는 경우가 많습니다. AWS 자격 증명을 구성하는 IAM 역할 및 정책은 이러한 모든 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자격 증명 풀 구성의 기본 모범 사례는 애플리케이션이 과도하거나 의도하지 않은 권한 없이 작업을 완료할 수 있도록 하는 것입니다. 잘못된 보안 구성을 방지하려면 프로덕션에 릴리스하려는 각 애플리케이션을 시작하기 전에 이러한 권장 사항을 검토하십시오.

주제

IAM 구성 모범 사례

게스트 또는 인증된 사용자가 애플리케이션에서 자격 증명 풀 자격 증명이 필요한 세션을 시작하면 애플리케이션은 IAM 역할에 대한 임시 AWS 자격 증명을 검색합니다. 자격 증명은 기본 역할, 자격 증명 풀 구성의 규칙에 따라 선택한 역할 또는 앱에서 선택한 사용자 지정 역할용일 수 있습니다. 각 역할에 할당된 권한으로 사용자는 AWS 리소스에 액세스할 수 있습니다.

일반적인 IAM 모범 사례에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 IAM 모범 사례를 참조하십시오.

IAM 역할에서 신뢰 정책 조건을 사용하십시오.

IAM을 사용하려면 자격 증명 풀의 역할에 신뢰 정책 조건이 하나 이상 있어야 합니다. 예를 들어 이 조건은 역할 범위를 인증된 사용자로만 설정할 수 있습니다. AWS STS 또한 계정 간 기본 인증 요청에는 두 가지 특정 조건 (및) 이 있어야 합니다. `cognito-identity.amazonaws.com:aud` `cognito-identity.amazonaws.com:amr` 가장 좋은 방법은 자격 증명 풀 서비스 보안 주체를 신뢰하는 모든 IAM 역할에 이 두 조건을 모두 적용하는 것입니다. `cognito-identity.amazonaws.com`

- `cognito-identity.amazonaws.com:aud`: 자격 증명 풀 토큰의 aud 클레임은 신뢰할 수 있는 자격 증명 풀 ID와 일치해야 합니다.
- `cognito-identity.amazonaws.com:amr`: 자격 증명 풀 토큰의 amr 클레임은 인증되거나 인증되지 않았어야 합니다. 이 조건을 사용하면 인증되지 않은 게스트 또는 인증된 사용자에게만 역할에 대한 액세스 권한을 예약할 수 있습니다. 예를 들어 이 조건의 값을 추가로 조정하여 역할을 특정 공급자의 사용자로 제한할 수 있습니다. `graph.facebook.com`

다음 예제 역할 신뢰 정책은 다음과 같은 조건에서 역할에 대한 액세스 권한을 부여합니다.

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

```
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Federated": "cognito-identity.amazonaws.com"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "cognito-identity.amazonaws.com:aud": "us-east-1:a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        },
        "ForAnyValue:StringLike": {
          "cognito-identity.amazonaws.com:amr": "authenticated"
        }
      }
    }
  ]
}
```

자격 증명 풀과 관련된 요소

- `"Federated": "cognito-identity.amazonaws.com"`: 사용자는 자격 증명 풀에 속해야 합니다.
- `"cognito-identity.amazonaws.com:aud": "us-east-1:a1b2c3d4-5678-90ab-cdef-example11111"`: 사용자는 특정 자격 증명 풀에 속해야 `us-east-1:a1b2c3d4-5678-90ab-cdef-example11111` 합니다.
- `"cognito-identity.amazonaws.com:amr": "authenticated"`: 사용자는 인증을 받아야 합니다. 게스트 사용자는 역할을 수임할 수 없습니다.

최소 권한 권한 적용

IAM 정책으로 인증된 액세스 또는 게스트 액세스에 대한 권한을 설정하는 경우 특정 작업을 수행하는 데 필요한 특정 권한만 부여하거나 최소 권한 권한만 부여하십시오. 다음 예제 IAM 정책은 역할에 적용될 경우 Amazon S3 버킷의 단일 이미지 파일에 대한 읽기 전용 액세스 권한을 부여합니다.

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

```
    {
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::mybucket/assets/my_picture.jpg"]
    }
  ]
}
```

자격 증명 풀 구성 모범 사례

자격 증명 풀에는 AWS 자격 증명 생성을 위한 유연한 옵션이 있습니다. 애플리케이션이 가장 안전한 방법으로 작동할 수 있는 경우에는 설계 단축키를 사용하지 마십시오.

게스트 액세스의 영향을 이해하세요.

인증되지 않은 게스트 액세스를 사용하면 사용자가 AWS 계정 로그인하기 전에 사이트에서 데이터를 검색할 수 있습니다. 자격 증명 풀 ID를 아는 사람은 누구나 인증되지 않은 자격 증명을 요청할 수 있습니다. 자격 증명 풀 ID는 기밀 정보가 아닙니다. 게스트 액세스를 활성화하면 인증되지 않은 세션에 부여한 AWS 권한을 모든 사람이 사용할 수 있습니다.

가장 좋은 방법은 게스트 액세스를 비활성화한 상태로 두고 사용자가 인증한 후에만 필요한 리소스를 가져오는 것입니다. 애플리케이션에서 로그인하기 전에 리소스에 액세스해야 하는 경우 다음 예방 조치를 취하세요.

- 인증되지 않은 역할에 적용되는 자동 제한 사항을 숙지하세요.
- 애플리케이션의 특정 요구 사항에 맞게 인증되지 않은 IAM 역할의 권한을 모니터링하고 조정하십시오.
- 특정 리소스에 대한 액세스 권한을 부여하십시오.
- 기본 미인증 IAM 역할의 신뢰 정책을 보호하십시오.
- 인터넷에 있는 누구에게나 IAM 역할 권한을 부여할 수 있다고 확신하는 경우에만 게스트 액세스를 활성화하십시오.

기본적으로 향상된 인증을 사용합니다.

기본 (클래식) 인증을 사용하면 Amazon Cognito는 IAM 역할 선택을 앱에 위임합니다. 반면, 향상된 흐름은 자격 증명 풀의 중앙 집중식 로직을 사용하여 IAM 역할을 결정합니다. 또한 IAM 권한의 상한선을

설정하는 범위 축소 정책을 통해 인증되지 않은 ID에 대한 추가 보안을 제공합니다. 향상된 흐름은 개발자의 노력을 최소화하면서도 가장 안전한 선택입니다. 이러한 옵션에 대한 자세한 내용은 을 참조하십시오자격 증명 풀(페더레이션 자격 증명) 인증 흐름.

기본 흐름은 자격 증명에 대한 AWS STS API 요청의 역할 선택 및 어셈블리에 들어가는 클라이언트 측 로직을 노출할 수 있습니다. 향상된 흐름은 로직과 역할 수임 요청을 모두 자격 증명 풀 자동화의 이면에 숨깁니다.

기본 인증을 구성할 때 IAM 모범 사례를 IAM 역할 및 권한에 적용하십시오.

개발자 공급자를 안전하게 사용하십시오.

개발자 인증 ID는 서버 측 애플리케이션을 위한 ID 풀의 기능입니다. 자격 증명 풀에서 개발자 인증을 위해 요구하는 유일한 인증 증거는 자격 증명 풀 개발자의 AWS 자격 증명입니다. 자격 증명 풀은 이 인증 흐름에 제시하는 개발자-제공자 식별자의 유효성에 어떠한 제한도 적용하지 않습니다.

가장 좋은 방법은 다음과 같은 조건에서만 개발자 제공자를 구현하는 것입니다.

- 개발자 인증 자격 증명의 사용에 대한 책임을 부여하려면 인증 소스를 나타내는 개발자 제공자 이름과 식별자를 설계해야 합니다. 예를 들면 `"Logins" : {"MyCorp provider" : "[provider application ID]"}`입니다.
- 수명이 긴 사용자 자격 증명은 피하세요. EC2 인스턴스 프로필 및 Lambda 실행 역할과 같은 서비스 연결 역할을 사용하여 ID를 요청하도록 서버 측 클라이언트를 구성합니다.
- 동일한 자격 증명 풀에서 내부 및 외부 신뢰 소스를 혼용하지 마십시오. 개발자 공급자와 싱글 사인온 (SSO) 공급자를 별도의 자격 증명 풀에 추가하세요.

액세스 제어에 속성 사용

액세스 제어를 위한 속성은 ABAC(속성 기반 액세스 제어)의 Amazon Cognito 자격 증명 풀 구현입니다. IAM 정책을 사용하여 사용자 속성을 기반으로 Amazon Cognito 자격 증명 풀을 통한 AWS 리소스 액세스를 제어할 수 있습니다. 이러한 속성은 소셜 및 기업 자격 증명 공급자에서 가져올 수 있습니다. 공급자의 액세스 및 ID 토큰 또는 SAML 어설션 내의 속성을 IAM 권한 정책에서 참조되는 태그에 매핑할 수 있습니다.

Amazon Cognito 자격 증명 풀에서 기본 매핑을 선택하거나 사용자 지정 매핑을 생성할 수 있습니다. 기본 매핑을 사용하면 고정된 사용자 속성 세트를 기반으로 IAM 정책을 작성할 수 있습니다. 사용자 지정 매핑을 사용하면 IAM 권한 정책에서 참조되는 사용자 속성 세트를 선택할 수 있습니다. Amazon Cognito 콘솔의 속성 이름이 IAM 권한 정책에서 참조되는 태그인 보안 주체의 태그 키에 매핑됩니다.

예를 들어 무료 멤버십과 유료 멤버십이 있는 미디어 스트리밍 서비스를 보유하고 있다고 가정해 보겠습니다. 미디어 파일을 Amazon S3에 저장하고 무료 또는 프리미엄 태그로 태깅합니다. 액세스 제어를 위한 속성을 사용하여 사용자 프로필의 일부인 사용자 멤버십 수준에 따라 무료 및 유료 콘텐츠에 대한 액세스를 허용할 수 있습니다. IAM 권한 정책에 전달될 보안 주체의 태그 키에 멤버십 속성을 매핑할 수 있습니다. 이렇게 하면 단일 권한 정책을 생성하고 콘텐츠 파일의 멤버십 수준 및 태그 값에 따라 프리미엄 콘텐츠에 대한 액세스를 조건부로 허용할 수 있습니다.

주제

- Amazon Cognito 자격 증명 풀에서 액세스 제어에 속성 사용
- 액세스 제어에 속성 사용 정책 예
- 액세스 제어를 위한 속성 끄기(콘솔)
- 기본 공급자 매핑

속성을 사용하여 액세스를 제어하면 여러 가지 장점이 있습니다.

- 액세스 제어를 위한 속성을 사용하면 더 효율적으로 권한을 관리할 수 있습니다. 직무별로 여러 개의 정책을 만드는 대신 사용자 속성을 사용하는 기본 권한 정책을 만들 수 있습니다.
- 애플리케이션에 리소스 또는 사용자를 추가하거나 제거할 때마다 정책을 업데이트할 필요가 없습니다. 권한 정책은 일치하는 사용자 속성을 가진 사용자에게만 액세스 권한을 부여합니다. 예를 들어 사용자의 직책에 따라 특정 S3 버킷에 대한 액세스를 제어해야 할 수 있습니다. 이 경우 정의된 작책 내의 사용자만 이러한 파일에 액세스하도록 허용하는 권한 정책을 생성할 수 있습니다. 자세한 내용은 IAM 자습서: ABAC에 SAML 세션 태그 사용을 참조하세요.
- 해당 속성의 값에 따라 권한을 허용하거나 거부하는 정책에 속성을 보안 주체 태그로 전달할 수 있습니다.

Amazon Cognito 자격 증명 풀에서 액세스 제어에 속성 사용

액세스 제어에 속성을 사용하려면 먼저 다음 사전 요구 사항을 충족해야 합니다.

- AWS 계정
- 사용자 풀
- 자격 증명 풀
- SDK 설정
- 통합된 자격 증명 공급자
- 보안 인증

액세스 제어를 위한 속성을 사용하기 위해서는 데이터 세트 소스로 설정한 클레임이 선택한 태그 키의 값을 설정합니다. Amazon Cognito는 태그 키와 값을 사용자 세션에 적용합니다. IAM 정책은 `${aws:PrincipalTag/tagkey}` 조건을 통해 사용자의 액세스를 평가할 수 있습니다. IAM은 정책과 비교하여 사용자 태그의 값을 평가합니다.

보안 인증 정보를 사용자에게 전달할 IAM 역할을 준비해야 합니다. 이러한 역할의 신뢰 정책에서 Amazon Cognito가 사용자에 대한 역할을 맡을 수 있도록 허용해야 합니다. 액세스 제어 속성의 경우 Amazon Cognito가 사용자 임시 세션에 보안 주체 태그를 적용할 수 있도록 허용해야 합니다. AssumeRoleWithWebIdentity 작업을 통해 이러한 역할을 맡을 수 있는 권한을 부여합니다. 권한 전용 작업 `sts:TagSession`을 사용하여 사용자 세션에 태그를 지정할 수 있는 권한을 부여합니다. 자세한 내용은 AWS Identity and Access Management 사용 설명서의 AWS Security Token Service에서 세션 태그 전달을 참조하세요. Amazon Cognito 서비스 보안 주체 `cognito-identity.amazonaws.com`에 `sts:AssumeRoleWithWebIdentity` 및 `sts:TagSession` 권한을 부여하는 신뢰 정책의 예는 액세스 제어에 속성 사용 정책 예 섹션을 참조하세요.

콘솔에서 액세스 제어를 위한 속성을 구성하려면

1. Amazon Cognito 콘솔에 로그인하고 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. ID 제공업체를 찾습니다. 편집할 ID 제공업체를 선택합니다. 새 IdP를 추가하려면 ID 제공업체 추가를 선택합니다.
4. Amazon Cognito가 이 공급자를 통해 인증한 사용자에게 보안 인증을 발급할 때 할당하는 보안 주체 태그를 변경하려면 액세스 제어를 위한 속성에서 편집을 선택합니다.
    a. 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.
    b. sub 및 aud 클레임 기반 보안 주체 태그를 적용하려면 기본 매핑 사용을 선택합니다.
    c. 보안 주체 태그에 대한 속성의 자체 사용자 지정 스키마를 생성하려면 사용자 지정 매핑 사용을 선택합니다. 그런 다음 태그에 표시하려는 각 클레임에서 소싱하려는 태그 키를 입력합니다.
5. 변경 사항 저장(Save changes)을 선택합니다.

액세스 제어에 속성 사용 정책 예

회사 법무 부서의 직원이 해당 부서에 속하고 보안 수준으로 분류된 버킷의 모든 파일을 나열해야 하는 시나리오를 생각해 보세요. 이 직원이 자격 증명 공급자로부터 가져온 토큰에 다음 클레임이 포함되어 있다고 가정합니다.

클레임

```
{ .
  .
"sub" : "57e7b692-4f66-480d-98b8-45a6729b4c88",
"department" : "legal",
"clearance" : "confidential",
 .
 .
}
```

이러한 속성은 태그에 매핑될 수 있으며 IAM 권한 정책에서 보안 주체 태그로 참조할 수 있습니다. 이제 자격 증명 제공자의 끝에서 사용자 프로파일을 변경하여 액세스를 관리할 수 있습니다. 또는 정책 자체를 변경하지 않고 이름이나 태그를 사용하여 리소스 측에서 속성을 변경할 수 있습니다.

다음 권한 정책은 두 가지 기능을 합니다.

- 사용자의 부서 이름과 일치하는 접두사로 끝나는 모든 S3 버킷에 대한 목록 액세스를 허용합니다.
- 파일의 승인 태그가 사용자의 승인 속성과 일치하는 한 이러한 버킷의 파일에 대한 읽기 액세스를 허용합니다.

권한 정책

```
{
 "Version": "2012-10-17",
 "Statement": [
     {
         "Effect": "Allow",
         "Action": "s3:List*",
         "Resource": "arn:aws:s3:::*-${aws:PrincipalTag/department}"
     },
     {
         "Effect": "Allow",
         "Action": "s3:GetObject*",
         "Resource": "arn:aws:s3:::*-${aws:PrincipalTag/department}/*",
         "Condition": {
             "StringEquals": {
                 "s3:ExistingObjectTag/clearance": "${aws:PrincipalTag/clearance}"
```

```
                }
            }
        }
    ]
}
```

신뢰 정책은 이 역할을 수임할 수 있는 사용자를 결정합니다. 신뢰 관계 정책은 `sts:AssumeRoleWithWebIdentity` 및 `sts:TagSession`을 사용하여 액세스하도록 허용합니다. 생성한 자격 증명 풀로 정책을 제한하는 조건을 추가하고 인증된 역할에만 정책이 적용되도록 합니다.

신뢰 정책

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "cognito-identity.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRoleWithWebIdentity",
        "sts:TagSession"
      ],
      "Condition": {
        "StringEquals": {
          "cognito-identity.amazonaws.com:aud": "IDENTITY-POOL-ID"
        },
        "ForAnyValue:StringLike": {
          "cognito-identity.amazonaws.com:amr": "authenticated"
        }
      }
    }
  ]
}
```

액세스 제어를 위한 속성 끄기(콘솔)

액세스 제어를 위한 속성을 비활성화하려면 다음 절차를 따릅니다.

액세스 제어를 위한 속성을 비활성화하려면

1. Amazon Cognito 콘솔에 로그인하고 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. ID 제공업체를 찾습니다. 편집할 ID 제공업체를 선택합니다.
4. 액세스 제어를 위한 속성에서 편집을 선택합니다.
5. 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.
6. 변경 사항 저장(Save changes)을 선택합니다.

기본 공급자 매핑

다음 표에는 Amazon Cognito가 지원하는 인증 공급자에 대한 기본 매핑 정보가 나와 있습니다.

제공자	토큰 유형입니다.	보안 주체 태그 값	예제
Amazon Cognito 사용자 풀	ID 토큰	aud(클라이언트 ID) 및 sub(사용자 ID)	"6jk8ltokc7ac9es6jrtg9q572f", "57e7b692-4f66-480d-98b8-45a6729b4c88"
Facebook	액세스 토큰	aud(app_id), sub(user_id)	"492844718097981", "112177216992379"
Google	ID 토큰	aud(클라이언트 ID) 및 sub(사용자 ID)	"620493171733-eebk7c0hcp5lj3e1tlqp1gntt3k0rncv.apps.googleusercontent.com", "109220063452404746097"
SAML	어설션	"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier" , "http://schemas.xm	"auth0|5e28d196f8f55a0eaaa95de3", "user123@gmail.com"

제공자	토큰 유형입니다.	보안 주체 태그 값	예제
		lsoap.org/ws/2005/05/identity/claims/name"	
Apple	ID 토큰	aud(클라이언트 ID) 및 sub(사용자 ID)	"com.amazonaws.ec2-54-80-172-243.compute-1.client", "001968.a6ca34e9c1e742458a26cf8005854be9.0733"
Amazon	액세스 토큰	aud (Client ID on Amzn Dev Ac), user_id(user ID)	"amzn1.application-oa2-client.9d70d9382d3446108aaee3dd763a0fa6", "amzn1.account.AGHNIFJQMFSBG3G6XCPVB35ORQAA"
표준 OIDC 공급자	ID 및 액세스 토큰	aud (as client_id), sub (as user ID)	"620493171733-eebk7c0hcp5lj3e1tlqp1gntt3k0rncv.apps.googleusercontent.com", "109220063452404746097"
Twitter	액세스 토큰	aud (app ID; app Secret), sub (user ID)	"DfwifTtKEX1FiIBRnOTIR0CFK; Xgj5xb8xIrIVCPjXgLIdkW7fXmwcJJrFvnoK9gwZkLexo1y5z1", "1269003884292222976"
DevAuth	맵	해당 사항 없음	"tag1", "tag2"

> **Note**
>
> [보안 주체의 태그 키(Tag Key for Principal)] 및 [속성 이름(Attribute names)]에 기본 속성 매핑 옵션이 자동으로 채워집니다. 기본 매핑은 변경할 수 없습니다.

역할 기반 액세스 제어 사용

Amazon Cognito 자격 증명 풀은 인증된 사용자에게 리소스에 액세스할 수 있는 제한된 권한의 임시 자격 증명 세트를 할당합니다. AWS 각 사용자의 권한은 생성하는 IAM 역할을 통해 제어됩니다. 사용자의 ID 토큰에 있는 클레임에 따라 각 사용자의 역할을 선택하는 규칙을 정의할 수 있습니다. 인증된 사용자의 기본 역할을 정의할 수 있습니다. 인증되지 않은 게스트 사용자의 권한이 제한된 개별 IAM 역할을 정의할 수도 있습니다.

역할 매핑에 사용할 역할 생성

각 역할에 적합한 신뢰 정책을 추가하여 자격 증명 풀의 인증된 사용자에게만 Amazon Cognito에서 역할을 위임할 수 있게 해야 합니다. 다음은 신뢰 정책의 예제입니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Federated": "cognito-identity.amazonaws.com"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "cognito-identity.amazonaws.com:aud": "us-east-1:12345678-corner-
cafe-123456790ab"
        },
        "ForAnyValue:StringLike": {
          "cognito-identity.amazonaws.com:amr": "authenticated"
        }
      }
    }
  ]
```

```
}
```

이 정책에 의해 cognito-identity.amazonaws.com의 연동 사용자(OpenID Connect 토큰의 발급자)가 이 역할을 수임할 수 있습니다. 또한 이 정책은 토큰의 aud(이 경우 자격 증명 풀 ID)가 자격 증명 풀과 일치하도록 제한합니다. 마지막으로, 정책은 authenticated 값을 지닌 Amazon Cognito GetOpenIdToken API 작업에서 발행한 토큰의 다중 값 amr 클레임의 배열 멤버 중 하나를 지정합니다

역할 전달 권한 부여

사용자가 자격 증명 풀에 대한 사용자의 기존 권한을 초과하는 권한으로 역할을 설정하도록 허용하려면, set-identity-pool-roles API에 역할을 전달할 수 있는 iam:PassRole 권한을 해당 사용자에게 부여하세요. 예를 들어, 사용자가 Amazon S3에 쓸 수 없지만 자격 증명 풀에 설정한 IAM 역할이 Amazon S3에 대한 쓰기 권한을 부여하면 역할에 대해 iam:PassRole 권한이 부여된 경우에만 사용자가 이 역할을 설정할 수 있습니다. 다음 예제 정책은 iam:PassRole 권한을 허용하는 방법을 보여줍니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/myS3WriteAccessRole"
            ]
        }
    ]
}
```

이 정책 예제에서는 iam:PassRole 역할에 대해 myS3WriteAccessRole 권한이 부여됩니다. 역할은 역할의 Amazon 리소스 이름(ARN)을 사용하여 지정됩니다. 또한 사용자에게 이 정책을 연결해야 합니다. 자세한 내용은 관리형 정책 작업을 참조하세요.

> **Note**
>
> Lambda 함수는 리소스 기반 정책을 사용합니다. 이때 정책은 Lambda 함수 자체에 직접 연결됩니다. Lambda 함수를 호출하는 규칙을 생성하는 경우 역할은 전달하지 않습니다. 따라서 규칙을 생성하는 사용자에게 `iam:PassRole` 권한이 필요하지 않습니다. Lambda 함수 권한 부여에 대한 자세한 내용은 권한 관리: Lambda 함수 정책 사용을 참조하세요.

토큰을 사용하여 사용자에게 역할 할당

Amazon Cognito 사용자 풀을 통해 로그인하는 사용자의 경우 사용자 풀에서 할당한 ID 토큰에서 역할을 전달할 수 있습니다. ID 토큰의 다음 클레임에 역할이 표시됩니다.

- `cognito:preferred_role` 클레임은 역할 ARN입니다.
- `cognito:roles` 클레임은 허용된 역할 ARN 집합을 포함하는 쉼표로 구분된 문자열입니다.

클레임은 다음과 같이 설정됩니다.

- `cognito:preferred_role` 클레임은 최상위(최저) `Precedence` 값을 가진 그룹의 역할로 설정됩니다. 허용된 역할이 하나뿐인 경우 `cognito:preferred_role`이 해당 역할로 설정됩니다. 여러 역할이 있고 우선 순위가 가장 높은 역할 하나가 없는 경우 이 클레임이 설정되지 않습니다.
- 역할이 하나라도 있는 경우 `cognito:roles` 클레임이 설정됩니다.

토큰을 사용하여 역할을 할당할 때 사용자에게 할당할 수 있는 역할이 여러 개인 경우 Amazon Cognito 자격 증명 풀(페더레이션 자격 증명)은 다음과 같이 역할을 선택합니다.

- GetCredentialsForIdentity`CustomRoleArn`파라미터가 설정되어 있고 클레임의 역할과 일치하는 경우 파라미터를 사용하십시오. `cognito:roles` 이 파라미터가 `cognito:roles`의 역할과 일치하지 않으면 액세스를 거부합니다.
- `cognito:preferred_role` 클레임이 설정된 경우 이 클레임을 사용합니다.
- `cognito:preferred_role`클레임이 설정되지 않고 `cognito:roles` 클레임이 `CustomRoleArn` 설정되고 호출에 지정되지 않은 경우 콘솔 또는 `AmbiguousRoleResolution` 필드 (SetIdentityPoolRolesAPI `RoleMappings` 매개 변수) 의 역할 해결 설정을 사용하여 할당할 역할을 결정합니다. `GetCredentialsForIdentity`

규칙 기반 매핑을 사용하여 사용자에게 역할 할당

규칙을 사용하여 자격 증명 공급자 토큰의 클레임을 IAM 역할에 매핑할 수 있습니다.

각 규칙은 토큰 클레임(예: Amazon Cognito 사용자 풀의 ID 토큰에 있는 사용자 속성), 일치 유형, 값 및 IAM 역할을 지정합니다. 이때 일치 유형은 `Equals`, `NotEqual`, `StartsWith` 또는 `Contains`가 될 수 있습니다. 사용자가 클레임에 대해 일치하는 값을 가진 경우 자격 증명을 얻으면 해당 역할을 수임할 수 있습니다. 예를 들어, `Sales`의 `custom:dept` 사용자 지정 속성 값으로 사용자의 특정한 IAM 역할을 할당하는 규칙을 만들 수 있습니다.

> **Note**
>
> 규칙 설정에서 표준 속성을 구별하기 위해 사용자 지정 속성에 `custom:` 접두사가 필요합니다.

규칙은 순서대로 평가되며 순서를 무시하도록 `CustomRoleArn`이 지정되지 않으면 첫 번째 일치 규칙의 IAM 역할이 사용됩니다. Amazon Cognito 사용자 풀의 사용자 속성에 대한 자세한 내용은 사용자 풀 속성 섹션을 참조하세요.

자격 증명 풀(연동 자격 증명) 콘솔에서 인증 공급자에 대한 여러 규칙을 설정할 수 있습니다. 규칙은 순서대로 적용됩니다. 규칙을 드래그하여 순서를 변경할 수 있습니다. 첫 번째 일치 규칙이 우선합니다. 일치 유형이 `NotEqual`이고 클레임이 없으면 규칙이 평가되지 않습니다. 일치하는 규칙이 없으면 역할 해결 설정이 기본 인증된 역할 사용 또는 거부에 적용됩니다.

API 및 CLI에서는 API의 `RoleMappings` 파라미터에 지정된 RoleMapping유형의 `AmbiguousRoleResolution` 필드에 일치하는 규칙이 없을 때 할당할 역할을 지정할 수 있습니다. SetIdentityPoolRoles

해당 유형의 필드를 사용하여 AWS CLI 또는 API에서 OpenID Connect (OIDC) 및 SAML ID 공급자에 대한 규칙 기반 매핑을 설정할 수 있습니다. `RulesConfiguration` RoleMapping API의 파라미터에 이 필드를 지정할 수 있습니다. `RoleMappings` SetIdentityPoolRoles AWS Management Console 현재 에서는 OIDC 또는 SAML 공급자에 대한 규칙을 추가할 수 없습니다.

예를 들어 다음 AWS CLI 명령은 OIDC IdP로 인증된 `arn:aws:iam::123456789012:role/Sacramento_team_S3_admin` Sacramento 위치의 사용자에게 역할을 할당하는 규칙을 추가합니다. `arn:aws:iam::123456789012:oidc-provider/myOIDCIdP`

```
aws cognito-identity set-identity-pool-roles --region us-east-1 --cli-input-json
 file://role-mapping.json
```

role-mapping.json의 콘텐츠:

```
{
    "IdentityPoolId": "us-east-1:12345678-corner-cafe-123456790ab",
    "Roles": {
        "authenticated": "arn:aws:iam::123456789012:role/myS3WriteAccessRole",
        "unauthenticated": "arn:aws:iam::123456789012:role/myS3ReadAccessRole"
    },
    "RoleMappings": {
        "arn:aws:iam::123456789012:oidc-provider/myOIDCIdP": {
            "Type": "Rules",
            "AmbiguousRoleResolution": "AuthenticatedRole",
            "RulesConfiguration": {
                "Rules": [
                    {
                        "Claim": "locale",
                        "MatchType": "Equals",
                        "Value": "Sacramento",
                        "RoleARN": "arn:aws:iam::123456789012:role/
Sacramento_team_S3_admin"
                    }
                ]
            }
        }
    }
}
```

자격 증명 풀에 구성된 각 사용자 풀 또는 기타 인증 공급자에 대해 규칙을 25개까지 생성할 수 있습니다. 이 제한은 조정할 수 없습니다. 자세한 내용은 Amazon Cognito의 할당량을 참조하세요.

규칙 기반 매핑에 사용할 토큰 클레임

Amazon Cognito

Amazon Cognito ID 토큰은 JWT(JSON Web Token)로 표시됩니다. `name`, `family_name` 및 `phone_number`와 같은 인증된 사용자의 자격 증명에 대한 클레임이 이 토큰에 포함됩니다. 표준 클레임에 대한 자세한 내용은 OpenID Connect 사양을 참조하세요. 다음은 표준 클레임 외에 Amazon Cognito와 관련된 추가 클레임입니다.

- `cognito:groups`
- `cognito:roles`
- `cognito:preferred_role`

Amazon

다음 클레임은 클레임의 가능한 값과 함께 Login with Amazon에서 사용할 수 있습니다.

- `iss`: www.amazon.com
- `aud`: 앱 ID
- `sub`: `sub` Login with Amazon 토큰

Facebook

다음 클레임은 클레임의 가능한 값과 함께 Facebook에서 사용할 수 있습니다.

- `iss`: graph.facebook.com
- `aud`: 앱 ID
- `sub`: `sub` Facebook 토큰

Google

Google 토큰에는 OpenID Connect 사양의 표준 클레임이 포함됩니다. OpenID 토큰의 모든 클레임은 규칙 기반 매핑에 사용할 수 있습니다. Google 토큰에서 사용할 수 있는 클레임에 대한 자세한 내용은 OpenID Connect를 참조하세요.

Apple

Apple 토큰에는 OpenID Connect 사양의 표준 클레임이 포함됩니다. Apple 토큰에서 사용할 수 있는 클레임에 대한 자세한 내용은 Apple 설명서에서 Sign in with Apple로 사용자 인증을 참조하세요. Apple의 토큰에 항상 `email`이 포함되어 있지는 않습니다.

OpenID

Open ID 토큰의 모든 클레임은 규칙 기반 매핑에 사용할 수 있습니다. 표준 클레임에 대한 자세한 내용은 OpenID Connect 사양을 참조하세요. 사용 가능한 추가 클레임에 대한 내용은 OpenID 제공업체 설명서를 참조하세요.

SAML

수신된 SAML 어설션에서 클레임을 구문 분석합니다. SAML 어설션에서 사용할 수 있는 모든 클레임은 규칙 기반 매핑에서 사용할 수 있습니다.

역할 기반 액세스 제어의 모범 사례

> **Important**
>
> 역할에 매핑하는 클레임을 최종 사용자가 수정할 수 있는 경우 최종 사용자가 역할을 맡고 그에 따라 정책을 설정할 수 있습니다. 최종 사용자가 직접 설정할 수 없는 클레임은 승격된 권한을 가진 역할에만 매핑하세요. Amazon Cognito 사용자 풀에서 각 사용자 속성에 대해 앱당 읽기 및 쓰기 권한을 설정할 수 있습니다.

> **Important**
>
> Amazon Cognito 사용자 풀에서 그룹의 역할을 설정하면 이 역할이 사용자의 ID 토큰을 통해 전달됩니다. 이 역할을 사용하려면 자격 증명 풀의 인증된 역할 선택에 대해 토큰으로부터 역할 선택을 설정해야 합니다.
> 콘솔의 역할 확인 설정과 SetIdentityPoolRolesAPI의 `RoleMappings` 매개변수를 사용하여 토큰에서 올바른 역할을 결정할 수 없는 경우의 기본 동작을 지정할 수 있습니다.

자격 증명 얻기

Amazon Cognito를 사용하여 제한된 권한의 임시 자격 증명을 애플리케이션에 제공하여 사용자가 리소스에 액세스할 수 있도록 할 수 있습니다. AWS 이 섹션에서는 자격 증명을 가져오는 방법과 자격 증명 풀에서 Amazon Cognito 자격 증명을 가져오는 방법을 설명합니다.

Amazon Cognito는 인증된 자격 증명과 인증되지 않은 자격 증명을 모두 지원합니다. 미인증 사용자는 자격 증명이 인증되지 않았으므로 앱 혹은 자격 증명의 인증이 중요하지 않은 경우 게스트 사용자 역할에 적합합니다. 인증받은 사용자는 타사 자격 증명 공급자(IdP)를 통해, 또는 자격 증명을 인증받은 사용자 풀을 통해 애플리케이션에 로그인합니다. 미인증 사용자의 액세스 권한을 허용하지 않도록 리소스 권한 범위를 충분히 정했는지 확인하세요.

Amazon Cognito 자격 증명(identity)은 자격 증명(credential)이 아닙니다. () 의 웹 자격 증명 연동 지원을 사용하여 자격 증명으로 교환됩니다. AWS Security Token Service AWS STS앱 사용자를 위해 AWS 자격 증명을 얻는 권장 방법은 `AWS.CognitoIdentityCredentials`를 사용하는 것입니다. 그런 다음 를 사용하여 자격 증명 개체의 ID를 자격 증명으로 교환합니다. AWS STS

> **Note**
>
> 2015년 2월 이전에 자격 증명 풀을 만든 경우 파라미터 역할 없이 `AWS.CognitoIdentityCredentials` 생성자를 사용하려면 역할을 자격 증명 풀과 다시 연결해야 합니다. 이렇게 하려면 Amazon Cognito 콘솔을 열고, 자격 증명 풀 관리(Manage Identity Pools)를 선택한 다음, 자격 증명 풀을 선택합니다. 자격 증명 풀 편집(Edit Identity Pool)을 선택하고, 인증된 역할과 인증되지 않은 역할을 지정한 다음, 변경 사항을 저장합니다.

웹 자격 증명 보안 인증 공급자는 AWS SDK의 기본 보안 인증 공급자 체인의 일부입니다. AWS SDK 또는 의 로컬 `config` 파일에 자격 증명 풀 토큰을 설정하려면 `web_identity_token_file` 프로필 항목을 추가하십시오. AWS CLI AWS SDK 및 도구 참조 가이드의 역할 자격 증명 공급자 위임을 참조하십시오.

SDK에 웹 자격 증명 보안 인증을 채우는 방법에 대한 자세한 내용은 SDK 개발자 안내서를 참조하세요. 최상의 결과를 얻으려면 기본 제공되는 자격 증명 풀 통합으로 프로젝트를 시작하세요. AWS Amplify

AWS 자격 증명 풀을 사용하여 자격 증명을 가져오고 설정하는 데 사용할 수 있는 SDK 리소스

- Amplify 개발자 센터의 자격 증명 풀 페더레이션(Android)
- Amplify 개발자 센터의 자격 증명 풀 페더레이션(iOS)
- 개발자 안내서에서 Amazon Cognito ID를 사용하여 사용자 인증하기 AWS SDK for JavaScript
- 개발자 안내서의 Amazon Cognito 자격 증명 공급자 AWS SDK for .NET
- 개발자 안내서에서 프로그래밍 방식으로 자격 증명을 지정하십시오. AWS SDK for Go
- AWS SDK for Java 2.x 개발자 안내서의 코드에 임시 자격 증명을 입력하십시오.
- assumeRoleWithWebIdentityCredentialProvider AWS SDK for PHP 개발자 안내서의 제공자
- AWS SDK for Python (Boto3) 설명서의 웹 ID 제공업체를 사용하여 역할 수임
- AWS SDK for Rust 개발자 안내서에 자격 증명 및 기본 지역 지정하기

다음 섹션에서는 일부 레거시 AWS SDK의 예제 코드를 제공합니다.

Android

Amazon Cognito를 사용하여 제한된 권한의 임시 자격 증명을 애플리케이션에 제공하여 사용자가 리소스에 액세스할 수 있도록 할 수 있습니다. AWS Amazon Cognito는 인증된 자격 증명과 인증되지 않은 자격 증명을 모두 지원합니다. 앱에 AWS 자격 증명을 제공하려면 아래 단계를 따르십시오.

안드로이드 앱에서 Amazon Cognito 자격 증명 풀을 사용하려면 설정해야 합니다. AWS Amplify자세한 내용은 Amplify 개발자 센터의 인증을 참조하세요.

Amazon Cognito 자격 증명 검색

인증되지 않은 사용자를 허용하는 경우 최종 사용자의 고유한 Amazon Cognito 식별자(자격 증명 ID)를 즉시 검색할 수 있습니다. 사용자를 인증하는 경우 자격 증명 공급자에서 로그인 토큰을 설정한 후 자격 증명 ID를 검색할 수 있습니다.

```
String identityId = credentialsProvider.getIdentityId();
Log.d("LogTag", "my ID is " + identityId);
```

> **Note**
>
> 애플리케이션의 기본 스레드에서 `getIdentityId()`, `refresh()` 또는 `getCredentials()`를 호출하지 마세요. Android 3.0 (API 레벨 11) 부터 기본 애플리케이션 스레드에서 네트워크 I/O를 NetworkOnMainThreadException수행하면 앱이 자동으로 실패하고 a가 발생합니다. `AsyncTask`를 사용하여 코드를 백그라운드 스레드로 이동해야 합니다. 자세한 내용은 Android 설명서를 참조하세요. 이미 로컬에 캐시된 경우에만 `getCachedIdentityId()`를 호출하여 ID를 검색할 수 있습니다. 그렇지 않으면 메서드가 null을 반환합니다.

iOS - Objective-C

Amazon Cognito를 사용하여 제한된 권한의 임시 자격 증명을 애플리케이션에 제공하여 사용자가 리소스에 액세스할 수 있도록 할 수 있습니다. AWS Amazon Cognito 자격 증명 풀은 인증된 자격 증명과 인증되지 않은 자격 증명을 모두 지원합니다. 앱에 AWS 자격 증명을 제공하려면 다음 단계를 완료하십시오.

iOS 앱에서 Amazon Cognito 자격 증명 풀을 사용하려면 설정해야 합니다. AWS Amplify자세한 내용은 Amplify 개발자 센터의 Swift 인증 및 Flutter 인증을 참조하세요.

Amazon Cognito 자격 증명 검색

인증되지 않은 사용자를 허용하는 경우 또는 사용자를 인증하는 경우 자격 증명 공급자에서 로그인 토큰을 설정한 후에 최종 사용자의 Amazon Cognito 식별자(자격 증명 ID)를 즉시 검색할 수 있습니다.

```
// Retrieve your Amazon Cognito ID
```

```
[[credentialsProvider getIdentityId] continueWithBlock:^id(AWSTask *task) {
    if (task.error) {
        NSLog(@"Error: %@", task.error);
    }
    else {
        // the task result will contain the identity id
        NSString *cognitoId = task.result;
    }
    return nil;
}];
```

> **Note**
>
> `getIdentityId`는 비동기식 호출입니다. 자격 증명 ID가 이미 공급자에 설정된 경우 `credentialsProvider.identityId`를 호출하여 해당 자격 증명을 검색할 수 있으며 이 자격 증명은 로컬에 캐시됩니다. 그러나 자격 증명 ID가 공급자에 설정되지 않은 경우 `credentialsProvider.identityId`를 호출하면 `nil`이 반환됩니다. 자세한 내용은 Amplify iOS SDK 참조를 참조하세요.

iOS - Swift

Amazon Cognito를 사용하여 제한된 권한의 임시 자격 증명을 애플리케이션에 제공하여 사용자가 리소스에 액세스할 수 있도록 할 수 있습니다. AWS Amazon Cognito는 인증된 자격 증명과 인증되지 않은 자격 증명을 모두 지원합니다. 앱에 AWS 자격 증명을 제공하려면 아래 단계를 따르십시오.

iOS 앱에서 Amazon Cognito 자격 증명 풀을 사용하려면 설정해야 합니다. AWS Amplify자세한 내용은 Amplify 개발자 센터의 Swift 인증을 참조하세요.

Amazon Cognito 자격 증명 검색

인증되지 않은 사용자를 허용하는 경우 또는 사용자를 인증하는 경우 자격 증명 공급자에서 로그인 토큰을 설정한 후에 최종 사용자의 Amazon Cognito 식별자(자격 증명 ID)를 즉시 검색할 수 있습니다.

```
// Retrieve your Amazon Cognito ID
credentialsProvider.getIdentityId().continueWith(block: { (task) -> AnyObject? in
    if (task.error != nil) {
        print("Error: " + task.error!.localizedDescription)
    }
    else {
```

```
        // the task result will contain the identity id
        let cognitoId = task.result!
        print("Cognito id: \(cognitoId)")
    }
    return task;
})
```

> **Note**
>
> `getIdentityId`는 비동기식 호출입니다. 자격 증명 ID가 이미 공급자에 설정된 경우 `credentialsProvider.identityId`를 호출하여 해당 자격 증명을 검색할 수 있으며 이 자격 증명은 로컬에 캐시됩니다. 그러나 자격 증명 ID가 공급자에 설정되지 않은 경우 `credentialsProvider.identityId`를 호출하면 `nil`이 반환됩니다. 자세한 내용은 Amplify iOS SDK 참조를 참조하세요.

JavaScript

자격 증명 풀을 아직 만들지 않았다면 `AWS.CognitoIdentityCredentials`를 사용하기 전에 Amazon Cognito 콘솔에서 자격 증명 풀을 만듭니다.

자격 증명 제공자로 자격 증명 풀을 구성한 후에는 `AWS.CognitoIdentityCredentials`를 사용하여 사용자를 인증할 수 있습니다. `AWS.CognitoIdentityCredentials`를 사용하도록 애플리케이션 자격 증명을 구성하려면, `credentials` 또는 서비스당 구성의 `AWS.Config` 속성을 설정하세요. 다음 예에는 `AWS.Config`가 사용됩니다.

```
// Set the region where your identity pool exists (us-east-1, eu-west-1)
AWS.config.region = 'us-east-1';

// Configure the credentials provider to use your identity pool
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
    IdentityPoolId: 'IDENTITY_POOL_ID',
    Logins: { // optional tokens, used for authenticated login
        'graph.facebook.com': 'FBTOKEN',
        'www.amazon.com': 'AMAZONTOKEN',
        'accounts.google.com': 'GOOGLETOKEN',
        'appleid.apple.com': 'APPLETOKEN'
    }
});

// Make the call to obtain credentials
```

```
AWS.config.credentials.get(function(){

    // Credentials will be available when this function is called.
    var accessKeyId = AWS.config.credentials.accessKeyId;
    var secretAccessKey = AWS.config.credentials.secretAccessKey;
    var sessionToken = AWS.config.credentials.sessionToken;

});
```

선택 사항인 Logins 속성은 공급자의 자격 증명 토큰에 대한 자격 증명 공급자 이름의 맵입니다. 자격 증명 공급자에게서 토큰을 받는 방법은 어떤 공급자를 사용하느냐에 따라 다릅니다. 예를 들어 Facebook이 자격 증명 공급자 중 하나인 경우 Facebook SDK의 FB.login 함수를 사용하여 자격 증명 공급자 토큰을 얻을 수 있습니다.

```
FB.login(function (response) {
    if (response.authResponse) { // logged in
        AWS.config.credentials = new AWS.CognitoIdentityCredentials({
          IdentityPoolId: 'us-east-1:1699ebc0-7900-4099-b910-2df94f52a030',
          Logins: {
            'graph.facebook.com': response.authResponse.accessToken
          }
        });

        console.log('You are now logged in.');
    } else {
        console.log('There was a problem logging you in.');
    }
});
```

Amazon Cognito 자격 증명 검색

인증되지 않은 사용자를 허용하는 경우 또는 사용자를 인증하는 경우 자격 증명 공급자에서 로그인 토큰을 설정한 후에 최종 사용자의 Amazon Cognito 식별자(자격 증명 ID)를 즉시 검색할 수 있습니다.

```
var identityId = AWS.config.credentials.identityId;
```

Unity

Amazon Cognito를 사용하여 제한된 권한의 임시 자격 증명을 애플리케이션에 제공하여 사용자가 리소스에 액세스할 수 있도록 할 수 있습니다. AWS Amazon Cognito는 인증된 자격 증명과 인증되지 않은 자격 증명을 모두 지원합니다. 앱에 AWS 자격 증명을 제공하려면 아래 단계를 따르십시오.

AWS SDK for Unity는 이제 AWS SDK for .NET의 일부입니다. 에서 Amazon Cognito를 시작하려면 개발자 안내서의 AWS SDK for .NETAmazon Cognito 자격 증명 공급자를 참조하십시오. AWS SDK for .NET 또는 Amplify 개발자 센터에서 앱을 빌드하기 위한 옵션을 참조하십시오. AWS Amplify

Amazon Cognito 자격 증명 검색

인증되지 않은 사용자를 허용하는 경우 또는 사용자를 인증하는 경우 자격 증명 공급자에서 로그인 토큰을 설정한 후에 최종 사용자의 Amazon Cognito 식별자(자격 증명 ID)를 즉시 검색할 수 있습니다.

```
credentials.GetIdentityIdAsync(delegate(AmazonCognitoIdentityResult<string> result) {
    if (result.Exception != null) {
        //Exception!
    }
    string identityId = result.Response;
});
```

Xamarin

Amazon Cognito를 사용하여 제한된 권한의 임시 자격 증명을 애플리케이션에 제공하여 사용자가 리소스에 액세스할 수 있도록 할 수 있습니다. AWS Amazon Cognito는 인증된 자격 증명과 인증되지 않은 자격 증명을 모두 지원합니다. 앱에 AWS 자격 증명을 제공하려면 아래 단계를 따르십시오.

AWS SDK for Xamarin은 이제 AWS SDK for .NET의 일부입니다. 에서 Amazon Cognito를 시작하려면 개발자 안내서의 AWS SDK for .NETAmazon Cognito 자격 증명 공급자를 참조하십시오. AWS SDK for .NET 또는 Amplify 개발자 센터에서 앱을 빌드하기 위한 옵션을 참조하십시오. AWS Amplify

> **Note**
>
> 참고: 2015년 2월 이전에 자격 증명 풀을 만든 경우 파라미터 역할 없이 이 생성자를 사용하려면 역할을 작업 증명 풀과 다시 연결해야 합니다. 이렇게 하려면 Amazon Cognito 콘솔을 열고, 자격 증명 풀 관리(Manage Identity Pools)를 선택한 다음, 자격 증명 풀을 선택합니다. 자격 증명 풀 편집(Edit Identity Pool)을 선택하고, 인증된 역할과 인증되지 않은 역할을 지정한 다음, 변경 사항을 저장합니다.

Amazon Cognito 자격 증명 검색

인증되지 않은 사용자를 허용하는 경우 또는 사용자를 인증하는 경우 자격 증명 공급자에서 로그인 토큰을 설정한 후에 최종 사용자의 Amazon Cognito 식별자(자격 증명 ID)를 즉시 검색할 수 있습니다.

```
var identityId = await credentials.GetIdentityIdAsync();
```

서비스 액세스 AWS

Amazon Cognito 자격 증명 공급자를 구성하고 자격 증명을 AWS 검색한 AWS 서비스 후 클라이언트를 생성할 수 있습니다.

AWS 클라이언트 생성을 위한 SDK 리소스

- AWS AWS SDK for C++ 개발자 안내서의 클라이언트 구성
- AWS SDK for Go 개발자 AWS SDK for Go 안내서와 함께 AWS 서비스 V2 사용
- AWS SDK for Java 2.x 개발자 안내서의 HTTP 클라이언트 구성
- AWS SDK for JavaScript 개발자 안내서의 서비스 개체 생성 및 호출
- AWS SDK for Python (Boto3) 설명서에서 클라이언트 만들기
- AWS SDK for Rust 개발자 안내서에서 서비스 클라이언트 만들기
- AWS SDK for Swift 개발자 안내서의 클라이언트 사용

다음 조각은 Amazon DynamoDB 클라이언트를 초기화합니다.

Android

안드로이드 앱에서 Amazon Cognito 자격 증명 풀을 사용하려면 설정해야 합니다. AWS Amplify자세한 내용은 Amplify 개발자 센터의 인증을 참조하세요.

```
// Create a service client with the provider
AmazonDynamoDB client = new AmazonDynamoDBClient(credentialsProvider);
```

자격 증명 공급자는 Amazon Cognito와 통신하여 인증된 사용자 및 인증되지 않은 사용자의 고유 식별자와 Mobile SDK에 대한 제한된 AWS 권한의 임시 자격 증명을 모두 검색합니다. AWS 검색된 자격 증명은 한시간 동안 유효하며 만료된 경우 공급자는 자격 증명을 새로 고칩니다.

iOS - Objective-C

iOS 앱에서 Amazon Cognito 자격 증명 풀을 사용하려면 설정해야 합니다. AWS Amplify자세한 내용은 Amplify 개발자 센터의 Swift 인증 및 Flutter 인증을 참조하세요.

```
// create a configuration that uses the provider
```

```
AWSServiceConfiguration *configuration = [AWSServiceConfiguration
 configurationWithRegion:AWSRegionUSEast1 provider:credentialsProvider];
// get a client with the default service configuration
AWSDynamoDB *dynamoDB = [AWSDynamoDB defaultDynamoDB];
```

자격 증명 공급자는 Amazon Cognito와 통신하여 인증된 사용자 및 인증되지 않은 사용자의 고유 식별자와 Mobile SDK에 대한 제한된 AWS 권한의 임시 자격 증명을 모두 검색합니다. AWS 검색된 자격 증명은 한시간 동안 유효하며 만료된 경우 공급자는 자격 증명을 새로 고칩니다.

iOS - Swift

iOS 앱에서 Amazon Cognito 자격 증명 풀을 사용하려면 설정해야 합니다. AWS Amplify자세한 내용은 Amplify 개발자 센터의 Swift 인증을 참조하세요.

```
// get a client with the default service configuration
let dynamoDB = AWSDynamoDB.default()

// get a client with a custom configuration
AWSDynamoDB.register(with: configuration!, forKey: "USWest2DynamoDB");
let dynamoDBCustom = AWSDynamoDB(forKey: "USWest2DynamoDB")
```

자격 증명 공급자는 Amazon Cognito와 통신하여 인증된 사용자 및 인증되지 않은 사용자의 고유 식별자와 Mobile SDK에 대한 제한된 AWS 권한의 임시 자격 증명을 모두 검색합니다. AWS 검색된 자격 증명은 한시간 동안 유효하며 만료된 경우 공급자는 자격 증명을 새로 고칩니다.

JavaScript

```
// Create a service client with the provider
var dynamodb = new AWS.DynamoDB({region: 'us-west-2'});
```

자격 증명 공급자는 Amazon Cognito와 통신하여 인증된 사용자 및 인증되지 않은 사용자의 고유 식별자와 Mobile SDK에 대한 권한이 제한된 임시 자격 증명을 모두 검색합니다. AWS AWS 검색된 자격 증명은 한시간 동안 유효하며 만료된 경우 공급자는 자격 증명을 새로 고칩니다.

Unity

AWS SDK for Unity는 이제 AWS SDK for .NET의 일부입니다. 에서 Amazon Cognito를 시작하려면 개발자 안내서의 AWS SDK for .NETAmazon Cognito 자격 증명 공급자를 참조하십시오. AWS SDK for .NET 또는 Amplify 개발자 센터에서 앱을 빌드하기 위한 옵션을 참조하십시오. AWS Amplify

```
// create a service client that uses credentials provided by Cognito
AmazonDynamoDBClient client = new AmazonDynamoDBClient(credentials, REGION);
```

자격 증명 공급자는 Amazon Cognito와 통신하여 인증된 사용자 및 인증되지 않은 사용자의 고유 식별자와 Mobile SDK에 대한 권한이 제한된 임시 자격 증명을 모두 검색합니다. AWS AWS 검색된 자격 증명은 한시간 동안 유효하며 만료된 경우 공급자는 자격 증명을 새로 고칩니다.

Xamarin

AWS SDK for Xamarin은 이제 AWS SDK for .NET의 일부입니다. 에서 Amazon Cognito를 시작하려면 개발자 안내서의 AWS SDK for .NETAmazon Cognito 자격 증명 공급자를 참조하십시오. AWS SDK for .NET 또는 Amplify 개발자 센터에서 앱을 빌드하기 위한 옵션을 참조하십시오. AWS Amplify

```
// create a service client that uses credentials provided by Cognito
var client = new AmazonDynamoDBClient(credentials, REGION)
```

자격 증명 공급자는 Amazon Cognito와 통신하여 인증된 사용자 및 인증되지 않은 사용자의 고유 식별자와 Mobile SDK에 대한 권한이 제한된 임시 자격 증명을 모두 검색합니다. AWS AWS 검색된 자격 증명은 한시간 동안 유효하며 만료된 경우 공급자는 자격 증명을 새로 고칩니다.

자격 증명 풀 외부 자격 증명 공급자

`logins` 속성을 사용하여 자격 증명 공급자(IdP)에게서 받은 자격 증명을 설정할 수 있습니다. 또한 자격 증명 풀을 여러 개의 IdPs 자격 증명 풀과 연결할 수 있습니다. 예를 들어 고유한 Amazon Cognito 자격 증명을 두 IdP 로그인 모두와 연결하도록 `logins` 속성에 Facebook 토큰과 Google 토큰을 둘 다 설정할 수 있습니다. 사용자는 어느 계정으로도 인증할 수 있지만 Amazon Cognito는 동일한 사용자 식별자를 반환합니다.

다음 지침은 Amazon Cognito 자격 증명 풀이 IdPs 지원하는 인증 절차를 안내합니다.

주제

- 페이스북을 아이덴티티 풀 IdP로 설정하기
- Amazon으로 로그인을 자격 증명 풀 IdP로 설정
- 구글을 아이덴티티 풀 IdP로 설정하기
- Apple을 자격 증명 풀 IdP로 사용하여 로그인 설정하기
- OIDC 공급자를 자격 증명 풀 IdP로 설정
- SAML 공급자를 자격 증명 풀 IdP로 설정

페이스북을 아이덴티티 풀 IdP로 설정하기

Amazon Cognito 자격 증명 풀은 Facebook과 통합되어 모바일 애플리케이션 사용자를 위해 페더레이션된 인증을 제공합니다. 이 섹션에서는 Facebook을 IdP로 사용하여 애플리케이션을 등록하고 설정하는 방법을 설명합니다.

Facebook 설정

Facebook 사용자 인증 및 Facebook API와의 상호 작용 이전에 Facebook을 통해 애플리케이션을 등록합니다.

Facebook 개발자 포털을 통해 애플리케이션을 설정할 수 있습니다. Amazon Cognito 자격 증명 풀에 Facebook을 통합하기 전에 이 절차를 수행합니다.

Facebook 설정

1. Facebook 개발자 포털에서 Facebook 자격 증명으로 로그인합니다.
2. 앱 메뉴에서 새 앱 추가를 선택합니다.
3. 플랫폼을 선택하고 빠른 시작 프로세스를 완료합니다.

Android

Android 앱을 Facebook 로그인과 통합하는 방법에 대한 자세한 내용은 Facebook Getting Started Guide(Facebook 시작 안내서)를 참조하세요.

iOS - Objective-C

iOS Objective-C 앱을 Facebook 로그인과 통합하는 방법에 대한 자세한 내용은 Facebook Getting Started Guide(Facebook 시작 안내서)를 참조하세요.

iOS - Swift

iOS Swift 앱을 Facebook 로그인과 통합하는 방법에 대한 자세한 내용은 Facebook Getting Started Guide(Facebook 시작 안내서)를 참조하세요.

JavaScript

JavaScript 웹 앱을 Facebook 로그인과 통합하는 방법에 대한 자세한 내용은 Facebook 시작 안내서를 참조하십시오.

Unity

Unity 앱을 Facebook 로그인과 통합하는 방법에 대한 자세한 내용은 Facebook Getting Started Guide(Facebook 시작 안내서)를 참조하세요.

Xamarin

Facebook 인증을 추가하려면 먼저 아래의 적절한 흐름에 따라 Facebook SDK를 애플리케이션에 통합합니다. Amazon Cognito 자격 증명 풀은 Facebook 액세스 토큰을 사용하여 Amazon Cognito 자격 증명에 연결된 고유한 사용자 식별자를 생성합니다.

- Xamarin이 제공하는 Facebook iOS SDK
- Xamarin이 제공하는 Facebook Android SDK

Amazon Cognito 자격 증명 풀 콘솔에서 ID 제공업체 구성

다음 절차를 사용하여 ID 제공업체를 구성합니다.

Facebook ID 제공업체(idP)를 추가하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. ID 제공업체 추가를 선택합니다.
4. Facebook을 선택합니다.
5. 개발자용 Meta에서 생성한 OAuth 프로젝트의 앱 ID를 입력합니다. 자세한 내용은 개발자용 Meta 문서의 Facebook 로그인을 참조하세요.
6. Amazon Cognito가 이 공급자를 통해 인증된 사용자에게 보안 인증을 발급할 때 요청하는 역할을 설정하려면 역할 설정을 구성하세요.

    - 인증된 역할을 구성할 때 설정한 기본 역할을 이 IdP의 사용자에게 할당하거나 규칙을 사용하여 역할 선택을 선택할 수 있습니다.

        i. 규칙을 사용하여 역할 선택을 선택한 경우 사용자 인증의 소스 클레임, 클레임 비교 기준으로 사용할 운영자, 이 역할 선택과 일치하도록 하는 값, 역할 할당이 일치할 때 할당할 역할을 입력합니다. 다른 조건에 따라 추가 규칙을 생성하려면 다른 항목 추가를 선택합니다.

        ii. 역할 해결을 선택합니다. 사용자의 클레임이 규칙과 일치하지 않는 경우 보안 인증을 거부하거나 인증된 역할의 보안 인증을 발급할 수 있습니다.

7. Amazon Cognito가 이 공급자를 통해 인증한 사용자에게 보안 인증을 발급할 때 할당하는 보안 주체 태그를 변경하려면 액세스 제어를 위한 속성을 구성합니다.

   a. 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.

   b. sub 및 aud 클레임 기반 보안 주체 태그를 적용하려면 기본 매핑 사용을 선택합니다.

   c. 보안 주체 태그에 대한 속성의 자체 사용자 지정 스키마를 생성하려면 사용자 지정 매핑 사용을 선택합니다. 그런 다음 태그에 표시하려는 각 클레임에서 소싱하려는 태그 키를 입력합니다.

8. 변경 사항 저장(Save changes)을 선택합니다.

Facebook 사용

Android

Facebook 인증을 추가하려면 먼저 Facebook 안내서를 따라 Facebook SDK를 애플리케이션에 통합합니다. 그런 다음 Facebook으로 로그인(Login with Facebook) 버튼을 Android 사용자 인터페이스에 추가합니다. Facebook SDK는 세션 객체를 사용하여 상태를 추적합니다. Amazon Cognito는 이 세션 객체의 액세스 토큰을 사용하여 사용자를 인증하고, 고유 식별자를 생성하고, 필요한 경우 사용자에게 다른 리소스에 대한 액세스 권한을 부여합니다. AWS

Facebook SDK로 사용자를 인증한 후에는 Amazon Cognito 자격 증명 공급자에 세션 토큰을 추가합니다.

Facebook SDK 4.0 이상:

```
Map<String, String> logins = new HashMap<String, String>();
logins.put("graph.facebook.com", AccessToken.getCurrentAccessToken().getToken());
credentialsProvider.setLogins(logins);
```

Facebook SDK 4.0 이전:

```
Map<String, String> logins = new HashMap<String, String>();
logins.put("graph.facebook.com", Session.getActiveSession().getAccessToken());
credentialsProvider.setLogins(logins);
```

Facebook 로그인 프로세스에서 해당 SDK의 싱글톤 세션을 초기화합니다. Facebook 세션 객체에는 Amazon Cognito가 인증된 최종 사용자의 자격 증명을 AWS 생성하는 데 사용하는 OAuth 토큰이 포함되어 있습니다. 또한 Amazon Cognito는 사용자 데이터베이스에 이 특정 Facebook 자격 증명과 일치하는 사용자가 있는지 확인하는 데에도 이 토큰을 사용합니다. 사용자가 이미 있으면 API가 기존의 식

별자를 반환합니다. 그렇지 않으면 API에서 새 식별자를 반환합니다. 클라이언트 SDK가 로컬 디바이스에서 식별자를 자동으로 캐시합니다.

> **Note**
>
> 로그인 맵을 설정한 후에는 `refresh` 또는 `get` 를 호출하여 자격 증명을 검색하십시오. AWS

iOS - Objective-C

Facebook 인증을 추가하려면 먼저 Facebook 안내서를 따라 Facebook SDK를 애플리케이션에 통합합니다. 그런 다음 Facebook으로 로그인 버튼을 귀하의 사용자 인터페이스에 추가합니다. Facebook SDK는 세션 객체를 사용하여 상태를 추적합니다. Amazon Cognito는 이 세션 객체의 액세스 토큰을 사용하여 사용자를 인증하고 고유한 Amazon Cognito 자격 증명 풀(페더레이션 자격 증명)에 바인딩합니다.

Amazon Cognito에 Facebook 액세스 토큰을 제공하려면 AWSIdentityProviderManager 프로토콜을 구현합니다.

`logins` 메서드를 구현할 때 `AWSIdentityProviderFacebook`이 포함된 사전을 반환합니다. 다음 코드 예와 같이 이 사전은 키 역할을 하고 인증된 Facebook 사용자에게서 받은 현재 액세스 토큰은 값 역할을 합니다.

```
- (AWSTask<NSDictionary<NSString *, NSString *> *> *)logins {
    FBSDKAccessToken* fbToken = [FBSDKAccessToken currentAccessToken];
    if(fbToken){
        NSString *token = fbToken.tokenString;
        return [AWSTask taskWithResult: @{ AWSIdentityProviderFacebook : token }];
    }else{
            return [AWSTask taskWithError:[NSError errorWithDomain:@"Facebook Login"
                                                              code:-1
                                                          userInfo:@{@"error":@"No current
 Facebook access token"}]];
    }
}
```

`AWSCognitoCredentialsProvider`를 인스턴스화할 때 생성자에서 `AWSIdentityProviderManager`의 값으로 `identityProviderManager`를 구현하는 클래스를 전달하십시오. 자세한 내용을 보려면 AWSCognitoCredentialsProvider참조 페이지로 이동하여 initWithRegionType:identityPoolId: identityProviderManager 을 선택하십시오.

iOS - Swift

Facebook 인증을 추가하려면 먼저 Facebook 안내서를 따라 Facebook SDK를 애플리케이션에 통합합니다. 그런 다음 Facebook으로 로그인 버튼을 귀하의 사용자 인터페이스에 추가합니다. Facebook SDK는 세션 객체를 사용하여 상태를 추적합니다. Amazon Cognito는 이 세션 객체의 액세스 토큰을 사용하여 사용자를 인증하고 고유한 Amazon Cognito 자격 증명 풀(페더레이션 자격 증명)에 바인딩합니다.

Amazon Cognito에 Facebook 액세스 토큰을 제공하려면 AWSIdentityProviderManager 프로토콜을 구현합니다.

`logins` 메서드를 구현할 때 `AWSIdentityProviderFacebook`이 포함된 사전을 반환합니다. 다음 코드 예와 같이 이 사전은 키 역할을 하고 인증된 Facebook 사용자에게서 받은 현재 액세스 토큰은 값 역할을 합니다.

```
class FacebookProvider: NSObject, AWSIdentityProviderManager {
    func logins() -> AWSTask<NSDictionary> {
        if let token = AccessToken.current?.authenticationToken {
            return AWSTask(result: [AWSIdentityProviderFacebook:token])
        }
        return AWSTask(error:NSError(domain: "Facebook Login", code: -1 , userInfo:
 ["Facebook" : "No current Facebook access token"]))
    }
}
```

`AWSCognitoCredentialsProvider`를 인스턴스화할 때 생성자에서 `AWSIdentityProviderManager`의 값으로 `identityProviderManager`를 구현하는 클래스를 전달하십시오. 자세한 내용을 보려면 AWSCognitoCredentialsProvider참조 페이지로 이동하여 initWithRegionType:identityPoolId: 을 선택하십시오identityProviderManager.

JavaScript

Facebook 인증을 추가하려면 Facebook Login for the Web(웹용 Facebook 로그인)에 따라 웹 사이트에 Facebook으로 로그인(Login with Facebook) 버튼을 추가합니다. Facebook SDK는 세션 객체를 사용하여 상태를 추적합니다. Amazon Cognito는 이 세션 객체의 액세스 토큰을 사용하여 사용자를 인증하고, 고유 식별자를 생성하고, 필요한 경우 사용자에게 다른 리소스에 대한 액세스 권한을 부여합니다. AWS

Facebook SDK로 사용자를 인증한 후에는 Amazon Cognito 자격 증명 공급자에 세션 토큰을 추가합니다.

```
FB.login(function (response) {

  // Check if the user logged in successfully.
  if (response.authResponse) {

    console.log('You are now logged in.');

    // Add the Facebook access token to the Amazon Cognito credentials login map.
    AWS.config.credentials = new AWS.CognitoIdentityCredentials({
      IdentityPoolId: 'IDENTITY_POOL_ID',
      Logins: {
        'graph.facebook.com': response.authResponse.accessToken
      }
    });

    // Obtain AWS credentials
    AWS.config.credentials.get(function(){
        // Access AWS resources here.
    });

  } else {
    console.log('There was a problem logging you in.');
  }

});
```

Facebook SDK는 Amazon Cognito가 인증된 최종 AWS 사용자의 자격 증명을 생성하는 데 사용하는 OAuth 토큰을 획득합니다. 또한 Amazon Cognito는 사용자 데이터베이스에 이 Facebook 자격 증명과 일치하는 사용자가 있는지 확인하는 데에도 이 토큰을 사용합니다. 사용자가 이미 있으면 API가 기존의 식별자를 반환합니다. 그렇지 않으면 새 식별자가 반환됩니다. 식별자는 로컬 디바이스에서 클라이언트 SDK에 의해 자동으로 캐시됩니다.

> **Note**
>
> 로그인 맵을 설정한 후 `refresh` 또는 `get`을 호출하여 자격 증명을 가져옵니다. 코드 예제는 README 파일의 “사용 사례 17, Cognito ID와 사용자 풀 통합”을 참조하십시오. JavaScript

Unity

Facebook 인증을 추가하려면 먼저 Facebook 안내서를 따라 Facebook SDK를 애플리케이션에 통합합니다. Amazon Cognito는 FB 객체의 Facebook 액세스 토큰을 사용하여 Amazon Cognito 자격 증명에 연결된 고유한 사용자 식별자를 생성합니다.

Facebook SDK로 사용자를 인증한 후에는 Amazon Cognito 자격 증명 공급자에 세션 토큰을 추가합니다.

```
void Start()
{
    FB.Init(delegate() {
        if (FB.IsLoggedIn) { //User already logged in from a previous session
            AddFacebookTokenToCognito();
        } else {
            FB.Login ("email", FacebookLoginCallback);
        }
    });
}

void FacebookLoginCallback(FBResult result)
{
    if (FB.IsLoggedIn)
    {
        AddFacebookTokenToCognito();
    }
    else
    {
        Debug.Log("FB Login error");
    }
}

void AddFacebookTokenToCognito()
{
    credentials.AddLogin ("graph.facebook.com",
 AccessToken.CurrentAccessToken.TokenString);
}
```

`FB.AccessToken`을 사용하기 전에 `FB.Login()`을 호출하고 `FB.IsLoggedIn`이 true인지 확인합니다.

Xamarin

Android용 Xamarin:

```
public void InitializeFacebook() {
    FacebookSdk.SdkInitialize(this.ApplicationContext);
    callbackManager = CallbackManagerFactory.Create();
    LoginManager.Instance.RegisterCallback(callbackManager, new FacebookCallback &lt;
 LoginResult &gt; () {
      HandleSuccess = loginResult = &gt; {
        var accessToken = loginResult.AccessToken;
        credentials.AddLogin("graph.facebook.com", accessToken.Token);
        //open new activity
      },
      HandleCancel = () = &gt; {
        //throw error message
      },
      HandleError = loginError = &gt; {
        //throw error message
      }
    });
    LoginManager.Instance.LogInWithReadPermissions(this, new List &lt; string &gt; {
      "public_profile"
    });
  }
```

iOS용 Xamarin:

```
public void InitializeFacebook() {
  LoginManager login = new LoginManager();
  login.LogInWithReadPermissions(readPermissions.ToArray(),
 delegate(LoginManagerLoginResult result, NSError error) {
    if (error != null) {
      //throw error message
    } else if (result.IsCancelled) {
      //throw error message
    } else {
      var accessToken = loginResult.AccessToken;
      credentials.AddLogin("graph.facebook.com", accessToken.Token);
      //open new view controller
    }
  });
}
```

Amazon으로 로그인을 자격 증명 풀 IdP로 설정

Amazon Cognito는 Login with Amazon과 통합하여 모바일 및 웹 앱 사용자를 위해 페더레이션된 인증을 제공합니다. 이 섹션에서는 Login with Amazon을 자격 증명 공급자(IdP)로 사용하여 애플리케이션을 등록하고 설정하는 방법을 설명합니다.

개발자 포털에서 Login with Amazon이 Amazon Cognito와 연동하도록 설정합니다. 자세한 내용은 Login with Amazon FAQ에서 Setting Up Login with Amazon(Login with Amazon 설정)을 참조하세요.

> **Note**
>
> Login with Amazon을 Xamarin 애플리케이션에 통합하려면 Xamarin Getting Started Guide(Xamarin 시작 안내서)를 따릅니다.

> **Note**
>
> Unity 플랫폼에서는 Login with Amazon을 기본적으로 통합할 수 없습니다. 대신 웹 보기를 사용하여 브라우저 로그인 흐름을 수행합니다.

Login with Amazon 설정

Login with Amazon 구현

Amazon 개발자 포털에서 OAuth 애플리케이션을 자격 증명 풀과 통합하도록 설정하고, Login with Amazon 설명서를 찾고, SDK를 다운로드할 수 있습니다. 개발자 콘솔(Developer console)을 선택한 다음 개발자 포털에서 Login with Amazon을 선택합니다. 애플리케이션에 대한 보안 프로필을 생성한 다음 앱에 대한 Login with Amazon 인증 메커니즘을 빌드할 수 있습니다. Login with Amazon 인증을 앱과 통합하는 방법에 대한 자세한 내용은 자격 증명 얻기 섹션을 참조하세요.

Amazon에서는 새 보안 프로필에 대한 OAuth 2.0 클라이언트 ID(client ID)를 발급합니다. 보안 프로필 웹 설정(Web Settings) 탭에서 클라이언트 ID(client ID)를 찾을 수 있습니다. 자격 증명 풀에서 Login with Amazon IdP의 App ID 필드에 보안 프로필 ID를 입력합니다.

> **Note**
>
> 자격 증명 풀에서 Login with Amazon IdP의 App ID 필드에 보안 프로필 ID를 입력합니다. 이는 클라이언트 ID를 사용하는 사용자 풀과 다릅니다.

Amazon Cognito 콘솔에서 외부 공급자 구성

Login with Amazon ID 제공업체(idP)를 추가하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. ID 제공업체 추가를 선택합니다.
4. Login with Amazon을 선택합니다.
5. Login with Amazon에서 생성한 OAuth 프로젝트의 앱 ID를 입력합니다. 자세한 내용은 Login with Amazon 설명서를 참조하세요.
6. Amazon Cognito가 이 공급자를 통해 인증된 사용자에게 보안 인증을 발급할 때 요청하는 역할을 설정하려면 역할 설정을 구성하세요.

   - 인증된 역할을 구성할 때 설정한 기본 역할을 이 IdP의 사용자에게 할당하거나 규칙을 사용하여 역할 선택을 선택할 수 있습니다.

     i. 규칙을 사용하여 역할 선택을 선택한 경우 사용자 인증의 소스 클레임, 클레임 비교 기준으로 사용할 운영자, 이 역할 선택과 일치하도록 하는 값, 역할 할당이 일치할 때 할당할 역할을 입력합니다. 다른 조건에 따라 추가 규칙을 생성하려면 다른 항목 추가를 선택합니다.

     ii. 역할 해결을 선택합니다. 사용자의 클레임이 규칙과 일치하지 않는 경우 보안 인증을 거부하거나 인증된 역할의 보안 인증을 발급할 수 있습니다.
7. Amazon Cognito가 이 공급자를 통해 인증한 사용자에게 보안 인증을 발급할 때 할당하는 보안 주체 태그를 변경하려면 액세스 제어를 위한 속성을 구성합니다.

   a. 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.

   b. sub 및 aud 클레임 기반 보안 주체 태그를 적용하려면 기본 매핑 사용을 선택합니다.

   c. 보안 주체 태그에 대한 속성의 자체 사용자 지정 스키마를 생성하려면 사용자 지정 매핑 사용을 선택합니다. 그런 다음 태그에 표시하려는 각 클레임에서 소싱하려는 태그 키를 입력합니다.
8. 변경 사항 저장(Save changes)을 선택합니다.

Login with Amazon 사용: Android

Amazon 로그인을 인증한 후에는 인터페이스의 OnSuccess 메서드에서 Amazon Cognito 자격 증명 공급자에게 토큰을 전달할 수 있습니다. TokenListener 코드는 다음과 같습니다.

```
@Override
public void onSuccess(Bundle response) {
    String token = response.getString(AuthzConstants.BUNDLE_KEY.TOKEN.val);
    Map<String, String> logins = new HashMap<String, String>();
    logins.put("www.amazon.com", token);
    credentialsProvider.setLogins(logins);
}
```

Login with Amazon 사용: iOS - Objective-C

Amazon 로그인을 인증한 후에는 AMZN의 메서드를 사용하여 Amazon Cognito 자격 증명 공급자에게 토큰을 전달할 수 있습니다. requestDidSucceed AccessTokenDelegate

```
- (void)requestDidSucceed:(APIResult \*)apiResult {
    if (apiResult.api == kAPIAuthorizeUser) {
        [AIMobileLib getAccessTokenForScopes:[NSArray arrayWithObject:@"profile"]
 withOverrideParams:nil delegate:self];
    }
    else if (apiResult.api == kAPIGetAccessToken) {
        credentialsProvider.logins = @{ @(AWSCognitoLoginProviderKeyLoginWithAmazon):
 apiResult.result };
    }
}}
```

Login with Amazon 사용: iOS - Swift

Amazon 로그인을 승인한 후 `AMZNAccessTokenDelegate`의 `requestDidSucceed` 메서드에서 Amazon Cognito 자격 증명 공급자에 토큰을 전달할 수 있습니다.

```
func requestDidSucceed(apiResult: APIResult!) {
    if apiResult.api == API.AuthorizeUser {
        AIMobileLib.getAccessTokenForScopes(["profile"], withOverrideParams: nil,
 delegate: self)
    } else if apiResult.api == API.GetAccessToken {
        credentialsProvider.logins =
 [AWSCognitoLoginProviderKey.LoginWithAmazon.rawValue: apiResult.result]
    }
}
```

Amazon 로그인 사용: JavaScript

사용자가 Login with Amazon으로 인증하고 다시 웹 사이트로 리디렉션되면 쿼리 문자열에 Login with Amazon 액세스 토큰이 제공됩니다. 이 토큰을 자격 증명 로그인 맵에 전달하세요.

```
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
   IdentityPoolId: 'IDENTITY_POOL_ID',
   Logins: {
       'www.amazon.com': 'Amazon Access Token'
   }
});
```

Login with Amazon 사용: Xamarin

Android용 Xamarin

```
AmazonAuthorizationManager manager = new AmazonAuthorizationManager(this,
 Bundle.Empty);

var tokenListener = new APIListener {
  Success = response => {
    // Get the auth token
    var token = response.GetString(AuthzConstants.BUNDLE_KEY.Token.Val);
    credentials.AddLogin("www.amazon.com", token);
  }
};

// Try and get existing login
manager.GetToken(new[] {
  "profile"
}, tokenListener);
```

iOS용 Xamarin

`AppDelegate.cs`에서 다음을 삽입합니다.

```
public override bool OpenUrl (UIApplication application, NSUrl url, string
 sourceApplication, NSObject annotation)
{
    // Pass on the url to the SDK to parse authorization code from the url
    bool isValidRedirectSignInURL = AIMobileLib.HandleOpenUrl (url, sourceApplication);
```

```
    if(!isValidRedirectSignInURL)
        return false;

    // App may also want to handle url
    return true;
}
```

그런 다음 ViewController.cs에서 다음을 수행합니다.

```
public override void ViewDidLoad ()
{
    base.LoadView ();

    // Here we create the Amazon Login Button
    btnLogin = UIButton.FromType (UIButtonType.RoundedRect);
    btnLogin.Frame = new RectangleF (55, 206, 209, 48);
    btnLogin.SetTitle ("Login using Amazon", UIControlState.Normal);
    btnLogin.TouchUpInside += (sender, e) => {
        AIMobileLib.AuthorizeUser (new [] { "profile"}, new AMZNAuthorizationDelegate
 ());
    };
    View.AddSubview (btnLogin);
}

// Class that handles Authentication Success/Failure
public class AMZNAuthorizationDelegate : AIAuthenticationDelegate
{
  public override void RequestDidSucceed(ApiResult apiResult)
    {
      // Your code after the user authorizes application for requested scopes
      var token = apiResult["access_token"];
      credentials.AddLogin("www.amazon.com",token);
    }

    public override void RequestDidFail(ApiError errorResponse)
    {
      // Your code when the authorization fails
      InvokeOnMainThread(() => new UIAlertView("User Authorization Failed",
 errorResponse.Error.Message, null, "Ok", null).Show());
    }
}
```

구글을 아이덴티티 풀 IdP로 설정하기

Amazon Cognito는 Google과 통합되어 모바일 애플리케이션 사용자를 위해 연동된 인증을 제공합니다. 이 섹션에서는 Google을 IdP로 사용하여 애플리케이션을 등록하고 설정하는 방법을 설명합니다.

Android

> **Note**
>
> 앱이 Google을 사용하고 여러 모바일 플랫폼에서 사용 가능한 경우 앱을 OpenID Connect 공급자로 구성해야 합니다. 더 나은 통합을 위해 생성된 모든 클라이언트 ID를 추가 대상 값으로 추가합니다. Google 클라이언트 간 자격 증명 모델에 대한 자세한 내용은 클라이언트 간 ID를 참조하세요.

Google 설정

Android용 Google 로그인을 활성화하려면 애플리케이션을 위한 Google 개발자 콘솔 프로젝트를 생성합니다.

1. Google 개발자 콘솔로 이동하여 새 프로젝트를 만듭니다.
2. API 및 서비스(APIs & Services)를 선택한 다음 OAuth 동의 화면(OAuth consent screen)을 선택합니다. Google에서 프로필 데이터를 앱과 공유하는 데 동의를 요청할 때 Google이 사용자에게 표시하는 정보를 사용자 지정합니다.
3. 자격 증명(Credentials)을 선택한 다음 자격 증명 생성(Create credentials)을 선택합니다. OAuth 클라이언트 ID(OAuth client ID)를 선택합니다. Android를 애플리케이션 유형(Application type)으로 선택합니다. 앱을 개발하는 각 플랫폼에 대해 별도의 클라이언트 ID를 생성합니다.
4. 자격 증명(Credentials)에서 서비스 계정 관리(Manage service accounts)를 선택합니다. 서비스 계정 생성(Create service account)을 선택합니다. 서비스 계정 세부 정보를 입력한 다음 생성 후 계속(Create and continue)을 선택합니다.
5. 서비스 계정에 프로젝트에 대한 액세스 권한을 부여합니다. 앱에 필요한 대로 사용자에게 서비스 계정에 대한 액세스 권한을 부여합니다.
6. 새 서비스 계정을 선택하고 키(Keys) 탭을 선택하고 키 추가(Add key)를 선택합니다. 새 JSON 키를 생성하고 다운로드합니다.

Google 개발자 콘솔을 사용하는 방법에 대한 자세한 내용은 Google 클라우드 설명서에서 프로젝트 만들기 및 관리를 참조하세요.

Google을 Android 앱에 통합하는 방법에 대한 자세한 내용은 Google ID 설명서에서 Google 로그인으로 사용자 인증을 참조하십시오.

Google ID 제공업체(idP)를 추가하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. ID 제공업체 추가를 선택합니다.
4. Google을 선택합니다.
5. Google Cloud Platform에서 생성한 OAuth 프로젝트의 클라이언트 ID를 입력합니다. 자세한 내용은 Google Cloud Platform Console 도움말의 OAuth 2.0 설정을 참조하세요.
6. Amazon Cognito가 이 공급자를 통해 인증된 사용자에게 보안 인증을 발급할 때 요청하는 역할을 설정하려면 역할 설정을 구성하세요.
   - 인증된 역할을 구성할 때 설정한 기본 역할을 이 IdP의 사용자에게 할당하거나 규칙을 사용하여 역할 선택을 선택할 수 있습니다.
     i. 규칙을 사용하여 역할 선택을 선택한 경우 사용자 인증의 소스 클레임, 클레임 비교 기준으로 사용할 운영자, 이 역할 선택과 일치하도록 하는 값, 역할 할당이 일치할 때 할당할 역할을 입력합니다. 다른 조건에 따라 추가 규칙을 생성하려면 다른 항목 추가를 선택합니다.
     ii. 역할 해결을 선택합니다. 사용자의 클레임이 규칙과 일치하지 않는 경우 보안 인증을 거부하거나 인증된 역할의 보안 인증을 발급할 수 있습니다.
7. Amazon Cognito가 이 공급자를 통해 인증한 사용자에게 보안 인증을 발급할 때 할당하는 보안 주체 태그를 변경하려면 액세스 제어를 위한 속성을 구성합니다.
   a. 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.
   b. sub 및 aud 클레임 기반 보안 주체 태그를 적용하려면 기본 매핑 사용을 선택합니다.
   c. 보안 주체 태그에 대한 속성의 자체 사용자 지정 스키마를 생성하려면 사용자 지정 매핑 사용을 선택합니다. 그런 다음 태그에 표시하려는 각 클레임에서 소싱하려는 태그 키를 입력합니다.
8. 변경 사항 저장(Save changes)을 선택합니다.

Google 사용

애플리케이션에서 Google로 로그인하도록 허용하려면 Android용 Google 설명서의 지침을 따르세요. 사용자가 로그인하면 Google에서 OpenID Connect 인증 토큰을 요청합니다. 그런 다음 Amazon Cognito에서 토큰을 사용하여 사용자를 인증하고 고유한 식별자를 생성합니다.

다음 코드 예는 Google Play 서비스에서 인증 토큰을 가져오는 방법을 보여줍니다.

```
GooglePlayServicesUtil.isGooglePlayServicesAvailable(getApplicationContext());
AccountManager am = AccountManager.get(this);
Account[] accounts = am.getAccountsByType(GoogleAuthUtil.GOOGLE_ACCOUNT_TYPE);
String token = GoogleAuthUtil.getToken(getApplicationContext(), accounts[0].name,
        "audience:server:client_id:YOUR_GOOGLE_CLIENT_ID");
Map<String, String> logins = new HashMap<String, String>();
logins.put("accounts.google.com", token);
credentialsProvider.setLogins(logins);
```

iOS - Objective-C

> **Note**
>
> 앱이 Google을 사용하고 여러 모바일 플랫폼에서 사용 가능한 경우 Google을 OpenID Connect 공급자로 구성합니다. 더 나은 통합을 위해 생성된 모든 클라이언트 ID를 추가 대상 값으로 추가합니다. Google 클라이언트 간 자격 증명 모델에 대한 자세한 내용은 클라이언트 간 ID를 참조하세요.

Google 설정

iOS용 Google 로그인을 사용하려면 애플리케이션을 위한 Google 개발자 콘솔 프로젝트를 생성합니다.

1. Google 개발자 콘솔로 이동하여 새 프로젝트를 만듭니다.
2. API 및 서비스(APIs & Services)를 선택한 다음 OAuth 동의 화면(OAuth consent screen)을 선택합니다. Google에서 프로필 데이터를 앱과 공유하는 데 동의를 요청할 때 Google이 사용자에게 표시하는 정보를 사용자 지정합니다.
3. 자격 증명(Credentials)을 선택한 다음 자격 증명 생성(Create credentials)을 선택합니다. OAuth 클라이언트 ID(OAuth client ID)를 선택합니다. iOS를 애플리케이션 유형(Application type)으로 선택합니다. 앱을 개발하는 각 플랫폼에 대해 별도의 클라이언트 ID를 생성합니다.

4. 자격 증명(Credentials)에서 서비스 계정 관리(Manage service accounts)를 선택합니다. 서비스 계정 생성(Create service account)을 선택합니다. 서비스 계정 세부 정보를 입력한 다음 생성 후 계속(Create and continue)을 선택합니다.
5. 서비스 계정에 프로젝트에 대한 액세스 권한을 부여합니다. 앱에 필요한 대로 사용자에게 서비스 계정에 대한 액세스 권한을 부여합니다.
6. 새 서비스 계정을 선택합니다. 키(Keys) 탭을 선택한 다음 키 추가(Add key)를 선택합니다. 새 JSON 키를 생성하고 다운로드합니다.

Google 개발자 콘솔을 사용하는 방법에 대한 자세한 내용은 Google 클라우드 설명서에서 프로젝트 만들기 및 관리를 참조하세요.

Google을 iOS 앱에 통합하는 방법에 대한 자세한 내용은 Google 자격 증명 설명서에서 Google Sign-In for iOS를 참조하세요.

Google ID 제공업체(idP)를 추가하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. ID 제공업체 추가를 선택합니다.
4. Google을 선택합니다.
5. Google Cloud Platform에서 생성한 OAuth 프로젝트의 클라이언트 ID를 입력합니다. 자세한 내용은 Google Cloud Platform Console 도움말의 OAuth 2.0 설정을 참조하세요.
6. Amazon Cognito가 이 공급자를 통해 인증된 사용자에게 보안 인증을 발급할 때 요청하는 역할을 설정하려면 역할 설정을 구성하세요.

    - 인증된 역할을 구성할 때 설정한 기본 역할을 이 IdP의 사용자에게 할당하거나 규칙을 사용하여 역할 선택을 선택할 수 있습니다.

        i. 규칙을 사용하여 역할 선택을 선택한 경우 사용자 인증의 소스 클레임, 클레임 비교 기준으로 사용할 운영자, 이 역할 선택과 일치하도록 하는 값, 역할 할당이 일치할 때 할당할 역할을 입력합니다. 다른 조건에 따라 추가 규칙을 생성하려면 다른 항목 추가를 선택합니다.

        ii. 역할 해결을 선택합니다. 사용자의 클레임이 규칙과 일치하지 않는 경우 보안 인증을 거부하거나 인증된 역할의 보안 인증을 발급할 수 있습니다.
7. Amazon Cognito가 이 공급자를 통해 인증한 사용자에게 보안 인증을 발급할 때 할당하는 보안 주체 태그를 변경하려면 액세스 제어를 위한 속성을 구성합니다.

a. 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.

b. sub 및 aud 클레임 기반 보안 주체 태그를 적용하려면 기본 매핑 사용을 선택합니다.

c. 보안 주체 태그에 대한 속성의 자체 사용자 지정 스키마를 생성하려면 사용자 지정 매핑 사용을 선택합니다. 그런 다음 태그에 표시하려는 각 클레임에서 소싱하려는 태그 키를 입력합니다.

8. 변경 사항 저장(Save changes)을 선택합니다.

Google 사용

애플리케이션에서 Google로 로그인하도록 허용하려면 iOS용 Google 설명서를 따르세요. 인증에 성공하면 OpenID Connect 인증 토큰을 가져오고 Amazon Cognito에서 이를 사용하여 사용자를 인증하고 고유 식별자를 생성합니다.

인증에 성공하면 `id_token`이 포함된 `GTMOAuth2Authentication` 객체를 가져오고 Amazon Cognito에서 이를 사용하여 사용자를 인증하고 고유 식별자를 생성합니다.

```
- (void)finishedWithAuth: (GTMOAuth2Authentication *)auth error: (NSError *) error {
        NSString *idToken = [auth.parameters objectForKey:@"id_token"];
        credentialsProvider.logins = @{ @(AWSCognitoLoginProviderKeyGoogle): idToken };
    }
```

iOS - Swift

> **Note**
>
> 앱이 Google을 사용하고 여러 모바일 플랫폼에서 사용 가능한 경우 Google을 OpenID Connect 공급자로 구성합니다. 더 나은 통합을 위해 생성된 모든 클라이언트 ID를 추가 대상 값으로 추가합니다. Google 클라이언트 간 자격 증명 모델에 대한 자세한 내용은 클라이언트 간 ID를 참조하세요.

Google 설정

iOS용 Google 로그인을 사용하려면 애플리케이션을 위한 Google 개발자 콘솔 프로젝트를 생성합니다.

1. Google 개발자 콘솔로 이동하여 새 프로젝트를 만듭니다.

2. API 및 서비스(APIs & Services)를 선택한 다음 OAuth 동의 화면(OAuth consent screen)을 선택합니다. Google에서 프로필 데이터를 앱과 공유하는 데 동의를 요청할 때 Google이 사용자에게 표시하는 정보를 사용자 지정합니다.
3. 자격 증명(Credentials)을 선택한 다음 자격 증명 생성(Create credentials)을 선택합니다. OAuth 클라이언트 ID(OAuth client ID)를 선택합니다. iOS를 애플리케이션 유형(Application type)으로 선택합니다. 앱을 개발하는 각 플랫폼에 대해 별도의 클라이언트 ID를 생성합니다.
4. 자격 증명(Credentials)에서 서비스 계정 관리(Manage service accounts)를 선택합니다. 서비스 계정 생성(Create service account)을 선택합니다. 서비스 계정 세부 정보를 입력한 다음 생성 후 계속(Create and continue)을 선택합니다.
5. 서비스 계정에 프로젝트에 대한 액세스 권한을 부여합니다. 앱에 필요한 대로 사용자에게 서비스 계정에 대한 액세스 권한을 부여합니다.
6. 새 서비스 계정을 선택하고 키(Keys) 탭을 선택하고 키 추가(Add key)를 선택합니다. 새 JSON 키를 생성하고 다운로드합니다.

Google 개발자 콘솔을 사용하는 방법에 대한 자세한 내용은 Google 클라우드 설명서에서 프로젝트 만들기 및 관리를 참조하세요.

Google을 iOS 앱에 통합하는 방법에 대한 자세한 내용은 Google 자격 증명 설명서에서 Google Sign-In for iOS를 참조하세요.

Amazon Cognito 콘솔 홈 페이지에서 자격 증명 풀 관리(Manage Identity Pools)를 선택합니다.

Amazon Cognito 콘솔에서 외부 공급자 구성

1. Google을 외부 공급자로 사용할 자격 증명 풀의 이름을 선택합니다. 자격 증명 풀에 대한 대시보드 페이지가 표시됩니다.
2. 대시보드 페이지의 우측 상단 모서리에서 자격 증명 풀 편집을 선택합니다. 자격 증명 풀 편집(Edit identity pool) 페이지가 표시됩니다.
3. 아래로 스크롤하고 인증 공급자(Authentication providers)를 선택하여 섹션을 확장합니다.
4. Google 탭을 선택합니다.
5. 잠금 해제를 선택합니다.
6. Google에서 가져온 Google 클라이언트 ID를 입력하고 변경 사항 저장(Save Changes)을 선택합니다.

Google 사용

애플리케이션에서 Google로 로그인하도록 허용하려면 iOS용 Google 설명서를 따르세요. 인증에 성공하면 OpenID Connect 인증 토큰이 생성됩니다. Amazon Cognito에서는 이 토큰을 사용하여 사용자를 인증하고 고유한 식별자를 생성합니다.

인증에 성공하면 `GTMOAuth2Authentication`를 포함하는 `id_token` 객체가 생성됩니다. Amazon Cognito에서는 이 토큰을 사용하여 사용자를 인증하고 고유한 식별자를 생성합니다.

```
func finishedWithAuth(auth: GTMOAuth2Authentication!, error: NSError!) {
    if error != nil {
      print(error.localizedDescription)
    }
    else {
      let idToken = auth.parameters.objectForKey("id_token")
      credentialsProvider.logins = [AWSCognitoLoginProviderKey.Google.rawValue:
 idToken!]
    }
}
```

JavaScript

> **Note**
>
> 앱이 Google을 사용하고 여러 모바일 플랫폼에서 사용 가능한 경우 Google을 OpenID Connect 공급자로 구성해야 합니다. 더 나은 통합을 위해 생성된 모든 클라이언트 ID를 추가 대상 값으로 추가합니다. Google 클라이언트 간 자격 증명 모델에 대한 자세한 내용은 클라이언트 간 ID를 참조하세요.

Google 설정

JavaScript 웹 앱에 Google 로그인을 사용하도록 설정하려면 애플리케이션용 Google Developers 콘솔 프로젝트를 만드세요.

1. Google 개발자 콘솔로 이동하여 새 프로젝트를 만듭니다.
2. API 및 서비스(APIs & Services)를 선택한 다음 OAuth 동의 화면(OAuth consent screen)을 선택합니다. Google에서 프로필 데이터를 앱과 공유하는 데 동의를 요청할 때 Google이 사용자에게 표시하는 정보를 사용자 지정합니다.
3. 자격 증명(Credentials)을 선택한 다음 자격 증명 생성(Create credentials)을 선택합니다. OAuth 클라이언트 ID(OAuth client ID)를 선택합니다. 웹 애플리케이션(Web application)을 애플리케이션

유형(Application type)으로 선택합니다. 앱을 개발하는 각 플랫폼에 대해 별도의 클라이언트 ID를 생성합니다.

4. 자격 증명(Credentials)에서 서비스 계정 관리(Manage service accounts)를 선택합니다. 서비스 계정 생성(Create service account)을 선택합니다. 서비스 계정 세부 정보를 입력한 다음 생성 후 계속(Create and continue)을 선택합니다.
5. 서비스 계정에 프로젝트에 대한 액세스 권한을 부여합니다. 앱에 필요한 대로 사용자에게 서비스 계정에 대한 액세스 권한을 부여합니다.
6. 새 서비스 계정을 선택하고 키(Keys) 탭을 선택하고 키 추가(Add key)를 선택합니다. 새 JSON 키를 생성하고 다운로드합니다.

Google 개발자 콘솔을 사용하는 방법에 대한 자세한 내용은 Google 클라우드 설명서에서 프로젝트 만들기 및 관리를 참조하세요.

Google을 웹 앱에 통합하는 방법에 대한 자세한 내용은 Google 자격 증명 설명서에서 Sign in With Google을 참조하세요.

Amazon Cognito 콘솔에서 외부 공급자 구성

Google ID 제공업체(idP)를 추가하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. ID 제공업체 추가를 선택합니다.
4. Google을 선택합니다.
5. Google Cloud Platform에서 생성한 OAuth 프로젝트의 클라이언트 ID를 입력합니다. 자세한 내용은 Google Cloud Platform Console 도움말의 OAuth 2.0 설정을 참조하세요.
6. Amazon Cognito가 이 공급자를 통해 인증된 사용자에게 보안 인증을 발급할 때 요청하는 역할을 설정하려면 역할 설정을 구성하세요.

    - 인증된 역할을 구성할 때 설정한 기본 역할을 이 IdP의 사용자에게 할당하거나 규칙을 사용하여 역할 선택을 선택할 수 있습니다.

        i. 규칙을 사용하여 역할 선택을 선택한 경우 사용자 인증의 소스 클레임, 클레임 비교 기준으로 사용할 운영자, 이 역할 선택과 일치하도록 하는 값, 역할 할당이 일치할 때 할당할 역할을 입력합니다. 다른 조건에 따라 추가 규칙을 생성하려면 다른 항목 추가를 선택합니다.

ii. 역할 해결을 선택합니다. 사용자의 클레임이 규칙과 일치하지 않는 경우 보안 인증을 거부하거나 인증된 역할의 보안 인증을 발급할 수 있습니다.

7. Amazon Cognito가 이 공급자를 통해 인증한 사용자에게 보안 인증을 발급할 때 할당하는 보안 주체 태그를 변경하려면 액세스 제어를 위한 속성을 구성합니다.

   a. 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.

   b. sub 및 aud 클레임 기반 보안 주체 태그를 적용하려면 기본 매핑 사용을 선택합니다.

   c. 보안 주체 태그에 대한 속성의 자체 사용자 지정 스키마를 생성하려면 사용자 지정 매핑 사용을 선택합니다. 그런 다음 태그에 표시하려는 각 클레임에서 소싱하려는 태그 키를 입력합니다.

8. 변경 사항 저장(Save changes)을 선택합니다.

Google 사용

애플리케이션에서 Google로 로그인하도록 허용하려면 웹용 Google 설명서를 따르세요.

인증에 성공하면 `id_token`이 포함된 응답 객체가 생성됩니다 Amazon Cognito에서는 이 객체를 사용하여 사용자를 인증하고 고유한 식별자를 생성합니다.

```
function signinCallback(authResult) {
  if (authResult['status']['signed_in']) {

     // Add the Google access token to the Amazon Cognito credentials login map.
     AWS.config.credentials = new AWS.CognitoIdentityCredentials({
        IdentityPoolId: 'IDENTITY_POOL_ID',
        Logins: {
           'accounts.google.com': authResult['id_token']
        }
     });

     // Obtain AWS credentials
     AWS.config.credentials.get(function(){
        // Access AWS resources here.
     });
  }
}
```

Unity

Google 설정

Unity 앱용 Google 로그인을 사용하려면 애플리케이션을 위한 Google 개발자 콘솔 프로젝트를 생성합니다.

1. Google 개발자 콘솔로 이동하여 새 프로젝트를 만듭니다.
2. API 및 서비스(APIs & Services)를 선택한 다음 OAuth 동의 화면(OAuth consent screen)을 선택합니다. Google에서 프로필 데이터를 앱과 공유하는 데 동의를 요청할 때 Google이 사용자에게 표시하는 정보를 사용자 지정합니다.
3. 자격 증명(Credentials)을 선택한 다음 자격 증명 생성(Create credentials)을 선택합니다. OAuth 클라이언트 ID(OAuth client ID)를 선택합니다. 웹 애플리케이션(Web application)을 애플리케이션 유형(Application type)으로 선택합니다. 앱을 개발하는 각 플랫폼에 대해 별도의 클라이언트 ID를 생성합니다.
4. Unity의 경우 OAuth 클라이언트 ID(OAuth client ID)를 Android용으로, 그리고 다른 하나는 iOS용으로 추가로 생성합니다.
5. 자격 증명(Credentials)에서 서비스 계정 관리(Manage service accounts)를 선택합니다. 서비스 계정 생성(Create service account)을 선택합니다. 서비스 계정 세부 정보를 입력한 다음 생성 후 계속(Create and continue)을 선택합니다.
6. 서비스 계정에 프로젝트에 대한 액세스 권한을 부여합니다. 앱에 필요한 대로 사용자에게 서비스 계정에 대한 액세스 권한을 부여합니다.
7. 새 서비스 계정을 선택하고 키(Keys) 탭을 선택하고 키 추가(Add key)를 선택합니다. 새 JSON 키를 생성하고 다운로드합니다.

Google 개발자 콘솔을 사용하는 방법에 대한 자세한 내용은 Google 클라우드 설명서에서 프로젝트 만들기 및 관리를 참조하세요.

IAM 콘솔에서 OpenID 제공업체 만들기

1. IAM 콘솔에서 OpenID 제공업체 생성 OpenID 제공업체를 설정하는 방법에 대한 자세한 내용은 OpenID Connect 자격 증명 제공업체 사용을 참조하세요.
2. 공급자 URL을 입력하라는 메시지가 나타나면 `"https://accounts.google.com"`을 입력합니다.
3. Audience 필드에 값을 입력하라는 메시지가 나타나면 전 단계에서 만든 클라이언트 ID 3개 중 하나를 입력합니다.

4. 공급자 이름을 선택하고 다른 클라이언트 ID 두 개를 사용하여 대상 두 개를 더 추가합니다.

Amazon Cognito 콘솔에서 외부 공급자 구성

Amazon Cognito 콘솔 홈 페이지에서 자격 증명 풀 관리(Manage Identity Pools)를 선택합니다.

Google ID 제공업체(idP)를 추가하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. ID 제공업체 추가를 선택합니다.
4. Google을 선택합니다.
5. Google Cloud Platform에서 생성한 OAuth 프로젝트의 클라이언트 ID를 입력합니다. 자세한 내용은 Google Cloud Platform Console 도움말의 OAuth 2.0 설정을 참조하세요.
6. Amazon Cognito가 이 공급자를 통해 인증된 사용자에게 보안 인증을 발급할 때 요청하는 역할을 설정하려면 역할 설정을 구성하세요.
   - 인증된 역할을 구성할 때 설정한 기본 역할을 이 IdP의 사용자에게 할당하거나 규칙을 사용하여 역할 선택을 선택할 수 있습니다.
     i. 규칙을 사용하여 역할 선택을 선택한 경우 사용자 인증의 소스 클레임, 클레임 비교 기준으로 사용할 운영자, 이 역할 선택과 일치하도록 하는 값, 역할 할당이 일치할 때 할당할 역할을 입력합니다. 다른 조건에 따라 추가 규칙을 생성하려면 다른 항목 추가를 선택합니다.
     ii. 역할 해결을 선택합니다. 사용자의 클레임이 규칙과 일치하지 않는 경우 보안 인증을 거부하거나 인증된 역할의 보안 인증을 발급할 수 있습니다.
7. Amazon Cognito가 이 공급자를 통해 인증한 사용자에게 보안 인증을 발급할 때 할당하는 보안 주체 태그를 변경하려면 액세스 제어를 위한 속성을 구성합니다.
   a. 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.
   b. sub 및 aud 클레임 기반 보안 주체 태그를 적용하려면 기본 매핑 사용을 선택합니다.
   c. 보안 주체 태그에 대한 속성의 자체 사용자 지정 스키마를 생성하려면 사용자 지정 매핑 사용을 선택합니다. 그런 다음 태그에 표시하려는 각 클레임에서 소싱하려는 태그 키를 입력합니다.
8. 변경 사항 저장(Save changes)을 선택합니다.

Unity Google 플러그인 설치

1. Unity용 Google Play Games 플러그인을 Unity 프로젝트에 추가합니다.
2. Unity의 Windows 메뉴에서 Android 및 iOS 플랫폼을 위한 ID 세 개를 사용하여 플러그인을 구성합니다.

Google 사용

다음 코드 예는 Google Play 서비스에서 인증 토큰을 가져오는 방법을 보여줍니다.

```
void Start()
{
  PlayGamesClientConfiguration config = new
 PlayGamesClientConfiguration.Builder().Build();
  PlayGamesPlatform.InitializeInstance(config);
  PlayGamesPlatform.DebugLogEnabled = true;
  PlayGamesPlatform.Activate();
  Social.localUser.Authenticate(GoogleLoginCallback);
}

void GoogleLoginCallback(bool success)
{
  if (success)
  {
    string token = PlayGamesPlatform.Instance.GetIdToken();
    credentials.AddLogin("accounts.google.com", token);
  }
  else
  {
    Debug.LogError("Google login failed. If you are not running in an actual Android/
iOS device, this is expected.");
  }
}
```

Xamarin

> **Note**
>
> Amazon Cognito에서는 기본적으로 Xamarin 플랫폼의 Google을 지원하지 않습니다. 현재로서는 웹 보기에서 브라우저 로그인 흐름을 통해 통합해야 합니다. Google 통합이 그 밖의 SDK와 연동하는 방법을 알아보려면 다른 플랫폼을 선택하세요.

애플리케이션에서 Google로 로그인을 사용하려면 사용자를 인증하고 사용자에게서 OpenID Connect 토큰을 받습니다. Amazon Cognito는 이 토큰을 사용하여 Amazon Cognito 자격 증명에 연결된 고유한 사용자 식별자를 생성합니다. 그러나 Xamarin용 Google SDK에서는 OpenID Connect 토큰 가져오기를 지원하지 않으므로 다른 클라이언트나 웹 보기의 웹 흐름을 사용합니다.

토큰을 받으면 `CognitoAWSCredentials`에서 토큰을 설정할 수 있습니다.

```
credentials.AddLogin("accounts.google.com", token);
```

> **Note**
>
> 앱이 Google을 사용하고 여러 모바일 플랫폼에서 사용 가능한 경우 Google을 OpenID Connect 공급자로 구성해야 합니다. 더 나은 통합을 위해 생성된 모든 클라이언트 ID를 추가 대상 값으로 추가합니다. Google 클라이언트 간 자격 증명 모델에 대한 자세한 내용은 클라이언트 간 ID를 참조하세요.

Apple을 자격 증명 풀 IdP로 사용하여 로그인 설정하기

Amazon Cognito는 Sign in with Apple과 통합되어 모바일 애플리케이션 및 웹 애플리케이션 사용자를 위해 페더레이션 인증을 제공합니다. 이 섹션에서는 Apple로 로그인을 자격 증명 공급자(IdP)로 사용하여 애플리케이션을 등록하고 설정하는 방법을 설명합니다.

Apple로 로그인을 인증 공급자로 자격 증명 풀에 추가하려면 두 단계를 진행해야 합니다. 먼저 애플리케이션에서 Apple로 로그인을 통합한 후 자격 증명 풀에서 Sign in with Apple을 구성합니다. Apple로 로그인 설정에 대한 자세한 up-to-date 내용은 Apple 개발자 설명서에서 Apple로 로그인하기 위한 환경 구성을 참조하십시오.

Sign in with Apple 설정

Apple로 로그인을 IdP로 구성하려면 애플리케이션을 Apple에 등록하여 클라이언트 ID를 받습니다.

1. Apple에서 개발자 계정을 생성합니다.
2. Apple 자격 증명으로 로그인합니다.
3. 왼쪽 탐색 창에서 Certificates, IDs & Profiles(인증서, ID 및 프로필)를 선택합니다.
4. 왼쪽 탐색 창에서 Identifiers(식별자)를 선택합니다.
5. 식별자(Identifiers) 페이지에서 +아이콘을 선택합니다.
6. 새 식별자 등록(Register a New Identifier) 페이지에서 앱 ID(App IDs)를 선택한 다음 계속(Continue)을 선택합니다.
7. 앱 ID 등록(Register an App ID) 페이지에서 다음을 수행합니다.
    a. 설명에 설명을 입력합니다.
    b. Bundle ID(번들 ID)에서 식별자를 입력합니다. 이 번들 ID(Bundle ID)를 적어 둡니다. Apple을 자격 증명 풀의 공급자로 구성하려면 이 값이 필요합니다.
    c. 기능(Capabilities)에서 Apple로 로그인(Sign In with Apple)을 선택한 다음 편집(Edit)을 선택합니다.
    d. Sign in with Apple: 앱 ID 구성 페이지에서 앱에 적절한 설정을 선택합니다. 그런 다음 저장을 선택합니다.
    e. 계속을 선택합니다.
8. 앱 ID 확인(Confirm your App ID) 페이지에서 등록(Register)을 선택합니다.
9. Sign In with Apple을 네이티브 iOS 애플리케이션과 통합하려면 10단계로 이동합니다. 11단계는 Sign in with Apple JS와 통합하려는 애플리케이션을 위한 것입니다.
10. 식별자(Identifiers) 페이지에서 앱 ID(App IDs) 메뉴를 선택한 다음 서비스 ID(Services IDs)를 선택합니다. + 아이콘을 선택합니다.
11. 새 식별자 등록(Register a New Identifier) 페이지에서 서비스 ID(Services IDs)를 선택한 다음 계속(Continue)을 선택합니다.
12. 서비스 ID 등록(Register a Services ID) 페이지에서 다음을 수행합니다.
    a. 설명에 설명을 입력합니다.
    b. Identifier(식별자)에 식별자를 입력합니다. 서비스 ID를 적어 둡니다. Apple을 자격 증명 풀의 공급자로 구성하려면 이 값이 필요합니다.
    c. Sign In with Apple을 선택한 다음 구성을 선택합니다.

d. 웹 인증 구성(Web Authentication Configuration) 페이지에서 기본 앱 ID(Primary App ID)를 선택합니다. 웹 사이트 URL(Website URLs)에서 + 아이콘을 선택합니다. Domains and Subdomains(도메인 및 하위 도메인)에 앱의 도메인 이름을 입력합니다. 반환 URL(Return URLs)에 Apple로 로그인을 통해 사용자를 인증한 후 권한 부여가 사용자를 리디렉션하는 콜백 URL을 입력합니다.

e. 다음을 선택합니다.

f. 계속(Continue), 등록(Register)을 차례로 선택합니다.

13. 왼쪽 탐색 창에서 키를 선택합니다.
14. 키(Keys) 페이지에 + 아이콘을 선택합니다.
15. 새 키 등록(Register a New Key) 페이지에서 다음을 수행합니다.

a. 키 이름에서 키 이름을 입력합니다.

b. Apple로 로그인(Sign In with Apple)을 선택한 다음 구성(Configure)을 선택합니다.

c. Configure Key(키 구성) 페이지에서 Primary App ID(기본 앱 ID)를 선택하고 저장을 선택합니다.

d. 계속(Continue), 등록(Register)을 차례로 선택합니다.

> **Note**
>
> Apple로 로그인을 네이티브 iOS 애플리케이션과 통합하려면 Implementing User Authentication with Sign in with Apple(Apple로 로그인으로 사용자 인증 구현)을 참조하세요. Apple로 로그인을 네이티브 iOS 이외의 플랫폼에서 통합하려면 Sign in with Apple JS(Apple JS로 로그인)를 참조하세요.

Amazon Cognito 페더레이션 자격 증명 콘솔에서 외부 공급자 구성

다음 절차를 사용하여 외부 공급자를 구성합니다.

Sign in with Apple ID 제공업체(idP)를 추가하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. ID 제공업체 추가를 선택합니다.
4. Sign in with Apple을 선택합니다.

5. Apple 개발자와 함께 생성한 OAuth 프로젝트의 서비스 ID를 입력합니다. 자세한 내용은 Sign in with Apple 설명서의 Sign in with Apple로 사용자 인증을 참조하세요.
6. Amazon Cognito가 이 공급자를 통해 인증된 사용자에게 보안 인증을 발급할 때 요청하는 역할을 설정하려면 역할 설정을 구성하세요.
   - 인증된 역할을 구성할 때 설정한 기본 역할을 이 IdP의 사용자에게 할당하거나 규칙을 사용하여 역할 선택을 선택할 수 있습니다.
     i. 규칙을 사용하여 역할 선택을 선택한 경우 사용자 인증의 소스 클레임, 클레임 비교 기준으로 사용할 운영자, 이 역할 선택과 일치하도록 하는 값, 역할 할당이 일치할 때 할당할 역할을 입력합니다. 다른 조건에 따라 추가 규칙을 생성하려면 다른 항목 추가를 선택합니다.
     ii. 역할 해결을 선택합니다. 사용자의 클레임이 규칙과 일치하지 않는 경우 보안 인증을 거부하거나 인증된 역할의 보안 인증을 발급할 수 있습니다.
7. Amazon Cognito가 이 공급자를 통해 인증한 사용자에게 보안 인증을 발급할 때 할당하는 보안 주체 태그를 변경하려면 액세스 제어를 위한 속성을 구성합니다.
   a. 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.
   b. sub 및 aud 클레임 기반 보안 주체 태그를 적용하려면 기본 매핑 사용을 선택합니다.
   c. 보안 주체 태그에 대한 속성의 자체 사용자 지정 스키마를 생성하려면 사용자 지정 매핑 사용을 선택합니다. 그런 다음 태그에 표시하려는 각 클레임에서 소싱하려는 태그 키를 입력합니다.
8. 변경 사항 저장(Save changes)을 선택합니다.

Amazon Cognito 페더레이션 자격 증명 CLI에서 Apple을 공급자로 사용하여 로그인 예제

이 예에서는 Apple로 로그인을 IdP로 사용하여 `MyIdentityPool`이라는 자격 증명 풀을 생성합니다.

```
aws cognito-identity create-identity-pool --identity-
pool-name MyIdentityPool --supported-login-providers
appleid.apple.com="sameple.apple.clientid"
```

자세한 내용은 자격 증명 풀 생성 섹션을 참조하세요.

Amazon Cognito 자격 증명 ID 생성

이 예에서는 Amazon Cognito ID를 생성(또는 검색)합니다. 이는 퍼블릭 API이므로 이 API를 호출하는 데 자격 증명이 필요하지 않습니다.

```
aws cognito-identity get-id --identity-pool-id SampleIdentityPoolId --
logins appleid.apple.com="SignInWithAppleIdToken"
```

자세한 내용은 get-id 섹션을 참조하세요.

Amazon Cognito 자격 증명 ID의 자격 증명 가져오기

이 예제에서는 제공된 자격 증명 ID 및 Sign in with Apple 로그인의 자격 증명을 반환합니다. 이는 퍼블릭 API이므로 이 API를 호출하는 데 자격 증명이 필요하지 않습니다.

```
aws cognito-identity get-credentials-for-identity --identity-id
SampleIdentityId --logins appleid.apple.com="SignInWithAppleIdToken"
```

자세한 내용은 을 참조하십시오. get-credentials-for-identity

Sign in with Apple 사용: Android

Apple은 Android용 Sign in with Apple을 지원하는 SDK를 제공하지 않습니다. 대신 웹 보기에서 웹 흐름을 사용할 수 있습니다.

- 애플리케이션에서 Apple로 로그인을 구성하려면 Apple 설명서에서 Configuring Your Web page for Sign In with Apple(Apple로 로그인용 웹 페이지 구성)을 따르세요.
- Apple로 로그인(Sign in with Apple) 버튼을 Android 사용자 인터페이스에 추가하려면 Apple 설명서에서 Displaying and Configuring Sign In with Apple Buttons(Apple로 로그인 버튼 표시 및 구성)를 따르세요.
- Apple로 로그인을 사용하여 사용자를 안전하게 인증하려면 Apple 설명서에서 Authenticating Users with Sign in with Apple(Apple로 로그인으로 사용자 인증)을 따르세요.

세션 객체를 사용하여 Apple로 로그인하여 상태를 추적합니다. Amazon Cognito는 이 세션 객체의 ID 토큰을 사용하여 사용자를 인증하고, 고유 식별자를 생성하고, 필요한 경우 사용자에게 다른 리소스에 대한 액세스 권한을 부여합니다. AWS

```
@Override
public void onSuccess(Bundle response) {
    String token = response.getString("id_token");
    Map<String, String> logins = new HashMap<String, String>();
```

```
    logins.put("appleid.apple.com", token);
    credentialsProvider.setLogins(logins);
}
```

Sign in with Apple 사용: iOS - Objective-C

Apple은 네이티브 iOS 애플리케이션에서 Sign in with Apple에 대한 SDK 지원을 제공했습니다. 네이티브 iOS 디바이스에서 Apple로 로그인으로 사용자 인증을 구현하려면 Apple 설명서에서 Implementing User Authentication with Sign in with Apple(Apple로 로그인으로 사용자 인증 구현)을 따르세요.

Amazon Cognito는 ID 토큰을 사용하여 사용자를 인증하고, 고유 식별자를 생성하고, 필요한 경우 사용자에게 다른 리소스에 대한 액세스 권한을 부여합니다. AWS

```
(void)finishedWithAuth: (ASAuthorizationAppleIDCredential *)auth error: (NSError *)
 error {
        NSString *idToken = [ASAuthorizationAppleIDCredential
 objectForKey:@"identityToken"];
        credentialsProvider.logins = @{ "appleid.apple.com": idToken };
    }
```

Sign in with Apple 사용: iOS - Swift

Apple은 네이티브 iOS 애플리케이션에서 Sign in with Apple에 대한 SDK 지원을 제공했습니다. 네이티브 iOS 디바이스에서 Apple로 로그인으로 사용자 인증을 구현하려면 Apple 설명서에서 Implementing User Authentication with Sign in with Apple(Apple로 로그인으로 사용자 인증 구현)을 따르세요.

Amazon Cognito는 ID 토큰을 사용하여 사용자를 인증하고, 고유 식별자를 생성하고, 필요한 경우 사용자에게 다른 리소스에 대한 액세스 권한을 부여합니다. AWS

iOS에서 Apple로 로그인 설정에 대한 자세한 내용은 Set up Sign in with Apple(Apple로 로그인 설정)을 참조하세요.

```
func finishedWithAuth(auth: ASAuthorizationAppleIDCredential!, error: NSError!) {
    if error != nil {
      print(error.localizedDescription)
    }
    else {
      let idToken = auth.identityToken,
      credentialsProvider.logins = ["appleid.apple.com": idToken!]
    }
```

```
}
```

Apple로 로그인 사용: JavaScript

Apple은 Apple로 로그인을 지원하는 SDK를 제공하지 않습니다. JavaScript 대신 웹 보기에서 웹 흐름을 사용할 수 있습니다.

- 애플리케이션에서 Apple로 로그인을 구성하려면 Apple 설명서에서 Configuring Your Web page for Sign In with Apple(Apple로 로그인용 웹 페이지 구성)을 따르세요.
- JavaScript 사용자 인터페이스에 Apple로 로그인 버튼을 추가하려면 Apple 설명서에서 Apple 버튼으로 로그인 표시 및 구성을 따르십시오.
- Apple로 로그인을 사용하여 사용자를 안전하게 인증하려면 Apple 설명서에서 Configuring Your Web page for Sign In with Apple(Apple로 로그인용 웹 페이지 구성)을 따르세요.

세션 객체를 사용하여 Apple로 로그인하여 상태를 추적합니다. Amazon Cognito는 이 세션 객체의 ID 토큰을 사용하여 사용자를 인증하고, 고유 식별자를 생성하고, 필요한 경우 사용자에게 다른 리소스에 대한 액세스 권한을 부여합니다. AWS

```
function signinCallback(authResult) {
     // Add the apple's id token to the Amazon Cognito credentials login map.
     AWS.config.credentials = new AWS.CognitoIdentityCredentials({
        IdentityPoolId: 'IDENTITY_POOL_ID',
        Logins: {
           'appleid.apple.com': authResult['id_token']
        }
     });

     // Obtain AWS credentials
     AWS.config.credentials.get(function(){
        // Access AWS resources here.
     });
}
```

Sign in with Apple 사용: Xamarin

Xamarin용 Sign in with Apple을 지원하는 SDK는 없습니다. 대신 웹 보기에서 웹 흐름을 사용할 수 있습니다.

- 애플리케이션에서 Apple로 로그인을 구성하려면 Apple 설명서에서 Configuring Your Web page for Sign In with Apple(Apple로 로그인용 웹 페이지 구성)을 따르세요.

- Apple로 로그인(Sign in with Apple) 버튼을 Xamarin 사용자 인터페이스에 추가하려면 Apple 설명서에서 Displaying and Configuring Sign In with Apple Buttons(Apple로 로그인 버튼 표시 및 구성)를 따르세요.
- Apple로 로그인을 사용하여 사용자를 안전하게 인증하려면 Apple 설명서에서 Apple로 로그인용 웹 페이지 구성(Configuring Your Web page for Sign In with Apple)을 따르세요.

세션 객체를 사용하여 Apple로 로그인하여 상태를 추적합니다. Amazon Cognito는 이 세션 객체의 ID 토큰을 사용하여 사용자를 인증하고, 고유 식별자를 생성하고, 필요한 경우 사용자에게 다른 리소스에 대한 액세스 권한을 부여합니다. AWS

토큰을 받으면 `CognitoAWSCredentials`에서 토큰을 설정할 수 있습니다.

```
credentials.AddLogin("appleid.apple.com", token);
```

OIDC 공급자를 자격 증명 풀 IdP로 설정

OpenID Connect는 여러 로그인 공급자가 지원하는 인증에 대한 개방형 표준입니다. Amazon Cognito는 AWS Identity and Access Management를 통해 구성한 OpenID Connect 공급자와 자격 증명을 연결할 수 있도록 지원합니다.

OpenID Connect 공급자 추가

OpenID Connect 공급자를 생성하는 방법에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 OIDC(OpenID Connect) ID 제공업체 생성을 참조하세요.

Amazon Cognito와 공급자 연결

OIDC ID 제공업체(idP)를 생성하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. ID 제공업체 추가를 선택합니다.
4. OpenId Connect(OIDC)를 선택합니다.
5. 내 IAM에서 OIDC 자격 증명 공급자를 선택하십시오. IdPs AWS 계정새 SAML 공급자를 추가하려면 새 공급자 생성을 선택하여 IAM 콘솔로 이동합니다.
6. Amazon Cognito가 이 공급자를 통해 인증된 사용자에게 보안 인증을 발급할 때 요청하는 역할을 설정하려면 역할 설정을 구성하세요.

- 인증된 역할을 구성할 때 설정한 기본 역할을 이 IdP의 사용자에게 할당하거나 규칙을 사용하여 역할 선택을 선택할 수 있습니다.

  i. 규칙을 사용하여 역할 선택을 선택한 경우 사용자 인증의 소스 클레임, 클레임 비교 기준으로 사용할 운영자, 이 역할 선택과 일치하도록 하는 값, 역할 할당이 일치할 때 할당할 역할을 입력합니다. 다른 조건에 따라 추가 규칙을 생성하려면 다른 항목 추가를 선택합니다.

  ii. 역할 해결을 선택합니다. 사용자의 클레임이 규칙과 일치하지 않는 경우 보안 인증을 거부하거나 인증된 역할의 보안 인증을 발급할 수 있습니다.

7. Amazon Cognito가 이 공급자를 통해 인증한 사용자에게 보안 인증을 발급할 때 할당하는 보안 주체 태그를 변경하려면 액세스 제어를 위한 속성을 구성합니다.

   a. 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.

   b. sub 및 aud 클레임 기반 보안 주체 태그를 적용하려면 기본 매핑 사용을 선택합니다.

   c. 보안 주체 태그에 대한 속성의 자체 사용자 지정 스키마를 생성하려면 사용자 지정 매핑 사용을 선택합니다. 그런 다음 태그에 표시하려는 각 클레임에서 소싱하려는 태그 키를 입력합니다.

8. 변경 사항 저장(Save changes)을 선택합니다.

여러 OpenID Connect 공급자를 단일 자격 증명 풀에 연결할 수 있습니다.

OpenID Connect 사용

로그인하고 ID 토큰을 받는 방법은 공급자 설명서를 참조하세요.

토큰을 받은 후에는 토큰을 로그인 맵에 추가합니다. 공급자의 URI를 키로 사용합니다.

OpenID Connect 토큰 검증

Amazon Cognito와 처음 통합할 때 `InvalidToken` 예외가 발생할 수 있습니다. Amazon Cognito가 OpenID Connect(OIDC) 토큰을 검증하는 방법을 이해해야 합니다.

> **Note**
>
> 여기(https://tools.ietf.org/html/rfc7523)에 명시된 바와 같이 Amazon Cognito는 시스템 간의 클럭 스큐를 처리하는 5분의 유예 기간을 제공합니다.

1. iss 파라미터가 로그인 맵에 사용된 키(예: login.provider.com)와 일치해야 합니다.
2. 서명이 유효해야 합니다. RSA 퍼블릭 키를 통해 서명을 확인할 수 있어야 합니다.
3. 인증서 퍼블릭 키의 지문은 OIDC 공급자를 생성할 때 IAM에서 설정한 지문과 일치합니다.
4. azp 파라미터가 있으면 이 값을 OIDC 공급자에서 나열된 클라이언트 ID와 비교하여 확인합니다.
5. azp 파라미터가 없으면 aud 파라미터를 OpenId Connect 공급자에서 나열된 클라이언트 ID와 비교하여 확인합니다.

jwt.io 웹 사이트는 토큰을 디코딩하고 이러한 값을 확인하는 데 사용할 수 있는 중요한 리소스입니다.

Android

```
Map<String, String> logins = new HashMap<String, String>();
logins.put("login.provider.com", token);
credentialsProvider.setLogins(logins);
```

iOS - Objective-C

```
credentialsProvider.logins = @{ "login.provider.com": token }
```

iOS - Swift

OIDC ID 토큰을 Amazon Cognito에 제공하려면 AWSIdentityProviderManager 프로토콜을 구현합니다.

logins 메서드를 구현할 때 구성한 OIDC 공급자 이름이 포함된 사전을 반환합니다. 다음 코드 예와 같이 이 사전은 키 역할을 하고 인증된 사용자에게서 받은 현재 ID 토큰은 값 역할을 합니다.

```
class OIDCProvider: NSObject, AWSIdentityProviderManager {
    func logins() -> AWSTask<NSDictionary> {
        let completion = AWSTaskCompletionSource<NSString>()
        getToken(tokenCompletion: completion)
        return completion.task.continueOnSuccessWith { (task) -> AWSTask<NSDictionary>?
 in
            //login.provider.name is the name of the OIDC provider as setup in the
 Amazon Cognito console
            return AWSTask(result:["login.provider.name":task.result!])
        } as! AWSTask<NSDictionary>
```

```
    }

    func getToken(tokenCompletion: AWSTaskCompletionSource<NSString>) -> Void {
        //get a valid oidc token from your server, or if you have one that hasn't
 expired cached, return it

        //TODO code to get token from your server
        //...

        //if error getting token, set error appropriately
        tokenCompletion.set(error:NSError(domain: "OIDC Login", code: -1 , userInfo:
 ["Unable to get OIDC token" : "Details about your error"]))
        //else
        tokenCompletion.set(result:"result from server id token")
    }
}
```

를 인스턴스화할 때 `AWSCognitoCredentialsProvider` 생성자에서 의 AWSIdentityProviderManager 값으로 구현하는 클래스를 전달하십시오. identityProviderManager 자세한 내용을 보려면 AWSCognitoCredentialsProvider참조 페이지로 이동하여 Type:: 을 선택하십시오 initWithRegion. identityPoolId identityProviderManager

JavaScript

```
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
 IdentityPoolId: 'IDENTITY_POOL_ID',
 Logins: {
    'login.provider.com': token
 }
});
```

Unity

```
credentials.AddLogin("login.provider.com", token);
```

Xamarin

```
credentials.AddLogin("login.provider.com", token);
```

SAML 공급자를 자격 증명 풀 IdP로 설정

Amazon Cognito는 보안 어설션 마크업 언어 2.0 (SAML 2.0IdPs) 을 통해 자격 증명 공급자 () 를 통한 인증을 지원합니다. Amazon Cognito에서 SAML을 지원하는 IdP를 사용하여 사용자를 위한 간단한 온보딩 흐름을 제공할 수 있습니다. SAML을 지원하는 IdP는 사용자가 수임할 수 있는 IAM 역할을 지정합니다. 이렇게 하면 사용자마다 서로 다른 권한 집합을 받을 수 있습니다.

SAML IdP에 맞게 자격 증명 풀 구성

다음 단계에서는 자격 증명 풀에서 SAML 기반 IdP를 사용하도록 구성하는 방법을 설명합니다.

> **Note**
>
> 자격 증명 풀에서 SAML 공급자를 지원하도록 구성하기 전에 먼저 IAM 콘솔에서 SAML IdP를 구성합니다. 자세한 내용은 IAM 사용 설명서에서 AWS와 서드 파티 SAML 솔루션 공급자 통합을 참조하세요.

SAML ID 제공업체(idP)를 추가하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. ID 제공업체 추가를 선택합니다.
4. SAML을 선택합니다.
5. 내 IAM에서 SAML 자격 증명 공급자를 선택하십시오. IdPs AWS 계정새 SAML 공급자를 추가하려면 새 공급자 생성을 선택하여 IAM 콘솔로 이동합니다.
6. Amazon Cognito가 이 공급자를 통해 인증된 사용자에게 보안 인증을 발급할 때 요청하는 역할을 설정하려면 역할 설정을 구성하세요.

   - 인증된 역할을 구성할 때 설정한 기본 역할을 이 IdP의 사용자에게 할당하거나 규칙을 사용하여 역할 선택을 선택할 수 있습니다.

     i. 규칙을 사용하여 역할 선택을 선택한 경우 사용자 인증의 소스 클레임, 클레임 비교 기준으로 사용할 운영자, 이 역할 선택과 일치하도록 하는 값, 역할 할당이 일치할 때 할당할 역할을 입력합니다. 다른 조건에 따라 추가 규칙을 생성하려면 다른 항목 추가를 선택합니다.

     ii. 역할 해결을 선택합니다. 사용자의 클레임이 규칙과 일치하지 않는 경우 보안 인증을 거부하거나 인증된 역할의 보안 인증을 발급할 수 있습니다.

7. Amazon Cognito가 이 공급자를 통해 인증한 사용자에게 보안 인증을 발급할 때 할당하는 보안 주체 태그를 변경하려면 액세스 제어를 위한 속성을 구성합니다.

   a. 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.

   b. sub 및 aud 클레임 기반 보안 주체 태그를 적용하려면 기본 매핑 사용을 선택합니다.

   c. 보안 주체 태그에 대한 속성의 자체 사용자 지정 스키마를 생성하려면 사용자 지정 매핑 사용을 선택합니다. 그런 다음 태그에 표시하려는 각 클레임에서 소싱하려는 태그 키를 입력합니다.

8. 변경 사항 저장(Save changes)을 선택합니다.

SAML IdP 구성

SAML 공급자를 생성한 후에 SAML IdP에서 IdP와 AWS간에 신뢰 당사자 신뢰를 추가하도록 구성합니다. 대부분의 IdPs 경우 IdP가 XML 문서에서 신뢰 당사자 정보와 인증서를 읽는 데 사용할 수 있는 URL을 지정할 수 있습니다. 의 AWS경우 https://signin.aws.amazon.com/static/saml-metadata.xml 을 사용할 수 있습니다. 다음 단계는 필요한 클레임을 채우도록 IdP의 SAML 어설션 응답을 구성하는 것입니다. AWS 클레임 구성에 대한 자세한 내용은 인증 응답을 위한 SAML 어설션 구성을 참조하세요.

SAML IdP가 SAML 메타데이터에 서명 인증서를 두 개 이상 포함하는 경우, 로그인 시 사용자 풀은 SAML 어설션이 SAML 메타데이터에 있는 인증서와 일치할 경우 유효하다고 판단합니다.

SAML로 사용자 역할 사용자 정의

Amazon Cognito Identity와 함께 SAML을 사용하면 최종 사용자에 맞게 역할을 사용자 지정할 수 있습니다. Amazon Cognito는 향상된 흐름을 SAML 기반 IdP를 통해서만 지원합니다. 자격 증명 풀에서 SAML 기반 IdP를 사용할 수 있도록 인증된 역할 또는 인증되지 않은 역할을 지정할 필요는 없습니다. `https://aws.amazon.com/SAML/Attributes/Role` 클레임 속성은 쉼표로 구분된 하나 이상의 역할 및 공급자 ARN 쌍을 지정합니다. 이는 사용자가 수임할 수 있는 역할입니다. SAML IdP가 IdP에서 얻을 수 있는 사용자 속성 정보를 기반으로 역할 속성을 채우도록 구성할 수 있습니다. SAML 어설션에서 여러 역할을 받으면 `getCredentialsForIdentity`를 호출하는 동안 선택적 `customRoleArn` 파라미터를 채웁니다. 역할이 SAML 어설션의 클레임에 있는 역할과 일치하는 경우 사용자가 이 `customRoleArn`을 수임합니다.

SAML IdP를 사용하여 사용자 인증

SAML 기반 IdP와 페더레이션하려면 사용자가 로그인을 시작하는 URL을 결정하십시오. AWS 페더레이션은 IdP에서 시작한 로그인을 사용합니다. AD FS 2.0에서 URL 형식은 `https://<fqdn>/adfs/ls/IdpInitiatedSignOn.aspx?loginToRp=urn:amazon:webservices`입니다.

Amazon Cognito에서 SAML IdP에 대한 지원을 추가하려면 먼저 iOS나 Android 애플리케이션에서 SAML 자격 증명 공급자로 사용자를 인증합니다. SAML IdP와 통합하고 인증하는 데 사용하는 코드는 SAML 공급자에 따라 다릅니다. 사용자가 인증된 후에 Amazon Cognito API를 사용하여 Amazon Cognito Identity에 결과 SAML 어설션을 제공할 수 있습니다.

자격 증명 풀 API 요청의`Logins` 맵에서 SAML 어설션을 반복하거나 재생할 수 없습니다. 재생된 SAML 어설션에는 이전 API 요청의 ID와 중복되는 어설션 ID가 있습니다. `Logins`맵에서 SAML 어설션을 수락할 수 있는 API 작업에는,, ID가 포함됩니다 GetId. GetCredentialsForIdentityGetOpenIdTokenGetOpenTokenForDeveloperIdentity 자격 증명 풀 인증 흐름에서 API 요청당 SAML 어설션 ID를 한 번 재생할 수 있습니다. 예를 들어 `GetId` 요청과 후속 `GetCredentialsForIdentity` 요청에는 동일한 SAML 어설션을 제공할 수 있지만 두 번째 `GetId` 요청에서는 제공할 수 없습니다.

Android

Android SDK를 사용하는 경우 다음과 같이 로그인 맵을 SAML 어설션으로 채울 수 있습니다.

```
Map logins = new HashMap();
logins.put("arn:aws:iam::aws account id:saml-provider/name", "base64 encoded assertion
 response");
// Now this should be set to CognitoCachingCredentialsProvider object.
CognitoCachingCredentialsProvider credentialsProvider = new
 CognitoCachingCredentialsProvider(context, identity pool id, region);
credentialsProvider.setLogins(logins);
// If SAML assertion contains multiple roles, resolve the role by setting the custom
 role
credentialsProvider.setCustomRoleArn("arn:aws:iam::aws account id:role/
customRoleName");
// This should trigger a call to the Amazon Cognito service to get the credentials.
credentialsProvider.getCredentials();
```

iOS

iOS SDK를 사용하는 경우 다음과 같이 `AWSIdentityProviderManager`에서 SAML 어설션을 제공할 수 있습니다.

```
- (AWSTask<NSDictionary<NSString*,NSString*> *> *) logins {
    //this is hardcoded for simplicity, normally you would asynchronously go to your
 SAML provider
    //get the assertion and return the logins map using a AWSTaskCompletionSource
```

```
    return [AWSTask taskWithResult:@{@"arn:aws:iam::aws account id:saml-provider/
name":@"base64 encoded assertion response"}];
}

// If SAML assertion contains multiple roles, resolve the role by setting the custom
 role.
// Implementing this is optional if there is only one role.
- (NSString *)customRoleArn {
    return @"arn:aws:iam::accountId:role/customRoleName";
}
```

개발자 인증 자격 증명(자격 증명 풀)

Amazon Cognito는 페이스북을 아이덴티티 풀 IdP로 설정하기, 구글을 아이덴티티 풀 IdP로 설정하기, Amazon으로 로그인을 자격 증명 풀 IdP로 설정, Apple을 자격 증명 풀 IdP로 사용하여 로그인 설정하기을 통한 웹 아이덴티티 페더레이션 외에 개발자 인증 자격 증명을 지원합니다. 개발자 인증 자격 증명을 사용하면 Amazon Cognito를 사용하여 사용자 데이터를 동기화하고 리소스에 액세스하면서 기존 인증 프로세스를 통해 사용자를 등록 및 인증할 수 있습니다. AWS 개발자 인증 자격 증명을 사용하려면 최종 사용자 디바이스, 인증을 위한 백엔드, Amazon Cognito 간의 상호 작용이 필요합니다. 자세한 내용은 블로그에서 Amazon Cognito 인증의 이해 2부: 개발자 인증 자격 증명을 참조하십시오. AWS

인증 흐름 이해

GetOpenIdTokenForDeveloperIdentityAPI 작업을 통해 고급 인증과 기본 인증 모두에 대한 개발자 인증을 시작할 수 있습니다. 이 API는 관리 자격 증명으로 요청을 인증합니다. `Logins`맵은 사용자 지정 식별자와 `login.mydevprovider` 짝을 이루는 것과 같은 자격 증명 풀 개발자 공급자 이름입니다.

예제

```
"Logins": {
        "login.mydevprovider": "my developer identifier"
    }
```

향상된 인증

토큰의 `cognito-identity.amazonaws.com` 이름과 값이 `Logins` 들어 있는 맵을 사용하여 GetCredentialsForIdentityAPI 작업을 `GetOpenIdTokenForDeveloperIdentity` 호출합니다.

예제

```
"Logins": {
        "cognito-identity.amazonaws.com": "eyJra12345EXAMPLE"
    }
```

`GetCredentialsForIdentity`개발자 인증 ID를 사용하면 자격 증명 풀의 기본 인증 역할에 대한 임시 자격 증명이 반환됩니다.

기본 인증

AssumeRoleWithWebIdentityAPI 작업을 호출하고 적절한 신뢰 관계가 `RoleArn` 정의된 IAM 역할을 요청합니다. 의 값을 에서 `GetOpenIdTokenForDeveloperIdentity` 가져온 `WebIdentityToken` 토큰으로 설정합니다.

개발자 인증 ID 인증 인증 흐름 및 외부 공급자 ID와 어떻게 다른지에 대한 자세한 내용은 을 참조하십시오. 자격 증명 풀(페더레이션 자격 증명) 인증 흐름

개발자 공급자 이름 정의 및 자격 증명 풀에 연결

개발자 인증 자격 증명을 사용하려면 개발자 공급자에 연결된 자격 증명 풀이 필요합니다. 이렇게 하려면 다음 단계를 따릅니다.

사용자 지정 개발자 공급자를 추가하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. ID 제공업체 추가를 선택합니다.
4. 사용자 지정 개발자 공급자를 선택합니다.
5. 개발자 공급자 이름을 입력합니다. 개발자 공급자는 추가 후에는 변경 또는 제거할 수 없습니다.
6. 변경 사항 저장(Save changes)을 선택합니다.

참고: 공급자 이름을 설정하면 해당 이름을 변경할 수 없습니다.

Amazon Cognito 콘솔 작업에 대한 추가 지침은 Amazon Cognito 콘솔 사용 섹션을 참조하세요.

자격 증명 공급자 구현

Android

개발자 인증 자격 증명을 사용하려면 AWSAbstractCognitoIdentityProvider를 확장하는 고유한 ID 제공업체 클래스를 구현합니다. 자격 증명 공급자 클래스는 토큰이 속성으로 포함된 응답 객체를 반환해야 합니다.

다음은 ID 제공업체의 기본 예입니다.

```
public class DeveloperAuthenticationProvider extends
 AWSAbstractCognitoDeveloperIdentityProvider {

  private static final String developerProvider = "<Developer_provider_name>";

  public DeveloperAuthenticationProvider(String accountId, String identityPoolId,
 Regions region) {
    super(accountId, identityPoolId, region);
    // Initialize any other objects needed here.
  }

  // Return the developer provider name which you choose while setting up the
  // identity pool in the &COG; Console

  @Override
  public String getProviderName() {
    return developerProvider;
  }

  // Use the refresh method to communicate with your backend to get an
  // identityId and token.

  @Override
  public String refresh() {

    // Override the existing token
    setToken(null);

    // Get the identityId and token by making a call to your backend
    // (Call to your backend)

    // Call the update method with updated identityId and token to make sure
    // these are ready to be used from Credentials Provider.
```

```
    update(identityId, token);
    return token;

  }

  // If the app has a valid identityId return it, otherwise get a valid
  // identityId from your backend.

  @Override
  public String getIdentityId() {

    // Load the identityId from the cache
    identityId = cachedIdentityId;

    if (identityId == null) {
       // Call to your backend
    } else {
       return identityId;
    }

  }
}
```

이 자격 증명 공급자를 사용하려면 해당 공급자를 CognitoCachingCredentialsProvider에 전달해야 합니다. 다음은 그 예입니다.

```
DeveloperAuthenticationProvider developerProvider = new
 DeveloperAuthenticationProvider( null, "IDENTITYPOOLID", context, Regions.USEAST1);
CognitoCachingCredentialsProvider credentialsProvider = new
 CognitoCachingCredentialsProvider( context, developerProvider, Regions.USEAST1);
```

iOS - Objective-C

개발자 인증 자격 증명을 사용하려면 AWSCognitoCredentialsProviderHelper를 확장하는 고유한 ID 제공업체 클래스를 구현합니다. 자격 증명 공급자 클래스는 토큰이 속성으로 포함된 응답 객체를 반환해야 합니다.

```
@implementation DeveloperAuthenticatedIdentityProvider
/*
 * Use the token method to communicate with your backend to get an
 * identityId and token.
```

```
 */

- (AWSTask <NSString*> *) token {
    //Write code to call your backend:
    //Pass username/password to backend or some sort of token to authenticate user
    //If successful, from backend call getOpenIdTokenForDeveloperIdentity with logins
 map
    //containing "your.provider.name":"enduser.username"
    //Return the identity id and token to client
    //You can use AWSTaskCompletionSource to do this asynchronously

    // Set the identity id and return the token
    self.identityId = response.identityId;
    return [AWSTask taskWithResult:response.token];
}

@end
```

이 자격 증명 공급자를 사용하려면 다음 예에 표시된 대로 해당 공급자를 `AWSCognitoCredentialsProvider`에 전달합니다.

```
DeveloperAuthenticatedIdentityProvider * devAuth =
 [[DeveloperAuthenticatedIdentityProvider alloc]
 initWithRegionType:AWSRegionYOUR_IDENTITY_POOL_REGION
                                          identityPoolId:@"YOUR_IDENTITY_POOL_ID"
                                         useEnhancedFlow:YES
                                identityProviderManager:nil];
AWSCognitoCredentialsProvider *credentialsProvider = [[AWSCognitoCredentialsProvider
 alloc]

 initWithRegionType:AWSRegionYOUR_IDENTITY_POOL_REGION
                                                        identityProvider:devAuth];
```

미인증 자격 증명과 개발자 인증 자격 증명을 모두 지원하려면 `AWSCognitoCredentialsProviderHelper` 구현에서 `logins` 메서드를 무시합니다.

```
- (AWSTask<NSDictionary<NSString *, NSString *> *> *)logins {
    if(/*logic to determine if user is unauthenticated*/) {
        return [AWSTask taskWithResult:nil];
    }else{
        return [super logins];
    }
```

```
}
```

개발자 인증 자격 증명 및 소셜 공급자를 지원하려면 `AWSCognitoCredentialsProviderHelper`의 `logins` 구현에서 현재 공급자가 누구인지 관리해야 합니다.

```
- (AWSTask<NSDictionary<NSString *, NSString *> *> *)logins {
    if(/*logic to determine if user is unauthenticated*/) {
        return [AWSTask taskWithResult:nil];
    }else if (/*logic to determine if user is Facebook*/){
        return [AWSTask taskWithResult: @{ AWSIdentityProviderFacebook :
 [FBSDKAccessToken currentAccessToken] }];
    }else {
        return [super logins];
    }
}
```

iOS - Swift

개발자 인증 자격 증명을 사용하려면 AWSCognitoCredentialsProviderHelper를 확장하는 고유한 ID 제공업체 클래스를 구현합니다. 자격 증명 공급자 클래스는 토큰이 속성으로 포함된 응답 객체를 반환해야 합니다.

```
import AWSCore
/*
 * Use the token method to communicate with your backend to get an
 * identityId and token.
 */
class DeveloperAuthenticatedIdentityProvider : AWSCognitoCredentialsProviderHelper {
    override func token() -> AWSTask<NSString> {
    //Write code to call your backend:
    //pass username/password to backend or some sort of token to authenticate user, if
 successful,
    //from backend call getOpenIdTokenForDeveloperIdentity with logins map containing
 "your.provider.name":"enduser.username"
    //return the identity id and token to client
    //You can use AWSTaskCompletionSource to do this asynchronously

    // Set the identity id and return the token
    self.identityId = resultFromAbove.identityId
    return AWSTask(result: resultFromAbove.token)
}
```

이 자격 증명 공급자를 사용하려면 다음 예에 표시된 대로 해당 공급자를 AWSCognitoCredentialsProvider에 전달합니다.

```
let devAuth =
 DeveloperAuthenticatedIdentityProvider(regionType: .YOUR_IDENTITY_POOL_REGION,
 identityPoolId: "YOUR_IDENTITY_POOL_ID", useEnhancedFlow: true,
 identityProviderManager:nil)
let credentialsProvider =
 AWSCognitoCredentialsProvider(regionType: .YOUR_IDENTITY_POOL_REGION,
 identityProvider:devAuth)
let configuration = AWSServiceConfiguration(region: .YOUR_IDENTITY_POOL_REGION,
 credentialsProvider:credentialsProvider)
AWSServiceManager.default().defaultServiceConfiguration = configuration
```

미인증 자격 증명과 개발자 인증 자격 증명을 모두 지원하려면 AWSCognitoCredentialsProviderHelper 구현에서 logins 메서드를 무시합니다.

```
override func logins () -> AWSTask<NSDictionary> {
    if(/*logic to determine if user is unauthenticated*/) {
        return AWSTask(result:nil)
    }else {
        return super.logins()
    }
}
```

개발자 인증 자격 증명 및 소셜 공급자를 지원하려면 AWSCognitoCredentialsProviderHelper의 logins 구현에서 현재 공급자가 누구인지 관리해야 합니다.

```
override func logins () -> AWSTask<NSDictionary> {
    if(/*logic to determine if user is unauthenticated*/) {
        return AWSTask(result:nil)
    }else if (/*logic to determine if user is Facebook*/){
        if let token = AccessToken.current?.authenticationToken {
            return AWSTask(result: [AWSIdentityProviderFacebook:token])
        }
        return AWSTask(error:NSError(domain: "Facebook Login", code: -1 , userInfo:
 ["Facebook" : "No current Facebook access token"]))
    }else {
        return super.logins()
    }
```

```
}
```

JavaScript

백엔드에서 자격 증명 ID 및 세션 토큰을 가져온 경우 이를 AWS.CognitoIdentityCredentials 공급자에 전달합니다. 다음은 그 예입니다.

```
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
   IdentityPoolId: 'IDENTITY_POOL_ID',
   IdentityId: 'IDENTITY_ID_RETURNED_FROM_YOUR_PROVIDER',
   Logins: {
      'cognito-identity.amazonaws.com': 'TOKEN_RETURNED_FROM_YOUR_PROVIDER'
   }
});
```

Unity

개발자 인증 자격 증명을 사용하려면 CognitoAWSCredentials를 확장하고 RefreshIdentity 메서드를 무시하여 백엔드에서 사용자 자격 증명 ID와 토큰을 검색하고 이를 반환해야 합니다. 다음은 'example.com'의 가상 백엔드에 접촉하는 ID 제공업체의 간단한 예입니다.

```
using UnityEngine;
using System.Collections;
using Amazon.CognitoIdentity;
using System.Collections.Generic;
using ThirdParty.Json.LitJson;
using System;
using System.Threading;

public class DeveloperAuthenticatedCredentials : CognitoAWSCredentials
{
    const string PROVIDER_NAME = "example.com";
    const string IDENTITY_POOL = "IDENTITY_POOL_ID";
    static readonly RegionEndpoint REGION = RegionEndpoint.USEast1;

    private string login = null;

    public DeveloperAuthenticatedCredentials(string loginAlias)
        : base(IDENTITY_POOL, REGION)
    {
        login = loginAlias;
    }
```

```
    protected override IdentityState RefreshIdentity()
    {
        IdentityState state = null;
        ManualResetEvent waitLock = new ManualResetEvent(false);
        MainThreadDispatcher.ExecuteCoroutineOnMainThread(ContactProvider((s) =>
        {
            state = s;
            waitLock.Set();
        }));
        waitLock.WaitOne();
        return state;
    }

    IEnumerator ContactProvider(Action<IdentityState> callback)
    {
        WWW www = new WWW("http://example.com/?username="+login);
        yield return www;
        string response = www.text;

        JsonData json = JsonMapper.ToObject(response);

        //The backend has to send us back an Identity and a OpenID token
        string identityId = json["IdentityId"].ToString();
        string token = json["Token"].ToString();

        IdentityState state = new IdentityState(identityId, PROVIDER_NAME, token,
 false);
        callback(state);
    }
}
```

위의 코드는 스레드 디스패처 객체를 사용하여 코루틴을 호출합니다. 프로젝트에서 이 작업을 수행할 방법이 없는 경우 장면에서 다음 스크립트를 사용할 수 있습니다.

```
using System;
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

public class MainThreadDispatcher : MonoBehaviour
{
    static Queue<IEnumerator> _coroutineQueue = new Queue<IEnumerator>();
```

```
    static object _lock = new object();

    public void Update()
    {
        while (_coroutineQueue.Count > 0)
        {
            StartCoroutine(_coroutineQueue.Dequeue());
        }
    }

    public static void ExecuteCoroutineOnMainThread(IEnumerator coroutine)
    {
        lock (_lock) {
            _coroutineQueue.Enqueue(coroutine);
        }
    }
}
```

Xamarin

개발자 인증 자격 증명을 사용하려면 CognitoAWSCredentials를 확장하고 RefreshIdentity 메서드를 무시하여 백엔드에서 사용자 자격 증명 ID와 토큰을 검색하고 이를 반환해야 합니다. 다음은 'example.com'의 가상 백엔드에 접촉하는 ID 제공업체의 기본 예입니다.

```
public class DeveloperAuthenticatedCredentials : CognitoAWSCredentials
{
    const string PROVIDER_NAME = "example.com";
    const string IDENTITY_POOL = "IDENTITY_POOL_ID";
    static readonly RegionEndpoint REGION = RegionEndpoint.USEast1;
    private string login = null;

    public DeveloperAuthenticatedCredentials(string loginAlias)
        : base(IDENTITY_POOL, REGION)
    {
        login = loginAlias;
    }

    protected override async Task<IdentityState> RefreshIdentityAsync()
    {
        IdentityState state = null;
        //get your identity and set the state
        return state;
    }
```

```
}
```

로그인 맵 업데이트(Android 및 iOS 전용)

Android

인증 시스템을 통해 사용자를 성공적으로 인증한 후 개발자 공급자 이름 및 개발자 사용자 식별자로 로그인 맵을 업데이트합니다. 이것은 인증 시스템에서 사용자를 고유하게 식별하는 영숫자 문자열입니다. `refresh`가 변경되었을 수 있으므로 로그인 맵을 업데이트한 후 `identityId` 메서드를 호출해야 합니다.

```
HashMap<String, String> loginsMap = new HashMap<String, String>();
loginsMap.put(developerAuthenticationProvider.getProviderName(),
 developerUserIdentifier);

credentialsProvider.setLogins(loginsMap);
credentialsProvider.refresh();
```

iOS - Objective-C

iOS SDK는 자격 증명이 없거나 만료된 경우 `logins` 메서드만 호출하여 최신 로그인 맵을 가져옵니다. SDK에서 새 자격 증명을 얻도록 강제하려면(예를 들어 최종 사용자가 미인증 상태에서 인증 상태로 변경되었으며 인증된 사용자에 대한 자격 증명을 원하는 경우) `credentialsProvider`에 대해 `clearCredentials`를 호출하세요.

```
[credentialsProvider clearCredentials];
```

iOS - Swift

iOS SDK는 자격 증명이 없거나 만료된 경우 `logins` 메서드만 호출하여 최신 로그인 맵을 가져옵니다. SDK에서 새 자격 증명을 얻도록 강제하려면(예: 최종 사용자가 미인증 상태에서 인증 상태로 변경되었으며 인증된 사용자에 대한 자격 증명을 원하는 경우) `clearCredentials`에 대해 `credentialsProvider`를 호출하세요.

```
credentialsProvider.clearCredentials()
```

토큰 가져오기(서버 측)

를 GetOpenIdTokenForDeveloperIdentity호출하여 토큰을 얻습니다. 이 API는 AWS 개발자 자격 증명을 사용하여 백엔드에서 호출해야 합니다. 클라이언트 SDK에서 호출해서는 안 됩니다. API는 Cognito 자격 증명 풀 ID, ID 제공업체 이름이 키로, 식별자가 값으로 포함된 로그인 맵 및 필요할 경우 Cognito 자격 증명 ID를 받습니다(예를 들어 미인증 사용자를 인증된 사용자로 변경하는 경우). 식별자는 사용자의 사용자 이름, 이메일 주소 또는 숫자 값일 수 있습니다. API는 사용자에 대한 고유한 Cognito ID와 최종 사용자에 대한 OpenID Connect 토큰을 사용하여 호출에 응답합니다.

`GetOpenIdTokenForDeveloperIdentity`에 의해 반환된 토큰에 대해 유의해야 할 몇 가지 사항은 다음과 같습니다.

- 캐시할 수 있도록 토큰에 대한 사용자 지정 만료 시간을 지정할 수 있습니다. 사용자 지정 만료 시간을 제공하지 않으면 토큰은 15분 동안 유효합니다.
- 설정할 수 있는 최대 토큰 지속 시간은 24시간입니다.
- 토큰 지속 시간의 증가에 따른 보안 영향을 주의하세요. 공격자가 이 토큰을 획득하면 토큰 기간 동안 이를 최종 사용자의 AWS 자격 증명으로 교환할 수 있습니다.

다음 Java 조각은 Amazon Cognito 클라이언트를 초기화하고 개발자 인증 자격 증명의 토큰을 검색하는 방법을 보여 줍니다.

```
// authenticate your end user as appropriate
// ....

// if authenticated, initialize a cognito client with your AWS developer credentials
AmazonCognitoIdentity identityClient = new AmazonCognitoIdentityClient(
  new BasicAWSCredentials("access_key_id", "secret_access_key")
);

// create a new request to retrieve the token for your end user
GetOpenIdTokenForDeveloperIdentityRequest request =
  new GetOpenIdTokenForDeveloperIdentityRequest();
request.setIdentityPoolId("YOUR_COGNITO_IDENTITY_POOL_ID");

request.setIdentityId("YOUR_COGNITO_IDENTITY_ID"); //optional, set this if your client
 has an
                                                   //identity ID that you want to link
 to this
                                                   //developer account
```

```
// set up your logins map with the username of your end user
HashMap<String,String> logins = new HashMap<>();
logins.put("YOUR_IDENTITY_PROVIDER_NAME","YOUR_END_USER_IDENTIFIER");
request.setLogins(logins);

// optionally set token duration (in seconds)
request.setTokenDuration(60 * 15l);
GetOpenIdTokenForDeveloperIdentityResult response =
  identityClient.getOpenIdTokenForDeveloperIdentity(request);

// obtain identity id and token to return to your client
String identityId = response.getIdentityId();
String token = response.getToken();

//code to return identity id and token to client
//...
```

앞의 단계에 따라 앱에 개발자 인증 자격 증명을 통합할 수 있어야 합니다. 문제나 질문이 있는 경우 자유롭게 포럼에 게시해 주세요.

기존 소셜 자격 증명에 연결

개발자 인증 자격 증명을 사용할 때 백엔드에서 모든 공급자 연결이 수행되어야 합니다. 사용자 지정 ID를 사용자의 소셜 ID (Amazon으로 로그인, Apple, Facebook 또는 Google로 로그인) 에 연결하려면 전화를 걸 GetOpenIdTokenForDeveloperIdentity때 로그인 맵에 ID 공급자 토큰을 추가하십시오. 이렇게 하려면 클라이언트 SDK에서 백엔드를 호출하여 최종 사용자를 인증할 때 최종 사용자의 소셜 공급자 토큰을 추가로 전달하세요.

예를 들어, 사용자 지정 자격 증명을 Facebook에 연결하려고 시도할 경우 `GetOpenIdTokenForDeveloperIdentity`를 호출할 때 로그인 맵에 자격 증명 공급자 식별자 이외에 Facebook 토큰도 추가합니다.

```
logins.put("YOUR_IDENTITY_PROVIDER_NAME","YOUR_END_USER_IDENTIFIER");
logins.put("graph.facebook.com","END_USERS_FACEBOOK_ACCESSTOKEN");
```

공급자 간 전환 지원

Android

애플리케이션에는 개발자 인증 자격 증명과 함께 퍼블릭 공급자(Login with Amazon, Sign in with Apple, Facebook 또는 Google)를 사용한 미인증 자격 증명 또는 인증 자격 증명 지원이 필요할 수 있습

니다. 개발자 인증 자격 증명과 다른 자격 증명(퍼블릭 공급자를 사용하는 미인증 자격 증명 및 인증 자격 증명) 간의 본질적인 차이점은 identityId 및 토큰을 얻는 방식입니다. 다른 자격 증명의 경우 모바일 애플리케이션은 인증 시스템과 접촉하는 대신 Amazon Cognito와 직접 상호 작용합니다. 따라서 모바일 애플리케이션은 앱 사용자가 선택한 사항에 따라 두 개의 고유 흐름을 지원할 수 있어야 합니다. 이를 위해 사용자 지정 ID 제공업체를 일부 변경해야 합니다.

`refresh` 메서드는 로그인 맵을 확인합니다. 맵이 비어 있지 않고 개발자 공급자 이름이 포함된 키가 있는 경우, 백엔드를 호출합니다. 그렇지 않으면 getIdentityId 메서드를 호출하고 null을 반환합니다.

```
public String refresh() {

   setToken(null);

   // If the logins map is not empty make a call to your backend
   // to get the token and identityId
   if (getProviderName() != null &&
      !this.loginsMap.isEmpty() &&
      this.loginsMap.containsKey(getProviderName())) {

      /**
       * This is where you would call your backend
       **/

      // now set the returned identity id and token in the provider
      update(identityId, token);
      return token;

   } else {
      // Call getIdentityId method and return null
      this.getIdentityId();
      return null;
   }
}
```

마찬가지로 `getIdentityId` 메서드에도 로그인 맵의 내용에 따라 두 개의 흐름이 있습니다.

```
public String getIdentityId() {

   // Load the identityId from the cache
   identityId = cachedIdentityId;

   if (identityId == null) {
```

```
      // If the logins map is not empty make a call to your backend
      // to get the token and identityId

      if (getProviderName() != null && !this.loginsMap.isEmpty()
         && this.loginsMap.containsKey(getProviderName())) {

         /**
           * This is where you would call your backend
          **/

         // now set the returned identity id and token in the provider
         update(identityId, token);
         return token;

      } else {
         // Otherwise call &COG; using getIdentityId of super class
         return super.getIdentityId();
      }

   } else {
      return identityId;
   }

}
```

iOS - Objective-C

애플리케이션에는 개발자 인증 자격 증명과 함께 퍼블릭 공급자(Login with Amazon, Sign in with Apple, Facebook 또는 Google)를 사용한 미인증 자격 증명 또는 인증 자격 증명 지원이 필요할 수 있습니다. 이렇게 하려면 현재 ID 공급자를 기반으로 올바른 로그인 맵을 반환할 수 있도록 AWSCognitoCredentialsProviderHelper`logins`메서드를 재정의하십시오. 이 예는 미인증, Facebook, 개발자 인증 간 피벗할 수 있는 방법을 보여 줍니다.

```
- (AWSTask<NSDictionary<NSString *, NSString *> *> *)logins {
    if(/*logic to determine if user is unauthenticated*/) {
        return [AWSTask taskWithResult:nil];
    }else if (/*logic to determine if user is Facebook*/){
        return [AWSTask taskWithResult: @{ AWSIdentityProviderFacebook :
 [FBSDKAccessToken currentAccessToken] }];
    }else {
        return [super logins];
    }
```

```
}
```

미인증 상태에서 인증 상태로 전환하면 `[credentialsProvider clearCredentials];`를 호출하여 SDK에서 새 인증 자격 증명을 얻도록 강제해야 합니다. 두 인증 공급자 간에 전환하고 두 공급자를 연결하려고 하지 않는 경우(예를 들어 로그인 딕셔너리에서 여러 공급자에 대한 토큰을 제공하지 않는 경우) `[credentialsProvider clearKeychain];`을 호출해야 합니다. 이렇게 하면 자격 증명이 모두 지워지고 SDK가 새 자격 증명을 얻도록 강제하게 됩니다.

iOS - Swift

애플리케이션에는 개발자 인증 자격 증명과 함께 퍼블릭 공급자(Login with Amazon, Sign in with Apple, Facebook 또는 Google)를 사용한 미인증 자격 증명 또는 인증 자격 증명 지원이 필요할 수 있습니다. 이렇게 하려면 현재 ID 제공자를 기반으로 올바른 로그인 맵을 반환할 수 있도록 AWSCognitoCredentialsProviderHelper`logins`메서드를 재정의하십시오. 이 예는 미인증, Facebook, 개발자 인증 간 피벗할 수 있는 방법을 보여 줍니다.

```
override func logins () -> AWSTask<NSDictionary> {
    if(/*logic to determine if user is unauthenticated*/) {
        return AWSTask(result:nil)
    }else if (/*logic to determine if user is Facebook*/){
        if let token = AccessToken.current?.authenticationToken {
            return AWSTask(result: [AWSIdentityProviderFacebook:token])
        }
        return AWSTask(error:NSError(domain: "Facebook Login", code: -1 , userInfo:
 ["Facebook" : "No current Facebook access token"]))
    }else {
        return super.logins()
    }
}
```

미인증 상태에서 인증 상태로 전환하면 `credentialsProvider.clearCredentials()`를 호출하여 SDK에서 새 인증 자격 증명을 얻도록 강제해야 합니다. 두 인증 공급자 간에 전환하고 두 공급자를 연결하려고 하지 않는 경우(즉, 로그인 딕셔너리에서 여러 공급자에 대한 토큰을 제공하지 않는 경우) `credentialsProvider.clearKeychain()`을 호출해야 합니다. 이렇게 하면 자격 증명이 모두 지워지고 SDK가 새 자격 증명을 얻도록 강제하게 됩니다.

Unity

애플리케이션에는 개발자 인증 자격 증명과 함께 퍼블릭 공급자(Login with Amazon, Sign in with Apple, Facebook 또는 Google)를 사용한 미인증 자격 증명 또는 인증 자격 증명 지원이 필요할 수 있습

니다. 개발자 인증 자격 증명과 다른 자격 증명(퍼블릭 공급자를 사용하는 미인증 자격 증명 및 인증 자격 증명) 간의 본질적인 차이점은 identityId 및 토큰을 얻는 방식입니다. 다른 자격 증명의 경우 모바일 애플리케이션은 인증 시스템과 접촉하는 대신 Amazon Cognito와 직접 상호 작용합니다. 모바일 애플리케이션은 앱 사용자가 선택한 사항에 따라 두 개의 고유 흐름을 지원할 수 있어야 합니다. 이를 위해 사용자 지정 자격 증명 공급자를 일부 변경해야 합니다.

Unity에서 이 작업을 수행하는 권장 방법은 AmazonCognitoEnhancedIdentityProvide 대신 에서 ID 공급자를 확장하고 AbstractCognitoIdentityProvider, 사용자가 자체 백엔드로 인증되지 않은 경우 자체 RefreshAsync 메서드 대신 부모 메서드를 호출하는 것입니다. 사용자가 인증 상태이면 이전에 설명한 것과 동일한 흐름을 사용할 수 있습니다.

Xamarin

애플리케이션에는 개발자 인증 자격 증명과 함께 퍼블릭 공급자(Login with Amazon, Sign in with Apple, Facebook 또는 Google)를 사용한 미인증 자격 증명 또는 인증 자격 증명 지원이 필요할 수 있습니다. 개발자 인증 자격 증명과 다른 자격 증명(퍼블릭 공급자를 사용하는 미인증 자격 증명 및 인증 자격 증명) 간의 본질적인 차이점은 identityId 및 토큰을 얻는 방식입니다. 다른 자격 증명의 경우 모바일 애플리케이션은 인증 시스템과 접촉하는 대신 Amazon Cognito와 직접 상호 작용합니다. 모바일 애플리케이션은 앱 사용자가 선택한 사항에 따라 두 개의 고유 흐름을 지원할 수 있어야 합니다. 이를 위해 사용자 지정 ID 제공업체를 일부 변경해야 합니다.

인증되지 않은 사용자를 인증된 사용자로 전환(자격 증명 풀)

Amazon Cognito 자격 증명 풀은 인증된 사용자와 인증되지 않은 사용자를 모두 지원합니다. 인증되지 않은 사용자는 자격 증명 공급자(IdP)로 로그인하지 않았더라도 AWS 리소스에 대한 액세스 권한을 받습니다. 이 액세스 권한 등급은 사용자가 로그인하기 전에 사용자에게 콘텐츠를 표시하는 데 유용합니다. 인증되지 않은 각 사용자는 개별적으로 로그인되지 않았으며 인증되지 않은 경우에도 자격 증명 풀에 고유한 자격 증명이 있습니다.

이 섹션에서는 사용자가 인증되지 않은 자격 증명으로 로그인하는 것에서 인증된 자격 증명을 사용하는 것으로 전환하도록 선택하는 경우에 대해 설명합니다.

Android

사용자는 인증되지 않은 게스트로 애플리케이션에 로그인할 수 있습니다. 결국 지원되는 IdP 중 하나를 사용하여 로그인하기로 결정할 수 있습니다. Amazon Cognito는 이전 자격 증명이 새 자격 증명과 동일한 고유 식별자를 유지하고, 프로파일 데이터가 자동으로 병합되도록 합니다.

IdentityChangedListener 인터페이스의 프로파일 병합을 통해 애플리케이션에 알립니다. 인터페이스에 identityChanged 메서드를 구현하여 이러한 메시지를 수신할 수 있습니다.

```
@override
public void identityChanged(String oldIdentityId, String newIdentityId) {
    // handle the change
}
```

iOS - Objective-C

사용자는 인증되지 않은 게스트로 애플리케이션에 로그인할 수 있습니다. 결국 지원되는 IdP 중 하나를 사용하여 로그인하기로 결정할 수 있습니다. Amazon Cognito는 이전 자격 증명이 새 자격 증명과 동일한 고유 식별자를 유지하고, 프로파일 데이터가 자동으로 병합되도록 합니다.

NSNotificationCenter는 애플리케이션에 프로파일 병합을 알립니다.

```
[[NSNotificationCenter defaultCenter] addObserver:self
                                      selector:@selector(identityIdDidChange:)
                                      name:AWSCognitoIdentityIdChangedNotification
                                      object:nil];

-(void)identityDidChange:(NSNotification*)notification {
    NSDictionary *userInfo = notification.userInfo;
    NSLog(@"identity changed from %@ to %@",
        [userInfo objectForKey:AWSCognitoNotificationPreviousId],
        [userInfo objectForKey:AWSCognitoNotificationNewId]);
}
```

iOS - Swift

사용자는 인증되지 않은 게스트로 애플리케이션에 로그인할 수 있습니다. 결국 지원되는 IdP 중 하나를 사용하여 로그인하기로 결정할 수 있습니다. Amazon Cognito는 이전 자격 증명이 새 자격 증명과 동일한 고유 식별자를 유지하고, 프로파일 데이터가 자동으로 병합되도록 합니다.

NSNotificationCenter는 애플리케이션에 프로파일 병합을 알립니다.

```
[NSNotificationCenter.defaultCenter().addObserver(observer: self
   selector:"identityDidChange"
   name:AWSCognitoIdentityIdChangedNotification
   object:nil)
```

```
func identityDidChange(notification: NSNotification!) {
  if let userInfo = notification.userInfo as? [String: AnyObject] {
    print("identity changed from: \(userInfo[AWSCognitoNotificationPreviousId])
    to: \(userInfo[AWSCognitoNotificationNewId])")
  }
}
```

JavaScript

처음에 인증되지 않은 사용자

사용자는 일반적으로 인증되지 않은 역할로 시작합니다. 이 역할의 경우 로그인 속성 없이 구성 객체의 인증 자격 증명 속성을 설정합니다. 이 경우, 기본 구성은 다음과 같을 수 있습니다.

```
// set the default config object
var creds = new AWS.CognitoIdentityCredentials({
    IdentityPoolId: 'us-east-1:1699ebc0-7900-4099-b910-2df94f52a030'
});
AWS.config.credentials = creds;
```

인증된 사용자로 전환

인증되지 않은 사용자가 IdP에 로그인한 상태에서 현재 사용자가 토큰을 갖고 있다면, 인증 자격 증명 객체를 업데이트하고 Logins 토큰을 추가하는 사용자 지정 함수를 호출하여 인증되지 않은 사용자를 인증된 사용자로 전환할 수 있습니다.

```
// Called when an identity provider has a token for a logged in user
function userLoggedIn(providerName, token) {
    creds.params.Logins = creds.params.Logins || {};
    creds.params.Logins[providerName] = token;

    // Expire credentials to refresh them on the next request
    creds.expired = true;
}
```

또한 `CognitoIdentityCredentials` 객체를 생성할 수 있습니다. 이 경우 업데이트된 인증 자격 증명 구성 정보를 반영하도록 기존 서비스 객체의 인증 자격 증명 속성을 재설정해야 합니다. 전역 구성 객체 사용을 참조하세요.

`CognitoIdentityCredentials` 객체에 대한 자세한 내용은 AWS SDK for JavaScript API 참조에서 AWS.CognitoIdentityCredentials를 참조하세요.

Unity

사용자는 인증되지 않은 게스트로 애플리케이션에 로그인할 수 있습니다. 결국 지원되는 IdP 중 하나를 사용하여 로그인하기로 결정할 수 있습니다. Amazon Cognito는 이전 자격 증명이 새 자격 증명과 동일한 고유 식별자를 유지하고, 프로파일 데이터가 자동으로 병합되도록 합니다.

`IdentityChangedEvent`를 구독하여 프로파일 병합에 대한 알림을 받을 수 있습니다.

```
credentialsProvider.IdentityChangedEvent += delegate(object sender,
 CognitoAWSCredentials.IdentityChangedArgs e)
{
    // handle the change
    Debug.log("Identity changed from " + e.OldIdentityId + " to " + e.NewIdentityId);
};
```

Xamarin

사용자는 인증되지 않은 게스트로 애플리케이션에 로그인할 수 있습니다. 결국 지원되는 IdP 중 하나를 사용하여 로그인하기로 결정할 수 있습니다. Amazon Cognito는 이전 자격 증명이 새 자격 증명과 동일한 고유 식별자를 유지하고, 프로파일 데이터가 자동으로 병합되도록 합니다.

```
credentialsProvider.IdentityChangedEvent += delegate(object sender,
 CognitoAWSCredentials.IdentityChangedArgs e){
    // handle the change
    Console.WriteLine("Identity changed from " + e.OldIdentityId + " to " +
 e.NewIdentityId);
};
```

Amazon Cognito Sync

⚠ Amazon Cognito Sync를 처음 사용하는 경우 AWS AppSync를 사용하세요. Amazon Cognito Sync와 마찬가지로, AWS AppSync도 디바이스 사이에서 애플리케이션 데이터를 동기화하는 서비스입니다.
앱 기본 설정이나 게임 상태 같은 사용자 데이터를 동기화할 수 있습니다. 또한 이러한 기능을 더욱 확장해, 복수의 사용자가 공유 데이터를 실시간으로 동기화하고 협업할 수 있게 합니다.

Amazon Cognito Sync는 여러 디바이스 간에 애플리케이션 관련 사용자 데이터를 동기화할 수 있는 AWS 서비스 및 클라이언트 라이브러리입니다. Amazon Cognito Sync를 사용하면 자체 백엔드를 사용하지 않고 모바일 디바이스 및 웹 간에 사용자 프로필 데이터를 동기화할 수 있습니다. 클라이언트 라이브러리는 디바이스 연결 상태와 관계없이 앱에서 데이터를 읽고 쓸 수 있도록 로컬로 데이터를 캐싱합니다. 디바이스가 온라인 상태일 때 데이터를 동기화할 수 있습니다. 푸시 동기화를 설정한 경우 업데이트가 있음을 다른 디바이스에 즉시 알릴 수 있습니다.

Amazon Cognito 자격 증명 리전 가용성에 대한 자세한 내용은 AWS 서비스 리전 가용성을 참조하세요.

Amazon Cognito Sync에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- Amazon Cognito Sync 시작하기
- 데이터 동기화
- 콜백 처리
- 푸시 동기화
- Amazon Cognito 스트림
- Amazon Cognito 이벤트

Amazon Cognito Sync 시작하기

⚠ Amazon Cognito Sync를 처음 사용하는 경우 AWS AppSync를 사용하세요. Amazon Cognito Sync와 마찬가지로, AWS AppSync도 디바이스 사이에서 애플리케이션 데이터를 동기화하는 서비스입니다.

앱 기본 설정이나 게임 상태 같은 사용자 데이터를 동기화할 수 있습니다. 또한 이러한 기능을 더욱 확장해, 복수의 사용자가 공유 데이터를 실시간으로 동기화하고 협업할 수 있게 합니다.

Amazon Cognito Sync는 애플리케이션 관련 사용자 데이터의 교차 디바이스 동기화를 지원하는 클라이언트 라이브러리 및 AWS 서비스입니다. 를 사용하여 모바일 디바이스와 웹 애플리케이션에서 사용자 프로파일 데이터를 동기화할 수 있습니다. 클라이언트 라이브러리는 디바이스 연결 상태와 관계없이 앱에서 데이터를 읽고 쓸 수 있도록 로컬로 데이터를 캐싱합니다. 디바이스가 온라인 상태인 경우 데이터를 동기화할 수 있으며, 푸시 동기화를 설정한 경우 업데이트가 가능함을 다른 디바이스에 즉시 알립니다.

Amazon Cognito에서 자격 증명 풀 설정

Amazon Cognito Sync에는 사용자 자격 증명을 제공하기 위해 Amazon Cognito 자격 증명 풀이 필요합니다. 따라서 먼저 자격 증명 풀을 설정해야 Amazon Cognito Sync를 사용할 수 있습니다. 자격 증명 풀을 생성하고 SDK를 설치하려면 Amazon Cognito 자격 증명 풀 시작하기를 참조하세요.

데이터 저장 및 동기화

자격 증명 풀을 설정하고 SDK를 설치하면 디바이스 간에 데이터를 저장하고 동기화하는 작업을 시작할 수 있습니다. 자세한 내용은 데이터 동기화 섹션을 참조하세요.

데이터 동기화

⚠ Amazon Cognito Sync를 처음 사용하는 경우 AWS AppSync를 사용하세요. Amazon Cognito Sync와 마찬가지로, AWS AppSync도 디바이스 사이에서 애플리케이션 데이터를 동기화하는 서비스입니다.
앱 기본 설정이나 게임 상태 같은 사용자 데이터를 동기화할 수 있습니다. 또한 이러한 기능을 더욱 확장해, 복수의 사용자가 공유 데이터를 실시간으로 동기화하고 협업할 수 있게 합니다.

Amazon Cognito를 사용하면 키-값 쌍이 포함된 데이터 세트에 사용자 데이터를 저장할 수 있습니다. Amazon Cognito는 이 데이터를 자격 증명 풀의 자격 증명과 연결하여 앱이 로그인 및 디바이스 전반에서 액세스할 수 있도록 합니다. Amazon Cognito 서비스와 최종 사용자의 디바이스 간에 이 데이터를 동기화하려면 동기화 메서드를 호출합니다. 각 데이터 세트의 최대 크기는 1MB일 수 있습니다. 자격 증명에 데이터 세트를 최대 20개까지 연결할 수 있습니다.

Amazon Cognito Sync 클라이언트는 자격 증명 데이터에 대한 로컬 캐시를 생성합니다. 앱에서 키를 읽고 쓸 때 이 로컬 캐시와 통신합니다. 이렇게 통신하면 디바이스에서 수행한 모든 변경 사항을 오프라인 상태에서도 디바이스에서 즉시 사용할 수 있습니다. 동기화 메서드가 호출되면 서비스에서 변경 사항을 디바이스로 가져오고 모든 로컬 변경 사항이 서비스로 푸시됩니다. 이때 변경 사항은 다른 디바이스에서 동기화하는 데 사용 가능합니다.

Amazon Cognito Sync 클라이언트 초기화

Amazon Cognito Sync 클라이언트를 초기화하려면 먼저 자격 증명 공급자를 생성해야 합니다. 자격 증명 공급자는 앱에서 AWS 리소스에 액세스할 수 있는 임시 AWS 자격 증명을 취득합니다. 필요한 헤더 파일도 가져와야 합니다. 다음 단계를 사용하여 Amazon Cognito Sync 클라이언트를 초기화합니다.

Android

1. 자격 증명 얻기의 지침에 따라 자격 증명 공급자를 생성합니다.
2. 다음과 같이 Amazon Cognito 패키지를 가져옵니다. `import com.amazonaws.mobileconnectors.cognito.*;`
3. Amazon Cognito Sync를 초기화합니다. 다음과 같이 Android 앱 컨텍스트, 자격 증명 풀 ID, AWS 리전 및 초기화된 Amazon Cognito 자격 증명 공급자를 전달합니다.

```
CognitoSyncManager client = new CognitoSyncManager(
    getApplicationContext(),
    Regions.YOUR_REGION,
    credentialsProvider);
```

iOS - Objective-C

1. 자격 증명 얻기의 지침에 따라 자격 증명 공급자를 생성합니다.
2. 다음과 같이 `AWSCore` 및 `Cognito`를 가져오고 `AWSCognito`를 초기화합니다.

```
#import <AWSiOSSDKv2/AWSCore.h>
#import <AWSCognitoSync/Cognito.h>

AWSCognito *syncClient = [AWSCognito defaultCognito];
```

3. CocoaPods를 사용하는 경우 `<AWSiOSSDKv2/AWSCore.h>`를 `AWSCore.h`로 바꿉니다. Amazon Cognito 가져오기에 대해 동일한 구문을 따릅니다.

iOS - Swift

1. 자격 증명 얻기의 지침에 따라 자격 증명 공급자를 생성합니다.
2. 다음과 같이 AWSCognito를 가져오고 초기화합니다.

```
import AWSCognito
let syncClient = AWSCognito.default()!
```

JavaScript

1. Amazon Cognito Sync Manager for JavaScript를 다운로드합니다.
2. 프로젝트에 Sync Manager 라이브러리를 포함합니다.
3. 자격 증명 얻기의 지침에 따라 자격 증명 공급자를 생성합니다.
4. 다음과 같이 Sync 관리자를 초기화합니다.

```
var syncManager = new AWS.CognitoSyncManager();
```

Unity

1. 자격 증명 얻기의 지침에 따라 CognitoAWSCredentials의 인스턴스를 생성합니다.
2. CognitoSyncManager의 인스턴스를 만듭니다. 다음과 같이 CognitoAwsCredentials 객체 및 AmazonCognitoSyncConfig를 전달하고 최소한 리전 세트를 포함합니다.

```
AmazonCognitoSyncConfig clientConfig = new AmazonCognitoSyncConfig { RegionEndpoint =
 REGION };
CognitoSyncManager syncManager = new CognitoSyncManager(credentials, clientConfig);
```

Xamarin

1. 자격 증명 얻기의 지침에 따라 CognitoAWSCredentials의 인스턴스를 생성합니다.
2. CognitoSyncManager의 인스턴스를 만듭니다. 다음과 같이 CognitoAwsCredentials 객체 및 AmazonCognitoSyncConfig를 전달하고 최소한 리전 세트를 포함합니다.

```
AmazonCognitoSyncConfig clientConfig = new AmazonCognitoSyncConfig { RegionEndpoint =
 REGION };
```

```
CognitoSyncManager syncManager = new CognitoSyncManager(credentials, clientConfig);
```

데이터 집합 이해

Amazon Cognito는 사용자 프로필 데이터를 데이터 세트로 구성합니다. 각 데이터 세트에는 최대 1MB의 데이터가 키 값 쌍의 양식으로 포함될 수 있습니다. 데이터 세트는 동기화할 수 있는 가장 세분화된 엔터티입니다. 데이터 세트에서 수행된 읽기 및 쓰기 작업은 동기화 메서드가 호출될 때까지 로컬 스토어에만 영향을 줍니다. Amazon Cognito는 고유한 문자열로 데이터 세트를 식별합니다. 다음과 같이 새 데이터 세트를 생성하거나 기존 데이터 세트를 열 수 있습니다.

Android

```
Dataset dataset = client.openOrCreateDataset("datasetname");
```

데이터 세트를 삭제하려면 다음과 같이 먼저 메서드를 호출하여 로컬 스토리지에서 데이터 세트를 제거한 다음 `synchronize` 메서드를 호출하여 Amazon Cognito에서 데이터 세트를 삭제합니다.

```
dataset.delete();
dataset.synchronize(syncCallback);
```

iOS - Objective-C

```
AWSCognitoDataset *dataset = [syncClient openOrCreateDataset:@"myDataSet"];
```

데이터 세트를 삭제하려면 다음과 같이 먼저 메서드를 호출하여 로컬 스토리지에서 데이터 세트를 제거한 다음 `synchronize` 메서드를 호출하여 Amazon Cognito에서 데이터 세트를 삭제합니다.

```
[dataset clear];
[dataset synchronize];
```

iOS - Swift

```
let dataset = syncClient.openOrCreateDataset("myDataSet")!
```

데이터 세트를 삭제하려면 다음과 같이 먼저 메서드를 호출하여 로컬 스토리지에서 데이터 세트를 제거한 다음 `synchronize` 메서드를 호출하여 Amazon Cognito에서 데이터 세트를 삭제합니다.

```
dataset.clear()
dataset.synchronize()
```

JavaScript

```
syncManager.openOrCreateDataset('myDatasetName', function(err, dataset) {
   // ...
});
```

Unity

```
string myValue = dataset.Get("myKey");
dataset.Put("myKey", "newValue");
```

데이터 세트에서 키를 삭제하려면 다음과 같이 Remove를 사용합니다.

```
dataset.Remove("myKey");
```

Xamarin

```
Dataset dataset = syncManager.OpenOrCreateDataset("myDatasetName");
```

데이터 세트를 삭제하려면 다음과 같이 먼저 메서드를 호출하여 로컬 스토리지에서 데이터 세트를 제거한 다음 synchronize 메서드를 호출하여 Amazon Cognito에서 데이터 세트를 삭제합니다.

```
dataset.Delete();
dataset.SynchronizeAsync();
```

데이터 집합의 데이터 읽기 및 쓰기

Amazon Cognito 데이터 세트는 키를 통해 값에 액세스할 수 있는 사전으로 작동합니다. 다음 예와 같이 데이터 세트가 사전인 것처럼 데이터 세트의 키와 값을 읽거나 추가하거나 수정할 수 있습니다.

데이터 세트에 작성된 값은 동기화 메서드를 호출할 때까지 로컬로 캐시된 데이터의 사본에만 영향을 줍니다.

Android

```
String value = dataset.get("myKey");
```

```
dataset.put("myKey", "my value");
```

iOS - Objective-C

```
[dataset setString:@"my value" forKey:@"myKey"];
NSString *value = [dataset stringForKey:@"myKey"];
```

iOS - Swift

```
dataset.setString("my value", forKey:"myKey")
let value = dataset.stringForKey("myKey")
```

JavaScript

```
dataset.get('myKey', function(err, value) {
  console.log('myRecord: ' + value);
});

dataset.put('newKey', 'newValue', function(err, record) {
  console.log(record);
});

dataset.remove('oldKey', function(err, record) {
  console.log(success);
});
```

Unity

```
string myValue = dataset.Get("myKey");
dataset.Put("myKey", "newValue");
```

Xamarin

```
//obtain a value
string myValue = dataset.Get("myKey");

// Create a record in a dataset and synchronize with the server
dataset.OnSyncSuccess += SyncSuccessCallback;
```

```
dataset.Put("myKey", "myValue");
dataset.SynchronizeAsync();

void SyncSuccessCallback(object sender, SyncSuccessEventArgs e) {
  // Your handler code here
}
```

Android

데이터 세트에서 키를 제거하려면 다음과 같이 `remove` 메서드를 사용합니다.

```
dataset.remove("myKey");
```

iOS - Objective-C

데이터 세트에서 키를 삭제하려면 다음과 같이 `removeObjectForKey`를 사용합니다.

```
[dataset removeObjectForKey:@"myKey"];
```

iOS - Swift

데이터 세트에서 키를 삭제하려면 다음과 같이 `removeObjectForKey`를 사용합니다.

```
dataset.removeObjectForKey("myKey")
```

Unity

데이터 세트에서 키를 삭제하려면 다음과 같이 `Remove`를 사용합니다.

```
dataset.Remove("myKey");
```

Xamarin

`Remove`를 사용하여 데이터 세트에서 키를 삭제할 수 있습니다.

```
dataset.Remove("myKey");
```

로컬 데이터를 동기화 스토어와 동기화

Android

synchronize 메서드는 로컬로 캐시된 데이터를 Amazon Cognito Sync 스토어에 저장된 데이터와 비교합니다. Amazon Cognito Sync 스토어에서 원격 변경 사항을 가져오고, 충돌이 발생할 경우 충돌 해결책이 호출되며, 디바이스의 업데이트된 값이 서비스로 푸시됩니다. 데이터 세트를 동기화하려면 해당 synchronize 메서드를 호출합니다.

```
dataset.synchronize(syncCallback);
```

아래 설명과 같이 synchronize 메서드는 SyncCallback 인터페이스의 구현을 받습니다.

synchronizeOnConnectivity() 메서드는 연결을 사용할 수 있는 경우 동기화하려고 시도합니다. 연결이 즉시 사용 가능하게 되면 synchronizeOnConnectivity()는 synchronize()처럼 동작합니다. 그렇지 않으면 연결 변경 사항을 모니터링하고 연결이 사용 가능하게 되면 동기화를 수행합니다. synchronizeOnConnectivity()가 여러 번 호출되면 마지막 동기화 요청만 유지되며 마지막 콜백만 실행됩니다. 데이터 세트 또는 콜백이 가비지 수집인 경우 이 메서드는 동기화를 수행하지 않으며 콜백이 실행되지 않습니다.

데이터 세트 동기화 및 다른 콜백에 대한 자세한 내용은 콜백 처리 섹션을 참조하세요.

iOS - Objective-C

synchronize 메서드는 로컬로 캐시된 데이터를 Amazon Cognito Sync 스토어에 저장된 데이터와 비교합니다. Amazon Cognito Sync 스토어에서 원격 변경 사항을 가져오고, 충돌이 발생할 경우 충돌 해결책이 호출되며, 디바이스의 업데이트된 값이 서비스로 푸시됩니다. 데이터 세트를 동기화하려면 해당 synchronize 메서드를 호출합니다.

synchronize 메서드는 비동기식이며 AWSTask 객체를 반환하여 응답을 처리합니다.

```
[[dataset synchronize] continueWithBlock:^id(AWSTask *task) {
    if (task.isCancelled) {
        // Task cancelled.
    } else if (task.error) {
        // Error while executing task.
    } else {
        // Task succeeded. The data was saved in the sync store.
    }
    return nil;
}];
```

synchronizeOnConnectivity 메서드는 디바이스가 연결된 경우 동기화하려고 시도합니다. 먼저 synchronizeOnConnectivity는 연결을 확인하며, 디바이스가 온라인 상태인 경우 즉시 synchronize를 호출하고 시도와 연관된 AWSTask 객체를 반환합니다.

디바이스가 오프라인인 경우 synchronizeOnConnectivity는 1) 다음에 디바이스가 온라인 상태가 될 때 동기화를 예약하며 2) nil 결과와 함께 AWSTask를 반환합니다. 예약된 동기화는 데이터 세트 객체의 수명 주기 동안 유효합니다. 다시 연결되기 전에 앱이 종료된 경우 데이터가 동기화되지 않습니다. 예약된 동기화 중 이벤트가 발생할 때 알림을 받으려면 AWSCognito에 있는 알림의 관찰자를 추가해야 합니다.

데이터 세트 동기화 및 다른 콜백에 대한 자세한 내용은 콜백 처리 섹션을 참조하세요.

iOS - Swift

synchronize 메서드는 로컬로 캐시된 데이터를 Amazon Cognito Sync 스토어에 저장된 데이터와 비교합니다. Amazon Cognito Sync 스토어에서 원격 변경 사항을 가져오고, 충돌이 발생할 경우 충돌 해결책이 호출되며, 디바이스의 업데이트된 값이 서비스로 푸시됩니다. 데이터 세트를 동기화하려면 해당 synchronize 메서드를 호출합니다.

synchronize 메서드는 비동기식이며 AWSTask 객체를 반환하여 응답을 처리합니다.

```
dataset.synchronize().continueWith(block: { (task) -> AnyObject? in

            if task.isCancelled {
                // Task cancelled.
            } else if task.error != nil {
                // Error while executing task
            } else {
                // Task succeeded. The data was saved in the sync store.
            }
            return task
})
```

synchronizeOnConnectivity 메서드는 디바이스가 연결된 경우 동기화하려고 시도합니다. 먼저 synchronizeOnConnectivity는 연결을 확인하며, 디바이스가 온라인 상태인 경우 즉시 synchronize를 호출하고 시도와 연관된 AWSTask 객체를 반환합니다.

디바이스가 오프라인인 경우 synchronizeOnConnectivity는 1) 다음에 디바이스가 온라인 상태가 될 때 동기화를 예약하며 2) nil 결과와 함께 AWSTask 객체를 반환합니다. 예약된 동기화는 데이터 세트 객체의 수명 주기 동안 유효합니다. 다시 연결되기 전에 앱이 종료된 경우 데이터가 동기화되지

않습니다. 예약된 동기화 중 이벤트가 발생할 때 알림을 받으려면 AWSCognito에 있는 알림의 관찰자를 추가해야 합니다.

데이터 세트 동기화 및 다른 콜백에 대한 자세한 내용은 콜백 처리 섹션을 참조하세요.

JavaScript

synchronize 메서드는 로컬로 캐시된 데이터를 Amazon Cognito Sync 스토어에 저장된 데이터와 비교합니다. Amazon Cognito Sync 스토어에서 원격 변경 사항을 가져오고, 충돌이 발생할 경우 충돌 해결책이 호출되며, 디바이스의 업데이트된 값이 서비스로 푸시됩니다. 데이터 세트를 동기화하려면 해당 synchronize 메서드를 호출합니다.

```
dataset.synchronize();
```

데이터 세트 동기화 및 다른 콜백에 대한 자세한 내용은 콜백 처리 섹션을 참조하세요.

Unity

synchronize 메서드는 로컬로 캐시된 데이터를 Amazon Cognito Sync 스토어에 저장된 데이터와 비교합니다. Amazon Cognito Sync 스토어에서 원격 변경 사항을 가져오고, 충돌이 발생할 경우 충돌 해결책이 호출되며, 디바이스의 업데이트된 값이 서비스로 푸시됩니다. 데이터 세트를 동기화하려면 해당 synchronize 메서드를 호출합니다.

```
dataset.Synchronize();
```

동기화는 비동기식으로 실행되며 데이터 세트에서 지정할 수 있는 여러 콜백 중 하나를 호출합니다.

데이터 세트 동기화 및 다른 콜백에 대한 자세한 내용은 콜백 처리 섹션을 참조하세요.

Xamarin

synchronize 메서드는 로컬로 캐시된 데이터를 Amazon Cognito Sync 스토어에 저장된 데이터와 비교합니다. Amazon Cognito Sync 스토어에서 원격 변경 사항을 가져오고, 충돌이 발생할 경우 충돌 해결책이 호출되며, 디바이스의 업데이트된 값이 서비스로 푸시됩니다. 데이터 세트를 동기화하려면 해당 synchronize 메서드를 호출합니다.

```
dataset.SynchronizeAsync();
```

데이터 세트 동기화 및 다른 콜백에 대한 자세한 내용은 콜백 처리 섹션을 참조하세요.

콜백 처리

⚠ Amazon Cognito Sync를 처음 사용하는 경우 AWS AppSync를 사용하세요. Amazon Cognito Sync와 마찬가지로, AWS AppSync도 디바이스 사이에서 애플리케이션 데이터를 동기화하는 서비스입니다.
앱 기본 설정이나 게임 상태 같은 사용자 데이터를 동기화할 수 있습니다. 또한 이러한 기능을 더욱 확장해, 복수의 사용자가 공유 데이터를 실시간으로 동기화하고 협업할 수 있게 합니다.

이 섹션에서는 콜백 처리 방법을 설명합니다.

Android

SyncCallback 인터페이스

`SyncCallback` 인터페이스를 구현하여 앱에서 데이터 세트 동기화에 대한 알림을 받을 수 있습니다. 그러면 앱에서는 로컬 데이터 삭제, 미인증 및 인증 프로파일 병합 및 동기화 충돌 해결에 대해 결정할 수 있습니다. 인터페이스에 필요한 다음 메서드를 구현해야 합니다.

- `onSuccess()`
- `onFailure()`
- `onConflict()`
- `onDatasetDeleted()`
- `onDatasetsMerged()`

모든 콜백을 지정하지 않으려면 `DefaultSyncCallback` 클래스를 사용할 수도 있습니다. 이 클래스는 모든 콜백에 대해 기본적으로 빈 구현을 제공합니다.

onSuccess

`onSuccess()` 콜백은 동기화 스토어에서 데이터 세트를 성공적으로 다운로드한 경우 트리거됩니다.

```
@Override
public void onSuccess(Dataset dataset, List<Record> newRecords) {
}
```

onFailure

동기화 중 예외가 발생할 경우 onFailure()가 호출됩니다.

```
@Override
public void onFailure(DataStorageException dse) {
}
```

onConflict

로컬 스토어와 동기화 스토어에서 동일한 키가 수정된 경우 충돌이 발생할 수 있습니다. onConflict() 메서드는 충돌 해결을 처리합니다. 이 메서드를 구현하지 않으면 Amazon Cognito Sync 클라이언트는 기본적으로 가장 최근 변경 사항을 사용합니다.

```
@Override
public boolean onConflict(Dataset dataset, final List<SyncConflict> conflicts) {
    List<Record> resolvedRecords = new ArrayList<Record>();
    for (SyncConflict conflict : conflicts) {
        /* resolved by taking remote records */
        resolvedRecords.add(conflict.resolveWithRemoteRecord());

        /* alternately take the local records */
        // resolvedRecords.add(conflict.resolveWithLocalRecord());

        /* or customer logic, say concatenate strings */
        // String newValue = conflict.getRemoteRecord().getValue()
        //     + conflict.getLocalRecord().getValue();
        // resolvedRecords.add(conflict.resolveWithValue(newValue);
    }
    dataset.resolve(resolvedRecords);

    // return true so that synchronize() is retried after conflicts are resolved
    return true;
}
```

onDatasetDeleted

데이터 세트가 삭제되면 Amazon Cognito 클라이언트는 SyncCallback 인터페이스를 사용하여 로컬로 캐싱된 데이터 세트의 사본도 삭제해야 하는지 여부를 확인합니다. onDatasetDeleted() 메서드를 구현하여 로컬 데이터를 사용하여 수행할 사항을 클라이언트 SDK에게 알립니다.

```
@Override
public boolean onDatasetDeleted(Dataset dataset, String datasetName) {
    // return true to delete the local copy of the dataset
```

```
    return true;
}
```

onDatasetMerged

이전에 연결되지 않은 두 자격 증명이 서로 연결되면 해당 데이터 세트가 모두 병합됩니다. 애플리케이션은 `onDatasetsMerged()` 메서드를 통해 병합에 대한 알림을 받습니다.

```
@Override
public boolean onDatasetsMerged(Dataset dataset, List<String> datasetNames) {
    // return false to handle Dataset merge outside the synchronization callback
    return false;
}
```

iOS - Objective-C

동기화 알림

Amazon Cognito 클라이언트는 동기화를 호출하는 중 여러 `NSNotification` 이벤트를 출력합니다. 표준 `NSNotificationCenter`를 통해 이러한 알림을 모니터링하도록 등록할 수 있습니다.

```
[NSNotificationCenter defaultCenter]
  addObserver:self
  selector:@selector(myNotificationHandler:)
  name:NOTIFICATION_TYPE
  object:nil];
```

Amazon Cognito는 아래에 나열된 대로 5가지 알림 유형을 지원합니다.

AWSCognitoDidStartSynchronizeNotification

동기화 작업이 시작될 때 호출됩니다. `userInfo`에는 동기화되는 데이터 세트의 이름인 핵심 데이터 세트가 포함되어 있습니다.

AWSCognitoDidEndSynchronizeNotification

동기화 작업이 완료(성공 등)될 때 호출됩니다. `userInfo`에는 동기화되는 데이터 세트의 이름인 핵심 데이터 세트가 포함되어 있습니다.

AWSCognitoDidFailToSynchronizeNotification

동기화 작업이 실패할 때 호출됩니다. `userInfo`에는 동기화되는 데이터 세트의 이름인 핵심 데이터 세트와 실패의 원인인 오류가 포함된 핵심 오류가 포함되어 있습니다.

AWSCognitoDidChangeRemoteValueNotification

로컬 변경 사항이 Amazon Cognito에 성공적으로 푸시된 경우 호출됩니다. `userInfo`에는 동기화되는 데이터 세트의 이름인 핵심 데이터 세트와 푸시된 레코드 키의 NSArray가 포함된 핵심 키가 포함되어 있습니다.

AWSCognitoDidChangeLocalValueFromRemoteNotification

동기화 작업으로 인해 로컬 값이 변경될 때 호출됩니다. `userInfo`에는 동기화되는 데이터 세트의 이름인 핵심 데이터 세트와 변경된 레코드 키의 NSArray가 포함된 핵심 키가 포함되어 있습니다.

충돌 해결 핸들러

동기화 작업 중 로컬 스토어와 동기화 스토어에서 동일한 키가 수정된 경우 충돌이 발생할 수 있습니다. 충돌 해결 핸들러를 설정하지 않은 경우 Amazon Cognito에서는 기본적으로 가장 최근 업데이트를 선택합니다.

AWSCognitoRecordConflictHandler를 구현하고 할당하면 기본 충돌 해결을 변경할 수 있습니다. AWSCognitoConflict 입력 파라미터 충돌에는 로컬로 캐시된 데이터와 동기화 스토어의 충돌 레코드 둘 모두에 대한 AWSCognitoRecord 객체가 포함됩니다. AWSCognitoConflict를 사용하면 로컬 레코드 [conflict resolveWithLocalRecord]와 원격 레코드 [conflict resolveWithRemoteRecord] 또는 새로운 값인 [conflict resolveWithValue:value]를 사용하여 충돌을 해결할 수 있습니다. 이 메서드에서 nil을 반환하면 동기화가 계속되지 않으며, 다음에 동기화 프로세스를 시작할 때 충돌이 다시 표시됩니다.

클라이언트 수준에서 충돌 해결 핸들러를 설정할 수 있습니다.

```
client.conflictHandler = ^AWSCognitoResolvedConflict* (NSString *datasetName,
 AWSCognitoConflict *conflict) {
    // always choose local changes
    return [conflict resolveWithLocalRecord];
};
```

또는 데이터 세트 수준에서 해당 핸들러를 설정할 수 있습니다.

```
dataset.conflictHandler = ^AWSCognitoResolvedConflict* (NSString *datasetName,
 AWSCognitoConflict *conflict) {
    // override and always choose remote changes
    return [conflict resolveWithRemoteRecord];
};
```

데이터 세트 삭제 핸들러

데이터 세트가 삭제되면 Amazon Cognito 클라이언트는 AWSCognitoDatasetDeletedHandler를 사용하여 로컬로 캐싱된 데이터 세트의 사본도 삭제해야 하는지 여부를 확인합니다. AWSCognitoDatasetDeletedHandler가 구현되지 않으면 로컬 데이터가 자동으로 제거됩니다. 제거하기 전에 로컬 데이터의 사본을 유지하거나 로컬 데이터를 유지하려면 AWSCognitoDatasetDeletedHandler를 구현하세요.

클라이언트 수준에서 데이터 세트 삭제 핸들러를 설정할 수 있습니다.

```
client.datasetDeletedHandler = ^BOOL (NSString *datasetName) {
    // make a backup of the data if you choose
    ...
    // delete the local data (default behavior)
    return YES;
};
```

또는 데이터 세트 수준에서 해당 핸들러를 설정할 수 있습니다.

```
dataset.datasetDeletedHandler = ^BOOL (NSString *datasetName) {
    // override default and keep the local data
    return NO;
};
```

데이터 세트 병합 핸들러

이전에 연결되지 않은 두 자격 증명이 서로 연결되면 해당 데이터 세트가 모두 병합됩니다. 애플리케이션은 DatasetMergeHandler를 통해 병합에 대한 알림을 받습니다. 핸들러는 루트 데이터 세트의 이름과 루트 데이터 세트의 병합으로 표시된 데이터 세트 이름의 어레이를 받습니다.

DatasetMergeHandler가 구현되지 않으면 이러한 데이터 세트가 무시되지만, 자격 증명의 20개 최대 총 데이터 세트에서 공간을 계속 소비합니다.

클라이언트 수준에서 데이터 세트 병합 핸들러를 설정할 수 있습니다.

```
client.datasetMergedHandler = ^(NSString *datasetName, NSArray *datasets) {
    // Blindly delete the datasets
    for (NSString *name in datasets) {
        AWSCognitoDataset *merged = [[AWSCognito defaultCognito]
 openOrCreateDataset:name];
       [merged clear];
       [merged synchronize];
    }
```

```
};
```

또는 데이터 세트 수준에서 해당 핸들러를 설정할 수 있습니다.

```
dataset.datasetMergedHandler = ^(NSString *datasetName, NSArray *datasets) {
    // Blindly delete the datasets
    for (NSString *name in datasets) {
        AWSCognitoDataset *merged = [[AWSCognito defaultCognito]
 openOrCreateDataset:name];
        // do something with the data if it differs from existing dataset
        ...
        // now delete it
        [merged clear];
        [merged synchronize];
    }
};
```

iOS - Swift

동기화 알림

Amazon Cognito 클라이언트는 동기화를 호출하는 중 여러 NSNotification 이벤트를 출력합니다. 표준 NSNotificationCenter를 통해 이러한 알림을 모니터링하도록 등록할 수 있습니다.

```
NSNotificationCenter.defaultCenter().addObserver(observer: self,
   selector: "myNotificationHandler",
   name:NOTIFICATION_TYPE,
   object:nil)
```

Amazon Cognito는 아래에 나열된 대로 5가지 알림 유형을 지원합니다.

AWSCognitoDidStartSynchronizeNotification

동기화 작업이 시작될 때 호출됩니다. userInfo에는 동기화되는 데이터 세트의 이름인 핵심 데이터 세트가 포함되어 있습니다.

AWSCognitoDidEndSynchronizeNotification

동기화 작업이 완료(성공 등)될 때 호출됩니다. userInfo에는 동기화되는 데이터 세트의 이름인 핵심 데이터 세트가 포함되어 있습니다.

AWSCognitoDidFailToSynchronizeNotification

동기화 작업이 실패할 때 호출됩니다. userInfo에는 동기화되는 데이터 세트의 이름인 핵심 데이터 세트와 실패의 원인인 오류가 포함된 핵심 오류가 포함되어 있습니다.

AWSCognitoDidChangeRemoteValueNotification

로컬 변경 사항이 Amazon Cognito에 성공적으로 푸시된 경우 호출됩니다. userInfo에는 동기화되는 데이터 세트의 이름인 핵심 데이터 세트와 푸시된 레코드 키의 NSArray가 포함된 핵심 키가 포함되어 있습니다.

AWSCognitoDidChangeLocalValueFromRemoteNotification

동기화 작업으로 인해 로컬 값이 변경될 때 호출됩니다. userInfo에는 동기화되는 데이터 세트의 이름인 핵심 데이터 세트와 변경된 레코드 키의 NSArray가 포함된 핵심 키가 포함되어 있습니다.

충돌 해결 핸들러

동기화 작업 중 로컬 스토어와 동기화 스토어에서 동일한 키가 수정된 경우 충돌이 발생할 수 있습니다. 충돌 해결 핸들러를 설정하지 않은 경우 Amazon Cognito에서는 기본적으로 가장 최근 업데이트를 선택합니다.

AWSCognitoRecordConflictHandler를 구현하고 할당하면 기본 충돌 해결을 변경할 수 있습니다. AWSCognitoConflict 입력 파라미터 충돌에는 로컬로 캐시된 데이터와 동기화 스토어의 충돌 레코드 둘 모두에 대한 AWSCognitoRecord 객체가 포함됩니다. AWSCognitoConflict를 사용하면 로컬 레코드 [conflict resolveWithLocalRecord]와 원격 레코드 [conflict resolveWithRemoteRecord] 또는 새로운 값인 [conflict resolveWithValue:value]를 사용하여 충돌을 해결할 수 있습니다. 이 메서드에서 nil을 반환하면 동기화가 계속되지 않으며, 다음에 동기화 프로세스를 시작할 때 충돌이 다시 표시됩니다.

클라이언트 수준에서 충돌 해결 핸들러를 설정할 수 있습니다.

```
client.conflictHandler = {
    (datasetName: String?, conflict: AWSCognitoConflict?) ->
 AWSCognitoResolvedConflict? in
    return conflict.resolveWithLocalRecord()
}
```

또는 데이터 세트 수준에서 해당 핸들러를 설정할 수 있습니다.

```
dataset.conflictHandler = {
    (datasetName: String?, conflict: AWSCognitoConflict?) ->
 AWSCognitoResolvedConflict? in
```

```
    return conflict.resolveWithLocalRecord()
}
```

데이터 세트 삭제 핸들러

데이터 세트가 삭제되면 Amazon Cognito 클라이언트는 AWSCognitoDatasetDeletedHandler를 사용하여 로컬로 캐싱된 데이터 세트의 사본도 삭제해야 하는지 여부를 확인합니다. AWSCognitoDatasetDeletedHandler가 구현되지 않으면 로컬 데이터가 자동으로 제거됩니다. 제거하기 전에 로컬 데이터의 사본을 유지하거나 로컬 데이터를 유지하려면 AWSCognitoDatasetDeletedHandler를 구현하세요.

클라이언트 수준에서 데이터 세트 삭제 핸들러를 설정할 수 있습니다.

```
client.datasetDeletedHandler = {
      (datasetName: String!) -> Bool in
      // make a backup of the data if you choose
      ...
      // delete the local data (default behaviour)
      return true
}
```

또는 데이터 세트 수준에서 해당 핸들러를 설정할 수 있습니다.

```
dataset.datasetDeletedHandler = {
      (datasetName: String!) -> Bool in
      // make a backup of the data if you choose
      ...
      // delete the local data (default behaviour)
      return true
}
```

데이터 집합 병합 핸들러

이전에 연결되지 않은 두 자격 증명이 서로 연결되면 해당 데이터 세트가 모두 병합됩니다. 애플리케이션은 DatasetMergeHandler를 통해 병합에 대한 알림을 받습니다. 핸들러는 루트 데이터 세트의 이름과 루트 데이터 세트의 병합으로 표시된 데이터 세트 이름의 어레이를 받습니다.

DatasetMergeHandler가 구현되지 않으면 이러한 데이터 세트가 무시되지만, 자격 증명의 20개 최대 총 데이터 세트에서 공간을 계속 소비합니다.

클라이언트 수준에서 데이터 세트 병합 핸들러를 설정할 수 있습니다.

```
client.datasetMergedHandler = {
    (datasetName: String!, datasets: [AnyObject]!) -> Void in
    for nameObject in datasets {
        if let name = nameObject as? String {
            let merged = AWSCognito.defaultCognito().openOrCreateDataset(name)
            merged.clear()
            merged.synchronize()
        }
    }
}
```

또는 데이터 세트 수준에서 해당 핸들러를 설정할 수 있습니다.

```
dataset.datasetMergedHandler = {
    (datasetName: String!, datasets: [AnyObject]!) -> Void in
    for nameObject in datasets {
        if let name = nameObject as? String {
            let merged = AWSCognito.defaultCognito().openOrCreateDataset(name)
            // do something with the data if it differs from existing dataset
              ...
              // now delete it
            merged.clear()
            merged.synchronize()
        }
    }
}
```

JavaScript

동기화 콜백

데이터 세트에 대해 synchronize()를 수행하면 선택적으로 콜백을 지정하여 다음의 각 상태를 처리할 수 있습니다.

```
dataset.synchronize({

   onSuccess: function(dataset, newRecords) {
      //...
   },

   onFailure: function(err) {
      //...
```

```
   },

   onConflict: function(dataset, conflicts, callback) {
      //...
   },

   onDatasetDeleted: function(dataset, datasetName, callback) {
      //...
   },

  onDatasetMerged: function(dataset, datasetNames, callback) {
      //...
  }

});
```

onSuccess()

onSuccess() 콜백은 동기화 스토어에서 데이터 세트를 성공적으로 업데이트한 경우 트리거됩니다. 콜백을 정의하지 않으면 동기화가 자동으로 수행됩니다.

```
onSuccess: function(dataset, newRecords) {
   console.log('Successfully synchronized ' + newRecords.length + ' new records.');
}
```

onFailure()

동기화 중 예외가 발생할 경우 onFailure()가 호출됩니다. 콜백을 정의하지 않으면 동기화가 자동으로 실패합니다.

```
onFailure: function(err) {
   console.log('Synchronization failed.');
   console.log(err);
}
```

onConflict()

로컬 스토어와 동기화 스토어에서 동일한 키가 수정된 경우 충돌이 발생할 수 있습니다. onConflict() 메서드는 충돌 해결을 처리합니다. 이 메서드를 구현하지 않으면 충돌이 있는 경우 동기화가 중단됩니다.

```
onConflict: function(dataset, conflicts, callback) {
```

```
   var resolved = [];

   for (var i=0; i<conflicts.length; i++) {

      // Take remote version.
      resolved.push(conflicts[i].resolveWithRemoteRecord());

      // Or... take local version.
      // resolved.push(conflicts[i].resolveWithLocalRecord());

      // Or... use custom logic.
      // var newValue = conflicts[i].getRemoteRecord().getValue() +
 conflicts[i].getLocalRecord().getValue();
      // resolved.push(conflicts[i].resovleWithValue(newValue);

   }

   dataset.resolve(resolved, function() {
      return callback(true);
   });

   // Or... callback false to stop the synchronization process.
   // return callback(false);

}
```

onDatasetDeleted()

데이터 세트가 삭제되면 Amazon Cognito 클라이언트는 `onDatasetDeleted()` 콜백을 사용하여 로컬로 캐시된 데이터 세트의 사본도 삭제해야 하는지 여부를 결정합니다. 기본적으로 데이터 세트는 삭제되지 않습니다.

```
onDatasetDeleted: function(dataset, datasetName, callback) {

   // Return true to delete the local copy of the dataset.
   // Return false to handle deleted datasets outside the synchronization callback.

   return callback(true);

}
```

onDatasetMerged()

이전에 연결되지 않은 두 자격 증명이 서로 연결되면 해당 데이터 세트가 모두 병합됩니다. 애플리케이션은 onDatasetsMerged() 콜백을 통해 병합에 대한 알림을 받습니다.

```
onDatasetMerged: function(dataset, datasetNames, callback) {

   // Return true to continue the synchronization process.
   // Return false to handle dataset merges outside the synchronization callback.

   return callback(false);

}
```

Unity

데이터 세트를 열거나 생성한 후 Synchronize 메서드를 사용할 때 트리거되는 여러 콜백을 해당 데이터 세트에 설정할 수 있습니다. 데이터 세트에 콜백을 등록하는 방법은 다음과 같습니다.

```
dataset.OnSyncSuccess += this.HandleSyncSuccess;
dataset.OnSyncFailure += this.HandleSyncFailure;
dataset.OnSyncConflict = this.HandleSyncConflict;
dataset.OnDatasetMerged = this.HandleDatasetMerged;
dataset.OnDatasetDeleted = this.HandleDatasetDeleted;
```

SyncSuccess 및 SyncFailure는 = 대신 +=를 사용하므로 이에 대해 둘 이상의 콜백을 구독할 수 있습니다.

OnSyncSuccess

OnSyncSuccess 콜백은 클라우드에서 데이터 세트를 성공적으로 업데이트한 경우 트리거됩니다. 콜백을 정의하지 않으면 동기화가 자동으로 수행됩니다.

```
private void HandleSyncSuccess(object sender, SyncSuccessEvent e)
{
    // Continue with your game flow, display the loaded data, etc.
}
```

OnSyncFailure

동기화 중 예외가 발생할 경우 OnSyncFailure가 호출됩니다. 콜백을 정의하지 않으면 동기화가 자동으로 실패합니다.

```
private void HandleSyncFailure(object sender, SyncFailureEvent e)
{
    Dataset dataset = sender as Dataset;
    if (dataset.Metadata != null) {
        Debug.Log("Sync failed for dataset : " + dataset.Metadata.DatasetName);
    } else {
        Debug.Log("Sync failed");
    }
    // Handle the error
    Debug.LogException(e.Exception);
}
```

OnSyncConflict

로컬 스토어와 동기화 스토어에서 동일한 키가 수정된 경우 충돌이 발생할 수 있습니다. `OnSyncConflict` 콜백은 충돌 해결을 처리합니다. 이 메서드를 구현하지 않으면 충돌이 있는 경우 동기화가 중단됩니다.

```
private bool HandleSyncConflict(Dataset dataset, List < SyncConflict > conflicts)
{
  if (dataset.Metadata != null) {
    Debug.LogWarning("Sync conflict " + dataset.Metadata.DatasetName);
  } else {
    Debug.LogWarning("Sync conflict");
  }
  List < Amazon.CognitoSync.SyncManager.Record > resolvedRecords = new List <
 Amazon.CognitoSync.SyncManager.Record > ();
  foreach(SyncConflict conflictRecord in conflicts) {
    // SyncManager provides the following default conflict resolution methods:
    //      ResolveWithRemoteRecord - overwrites the local with remote records
    //      ResolveWithLocalRecord - overwrites the remote with local records
    //      ResolveWithValue - to implement your own logic
    resolvedRecords.Add(conflictRecord.ResolveWithRemoteRecord());
  }
  // resolves the conflicts in local storage
  dataset.Resolve(resolvedRecords);
  // on return true the synchronize operation continues where it left,
  //      returning false cancels the synchronize operation
  return true;
}
```

OnDatasetDeleted

데이터 세트가 삭제되면 Amazon Cognito 클라이언트는 OnDatasetDeleted 콜백을 사용하여 로컬로 캐시된 데이터 세트의 사본도 삭제해야 하는지 여부를 결정합니다. 기본적으로 데이터 세트는 삭제되지 않습니다.

```
private bool HandleDatasetDeleted(Dataset dataset)
  {
      Debug.Log(dataset.Metadata.DatasetName + " Dataset has been deleted");
      // Do clean up if necessary
      // returning true informs the corresponding dataset can be purged in the local
 storage and return false retains the local dataset
      return true;
  }
```

OnDatasetMerged

이전에 연결되지 않은 두 자격 증명이 서로 연결되면 해당 데이터 세트가 모두 병합됩니다. 애플리케이션은 OnDatasetsMerged 콜백을 통해 병합에 대한 알림을 받습니다.

```
public bool HandleDatasetMerged(Dataset localDataset, List<string> mergedDatasetNames)
{
    foreach (string name in mergedDatasetNames)
    {
        Dataset mergedDataset = syncManager.OpenOrCreateDataset(name);
        //Lambda function to delete the dataset after fetching it
        EventHandler<SyncSuccessEvent> lambda;
        lambda = (object sender, SyncSuccessEvent e) => {
            ICollection<string> existingValues = localDataset.GetAll().Values;
            ICollection<string> newValues = mergedDataset.GetAll().Values;

            //Implement your merge logic here

            mergedDataset.Delete(); //Delete the dataset locally
            mergedDataset.OnSyncSuccess -= lambda; //We don't want this callback to be
 fired again
            mergedDataset.OnSyncSuccess += (object s2, SyncSuccessEvent e2) => {
                localDataset.Synchronize(); //Continue the sync operation that was
 interrupted by the merge
            };
            mergedDataset.Synchronize(); //Synchronize it as deleted, failing to do so
 will leave us in an inconsistent state
        };
        mergedDataset.OnSyncSuccess += lambda;
        mergedDataset.Synchronize(); //Asnchronously fetch the dataset
```

```
    }

    // returning true allows the Synchronize to continue and false stops it
    return false;
}
```

Xamarin

데이터 세트를 열거나 생성한 후 Synchronize 메서드를 사용할 때 트리거되는 여러 콜백을 해당 데이터 세트에 설정할 수 있습니다. 데이터 세트에 콜백을 등록하는 방법은 다음과 같습니다.

```
dataset.OnSyncSuccess += this.HandleSyncSuccess;
dataset.OnSyncFailure += this.HandleSyncFailure;
dataset.OnSyncConflict = this.HandleSyncConflict;
dataset.OnDatasetMerged = this.HandleDatasetMerged;
dataset.OnDatasetDeleted = this.HandleDatasetDeleted;
```

SyncSuccess 및 SyncFailure는 = 대신 +=를 사용하므로 이에 대해 둘 이상의 콜백을 구독할 수 있습니다.

OnSyncSuccess

OnSyncSuccess 콜백은 클라우드에서 데이터 세트를 성공적으로 업데이트한 경우 트리거됩니다. 콜백을 정의하지 않으면 동기화가 자동으로 수행됩니다.

```
private void HandleSyncSuccess(object sender, SyncSuccessEventArgs e)
{
    // Continue with your game flow, display the loaded data, etc.
}
```

OnSyncFailure

동기화 중 예외가 발생할 경우 OnSyncFailure가 호출됩니다. 콜백을 정의하지 않으면 동기화가 자동으로 실패합니다.

```
private void HandleSyncFailure(object sender, SyncFailureEventArgs e)
{
    Dataset dataset = sender as Dataset;
    if (dataset.Metadata != null) {
        Console.WriteLine("Sync failed for dataset : " + dataset.Metadata.DatasetName);
    } else {
        Console.WriteLine("Sync failed");
```

```
    }
}
```

OnSyncConflict

로컬 스토어와 동기화 스토어에서 동일한 키가 수정된 경우 충돌이 발생할 수 있습니다. `OnSyncConflict` 콜백은 충돌 해결을 처리합니다. 이 메서드를 구현하지 않으면 충돌이 있는 경우 동기화가 중단됩니다.

```
private bool HandleSyncConflict(Dataset dataset, List < SyncConflict > conflicts)
{
  if (dataset.Metadata != null) {
    Console.WriteLine("Sync conflict " + dataset.Metadata.DatasetName);
  } else {
    Console.WriteLine("Sync conflict");
  }
  List < Amazon.CognitoSync.SyncManager.Record > resolvedRecords = new List <
 Amazon.CognitoSync.SyncManager.Record > ();
  foreach(SyncConflict conflictRecord in conflicts) {
    // SyncManager provides the following default conflict resolution methods:
    //      ResolveWithRemoteRecord - overwrites the local with remote records
    //      ResolveWithLocalRecord - overwrites the remote with local records
    //      ResolveWithValue - to implement your own logic
    resolvedRecords.Add(conflictRecord.ResolveWithRemoteRecord());
  }
  // resolves the conflicts in local storage
  dataset.Resolve(resolvedRecords);
  // on return true the synchronize operation continues where it left,
  //      returning false cancels the synchronize operation
  return true;
}
```

OnDatasetDeleted

데이터 세트가 삭제되면 Amazon Cognito 클라이언트는 `OnDatasetDeleted` 콜백을 사용하여 로컬로 캐시된 데이터 세트의 사본도 삭제해야 하는지 여부를 결정합니다. 기본적으로 데이터 세트는 삭제되지 않습니다.

```
private bool HandleDatasetDeleted(Dataset dataset)
{
    Console.WriteLine(dataset.Metadata.DatasetName + " Dataset has been deleted");
    // Do clean up if necessary
```

```
    // returning true informs the corresponding dataset can be purged in the local
 storage and return false retains the local dataset
    return true;
}
```

OnDatasetMerged

이전에 연결되지 않은 두 자격 증명이 서로 연결되면 해당 데이터 세트가 모두 병합됩니다. 애플리케이션은 OnDatasetsMerged 콜백을 통해 병합에 대한 알림을 받습니다.

```
public bool HandleDatasetMerged(Dataset localDataset, List<string> mergedDatasetNames)
{
    foreach (string name in mergedDatasetNames)
    {
        Dataset mergedDataset = syncManager.OpenOrCreateDataset(name);

            //Implement your merge logic here

        mergedDataset.OnSyncSuccess += lambda;
        mergedDataset.SynchronizeAsync(); //Asnchronously fetch the dataset
    }

    // returning true allows the Synchronize to continue and false stops it
    return false;
}
```

푸시 동기화

⚠ Amazon Cognito Sync를 처음 사용하는 경우 AWS AppSync를 사용하세요. Amazon Cognito Sync와 마찬가지로, AWS AppSync도 디바이스 사이에서 애플리케이션 데이터를 동기화하는 서비스입니다.
앱 기본 설정이나 게임 상태 같은 사용자 데이터를 동기화할 수 있습니다. 또한 이러한 기능을 더욱 확장해, 복수의 사용자가 공유 데이터를 실시간으로 동기화하고 협업할 수 있게 합니다.

Amazon Cognito는 자격 증명과 디바이스 간의 연결 관계를 자동으로 추적합니다. 푸시 동기화 기능을 사용하면 자격 증명 데이터가 변경될 경우 지정한 자격 증명의 모든 인스턴스가 통지되도록 할 수 있습니다. 특정 자격 증명에 대해 동기화 스토어 데이터가 변경될 때마다 푸시 동기화는 해당 자격 증명과 연결된 모든 디바이스가 변경 사항에 대해 알리는 자동 푸시 알림을 받도록 합니다.

> **Note**
>
> 푸시 동기화는 JavaScript, Unity 또는 Xamarin에 대해 지원되지 않습니다.

먼저 푸시 동기화에 대한 계정을 설정하고 Amazon Cognito 콘솔에서 푸시 동기화를 활성화해야 푸시 동기화를 사용할 수 있습니다.

Amazon Simple Notification Service(Amazon SNS) 앱 생성

SNS 개발자 가이드의 설명에 따라, 지원되는 플랫폼에 대해 Amazon SNS 앱을 생성하고 구성합니다.

Amazon Cognito 콘솔에서 푸시 동기화 사용

Amazon Cognito 콘솔을 통해 푸시 동기화를 활성화할 수 있습니다. 콘솔 홈 페이지에서 다음을 수행합니다.

1. 푸시 동기화를 활성화할 자격 증명 풀 이름을 클릭합니다. 자격 증명 풀에 대한 대시보드 페이지가 표시됩니다.
2. 대시보드(Dashboard) 페이지의 우측 상단 모서리에서 자격 증명 풀 관리(Manage Identity Pools)를 클릭합니다. 연동 자격 증명 페이지가 나타납니다.
3. 아래로 스크롤하고 푸시 동기화를 클릭하여 확장합니다.
4. 서비스 역할 드롭다운 메뉴에서 SNS 알림을 전송할 권한을 Cognito에 부여하는 IAM 역할을 선택합니다. AWS IAM 콘솔에서 [역할 생성(Create role)]을 클릭하여 자격 증명 풀과 연결된 역할을 생성하거나 수정합니다.
5. 플랫폼 애플리케이션을 선택한 다음 변경 사항 저장을 클릭합니다.
6. 애플리케이션에 SNS 액세스 권한 부여

AWS Identity and Access Management 콘솔에서 전체 Amazon SNS 액세스 권한을 갖도록 IAM 역할을 구성하거나, 전체 Amazon SNS 액세스 권한이 있는 새 역할을 생성합니다. 다음 역할 트러스트 정책 예제는 Amazon Cognito Sync에 IAM 역할을 수임할 수 있는 제한적인 권한을 부여합니다. Amazon Cognito Sync는 `aws:SourceArn` 조건의 자격 증명 풀과 `aws:SourceAccount` 조건의 계정을 둘 다 대신하는 경우에만 역할을 수임할 수 있습니다.

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

```
            "Effect": "Allow",
            "Principal": {
                "Service": "cognito-sync.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "AWS:SourceAccount": "123456789012"
                },
                "ArnLike": {
                    "AWS:SourceArn": "arn:aws:cognito-identity:us-
east-1:123456789012:identitypool/us-east-1:177a950c-2c08-43f0-9983-28727EXAMPLE"
                }
            }
        }
    ]
}
```

IAM 역할에 대한 자세한 내용은 역할(위임 및 연동)을 참조하세요.

앱에서 푸시 동기화 사용: Android

애플리케이션에서는 Google Play 서비스를 가져와야 합니다. Android SDK Manager를 통해 최신 버전의 Google Play SDK를 다운로드할 수 있습니다. Android 구현의 Android 설명서에 따라 앱을 등록하고 GCM으로부터 등록 ID를 받습니다. 등록 ID가 있는 경우 아래 조각에 표시된 대로 Amazon Cognito를 사용하여 디바이스를 등록해야 합니다.

```
String registrationId = "MY_GCM_REGISTRATION_ID";
try {
    client.registerDevice("GCM", registrationId);
} catch (RegistrationFailedException rfe) {
    Log.e(TAG, "Failed to register device for silent sync", rfe);
} catch (AmazonClientException ace) {
    Log.e(TAG, "An unknown error caused registration for silent sync to fail", ace);
}
```

이제 디바이스를 구독하여 특정 데이터 세트에서 업데이트를 받을 수 있습니다.

```
Dataset trackedDataset = client.openOrCreateDataset("myDataset");
if (client.isDeviceRegistered()) {
    try {
        trackedDataset.subscribe();
```

```
    } catch (SubscribeFailedException sfe) {
        Log.e(TAG, "Failed to subscribe to datasets", sfe);
    } catch (AmazonClientException ace) {
        Log.e(TAG, "An unknown error caused the subscription to fail", ace);
    }
}
```

데이터 세트에서 푸시 알림을 받는 것을 중지하려면 unsubscribe 메서드를 호출하면 됩니다. `CognitoSyncManager` 객체에서 모든 데이터 세트(또는 특정 하위 집합)를 구독하려면 `subscribeAll()`을 사용하세요.

```
if (client.isDeviceRegistered()) {
    try {
        client.subscribeAll();
    } catch (SubscribeFailedException sfe) {
        Log.e(TAG, "Failed to subscribe to datasets", sfe);
    } catch (AmazonClientException ace) {
        Log.e(TAG, "An unknown error caused the subscription to fail", ace);
    }
}
```

Android BroadcastReceiver 객체의 구현에서 수정된 데이터 세트의 최신 버전을 확인하고 앱에서 다시 동기화해야 하는지 여부를 결정할 수 있습니다.

```
@Override
public void onReceive(Context context, Intent intent) {

    PushSyncUpdate update = client.getPushSyncUpdate(intent);

    // The update has the source (cognito-sync here), identityId of the
    // user, identityPoolId in question, the non-local sync count of the
    // data set and the name of the dataset. All are accessible through
    // relevant getters.

    String source = update.getSource();
    String identityPoolId = update.getIdentityPoolId();
    String identityId = update.getIdentityId();
    String datasetName = update.getDatasetName;
    long syncCount = update.getSyncCount;

    Dataset dataset = client.openOrCreateDataset(datasetName);
```

```
    // need to access last sync count. If sync count is less or equal to
    // last sync count of the dataset, no sync is required.

    long lastSyncCount = dataset.getLastSyncCount();
    if (lastSyncCount < syncCount) {
        dataset.synchronize(new SyncCallback() {
            // ...
        });
    }

}
```

다음 키는 푸시 알림 페이로드에서 사용할 수 있습니다.

- `source`: cognito-sync. 이 키는 알림 간 차별화 요소의 역할을 수행할 수 있습니다.
- `identityPoolId`: 자격 증명 풀 ID입니다. 수신자의 관점에서 필수가 아닌 경우에도 확인 또는 추가 정보를 위해 이 키를 사용할 수 있습니다.
- `identityId`: 풀 내의 자격 증명 ID입니다.
- `datasetName`: 업데이트된 데이터 세트의 이름입니다. openOrCreateDataset 호출을 위해 이 키를 사용할 수 있습니다.
- `syncCount`: 원격 데이터 세트에 대한 동기화 수입니다. 로컬 데이터 세트가 최신이 아니며 수신되는 동기화가 최신임을 확인하는 방법으로 이 키를 사용할 수 있습니다.

앱에서 푸시 동기화 사용: iOS - Objective-C

앱에 대한 디바이스 토큰을 얻으려면 원격 알림 등록의 Apple 설명서를 따르세요. APN에서 NSData 객체로 디바이스 토큰을 받은 경우 아래에 표시된 대로 동기화 클라이언트의 `registerDevice:` 메서드를 사용하여 디바이스를 Amazon Cognito에 등록해야 합니다.

```
AWSCognito *syncClient = [AWSCognito defaultCognito];
    [[syncClient registerDevice: devToken] continueWithBlock:^id(AWSTask *task) {
        if(task.error){
            NSLog(@"Unable to registerDevice: %@", task.error);
        } else {
            NSLog(@"Successfully registered device with id: %@", task.result);
        }
        return nil;
      }
    ];
```

디버깅 모드에서는 디바이스를 APN 샌드박스에 등록하고 릴리스 모드에서는 APN에 등록합니다. 특정 데이터 세트에서 업데이트를 받으려면 subscribe 메서드를 사용하세요.

```
[[[syncClient openOrCreateDataset:@"MyDataset"] subscribe]
 continueWithBlock:^id(AWSTask *task) {
        if(task.error){
            NSLog(@"Unable to subscribe to dataset: %@", task.error);
        } else {
            NSLog(@"Successfully subscribed to dataset: %@", task.result);
        }
        return nil;
      }
    ];
```

데이터 세트에서 푸시 알림을 받는 것을 중지하려면 unsubscribe 메서드를 호출하면 됩니다.

```
[[[syncClient openOrCreateDataset:@”MyDataset”] unsubscribe]
 continueWithBlock:^id(AWSTask *task) {
        if(task.error){
            NSLog(@"Unable to unsubscribe from dataset: %@", task.error);
        } else {
            NSLog(@"Successfully unsubscribed from dataset: %@", task.result);
        }
        return nil;
      }
    ];
```

AWSCognito 객체에서 모든 데이터 세트를 구독하려면 subscribeAll을 호출하세요.

```
[[syncClient subscribeAll] continueWithBlock:^id(AWSTask *task) {
        if(task.error){
            NSLog(@"Unable to subscribe to all datasets: %@", task.error);
        } else {
            NSLog(@"Successfully subscribed to all datasets: %@", task.result);
        }
        return nil;
      }
    ];
```

subscribeAll을 호출하기 전에 각 데이터 세트에 대해 한 번 이상 동기화하여 데이터 세트가 서버에 존재하도록 해야 합니다.

푸시 알림에 대응하려면 앱 위임에 didReceiveRemoteNotification 메서드를 구현해야 합니다.

```
- (void)application:(UIApplication *)application didReceiveRemoteNotification:
(NSDictionary *)userInfo
    {
        [[NSNotificationCenter defaultCenter]
 postNotificationName:@"CognitoPushNotification" object:userInfo];
    }
```

알림 핸들러를 사용하여 알림을 게시한 경우 데이터 세트를 처리하는 애플리케이션의 다른 곳에서 알림에 대응할 수 있습니다. 다음과 같이 알림을 구독할 경우

```
[[NSNotificationCenter defaultCenter] addObserver:self
 selector:@selector(didReceivePushSync:)
    name: :@"CognitoPushNotification" object:nil];
```

알림에 대해 다음과 같이 작동합니다.

```
- (void)didReceivePushSync:(NSNotification*)notification
    {
        NSDictionary * data = [(NSDictionary *)[notification object]
 objectForKey:@"data"];
        NSString * identityId = [data objectForKey:@"identityId"];
        NSString * datasetName = [data objectForKey:@"datasetName"];
        if([self.dataset.name isEqualToString:datasetName] && [self.identityId
 isEqualToString:identityId]){
            [[self.dataset synchronize] continueWithBlock:^id(AWSTask *task) {
                if(!task.error){
                    NSLog(@"Successfully synced dataset");
                }
                return nil;
            }];
        }
    }
```

다음 키는 푸시 알림 페이로드에서 사용할 수 있습니다.

- source: cognito-sync. 이 키는 알림 간 차별화 요소의 역할을 수행할 수 있습니다.
- identityPoolId: 자격 증명 풀 ID입니다. 수신자의 관점에서 필수가 아닌 경우에도 확인 또는 추가 정보를 위해 이 키를 사용할 수 있습니다.

- `identityId`: 풀 내의 자격 증명 ID입니다.
- `datasetName`: 업데이트된 데이터 세트의 이름입니다. `openOrCreateDataset` 호출을 위해 이 키를 사용할 수 있습니다.
- `syncCount`: 원격 데이터 세트에 대한 동기화 수입니다. 로컬 데이터 세트가 최신이 아니며 수신되는 동기화가 최신임을 확인하는 방법으로 이 키를 사용할 수 있습니다.

앱에서 푸시 동기화 사용: iOS - Swift

앱에 대한 디바이스 토큰을 얻으려면 원격 알림 등록의 Apple 설명서를 따르세요. APN에서 NSData 객체로 디바이스 토큰을 받은 경우 아래에 표시된 대로 동기화 클라이언트의 registerDevice: 메서드를 사용하여 디바이스를 Amazon Cognito에 등록해야 합니다.

```
let syncClient = AWSCognito.default()
syncClient.registerDevice(devToken).continueWith(block: { (task: AWSTask!) ->
 AnyObject! in
    if (task.error != nil) {
        print("Unable to register device: " + task.error.localizedDescription)

    } else {
        print("Successfully registered device with id: \(task.result)")
    }
    return task
})
```

디버깅 모드에서는 디바이스를 APN 샌드박스에 등록하고 릴리스 모드에서는 APN에 등록합니다. 특정 데이터 세트에서 업데이트를 받으려면 `subscribe` 메서드를 사용하세요.

```
syncClient.openOrCreateDataset("MyDataset").subscribe().continueWith(block: { (task:
 AWSTask!) -> AnyObject! in
  if (task.error != nil) {
      print("Unable to subscribe to dataset: " + task.error.localizedDescription)

  } else {
      print("Successfully subscribed to dataset: \(task.result)")
  }
  return task
})
```

데이터 세트에서 푸시 알림을 받는 것을 중지하려면 `unsubscribe` 메서드를 호출하세요.

```
syncClient.openOrCreateDataset("MyDataset").unsubscribe().continueWith(block: { (task:
 AWSTask!) -> AnyObject! in
  if (task.error != nil) {
      print("Unable to unsubscribe to dataset: " + task.error.localizedDescription)

  } else {
      print("Successfully unsubscribed to dataset: \(task.result)")
  }
  return task
})
```

AWSCognito 객체에서 모든 데이터 세트를 구독하려면 subscribeAll을 호출하세요.

```
syncClient.openOrCreateDataset("MyDataset").subscribeAll().continueWith(block: { (task:
 AWSTask!) -> AnyObject! in
  if (task.error != nil) {
      print("Unable to subscribe to all datasets: " + task.error.localizedDescription)

  } else {
      print("Successfully subscribed to all datasets: \(task.result)")
  }
  return task
})
```

subscribeAll을 호출하기 전에 각 데이터 세트에 대해 한 번 이상 동기화하여 데이터 세트가 서버에 존재하도록 해야 합니다.

푸시 알림에 대응하려면 앱 위임에 didReceiveRemoteNotification 메서드를 구현해야 합니다.

```
func application(application: UIApplication, didReceiveRemoteNotification userInfo:
 [NSObject : AnyObject],
  fetchCompletionHandler completionHandler: (UIBackgroundFetchResult) -> Void) {

 NSNotificationCenter.defaultCenter().postNotificationName("CognitoPushNotification",
 object: userInfo)
})
```

알림 핸들러를 사용하여 알림을 게시한 경우 데이터 세트를 처리하는 애플리케이션의 다른 곳에서 알림에 대응할 수 있습니다. 다음과 같이 알림을 구독할 경우

```
NSNotificationCenter.defaultCenter().addObserver(observer:self,
   selector:"didReceivePushSync:",
```

```
    name:"CognitoPushNotification",
    object:nil)
```

알림에 대해 다음과 같이 작동합니다.

```
func didReceivePushSync(notification: NSNotification) {
    if let data = (notification.object as! [String: AnyObject])["data"] as? [String:
 AnyObject] {
        let identityId = data["identityId"] as! String
        let datasetName = data["datasetName"] as! String

        if self.dataset.name == datasetName && self.identityId == identityId {
          dataset.synchronize().continueWithBlock {(task) -> AnyObject! in
              if task.error == nil {
                print("Successfully synced dataset")
              }
              return nil
          }
        }
    }
}
```

다음 키는 푸시 알림 페이로드에서 사용할 수 있습니다.

- `source`: cognito-sync. 이 키는 알림 간 차별화 요소의 역할을 수행할 수 있습니다.
- `identityPoolId`: 자격 증명 풀 ID입니다. 수신자의 관점에서 필수가 아닌 경우에도 확인 또는 추가 정보를 위해 이 키를 사용할 수 있습니다.
- `identityId`: 풀 내의 자격 증명 ID입니다.
- `datasetName`: 업데이트된 데이터 세트의 이름입니다. `openOrCreateDataset` 호출을 위해 이 키를 사용할 수 있습니다.
- `syncCount`: 원격 데이터 세트에 대한 동기화 수입니다. 로컬 데이터 세트가 최신이 아니며 수신되는 동기화가 최신임을 확인하는 방법으로 이 키를 사용할 수 있습니다.

Amazon Cognito 스트림

⚠ Amazon Cognito Sync를 처음 사용하는 경우 AWS AppSync를 사용하세요. Amazon Cognito Sync와 마찬가지로, AWS AppSync도 디바이스 사이에서 애플리케이션 데이터를 동기화하는 서비스입니다.

앱 기본 설정이나 게임 상태 같은 사용자 데이터를 동기화할 수 있습니다. 또한 이러한 기능을 더욱 확장해, 복수의 사용자가 공유 데이터를 실시간으로 동기화하고 협업할 수 있게 합니다.

Amazon Cognito 스트림은 개발자에게 Amazon Cognito에 저장된 데이터에 대한 제어와 통찰력을 제공합니다. 개발자는 이제 데이터가 업데이트 및 동기화될 때 이벤트를 수신하도록 Kinesis 스트림을 구성할 수 있습니다. Amazon Cognito는 각 데이터 집합 변경 사항을 사용자가 소유하는 Kinesis 스트림으로 실시간으로 푸시합니다.

Amazon Cognito 스트림을 사용하여 모든 동기화 데이터를 Kinesis로 이동할 수 있습니다. 그러면 추가 분석을 위한 Amazon Redshift와 같이 데이터 웨어하우스 도구로 스트리밍될 수 있습니다. Kinesis에 대한 자세한 내용은 Amazon Kinesis 사용 시작하기를 참조하세요.

스트림 구성

Amazon Cognito 스트림은 Amazon Cognito 콘솔에서 설정할 수 있습니다. Amazon Cognito 콘솔에서 Amazon Cognito 스트림을 사용하려면 게시할 Kinesis 스트림과 선택한 스트림에 이벤트를 게시할 Amazon Cognito 권한을 부여하는 IAM 역할을 선택해야 합니다.

콘솔 홈 페이지에서 다음을 수행합니다.

1. Amazon Cognito 스트림을 설정할 자격 증명 풀 이름을 클릭합니다. 자격 증명 풀에 대한 대시보드 페이지가 표시됩니다.
2. 대시보드(Dashboard) 페이지의 우측 상단 모서리에서 자격 증명 풀 관리(Manage Identity Pools)를 클릭합니다. 연동 자격 증명 관리(Manage Federated Identities) 페이지가 나타납니다.
3. 아래로 스크롤하고 Cognito 스트림을 클릭하여 확장합니다.
4. 스트림 이름 드롭다운 메뉴에서 기존 Kinesis 스트림의 이름을 선택합니다. 또는 스트림 생성을 클릭하여 스트림 하나를 생성하고 스트림 이름과 샤드 수를 입력합니다. 샤드에 대한 자세한 내용과 스트림에 필요한 샤드의 수 예상에 대한 도움말은 Kinesis 개발자 가이드를 참조하세요.
5. [게시 역할(Publish role)] 드롭다운 메뉴에서 스트림을 게시할 Amazon Cognito 권한을 부여하는 IAM 역할을 선택합니다. AWS IAM 콘솔에서 [역할 생성(Create role)]을 클릭하여 자격 증명 풀과 연결된 역할을 생성하거나 수정합니다.
6. 스트림 상태 드롭다운 메뉴에서 활성을 선택하여 스트림 업데이트를 활성화합니다. 변경 사항 저장을 클릭합니다.

Amazon Cognito 스트림을 성공적으로 구성한 후 이 자격 증명 풀의 데이터 세트에 대한 모든 후속 업데이트가 스트림으로 전송됩니다.

스트림 콘텐츠

스트림으로 전송된 각 레코드는 단일 동기화를 나타냅니다. 다음은 스트림으로 전송된 레코드의 예제입니다.

```
{
    "identityPoolId": "Pool Id",
    "identityId": "Identity Id",
    "dataSetName": "Dataset Name",
    "operation": "(replace|remove)",
    "kinesisSyncRecords": [
        {
            "key": "Key",
            "value": "Value",
            "syncCount": 1,
            "lastModifiedDate": 1424801824343,
            "deviceLastModifiedDate": 1424801824343,
            "op": "(replace|remove)"
        },
        ...
    ],
    "lastModifiedDate": 1424801824343,
    "kinesisSyncRecordsURL": "S3Url",
    "payloadType": "(S3Url|Inline)",
    "syncCount": 1
}
```

Kinesis 최대 페이로드 크기인 1MB보다 큰 업데이트의 경우 Amazon Cognito는 업데이트의 전체 콘텐츠가 포함된 미리 서명된 Amazon S3 URL을 포함합니다.

Amazon Cognito 스트림을 구성한 후 Kinesis 스트림을 삭제하거나 Amazon Cognito Sync가 더 이상 역할을 맡을 수 없도록 역할 신뢰 권한을 변경하는 경우 Amazon Cognito 스트림을 해제합니다. Kinesis 스트림을 다시 생성하거나 역할을 수정한 다음 스트림을 다시 설정해야 합니다.

대량 게시

Amazon Cognito 스트림을 구성한 경우 자격 증명 풀의 기존 데이터에 대해 대량 게시 작업을 실행할 수 있습니다. 대량 게시 작업을 시작한 후 Amazon Cognito는 콘솔을 통하거나 API를 통해 직접 업데이트를 수신하는 동일한 스트림에 이 데이터의 게시를 시작합니다.

Amazon Cognito는 대량 게시 작업을 사용할 때 스트림에 전송된 데이터의 고유성을 보장하지 않습니다. 업데이트 및 대량 게시의 일부로 동일한 업데이트를 받을 수 있습니다. 스트림에서 레코드를 처리할 때 이 점을 유의하시기 바랍니다.

모든 스트림을 대량으로 게시하려면 스트림 구성의 1-6단계를 따른 다음 대량 게시 시작(Start bulk publish)을 클릭합니다. 대량 게시 작업은 지정된 시간에 한 번으로 제한되며 대량 게시 요청은 24시간마다 한 번으로 제한됩니다.

Amazon Cognito 이벤트

> ⚠ Amazon Cognito Sync를 처음 사용하는 경우 AWS AppSync를 사용하세요. Amazon Cognito Sync와 마찬가지로, AWS AppSync도 디바이스 사이에서 애플리케이션 데이터를 동기화하는 서비스입니다.
> 앱 기본 설정이나 게임 상태 같은 사용자 데이터를 동기화할 수 있습니다. 또한 이러한 기능을 더욱 확장해, 복수의 사용자가 공유 데이터를 실시간으로 동기화하고 협업할 수 있게 합니다.

Amazon Cognito 이벤트를 사용하면 Amazon Cognito에서 중요한 이벤트에 대한 응답으로 AWS Lambda 함수를 실행할 수 있습니다. Amazon Cognito는 데이터 집합이 동기화될 경우 동기화 트리거 이벤트를 유발합니다. 사용자가 데이터를 업데이트할 때 동기화 트리거 이벤트를 사용하여 작업을 수행할 수 있습니다. 이 함수는 클라우드에 저장되고 사용자의 다른 디바이스에 동기화되기 전에 데이터를 평가하고 선택적으로 조작합니다. 사용자의 다른 디바이스에 동기화되기 전에 디바이스에서 제공된 데이터를 검증하거나 플레이어가 새 레벨에 도달할 때 상 제공 등 수신 데이터에 따라 데이터 세트의 다른 값을 업데이트하는 데 유용합니다.

아래 단계는 Amazon Cognito 데이터 집합이 동기화될 때마다 실행하는 Lambda 함수를 설정하는 방법을 보여줍니다.

> **Note**
>
> Amazon Cognito 이벤트를 사용할 때는 Amazon Cognito Identity에서 받은 자격 증명만 사용할 수 있습니다. 연결된 Lambda 함수가 있지만 AWS 계정 자격 증명(개발자 자격 증명)을 사용하여 `UpdateRecords`를 호출하는 경우 Lambda 함수가 호출되지 않습니다.

AWS Lambda에서 함수 생성

Amazon Cognito와 Lambda를 통합하려면 먼저 Lambda에서 함수를 생성해야 합니다. 그렇게 하려면 다음을 수행하세요.

Amazon Cognito에서 Lambda 함수 선택

1. Lambda 콘솔을 엽니다.
2. [Lambda 함수 생성(Create a Lambda function)]을 클릭합니다.
3. 블루프린트 선택(Select blueprint) 화면에서 "cognito-sync-trigger"를 검색하고 선택합니다.
4. 이벤트 소스 구성(Configure event sources) 화면에서 이벤트 소스 유형을 "Cognito Sync 트리거"로 설정해 두고 자격 증명 풀을 선택합니다. 다음(Next)을 클릭합니다.

> **Note**
>
> 콘솔 외부에서 Amazon Cognito Sync 트리거를 구성하는 경우 Amazon Cognito가 함수를 호출할 수 있도록 Lambda 리소스 기반 권한을 추가해야 합니다. Lambda 콘솔에서(AWS Lambda에 대한 리소스 기반 정책 사용 참조) 또는 Lambda AddPermission 작업을 사용하여 이 권한을 추가할 수 있습니다.
> Lambda 리소스 기반 정책 예제
> 다음 AWS Lambda 리소스 기반 정책은 Amazon Cognito에 Lambda 함수를 호출할 수 있는 제한적인 권한을 부여합니다. Amazon Cognito는 `aws:SourceArn` 조건의 자격 증명 풀과 `aws:SourceAccount` 조건의 계정을 대신하는 경우에만 함수를 호출할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Id": "default",
    "Statement": [
        {
            "Sid": "lambda-allow-cognito-my-function",
            "Effect": "Allow",
            "Principal": {
                "Service": "cognito-sync.amazonaws.com"
            },
            "Action": "lambda:InvokeFunction",
            "Resource": "<your Lambda function ARN>",
            "Condition": {
                "StringEquals": {
                    "AWS:SourceAccount": "<your account number>"
                },
                "ArnLike": {
                    "AWS:SourceArn": "<your identity pool ARN>"
```

```
                }
            }
        }
    ]
}
```

5. 함수 구성(Configure function) 화면에서 함수 이름과 설명을 입력합니다. 런타임(Runtime)을 "Node.js"로 설정합니다. 이 예제에서는 코드를 변경하지 않고 그대로 둡니다. 기본 예제는 동기화되는 데이터를 변경하지 않으며 Amazon Cognito Sync 트리거 이벤트가 발생했다는 사실만 기록합니다. 핸들러 이름을 "index.handler"로 설정해 둡니다. 역할의 경우 AWS Lambda에 액세스할 코드 권한을 부여하는 IAM 역할을 선택합니다. 역할을 수정하려면 IAM 콘솔을 참조하세요. 고급(Advanced) 설정을 변경하지 않고 그대로 둡니다. 다음(Next)을 클릭합니다.
6. 검토(Review) 화면에서 세부 정보를 검토하고 함수 생성(Create function)을 클릭합니다. 다음 페이지에는 새 Lambda 함수가 표시됩니다.

Lambda에 작성된 적절한 함수가 있는 경우 해당 함수를 Amazon Cognito Sync 트리거 이벤트에 대한 핸들러로 선택해야 합니다. 아래 단계는 이 프로세스를 소개합니다.

콘솔 홈 페이지에서 다음을 수행합니다.

1. Amazon Cognito 이벤트를 설정할 자격 증명 풀 이름을 클릭합니다. 자격 증명 풀에 대한 대시보드 페이지가 표시됩니다.
2. 대시보드(Dashboard) 페이지의 우측 상단 모서리에서 연동 자격 증명 관리(Manage Federated Identities)를 클릭합니다. 연동 자격 증명 관리(Manage Federated Identities) 페이지가 나타납니다.
3. 아래로 스크롤하고 Cognito 이벤트(Cognito Events)를 클릭하여 확장합니다.
4. 동기화 트리거 드롭다운 메뉴에서 동기화 이벤트가 발생할 때 트리거할 Lambda 함수를 선택합니다.
5. 변경 사항 저장을 클릭합니다.

이제 데이터 세트가 동기화될 때마다 Lambda 함수가 실행됩니다. 다음 섹션에서는 함수에서 데이터가 동기화될 때 이 데이터를 읽고 수정하는 방법에 대해 설명합니다.

동기화 트리거에 대한 Lambda 함수 작성

동기화 트리거는 서비스 공급자 인터페이스에서 사용하는 프로그래밍 패턴을 따릅니다. Amazon Cognito는 Lambda 함수에 입력을 다음 JSON 형식으로 제공합니다.

```
{
  "version": 2,
  "eventType": "SyncTrigger",
  "region": "us-east-1",
  "identityPoolId": "identityPoolId",
  "identityId": "identityId",
  "datasetName": "datasetName",
  "datasetRecords": {
    "SampleKey1": {
      "oldValue": "oldValue1",
      "newValue": "newValue1",
      "op": "replace"
    },
    "SampleKey2": {
      "oldValue": "oldValue2",
      "newValue": "newValue2",
      "op": "replace"
    },..
  }
}
```

Amazon Cognito는 입력과 동일한 형식으로 함수의 반환 값을 예상합니다.

동기화 트리거 이벤트에 대한 함수를 작성할 때 다음 사항을 준수합니다.

- Amazon Cognito가 UpdateRecords 동안 Lambda 함수를 호출하면 함수는 5초 이내에 응답해야 합니다. 그렇지 않으면 Amazon Cognito Sync 서비스에서 `LambdaSocketTimeoutException` 예외가 발생합니다. 이 제한 시간 값은 늘릴 수 없습니다.
- `LambdaThrottledException` 예외가 발생하면 동기화 작업을 다시 시도하여 레코드를 업데이트하세요.
- Amazon Cognito는 데이터 세트에 있는 모든 레코드를 함수에 대한 입력으로 제공합니다.
- 앱 사용자가 업데이트하는 레코드에 `replace`로 설정된 op 필드가 있습니다. 삭제된 레코드에 `remove`로 설정된 op 필드가 있습니다.
- 앱 사용자가 레코드를 업데이트하지 않은 경우에도 레코드를 수정할 수 있습니다.
- datasetRecords를 제외한 모든 필드는 읽기 전용이며 변경할 수 없습니다. 이러한 필드를 변경하면 레코드를 업데이트할 수 없습니다.
- 레코드 값을 수정하려면 이 값을 업데이트하고 op를 `replace`로 업데이트합니다.
- 레코드를 제거하려면 op를 `remove`로 설정하거나 이 값을 null로 설정합니다.

- 레코드를 추가하려면 datasetRecords 배열에 새 레코드를 추가합니다.
- Amazon Cognito는 Amazon이 레코드를 업데이트할 때 응답에서 생략된 레코드를 무시합니다.

샘플 Lambda 함수

다음은 데이터에 액세스하고 데이터를 수정 및 제거하는 방법을 보여주는 Lambda 함수의 예입니다.

```
console.log('Loading function');

exports.handler = function(event, context) {
    console.log(JSON.stringify(event, null, 2));

    //Check for the event type
    if (event.eventType === 'SyncTrigger') {

        //Modify value for a key
        if('SampleKey1' in event.datasetRecords){
            event.datasetRecords.SampleKey1.newValue = 'ModifyValue1';
            event.datasetRecords.SampleKey1.op = 'replace';
        }

        //Remove a key
        if('SampleKey2' in event.datasetRecords){
            event.datasetRecords.SampleKey2.op = 'remove';
        }

        //Add a key
        if(!('SampleKey3' in event.datasetRecords)){
            event.datasetRecords.SampleKey3={'newValue':'ModifyValue3', 'op' :
 'replace'};
        }

    }
    context.done(null, event);
};
```

Amazon Cognito 콘솔 사용

Amazon Cognito 콘솔을 사용하여 사용자 풀 및 자격 증명 풀을 생성 및 관리할 수 있습니다.

이 가이드에서는 Amazon Cognito step-by-step 콘솔에서의 일반적인 Amazon Cognito 사용자 풀 작업에 대한 안내를 제공합니다.

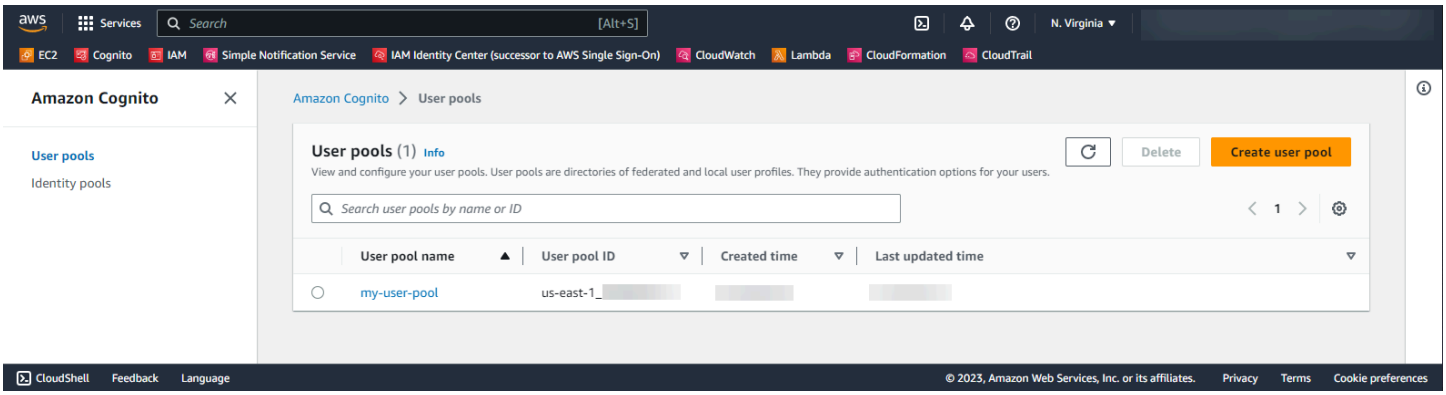
Amazon Cognito 콘솔을 사용하려면

1. Amazon Cognito를 사용하려면 계정을 등록해야 합니다. AWS
2. Amazon Cognito 콘솔로 이동합니다. AWS 자격 증명을 입력하라는 메시지가 표시될 수 있습니다.
3. 사용자 풀을 생성하거나 편집하려면 왼쪽 탐색 창에서 [사용자 풀(User Pools)]을 선택합니다.

   자세한 정보는 사용자 풀 시작하기을 참조하세요.
4. 자격 증명 풀을 생성하거나 편집하려면 자격 증명 풀을 선택합니다. Amazon Cognito 자격 증명 풀의 기존 콘솔로 리디렉션됩니다.

   자세한 정보는 Amazon Cognito 자격 증명 풀 시작하기을 참조하세요.

Amazon Cognito 콘솔은 계정 및 AWS Management Console청구에 대한 정보를 제공하는 의 일부입니다. 자세한 내용은 AWS Management Console작업 단원을 참조하세요.



주제

- 사용자 풀 콘솔
- 자격 증명 풀 콘솔

사용자 풀 콘솔

Amazon Cognito 콘솔의 사용자 풀 보기에 있는 목록에서 사용자 풀을 선택하여 세부 정보를 확인합니다. 세부 보기에서 콘솔 상단에 있는 사용자 풀 개요에는 사용자 풀에 대한 기본 정보가 포함되어 있습니다. 다음 탭은 사용자 풀 구성을 관련 기능으로 구성한 것입니다.

사용자

사용자 탭에는 사용자 및 CSV 파일에서 가져온 사용자에 대한 정보가 들어 있습니다. 이 탭에서 사용자를 추가, 제거, 편집할 수 있습니다.

참조

- 사용자 풀의 사용자 관리
- CSV 파일에서 사용자 풀로 사용자 가져오기

그룹

그룹 탭에는 사용자 그룹에 대한 정보가 들어 있습니다. 그룹의 멤버십을 추가, 수정 및 변경하고 자격 증명 풀 통합을 위해 그룹과 연결된 IAM 역할을 변경할 수 있습니다.

참조

- 사용자 풀에 그룹 추가

로그인 경험

로그인 경험 탭에는 사용자가 사용자 풀에 로그인하는 방법에 대한 정보가 들어 있습니다. 이 탭에는 서드 파티 ID 제공업체, 사용자 이름 옵션, 암호 정책, 다중 인증(MFA) 구성, 암호 찾기 동작, 디바이스 기억 등이 있습니다. ID 제공업체를 추가 및 수정하고 사용자 풀의 전반적인 로그인 동작을 변경할 수 있습니다.

참조

- 서드 파티를 통한 사용자 풀 로그인 추가
- 로그인 속성 사용자 지정
- 사용자 풀 암호 요구 사항 추가
- 사용자 풀에 MFA 추가
- 사용자 계정 복구
- 사용자 풀의 사용자 디바이스 작업

가입 경험

가입 경험 탭에는 셀프 서비스 가입, 필수 속성, 전화번호 및 이메일 주소 검증, 사용자 지정 속성에 대한 정보가 포함되어 있습니다.

참조

- 사용자 계정 가입 및 확인
- 사용자 풀 속성
- 가입 시 연락처 정보 확인

메시지 전송

메시징 탭에는 사용자에게 이메일 및 SMS 메시지를 보내는 데 사용할 AWS 서비스 및 사용자에게 보낼 메시지 형식에 대한 정보가 포함되어 있습니다.

참조

- Amazon Cognito 사용자 풀에 대한 이메일 설정
- Amazon Cognito 사용자 풀의 SMS 메시지 설정
- SMS 및 이메일 확인 메시지와 사용자 초대 메시지 구성

앱 통합

앱 통합 탭에는 사용자 풀 앱 클라이언트, 사용자 풀 서비스 엔드포인트에 할당하는 도메인, API 리소스 서버, 호스팅 UI 및 고급 보안에 대한 정보가 포함되어 있습니다. 각 앱 클라이언트를 자세히 분석하여 다음을 구성할 수 있습니다.

1. 토큰 설정
2. 콜백 URL
3. 인증 흐름
4. 속성 권한
5. 앱별 고급 보안 및 호스팅 UI 설정
6. Amazon Pinpoint 분석

참조

- 사용자 풀 앱 클라이언트
- Amazon Cognito 호스팅 UI 및 페더레이션 엔드포인트 설정 및 사용
- 사용자 풀 도메인 구성
- 리소스 서버를 통한 범위, M2M 및 API 인증

- 사용자 풀에 고급 보안 기능 추가
- Amazon Cognito 사용자 풀에서 Amazon Pinpoint 분석 사용

사용자 풀 속성

사용자 풀 속성 탭에는 Lambda 트리거 AWS WAF , 웹 ACL 보호, 삭제 보호, 리소스 태그 등 사용자와 직접 관련이 없는 사용자 풀 구성에 대한 정보가 포함되어 있습니다.

참조

- Lambda 트리거를 사용하여 사용자 풀 워크플로 사용자 정의
- 웹 ACL을 사용자 풀과 AWS WAF 연결
- 사용자 풀 삭제 방지
- 리소스에 태그 지정 AWS

자격 증명 풀 콘솔

Amazon Cognito 콘솔의 자격 증명 풀 보기에 있는 목록에서 자격 증명 풀을 선택하여 세부 정보를 확인합니다. 세부 보기에서 콘솔 상단에 있는 자격 증명 풀 개요에는 사용자 풀에 대한 기본 정보가 포함되어 있습니다. 다음 탭은 사용자 풀 구성을 관련 기능으로 구성한 것입니다.

사용자 통계

사용자 통계 탭에는 자격 증명 풀에서 자격 증명을 생성한 사용자에 대한 통계 정보가 표시됩니다. 이 탭에서는 자격 증명 풀 설정을 구성할 수 없습니다.

자격 증명 브라우저

자격 증명 브라우저 탭에는 사용자가 자격 증명 풀에서 생성한 개별 자격 증명에 대한 정보가 들어 있습니다. 자격 증명을 보고 삭제할 수 있습니다.

참조

- Amazon Cognito 자격 증명 풀 시작하기

사용자 액세스

사용자 액세스 탭에는 자격 증명 풀에 연결한 ID 제공업체, 개발자 제공업체, 자격 증명에 할당된 기본 IAM 역할, 인증되지 않은 게스트 액세스 구성에 대한 정보가 포함되어 있습니다. 각 ID 제공업체를 자세히 분석하여 다음을 구성할 수 있습니다.

1. IAM 역할 선택을 통한 역할 기반 액세스 제어

2. 액세스 제어 속성을 사용한 속성 기반 액세스 제어

참조

- 자격 증명 풀 외부 자격 증명 공급자
- IAM 역할
- 인증된 자격 증명 및 인증되지 않은 자격 증명
- 개발자 인증 자격 증명(자격 증명 풀)
- 역할 기반 액세스 제어 사용
- 액세스 제어에 속성 사용

자격 증명 풀 속성

자격 증명 풀 속성 탭에는 기본(클래식) 인증 및 리소스 태그와 같은 기타 자격 증명 풀 구성에 대한 정보가 포함되어 있습니다.

- 자격 증명 풀(페더레이션 자격 증명) 인증 흐름
- 리소스 태그 지정 AWS

Amazon Cognito의 보안

클라우드 AWS 보안이 최우선 과제입니다. AWS 고객은 가장 보안에 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

보안은 기업과 기업 간의 AWS 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호하는 역할을 합니다. AWS AWS 또한 안전하게 사용할 수 있는 서비스를 제공합니다. Amazon Cognito에 적용되는 규정 준수 프로그램에 대해 자세히 알아보려면 규정 준수 프로그램별 범위 내AWS 서비스 규정 준수 참조하십시오.
- 클라우드에서의 보안 — 사용하는 AWS 서비스에 따라 책임이 결정됩니다. 또한 여러분은 데이터의 민감도, 회사 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다

이 문서는 Amazon Cognito를 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 보안 및 규정 준수 목표에 맞게 Amazon Cognito를 구성하는 방법을 보여줍니다. 또한 Amazon Cognito 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법도 알아봅니다.

내용

Amazon Cognito의 데이터 보호

AWS 공유 책임 모델 공유 책임 모델 이 모델에 설명된 대로 은 (AWS 는) 모든 클라우드를 실행하는 글로벌 인프라를 보호하는 역할을 합니다 AWS . 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야

합니다. 이 콘텐츠에는 사용하는 AWS 서비스의 보안 구성 및 관리 작업이 포함됩니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요.

데이터 보호를 위해 AWS 계정 자격 증명을 보호하고 AWS Identity and Access Management (IAM) 을 사용하여 개별 사용자 계정을 설정하는 것이 좋습니다. 이러한 방식에서는 각 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 리소스와 통신하세요. AWS
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다. AWS CloudTrail
- AWS 서비스 내의 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용하십시오.
- Amazon S3에 저장된 개인 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용합니다.

이름 필드와 같은 자유 형식 필드에 고객 계정 번호와 같은 중요 식별 정보를 절대 입력하지 마십시오. 여기에는 콘솔 AWS CLI, API 또는 SDK를 사용하여 Amazon Cognito 또는 기타 AWS 서비스를 사용하는 경우가 포함됩니다. AWS Amazon Cognito 또는 기타 서비스에 입력하는 모든 데이터를 진단 로그에 포함할 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명 정보를 URL에 포함시키지 마십시오.

데이터 암호화

데이터 암호화는 일반적으로 저장 데이터 암호화와 전송 중 데이터 암호화의 두 가지 범주로 나뉩니다.

저장된 데이터 암호화

Amazon Cognito 내의 데이터는 업계 표준에 따라 암호화된 상태로 저장됩니다.

전송 중 데이터 암호화

관리형 서비스인 Amazon Cognito는 AWS 글로벌 네트워크 보안으로 보호됩니다. AWS 보안 서비스 및 인프라 AWS 보호 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하십시오. 인프라 보안 모범 사례를 사용하여 AWS 환경을 설계하려면 Security Pillar AWS Well-Architected Framework의 인프라 보호를 참조하십시오.

AWS 게시된 API 호출을 사용하여 네트워크를 통해 Amazon Cognito에 액세스할 수 있습니다. 고객은 다음을 지원해야 합니다.

- 전송 계층 보안(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.

Amazon Cognito 사용자 풀과 자격 증명 풀에는 IAM 인증, 비인증 및 토큰 승인 API 작업이 있습니다. 비인증 및 토큰 승인 API 작업은 앱의 최종 사용자인 고객을 대상으로 합니다. 비인증 및 토큰 승인 API 작업은 저장 및 전송 중에 암호화됩니다. 자세한 정보는 Amazon Cognito 사용자 풀 인증 및 미인증 API 작업을 참조하세요.

> **Note**
>
> Amazon Cognito는 사용자의 콘텐츠를 내부적으로 암호화하며 고객 제공 키를 지원하지 않습니다.

Amazon Cognito의 Identity and Access Management

AWS Identity and Access Management (IAM) 은 관리자가 리소스에 대한 액세스를 안전하게 제어할 수 AWS 서비스 있도록 AWS 도와줍니다. IAM 관리자는 어떤 사용자가 Amazon Cognito 리소스를 사용할 수 있는 인증(로그인) 및 권한(권한 있음)을 받을 수 있는지 제어합니다. IAM은 추가 AWS 서비스 비용 없이 사용할 수 있습니다.

주제

- 고객
- ID를 통한 인증
- 정책을 사용한 액세스 관리
- Amazon Cognito에서 IAM을 사용하는 방법
- Amazon Cognito의 자격 증명 기반 정책 예제
- Amazon Cognito 자격 증명 및 액세스 문제 해결
- Amazon Cognito에 서비스 연결 역할 사용

고객

Amazon Cognito에서 수행하는 작업에 따라 사용 방법 AWS Identity and Access Management (IAM) 이 다릅니다.

서비스 사용자 - Amazon Cognito 서비스를 사용하여 작업을 수행하는 경우 필요한 보안 인증 정보와 권한을 관리자가 제공합니다. 더 많은 Amazon Cognito 기능을 사용하여 작업을 수행하게 되면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방식을 이해하면 적절한 권한을 관리자에게 요청할 수 있습니다. Amazon Cognito의 기능에 액세스할 수 없는 경우 Amazon Cognito 자격 증명 및 액세스 문제 해결 섹션을 참조하세요.

서비스 관리자 - 회사에서 Amazon Cognito 리소스를 책임지고 있는 경우 Amazon Cognito에 대한 전체 액세스 권한을 가지고 있을 것입니다. 서비스 관리자는 서비스 사용자가 액세스해야 하는 Amazon Cognito 기능과 리소스를 결정합니다. 그런 다음, IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해하십시오. 회사가 Amazon Cognito에서 IAM을 사용하는 방법에 대해 자세히 알아보려면 Amazon Cognito에서 IAM을 사용하는 방법 섹션을 참조하세요.

IAM 관리자 - IAM 관리자라면 Amazon Cognito에 대한 액세스 권한 관리 정책 작성 방법을 자세히 알고 싶을 것입니다. IAM에서 사용할 수 있는 Amazon Cognito 자격 증명 기반 정책 예제를 보려면 Amazon Cognito의 자격 증명 기반 정책 예제 섹션을 참조하세요.

ID를 통한 인증

인증은 자격 증명 자격 증명을 AWS 사용하여 로그인하는 방법입니다. IAM 사용자로 인증 (로그인 AWS) 하거나 IAM 역할을 맡아 인증 (로그인) 해야 합니다. AWS 계정 루트 사용자

ID 소스를 통해 제공된 자격 증명을 사용하여 페더레이션 ID로 로그인할 수 있습니다. AWS AWS IAM Identity Center (IAM ID 센터) 사용자, 회사의 싱글 사인온 인증, Google 또는 Facebook 자격 증명이 페더레이션 ID의 예입니다. 연동 자격 증명으로 로그인할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 페더레이션을 사용하여 액세스하는 경우 AWS 간접적으로 역할을 맡게 됩니다.

사용자 유형에 따라 AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. 로그인에 대한 자세한 내용은 AWS 로그인 사용 설명서의 내 로그인 방법을 참조하십시오. AWS AWS 계정

AWS 프로그래밍 방식으로 액세스하는 경우 자격 증명을 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트 (SDK) 와 명령줄 인터페이스 (CLI) 를 AWS 제공합니다. AWS 도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. 권장 방법을 사용하여 직접 요청에 서명하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 AWS API 요청 서명을 참조하십시오.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어, AWS 계정의 보안을 강화하기 위해 다단계 인증 (MFA) 을 사용할 것을 권장합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서의 다중 인증 및 IAM 사용 설명서의 AWS에서 다중 인증(MFA) 사용을 참조하세요.

AWS 계정 루트 사용자

계정을 AWS 계정만들 때는 먼저 계정의 모든 AWS 서비스 리소스에 대한 완전한 액세스 권한을 가진 하나의 로그인 ID로 시작합니다. 이 ID를 AWS 계정 루트 사용자라고 하며, 계정을 만들 때 사용한 이메일 주소와 비밀번호로 로그인하여 액세스할 수 있습니다. 일상적인 태스크에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 보안 인증 정보를 보호하고 루트 사용자만 수행할 수 있는 태스크를 수행하는 데 사용하세요. 루트 사용자로 로그인해야 하는 태스크의 전체 목록은 IAM 사용자 안내서의 루트 사용자 보안 인증이 필요한 태스크를 참조하세요.

연동 자격 증명

가장 좋은 방법은 관리자 액세스가 필요한 사용자를 비롯한 수동 AWS 서비스 사용자가 ID 공급자와의 페더레이션을 사용하여 임시 자격 증명을 사용하여 액세스하도록 하는 것입니다.

페더레이션 ID는 기업 사용자 디렉토리, 웹 ID 공급자, Identity Center 디렉터리의 사용자 또는 ID 소스를 통해 제공된 자격 증명을 사용하여 액세스하는 AWS 서비스 모든 사용자를 말합니다. AWS Directory Service페더레이션 ID에 AWS 계정액세스하면 이들이 역할을 맡고 역할은 임시 자격 증명을 제공합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center(을)를 사용하는 것이 좋습니다. IAM Identity Center에서 사용자 및 그룹을 생성하거나 자체 ID 소스의 사용자 및 그룹 집합에 연결하고 동기화하여 모든 사용자 및 애플리케이션에서 사용할 수 있습니다. AWS 계정 IAM Identity Center에 대한 자세한 내용은AWS IAM Identity Center 사용 설명서에서 IAM Identity Center란 무엇입니까?를 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 사용자 또는 애플리케이션에 대한 특정 권한을 AWS 계정 가진 사용자 내 자격 증명입니다. 가능하면 암호 및 액세스 키와 같은 장기 자격 증명이 있는 IAM 사용자를 생성하는 대신 임시 자격 증명을 사용하는 것이 좋습니다. 하지만 IAM 사용자의 장기 자격 증명이 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 참조하세요.

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 사용

자 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 인증을 가지고 있지만, 역할은 임시 보안 인증만 제공합니다. 자세한 정보는 IAM 사용 설명서의 IAM 사용자를 만들어야 하는 경우(역할이 아님)를 참조하세요.

IAM 역할

IAM 역할은 특정 권한을 가진 사용자 AWS 계정 내의 자격 증명입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. 역할을 AWS Management Console 전환하여 에서 일시적으로 IAM 역할을 맡을 수 있습니다. AWS CLI 또는 AWS API 작업을 호출하거나 사용자 지정 URL을 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 정보는 IAM 사용 설명서의 IAM 역할 사용을 참조하세요.

임시 보안 인증이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 페더레이션 사용자 액세스 - 연동 자격 증명에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 연동 자격 증명이 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 역할에 대한 자세한 내용은 IAM 사용 설명서의 타사 자격 증명 공급자의 역할 만들기를 참조하세요. IAM Identity Center를 사용하는 경우 권한 세트를 구성합니다. 인증 후 아이덴티티가 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 세트를 IAM의 역할과 연관 짓습니다. 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 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 서비스경우에는 역할을 프록시로 사용하는 대신 정책을 리소스에 직접 연결할 수 있습니다. 크로스 계정 액세스를 위한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하세요.
- 서비스 간 액세스 — 일부는 다른 AWS 서비스서비스의 기능을 AWS 서비스 사용합니다. 예컨대, 어떤 서비스에서 호출을 수행하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는 호출하는 보안 주체의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.
  - 순방향 액세스 세션 (FAS) — IAM 사용자 또는 역할을 사용하여 작업을 수행하는 경우 보안 AWS 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을

수행할 수 있습니다. FAS는 전화를 거는 주체의 권한을 다운스트림 AWS 서비스서비스에 AWS 서비스 요청하기 위한 요청과 결합하여 사용합니다. FAS 요청은 다른 서비스 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 서비스가 수신한 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 에 연결된 서비스 역할의 한 유형입니다. AWS 서비스서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은 사용자에게 AWS 계정 표시되며 해당 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

- Amazon EC2에서 실행되는 애플리케이션 — IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 API 요청을 AWS CLI 하는 애플리케이션의 임시 자격 증명을 관리할 수 있습니다. AWS 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS 역할을 할당하고 모든 애플리케이션에서 사용할 수 있게 하려면 인스턴스에 연결된 인스턴스 프로필을 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.

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

정책을 사용한 액세스 관리

정책을 생성하고 이를 AWS ID 또는 리소스에 AWS 연결하여 액세스를 제어할 수 있습니다. 정책은 ID 또는 리소스와 연결될 때 AWS 해당 권한을 정의하는 객체입니다. AWS 주도자 (사용자, 루트 사용자 또는 역할 세션) 가 요청할 때 이러한 정책을 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는 지를 결정합니다. 대부분의 정책은 JSON 문서로 AWS 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 정보는 IAM 사용 설명서의 JSON 정책 개요를 참조하세요.

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

기본적으로, 사용자와 역할에는 어떠한 권한도 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 수임할 수 있습니다.

IAM 정책은 작업을 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, `iam:GetRole`태스크를 허용하는 정책이 있다고 가정합니다. 해당 정책을 사용하는 사용자는 AWS Management Console, AWS CLI, 또는 AWS API에서 역할 정보를 가져올 수 있습니다.

ID 기반 정책

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 자격 증명에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는 지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

자격 증명 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은 내 여러 사용자, 그룹 및 역할에 연결할 수 있는 독립형 정책입니다. AWS 계정관리형 정책에는 AWS 관리형 정책과 고객 관리형 정책이 포함됩니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책과 인라인 정책의 선택을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는 지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 연동 사용자 등이 포함될 수 있습니다. AWS 서비스

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. IAM의 AWS 관리형 정책은 리소스 기반 정책에 사용할 수 없습니다.

액세스 제어 목록(ACLs)

액세스 제어 목록(ACL)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACLs는 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

ACL을 지원하는 서비스의 예로는 아마존 S3와 아마존 VPC가 있습니다. AWS WAF ACL에 대해 자세히 알아보려면 Amazon Simple Storage Service 개발자 안내서의 액세스 제어 목록(ACL) 개요를 참조하세요.

기타 정책 타입

AWS 일반적이지 않은 추가 정책 유형을 지원합니다. 이러한 정책 타입은 더 일반적인 정책 타입에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 – 권한 경계는 보안 인증 기반 정책에 따라 IAM 엔터티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 엔터티의 자격 증명 기반 정책과 그 권한 경계의 교집합입니다. `Principal` 필드에서 사용자나 역할을 보안 주체로 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 정보는 IAM 사용 설명서의 IAM 엔터티에 대한 권한 경계를 참조하세요.
- 서비스 제어 정책 (SCP) - SCP는 조직 또는 조직 단위 (OU) 에 대한 최대 권한을 지정하는 JSON 정책입니다. AWS Organizations AWS Organizations 사업체가 소유한 여러 AWS 계정 개를 그룹화하고 중앙에서 관리하는 서비스입니다. 조직에서 모든 기능을 활성화할 경우 서비스 제어 정책 (SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 구성원 계정의 엔티티 (각 엔티티 포함) 에 대한 권한을 제한합니다. AWS 계정 루트 사용자조직 및 SCP에 대한 자세한 정보는 AWS Organizations 사용 설명서의 SCP 작동 방식을 참조하세요.
- 세션 정책 – 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할 자격 증명 기반 정책의 교차 및 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 정보는 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 타입

여러 정책 타입이 요청에 적용되는 경우 결과 권한은 이해하기가 더 복잡합니다. 여러 정책 유형이 관련되어 있을 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하십시오.

Amazon Cognito에서 IAM을 사용하는 방법

IAM을 사용하여 Amazon Cognito에 대한 액세스를 관리하기 전에 Amazon Cognito에서 사용할 수 있는 IAM 기능을 알아보세요.

Amazon Cognito에서 사용할 수 있는 IAM 기능

IAM 특성	Amazon Cognito 지원
ID 기반 정책	예
리소스 기반 정책	아니요
정책 작업	예
정책 리소스	예
정책 조건 키	예
ACLs	아니요
ABAC(정책 내 태그)	부분
임시 보안 인증	예
보안 주체 권한	아니요
서비스 역할	예
서비스 연결 역할	예

Amazon Cognito 및 AWS 기타 서비스가 대부분의 IAM 기능과 어떻게 작동하는지 자세히 알아보려면 IAM 사용 설명서의 IAM과 함께 작동하는 서비스를AWS 참조하십시오.

Amazon Cognito의 자격 증명 기반 정책

ID 기반 정책 지원	예

자격 증명 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 자격 증명에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는 지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

IAM 자격 증명 기반 정책을 사용하면 허용되거나 거부되는 작업와 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. 자격 증명 기반 정책에서는 보안 주체가 연결된 사용자 또는 역할에 적용되므로 보안 주체를 지정할 수 없습니다. JSON 정책에서 사용하는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

Amazon Cognito의 자격 증명 기반 정책 예제

Amazon Cognito 자격 증명 기반 정책 예제를 보려면 Amazon Cognito의 자격 증명 기반 정책 예제 섹션을 참조하세요.

Amazon Cognito 내의 리소스 기반 정책

리소스 기반 정책 지원	아니요

리소스 기반 정책은 리소스에 연결하는 JSON 정책 문서입니다. 리소스 기반 정책의 예는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는 지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 연동 사용자 등이 포함될 수 있습니다. AWS 서비스

계정 간 액세스를 활성화하려는 경우 전체 계정이나 다른 계정의 IAM 엔터티를 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. 보안 주체와 리소스가 다른 AWS 계정경우 신뢰할 수 있는 계정의 IAM 관리자는 보안 주체 개체 (사용자 또는 역할) 에게 리소스에 액세스할 수 있는 권한도 부여해야 합니다. 개체에 자격 증명 기반 정책을 연결하여 권한을 부여합니다. 하지만 리소스 기반 정책이 동일 계정의 보안 주체에 액세스를 부여하는 경우 추가 자격 증명 기반 정책이 필요하지 않습니다. 자세한 정보는 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하세요.

Amazon Cognito의 정책 작업

정책 작업 지원	예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 `Action`요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 태스크를 설명합니다. 정책 작업은 일반적으로 관련 AWS API 작업과 이름이 같습니다. 일치하는 API 작업이 없는 권한 전용 작업 같은 몇 가지 예외도 있습니다. 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 일컬어 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하십시오.

Amazon Cognito 작업 목록을 보려면 서비스 권한 부여 참조에서 Amazon Cognito에서 정의한 작업을 참조하세요.

Amazon Cognito의 정책 작업은 작업 앞에 다음 접두사를 사용합니다.

```
cognito-identity
```

단일 문에서 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
      "cognito-identity:action1",
      "cognito-identity:action2"
         ]
```

서명된 API vs 서명되지 않은 API

AWS 자격 증명을 사용하여 Amazon Cognito API 요청에 서명하면 AWS Identity and Access Management (IAM) 정책에서 해당 요청을 제한할 수 있습니다. AWS 보안 인증 정보로 서명해야 하는 API 요청에는 `AdminInitiateAuth`를 사용한 서버 측 로그인과 `UpdateUserPool`과 같은 Amazon Cognito 리소스를 생성, 확인 또는 수정하는 작업이 포함됩니다. 서명된 API 요청에 대한 자세한 내용은 API 요청 서명을 AWS 참조하십시오.

Amazon Cognito 일반 사용자에게 제공하려는 앱을 위한 소비자 ID 제품이므로 다음과 같은 서명되지 않은 API에 액세스할 수 있습니다. 앱은 사용자와 잠재 사용자를 위해 이러한 API 요청을 수행합니다. 일부 API는 새 인증 세션을 시작하기 위해 `InitiateAuth`와 같은 사전 승인이 필요하지 않습니다. 일부 API는 인증된 기존 세션이 있는 사용자에 대한 MFA 설정을 완료하기 위해 `VerifySoftwareToken`과 같이 권한 부여를 위한 액세스 토큰 또는 세션 키를 사용합니다. 서명되지 않고 승인된 Amazon Cognito 사용자 풀 API는 Amazon Cognito API 참조에 표시된 요청 구문에서 `Session` 또는 `AccessToken` 파라미터를 지원합니다. 서명되지 않은 Amazon Cognito Identity API는 Amazon Cognito Federated Identities API 참조에 표시된 대로 `IdentityId` 파라미터를 지원합니다.

Amazon Cognito 사용자 풀 API 작업의 인증 모델과 역할에 대한 자세한 내용은 Amazon Cognito 사용자 풀 인증 및 미인증 API 작업을 참조하세요.

Amazon Cognito 자격 증명 풀 API 작업

- GetId
- GetOpenIdToken
- GetCredentialsForIdentity
- UnlinkIdentity

Amazon Cognito 사용자 풀 API 작업

- AssociateSoftwareToken
- ChangePassword
- ConfirmDevice
- ConfirmForgotPassword
- ConfirmSignUp
- DeleteUser
- DeleteUserAttributes
- ForgetDevice
- ForgotPassword
- GetDevice
- GetUser
- GetUserAttributeVerificationCode
- GlobalSignOut
- InitiateAuth
- ListDevices
- ResendConfirmationCode
- RespondToAuthChallenge
- RevokeToken
- SetUserMFAPreference
- SetUserSettings
- SignUp
- UpdateAuthEventFeedback
- UpdateDeviceStatus

- `UpdateUserAttributes`
- `VerifySoftwareToken`
- `VerifyUserAttribute`

Amazon Cognito 자격 증명 기반 정책 예제를 보려면 Amazon Cognito의 자격 증명 기반 정책 예제 섹션을 참조하세요.

Amazon Cognito의 정책 리소스

정책 리소스 지원	예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지 지정할 수 있습니다.

`Resource` JSON 정책 요소는 작업이 적용되는 하나 이상의 개체를 지정합니다. 문장에는 `Resource`또는 `NotResource`요소가 반드시 추가되어야 합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한이라고 하는 특정 리소스 타입을 지원하는 작업에 대해 이 작업을 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

Amazon 리소스 이름(ARN)

Amazon Cognito 페더레이션 자격 증명의 ARN

Amazon Cognito 자격 증명 풀(페더레이션 자격 증명)에서는 다음 예와 같이 Amazon 리소스 이름(ARN) 형식을 사용하여 특정 자격 증명 풀에 대한 IAM 사용자의 액세스를 제한할 수 있습니다. ARN에 대한 자세한 내용은 IAM 식별자를 참조하세요.

```
arn:aws:cognito-identity:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID
```

Amazon Cognito Sync의 ARN

Amazon Cognito Sync에서도 고객은 자격 증명 풀 ID, 자격 증명 ID 및 데이터 집합 이름으로 액세스를 제한할 수 있습니다.

자격 증명 풀에 대해 작동하는 API의 경우, 자격 증명 풀 ARN 형식은 Amazon Cognito 페더레이션 자격 증명과 동일하지만 서비스 이름이 `cognito-identity`가 아니라 `cognito-sync`입니다

```
arn:aws:cognito-sync:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID
```

`RegisterDevice`와 같이 단일 자격 증명에 대해 작동하는 API의 경우, 다음 ARN 형식으로 개별 자격 증명을 참조할 수 있습니다.

```
arn:aws:cognito-sync:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID/
identity/IDENTITY_ID
```

`UpdateRecords` 및 `ListRecords`와 같이 데이터 집합에 대해 작동하는 API의 경우, 다음 ARN 형식을 사용하여 개별 데이터 집합을 참조할 수 있습니다.

```
arn:aws:cognito-sync:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID/
identity/IDENTITY_ID/dataset/DATASET_NAME
```

Amazon Cognito 사용자 풀의 ARN

Amazon Cognito 사용자 풀의 경우 다음 ARN 형식을 사용하여 특정 사용자 풀에 대한 사용자의 액세스를 제한할 수 있습니다.

```
arn:aws:cognito-idp:REGION:ACCOUNT_ID:userpool/USER_POOL_ID
```

Amazon Cognito 리소스 유형 및 해당 ARN의 목록을 보려면 서비스 권한 부여 참조에서 Amazon Cognito에서 정의한 리소스를 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 Amazon Cognito에서 정의한 작업을 참조하세요.

Amazon Cognito 자격 증명 기반 정책 예제를 보려면 Amazon Cognito의 자격 증명 기반 정책 예제 섹션을 참조하세요.

Amazon Cognito에 사용되는 정책 조건 키

서비스별 정책 조건 키 지원	예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지 지정할 수 있습니다.

`Condition` 요소(또는 `Condition` 블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. `Condition` 요소는 옵션입니다. 같거나 작음과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다.

한 문에서 여러 `Condition`요소를 지정하거나 단일 `Condition`요소에서 여러 키를 지정하는 경우 AWS 는 논리적 `AND`태스크를 사용하여 평가합니다. 단일 조건 키에 여러 값을 지정하는 경우 는 논리적 `OR` 연산을 사용하여 조건을 AWS 평가합니다. 명문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 예를 들어, IAM 사용자에게 IAM 사용자 이름으로 태그가 지정된 경우에만 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 정책 요소: 변수 및 태그를 참조하세요.

AWS 글로벌 조건 키 및 서비스별 조건 키를 지원합니다. 모든 AWS 글로벌 조건 키를 보려면 IAM 사용 AWS 설명서의 글로벌 조건 컨텍스트 키를 참조하십시오.

Amazon Cognito 조건 키 목록을 보려면 서비스 권한 부여 참조에서 Amazon Cognito의 조건 키를 참조하세요. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 Amazon Cognito에서 정의한 작업을 참조하세요.

Amazon Cognito 자격 증명 기반 정책 예제를 보려면 Amazon Cognito의 자격 증명 기반 정책 예제 섹션을 참조하세요.

Amazon Cognito의 액세스 제어 목록(ACL)

ACL 지원	아니요

액세스 제어 목록(ACLs)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는 지를 제어합니다. ACLs는 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Amazon Cognito에서 속성 기반 액세스 제어(ABAC)

ABAC(정책 내 태그) 지원	부분

속성 기반 액세스 제어(ABAC)는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. AWS에서는 이러한 속성을 태그라고 합니다. IAM 개체 (사용자 또는 역할) 및 여러 AWS 리소스에 태그를 첨부할 수 있습니다. ABAC의 첫 번째 단계로 개체 및 리소스에 태그를 지정합니다. 그런 다음 보안 주체의 태그가 액세스하려는 리소스의 태그와 일치할 때 작업을 허용하도록 ABAC 정책을 설계합니다.

ABAC는 빠르게 성장하는 환경에서 유용하며 정책 관리가 번거로운 상황에 도움이 됩니다.

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 타입에 대해 세 가지 조건 키를 모두 지원하는 경우, 값은 서비스에 대해 예입니다. 서비스가 일부 리소스 타입에 대해서만 세 가지 조건 키를 모두 지원하는 경우, 값은 부분적입니다.

ABAC에 대한 자세한 정보는 IAM 사용 설명서의 ABAC란 무엇인가요?를 참조하세요. ABAC 설정 단계가 포함된 자습서를 보려면 IAM 사용 설명서의 속성 기반 액세스 제어(ABAC) 사용을 참조하세요.

Amazon Cognito에서 임시 보안 인증 정보 사용

임시 보안 인증 지원	예

임시 자격 증명을 사용하여 로그인하면 작동하지 AWS 서비스 않는 것도 있습니다. 임시 자격 증명을 사용하는 방법을 AWS 서비스 비롯한 추가 정보는 IAM 사용 설명서의 IAM과AWS 서비스 연동되는 내용을 참조하십시오.

사용자 이름과 암호를 제외한 다른 방법을 AWS Management Console 사용하여 로그인하면 임시 자격 증명을 사용하는 것입니다. 예를 들어 회사의 SSO (Single Sign-On) 링크를 AWS 사용하여 액세스하는 경우 이 프로세스에서 자동으로 임시 자격 증명을 생성합니다. 또한 콘솔에 사용자로 로그인한 다음 역할을 전환할 때 임시 보안 인증을 자동으로 생성합니다. 역할 전환에 대한 자세한 정보는 IAM 사용 설명서의 역할로 전환(콘솔)을 참조하세요.

또는 API를 사용하여 임시 자격 증명을 수동으로 생성할 수 있습니다 AWS CLI . AWS 그런 다음 해당 임시 자격 증명을 사용하여 액세스할 수 AWS있습니다. AWS 장기 액세스 키를 사용하는 대신 임시 자격 증명을 동적으로 생성할 것을 권장합니다. 자세한 정보는 IAM의 임시 보안 인증 섹션을 참조하세요.

Amazon Cognito의 서비스 간 보안 주체 권한

전달 액세스 세션(FAS) 지원	아니요

IAM 사용자 또는 역할을 사용하여 작업을 수행하는 AWS경우 보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 전화를 거는 주체의 권한을 다운스트림 서비스에 AWS 서비스 요청하기 위한 요청과 함께 사용합니다. AWS 서비스 FAS 요청은 다른 서비스 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 서비스가 수신한 경우에만 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

Amazon Cognito의 서비스 역할

서비스 역할 지원	예

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.

Amazon Cognito 서비스 역할에 대한 자세한 내용은 푸시 동기화 활성화 및 푸시 동기화 섹션을 참조하세요.

> **Warning**
>
> 서비스 역할에 대한 권한을 변경하면 Amazon Cognito 기능이 중단될 수 있습니다. Amazon Cognito가 관련 지침을 제공하는 경우에만 서비스 역할을 편집하세요.

Amazon Cognito의 서비스 연결 역할

서비스 링크 역할 지원	예

서비스 연결 역할은 에 연결된 서비스 역할의 한 유형입니다. AWS 서비스서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은 사용자에게 AWS 계정 표시되며 해당 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

Amazon Cognito 서비스 연결 역할 생성 또는 관리에 대한 자세한 내용은 Amazon Cognito에 서비스 연결 역할 사용 섹션을 참조하세요.

Amazon Cognito의 자격 증명 기반 정책 예제

기본적으로 사용자 및 역할은 Amazon Cognito 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 또한 AWS Management Console, AWS Command Line Interface (AWS CLI) 또는 AWS API를 사용하여 작업을 수행할 수 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 맡을 수 있습니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

각 리소스 유형에 대한 ARN 형식을 비롯하여 Amazon Cognito에서 정의되는 작업 및 리소스 유형에 대한 자세한 내용은 서비스 승인 참조의 Amazon Cognito에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

주제

정책 모범 사례

ID 기반 정책에 따라 계정에서 사용자가 Amazon Cognito 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. 자격 증명 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따르십시오.

- AWS 관리형 정책으로 시작하고 최소 권한 권한으로 이동 — 사용자와 워크로드에 권한을 부여하려면 여러 일반적인 사용 사례에 권한을 부여하는 AWS 관리형 정책을 사용하세요. 해당 내용은 에서 사용할 수 있습니다. AWS 계정사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 더 줄이는 것이 좋습니다. 자세한 정보는 IAM 사용 설명서의 AWS 관리형 정책 또는 AWS 직무에 대한 관리형 정책을 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우 태스크를 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 IAM의 정책 및 권한을 참조하세요.

- IAM 정책의 조건을 사용하여 액세스 추가 제한 – 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어 SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. 예를 AWS 서비스들어 특정 작업을 통해 서비스 작업을 사용하는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 AWS CloudFormation있습니다. 자세한 정보는 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 검증하여 안전하고 기능적인 권한 보장 – IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 신규 및 기존 정책을 검증합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 정보는 IAM 사용 설명서의 IAM Access Analyzer 정책 검증을 참조하세요.
- 멀티 팩터 인증 (MFA) 필요 - IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 AWS 계정 MFA를 활성화하십시오. API 작업을 직접 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 정보는 IAM 사용 설명서의 MFA 보호 API 액세스 구성을 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

> **Note**
>
> Amazon Cognito 콘솔의 원래 버전과 새 버전은 Amazon Cognito 리소스를 보고 수정할 때 기본 동작이 다릅니다. `aws:ViaAWSService` 조건이 true인 경우에만 `cognito-idp` 서비스 접두사에서 작업에 대한 권한을 부여한 경우, 영향을 받는 IAM 보안 주체가 원래 콘솔에서 Amazon Cognito 리소스에 유효할 수 있겠지만 새 콘솔에서는 그렇지 않습니다. Amazon Cognito 콘솔에서 작업하려면 IAM 정책에서 Amazon Cognito 권한에 대한 `aws:ViaAWSService` 조건을 설정하지 마세요.

Amazon Cognito 콘솔 사용

Amazon Cognito 콘솔에 액세스하려면 최소한의 권한 세트가 있어야 합니다. 이러한 권한을 통해 내 Amazon Cognito 리소스를 나열하고 해당 리소스에 대한 세부 정보를 볼 수 있어야 합니다. AWS 계정 최소 필수 권한보다 더 제한적인 자격 증명 기반 정책을 만들면 콘솔이 해당 정책에 연결된 엔터티(사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

AWS CLI 또는 AWS API만 호출하는 사용자에게 최소 콘솔 권한을 허용할 필요는 없습니다. 그 대신, 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.

사용자와 역할이 Amazon Cognito 콘솔을 계속 사용할 수 있도록 하려면 Amazon `ConsoleAccess` Cognito `ReadOnly` AWS 또는 관리형 정책도 엔티티에 연결하십시오. 자세한 내용은 IAM 사용 설명서의 사용자에게 권한 추가를 참조하십시오.

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예시는 IAM 사용자가 자신의 사용자 자격 증명에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 API를 사용하여 프로그래밍 방식으로 이 작업을 완료할 수 있는 AWS CLI 권한이 포함됩니다. AWS

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

특정 자격 증명 풀로 콘솔 액세스 제한

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cognito-identity:ListIdentityPools"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "cognito-identity:*"
      ],
      "Resource": "arn:aws:cognito-identity:us-east-1:0123456789:identitypool/us-
east-1:1a1a1a1a-ffff-1111-9999-12345678"
    },
    {
      "Effect": "Allow",
      "Action": [
        "cognito-sync:*"
      ],
      "Resource": "arn:aws:cognito-sync:us-east-1:0123456789:identitypool/us-
east-1:1a1a1a1a-ffff-1111-9999-12345678"
    }
  ]
}
```

풀의 모든 자격 증명에 특정 데이터 집합에 대한 액세스 허용

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cognito-sync:ListRecords",
        "cognito-sync:UpdateRecords"
      ],
```

```
      "Resource": "arn:aws:cognito-sync:us-east-1:0123456789:identitypool/us-
east-1:1a1a1a1a-ffff-1111-9999-12345678/identity/*/dataset/UserProfile"
    }
  ]
}
```

Amazon Cognito 자격 증명 및 액세스 문제 해결

다음 정보를 사용하여 Amazon Cognito와 IAM에서 작업할 때 발생할 수 있는 공통적인 문제를 진단하고 수정할 수 있습니다.

주제

- Amazon Cognito에서 작업을 수행할 권한이 없음
- 저는 IAM을 수행할 권한이 없습니다. PassRole
- 관리자인데, 다른 사용자가 Amazon Cognito에 액세스하도록 허용하기를 원함
- 내 AWS 계정 외부의 사용자가 내 Amazon Cognito 리소스에 액세스할 수 있도록 허용하고 싶습니다.

Amazon Cognito에서 작업을 수행할 권한이 없음

작업을 수행할 권한이 없다는 오류가 수신되면, 작업을 수행할 수 있도록 정책을 업데이트해야 합니다.

다음 예제 오류는 mateojacksonIAM 사용자가 콘솔을 사용하여 가상 *my-example-widget*리소스에 대한 세부 정보를 보려고 하지만 가상 cognito-identity:*GetWidget*권한이 없을 때 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 cognito-identity:GetWidget on resource: my-example-widget
```

이 경우 cognito-identity:*GetWidget*작업을 사용하여 *my-example-widget*리소스에 액세스할 수 있도록 mateojackson사용자 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

저는 IAM을 수행할 권한이 없습니다. PassRole

`iam:PassRole` 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 Amazon Cognito에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

새 서비스 역할 또는 서비스 연결 역할을 만드는 대신 기존 역할을 해당 서비스에 전달할 AWS 서비스 수 있는 기능도 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예제 오류는 `marymajor`라는 IAM 사용자가 콘솔을 사용하여 Amazon Cognito에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 수 있는 권한을 가지고 있지 않습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우 Mary가 `iam:PassRole`작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요하면 관리자에게 문의하세요. AWS 관리자는 로그인 자격 증명을 제공한 사람입니다.

관리자인데, 다른 사용자가 Amazon Cognito에 액세스하도록 허용하기를 원함

다른 사용자가 Amazon Cognito에 액세스하도록 허용하려면 액세스 권한이 필요한 사용자나 애플리케이션에 대한 IAM 엔터티(사용자 또는 역할)를 생성해야 합니다. 다른 사용자들은 해당 엔터티에 대한 보안 인증을 사용해 AWS에 액세스합니다. 그런 다음 Amazon Cognito에서 올바른 권한을 부여하는 정책을 엔터티에 연결해야 합니다.

바로 시작하려면 IAM 사용 설명서의 첫 번째 IAM 위임 사용자 및 그룹 생성을 참조하십시오.

내 AWS 계정 외부의 사용자가 내 Amazon Cognito 리소스에 액세스할 수 있도록 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하세요.

- Amazon Cognito에서 이러한 기능을 지원하는지 여부를 알아보려면 Amazon Cognito에서 IAM을 사용하는 방법 섹션을 참조하세요.

- 소유하고 AWS 계정 있는 모든 리소스에 대한 액세스를 제공하는 방법을 알아보려면 IAM 사용 설명서의 다른 AWS 계정 IAM 사용자에게 액세스 권한 제공을 참조하십시오.
- 제3자에게 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사 AWS 계정AWS 계정 소유에 대한 액세스 제공을 참조하십시오.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하세요.
- 크로스 계정 액세스를 위한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하세요.

Amazon Cognito에 서비스 연결 역할 사용

Amazon Cognito는 AWS Identity and Access Management (IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 역할을 맡을 수 있도록 허용하는 신뢰 정책이 있는 고유한 유형의 IAM 역할입니다. AWS 서비스 서비스 연결 역할은 Amazon Cognito에서 사전 정의하며 서비스가 사용자를 대신하여 다른 AWS 서비스를 호출하는 데 필요한 모든 권한을 포함합니다.

필요한 권한을 수동으로 추가할 필요가 없으므로 서비스 연결 역할은 Amazon Cognito를 더 쉽게 설정할 수 있습니다. Amazon Cognito에서 서비스 연결 역할의 권한을 정의하므로 다르게 정의되지 않은 한, Amazon Cognito만 해당 역할을 수임할 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며 이 권한 정책은 다른 IAM 엔터티에 연결할 수 없습니다.

먼저 관련 리소스를 삭제한 후에만 서비스 연결 역할을 삭제할 수 있습니다. 이렇게 하면 리소스에 대한 액세스 권한을 실수로 삭제할 수 없기 때문에 Amazon Cognito 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용은 IAM으로 작업하는AWS 서비스를 참조하고 서비스 연결 역할(Service-Linked Role) 열에 예(Yes)가 있는 서비스를 찾아보세요. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 링크가 있는 예를 선택합니다.

Amazon Cognito에 대한 서비스 연결 역할 권한

Amazon Cognito는 다음 서비스 연결 역할을 사용합니다.

- AWSServiceRoleForAmazonCognitoIdpEmailService— Amazon Cognito 사용자 풀 서비스가 Amazon SES 자격 증명을 사용하여 이메일을 보낼 수 있도록 허용합니다.

- AWSServiceRoleForAmazonCognitoIdp— Amazon Cognito 사용자 풀이 Amazon Pinpoint 프로젝트에 대한 이벤트를 게시하고 엔드포인트를 구성할 수 있습니다.

AWSServiceRoleForAmazonCognitoIdpEmailService

`AWSServiceRoleForAmazonCognitoIdpEmailService` 서비스 연결 역할은 역할을 수임하기 위해 다음 서비스를 신뢰합니다.

- `email.cognito-idp.amazonaws.com`

역할 권한 정책은 Amazon Cognito가 지정된 리소스에서 다음 작업을 완료하도록 허용합니다.

허용되는 작업 대상: AWSServiceRoleForAmazonCognitoIdpEmailService

- 작업: `ses:SendEmail` 및 `ses:SendRawEmail`
- 리소스: *

이 정책은 Amazon Cognito가 지정된 리소스에서 다음 작업을 완료하는 기능을 거부합니다.

거부된 작업

- 작업: `ses:List*`
- 리소스: *

이러한 권한을 통해 Amazon Cognito는 Amazon SES에서 확인된 이메일 주소를 사용자에게 이메일로 보내는 용도로만 사용할 수 있습니다. Amazon Cognito는 사용자가 클라이언트 앱에서 사용자 풀에 대한 특정 작업(예: 가입 또는 암호 재설정)을 수행할 때 사용자에게 이메일을 보냅니다.

IAM 엔터티(사용자, 그룹, 역할 등)가 서비스 링크 역할을 생성하고 편집하거나 삭제할 수 있도록 권한을 구성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 권한을 참조하세요.

AWSServiceRoleForAmazonCognitoIdp

AWSServiceRoleForAmazonCognitoIdp 서비스 연결 역할은 다음 서비스가 역할을 맡을 것으로 신뢰합니다.

- `email.cognito-idp.amazonaws.com`

역할 권한 정책은 Amazon Cognito가 지정된 리소스에서 다음 작업을 완료하도록 허용합니다.

허용된 조치 대상 AWSServiceRoleForAmazonCognitoIdp

- 작업: `cognito-idp:Describe`

- 리소스: *

이 권한을 통해 Amazon Cognito는 사용자를 위해 `Describe` Amazon Cognito API 작업을 호출할 수 있습니다.

> **Note**
>
> `createUserPoolClient` 및 `updateUserPoolClient`를 사용하여 Amazon Cognito를 Amazon Pinpoint와 통합하면 리소스 권한이 인라인 정책으로 SLR에 추가됩니다. 이 인라인 정책은 `mobiletargeting:UpdateEndpoint` 및 `mobiletargeting:PutEvents` 권한을 제공합니다. 이러한 권한을 통해 Amazon Cognito는 Cognito와 통합하는 Pinpoint 프로젝트의 이벤트를 게시하고 엔드포인트를 구성할 수 있습니다.

Amazon Cognito에 대한 서비스 연결 역할 생성

서비스 링크 역할은 수동으로 생성할 필요가 없습니다. Amazon SES 구성을 사용하여 AWS Management Console AWS CLI, 또는 Amazon Cognito API에서 이메일 전송을 처리하도록 사용자 풀을 구성하면 Amazon Cognito가 서비스 연결 역할을 생성합니다.

이 서비스 연결 역할을 삭제했다가 다시 생성해야 하는 경우 동일한 프로세스를 사용하여 계정에서 역할을 다시 생성할 수 있습니다. Amazon SES 구성을 사용하여 이메일 전송을 처리하도록 사용자 풀을 구성할 경우 Amazon Cognito가 서비스 연결 역할을 다시 생성합니다.

Amazon Cognito가 이 역할을 생성하려면 먼저 사용자 풀을 설정하는 데 사용하는 IAM 권한에 `iam:CreateServiceLinkedRole` 작업을 포함해야 합니다. IAM에서의 권한 업데이트에 대한 자세한 내용은 IAM 사용 설명서에서 IAM 사용자의 권한 변경을 참조하세요.

Amazon Cognito에 대한 서비스 연결 역할 편집

에서는 AmazonCognitoIdp 또는 AmazonCognitoIdpEmailService 서비스 연결 역할을 편집할 수 없습니다. AWS Identity and Access Management서비스 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 편집을 참조하세요.

Amazon Cognito에 대한 서비스 연결 역할 삭제

서비스 연결 역할이 필요한 기능 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제하는 것이 좋습니다. 이러한 역할을 삭제하면 Amazon Cognito가 적극적으로 모니터링하거나 유지하는 엔터

티만 보유합니다. 역할을 AmazonCognitoIdp 삭제하거나 AmazonCognitoIdpEmailService 서비스에 연결된 역할을 삭제하려면 먼저 해당 역할을 사용하는 각 사용자 풀에 대해 다음 중 하나를 수행해야 합니다.

- 사용자 풀을 삭제합니다.
- 기본 이메일 기능을 사용하도록 사용자 풀의 이메일 설정을 업데이트합니다. 기본 설정에서는 서비스 연결 역할을 사용하지 않습니다.

역할을 사용하는 사용자 풀을 AWS 리전 사용하여 각 그룹에서 작업을 수행해야 한다는 점을 기억하십시오.

> **Note**
>
> 리소스를 삭제하려고 할 때 Amazon Cognito 서비스가 역할을 사용 중이면 삭제에 실패할 수 있습니다. 이 문제가 발생하면 몇 분 기다렸다가 작업을 다시 시도하세요.

Amazon Cognito 사용자 풀을 삭제하려면

1. 에서 Amazon Cognito 콘솔에 AWS Management Console 로그인하고 엽니다. https://console.aws.amazon.com/cognito
2. 사용자 풀 관리를 선택합니다.
3. [사용자 풀(Your User Pools)] 페이지에서 삭제할 사용자 풀을 선택합니다.
4. 풀 삭제를 선택합니다.
5. [사용자 풀 삭제(Delete user pool)] 창에서 **`delete`**를 입력하고 [풀 삭제(Delete pool)]를 선택합니다.

기본 이메일 기능을 사용하도록 Amazon Cognito 사용자 풀을 업데이트하려면

1. 에서 Amazon Cognito 콘솔에 AWS Management Console 로그인하고 엽니다. https://console.aws.amazon.com/cognito
2. 사용자 풀 관리를 선택합니다.
3. [사용자 풀(Your User Pools)] 페이지에서 업데이트할 사용자 풀을 선택합니다.
4. 왼쪽의 탐색 메뉴에서 메시지 사용자 지정을 선택합니다.

5. [Amazon SES 구성을 통해 이메일을 보내시겠습니까?(Do you want to send emails through your Amazon SES Configuration?)] 아래에서 [아니요 - Cognito(기본값)을 사용합니다(No - Use Cognito (Default))]를 선택합니다.
6. 이메일 계정 옵션 설정을 마치면 변경 사항 저장을 선택합니다.

IAM을 사용하여 수동으로 서비스 연결 역할을 삭제하려면

IAM 콘솔 AWS CLI, 또는 AWS API를 사용하여 AmazonCognitoIdpEmailService 서비스에 연결된 역할을 삭제하십시오 AmazonCognitoIdp . 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 삭제를 참조하세요.

Amazon Cognito 서비스 연결 역할이 지원되는 리전

Amazon Cognito는 서비스가 제공되는 모든 AWS 리전 곳에서 서비스 연결 역할을 지원합니다. 자세한 내용은 AWS 리전 및 엔드포인트를 참조하세요.

Amazon Cognito의 로깅 및 모니터링

모니터링은 Amazon Cognito 및 기타 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 데 있어 중요한 부분입니다. Amazon Cognito는 현재 다음 AWS 서비스 를 지원하므로, 조직 및 조직 내에서 발생하는 작업을 모니터링할 수 있습니다.

- AWS CloudTrail — CloudTrail 를 사용하면 Amazon Cognito 콘솔에서 그리고 코드 호출에서 Amazon Cognito API 작업에 대한 API 호출을 캡처할 수 있습니다. 예를 들어, 사용자가 인증할 때 요청의 IP 주소, 요청한 사람, 요청 시기와 같은 세부 정보를 기록할 CloudTrail 수 있습니다.
- Amazon CloudWatch Logs — CloudWatch Logs를 사용하면 사용자 활동에 대한 세밀한 로그를 로그 그룹에 보낼 수 있습니다. 예를 들어, 자세한 사용자 활동 로그를 검토하여 사용자에게 이메일과 SMS 메시지를 전송할 때의 문제를 해결할 수 있습니다.
- Amazon CloudWatch Metrics — CloudWatch 지표를 사용하면 이벤트 발생 시 거의 실시간으로 모니터링, 보고 및 자동 조치를 취할 수 있습니다. 예를 들어 제공된 지표에 CloudWatch 대시보드를 생성하여 Amazon Cognito 사용자 풀을 모니터링하거나 제공된 지표에 경보를 CloudWatch 생성하여 설정된 임계값 위반 시 알림을 받을 수 있습니다.
- Amazon CloudWatch Logs Insights — CloudWatch 로그 인사이트를 사용하면 Amazon Cognito CloudTrail 로그 파일을 모니터링하기 CloudWatch 위해 이벤트를 전송하도록 구성할 CloudTrail 수 있습니다.

주제

- 모니터링 비용
- 및 Service Quotas의 CloudWatch 할당량 및 사용량 추적
- 를 사용하여 Amazon Cognito API 호출을 로깅합니다. AWS CloudTrail

모니터링 비용

Amazon Cognito는 다음과 같은 사용량 차원에 대해 요금을 부과합니다.

- 사용자 풀 월간 활성 사용자 (MAU)
- OIDC 또는 SAML 페더레이션으로 로그인한 사용자 풀 MAU
- 고급 보안 기능을 갖춘 사용자 풀의 MAU
- 활성 사용자 풀 앱 클라이언트 및 클라이언트 자격 증명 부여를 통한 M2M (Machine to Machine) 인증을 위한 요청 볼륨
- 일부 범주의 사용자 풀 API에서 기본 할당량을 초과하여 구매한 사용량을 초과했습니다.

또한 이메일 메시지, SMS 메시지, Lambda 트리거와 같은 사용자 풀 기능으로 인해 종속 서비스에서 비용이 발생할 수 있습니다. 전체 개요는 Amazon Cognito 요금을 참조하십시오.

비용 보기 및 예상

AWS Billing and Cost Management 콘솔에서 AWS 비용을 보고 보고할 수 있습니다. 청구 및 결제 섹션에서 Amazon Cognito에 대한 가장 최근 요금을 확인할 수 있습니다. 청구서, 서비스별 요금에서 `Cognito` 필터링하여 사용량을 확인하세요. 자세한 내용은 AWS Billing 사용 설명서에서 결제 보기를 참조하세요.

API 요청 비율을 모니터링하려면 Service Quotas 콘솔에서 사용률 지표를 검토하세요. 예를 들어 클라이언트 자격 증명 요청은 요청 비율로 표시됩니다. ClientAuthentication 청구서에서 이러한 요청은 요청을 생성한 앱 클라이언트와 연결됩니다. 이 정보를 사용하면 멀티테넌트 아키텍처에서 테넌트에게 비용을 공평하게 할당할 수 있습니다.

일정 기간 동안의 M2M 요청 수를 파악하기 위해 Logs로 이벤트를AWS CloudTrail 전송하여 분석할 수도 있습니다. CloudWatch 클라이언트 자격 증명 CloudTrail 부여로 `Token_POST` 이벤트에서 이벤트를 쿼리하세요. 다음 CloudWatch Insights 쿼리는 이 수를 반환합니다.

```
filter eventName = "Token_POST" and @message like '"grant_type":["client_credentials"]'
 | stats count(*)
```

비용 관리

Amazon Cognito는 사용자 수, 기능 사용량 및 요청량을 기준으로 요금을 청구합니다. 다음은 Amazon Cognito에서 비용을 관리하기 위한 몇 가지 팁입니다.

비활성 사용자를 활성화하지 마세요.

사용자를 활성화시키는 일반적인 작업은 로그인, 가입, 암호 재설정입니다. 자세한 목록은 을 참조하십시오. 월별 활성 사용자) Amazon Cognito는 비활성 사용자를 청구서에 포함하지 않습니다. 사용자를 활성 상태로 설정하는 작업은 피하십시오. AdminGetUserAPI 작업 대신 작업을 사용하여 사용자를 쿼리하세요. ListUsers 비활성 사용자를 대상으로 사용자 풀 작업에 대한 대량의 관리 테스트를 수행하지 마세요.

페더레이션 사용자 연결

SAML 2.0 또는 OpenID Connect (OIDC) ID 공급자로 로그인하는 사용자는 로컬 사용자보다 비용이 많이 듭니다. 이러한 사용자를 로컬 사용자 프로필에 연결할 수 있습니다. 연결된 사용자는 페더레이션 사용자와 함께 제공되는 속성 및 액세스 권한을 사용하여 로컬 사용자로 로그인할 수 있습니다. IdPs 한 달 동안 연결된 로컬 계정으로만 로그인하는 SAML 또는 OIDC 사용자는 로컬 사용자로 요금이 청구됩니다.

요청 비율 관리

사용자 풀이 할당량 상한에 가까워지면 볼륨을 처리할 추가 용량 구매를 고려할 수 있습니다. 애플리케이션에서 요청의 양을 줄일 수 있을 수도 있습니다. 자세한 정보는 할당량 한도에 대한 요청 비율을 최적화하세요.을 참조하세요.

필요한 경우에만 새 토큰을 요청하세요.

클라이언트 자격 증명을 통한 M2M (Machine to Machine) 인증은 많은 양의 토큰 요청에 도달할 수 있습니다. 각각의 새 토큰 요청은 요청률 할당량 및 청구 규모에 영향을 미칩니다. 비용을 최적화하려면 애플리케이션 설계에 토큰 만료 설정 및 토큰 처리를 포함하세요.

- 애플리케이션에서 새 토큰을 요청할 때 이전에 발행한 토큰의 캐시된 버전을 받을 수 있도록 액세스 토큰을 캐시합니다. 이 방법을 구현하면 캐싱 프록시는 이전에 획득한 토큰의 만료를 알지 못한 채 액세스 토큰을 요청하는 애플리케이션을 차단하는 역할을 합니다. 캐싱 토큰은 Lambda 함수 및 Docker 컨테이너와 같은 수명이 짧은 마이크로서비스에 적합합니다.
- 토큰 만료를 고려한 토큰 처리 메커니즘을 애플리케이션에 구현하십시오. 이전 토큰이 만료되기 전까지는 새 토큰을 요청하지 마세요. 각 애플리케이션의 기밀성 및 가용성 요구 사항을 평가하고 적절

한 유효 기간의 액세스 토큰을 발급하도록 사용자 풀 앱 클라이언트를 구성하십시오. 사용자 지정 토큰 기간은 자격 증명 요청 빈도를 지속적으로 관리할 수 있는 수명이 긴 API와 서버에 가장 적합합니다.

사용하지 않는 클라이언트 자격 증명 (앱 클라이언트) 삭제

M2M 승인 청구는 토큰 요청 비율과 클라이언트 자격 증명을 부여하는 앱 클라이언트 수라는 두 가지 요소를 기반으로 합니다. M2M 인증을 위한 앱 클라이언트를 사용하지 않는 경우 해당 클라이언트를 삭제하거나 클라이언트 자격 증명을 발급할 수 있는 권한을 제거하세요. 앱 클라이언트 구성 관리에 대한 자세한 내용은 을 참조하십시오. 사용자 풀 앱 클라이언트

고급 보안 관리

사용자 풀에서 고급 보안 기능을 구성하면 사용자 풀의 모든 MAU에 고급 보안 청구 요금이 적용됩니다. 고급 보안 기능이 필요하지 않은 사용자가 있는 경우 사용자를 다른 사용자 풀로 분리하세요.

및 Service Quotas의 CloudWatch 할당량 및 사용량 추적

Amazon을 CloudWatch 사용하거나 Service Quotas를 사용하여 Amazon Cognito 사용자 풀을 모니터링할 수 있습니다. Service Quotas에서 자격 증명 풀 사용을 모니터링할 수도 있습니다. CloudWatch 원시 데이터를 수집하여 읽을 수 있는 거의 실시간 지표로 처리합니다. CloudWatch에서는 특정 임계값을 감시하고 해당 임계값이 충족되면 알림을 보내거나 조치를 취하는 경보를 설정할 수 있습니다. 서비스 할당량에 대한 CloudWatch 경보를 만들려면 경보 만들기를 참조하십시오. CloudWatch Amazon Cognito 지표는 5분 간격으로 제공됩니다. 보존 기간에 대한 자세한 내용은 Amazon CloudWatch FAQ 페이지를 참조하십시오. CloudWatch

Service Quotas를 사용하여 Amazon Cognito 사용자 풀 및 자격 증명 풀 할당량 사용량을 보고 관리할 수 있습니다. Service Quotas 콘솔에는 서비스 할당량 보기, 서비스 할당량 증가 요청, 현재 사용률 보기 등의 세 가지 기능이 있습니다. 첫 번째 기능을 사용하여 할당량을 보고 할당량 조정이 가능한지 확인할 수 있습니다. 두 번째 기능을 사용하여 Service Quotas 증가를 요청할 수 있습니다. 마지막 기능을 사용하여 할당량 사용률을 볼 수 있습니다. 이 기능은 계정이 일정 기간 동안 활성화되어 있었던 경우에만 사용할 수 있습니다. Service Quotas 콘솔에서 할당량을 확인하는 방법에 대한 자세한 내용은 Service Quotas 보기를 참조하세요.

> **Note**
>
> Amazon Cognito 지표는 5분 간격으로 제공됩니다. 보존 기간에 대한 자세한 내용은 Amazon CloudWatch FAQ 페이지를 참조하십시오. CloudWatch

CloudWatch교차 계정 Observability에서 모니터링 계정으로 설정된 계정에 로그인하면 AWS 계정 해당 모니터링 계정을 사용하여 서비스 할당량을 시각화하고 해당 모니터링 계정에 연결된 소스 계정의 지표에 대한 경보를 설정할 수 있습니다. 자세한 내용은 계정 간 옵저버빌리티를 참조하십시오. CloudWatch

주제

- Amazon Cognito 사용자 풀의 추가 활동 로깅
- Amazon Cognito 사용자 풀에 대한 지표
- Amazon Cognito 사용자 풀에 대한 차원
- Service Quotas 콘솔을 사용하여 지표 추적
- CloudWatch 콘솔을 사용하여 지표를 추적하세요.
- 할당량에 대한 CloudWatch 경보를 생성하세요.

Amazon Cognito 사용자 풀의 추가 활동 로깅

일부 추가 활동의 세부 로그를 로그 그룹에 보내도록 사용자 풀을 구성할 수 있습니다 CloudWatch . 이러한 로그는 의 로그보다 세분화되므로 사용자 풀 문제를 해결하는 데 유용할 수 있습니다. AWS CloudTrail이 기능을 활성화하면 Amazon Cognito에서 로그를 전송할 로그 그룹을 선택할 수 있습니다. 사용자 활동 로깅은 사용자 풀이 Amazon SNS 및 Amazon SES를 통해 전송하는 이메일 및 SMS 메시지의 상태를 확인하려는 경우에 유용합니다.

현재는 사용자 풀에서 오류 수준의 사용자 알림 로그만 전송할 수 있습니다.

세부 로깅은 사용자 풀의 다음 로그 기능을 대체하거나 변경하지 않습니다.

1. CloudTrail 가입 및 로그인과 같은 일상적인 사용자 활동 로그.
2. 메트릭을 통한 CloudWatch 대규모 사용자 활동 분석.

이와 별도로 Logs에서 사용자 가져오기 작업 및 Lambda 트리거에서 로그를 찾을 수도 있습니다. CloudWatch Amazon Cognito와 Lambda는 세부 활동 로그에 지정한 로그 그룹과는 다른 로그 그룹에 이러한 로그를 저장합니다.

API 요청에서 Amazon Cognito 사용자 풀 API를 사용하여 세부 활동 로그를 구성할 수 있습니다 SetLogDeliveryConfiguration. GetLogDeliveryConfigurationAPI 요청에서 사용자 풀의 로깅 구성을 볼 수 있습니다.

다음 권한이 있는 AWS 자격 증명으로 이러한 요청을 승인해야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
            {
            "Sid": "ManageUserPoolLogs",
            "Action": [
                "cognito-idp:SetLogDeliveryConfiguration",
                "cognito-idp:GetLogDeliveryConfiguration",
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        },
        {
            "Sid": "CognitoLog",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        },
        {
            "Sid": "CognitoLoggingCWL",
            "Action": [
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

다음은 사용자 풀의 이벤트 예시입니다. 이 로그 스키마는 변경될 수 있습니다. 일부 필드는 null 값으로 로깅될 수 있습니다.

```
{
    "eventTimestamp": "1687297330677",
    "eventSource": "USER_NOTIFICATION",
    "logLevel": "ERROR",
    "message": {
        "details": "String"
    },
    "logSourceId": {
        "userPoolId": "String"
    }
}
```

Amazon Cognito에서 로그를 전송하는 것은 최선의 작업을 기반으로 합니다. 사용자 풀이 제공하는 로그의 양과 로그의 서비스 할당량은 CloudWatch 로그 전달에 영향을 줄 수 있습니다.

CloudWatch 로그 전송이 활성화된 경우 로그 요금이 적용됩니다. 자세한 내용은 Amazon CloudWatch 요금의 벤드 로그를 참조하십시오.

리소스 정책 크기가 5,120자보다 큰 로그 그룹에 로그를 전송하려면 경로가 `/aws/vendedlogs`로 시작하는 로그 그룹을 구성합니다. 자세한 내용은 특정 AWS 서비스에서 로깅 활성화를 참조하십시오.

Amazon Cognito 사용자 풀에 대한 지표

다음 표에는 Amazon Cognito 사용자 풀에 대해 제공되는 지표가 나열되어 있습니다. Amazon Cognito의 Amazon CloudWatch 메트릭 네임스페이스는 `AWS/Cognito`입니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 네임스페이스를 참조하십시오.

> **Note**
>
> 지난 2주 동안 새로운 데이터 포인트가 없는 지표는 콘솔에 나타나지 않습니다. 콘솔의 [모든 지표(All metrics)] 탭에서 검색 상자에 지표 이름이나 차원 이름을 입력할 때도 표시되지 않습니다. 또한 list-metrics 명령의 결과에서 반환되지 않습니다. 이러한 메트릭을 검색하는 가장 좋은 방법은 AWS CLI에서 `get-metric-data` 또는 `get-metric-statistics` 명령을 사용하는 것입니다.

지표	설명
SignUpSuccesses	Amazon Cognito 사용자 풀에 완료된 사용자 등록 요청의 총 개수를 제공합니다. 성공적인 사용자 등록 요청은 1의 값을 생성하지만 실패한 요청은 0의 값을 생성합니다. 제한된 요청은 실패한 요청으로 간주되므로 제한된 요청은 0의 카운트를 생성합니다. 성공한 사용자 등록 요청의 백분율을 찾으려면 이 메트릭에 대한 Average 통계를 사용합니다. 총 사용자 등록 요청 수를 계산하려면 이 메트릭에 대한 Sample Count 통계를 사용합니다. 성공한 사용자 등록 요청의 총 수를 계산하려면 이 메트릭에 대한 Sum 통계를 사용합니다. 실패한 사용자 등록 요청의 총 수를 계산하려면 CloudWatch Math 식을 사용하고 Sum 통계에서 Sample Count 통계를 빼십시오. 이 지표는 각 사용자 풀 클라이언트마다 사용자 풀별로 게시됩니다. 관리자가 사용자 등록을 수행하는 경우 지표는 사용자 풀 클라이언트가 Admin인 상태로 게시됩니다. 사용자 가져오기 및 사용자 마이그레이션의 경우 이 지표를 내보내지 않습니다. 메트릭 차원: UserPool, UserPoolClient 단위: 개
SignUpThrottles	Amazon Cognito 사용자 풀에 대한 제한된 사용자 등록 요청의 총 수를 제공합니다. 사용자 등록 요청이 제한될 때마다 1개가 게시됩니다. 제한된 사용자 등록 요청의 총 수를 계산하려면 이 메트릭에 대한 Sum 통계를 사용합니다.

지표	설명
	이 지표는 각 클라이언트마다 사용자 풀별로 게시됩니다. 관리자가 제한된 요청을 하는 경우 지표는 사용자 풀 클라이언트가 `Admin`인 상태로 게시됩니다. 메트릭 차원: `UserPool`, `UserPoolClient` 단위: 개

지표	설명
SignInSuccesses	Amazon Cognito 사용자 풀에 성공한 사용자 인증 요청의 총 수를 제공합니다. 인증 토큰이 사용자에게 발급되면 사용자 인증이 성공한 것으로 간주됩니다. 성공한 인증은 1의 값을 생성하지만 실패한 요청은 0의 값을 생성합니다. 제한된 요청은 실패한 요청으로 간주되므로 제한된 요청은 0의 카운트를 생성합니다. 성공한 사용자 인증 요청의 백분율을 찾으려면 이 메트릭에 대한 Average 통계를 사용합니다. 총 사용자 인증 요청 수를 계산하려면 이 메트릭에 대한 Sample Count 통계를 사용합니다. 성공한 사용자 인증 요청의 총 수를 계산하려면 이 메트릭에 대한 Sum 통계를 사용합니다. 실패한 사용자 인증 요청의 총 수를 계산하려면 CloudWatch Math 식을 사용하고 Sum 통계에서 Sample Count 통계를 빼십시오. 이 지표는 각 클라이언트마다 사용자 풀별로 게시됩니다. 잘못된 사용자 풀 클라이언트에 요청이 제공되는 경우 지표의 해당 사용자 풀 클라이언트 값에는 요청에 전송된 실제 유효하지 않은 값 대신 고정 값 Invalid가 포함됩니다. Amazon Cognito 토큰 새로 고침 요청은 이 지표에 포함되지 않습니다. Refresh 토큰 통계를 제공하는 별도의 지표가 있습니다. 메트릭 차원: UserPool, UserPoolClient 단위: 개

지표	설명
SignInThrottles	Amazon Cognito 사용자 풀에서 이루어진 제한된 사용자 인증 요청의 총 수를 제공합니다. 인증 요청이 제한될 때마다 1개가 게시됩니다. 제한된 사용자 인증 요청의 총 수를 계산하려면 이 메트릭에 대한 Sum 통계를 사용합니다. 이 지표는 각 클라이언트마다 사용자 풀별로 게시됩니다. 잘못된 사용자 풀 클라이언트에 요청이 제공되는 경우 지표의 해당 사용자 풀 클라이언트 값에는 요청에 전송된 실제 유효하지 않은 값 대신 고정 값 Invalid가 포함됩니다. Amazon Cognito 토큰 새로 고침 요청은 이 지표에 포함되지 않습니다. Refresh 토큰 통계를 제공하는 별도의 지표가 있습니다. 메트릭 차원: UserPool, UserPoolClient 단위: 개

지표	설명
TokenRefreshSuccesses	Amazon Cognito 사용자 풀에 대해 만들어진 Amazon Cognito 토큰을 새로 고치려면 성공한 총 요청 수를 제공합니다. 새로 고침에 성공한 Amazon Cognito 토큰 요청은 값 1을 생성하지만 실패한 요청은 값 0을 생성합니다. 제한된 요청은 실패한 요청으로 간주되므로 제한된 요청은 0의 카운트를 생성합니다. Amazon Cognito 토큰을 새로 고치기 위해 성공한 요청의 백분율을 구하려면 이 지표의 Average 통계를 사용합니다. Amazon Cognito 토큰을 새로 고치기 위한 총 요청 수를 계산하려면 이 지표의 Sample Count 통계를 사용합니다. Amazon Cognito 토큰을 새로 고치기 위해 성공한 총 요청 수를 계산하려면 이 지표의 Sum 통계를 사용합니다. Amazon Cognito 토큰 새로 고침에 실패한 총 요청 수를 계산하려면 CloudWatch Math 식을 사용하고 Sum 통계에서 통계를 빼십시오. Sample Count 이 지표는 각 사용자 풀 클라이언트별로 게시됩니다. 요청에 잘못된 사용자 풀 클라이언트가 있는 경우 사용자 풀 클라이언트 값에 Invalid라는 고정 값이 포함됩니다. 메트릭 차원: UserPool, UserPoolClient 단위: 개

지표	설명
TokenRefreshThrottles	Amazon Cognito 사용자 풀에 대해 만들어진 Amazon Cognito 토큰을 새로 고치려면 제한된 총 요청 수를 제공합니다. 새로 고침 Amazon Cognito 토큰 요청이 제한될 때마다 1개가 게시됩니다. Amazon Cognito 토큰을 새로 고치기 위한 제한된 총 요청 수를 계산하려면 이 지표의 Sum 통계를 사용합니다. 이 지표는 각 클라이언트마다 사용자 풀별로 게시됩니다. 잘못된 사용자 풀 클라이언트에 요청이 제공되는 경우 지표의 해당 사용자 풀 클라이언트 값에는 요청에 전송된 실제 유효하지 않은 값 대신 고정 값 Invalid가 포함됩니다. 메트릭 차원: UserPool, UserPoolClient 단위: 개

지표	설명
FederationSuccesses	Amazon Cognito 사용자 풀에 성공한 ID 페더레이션 요청의 총 수를 제공합니다. Amazon Cognito가 사용자에게 인증 토큰을 발급하면 아이덴티티 페더레이션이 성공한 것으로 간주됩니다. 자격 증명 연동 요청이 성공하면 값 1이 생성되고 요청이 실패하면 값이 0입니다. 인증 코드는 생성하지만 토큰은 생성하지 않는 요청과 제한된 요청은 값이 0입니다. 성공한 자격 증명 연동 요청의 비율을 찾으려면 이 메트릭의 Average 통계를 사용합니다. 총 자격 증명 연동 요청 수를 계산하려면 이 메트릭의 Sample Count 통계를 사용합니다. 성공한 자격 증명 연동 요청의 총 수를 계산하려면 이 메트릭의 Sum 통계를 사용합니다. 실패한 ID 페더레이션 요청의 총 수를 계산하려면 CloudWatch Math 식을 사용하고 통계에서 통계를 빼십시오. Sum Sample Count 메트릭 차원: UserPool, UserPoolClient , IdentityProvider 단위: 개
FederationThrottles	Amazon Cognito 사용자 풀에 대한 제한된 ID 페더레이션 요청의 총 수를 제공합니다. ID 페더레이션 요청이 제한될 때마다 1개가 게시됩니다. 제한된 자격 증명 연동 요청의 총 수를 계산하려면 이 메트릭에 대한 Sum 통계를 사용합니다. 메트릭 차원: UserPool, UserPoolClient , IdentityProvider 단위: 개

지표	설명
CallCount	범주와 관련된 고객의 총 호출 수를 제공합니다. 이 지표에는 제한된 호출, 실패한 호출, 성공한 호출 등 모든 호출이 포함됩니다. 이 지표는 Usage nameSpace 에서 사용할 수 있습니다. 범주 할당량은 AWS 계정 및 지역의 모든 사용자 풀에서 각 계정에 적용됩니다. 이 지표의 Sum 통계를 사용하여 범주의 총 호출 수를 계산할 수 있습니다. 지표 차원: 서비스, 유형, 리소스, 클래스 단위: 개
ThrottleCount	범주와 관련된 총 제한된 호출 수를 제공합니다. 이 지표는 Usage nameSpace 에서 사용할 수 있습니다. 이 지표는 계정 수준에서 게시됩니다. 이 지표의 Sum 통계를 사용하여 범주의 총 호출 수를 계산할 수 있습니다. 지표 차원: 서비스, 유형, 리소스, 클래스 단위: 개

Amazon Cognito 사용자 풀에 대한 차원

다음 차원은 Amazon Cognito에 의해 게시되는 사용량 지표를 구체화하는 데 사용됩니다. 차원은 CallCount 및 ThrottleCount 지표에만 적용됩니다.

측정기준	설명
Service	리소스가 포함된 AWS 서비스의 이름. Amazon Cognito 사용량 지표의 경우 이 차원 값은 `Cognito user pool`입니다.
유형	보고되는 엔터티의 유형입니다. Amazon Cognito 사용량 지표에 대한 유일한 유효 값은 API입니다.
Resource	실행 중인 리소스의 유형입니다. 유일한 유효 값은 범주 이름입니다.
Class	추적 중인 리소스의 클래스입니다. Amazon Cognito는 클래스 차원을 사용하지 않습니다.

Service Quotas 콘솔을 사용하여 지표 추적

Service Quotas로 중앙 위치에서 Amazon Cognito 사용자 풀 및 자격 증명 풀 할당량을 보고 관리할 수 있습니다. Service Quotas 콘솔을 사용하여 특정 할당량에 대한 세부 정보를 확인하고, 할당량 사용률을 모니터링하고, 할당량 증가를 요청할 수 있습니다. 일부 할당량 유형의 경우 CloudWatch 경보를 생성하여 할당량 사용률을 추적할 수 있습니다. 추적할 수 있는 Amazon Cognito 지표에 대한 자세한 내용은 할당량 사용량 추적을 참조하세요.

Amazon Cognito 사용자 풀 및 자격 증명 풀 서비스 할당량 사용률을 보려면 다음 단계를 수행합니다.

1. Service Quotas 콘솔을 엽니다.
2. 탐색 창에서 AWS 서비스를 선택합니다.
3. AWS 서비스 목록에서 Amazon Cognito 사용자 풀 또는 Amazon Cognito 페더레이션 ID를 검색하고 선택합니다. 서비스 할당량 페이지가 표시됩니다.
4. CloudWatch 모니터링을 지원하는 할당량을 선택합니다. 예를 들어, Amazon Cognito 사용자 풀에서 `Rate of UserAuthentication requests`를 선택합니다.
5. 아래로 스크롤하여 [모니터링(Monitoring)]으로 이동합니다. 이 섹션은 모니터링을 지원하는 CloudWatch 할당량에만 표시됩니다.
6. [모니터링(Monitoring)에서 현재 서비스 할당량 사용률을 그래프로 볼 수 있습니다.
7. [모니터링(Monitoring)]에서 1시간, 3시간, 12시간, 1일, 3일 또는 1주를 선택합니다.

8. 선택하여 서비스 할당량 사용률을 보려면 그래프 내의 아무 영역이나 선택합니다. 여기에서 대시보드에 그래프를 추가하거나 작업 메뉴를 사용하여 지표에서 보기를 선택할 수 있습니다. 그러면 콘솔의 관련 지표로 이동합니다. CloudWatch

CloudWatch 콘솔을 사용하여 지표를 추적하세요.

를 사용하여 Amazon Cognito 사용자 풀 지표를 추적하고 수집할 수 있습니다. CloudWatch CloudWatch 대시보드에는 사용하는 모든 AWS 서비스에 대한 지표가 표시됩니다. 를 CloudWatch 사용하여 지표 경보를 만들 수 있습니다. 알림을 보내거나 모니터링 중인 특정 리소스를 변경하도록 경보를 설정할 수 있습니다. 에서 CloudWatch 서비스 할당량 지표를 보려면 다음 단계를 완료하세요.

1. CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표(Metrics)를 선택합니다.
3. [모든 지표(All metrics)]에서 지표와 차원을 선택합니다.
4. 지표 옆의 확인란을 선택합니다. 지표가 그래프에 표시됩니다.

> **Note**
>
> 지난 2주 동안 새로운 데이터 포인트가 없는 지표는 콘솔에 나타나지 않습니다. 콘솔의 모든 지표 탭에 있는 검색 상자에 지표 이름이나 차원 이름을 입력할 때도 나타나지 않으며 list-metrics 명령의 결과에도 반환되지 않습니다. 이러한 지표를 검색하는 가장 좋은 방법은 AWS CLI에서 `get-metric-data` 또는 `get-metric-statistics` 명령을 사용하는 것입니다.

할당량에 대한 CloudWatch 경보를 생성하세요.

Amazon Cognito는 CloudWatch 및 API의 AWS 서비스 할당량에 해당하는 사용량 지표를 제공합니다. `CallCount` `ThrottleCount` 의 사용량 추적에 대한 자세한 내용은 을 참조하십시오. CloudWatch 할당량 사용량 추적

Service Quotas 콘솔에서 사용량이 서비스 할당량에 가까워지면 알리는 경보를 생성할 수 있습니다. Service Quotas 콘솔을 사용하여 CloudWatch 경보를 설정하는 방법을 알아보려면 Service Quotas 및 경보를 참조하십시오. CloudWatch

를 사용하여 Amazon Cognito API 호출을 로깅합니다. AWS CloudTrail

Amazon Cognito는 Amazon Cognito에서 사용자 AWS CloudTrail, 역할 또는 서비스가 수행한 작업의 기록을 제공하는 AWS 서비스와 통합되어 있습니다. CloudTrail Amazon Cognito에 대한 API 호출의 하위 집합을 이벤트로 캡처합니다. 여기에는 Amazon Cognito 콘솔에서의 호출 및 Amazon Cognito API 작업에 대한 코드 호출이 포함됩니다. 트레일을 생성하는 경우 Amazon Cognito용 CloudTrail 이벤트를 포함하여 Amazon S3 버킷으로 이벤트를 전송하도록 선택할 수 있습니다. 트레일을 구성하지 않아도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 계속 볼 수 있습니다. 에서 수집한 CloudTrail 정보를 사용하여 Amazon Cognito에 이루어진 요청, 요청이 이루어진 IP 주소, 요청한 사람, 요청 시기 및 추가 세부 정보를 확인할 수 있습니다.

구성 및 활성화 방법을 CloudTrail 포함하여 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하십시오.

특정 CloudTrail 이벤트에 대한 Amazon CloudWatch 경보를 생성할 수도 있습니다. 예를 들어, 자격 증명 풀 구성이 변경될 경우 경보가 CloudWatch 트리거되도록 설정할 수 있습니다. 자세한 내용은 CloudTrail 이벤트에 대한 CloudWatch 경보 생성: 예제를 참조하십시오.

주제

- 아마존 코그니토 정보 CloudTrail
- Amazon Cognito 로그인 이벤트 이해
- 아마존 로그 인사이트를 사용하여 Amazon Cognito CloudTrail 이벤트 분석하기 CloudWatch

아마존 코그니토 정보 CloudTrail

CloudTrail 를 생성하면 AWS 계정활성화됩니다. Amazon Cognito에서 지원되는 이벤트 활동이 발생하면 해당 활동이 이벤트 기록의 다른 AWS 서비스 CloudTrail 이벤트와 함께 이벤트에 기록됩니다. AWS 계정에서 최근 이벤트를 보고, 검색하고, 다운로드할 수 있습니다. 자세한 내용은 이벤트 기록으로 CloudTrail 이벤트 보기를 참조하십시오.

Amazon Cognito의 이벤트를 포함하여 AWS 계정의 지속적인 이벤트 기록을 보려면 트레일을 생성하십시오. CloudTrail 트레일은 Amazon S3 버킷으로 로그 파일을 전송합니다. 콘솔에서 추적을 생성하면 기본적으로 모든 지역에 추적이 적용됩니다. 트레일은 AWS 파티션에 있는 모든 지역의 이벤트를 기록하고 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 또한 CloudTrail 로그에서 수집된 이벤트 데이터를 추가로 분석하고 이에 따라 조치를 취하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음을 참조하세요.

- 추적 생성 개요

- CloudTrail 지원되는 서비스 및 통합
- 다음에 대한 아마존 SNS 알림 구성 CloudTrail
- 여러 지역에서 CloudTrail 로그 파일 수신 및 여러 계정으로부터 CloudTrail 로그 파일 수신

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. ID 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 아니면 IAM 사용자 보안 인증 정보로 했는지 여부.
- 역할 또는 페더레이션 사용자의 임시 보안 인증을 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청했는지 여부.

자세한 내용은 CloudTrail UserIdentity 요소를 참조하십시오.

의 기밀 데이터 AWS CloudTrail

사용자 풀과 자격 증명 풀은 사용자 데이터를 처리하기 때문에 Amazon Cognito는 값이 있는 이벤트의 일부 프라이빗 필드를 숨깁니다. CloudTrail `HIDDEN_FOR_SECURITY_REASONS` Amazon Cognito가 이벤트에 채우지 않는 필드의 예는 Amazon Cognito 로그인 이벤트 이해 섹션을 참조하세요. Amazon Cognito는 암호 및 토큰과 같이 일반적으로 사용자 정보를 포함하는 일부 필드만 가립니다. Amazon Cognito는 API 요청의 비공개 필드가 아닌 필드에 입력하는 개인 식별 정보를 자동으로 탐지하거나 마스킹하지 않습니다.

Amazon Cognito 사용자 풀

Amazon Cognito는 사용자 풀 작업 페이지에 나열된 모든 작업을 CloudTrail 로그 파일의 이벤트로 기록할 수 있도록 지원합니다. Amazon Cognito는 사용자 풀 이벤트를 관리 이벤트로 기록합니다 CloudTrail .

Amazon Cognito 사용자 풀 CloudTrail 항목의 `eventType` 필드는 앱이 Amazon Cognito 사용자 풀 API에 요청했는지, 아니면 OpenID Connect, SAML 2.0 또는 호스팅된 UI용 리소스를 제공하는 엔드포인트에 요청했는지를 알려줍니다. API 요청의 `eventType`은 `AwsApiCall`이고 엔드포인트 요청의 `eventType`은 `AwsServiceEvent`입니다.

Amazon Cognito는 다음과 같은 호스팅된 UI 요청을 호스팅된 UI에 이벤트로 기록합니다. CloudTrail

의 호스팅된 UI 작업 CloudTrail

Operation	설명
Login_GET , CognitoAuthentication	사용자가 보안 인증을 보거나 Login 엔드포인트에 제출합니다.
OAuth2_Authorize_GET , Beta_Authorize_GET	사용자가 권한 부여 엔드포인트를 봅니다.
OAuth2Response_GET , OAuth2Response_POST	사용자가 /oauth2/idpresponse 엔드포인트에 IdP 토큰을 제출합니다.
SAML2Response_POST , Beta_SAML2Response_POST	사용자가 /saml2/idpresponse 엔드포인트에 IdP SAML 어설션을 제출합니다.
Login_OIDC_SAML_POST	사용자가 Login 엔드포인트에 사용자 이름을 입력하고 IdP 식별자와 일치합니다.
Token_POST , Beta_Token_POST	사용자가 Token 엔드포인트에 인증 코드를 제출합니다.
Signup_GET , Signup_POST	사용자가 /signup 엔드포인트에 가입 정보를 제출합니다.
Confirm_GET , Confirm_POST	사용자가 호스팅 UI에서 확인 코드를 제출합니다.
ResendCode_POST	사용자가 호스팅 UI에서 확인 코드를 다시 보내라는 요청을 제출합니다.
ForgotPassword_GET , ForgotPassword_POST	사용자가 /forgotPassword 엔드포인트에 암호를 재설정하라는 요청을 제출합니다.
ConfirmForgotPassword_GET , ConfirmForgotPassword_POST	사용자가 /confirmForgotPassword 엔드포인트에 ForgotPassword 요청을 확인하는 코드를 제출합니다.
ResetPassword_GET , ResetPassword_POST	사용자가 호스팅 UI에서 새 암호를 제출합니다.

Operation	설명
`Mfa_GET`, `Mfa_POST`	사용자가 호스팅 UI에서 다중 인증(MFA) 코드를 제출합니다.
`MfaOption_GET` , `MfaOption_POST`	사용자가 호스팅 UI에서 MFA에 대해 선호하는 방법을 선택합니다.
`MfaRegister_GET` , `MfaRegister_POST`	MFA 등록 시 사용자가 호스팅된 UI에서 다중 인증(MFA) 코드를 제출합니다.
`Logout`	사용자가 `/logout` 엔드포인트에서 로그아웃합니다.
`SAML2Logout_POST`	사용자가 `/saml2/logout` 엔드포인트에서 로그아웃합니다.
`Error_GET`	사용자가 호스팅 UI에서 오류 페이지를 봅니다.
`UserInfo_GET` , `UserInfo_POST`	사용자 또는 IdP가 UserInfo 엔드포인트와 정보를 교환합니다.
`Confirm_With_Link_GET`	사용자가 Amazon Cognito가 이메일 메시지로 보낸 링크를 기반으로 확인을 제출합니다.
`Event_Feedback_GET`	사용자가 Amazon Cognito에 고급 보안 기능 이벤트에 대한 피드백을 제출합니다.

> **Note**
>
> Amazon Cognito는 특정 사용자에 대한 요청을 `UserSub` 기록하지만 CloudTrail 로그에는 기록하지 않습니다`UserName`. `ListUsers` API를 호출하고 하위 집합을 표시하는 필터를 사용하여 특정 `UserSub`의 사용자를 찾을 수 있습니다.

Amazon Cognito 자격 증명 풀

데이터 이벤트

Amazon Cognito는 다음과 같은 Amazon Cognito 아이덴티티 이벤트를 데이터 이벤트로 CloudTrail 기록합니다. 데이터 이벤트는 기본적으로 CloudTrail 기록되지 않는 대용량 데이터 플레인 API 작업입니다. 데이터 이벤트에는 추가 요금이 적용됩니다.

- GetCredentialsForIdentity
- GetId
- GetOpenIdToken
- GetOpenIdTokenForDeveloperIdentity
- UnlinkIdentity

이러한 API 작업에 대한 CloudTrail 로그를 생성하려면 트레일에서 데이터 이벤트를 활성화하고 Cognito 자격 증명 풀의 이벤트 선택기를 선택해야 합니다. 자세한 내용은 AWS CloudTrail 사용 설명서의 추적을 위해 데이터 이벤트 로깅을 참조하십시오.

다음 CLI 명령을 사용해서도 추적에 자격 증명 풀 이벤트 선택기를 추가할 수 있습니다.

```
aws cloudtrail put-event-selectors --trail-name <trail name> --advanced-event-selectors
 \
"{\
   \"Name\": \"Cognito Selector\",\
   \"FieldSelectors\": [\
      {\
         \"Field\": \"eventCategory\",\
         \"Equals\": [\
            \"Data\"\
         ]\
      },\
      {\
         \"Field\": \"resources.type\",\
         \"Equals\": [\
            \"AWS::Cognito::IdentityPool\"\
         ]\
      }\
   ]\
}"
```

관리 이벤트

Amazon Cognito는 나머지 Amazon Cognito 자격 증명 풀 API 작업을 관리 이벤트로 기록합니다. CloudTrail 기본적으로 관리 이벤트 API 작업을 기록합니다.

Amazon Cognito가 CloudTrail 로깅하는 Amazon Cognito 자격 증명 풀 API 작업의 목록은 Amazon Cognito 자격 증명 풀 API 참조를 참조하십시오.

Amazon Cognito Sync

Amazon Cognito는 모든 Amazon Cognito Sync API 작업을 관리 이벤트로 로깅합니다. Amazon Cognito가 CloudTrail 로깅하는 Amazon Cognito Sync API 작업 목록은 Amazon Cognito Sync API 레퍼런스를 참조하십시오.

Amazon Cognito 로그인 이벤트 이해

트레일은 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 전송할 수 있습니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함되어 있습니다. 이벤트는 모든 소스의 단일 요청을 나타내며 요청된 작업, 작업 날짜 및 시간, 요청 매개 변수 등에 대한 정보를 포함합니다. CloudTrail 로그 파일은 공개 API 호출의 정렬된 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

주제

- 호스팅된 UI 가입을 위한 예제 CloudTrail 이벤트
- CloudTrail SAML 요청의 예제 이벤트
- 토큰 CloudTrail 엔드포인트에 대한 요청의 예제 이벤트
- 에 대한 예제 CloudTrail 이벤트 CreateIdentityPool
- 에 대한 예제 CloudTrail 이벤트 GetCredentialsForIdentity
- 에 대한 예제 CloudTrail 이벤트 GetId
- 에 대한 예제 CloudTrail 이벤트 GetOpenIdToken
- 에 대한 예제 CloudTrail 이벤트 GetOpenIdTokenForDeveloperIdentity
- 에 대한 예제 CloudTrail 이벤트 UnlinkIdentity

호스팅된 UI 가입을 위한 예제 CloudTrail 이벤트

다음 예제 CloudTrail 이벤트는 사용자가 호스팅된 UI를 통해 가입할 때 Amazon Cognito가 기록하는 정보를 보여줍니다.

Amazon Cognito는 신규 사용자가 앱의 로그인 페이지로 이동할 때 다음 이벤트를 로그합니다.

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
```

```
        "accountId": "123456789012"
    },
    "eventTime": "2022-04-06T05:38:12Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Login_GET",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "errorCode": "",
    "errorMessage": "",
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200.0
        },
        "requestParameters":
        {
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "response_type":
            [
                "token"
            ],
            "client_id":
            [
                "1example23456789"
            ]
        }
    },
    "eventID": "382ae09a-151d-4116-8f2b-6ac0a804a38c",
    "readOnly": true,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

Amazon Cognito는 신규 사용자가 앱의 로그인 페이지에서 가입을 선택할 때 다음 이벤트를 로그합니다.


```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-05T23:21:43Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Signup_GET",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
            "response_type":
            [
                "code"
            ],
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "client_id":
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "7a63e7c2-b057-4f3d-a171-9d9113264fff",
    "eventID": "5e7b27a0-6870-4226-adb4-f86cd51ac5d8",
```

```
    "readOnly": true,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

Amazon Cognito는 신규 사용자가 사용자 이름을 선택하고 이메일 주소를 입력한 다음 앱의 로그인 페이지에서 암호를 선택할 때 다음 이벤트를 로깅합니다. Amazon Cognito는 사용자 자격 증명에 대한 식별 정보를 기록하지 않습니다. CloudTrail

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-05T23:22:05Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Signup_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 302
        },
        "requestParameters":
        {
            "password":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "requiredAttributes[email]":
            [
```

```
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "response_type":
            [
                "code"
            ],
            "_csrf":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "client_id":
            [
                "1example23456789"
            ],
            "username":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "9ad58dd8-3517-4aa8-96a5-d17a01df9eb4",
    "eventID": "c75eb7a5-eb8c-43d1-8331-f4412e756e69",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

Amazon Cognito는 신규 사용자가 가입한 후 호스팅 UI의 사용자 확인 페이지에 액세스할 때 다음 이벤트를 로그합니다.

```
{
```

```
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-05T23:22:06Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Confirm_GET",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
            "response_type":
            [
                "code"
            ],
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "client_id":
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "58a5b170-3127-45bb-88cc-3e652d779e0b",
    "eventID": "7f87291a-6d50-409a-822f-e3a5ec7e60da",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
```

```
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

Amazon Cognito는 호스팅 UI의 사용자 확인 페이지에서 사용자가 Amazon Cognito가 이메일 메시지로 보낸 코드를 입력할 때 다음 이벤트를 로그합니다.

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-05T23:23:32Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Confirm_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 302
        },
        "requestParameters":
        {
            "confirm":
            [
                ""
            ],
            "deliveryMedium":
            [
                "EMAIL"
            ],
            "sub":
            [
                "704b1e47-34fe-40e9-8c41-504997494531"
            ],
```

```
            "code":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "destination":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "response_type":
            [
                "code"
            ],
            "_csrf":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "cognitoAsfData":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "client_id":
            [
                "1example23456789"
            ],
            "username":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "9764300a-ed35-4f87-8a0f-b18b3fe2b11e",
    "eventID": "e24ac6e5-2f70-4c6e-ad4e-2f08a547bb36",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
```

```
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

CloudTrail SAML 요청의 예제 이벤트

Amazon Cognito는 SAML IdP로 인증된 사용자가 `/saml2/idpresponse` 엔드포인트에 SAML 어설션을 제출할 때 다음 이벤트를 로그합니다.

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-06T00:50:57Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "SAML2Response_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 302
        },
        "requestParameters":
        {
            "RelayState":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "SAMLResponse":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
```

```
    },
    "requestID": "4f6f15d1-c370-4a57-87f0-aac4817803f7",
    "eventID": "9824b50f-d9d1-4fb8-a2c1-6aa78ca5902a",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "625647942648",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

토큰 CloudTrail 엔드포인트에 대한 요청의 예제 이벤트

다음은 Token 엔드포인트에 대한 요청의 이벤트 예제입니다.

Amazon Cognito는 인증되고 인증 코드를 받은 사용자가 /oauth2/token 엔드포인트에 코드를 제출할 때 다음 이벤트를 로그합니다.

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-12T22:12:30Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Token_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
```

```
            "code":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "grant_type":
            [
                "authorization_code"
            ],
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "client_id":
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "f257f752-cc14-4c52-ad5b-152a46915238",
    "eventID": "0bd1586d-cd3e-4d7a-abaf-fd8bfc3912fd",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

Amazon Cognito는 백엔드 시스템이 `/oauth2/token` 엔드포인트에 액세스 토큰에 대한 `client_credentials` 요청을 제출할 때 다음 이벤트를 로깅합니다.

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-12T21:07:05Z",
```

```
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Token_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
            "grant_type":
            [
                "client_credentials"
            ],
            "client_id":
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "4f871256-6825-488a-871b-c2d9f55caff2",
    "eventID": "473e5cbc-a5b3-4578-9ad6-3dfdcb8a6d34",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

Amazon Cognito는 앱이 `/oauth2/token` 엔드포인트와 새 ID에 대한 새로 고침 토큰 및 액세스 토큰을 교환할 때 다음 이벤트를 로그합니다.

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-12T22:16:40Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Token_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
            "refresh_token":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "grant_type":
            [
                "refresh_token"
            ],
            "client_id":
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "2829f0c6-a3a9-4584-b046-11756dfe8a81",
    "eventID": "12bd3464-59c7-44fa-b8ff-67e1cf092018",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
```

```
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

에 대한 예제 CloudTrail 이벤트 CreateIdentityPool

다음 예제는 `CreateIdentityPool` 작업의 요청에 대한 로그 항목입니다. Alice라는 IAM 사용자가 한 요청입니다.

```
{
    "eventVersion": "1.03",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/Alice",
        "accountId": "123456789012",
        "accessKeyId": "['EXAMPLE_KEY_ID']",
        "userName": "Alice"
    },
    "eventTime": "2016-01-07T02:04:30Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "CreateIdentityPool",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "USER_AGENT",
    "requestParameters": {
        "identityPoolName": "TestPool",
        "allowUnauthenticatedIdentities": true,
        "supportedLoginProviders": {
            "graph.facebook.com": "000000000000000"
        }
    },
    "responseElements": {
        "identityPoolName": "TestPool",
        "identityPoolId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE",
        "allowUnauthenticatedIdentities": true,
        "supportedLoginProviders": {
            "graph.facebook.com": "000000000000000"
        }
```

```
    },
    "requestID": "15cc73a1-0780-460c-91e8-e12ef034e116",
    "eventID": "f1d47f93-c708-495b-bff1-cb935a6064b2",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

에 대한 예제 CloudTrail 이벤트 GetCredentialsForIdentity

다음 예제는 `GetCredentialsForIdentity` 작업의 요청에 대한 로그 항목입니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown"
    },
    "eventTime": "2023-01-19T16:55:08Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "GetCredentialsForIdentity",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.4",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off
 command/cognito-identity.get-credentials-for-identity",
    "requestParameters": {
        "logins": {
            "cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa":
 "HIDDEN_DUE_TO_SECURITY_REASONS"
        },
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
    },
    "responseElements": {
        "credentials": {
            "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
            "sessionToken": "aAaAaAaAaAaAab1111111111EXAMPLE",
            "expiration": "Jan 19, 2023 5:55:08 PM"
        },
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
    },
    "requestID": "659dfc23-7c4e-4e7c-858a-1abce884d645",
    "eventID": "6ad1c766-5a41-4b28-b5ca-e223ccb00f0d",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
```

```
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
}
```

에 대한 예제 CloudTrail 이벤트 GetId

다음 예제는 `GetId` 작업의 요청에 대한 로그 항목입니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown"
    },
    "eventTime": "2023-01-19T16:55:05Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "GetId",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.4",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off
 command/cognito-identity.get-id",
    "requestParameters": {
        "identityPoolId": "us-east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE",
        "logins": {
            "cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa":
 "HIDDEN_DUE_TO_SECURITY_REASONS"
        }
    },
    "responseElements": {
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
    },
    "requestID": "dc28def9-07c8-460a-a8f3-3816229e6664",
    "eventID": "c5c459d9-40ec-41fd-8f6b-57865d5a9975",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
```

```
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
}
```

에 대한 예제 CloudTrail 이벤트 GetOpenIdToken

다음 예제는 GetOpenIdToken 작업의 요청에 대한 로그 항목입니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown"
    },
    "eventTime": "2023-01-19T16:55:08Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "GetOpenIdToken",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.4",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off
 command/cognito-identity.get-open-id-token",
    "requestParameters": {
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE",
        "logins": {
            "cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa":
 "HIDDEN_DUE_TO_SECURITY_REASONS"
        }
    },
    "responseElements": {
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
    },
    "requestID": "a506ba18-10d7-4fdb-9548-a8187b2e38bb",
    "eventID": "19ffc1a6-6ed8-4580-a4e1-3062c5ce6457",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
```

```
    "eventCategory": "Data"
}
```

에 대한 예제 CloudTrail 이벤트 GetOpenIdTokenForDeveloperIdentity

다음 예제는 `GetOpenIdTokenForDeveloperIdentity` 작업의 요청에 대한 로그 항목입니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA1EXAMPLE:johns-AssumedRoleSession",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/johns-AssumedRoleSession",
        "accountId": "111122223333",
        "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROA1EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "attributes": {
                "creationDate": "2023-01-19T16:53:14Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2023-01-19T16:55:08Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "GetOpenIdTokenForDeveloperIdentity",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "27.0.3.154",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off
 command/cognito-identity.get-open-id-token-for-developer-identity",
    "requestParameters": {
        "tokenDuration": 900,
        "identityPoolId": "us-east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE",
        "logins": {
            "JohnsDeveloperProvider": "HIDDEN_DUE_TO_SECURITY_REASONS"
        }
    },
    "responseElements": {
```

```
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
    },
    "requestID": "b807df87-57e7-4dd6-b90c-b06f46a61c21",
    "eventID": "f26fed91-3340-4d70-91ae-cdf555547b76",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
}
```

에 대한 예제 CloudTrail 이벤트 UnlinkIdentity

다음 예제는 UnlinkIdentity 작업의 요청에 대한 로그 항목입니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown"
    },
    "eventTime": "2023-01-19T16:55:08Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "UnlinkIdentity",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.4",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off
 command/cognito-identity.unlink-identity",
    "requestParameters": {
        "logins": {
            "cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa":
 "HIDDEN_DUE_TO_SECURITY_REASONS"
        },
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE",
        "loginsToRemove": ["cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa"]
    },
    "responseElements": null,
    "requestID": "99c2c8e2-9c29-416f-bb17-b650a5cbada9",
    "eventID": "d8e26126-202a-43c2-b458-3f225efaedc7",
```

```
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
}
```

아마존 로그 인사이트를 사용하여 Amazon Cognito CloudTrail 이벤트 분석하기 CloudWatch

Amazon CloudWatch Logs Insights를 사용하여 Amazon Cognito CloudTrail 이벤트를 검색하고 분석할 수 있습니다. CloudWatch Logs에 이벤트를 전송하도록 트레일을 구성하면 트레일 설정과 일치하는 이벤트만 CloudTrail 전송합니다.

Amazon Cognito CloudTrail 이벤트를 쿼리하거나 조사하려면 CloudTrail 콘솔에서 트레일 설정에서 Management events 옵션을 선택해야 리소스에서 수행되는 관리 작업을 모니터링할 수 있습니다. AWS 오류, 비정상적인 활동 또는 비정상적인 사용자 행동을 식별하려는 경우 추적 설정에서 Insights 이벤트 옵션을 선택할 수도 있습니다.

Amazon Cognito 쿼리 샘플

Amazon CloudWatch 콘솔에서 다음 쿼리를 사용할 수 있습니다.

일반 쿼리

최근에 추가된 로그 이벤트 25개를 찾습니다.

```
fields @timestamp, @message | sort @timestamp desc | limit 25
| filter eventSource = "cognito-idp.amazonaws.com"
```

예외를 포함하여 최근에 추가된 로그 이벤트 25개의 목록을 표시합니다.

```
fields @timestamp, @message | sort @timestamp desc | limit 25
| filter eventSource = "cognito-idp.amazonaws.com" and @message like /Exception/
```

예외 및 오류 쿼리

Amazon Cognito 사용자 풀 sub를 사용하여 오류 코드 NotAuthorizedException과 함께 최근에 추가된 로그 이벤트 25개를 찾습니다.

```
fields @timestamp, additionalEventData.sub as user | sort @timestamp desc | limit 25
| filter eventSource = "cognito-idp.amazonaws.com" and errorCode=
 "NotAuthorizedException"
```

sourceIPAddress 및 해당 eventName이 있는 레코드의 수를 찾습니다.

```
filter eventSource = "cognito-idp.amazonaws.com"
| stats count(*) by sourceIPAddress, eventName
```

NotAuthorizedException 오류를 트리거한 상위 25개 IP 주소를 찾습니다.

```
filter eventSource = "cognito-idp.amazonaws.com" and errorCode=
 "NotAuthorizedException"
| stats count(*) as count by sourceIPAddress, eventName
| sort count desc | limit 25
```

ForgotPassword API를 호출한 상위 25개 IP 주소를 찾습니다.

```
filter eventSource = "cognito-idp.amazonaws.com" and eventName = 'ForgotPassword'
| stats count(*) as count by sourceIPAddress
| sort count desc | limit 25
```

Amazon Cognito에 대한 규정 준수 검증

타사 감사자는 AWS 여러 규정 준수 프로그램의 일환으로 Amazon Cognito의 보안 및 규정 준수를 평가합니다. 여기에는 SOC, PCI, FedRAMP, HIPAA 등이 포함됩니다.

특정 규정 준수 프로그램 범위 내 AWS 서비스 목록은 규정 준수 프로그램별 범위 내AWS 서비스 규정 준수 참조하십시오. 일반적인 내용은 AWS 규정 준수 프로그램을 참조하세요.

를 사용하여 타사 감사 보고서를 다운로드할 수 AWS Artifact있습니다. 자세한 내용은 보고서 https://docs.aws.amazon.com/artifact/latest/ug/downloading-documents.html 참조하십시오 AWS Artifact.

Amazon Cognito 사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 결정됩니다. AWS 에서는 규정 준수를 지원할 다음과 같은 리소스를 제공합니다.

- 보안 및 규정 준수 빠른 시작 안내서 – 이 배포 안내서에서는 아키텍처 고려 사항에 관해 설명하고 AWS에서 보안 및 규정 준수에 중점을 둔 기본 환경을 배포하기 위한 단계를 제공합니다.
- HIPAA 보안 및 규정 준수를 위한 설계 백서 — 이 백서는 기업이 HIPAA 준수 애플리케이션을 개발하는 데 사용할 AWS 수 있는 방법을 설명합니다.
- AWS 규정 준수 리소스 규정 — 이 통합 문서 및 가이드 모음은 해당 산업 및 지역에 적용될 수 있습니다.
- AWS Config 개발자 안내서의 규칙을 사용하여 리소스 평가 — AWS Config; 는 리소스 구성이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가합니다.
- AWS Security Hub— 이 AWS 서비스는 보안 업계 표준 및 모범 사례를 준수하는지 확인하는 데 도움이 AWS 되는 내부 보안 상태를 종합적으로 보여줍니다.

Amazon Cognito의 복원성

AWS 글로벌 인프라는 지역 및 가용 AWS 영역을 중심으로 구축됩니다. 리전은 물리적으로 분리되고 격리된 다수의 가용 영역을 제공하며, 이러한 영역은 짧은 지연 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워크를 통해 연결되어 있습니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 지역 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하십시오.

주제

- 리전 데이터 고려 사항

리전 데이터 고려 사항

Amazon Cognito 사용자 풀은 각각 한 AWS 지역에서 생성되며 해당 지역에만 사용자 프로필 데이터를 저장합니다. 선택적 기능의 구성 방식에 따라 사용자 풀은 사용자 데이터를 다른 AWS 지역으로 보낼 수 있습니다.

- Amazon Cognito 사용자 풀을 사용한 이메일 주소 라우팅 확인에 기본 `no-reply@verificationemail.com` 이메일 주소 설정을 사용할 경우, 이메일은 연결된 사용자 풀과 동일한 리전을 통해 라우팅됩니다.
- Amazon Cognito 사용자 풀을 사용하여 Amazon Simple Email Service (Amazon SES) 를 구성하는 데 다른 이메일 주소를 사용하는 경우, 해당 이메일 주소는 Amazon SES의 이메일 주소와 연결된 지역을 AWS 통해 라우팅됩니다.

- Amazon Cognito 사용자 풀에서 전송된 SMS 메시지는 이메일 또는 전화 확인 구성에 다르게 명시되어 있지 않는 한 동일한 리전 Amazon SNS를 통해 라우팅됩니다.
- Amazon Cognito 사용자 풀과 함께 Amazon Pinpoint 분석이 사용되는 경우 이벤트 데이터는 미국 동부(버지니아 북부) 리전으로 라우팅됩니다.

> **Note**
>
> Amazon Pinpoint는 북미, 유럽, 아시아 및 오세아니아의 여러 AWS 지역에서 사용할 수 있습니다. Amazon Pinpoint 리전에는 Amazon Pinpoint API가 포함되어 있습니다. Amazon Cognito에서 해당 Amazon Pinpoint 리전을 지원하는 경우 Amazon Cognito는 동일한 Amazon Pinpoint 리전 내의 Amazon Pinpoint 프로젝트로 이벤트를 전송합니다. Amazon Pinpoint에서 해당 리전을 지원하지 않는 경우 Amazon Cognito는 us-east-1에서만 이벤트 전송을 지원합니다. 자세한 Amazon Pinpoint 리전 정보는 Amazon Pinpoint 엔드포인트 및 할당량과 Amazon Cognito 사용자 풀로 Amazon Pinpoint 분석 사용을 참조하세요.

Amazon Cognito의 인프라 보안

관리형 서비스인 Amazon Cognito는 AWS 글로벌 네트워크 보안으로 보호됩니다. AWS 보안 서비스 및 인프라 AWS 보호 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하십시오. 인프라 보안 모범 사례를 사용하여 AWS 환경을 설계하려면 Security Pillar AWS Well-Architected Framework의 인프라 보호를 참조하십시오.

AWS 게시된 API 호출을 사용하여 네트워크를 통해 Amazon Cognito에 액세스할 수 있습니다. 고객은 다음을 지원해야 합니다.

- 전송 계층 보안(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.

Amazon Cognito 사용자 풀의 구성 및 취약성 분석

AWS 게스트 운영 체제 (OS) 및 데이터베이스 패치, 방화벽 구성, 재해 복구와 같은 기본 보안 작업을 처리합니다. 적합한 제3자가 이 절차를 검토하고 인증하였습니다. 자세한 내용은 다음 리소스를 참조하세요.

- Amazon Cognito에 대한 규정 준수 검증
- 공동 책임 모델

AWS 아마존 Cognito에 대한 관리형 정책

사용자, 그룹 및 역할에 권한을 추가하려면 정책을 직접 작성하는 것보다 AWS 관리형 정책을 사용하는 것이 더 쉽습니다. 팀에 필요한 권한만 제공하는 IAM 고객 관리형 정책을 생성하기 위해서는 시간과 전문 지식이 필요합니다. 빠르게 시작하려면 AWS 관리형 정책을 사용할 수 있습니다. 이러한 정책은 일반적인 사용 사례를 다루며 AWS 계정에서 사용할 수 있습니다. AWS 관리형 정책에 대한 자세한 내용은 IAM 사용 설명서의AWS 관리형 정책을 참조하십시오.

AWS 서비스는 AWS 관리형 정책을 유지 관리하고 업데이트합니다. AWS 관리형 정책에서는 권한을 변경할 수 없습니다. 서비스에서 때때로 추가 권한을 AWS 관리형 정책에 추가하여 새로운 기능을 지원합니다. 이 타입의 업데이트는 정책이 연결된 모든 보안 인증(사용자, 그룹 및 역할)에 적용됩니다. 서비스는 새로운 기능이 시작되거나 새 작업을 사용할 수 있을 때 AWS 관리형 정책에 업데이트됩니다. 서비스는 AWS 관리형 정책에서 권한을 제거하지 않으므로 정책 업데이트로 인해 기존 권한이 손상되지 않습니다.

또한 여러 서비스에 걸친 작업 기능에 대한 관리형 정책을 AWS 지원합니다. 예를 들어 ReadOnlyAccess AWS 관리형 정책은 모든 AWS 서비스와 리소스에 대한 읽기 전용 액세스를 제공합니다. 서비스가 새 기능을 시작하면 새 작업 및 리소스에 대한 읽기 전용 권한이 AWS 추가됩니다. 직무 정책의 목록과 설명은 IAM 사용 설명서의 직무에 관한AWS 관리형 정책을 참조하세요.

IAM 콘솔을 통해 Amazon Cognito에 대한 액세스 권한을 부여하는 데 사용할 수 있는 다양한 정책이 제공됩니다.

- `AmazonCognitoPowerUser` - 자격 증명 풀 및 사용자 풀의 모든 측면을 액세스하고 관리할 수 있는 권한입니다. 이 정책에 대한 권한을 보려면 을 참조하십시오 AmazonCognitoPowerUser.
- `AmazonCognitoReadOnly` - 자격 증명 풀 및 사용자 풀에 대한 읽기 전용 액세스 권한입니다. 이 정책에 대한 권한을 보려면 을 참조하십시오 AmazonCognitoReadOnly.

- `AmazonCognitoDeveloperAuthenticatedIdentities` - 인증 시스템을 Amazon Cognito와 통합할 권한입니다. 이 정책에 대한 권한을 보려면 을 참조하십시오 AmazonCognitoDeveloperAuthenticatedIdentities.

이러한 정책은 Amazon Cognito 팀에서 관리하므로 새 API가 추가되더라도 IAM 사용자는 계속해서 동일한 수준의 액세스 권한을 갖게 됩니다.

> **Note**
>
> 새 자격 증명 풀을 생성할 때 인증된 사용자 및 게스트 사용자 액세스를 위한 새 역할을 자동으로 생성할 수 있습니다. 새 IAM 역할로 자격 증명 풀을 생성하는 관리자에게는 역할을 생성할 수 있는 IAM 권한도 있어야 합니다.

인증되지 않은 게스트 액세스가 있는 ID 풀은 인증되지 않은 사용자에게 추가 AWS 관리형 정책을 세션 정책으로 적용합니다. `AmazonCognitoUnAuthedIdentitiesSessionPolicy` 이 AWS 관리형 정책은 관리상의 용도가 아닙니다. 대신, 자격 증명 풀 향상된 인증 흐름에서 게스트 사용자에게 적용할 수 있는 권한의 범위를 제한합니다. 자세한 정보는 IAM 역할을 참조하세요.

Amazon Cognito에서 관리형 정책을 업데이트했습니다. AWS

이 서비스가 이러한 변경 사항을 추적하기 시작한 이후 Amazon Cognito의 AWS 관리형 정책 업데이트에 대한 세부 정보를 확인하십시오. 이 페이지의 변경 사항에 대한 자동 알림을 받아보려면 Amazon Cognito Document history(문서 기록) 페이지에서 RSS 피드를 구독하세요.

변경 사항	설명	날짜
`AmazonCognitoUnAuthedIdentitiesSessionPolicy` - 새 정책	자격 증명 풀에서 게스트 사용자의 권한 범위 축소를 위한 AWS 관리형 정책이 추가되었습니다.	2023년 7월 14일

변경 사항	설명	날짜
`AmazonCognitoPowerUser` 및 `AmazonCognitoReadOnly` - 기존 정책 업데이트	고급 사용자가 Amazon Cognito 사용자 풀에 대한 AWS WAF 웹 ACL의 연결을 보고 관리할 수 있는 새로운 권한이 추가되었습니다. 읽기 전용 사용자가 Amazon Cognito 사용자 풀에 대한 AWS WAF 웹 ACL의 연결을 볼 수 있는 새 권한이 추가되었습니다.	2022년 7월 19일
`AmazonCognitoPowerUser` -기존 정책 업데이트	Amazon Cognito가 Amazon Simple Notification Service `PutIdentityPolicy` 및 `ListConfigurationSets` 작업을 호출할 수 있도록 허용하는 새 권한을 추가했습니다. 이 변경으로 인해 Amazon Cognito 사용자 풀은 Amazon SES 전송 권한 부여 정책을 업데이트하고 사용자 풀에서 이메일 전송을 구성할 때 Amazon SES 구성 집합을 적용할 수 있습니다.	2021년 11월 17일

변경 사항	설명	날짜
`AmazonCognitoPowerUser` -기존 정책 업데이트	Amazon Cognito가 Amazon Simple Notification Service의 `GetSMSSandboxAccountStatus` 작업을 호출하도록 허용하는 새 권한이 추가되었습니다. 이 변경으로 인해 Amazon Cognito 사용자 풀은 사용자 풀을 통해 모든 최종 사용자에게 메시지를 전송하기 위해 Amazon Simple Notification Service 샌드박스를 종료해야 할지 여부를 결정할 수 있습니다.	2021년 6월 1일
Amazon Cognito가 변경 사항 추적 시작	Amazon Cognito는 AWS 관리형 정책의 변경 사항을 추적하기 시작했습니다.	2021년 3월 1일

Amazon Cognito 리소스 태깅

태그는 사용자 또는 AWS가 AWS 리소스에 할당하는 메타데이터 레이블입니다. 각 태그는 키와 값으로 구성됩니다. 사용자가 할당하는 태그에 대해 키와 값을 정의합니다. 예를 들어 키를 `stage`로 정의하고 리소스 하나의 값을 `test`로 정의할 수 있습니다.

태그는 다음을 지원합니다.

- AWS 리소스를 식별하고 정리합니다. 많은 AWS 서비스가 태깅을 지원하므로 다양한 서비스의 리소스에 동일한 태그를 할당할 수 있습니다. 이렇게 하면 어떤 리소스가 관련되어 있는지 알 수 있습니다. 예를 들어 Amazon DynamoDB 테이블에 할당한 것과 동일한 태그를 Amazon Cognito 사용자 풀에 할당할 수 있습니다.
- AWS 비용을 추적합니다. AWS Billing and Cost Management 대시보드에서 태그를 활성화할 수 있습니다. AWS는 비용 할당 태그를 사용하여 비용을 분류하고 월별 비용 할당 보고서를 전달합니다. 자세한 내용은 AWS Billing 사용 설명서의 비용 할당 태그 사용을 참조하세요.
- 할당된 태그를 기반으로 리소스에 대한 액세스를 제어합니다. AWS Identity and Access Management(IAM) 정책 조건에서 태그 키와 값을 지정하여 액세스를 제어할 수 있습니다. 예를 들어 사용자 풀에 값이 해당 사용자의 이름인 `owner` 태그가 있을 때만 사용자가 사용자 풀을 업데이트하도록 허용할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 태그를 사용한 액세스 제어를 참조하세요.

AWS Command Line Interface 또는 Amazon Cognito API를 사용하여 사용자 및 자격 증명 풀에 대한 태그를 추가, 편집 또는 삭제할 수 있습니다. Amazon Cognito 콘솔을 사용하여 사용자 풀에 대한 태그를 관리할 수도 있습니다.

태그 사용 방법에 대한 팁은 AWS Answers 블로그의 AWS 태깅 전략 게시글을 참조하세요.

다음 섹션에서는 Amazon Cognito의 태그에 대한 추가 정보를 제공합니다.

Amazon Cognito에서 지원되는 리소스

Amazon Cognito의 다음 리소스는 태깅을 지원합니다.

- 사용자 풀
- 자격 증명 풀

태그 제한

Amazon Cognito 리소스에 대한 태그에 적용되는 제한 사항은 다음과 같습니다.

- 리소스에 할당할 수 있는 최대 태그 수 - 50개
- 최대 키 길이 - 유니코드 128자
- 최대 값 길이 - 유니코드 256자
- 키 및 값에 사용할 수 있는 문자 - a-z, A-Z, 0-9, 공백, _ . : / = + - @ 문자
- 키와 값은 대/소문자를 구분합니다
- 키 접두사로 `aws:`를 사용하지 마세요. AWS 전용입니다.

Amazon Cognito 콘솔을 사용하여 태그 관리

Amazon Cognito 콘솔을 사용하여 사용자 풀에 할당된 태그를 관리할 수 있습니다.

사용자 풀에 태그를 추가하려면

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 AWS 자격 증명을 입력합니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. [사용자 풀 속성(User pool properties)] 탭을 선택하고 [태그(Tags)]를 찾습니다.
5. [태그 추가(Add tags)]를 선택하여 첫 번째 태그를 추가합니다. 이전에 이 사용자 풀에 태그를 할당한 경우 [태그 관리(Manage tags)]에서 [다른 태그 추가(Add another)]를 선택합니다.
6. 태그 키(Tag Key)와 태그 값(Tag Value)의 값을 지정합니다.
7. 추가하려는 각 태그에 대해 [다른 태그 추가(Add another)]를 선택합니다.
8. 태그 추가가 완료되면 변경 사항 저장(Save changes)을 선택합니다.

[태그 관리(Manage tags)] 페이지에서 기존 태그의 키와 값을 편집할 수도 있습니다. 태그를 제거하려면 제거를 선택합니다.

AWS CLI 예제

AWS CLI에서는 Amazon Cognito 사용자 풀과 자격 증명 풀에 할당하는 태그를 관리하는 데 도움이 되는 명령을 제공합니다.

태그 할당

다음 명령을 사용하여 기존 사용자 풀 및 자격 증명 풀에 태그를 할당합니다.

Example 사용자 풀에 대한 **tag-resource** 명령

cognito-idp 명령 세트에서 tag-resource를 사용하여 사용자 풀에 태그를 할당합니다.

```
$ aws cognito-idp tag-resource \
> --resource-arn user-pool-arn \
> --tags Stage=Test
```

이 명령에는 다음 파라미터가 포함되어 있습니다.

- resource-arn - 태그를 적용할 사용자 풀의 Amazon 리소스 이름(ARN)입니다. ARN을 조회하려면 Amazon Cognito 콘솔에서 사용자 풀을 선택하고 일반 설정(General settings) 탭에서 풀 ARN(Pool ARN) 값을 확인합니다.
- tags - 태그의 키-값 페어(*key*=*value* 포맷)입니다.

여러 태그를 한 번에 할당하려면 쉼표로 구분된 목록으로 지정합니다.

```
$ aws cognito-idp tag-resource \
> --resource-arn user-pool-arn \
> --tags Stage=Test,CostCenter=80432,Owner=SysEng
```

Example 자격 증명 풀에 대한 **tag-resource** 명령

cognito-identity 명령 세트에서 tag-resource를 사용하여 자격 증명 풀에 태그를 할당합니다.

```
$ aws cognito-identity tag-resource \
> --resource-arn identity-pool-arn \
> --tags Stage=Test
```

이 명령에는 다음 파라미터가 포함되어 있습니다.

- resource-arn - 태그를 적용할 자격 증명 풀의 Amazon 리소스 이름(ARN)입니다. ARN을 조회하려면 Amazon Cognito 콘솔에서 자격 증명 풀을 선택하고 [자격 증명 풀 편집(Edit identity pool)]을 선택합니다. 그런 다음 자격 증명 풀 ID(Identity pool ID)에서 ARN 표시(Show ARN)를 선택합니다.
- tags - 태그의 키-값 페어(*key*=*value* 포맷)입니다.

여러 태그를 한 번에 할당하려면 쉼표로 구분된 목록으로 지정합니다.

```
$ aws cognito-identity tag-resource \
> --resource-arn identity-pool-arn \
> --tags Stage=Test,CostCenter=80432,Owner=SysEng
```

태그 보기

다음 명령을 사용하여 사용자 풀 및 자격 증명 풀에 할당한 태그를 봅니다.

Example 사용자 풀에 대한 **list-tags-for-resource** 명령

cognito-idp 명령 세트에서 list-tags-for-resource를 사용하여 사용자 풀에 할당한 태그를 봅니다.

```
$ aws cognito-idp list-tags-for-resource --resource-arn user-pool-arn
```

Example 자격 증명 풀에 대한 **list-tags-for-resource** 명령

cognito-identity 명령 세트에서 list-tags-for-resource를 사용하여 자격 증명 풀에 할당한 태그를 봅니다.

```
$ aws cognito-identity list-tags-for-resource --resource-arn identity-pool-arn
```

태그 제거

다음 명령을 사용하여 사용자 풀 및 자격 증명 풀에서 태그를 제거합니다.

Example 사용자 풀에 대한 **untag-resource** 명령

cognito-idp 명령 세트에서 untag-resource를 사용하여 사용자 풀에서 태그를 제거합니다.

```
$ aws cognito-idp untag-resource \
> --resource-arn user-pool-arn \
> --tag-keys Stage CostCenter Owner
```

--tag-keys 파라미터의 경우 태그 키를 하나 이상 지정합니다. 태그 값을 포함해서는 안 됩니다. 키를 공백으로 구분합니다.

Example 자격 증명 풀에 대한 **untag-resource** 명령

cognito-identity 명령 세트에서 untag-resource를 사용하여 자격 증명 풀에서 태그를 제거합니다.

```
$ aws cognito-identity untag-resource \
> --resource-arn identity-pool-arn \
> --tag-keys Stage CostCenter Owner
```

--tag-keys 파라미터의 경우 태그 키를 하나 이상 지정합니다. 태그 값을 포함해서는 안 됩니다.

> **Important**
>
> 사용자 또는 자격 증명 풀을 삭제한 후에도 삭제된 풀과 관련한 태그가 삭제 후 최대 30일 동안 콘솔 또는 API 호출에 계속 표시될 수 있습니다.

리소스를 생성할 때 태그 적용

다음 명령을 사용하여 사용자 풀 또는 자격 증명 풀을 생성할 때 태그를 할당합니다.

Example 태그가 있는 **create-user-pool** 명령

create-user-pool 명령을 사용하여 사용자 풀을 생성할 때 --user-pool-tags 파라미터로 태그를 지정할 수 있습니다.

```
$ aws cognito-idp create-user-pool \
> --pool-name user-pool-name \
> --user-pool-tags Stage=Test,CostCenter=80432,Owner=SysEng
```

태그의 키-값 페어는 *key*=*value* 포맷이어야 합니다. 여러 태그를 추가하는 경우 태그를 쉼표로 구분된 목록으로 지정합니다.

Example 태그가 있는 **create-identity-pool** 명령

create-identity-pool 명령을 사용하여 자격 증명 풀을 생성할 때 --identity-pool-tags 파라미터로 태그를 지정할 수 있습니다.

```
$ aws cognito-identity create-identity-pool \
> --identity-pool-name identity-pool-name \
```

```
> --allow-unauthenticated-identities \
> --identity-pool-tags Stage=Test,CostCenter=80432,Owner=SysEng
```

태그의 키-값 페어는 *key*=*value* 포맷이어야 합니다. 여러 태그를 추가하는 경우 태그를 쉼표로 구분된 목록으로 지정합니다.

Amazon Cognito API를 사용하여 태그 관리

Amazon Cognito API에서 다음 작업을 사용하여 사용자 풀 및 자격 증명 풀의 태그를 관리할 수 있습니다.

사용자 풀 태그에 대한 API 작업

다음 API 작업을 사용하여 사용자 풀에 대한 태그를 할당하고 보고 제거합니다.

- TagResource
- ListTagsForResource
- UntagResource
- CreateUserPool

자격 증명 풀 태그에 대한 API 작업

다음 API 작업을 사용하여 자격 증명 풀에 대한 태그를 할당하고 보고 제거합니다.

- TagResource
- ListTagsForResource
- UntagResource
- CreateIdentityPool

Amazon Cognito의 할당량

Amazon Cognito에는 계정에서 수행할 수 있는 최대 작업 수에 대한 기본 할당량(구 명칭: 한도)이 있습니다. Amazon Cognito에는 또한 Amazon Cognito 리소스의 최대 수와 크기에 대한 할당량이 지정되어 있습니다.

각 Amazon Cognito 할당량은 일대일 요청의 최대 양을 나타냅니다 AWS 리전 . AWS 계정예를 들어 앱은 미국 동부(버지니아 북부)의 모든 사용자 풀을 대상으로 하는 `UserAuthentication` 작업에 최대 기본 할당량(RPS) 요율로 API 요청을 보낼 수 있습니다. 아시아 태평양 (도쿄) 의 앱은 해당 지역의 모든 사용자 풀에 대해 동일한 양의 요청을 생성할 수 있습니다. AWS 한 번에 한 지역에서만 할당량 증가 요청을 허용할 수 있습니다. 미국 동부(버지니아 북부)의 할당량 증가는 아시아 태평양(도쿄)의 최대 요청률에 영향을 주지 않습니다.

주제

- API 요청률 할당량 이해
- API 요청률 할당량 관리
- Amazon Cognito 사용자 풀 API 작업 범주 및 요청 속도 할당량
- Amazon Cognito 자격 증명 풀(페더레이션 자격 증명) API연산 요청 속도 할당량
- 리소스 수 및 크기에 대한 할당량

API 요청률 할당량 이해

할당량 분류

Amazon Cognito는 API 작업에 대해 최대 요청 속도를 적용합니다. Amazon Cognito에서 제공하는 API 작업에 대한 자세한 내용은 Amazon Cognito API 및 엔드포인트 참조 섹션을 참조하세요. 사용자 풀의 경우 이러한 작업은 `UserAuthentication` 또는 `UserCreation` 같은 일반적인 사용 사례의 범주로 그룹화됩니다. 범주별 사용자 풀 API 작업 목록은 을 참조하십시오. Amazon Cognito 사용자 풀 API 작업 범주 및 요청 속도 할당량

Service Quotas 콘솔에서 범주 사용자 풀 및 자격 증명 풀별로 할당량 사용량을 추적할 수 있습니다. Amazon Cognito 사용자 풀의 요청 비율이 할당량을 초과하거나 할당량을 초과하는 경우 추가 용량을 구입할 수 있습니다. Service Quotas 콘솔에서 범주별 사용자 풀 할당량 사용량 및 구매 할당량 증가를 추적할 수 있습니다.

작업 할당량은 범주 내의 모든 작업에 대한 최대 초당 요청 수(RPS)로 정의됩니다. Amazon Cognito 사용자 풀 서비스는 각 범주의 모든 작업에 할당량을 적용합니다. 예를 들어 UserCreation 범주에는 SignUp, ConfirmSignUp, AdminCreateUser, AdminConfirmSignUp이라는 네 가지 작업이 포함됩니다. 총 할당량을 기준으로 50RPS가 할당됩니다. 동시에 여러 작업이 수행되는 경우, 이 범주 내의 각 작업은 최대 50개의 RPS를 개별적으로 호출하거나 조합할 수 있습니다.

> **Note**
>
> 범주 할당량은 사용자 풀에만 적용됩니다. Amazon Cognito는 각 ID 풀 할당량을 단일 작업에 적용합니다. 카테고리별 및 작업별 요청 속도 할당량 모두에 대해 한 지역의 모든 사용자 풀 또는 ID 풀에서 발생한 모든 요청의 총 비율을 AWS 측정합니다. AWS 계정

특별 요청 속도 처리가 포함된 Amazon Cognito 사용자 풀 API 작업

작업 할당량은 범주 수준에서 합산된 전체 요청을 기준으로 측정되고 적용됩니다. 단, 특수 처리 규칙이 적용되는 AdminRespondToAuthChallenge 및 RespondToAuthChallenge 작업은 예외입니다.

UserAuthentication카테고리에는 Amazon Cognito 사용자 풀 API의 네 가지 작업(AdminInitiateAuth, InitiateAuthAdminRespondToAuthChallenge, 및) 이 포함됩니다. RespondToAuthChallenge 또한 호스팅된 UI의 사용자 인증이 이 할당량에 기여합니다. InitiateAuth 및 AdminInitiateAuth 작업은 범주 할당량에 따라 측정되고 적용됩니다. RespondToAuthChallenge과 AdminRespondToAuthChallenge의 매칭 작업은 UserAuthentication 범주 제한의 세 배에 해당하는 별도 할당량이 적용됩니다. 이렇게 증가된 할당량은 앱에 설정된 여러 인증 문제를 수용할 수 있습니다. 해당 할당량은 대다수의 사용 사례를 수용하기에 충분합니다. 앱이 인증 챌린지에 최대 3회까지 응답하면 추가 요청은 UserAuthentication 카테고리 할당량에 포함됩니다. 다단계 인증 (MFA), 디바이스 인증 및 사용자 지정 인증은 모두 사용자 풀에 통합할 수 있는 챌린지 프롬프트의 예입니다.

예를 들어 UserAuthentication 카테고리의 할당량이 80RPS인 경우 최대 240RPS (3 x 80 RPS)의 AdminRespondToAuthChallenge 속도로 RespondToAuthChallenge 또는 전화를 걸 수 있습니다. 사용자 풀에서 인증당 4라운드의 챌린지를 요청하고 초당 70명의 사용자가 로그인하는 경우 총 RespondToAuthChallenge 280RPS (70 x 4) 로 할당량보다 40RPS 높은 수치입니다. 추가 40RPS가 70건의 InitiateAuth 호출에 추가되어 UserAuthentication 범주의 총 사용량은 110RPS(40+70)가 됩니다. 이 값이 80RPS x 30RPS로 설정된 카테고리 할당량을 초과하기 때문에 Amazon Cognito는 앱으로부터의 요청을 제한합니다.

월별 활성 사용자)

Amazon Cognito는 사용자 풀 청구를 계산할 때 월별 활성 사용자 (MAU) 별로 요금을 청구합니다. 할당량 증가 요청을 계획할 때 현재 및 예상 MAU 수를 고려하십시오. 한 달 이내에 관련된 자격 증명 작업이 있는 사용자는 MAU로 계산됩니다. 사용자를 활성으로 만드는 활동에는 다음이 포함됩니다.

- 사용자 가입 또는 관리 생성
- 로그인
- 로그아웃
- 사용자 계정 확인 또는 속성 확인
- 암호 재설정
- 사용자 속성, 그룹 멤버십 또는 MFA 기본 설정 변경
- 사용자의 세부 속성 쿼리
- 사용자 활성화, 비활성화 또는 삭제

> **Note**
>
> 사용자의 쿼리 세부 속성 카테고리에는 API 작업이 AdminGetUser포함되지만 포함되지 않습니다. ListUsers 대규모 사용자 풀의 세부 user-by-user 쿼리는 AWS 청구서에 큰 영향을 미칠 수 있습니다. 과도한 요금이 부과되지 않도록 하려면 외부 데이터베이스를 통해 사용자 데이터를 `ListUsers` 수집하거나 외부 데이터베이스에 사용자 정보를 저장하세요.

API 요청률 할당량 관리

할당량 요구 사항 파악

> **Important**
>
> `UserAuthenticationUserCreationAccountRecovery`, 또는 등의 카테고리에 대한 Amazon Cognito 할당량을 늘리는 경우 다른 카테고리의 할당량을 늘려야 할 수 있습니다. AWS 서비스예를 들어 Amazon Cognito가 Amazon Simple Notification Service(Amazon SNS) 및 Amazon Simple Email Service(Amazon SES)를 통해 보내는 메시지는 해당 서비스의 요청 속도 할당량이 충분하지 않은 경우 실패할 수 있습니다.

할당량 요구 사항을 계산하려면 특정 기간 동안 애플리케이션과 상호 작용할 활성 사용자 수를 구합니다. 예를 들어 애플리케이션에서 평균 100만 명의 활성 사용자가 8시간 내에 로그인할 것으로 예상되는 경우 초당 평균 35명의 사용자를 인증할 수 있어야 합니다.

또한 평균 사용자 세션이 2시간이라고 가정하고 토큰이 1시간 후에 만료되도록 구성한 경우 각 사용자가 세션 중에 토큰을 한 번 새로 고쳐야 합니다. 이 경우 `UserAuthentication` 범주에서 부하를 지원하는 데 필요한 평균 할당량은 70RPS입니다.

8시간 동안의 사용자 로그인 빈도 차이를 고려하여 peak-to-average 비율을 3:1 로 가정할 경우 원하는 할당량인 200RPS가 필요합니다. `UserAuthentication`

> **Note**
>
> 각 사용자 작업에 대해 여러 작업을 호출하는 경우 범주 수준에서 개별 작업 호출 속도를 합산해야 합니다.

할당량 한도에 대한 요청 비율을 최적화하세요.

API 속도 한도를 높이면 AWS 청구서에 비용이 추가되므로 할당량 증가를 요청하기 전에 사용량 모델을 조정하는 것이 좋습니다. 다음은 요청 비율을 최적화하는 앱 아키텍처의 몇 가지 예시입니다.

백오프 대기 기간 후에 다시 시도

각 API 호출에서 오류를 포착한 다음 백오프 기간 후에 다시 시도할 수 있습니다. 비즈니스 요구 사항과 로드에 따라 백오프 알고리즘을 조정할 수 있습니다. Amazon SDK에는 재시도 로직이 내장되어 있습니다. 자세한 내용은 AWS빌드 기반 도구를 참조하십시오.

자주 업데이트되는 속성에 외부 데이터베이스 사용

애플리케이션에서 사용자 지정 속성을 읽거나 쓰기 위해 사용자 풀을 여러 번 호출해야 하는 경우 외부 스토리지를 사용합니다. 기본 설정 데이터베이스를 사용하여 사용자 지정 속성을 저장하거나 로그인 중에 캐시 계층을 사용하여 사용자 프로파일을 로드할 수 있습니다. 필요한 경우 사용자 풀에서 사용자 프로파일을 다시 로드하는 대신, 캐시에서 이 프로파일을 참조할 수 있습니다.

클라이언트 측에서 JSON 웹 토큰 (JWT) 의 유효성을 검사하세요.

애플리케이션은 JWT 토큰을 신뢰하기 전에 먼저 검증해야 합니다. API 요청을 사용자 풀에 보내지 않고도 클라이언트 측에서 토큰의 서명과 유효성을 확인할 수 있습니다. 토큰을 검증한 후에는 토큰의 클레임을 신뢰하고 클레임을 더 많은 `getUser` API 호출을 하는 대신 클레임을 사용할 수 있습니다. 자세한 내용은 JSON 웹 토큰 확인을 참조하세요.

대기실에서 웹 애플리케이션에 대한 트래픽 제한

시험 응시나 라이브 이벤트 참석 등 제한 시간이 있는 이벤트 중에 로그인한 많은 사용자의 트래픽이 예상되는 경우 자체 조절 메커니즘을 사용하여 요청 트래픽을 최적화할 수 있습니다. 예를 들어, 사용자가 세션을 사용할 수 있을 때까지 대기실을 설정하여 사용 가능한 용량이 있을 때 요청을 처리할 수 있습니다. 대기실 참조 아키텍처에 대해서는 AWS 가상 대기실 솔루션을 참조하세요.

캐시 JWT

만료될 때까지 액세스 토큰을 재사용하십시오. API Gateway에서 토큰 캐싱을 사용하는 예제 프레임워크는 을 참조하십시오캐싱 토큰. 사용자 정보를 쿼리하기 위해 API 요청을 생성하는 대신 ID 토큰을 만료될 때까지 캐시하고 캐시에서 사용자 속성을 읽습니다.

에서 API 요청 속도를 사용하는 방법에 대한 자세한 내용은 AWS워크로드의 API 제한 관리 및 모니터링을 참조하십시오. 청구서에 비용을 추가하는 Amazon Cognito 작업을 최적화하는 방법에 대한 자세한 내용은 을 AWS 참조하십시오. 비용 관리

할당량 사용량 추적

Amazon Cognito는 Amazon에서 계정 수준에서 각 API 작업 범주에 CloudWatch 대한 `ThrottleCount` 지표를 `CallCount` 생성하고 측정합니다. `CallCount`를 사용하여 범주와 관련된 고객의 총 호출 수를 추적할 수 있습니다. `ThrottleCount`를 사용하여 범주와 관련된 총 제한된 호출 수를 추적할 수 있습니다. `CallCount` 및 `ThrottleCount` 지표와 `Sum` 통계를 사용하여 범주의 총 호출 수를 계산할 수 있습니다. 자세한 내용은 CloudWatch 사용량 지표를 참조하십시오.

서비스 할당량을 모니터링할 때 사용률은 사용 중인 서비스 할당량의 백분율입니다. 예를 들어 할당량 값이 리소스 200개이고, 150개의 리소스가 사용 중인 경우 사용률은 75%입니다. 사용량은 서비스 할당량으로 사용 중인 리소스 또는 작업의 수입니다.

CloudWatch 지표를 통한 사용량 추적

를 사용하여 Amazon Cognito 사용자 풀 사용률 지표를 추적하고 수집할 수 있습니다. CloudWatch CloudWatch대시보드에는 사용하는 모든 항목에 대한 AWS 서비스 지표가 표시됩니다. 를 사용하면 지표 경보를 생성하여 알림을 받거나 모니터링 중인 특정 리소스를 변경할 수 있습니다. CloudWatch CloudWatch 지표에 대한 자세한 내용은 CloudWatch 사용량 지표 추적을 참조하십시오.

Service Quotas 지표를 통한 사용률 추적

Amazon Cognito 사용자 풀은 서비스 할당량 사용량을 표시하고 관리하는 콘솔 인터페이스인 Service Quotas와 통합되어 있습니다. Service Quotas 콘솔에서 특정 할당량의 값을 조회하고, 모니터링 정보

를 보고, 할당량 증가를 요청하거나, 경보를 설정할 수 있습니다. CloudWatch 계정이 한동안 활성화되면 리소스 사용률 그래프를 볼 수 있습니다.

Amazon Cognito 사용자 풀 및 Amazon Cognito 자격 증명 풀에 대한 Service Quotas 콘솔의 적용된 계정 수준 할당량 값 열에는 현재 할당량이 표시됩니다. 사용률 열에는 현재 할당량 사용률이 표시됩니다. 조정 가능한 Amazon Cognito 사용자 풀 requests-per-second (RPS) 할당량은 현재 사용량을 표시합니다. Service Quotas 콘솔에서는 지표로 CloudWatch 이동하여 선택한 할당량 지표를 자세히 살펴볼 수도 있습니다. Service Quotas 콘솔에서 할당량을 확인하는 방법에 대한 자세한 내용은 Service Quotas 보기를 참조하세요.

월간 실사용자 (MAU) 를 추적하세요.

사용자 풀의 월간 활성 사용자 (MAU) 수는 요청률 할당량 증가를 계획하는 데 중요한 데이터를 제공합니다. API 요청 비율을 특정 기간 동안 활동한 사용자 수와 비교할 수 있습니다. 이러한 지식을 바탕으로 애플리케이션의 활성 사용자 증가가 사용 모델의 할당량에 어떤 영향을 미치는지 계산할 수 있습니다. 예를 들어, 미국 서부 (오레곤) 에서 애플리케이션을 통합하여 한 달에 2백만 명의 활성 사용자가 발생했고 해당 `UserAuthentication` 범주의 기본 할당량인 초당 요청 120개 (RPS) 로 인한 제한 오류가 발생했다고 가정해 보겠습니다. 광고 캠페인이 성공하기 전인 지난 달에는 MAU가 100만 개에 달했고 애플리케이션이 80RPS를 초과한 적이 없었습니다. 새로운 TV 광고로 인해 이와 비슷한 급증세를 보일 것으로 예상되면 40RPS를 추가로 구매하여 다음 백만 명의 사용자를 수용할 수 있도록 조정된 할당량을 160RPS로 설정할 수 있습니다.

MAU를 검토하려면

AWS Billing 콘솔에 접속하여 최근 청구서를 검토하세요. 서비스별 요금에서 Cognito를 필터링하여 해당 청구 기간의 MAU 내역을 볼 수 있습니다.

할당량 증가 요청

Amazon Cognito에는 각 사용자 풀과 자격 증명 풀에서 수행할 수 있는 초당 최대 작업 수에 대한 할당량이 있습니다. AWS 리전조정 가능한 Amazon Cognito 사용자 풀 API 요청 속도 할당량을 인상하여 구매할 수 있습니다. 현재 할당량을 확인하고 Service Quotas 콘솔 또는 Service Quotas API 작업 및 에서 증가된 할당량을 구매하세요. `ListAWSDefaultServiceQuotas` `RequestServiceQuotaIncrease`

- Service Quotas 콘솔을 사용하여 할당량 증가를 구매하려면 Service Quotas 사용 설명서의 API 할당량 증가 요청을 참조하세요.
- AWS 할당량 증가 요청을 10일 이내에 완료하는 것을 목표로 합니다. 하지만 몇 가지 고려 사항으로 인해 요청 처리 시간이 10일을 초과할 수 있습니다. 예를 들어 일부 요청의 경우 Amazon Cognito에

서 추가 하드웨어 용량을 프로비저닝해야 할 수 있으며, 요청 볼륨이 계절적으로 증가하면 지연이 발생할 수 있습니다.

- Service Quotas에서 아직 할당량을 사용할 수 없는 경우 서비스 한도 증가 양식을 사용합니다.

> **Important**
>
> 조정 가능한 할당량만 늘릴 수 있습니다. 증가된 할당량 용량을 구매해야 합니다. 할당량 인상 요금은 Amazon Cognito 요금을 참조하십시오.

Amazon Cognito 사용자 풀 API 작업 범주 및 요청 속도 할당량

Amazon Cognito에는 권한 부여 모델이 서로 다른 API 작업 클래스가 중복되므로, 각 작업은 단일 범주에 속합니다. 각 범주에는 계정의 단일 AWS 리전 에 있는 모든 사용자 풀에 대한 모든 멤버 API 작업에 대한 자체 풀링된 할당량이 있습니다. 조정 가능한 범주 할당량 증가만 요청할 수 있습니다. 자세한 정보는 할당량 증가 요청을 참조하세요. 할당량 조정은 단일 리전의 계정에 있는 사용자 풀에 적용됩니다. Amazon Cognito는 일부 범주[3]의 작업을 사용자 풀당 초당 요청(RPS) 5개로 제한합니다. 기본 할당량 (RPS) 은 추가로 모든 사용자 풀에 적용됩니다. AWS 계정

> **Note**
>
> 각 범주의 할당량은 월간 활성 사용자(MAU) 수로 측정됩니다. MAU가 200만 개 미만인 AWS 계정 는 기본 할당량 내에서 작동할 수 있습니다. MAU가 백만 개 미만이고 Amazon Cognito가 요청을 제한하고 있는 경우 앱 최적화를 고려해 보십시오. 자세한 정보는 할당량 한도에 대한 요청 비율을 최적화하세요.을 참조하세요.

범주 작업 할당량은 단일 AWS 리전내 모든 사용자 풀의 모든 사용자에게 적용됩니다. 또한 Amazon Cognito는 앱이 사용자 한 명에 대해 생성할 수 있는 요청 수에 대한 할당량을 유지합니다. 사용자별 API 요청을 다음 표와 같이 제한해야 합니다.

Amazon Cognito 사용자 풀 사용자별 요청률 할당량

Operation	사용자별 초당 연산
사용자 프로필 읽기	10

Operation	사용자별 초당 연산
예: `GetUser`, `GetDevice`	
사용자 프로필 쓰기 예: `UpdateUserAttributes` , `SetUserSettings`	10

범주별 API 요청을 다음 표와 같이 제한해야 합니다.

Amazon Cognito 사용자 풀 범주별 요청률 할당량

범주	설명	기본 할당량(RPS)	조정 가능
`UserAuthentication` • InitiateAuth • `InitiateAuth` 또는 Token 엔드포인트를 사용한 토큰 새로 고침 • RespondToAuthChallenge[1] • AdminInitiateAuth • AdminRespondToAuthChallenge[1] • 인증 코드 또는 암시적 권한 부여의 호스팅 UI 로그인 및 MFA[2]	사용자를 인증(로그인)하는 작업입니다. 이러한 작업에는 특별 요청 속도 처리가 포함된 Amazon Cognito 사용자 풀 API 작업 이(가) 적용됩니다.	120	예
`UserCreation` • SignUp	Amazon Cognito 로컬 사용자를 생성하거나	50	예

범주	설명	기본 할당량(RPS)	조정 가능
• ConfirmSignUp • AdminCreateUser • AdminConfirmSignUp	확인하는 작업입니다. Amazon Cognito 사용자 풀에서 직접 생성 및 검증하는 사용자입니다.		
`UserFederation` 서드 파티 ID 공급자를 사용하여 사용자를 Amazon Cognito 사용자 풀로 페더레이션(인증)하는 작업입니다.	사용자 풀 페더레이션 엔드포인트에 IdP 응답을 제출하는 작업입니다. IdP 토큰을 생성하는 OIDC 또는 소셜 제공업체 작업과 모든 SAML 요청이 이 할당량에 기여합니다.	25	예
`UserAccountRecovery` • ChangePassword • ConfirmForgotPassword • ForgotPassword • AdminResetUserPassword • AdminSetUserPassword • RespondToAuthChallenge[1] • AdminRespondToAuthChallenge[1] • 호스팅 UI 암호 재설정	사용자 계정을 복구하거나 사용자 암호를 변경 또는 업데이트하는 작업입니다.	30	아니요

범주	설명	기본 할당량(RPS)	조정 가능
`UserRead` • AdminGetUser • GetUser	사용자 풀에서 사용자를 검색하는 작업입니다.	120	예
`UserUpdate` • AdminAddUserToGroup • AdminDeleteUserAttributes • AdminUpdateUserAttributes • AdminDeleteUser • AdminDisableUser • AdminEnableUser • AdminLinkProviderForUser • AdminDisableProviderForUser • VerifyUserAttribute • DeleteUser • DeleteUserAttributes • UpdateUserAttributes • AdminUserGlobalSignOut • GlobalSignOut • AdminRemoveUserFromGroup	사용자 및 사용자 속성을 관리하는 데 사용하는 작업입니다.	25	아니요

범주	설명	기본 할당량(RPS)	조정 가능
`UserToken` • RevokeToken	토큰 관리 작업	120	예
`UserResourceRead` • AdminGetDevice • AdminListGroupsForUser • AdminListDevices • GetDevice • ListDevices • GetUserAttributeVerificationCode • ResendConfirmationCode • AdminListUserAuthEvents	Amazon Cognito에서 저장된 디바이스 또는 그룹 멤버십과 같은 사용자 리소스 정보를 검색하는 작업입니다.	50	예

범주	설명	기본 할당량(RPS)	조정 가능
`UserResourceUpdate` • AdminForgetDevice • AdminUpdateAuthEventFeedback • AdminSetUserMFA/레퍼런스 • AdminSetUserSettings • AdminUpdateDeviceStatus • UpdateDeviceStatus • UpdateAuthEventFeedback • ConfirmDevice • SetUserMFAP 레퍼런스 • SetUserSettings • VerifySoftwareToken • AssociateSoftwareToken • ForgetDevice	저장된 디바이스 또는 그룹 멤버십과 같은 사용자에 대한 리소스 정보를 업데이트하는 작업입니다.	25	아니요
`UserList` • ListUsers • ListUsersInGroup	사용자 목록을 반환하는 작업입니다.	30	아니요

범주	설명	기본 할당량(RPS)	조정 가능
`UserPoolRead` • DescribeUserPool • ListUserPools	사용자 풀을 읽는 작업입니다.	15	아니요
`UserPoolUpdate` • CreateUserPool • UpdateUserPool • DeleteUserPool	사용자 풀을 생성, 업데이트 또는 삭제하는 작업입니다.	15	아니요

범주	설명	기본 할당량(RPS)	조정 가능
`UserPoolResourceRead` • DescribeIdentityProvider • DescribeResourceServer • DescribeUserImportJob • DescribeUserPoolDomain • GetCSVHeader • GetGroup • GetSigningCertificate • GetIdentityProviderByIdentifier • GetUserPoolMfaConfig • ListGroups • ListIdentityProviders • ListResourceServers • ListTagsForResource • ListUserImportJobs • DescribeRiskConfiguration • GetUICustomization	사용자 풀에서 그룹 또는 리소스 서버와 같은 리소스에 대한 정보를 검색하는 작업입니다.[3]	20	아니요

범주	설명	기본 할당량(RPS)	조정 가능
`UserPoolResourceUpdate` • AddCustomAttributes • CreateGroup • CreateIdentityProvider • CreateResourceServer • CreateUserImportJob • CreateUserPoolDomain • DeleteGroup • DeleteIdentityProvider • DeleteResourceServer • DeleteUserPoolDomain • SetUserPoolMfaConfig • StartUserImportJob • StopUserImportJob • UpdateGroup • UpdateIdentityProvider • UpdateResourceServer • UpdateUserPoolDomain	사용자 풀에서 그룹 또는 리소스 서버와 같은 리소스를 수정하는 작업입니다.[3]	15	아니요

범주	설명	기본 할당량(RPS)	조정 가능
• SetRiskConfiguration • SetUICustomization • TagResource • UntagResource			
UserPoolClientRead • DescribeUserPoolClient • ListUserPoolClients	사용자 풀 클라이언트에 대한 정보를 검색하는 작업입니다.[3]	15	아니요
UserPoolClientUpdate • CreateUserPoolClient • DeleteUserPoolClient • UpdateUserPoolClient	사용자 풀 클라이언트를 생성, 업데이트 및 삭제하는 작업입니다.[3]	15	아니요
ClientAuthentication client_credentials 허용 유형이 토큰 엔드포인트에 요청합니다.	요청 승인에 사용할 자격 증명을 생성하는 작업 machine-to-machine	150	아니요

[1A] RespondToAuthChallenge 또는 a가 ChallengeName 포함된 AdminRespondToAuthChallenge NEW_PASSWORD_REQUIRED 응답은 범주에 포함됩니다. UserAccountRecovery 다른 모든 챌린지 응답은 UserAuthentication 카테고리에 포함됩니다.

[2] 로그인 시 호스팅된 각 UI 작업은 할당량에 요청 1개를 할당합니다. 예를 들어, 로그인하고 MFA 코드를 제공하는 사용자는 요청 2개를 사용합니다. 인증 코드 부여의 토큰 사용에는 해당 카테고리의 할당량과 동일한 요율로 추가 할당량이 할당됩니다. `UserAuthentication`

[3] 이 카테고리의 모든 개별 작업에는 단일 사용자 풀에 대해 5RPS를 초과하는 속도로 작업이 호출되지 않도록 하는 제약이 있습니다.

Amazon Cognito 자격 증명 풀(페더레이션 자격 증명) API연산 요청 속도 할당량

Operation	설명	기본 할당량 (RPS)[1]	조정 가능	할당량 증가 자격 여부
`GetId`	자격 증명 풀에서 자격 증명 ID를 검색합니다.	25	예	계정 팀에 문의하세요.
`GetOpenIdToken`	클래식 워크플로의 자격 증명 풀에서 OpenID 토큰을 검색합니다.	200	예	계정 팀에 문의하세요.
`GetCredentialsForIdentity`	향상된 워크플로의 AWS 자격 증명 풀에서 자격 증명을 검색합니다.	200	예	계정 팀에 문의하세요.
`GetOpenIdTokenForDeveloperIdentity`	개발자 워크플로의 자격 증명 풀에서 OpenID 토큰을 검색합니다.	50	예	계정 팀에 문의하세요.
`ListIdentities`	자격 증명 풀의 자격 증명 ID 목록을 검색합니다.	5	예	계정 팀에 문의하세요.

Operation	설명	기본 할당량 (RPS)[1]	조정 가능	할당량 증가 자격 여부
DeleteIdentities	자격 증명 풀에서 하나 이상의 등록된 자격 증명을 삭제합니다.	10	예	계정 팀에 문의하세요.
TagResource	자격 증명 풀에 태그를 적용합니다.	5	예	계정 팀에 문의하세요.
UntagResource	자격 증명 풀에서 태그를 제거합니다.	5	예	계정 팀에 문의하세요.
ListTagsForResource	자격 증명 풀에 적용된 태그 목록을 표시합니다.	10	예	계정 팀에 문의하세요.

[1] 기본 할당량은 모든 AWS 리전 자격 증명 풀의 최소 요청 속도 AWS 계정할당량입니다. 일부 리전에서는 RPS 할당량이 더 높을 수 있습니다.

리소스 수 및 크기에 대한 할당량

리소스 할당량은 Amazon Cognito의 리소스, 입력 필드, 기간 및 기타 기능의 최대 수 또는 크기입니다.

Service Quotas 콘솔 또는 서비스 한도 증가 양식에서 일부 리소스 할당량에 대한 조정을 요청할 수 있습니다. Service Quotas 콘솔에서 할당량을 요청하려면 Service Quotas 사용 설명서에서 할당량 증가 요청을 참조하세요. Service Quotas에서 아직 할당량을 사용할 수 없는 경우 서비스 한도 증가 양식을 사용합니다.

> **Note**
>
> 리전별 사용자 풀과 같은 AWS 계정 수준의 리소스 할당량은 각 레벨의 Amazon Cognito 리소스에 적용됩니다. AWS 리전예를 들어 미국 동부(버지니아 북부)에 사용자 풀 1,000개가 있고 유럽(스톡홀름)에 1,000개가 있을 수 있습니다.

다음 표에는 기본 리소스 할당량과 조정 가능 여부가 나와 있습니다.

Amazon Cognito 사용자 풀 리소스 할당량

Resource	할당량	조정 가능	최대 할당량
사용자 풀당 앱 클라이언트	1,000	예	10,000개
리전당 사용자 풀	1,000	예	10,000개
사용자 풀당 자격 증명 공급자	300	예	1,000
사용자 풀당 리소스 서버	25	예	300
사용자 풀당 사용자	40,000,000	예	계정 팀에 문의하세요.
사전 토큰 생성 Lambda 트리거의 변경 총합[1]	5,000	예	계정 팀에 문의하세요.
사용자 풀당 사용자 지정 속성	50	아니요	N/A
속성당 글자 수	2,048바이트	아니요	N/A
사용자 지정 속성 이름의 글자 수	20	아니요	N/A
암호 정책에서 필요한 최소 암호 글자 수	6~99	아니요	N/A
매일 전송되는 이메일 메시지 AWS 계정[2]	50	아니요	N/A
이메일 제목의 글자 수	140	아니요	N/A

Resource	할당량	조정 가능	최대 할당량
이메일 메시지의 글자 수	20,000건	아니요	N/A
SMS 확인 메시지의 글자 수	140	아니요	N/A
암호의 글자 수	256	아니요	N/A
자격 증명 공급자 이름의 글자 수	32	아니요	N/A
자격 증명 공급자당 식별자	50	아니요	N/A
사용자에 연결된 자격 증명	5	아니요	N/A
앱 클라이언트당 콜백 URL	100	아니요	N/A
앱 클라이언트당 로그아웃 URL	100	아니요	N/A
리소스 서버당 범위	100	아니요	N/A
앱 클라이언트당 범위	50	아니요	N/A
계정당 사용자 지정 도메인	4	아니요	N/A
각 사용자가 속할 수 있는 그룹	100	아니요	N/A
사용자 풀당 그룹	10,000개	아니요	N/A

[1] 이 할당량은 사전 토큰 생성 Lambda 트리거의 토큰에서 발생할 수 있습니다. 기존 클레임 및 추가된 클레임 수와 액세스 및 자격 증명 토큰의 범위를 더한 수를 합하면 이 할당량보다 작거나 같아야 합니다. 차단된 클레임 및 범위는 이 할당량에 포함되지 않습니다.

[2] 이 할당량은 Amazon Cognito 사용자 풀에 기본 이메일 기능을 사용하는 경우에만 적용됩니다. 더 많은 이메일 전송 볼륨을 활성화하려면 Amazon SES 이메일 구성을 사용하도록 사용자 풀을 구성합니다. 자세한 정보는 Amazon Cognito 사용자 풀에 대한 이메일 설정을 참조하세요.

Amazon Cognito 사용자 풀 세션 유효성 파라미터

토큰	할당량
ID 토큰	5분~1일
새로 고침 토큰	1시간~3,650일
액세스 토큰	5분~1일
호스트된 UI 세션 쿠키	1시간
인증 세션 토큰	3분~15분

Amazon Cognito 사용자 풀 코드 보안 리소스 할당량(조정 불가능)

Resource	할당량
가입 확인 코드 유효 기간	24시간
사용자 속성 확인 코드 유효 기간	24시간
다중 인증(MFA) 코드 유효 기간	3~15분
암호 찾기 코드 유효 기간	1시간
시간당 사용자별 최대 `ConfirmForgotPassword` 및 `ForgotPassword` 요청 수[1]	5~20

Resource	할당량
시간당 사용자별 최대 ResendConfirmationCode 요청 수	5
시간당 사용자별 최대 ConfirmSignUp 요청 수	15
시간당 사용자별 최대 ChangePassword 요청 수	5
시간당 사용자별 최대 GetUserAttributeVerificationCode 요청 수	5
시간당 사용자별 최대 VerifyUserAttribute 요청 수	15

[1] Amazon Cognito는 암호 업데이트 요청의 위험 요소를 평가하고 평가된 위험 수준에 맞는 할당량을 할당합니다. 자세한 정보는 암호 찾기 동작을 참조하세요.

Amazon Cognito 사용자 풀 사용자 가져오기 작업 리소스 할당량

Resource	할당량	조정 가능	최대 할당량
사용자 풀당 사용자 가져오기 작업	1,000	예	계정 팀에 문의하세요.
사용자 가져오기 CSV 행당 최대 문자 수	16,000	아니요	N/A
최대 CSV 파일 크기	100MB	아니요	N/A
CSV 파일당 최대 사용자 수	500,000	아니요	N/A

Amazon Cognito 자격 증명 풀(페더레이션 자격 증명) 리소스 할당량

Resource	할당량	조정 가능	최대 할당량
계정당 자격 증명 풀	1,000	예	N/A
자격 증명 풀당 Amazon Cognito 사용자 풀 공급자	50	예	1000
자격 증명 풀 이름에 대한 글자 길이	128B	아니요	N/A
로그인 공급자 이름에 대한 글자 길이	2,048바이트	아니요	N/A
자격 증명 풀당 자격 증명	무제한	아니요	N/A
역할 매핑을 지정할 수 있는 자격 증명 공급자	10	아니요	N/A
단일 list 또는 lookup 호출의 결과	60	아니요	N/A
역할 기반 액세스 제어(RBAC) 규칙	25	아니요	N/A

Amazon Cognito Sync 리소스 할당량

Resource	할당량	조정 가능	최대 할당량
자격 증명별 데이터 집합	20	예	계정 팀에 문의하세요.
데이터 집합당 레코드	1,024	예	계정 팀에 문의하세요.
단일 데이터 집합의 크기	1MB	예	계정 팀에 문의하세요.

Resource	할당량	조정 가능	최대 할당량
데이터 집합 이름의 글자 수	128바이트	아니요	N/A
요청 성공 후 대량 게시를 위한 대기 시간	24시간	아니요	N/A

Amazon Cognito API 및 엔드포인트 참조

다음 참조는 각 Amazon Cognito 기능의 서비스 엔드포인트를 설명합니다. Amazon Cognito 사용자 풀에는 사용자 풀 도메인이 있는 사용자 풀 엔드포인트와 사용자 풀 API라는 옵션이 있습니다. Amazon Cognito 사용자 풀 사용자 풀 API를 사용하는 API 작업 클래스에 대한 자세한 내용은 Amazon Cognito 사용자 풀 API 및 사용자 풀 엔드포인트 사용 섹션을 참조하세요.

AWS 리전별 사용자 풀 API의 서비스 엔드포인트 목록은 AWS 일반 참조의 서비스 엔드포인트를 참조하세요.

주제

- 사용자 풀 페더레이션 엔드포인트 및 호스팅 UI 참조
- Amazon Cognito 사용자 풀 API 참조
- Amazon Cognito 자격 증명 풀(페더레이션 자격 증명) API 참조
- Amazon Cognito Sync API 참조

사용자 풀 페더레이션 엔드포인트 및 호스팅 UI 참조

Amazon Cognito는 도메인을 사용자 풀에 할당할 때 여기에 나열된 공개 웹 페이지를 활성화합니다. 도메인은 모든 앱 클라이언트의 중앙 액세스 지점 역할을 합니다. 여기에는 사용자가 가입 및 로그인(Login 엔드포인트)하고 로그아웃(Logout 엔드포인트)할 수 있는 호스팅 UI가 포함됩니다. 이러한 리소스에 대한 자세한 내용은 Amazon Cognito 호스팅 UI 및 페더레이션 엔드포인트 설정 및 사용 섹션을 참조하세요.

이러한 페이지에는 사용자 풀이 타사 SAML, OpenID Connect (OIDC) 및 OAuth 2.0 ID 공급자 () 와 통신할 수 있도록 하는 공개 웹 리소스도 포함되어 있습니다. IdPs 페더레이션 ID 제공업체를 사용하여 사용자를 로그인하려면 사용자가 호스팅 UI Login 엔드포인트 또는 OIDC 권한 부여 엔드포인트에 대한 대화형 요청을 시작해야 합니다. 권한 부여 엔드포인트는 사용자를 호스팅 UI 또는 IdP 로그인 페이지로 리디렉션합니다.

앱은 Amazon Cognito 사용자 풀 API를 사용하여 로컬 사용자를 로그인할 수도 있습니다. 로컬 사용자는 외부 IdP를 통한 페더레이션 없이 사용자 풀 디렉터리에만 존재합니다.

호스팅된 UI 및 페더레이션 엔드포인트 외에도 Amazon Cognito는 안드로이드, iOS JavaScript 등용 SDK와 통합됩니다. SDK는 Amazon Cognito API 서비스 엔드포인트를 사용하여 사용자 풀 API 엔드포인트 작업을 수행할 수 있는 도구를 제공합니다. 서비스 엔드포인트에 대한 자세한 내용은 Amazon

Cognito Identity endpoints and quotas(Amazon Cognito 자격 증명 엔드포인트 및 할당량)를 참조하세요.

> **Warning**
>
> Amazon Cognito 도메인의 최종 개체 또는 중간 전송 계층 보안 (TLS) 인증서를 고정하지 마십시오. AWS 모든 사용자 풀 엔드포인트 및 접두사 도메인의 모든 인증서를 관리합니다. Amazon Cognito 인증서를 지원하는 신뢰 체인의 인증 기관(CA)은 동적으로 교체되고 갱신됩니다. 앱을 중간 또는 리프 인증서에 고정하면 인증서를 AWS 교체할 때 알림 없이 앱이 실패할 수 있습니다.
> 애플리케이션을 사용 가능한 Amazon 루트 인증서에 고정하세요. 자세한 내용은 AWS Certificate Manager 사용 설명서의 인증서 고정에서 모범 사례 및 권장 사항을 참조하세요.

주제

- 호스팅 UI 엔드포인트 참조
- OAuth 2.0, OpenID Connect, SAML 2.0 페더레이션 엔드포인트 참조
- OAuth 2.0 권한 부여
- Amazon Cognito 사용자 풀과 함께 권한 부여 코드 부여에 PKCE 사용
- 호스팅 UI 및 페더레이션 오류 응답

호스팅 UI 엔드포인트 참조

도메인을 사용자 풀에 추가하면 Amazon Cognito는 이 섹션의 호스팅 UI 엔드포인트를 활성화합니다. 사용자가 사용자 풀의 핵심 인증 작업을 완료할 수 있는 웹페이지입니다. 여기에는 암호 관리, 다중 인증(MFA), 속성 확인 페이지가 포함되어 있습니다. 호스팅 UI의 사용자 경험에 대한 자세한 내용은 호스팅 UI로 가입 및 로그인 섹션을 참조하세요.

호스팅 UI를 구성하는 웹 페이지는 고객과의 대화형 사용자 세션을 위한 프런트 엔드 웹 애플리케이션입니다. 앱은 사용자 브라우저에서 호스팅 UI를 호출해야 합니다. Amazon Cognito는 이 장의 웹 페이지에 대한 프로그래밍 방식의 액세스를 지원하지 않습니다. JSON 응답을 반환하는 OAuth 2.0, OpenID Connect, SAML 2.0 페더레이션 엔드포인트 참조의 페더레이션 엔드포인트는 앱 코드에서 직접 쿼리할 수 있습니다. 권한 부여 엔드포인트는 호스팅 UI 또는 IdP 로그인 페이지로 리디렉션되며 사용자 브라우저에서도 열어야 합니다.

이 안내서의 주제에서는 자주 사용하는 호스팅 UI 엔드포인트를 자세히 설명합니다. Amazon Cognito는 도메인을 사용자 풀에 할당할 때 다음 웹 페이지를 사용할 수 있도록 합니다.

호스팅 UI 엔드포인트

엔드포인트 URL	설명	액세스 방법
https://### # ###/login	사용자 풀 로컬 및 페더레이션 사용자를 로그인합니다.	권한 부여 엔드포인트, /logout 및 /confirmforgotPassword 같은 엔드포인트에서 리디렉션합니다. Login 엔드포인트를 참조하세요.
https://### # ###/logout	사용자 풀 사용자를 로그아웃합니다.	직접 링크. Logout 엔드포인트를 참조하세요.
https://### # ###/confirmUser	사용자 계정을 확인하기 위해 이메일 링크를 선택한 사용자를 확인합니다.	이메일 메시지에서 사용자가 선택한 링크.
https://### # ###/signup	새 사용자를 등록합니다. /login 페이지에서 사용자가 Sign up(가입)을 선택하면 /signup으로 이동하게 됩니다.	/oauth2/authorize 과 파라미터가 동일한 직접 링크.
https://### # ###/confirm	가입한 사용자에게 사용자 풀이 확인 코드를 보낸 후 코드를 입력하라는 메시지가 사용자에게 표시됩니다.	/signup에서 리디렉션만.
https://### # ###/forgotPassword	사용자 이름을 입력하라는 메시지를 표시하고 암호 재설정 코드를 전송합니다. /login 페이지에서 사용자가 Forgot your password?(암호가 생각나지 않는 경우)를 선택하면 /forgotPassword 로 이동하게 됩니다.	1. /login에 있는 암호 찾기 링크에서. 2. /oauth2/authorize 과 파라미터가 동일한 직접 링크.
https://### # ###/confirmforgotPassword	사용자에게 암호 재설정 코드와 새 암호를 입력하라	/forgotPassword 에서 리디렉션만.

엔드포인트 URL	설명	액세스 방법
	는 메시지가 표시됩니다. `/forgotPassword` 페이지에서 사용자가 Reset your password(암호 재설정)를 선택하면 `/confirmforgotPassword` 로 이동하게 됩니다.	
https://### # ###/resendcode	사용자 풀의 가입한 사용자에게 새 확인 코드를 보냅니다.	`/confirm`에 있는 새 코드 보내기 링크에서 리디렉션만.

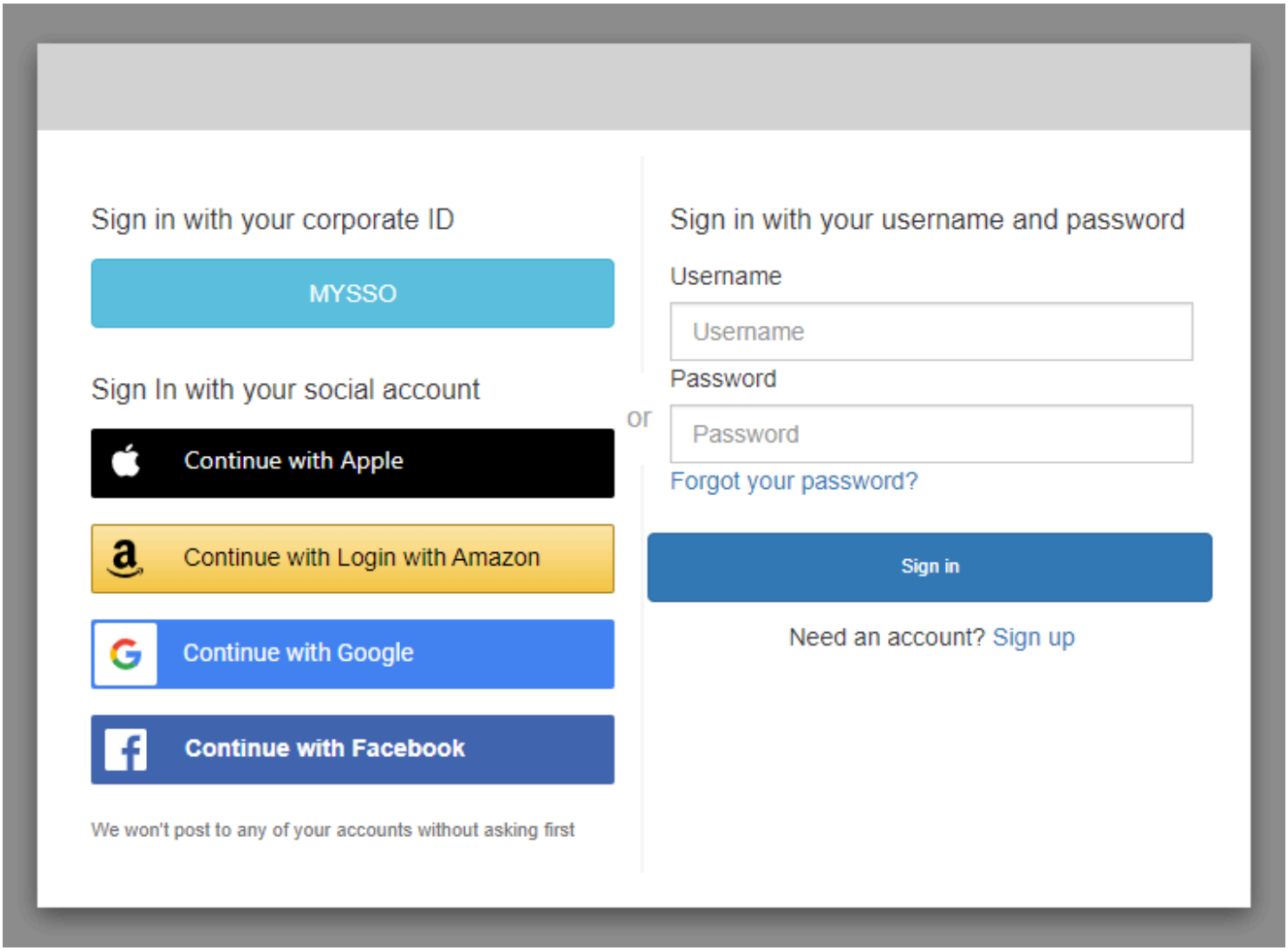
주제

- Login 엔드포인트
- Logout 엔드포인트

Login 엔드포인트

로그인 엔드포인트는 인증 서버이자 권한 부여 엔드포인트의 리디렉션 대상입니다. ID 제공업체를 지정하지 않는 경우 호스팅 UI의 진입점이 됩니다. 로그인 엔드포인트에 대한 리디렉션을 생성하는 경우, 로그인 엔드포인트는 로그인 페이지를 로드하고 클라이언트에 대해 구성된 인증 옵션을 사용자에게 표시합니다.

> **Note**
>
> 로그인 엔드포인트는 호스팅 UI의 구성 요소입니다. 앱에서 로그인 엔드포인트로 리디렉션되는 페더레이션 및 호스팅 UI 페이지를 호출합니다. 사용자가 로그인 엔드포인트에 직접 액세스하는 것은 모범 사례가 아닙니다.



/login 획득

`/login` 엔드포인트는 사용자의 초기 요청에서만 `HTTPS GET`을 지원합니다. 앱은 Chrome 또는 Firefox와 같은 브라우저에서 페이지를 호출합니다. `/login`에서 로 리디렉션하면 초기 권한 부여 엔드포인트 요청에서 제공한 모든 파라미터가 함께 전달됩니다. 로그인 엔드포인트는 권한 부여 엔드포인트의 모든 요청 파라미터를 지원합니다. 로그인 엔드포인트에 직접 액세스할 수도 있습니다. 가장 좋은 방법은 모든 사용자 세션을 `/oauth2/authorize`에서 시작하는 것입니다.

예 - 사용자에게 로그인하라는 메시지 표시

이 예제는 로그인 화면에 표시됩니다.

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/login?
                response_type=code&
                client_id=ad398u21ijw3s9w3939&
                redirect_uri=https://YOUR_APP/redirect_uri&
```

```
                state=STATE&
                scope=openid+profile+aws.cognito.signin.user.admin
```

예 - 응답

인증 서버는 권한 부여 코드 및 상태와 함께 사용자의 앱으로 리디렉션합니다. 이 서버는 코드 및 상태를 조각이 아닌 쿼리 문자열 파라미터로 반환해야 합니다.

```
HTTP/1.1 302 Found
                    Location: https://YOUR_APP/redirect_uri?
code=AUTHORIZATION_CODE&state=STATE
```

사용자가 시작한 로그인 요청

사용자는 /login 엔드포인트를 로드한 후 사용자 이름과 암호를 입력하고 로그인을 선택할 수 있습니다. 이렇게 하면 GET 요청과 동일한 헤더 요청 파라미터와 사용자 이름, 암호 및 디바이스 지문이 포함된 요청 본문을 갖는 HTTPS POST 요청이 생성됩니다.

Logout 엔드포인트

/logout 엔드포인트는 리디렉션 엔드포인트입니다. 사용자를 로그아웃시키고 앱 클라이언트의 승인된 로그아웃 URL 또는 엔드포인트로 리디렉션합니다. /login /logout 엔드포인트에 대한 GET 요청에서 사용 가능한 파라미터는 Amazon Cognito 호스팅 UI 사용 사례에 맞게 조정됩니다.

다시 로그인할 수 있도록 사용자를 호스팅 UI로 리디렉션하려면 요청에 redirect_uri 파라미터를 추가하세요. redirect_uri 파라미터가 있는 logout 요청에 는client_id, response_type, scope 등 Login 엔드포인트에 대한 후속 요청을 위한 파라미터도 포함되어야 합니다.

로그아웃 엔드포인트는 고객과의 대화형 사용자 세션을 위한 프런트 엔드 웹 애플리케이션입니다. 앱은 사용자 브라우저에서 이 엔드포인트와 다른 호스팅 UI 엔드포인트를 호출해야 합니다.

선택한 페이지로 사용자를 리디렉션하려면 허용된 로그아웃 URL을 앱 클라이언트에 추가하세요. logout 엔드포인트에 대한 사용자의 요청에 logout_uri 및 client_id 파라미터를 추가합니다. logout_uri 값이 앱 클라이언트의 허용된 로그아웃 URL 중 하나인 경우 Amazon Cognito는 사용자를 해당 URL로 리디렉션합니다.

SAML 2.0용 싱글 로그아웃 (SLO) 을 사용하면 Amazon IdPs Cognito는 먼저 사용자를 IdP 구성에서 정의한 SLO 엔드포인트로 리디렉션합니다. IdP가 사용자를 다시 리디렉션하면 Amazon saml2/

`logout` Cognito는 요청에서 OR로 한 번 더 리디렉션하여 응답합니다. `redirect_uri` `logout_uri` 자세한 정보는 SAML 로그아웃 플로우을 참조하세요.

로그아웃 엔드포인트는 OIDC 또는 소셜 ID 공급자 () 에서 사용자를 로그아웃시키지 않습니다. IdPs 외부 IdP를 사용한 세션에서 사용자를 로그아웃하려면 해당 공급자의 로그아웃 페이지로 안내하세요.

GET /logout

`/logout` 엔드포인트는 `HTTPS GET`만 지원합니다. 사용자 풀 클라이언트는 일반적으로 시스템 브라우저를 통해 이 요청을 수행합니다. 브라우저는 일반적으로 Android의 Custom Chrome Tab 또는 iOS의 Safari View Control입니다.

요청 파라미터

client_id

앱에 대한 앱 클라이언트 ID입니다. 앱 클라이언트 ID를 얻으려면 해당 앱을 사용자 풀에 등록해야 합니다. 자세한 내용은 사용자 풀 앱 클라이언트 섹션을 참조하세요.

필수 사항입니다.

logout_uri

logout_uri 파라미터가 있는 사용자 지정 로그아웃 페이지로 사용자를 리디렉션합니다. 로그아웃한 사용자를 리디렉션하려는 앱 클라이언트 로그아웃 URL로 값을 설정합니다. logout_uri는 client_id 파라미터와 함께만 사용합니다. 자세한 정보는 사용자 풀 앱 클라이언트을 참조하세요.

logout_uri 파라미터를 사용하여 사용자를 다른 앱 클라이언트의 로그인 페이지로 리디렉션할 수도 있습니다. 다른 앱 클라이언트의 로그인 페이지를 앱 클라이언트의 허용된 콜백 URL로 설정합니다. `/logout` 엔드포인트에 대한 요청에서 logout_uri 파라미터의 값을 URL 인코딩된 로그인 페이지로 설정합니다.

Amazon Cognito에서는 `/logout` 엔드포인트를 요청할 때 logout_uri 또는 redirect_uri 파라미터가 필요합니다. logout_uri 파라미터는 사용자를 다른 웹 사이트로 리디렉션합니다. logout_uri 및 redirect_uri 파라미터 모두가 `/logout` 엔드포인트에 대한 요청에 포함되어 있는 경우 Amazon Cognito는 logout_uri 파라미터만 사용하여 redirect_uri 파라미터를 재정의합니다.

redirect_uri

사용자를 로그인 페이지로 리디렉션하여 redirect_uri 파라미터를 사용하여 인증합니다. 다시 로그인한 사용자를 리디렉션하려는 앱 클라이언트 허용된 콜백 URL로 값을 설정합니다. `/login` 엔드포인트에 전달하려는 client_id, scope, state, response_type 파라미터를 추가합니다.

Amazon Cognito에서는 `/logout` 엔드포인트를 요청할 때 logout_uri 또는 redirect_uri 파라미터가 필요합니다. 사용자를 *`/login`* 엔드포인트로 리디렉션하여 재인증하고 앱에 토큰을 전달하려면 redirect_uri 매개변수를 추가하세요. logout_uri 파라미터와 redirect_uri 파라미터가 모두 엔드포인트에 대한 요청에 포함되어 있는 경우, */logout* Amazon Cognito는 redirect_uri 파라미터를 재정의하고 logout_uri 파라미터를 배타적으로 처리합니다.

response_type

사용자가 로그인한 후 Amazon Cognito로부터 수신할 OAuth 2.0 응답입니다. `code` 및 `token`은 response_type 파라미터에 대한 유효한 값입니다.

redirect_uri 파라미터를 사용하는 경우 필수입니다.

state

애플리케이션이 요청에 상태 파라미터를 추가하면 `/oauth2/logout` 엔드포인트가 사용자를 리디렉션할 때 Amazon Cognito가 해당 값을 앱에 반환합니다.

이 값을 요청에 추가하여 CSRF 공격으로부터 보호할 수 있습니다.

`state` 파라미터의 값을 URL 인코딩 JSON 문자열로 설정할 수 없습니다. 이 형식과 일치하는 문자열을 `state` 파라미터로 전달하려면 문자열을 base64로 인코딩한 다음 애플리케이션에서 디코딩하십시오.

redirect_uri 파라미터를 사용하는 경우 적극 권장됩니다.

scope

redirect_uri 파라미터를 사용하여 로그아웃한 후 Amazon Cognito로부터 요청할 OAuth 2.0 범위입니다. Amazon Cognito가 `/logout` 엔드포인트에 대한 요청에 scope 파라미터를 사용하여 사용자를 `/login` 엔드포인트로 리디렉션합니다.

redirect_uri 파라미터를 사용하는 경우에 선택 사항입니다. scope 파라미터를 포함하지 않는 경우 Amazon Cognito가 scope 파라미터를 사용하여 사용자를 `/login` 엔드포인트로 리디렉션합니다. Amazon Cognito가 사용자를 리디렉션하고 `scope`를 자동으로 채우는 경우 이 파라미터에 앱 클라이언트의 권한 부여된 모든 범위가 포함됩니다.

예제 요청

예 — 로그아웃하고 사용자를 클라이언트로 리디렉션

`logout_uri`및 `client_id` 를 제외하고 이 엔드포인트에 사용할 수 있는 모든 쿼리 매개변수는 로 전달됩니다. 권한 부여 엔드포인트 요청에 `logout_uri` 및 `client_id`가 포함된 경우 Amazon

Cognito는 다른 모든 요청 파라미터를 무시하고 사용자 세션을 `logout_uri`의 값에 속한 URL로 리디렉션합니다. 이 URL은 앱 클라이언트의 승인된 로그아웃 URL이어야 합니다.

다음은 로그아웃 및 `https://www.example.com/welcome` 리디렉션 요청의 예시입니다.

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/logout?
  client_id=1example23456789&
  logout_uri=https%3A%2F%2Fwww.example.com%2Fwelcome
```

예: 로그아웃하고 사용자에게 다른 사용자로 로그인하라는 메시지 표시

요청에 `logout_uri`가 없지만 승인 엔드포인트에 올바른 형식의 요청을 구성하는 파라미터를 제공하는 경우 Amazon Cognito는 사용자를 호스팅된 UI 로그인으로 리디렉션합니다. 로그아웃 엔드포인트는 원래 요청의 파라미터를 리디렉션 대상에 추가합니다. 로그아웃 엔드포인트에 대한 요청의 파라미터 `redirect_uri`는 로그아웃 URL이 아니라 승인 엔드포인트로 전달하려는 로그인 URL입니다.

다음은 사용자를 로그아웃하고, 로그인 페이지로 리디렉션하고, 로그인한 후 인증 코드를 제공하는 요청의 예입니다. `https://www.example.com`

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/logout?
  response_type=code&
  client_id=1example23456789&
  redirect_uri=https%3A%2F%2Fwww.example.com&
  state=example-state-value&
  nonce=example-nonce-value&
  scope=openid+profile+aws.cognito.signin.user.admin
```

OAuth 2.0, OpenID Connect, SAML 2.0 페더레이션 엔드포인트 참조

도메인을 사용자 풀에 추가하면 Amazon Cognito는 이 섹션의 엔드포인트를 활성화합니다. 페더레이션 엔드포인트는 사용자 대화형이 아닙니다. 쿠키는 앱이 타사 OAuth 2.0, OIDC 및 SAML 2.0 ID 제공자와 통신할 수 있도록 서비스 역할을 수행합니다 (). IdPs

이 안내서의 주제에서는 자주 사용되는 몇 가지 OAuth 2.0 및 OIDC 엔드포인트를 설명합니다. 도메인을 사용자 풀에 할당하면 Amazon Cognito는 다음과 같은 엔드포인트를 생성합니다.

사용자 풀 페더레이션 엔드포인트

엔드포인트 URL	설명	액세스 방법
https://*### # ###* /oauth2/authorize	사용자를 호스팅 UI로 리디렉션하거나 IdP를 사용하여 로그인합니다.	사용자 인증을 시작하기 위해 고객 브라우저에서 호출됩니다. 권한 부여 엔드포인트를 참조하세요.
https://*### # ###*/oauth2/token	인증 코드 또는 클라이언트 보안 인증 요청에 따라 토큰을 반환합니다.	앱에서 토큰을 검색하도록 요청했습니다. Token 엔드포인트를 참조하세요.
https://*### # ###*/oauth2/userInfo	액세스 토큰의 OAuth 2.0 범위 및 사용자 자격 증명을 기반으로 사용자 속성을 반환합니다.	앱에서 사용자 프로필을 검색하도록 요청했습니다. UserInfo 엔드포인트를 참조하세요.
https://*### # ###*/oauth2/revoke	새로 고침 토큰 및 연결된 액세스 토큰을 취소합니다.	앱에서 토큰 취소를 요청했습니다. 취소 엔드포인트를 참조하세요.
https://cognito-idp.*Region*.amazonaws.com/*your user pool ID*/.well-known/openid-configuration	사용자 풀의 OIDC 아키텍처 디렉터리입니다.	사용자 풀 발급자 메타데이터를 찾기 위해 앱에서 요청했습니다.
https://cognito-idp.*Region*.amazonaws.com/*your user pool ID*/.well-known/jwks.json	Amazon Cognito 토큰을 검증하는 데 사용할 수 있는 퍼블릭 키입니다.	JWT를 확인하기 위해 앱에서 요청했습니다.
https://*Your user pool domain*/oauth2/idpresponse	소셜 아이덴티티 제공업체는 권한 부여 코드를 통해 사용자를 이 엔드포인트로 리디렉션해야 합니다. Amazon Cognito는 페더레이션 사용자를 인증할 때 토큰의 코드를 사용합니다.	OIDC IdP 로그인에서 IdP 클라이언트 콜백 URL로 리디렉션됩니다.

엔드포인트 URL	설명	액세스 방법
https://*Your user pool domain*/saml2/idpresponse	SAML 2.0 ID 공급자와의 통합을 위한 어설션 소비자 응답 (ACS) URL.	SAML 2.0 IdP에서 ACS URL로 리디렉션되거나 IdP에서 시작한 로그인의 시작 지점으로 리디렉션됩니다. [1]
https://### 풀 도메인 / saml2/로그아웃	SAML 2.0 ID 공급자와의 통합을 위한 단일 로그아웃 (SLO) URL.	SAML 2.0 IdP에서 단일 로그아웃 (SLO) URL로 리디렉션되었습니다. POST 바인딩만 허용합니다.

[1] IdP에서 시작한 SAML 로그인에 대한 자세한 내용은 을 참조하십시오. IDP에서 시작한 SAML 로그인 사용

OpenID Connect 및 OAuth 표준에 대한 자세한 내용은 OpenID Connect 1.0 및 OAuth 2.0을 참조하세요.

주제

- 권한 부여 엔드포인트
- Token 엔드포인트
- UserInfo 엔드포인트
- 취소 엔드포인트
- saml2/idresponse 엔드포인트

권한 부여 엔드포인트

`/oauth2/authorize` 엔드포인트는 두 개의 리디렉션 대상을 지원하는 리디렉션 엔드포인트입니다. URL에 `identity_provider` 또는 `idp_identifier` 파라미터를 포함하면 사용자를 해당 ID 제공업체(IdP)의 로그인 페이지로 자동 리디렉션합니다. 그렇지 않으면 요청에 포함된 것과 동일한 URL 파라미터를 사용하여 Login 엔드포인트로 리디렉션됩니다.

권한 부여 엔드포인트는 호스팅 UI 또는 IdP 로그인 페이지로 리디렉션합니다. 이 엔드포인트에서 사용자 세션의 대상은 사용자가 브라우저에서 직접 상호 작용해야 하는 웹 페이지입니다.

권한 부여 엔드포인트를 사용하려면 사용자 풀에 다음 사용자 풀 세부 정보에 대한 정보를 제공하는 매개변수를 사용하여 `/oauth2/authorize`에서 사용자 브라우저를 호출하세요.

- 로그인할 앱 클라이언트입니다.
- 최종 콜백 URL입니다.
- 사용자의 액세스 토큰에서 요청할 OAuth 2.0 범위입니다.
- 필요에 따라 로그인하는 데 사용할 서드 파티 IdP입니다.

Amazon Cognito가 수신 클레임을 검증하는 데 사용하는 `state` 및 `nonce` 파라미터를 제공할 수도 있습니다.

GET **`/oauth2/authorize`**

`/oauth2/authorize` 엔드포인트는 `HTTPS GET`만 지원합니다. 대체로 앱은 사용자의 브라우저에서 이 요청을 시작합니다. HTTPS를 통해서만 `/oauth2/authorize` 엔드포인트에 요청할 수 있습니다.

권한 부여 엔드포인트에서 OpenID Connect(OIDC) 표준의 권한 부여 엔드포인트 정의에 대해 자세히 알아볼 수 있습니다.

요청 파라미터

`response_type`

(필수) 응답 유형. `code` 또는 `token`이어야 합니다.

`code`의 `response_type`이 있는 성공적인 요청은 권한 부여 코드 부여를 반환합니다. 권한 부여 코드 부여는 Amazon Cognito가 리디렉션 URL에 추가하는 `code` 파라미터입니다. 앱에서는 액세스, ID 및 새로 고침 토큰을 위해 Token 엔드포인트와 코드를 교환할 수 있습니다. 보안 모범 사례로 사용자를 위한 새로 고침 토큰을 받으려면 앱에서 권한 부여 코드 부여를 사용하세요.

`token`의 `response_type`이 있는 성공적인 요청은 암시적 권한 부여를 반환합니다. 암시적 권한 부여는 Amazon Cognito가 리디렉션 URL에 추가하는 ID 및 액세스 토큰입니다. 암시적 권한 부여는 토큰과 잠재적인 식별 정보를 사용자에게 노출하기 때문에 덜 안전합니다. 앱 클라이언트 구성에서 암시적 권한 부여에 대한 지원을 비활성화할 수 있습니다.

`client_id`

(필수) 앱 클라이언트 ID.

`client_id` 값은 요청한 사용자 풀에 있는 앱 클라이언트의 ID여야 합니다. 앱 클라이언트는 Amazon Cognito 로컬 사용자 또는 하나 이상의 서드 파티 IdP 로그인을 지원해야 합니다.

redirect_uri

(필수) Amazon Cognito가 사용자에게 권한을 부여한 후 인증 서버가 브라우저를 리디렉션하는 URL입니다.

리디렉션 URI(Uniform Resource Identifier)의 속성은 다음과 같아야 합니다.

- 절대 URI이어야 합니다.
- 클라이언트를 사용하여 URI를 미리 등록했어야 합니다.
- 여기에는 조각 구성 요소가 없어야 합니다.

OAuth 2.0 - Redirection Endpoint 섹션을 참조하세요.

Amazon Cognito를 사용하려면 리디렉션 URI에서 테스트 목적으로 콜백 URL로 설정할 수 있는 HTTPS를 사용해야 합니다(`http://localhost` 제외).

또한 Amazon Cognito는 `myapp://example`과 같은 앱 콜백 URL을 지원합니다.

state

(선택 사항, 권장) 앱이 요청에 상태 파라미터를 추가하면 `/oauth2/authorize` 엔드포인트가 사용자를 리디렉션할 때 Amazon Cognito가 해당 값을 앱에 반환합니다.

이 값을 요청에 추가하여 CSRF 공격으로부터 보호할 수 있습니다.

`state` 파라미터의 값을 URL 인코딩 JSON 문자열로 설정할 수 없습니다. 이 형식과 일치하는 문자열을 `state` 파라미터로 전달하려면 문자열을 base64로 인코딩한 다음 앱에서 디코딩하십시오.

identity_provider

(선택사항) 호스팅된 UI를 우회하고 사용자를 제공업체 로그인 페이지로 리디렉션하려면 이 매개변수를 추가합니다. identity_provider 파라미터의 값은 사용자 프로필에 나타나는 대로 자격 증명 공급자(IdP)의 이름입니다.

- 소셜 공급자의 경우 identity_provider 값, 및 를 사용할 수 있습니다. `Facebook` `Google` `LoginWithAmazon` `SignInWithApple`
- Amazon Cognito 사용자 풀의 경우 값을 사용하십시오. `COGNITO`
- SAML 2.0 및 OpenID Connect (OIDC) ID 공급자 (IdPs) 의 경우 사용자 풀의 IdP에 할당한 이름을 사용합니다.

idp_identifier

(선택 사항) identity_provider 이름의 대체 이름을 사용하여 공급자로 리디렉션하려면 이 매개 변수를 추가합니다. Amazon Cognito 콘솔의 로그인 경험 탭에서 SAML 2.0 및 IdPs OIDC에 대한 식별자를 입력할 수 있습니다.

scope

(선택 사항) 모든 시스템 예약 범위 또는 클라이언트와 연결된 사용자 지정 범위를 조합할 수 있습니다. 범위는 공백으로 구분해야 합니다. 시스템에 예약된 범위로는 `openid`, `email`, `phone`, `profile` 및 `aws.cognito.signin.user.admin`이 있습니다. 사용된 범위는 클라이언트와 연결되어 있어야 합니다. 그렇지 않으면 런타임 시 무시됩니다.

클라이언트가 범위를 요청하지 않은 경우 인증 서버에서는 클라이언트와 연결된 모든 범위를 사용합니다.

`openid` 범위가 요청될 경우에만 ID 토큰이 반환됩니다. `aws.cognito.signin.user.admin` 범위가 요청된 경우에만 Amazon Cognito 사용자 풀에 대해 액세스 토큰을 사용할 수 있습니다. `phone` 범위도 요청된 경우에만 `email`, `profile` 및 `openid` 범위를 요청할 수 있습니다. 이러한 범위는 ID 토큰 내부로 들어가는 클레임을 지정합니다.

code_challenge_method

(선택 사항) 챌린지를 생성하는 데 사용한 해싱 프로토콜. PKCE RFC는 S256 및 일반의 두 가지 메서드를 정의하지만 Amazon Cognito 인증 서버는 S256만 지원합니다.

code_challenge

(선택 사항) 에서 생성한 챌린지. `code_verifier`

`code_challenge_method` 파라미터를 지정하는 경우에만 필수입니다.

nonce

(선택 사항) 요청에 추가할 수 있는 임의의 값입니다. 제공한 임시 값은 Amazon Cognito가 발행하는 ID 토큰에 포함되어 있습니다. 재생 공격을 방지하기 위해 앱은 ID 토큰의 `nonce` 클레임을 검사하고 생성한 것과 비교할 수 있습니다. `nonce` 클레임에 대한 자세한 내용은 OpenID Connect 표준의 ID 토큰 유효성 검사를 참조하세요.

긍정적인 응답이 있는 요청의 예

다음 예는 `/oauth2/authorize` 엔드포인트에 대한 HTTP 요청의 형식을 보여줍니다.

인증 코드 권한 부여

다음은 인증 코드 부여에 대한 요청의 예입니다.

예 — GET 요청

다음 요청은 사용자가 `redirect_uri` 목적지의 앱에 전달하는 인증 코드를 검색하는 세션을 시작합니다. 이 세션에서는 사용자 속성에 대한 범위와 Amazon Cognito 셀프 서비스 API 작업에 대한 액세스 범위를 요청합니다.

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&
client_id=1example23456789&
redirect_uri=https://www.example.com&
state=abcdefg&
scope=openid+profile+aws.cognito.signin.user.admin
```

예 — 응답

Amazon Cognito 인증 서버는 권한 부여 코드 및 상태를 통해 앱으로 다시 리디렉션합니다. 인증 코드는 5분 동안 유효합니다.

```
HTTP/1.1 302 Found
Location: https://www.example.com?code=a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111&state=abcdefg
```

PKCE를 통한 인증 코드 권한 부여

다음은 PKCE의 인증 코드 부여 요청 예시입니다.

예 — GET 요청

다음 요청은 이전 요청에 `code_challenge` 매개변수를 추가합니다. 코드를 토큰으로 교환하려면 `/oauth2/token` 엔드포인트에 대한 요청에 `code_verifier` 파라미터를 포함해야 합니다.

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&
client_id=1example23456789&
redirect_uri=https://www.example.com&
state=abcdefg&
scope=aws.cognito.signin.user.admin&
code_challenge_method=S256&
```

```
code_challenge=a1b2c3d4...
```

예 — 응답

인증 서버는 인증 코드 및 상태를 사용하여 애플리케이션으로 다시 리디렉션합니다. 코드와 상태는 프래그먼트가 아닌 쿼리 문자열 파라미터에 반환되어야 합니다.

```
HTTP/1.1 302 Found
Location: https://www.example.com?code=a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111&state=abcdefg
```

openid 범위가 없는 토큰 부여

다음은 암시적 승인을 생성하고 JWT를 사용자 세션에 직접 반환하는 예제 요청입니다.

예 — GET 요청

다음은 권한 부여 서버의 암시적 권한 부여에 대한 요청입니다. Amazon Cognito의 액세스 토큰은 셀프 서비스 API 작업을 승인합니다.

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=token&
client_id=1example23456789&
redirect_uri=https://www.example.com&
state=abcdefg&
scope=aws.cognito.signin.user.admin
```

예 — 응답

Amazon Cognito 권한 부여 서버는 액세스 토큰을 통해 앱으로 다시 리디렉션합니다. openid 범위가 요청되지 않았기 때문에 Amazon Cognito에서 ID 토큰을 반환하지 않습니다. 또한 Amazon Cognito는 이 흐름에서 새로 고침 토큰을 반환하지 않습니다. Amazon Cognito는 쿼리 문자열이 아닌 프래그먼트에서 액세스 토큰과 상태를 반환합니다.

```
HTTP/1.1 302 Found
Location: https://YOUR_APP/
redirect_uri#access_token=ACCESS_TOKEN&token_type=bearer&expires_in=3600&state=STATE
```

openid 범위가 있는 토큰 부여

다음은 암시적 권한을 생성하고 JWT를 사용자 세션에 직접 반환하는 예제 요청입니다.

예 — GET 요청

다음은 권한 부여 서버의 암시적 권한 부여에 대한 요청입니다. Amazon Cognito의 액세스 토큰은 사용자 속성 및 셀프 서비스 API 작업에 대한 액세스를 승인합니다.

```
GET
https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=token&
client_id=1example23456789&
redirect_uri=https://www.example.com&
state=abcdefg&
scope=aws.cognito.signin.user.admin+openid+profile
```

예 — 응답

권한 부여 서버는 액세스 토큰과 ID 토큰을 사용하여 앱으로 다시 리디렉션합니다 (openid범위가 포함되었기 때문).

```
HTTP/1.1 302 Found
Location: https://
www.example.com#id_token=eyJra67890EXAMPLE&access_token=eyJra12345EXAMPLE&token_type=bearer&exp
```

부정 응답 예제

Amazon Cognito는 요청을 거부할 수 있습니다. 부정적인 요청에는 요청 파라미터를 수정하는 데 사용할 수 있는 HTTP 오류 코드와 설명이 함께 제공됩니다. 다음은 부정적인 응답의 예입니다.

- client_id및 redirect_uri 가 유효하지만 요청 매개 변수의 형식이 올바르지 않은 경우 인증 서버는 오류를 클라이언트의 서버로 redirect_uri 리디렉션하고 URL 매개 변수에 오류 메시지를 추가합니다. 다음은 잘못된 형식의 예입니다.
  - 요청에는 response_type 매개변수가 포함되어 있지 않습니다.
  - 권한 부여 요청에서 code_challenge 매개변수는 제공했지만 code_challenge_method 매개변수는 제공하지 않았습니다.
  - code_challenge_method매개변수 값은 그렇지 않습니다S256.

  다음은 잘못된 형식의 예제 요청에 대한 응답입니다.

  ```
  HTTP 1.1 302 Found Location: https://client_redirect_uri?error=invalid_request
  ```

- 클라이언트가 요청했지만 이러한 요청에 code 대한 권한이 없는 경우 Amazon Cognito 권한 부여 서버는 다음과 같이 클라이언트의 redirect_uri 서버로 돌아갑니다unauthorized_client. token response_type

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?error=unauthorized_client
```

- 클라이언트가 알 수 없거나, 형식이 잘못되었거나, 유효하지 않은 범위를 요청한 경우 Amazon Cognito 권한 부여 서버에서 다음과 같이 invalid_scope를 클라이언트의 redirect_uri에 반환합니다.

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?error=invalid_scope
```

- 서버에 예상치 못한 오류가 발생하는 경우 인증 서버는 클라이언트의 redirect_uri 서버로 server_error 돌아갑니다. HTTP 500 오류가 클라이언트에 전송되지 않기 때문에 오류가 사용자 브라우저에 표시되지 않습니다. 권한 부여 서버가 다음 오류를 반환합니다.

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?error=server_error
```

- Amazon Cognito가 제3자와의 페더레이션을 통해 인증하는 경우 Amazon IdPs Cognito에서 다음과 같은 연결 문제가 발생할 수 있습니다.
  - IdP에게 토큰을 요청하는 동안 연결 제한 시간이 발생하면 인증 서버가 다음과 같이 오류를 클라이언트의 redirect_uri로 리디렉션합니다.

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?
error=invalid_request&error_description=Timeout+occurred+in+calling+IdP+token
+endpoint
```

  - ID 토큰 검증을 위해 jwks_uri 엔드포인트를 호출하는 동안 연결 제한 시간이 초과되면 인증 서버가 오류와 함께 다음과 같이 클라이언트의 서버로 리디렉션합니다. redirect_uri

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?
error=invalid_request&error_description=error_description=Timeout+in+calling+jwks
+uri
```

- 타사와 페더레이션하여 인증하는 경우 공급자가 오류 응답을 IdPs 반환할 수 있습니다. 이는 구성 오류나 다음과 같은 기타 이유 때문일 수 있습니다.
  - 다른 공급자로부터 오류 응답이 수신되면 인증 서버가 다음과 같이 오류를 클라이언트의 redirect_uri로 리디렉션합니다.

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?
error=invalid_request&error_description=[IdP name]+Error+-+[status code]+error
 getting token
```

- Google로부터 오류 응답이 수신되면 인증 서버가 다음과 같이 오류를 클라이언트의 `redirect_uri`로 리디렉션합니다.

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?
error=invalid_request&error_description=Google+Error+-+[status code]+[Google-
provided error code]
```

- Amazon Cognito에서 외부 IdP에 연결할 때 통신 예외가 발생하면 인증 서버가 오류와 함께 다음 메시지 중 하나를 사용하여 클라이언트의 `redirect_uri` 서버로 리디렉션합니다.
  - ```
    HTTP 1.1 302 Found Location: https://client_redirect_uri?
    error=invalid_request&error_description=Connection+reset
    ```
  - ```
    HTTP 1.1 302 Found Location: https://client_redirect_uri?
    error=invalid_request&error_description=Read+timed+out
    ```

Token 엔드포인트

`/oauth2/token`의 OAuth 2.0 토큰 엔드포인트는 JSON 웹 토큰(JWT)을 발급합니다.

사용자 풀 OAuth 2.0 인증 서버는 토큰 엔드포인트에서 다음 유형의 세션으로 JSON 웹 토큰 (JWT) 을 발급합니다.

1. 인증 코드 부여 요청을 완료한 사용자. 코드를 성공적으로 사용하면 ID, 액세스 및 새로 고침 토큰이 반환됩니다.
2. 클라이언트 자격 증명 부여를 완료한 M achine-to-machine (M2M) 세션. 클라이언트 암호로 인증에 성공하면 액세스 토큰이 반환됩니다.
3. 이전에 로그인하여 새로 고침 토큰을 받은 사용자. 새로 고침 토큰 인증은 새 ID 및 액세스 토큰을 반환합니다.

> **Note**
>
> 호스팅된 UI에서 또는 페더레이션을 통해 인증 코드 부여로 로그인한 사용자는 언제든지 토큰 엔드포인트에서 토큰을 새로 고칠 수 있습니다. API 작업을 `InitiateAuth` 통

해 로그인한 사용자는 사용자 풀에서 기억된 디바이스가 활성 상태가 아닐 때 토큰 엔드포인트를 사용하여 토큰을 새로 고칠 AdminInitiateAuth 수 있습니다. 기억된 디바이스가 활성 상태인 경우 AuthFlow of REFRESH_TOKEN_AUTH in InitiateAuth 또는 AdminInitiateAuth API 요청으로 토큰을 새로 고치세요.

도메인을 사용자 풀에 추가하면 토큰 엔드포인트를 공개적으로 사용할 수 있게 됩니다. 토큰 엔드포인트는 HTTP POST 요청을 수락합니다. 애플리케이션 보안을 위해 PKCE를 인증 코드 로그인 이벤트와 함께 사용하십시오. PKCE는 인증 코드를 전달하는 사용자가 인증을 받은 사용자와 동일한지 확인합니다. PKCE에 대한 자세한 내용은 IETF RFC 7636을 참조하십시오.

사용자 풀 앱 클라이언트와 해당 권한 부여 유형, 클라이언트 암호, 승인된 범위 및 클라이언트 ID에 대한 자세한 내용은 사용자 풀 앱 클라이언트에서 확인할 수 있습니다. 에서 M2M 인증, 클라이언트 자격 증명 부여, 액세스 토큰 범위를 통한 권한 부여에 대해 자세히 알아볼 수 있습니다. 리소스 서버를 통한 범위, M2M 및 API 인증

액세스 토큰에서 사용자에 대한 정보를 검색하려면 이를 사용자 UserInfo 엔드포인트 또는 API 요청으로 GetUser전달하십시오.

POST /oauth2/token

/oauth2/token 엔드포인트는 HTTPS POST만 지원합니다. 앱은 사용자 브라우저를 통하지 않고 직접 이 엔드포인트에 요청을 수행합니다.

토큰 엔드포인트는 client_secret_basic 및 client_secret_post 인증을 지원합니다. OpenID Connect 사양에 대한 자세한 내용은 클라이언트 인증을 참조하십시오. OpenID Connect 사양의 Token 엔드포인트에 대한 자세한 내용은 토큰 엔드포인트를 참조하세요.

헤더의 요청 파라미터

Authorization

클라이언트에 암호가 발급된 경우 클라이언트는 권한 부여 헤더에서 client_secret_basic HTTP 권한 부여로 해당 client_id 및 client_secret을 전달할 수 있습니다. 요청 본문에 client_secret_post 권한 부여로 client_id 및 client_secret을 포함할 수도 있습니다.

인증 헤더 문자열은 기본 Base64Encode(client_id:client_secret) 입니다. 다음 예는 djc98u3jiedmi283eu928 Base64로 인코딩된 버전의 문자열을 사용하는 클라이언트 abcdef01234567890 암호가 있는 앱 클라이언트의 인증 헤더입니다.
djc98u3jiedmi283eu928:abcdef01234567890

```
Authorization: Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw
```

Content-Type

이 파라미터의 값을 'application/x-www-form-urlencoded'으로 설정합니다.

본문의 요청 파라미터

grant_type

(필수) 요청하려는 OIDC 권한 유형.

authorization_code, refresh_token 또는 client_credentials가 있습니다. 다음 조건에서 토큰 엔드포인트에서 사용자 지정 범위에 대한 액세스 토큰을 요청할 수 있습니다.

- 앱 클라이언트 구성에서 요청된 범위를 활성화했습니다.
- 클라이언트 비밀번호로 앱 클라이언트를 구성했습니다.
- 앱 클라이언트에서 클라이언트 자격 증명 부여를 활성화합니다.

client_id

(선택 사항) 사용자 풀에 있는 앱 클라이언트의 ID. 사용자를 인증한 것과 동일한 앱 클라이언트를 지정하십시오.

클라이언트가 공개되어 있고 비밀이 없거나 client_secret_post 권한이 있는 경우 이 매개변수를 제공해야 합니다. client_secret

client_secret

(선택 사항) 사용자를 인증한 앱 클라이언트의 클라이언트 비밀번호입니다. 앱 클라이언트에 클라이언트 암호가 있고 Authorization 헤더를 보내지 않은 경우 필수입니다.

scope

(선택사항) 앱 클라이언트와 연결된 모든 사용자 지정 범위의 조합일 수 있습니다. 요청하는 모든 범위는 앱 클라이언트에 대해 활성화되어야 합니다. 그렇지 않으면 Amazon Cognito는 이를 무시합니다. 클라이언트가 범위를 요청하지 않는 경우 인증 서버는 앱 클라이언트 구성에서 승인한 모든 사용자 지정 범위를 할당합니다.

grant_type이 client_credentials인 경우에만 사용됩니다.

redirect_uri

(선택 사항) 로그인할 때 사용한 redirect_uri 것과 같아야 합니다. authorization_code / oauth2/authorize

있는 경우 grant_type 이 매개변수를 제공해야 authorization_code 합니다.

refresh_token

(선택 사항) 사용자 세션에 대한 새 액세스 및 ID 토큰을 생성하려면 /oauth2/token 요청의 refresh_token 매개변수 값을 동일한 앱 클라이언트에서 이전에 발급한 새로 고침 토큰으로 설정합니다.

code

(선택 사항) 승인 코드 부여의 인증 코드. 승인 요청에 다음 grant_type 중 하나가 포함된 경우 이 매개변수를 제공해야 authorization_code 합니다.

code_verifier

(선택 사항) PKCE의 승인 코드 부여 code_challenge 요청에서 계산하는 데 사용한 임의 값입니다.

긍정적인 응답이 있는 요청의 예

인증 코드를 토큰으로 교환

예 — POST 요청

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token&
                            Content-Type='application/x-www-form-urlencoded'&

 Authorization=Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw

                            grant_type=authorization_code&
                            client_id=1example23456789&
                            code=AUTHORIZATION_CODE&
                            redirect_uri=com.myclientapp://myclient/redirect
```

예제 — 응답

```
HTTP/1.1 200 OK
```

```
                            Content-Type: application/json

                            {
                             "access_token":"eyJra1example",
                             "id_token":"eyJra2example",
                             "refresh_token":"eyJj3example",
                             "token_type":"Bearer",
                             "expires_in":3600
                            }
```

> **Note**
>
> 토큰 엔드포인트는 `grant_type`이 `authorization_code`인 경우에만 `refresh_token`을 반환됩니다.

클라이언트 보안 인증 정보를 액세스 토큰으로 교환: 권한 부여 헤더의 클라이언트 암호

예 — POST 요청

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token >
                            Content-Type='application/x-www-form-urlencoded'&

 Authorization=Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw

                            grant_type=client_credentials&
                            client_id=1example23456789&

 scope=resourceServerIdentifier1/scope1 resourceServerIdentifier2/scope2
```

예제 — 응답

```
HTTP/1.1 200 OK
                            Content-Type: application/json

                            {
                            "access_token":"eyJra1example",
                            "token_type":"Bearer",
                            "expires_in":3600
                            }
```

클라이언트 보안 인증 정보를 액세스 토큰으로 교환: 요청 본문의 클라이언트 암호

예 — POST 요청

```
POST /oauth2/token HTTP/1.1
Content-Type: application/x-www-form-urlencoded
X-Amz-Target: AWSCognitoIdentityProviderService.Client credentials request
User-Agent: USER_AGENT
Accept: /
Accept-Encoding: gzip, deflate, br
Content-Length: 177
Referer: http://auth.example.com/oauth2/token
Host: auth.example.com
Connection: keep-alive

grant_type=client_credentials&client_id=1example23456789&scope=my_resource_server_identifier%2F
```

예제 — 응답

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Tue, 05 Dec 2023 16:11:11 GMT
x-amz-cognito-request-id: 829f4fe2-a1ee-476e-b834-5cd85c03373b

{
"access_token": "eyJra12345EXAMPLE",
"expires_in": 3600,
"token_type": "Bearer"
}
```

PKCE를 통한 인증 코드 권한 부여를 토큰으로 교환

예 — POST 요청

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token
                                Content-Type='application/x-www-form-urlencoded'&

 Authorization=Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw

                                grant_type=authorization_code&
```

```
                                client_id=1example23456789&
                                code=AUTHORIZATION_CODE&
                                code_verifier=CODE_VERIFIER&
                                redirect_uri=com.myclientapp://myclient/redirect
```

예제 — 응답

```
HTTP/1.1 200 OK
                               Content-Type: application/json

                               {
                               "access_token":"eyJra1example",
                               "id_token":"eyJra2example",
                               "refresh_token":"eyJj3example",
                               "token_type":"Bearer",
                               "expires_in":3600
                               }
```

> **Note**
>
> 토큰 엔드포인트는 `grant_type`이 `authorization_code`인 경우에만 `refresh_token`을 반환됩니다.

새로 고침 토큰을 토큰으로 교환

예 — POST 요청

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token >
                              Content-Type='application/x-www-form-urlencoded'&

 Authorization=Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw

                              grant_type=refresh_token&
                              client_id=1example23456789&
                              refresh_token=eyJj3example
```

예제 — 응답

```
HTTP/1.1 200 OK
                            Content-Type: application/json

                            {
                            "access_token":"eyJra1example",
                            "id_token":"eyJra2example",
                            "token_type":"Bearer",
                            "expires_in":3600
                                  }
```

> **Note**
>
> 토큰 엔드포인트는 `grant_type`이 `authorization_code`인 경우에만 `refresh_token`을 반환됩니다.

부정 응답 예제

예 — 오류 응답

```
HTTP/1.1 400 Bad Request
                            Content-Type: application/json;charset=UTF-8

                            {
                            "error":"invalid_request|invalid_client|invalid_grant|
unauthorized_client|unsupported_grant_type"
                            }
```

`invalid_request`

요청에 필수 파라미터가 누락되거나, 지원되지 않는 파라미터 값(`unsupported_grant_type` 아님)이 포함되거나, 잘못되었습니다. 예를 들어, `grant_type`이 `refresh_token`이지만 `refresh_token`이 포함되어 있지 않습니다.

`invalid_client`

클라이언트 인증에 실패했습니다. 예를 들어, 클라이언트가 권한 부여 헤더에 `client_id` 및 `client_secret`을 포함하지만 `client_id` 및 `client_secret`이 있는 클라이언트가 없는 경우입니다.

invalid_grant

새로 고침 토큰이 취소되었습니다.

권한 부여 코드를 이미 사용했거나 해당 코드가 존재하지 않습니다.

앱 클라이언트에는 요청한 범위의 일부 속성에 대한 읽기 액세스 권한이 없습니다. 예를 들어, 앱이 `email` 범위를 요청하면 앱 클라이언트는 `email` 속성을 읽을 수 있지만, `email_verified` 속성은 읽을 수 없습니다.

unauthorized_client

클라이언트가 코드 부여 흐름이나 토큰 새로 고침에 대해 허용되지 않습니다.

unsupported_grant_type

`grant_type`이 `authorization_code` 또는 `refresh_token` 또는 `client_credentials` 이외의 것인 경우 반환됩니다.

UserInfo 엔드포인트

`userInfo` 엔드포인트는 OpenID Connect(OIDC) userInfo 엔드포인트입니다. 이 엔드포인트는 서비스 공급자가 Token 엔드포인트에서 발급한 액세스 토큰을 제시하면 사용자 속성을 이용해 응답합니다. 사용자 액세스 토큰의 범위는 userInfo 엔드포인트가 응답에서 반환하는 사용자 특성을 정의합니다. `openid` 범위는 액세스 토큰 클레임 중 하나여야 합니다.

Amazon Cognito는 InitiateAuth 같은 사용자 풀 API 요청에 대한 응답으로 액세스 토큰을 발급합니다. 범위가 없기 때문에 userInfo 엔드포인트는 이러한 액세스 토큰을 수락하지 않습니다. 대신 토큰 엔드포인트에서 액세스 토큰을 제시해야 합니다.

OAuth 2.0 타사 ID 제공업체(IdP)도 userInfo 엔드포인트를 호스팅합니다. 사용자가 해당 IdP로 인증하면 Amazon Cognito는 자동으로 IdP 엔드포인트와 인증 코드를 교환합니다. `token` 사용자 풀은 IdP 액세스 토큰을 전달하여 IdP 엔드포인트에서 사용자 정보 검색을 승인합니다. `userInfo`

GET /oauth2/userInfo

앱은 브라우저를 통하지 않고 직접 이 엔드포인트에 요청합니다.

자세한 내용은 OpenID Connect(OIDC) 사양의 UserInfoEndpoint(엔드포인트)를 참조하세요.

주제

- 헤더의 요청 파라미터

- 예 — 요청
- 예 — 긍정적인 응답
- 부정적인 반응의 예

헤더의 요청 파라미터

Authorization: Bearer *<access_token>*

권한 부여 헤더 필드에 액세스 토큰을 전달하십시오.

필수 사항입니다.

예 — 요청

```
GET /oauth2/userInfo HTTP/1.1
Content-Type: application/x-amz-json-1.1
Authorization: Bearer eyJra12345EXAMPLE
User-Agent: [User agent]
Accept: */*
Host: auth.example.com
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

예 — 긍정적인 응답

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: [Integer]
Date: [Timestamp]
x-amz-cognito-request-id: [UUID]
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
X-Frame-Options: DENY
Server: Server
Connection: keep-alive
{
```

```
    "sub": "[UUID]",
    "email_verified": "true",
    "custom:mycustom1": "CustomValue",
    "phone_number_verified": "true",
    "phone_number": "+12065551212",
    "email": "bob@example.com",
    "username": "bob"
}
```

OIDC 신청 목록은 표준 신청을 참조하세요. 현재 Amazon Cognito는 `email_verified` 및 `phone_number_verified`의 값을 문자열로 반환합니다.

부정적인 반응의 예

예 — 잘못된 요청

```
HTTP/1.1 400 Bad Request
WWW-Authenticate: error="invalid_request",
error_description="Bad OAuth2 request at UserInfo Endpoint"
```

`invalid_request`

요청에 필수 매개변수가 누락되었거나, 지원되지 않는 매개변수 값이 포함되어 있거나, 형식이 잘못되었습니다.

예 — 잘못된 토큰

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: error="invalid_token",
error_description="Access token is expired, disabled, or deleted, or the user has
 globally signed out."
```

`invalid_token`

액세스 토큰이 만료, 취소, 형식이 잘못되었거나 유효하지 않습니다.

취소 엔드포인트

`/oauth2/revoke`엔드포인트는 Amazon Cognito에서 사용자가 제공한 새로 고침 토큰으로 처음 발급한 사용자의 액세스 토큰을 취소합니다. 또한 이 엔드포인트는 동일한 새로 고침 토큰에서 이후의 모든

액세스 및 자격 증명 토큰을 취소합니다. 엔드포인트가 토큰을 취소한 후에는 취소된 액세스 토큰을 사용하여 Amazon Cognito 토큰이 인증하는 API에 액세스할 수 없습니다.

POST /oauth2/revoke

`/oauth2/revoke` 엔드포인트는 HTTPS `POST`만 지원합니다. 사용자 풀 클라이언트는 시스템 브라우저를 통해서가 아닌 직접 이 엔드포인트를 요청합니다.

헤더의 요청 파라미터

`Authorization`

앱 클라이언트에 클라이언트 암호가 있는 경우 애플리케이션은 기본 HTTP 인증을 통해 권한 부여 헤더에 클라이언트 암호를 전달해야 합니다. `client_id` `client_secret` 보안은 기본 `Base64Encode(client_id:client_secret)`입니다.

`Content-Type`

항상 `'application/x-www-form-urlencoded'`여야 합니다.

본문의 요청 파라미터

`token`

(필수) 클라이언트가 취소하려는 새로 고침 토큰. 이 요청은 Amazon Cognito가 이 새로 고침 토큰에서 발급한 모든 액세스 토큰도 취소합니다.

필수 사항입니다.

`client_id`

(선택사항) 취소하려는 토큰의 앱 클라이언트 ID.

클라이언트가 퍼블릭이고 보안 암호가 없는 경우 필수 사항입니다.

철회 요청 예제

예 1: 클라이언트 암호가 없는 앱 클라이언트의 토큰 취소

```
POST /oauth2/revoke HTTP/1.1
    Host: https://mydomain.auth.us-east-1.amazoncognito.com
    Accept: application/json
```

```
    Content-Type: application/x-www-form-urlencoded
    token=2YotnFZFEjr1zCsicMWpAA&
    client_id=djc98u3jiedmi283eu928
```

예 2: 클라이언트 암호가 있는 앱 클라이언트의 토큰 취소

```
POST /oauth2/revoke HTTP/1.1
    Host: https://mydomain.auth.us-east-1.amazoncognito.com
    Accept: application/json
    Content-Type: application/x-www-form-urlencoded
    Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
    token=2YotnFZFEjr1zCsicMWpAA
```

오류 응답 철회

성공적인 응답에 빈 본문이 포함됩니다. 오류 응답은 `error` 필드를 포함하고 경우에 따라 `error_description` 필드도 포함하는 JSON 객체입니다.

엔드포인트 오류

- 요청에 토큰이 없는 경우 또는 앱 클라이언트에 대해 기능이 비활성화된 경우 HTTP 400 및 오류 `invalid_request`가 발생합니다.
- Amazon Cognito가 취소 요청에서 보낸 토큰이 새로 고침 토큰이 아닌 경우 HTTP 400 및 오류 `unsupported_token_type`이 발생합니다.
- 클라이언트 자격 증명이 유효하지 않은 경우 HTTP 401 및 오류 `invalid_client`가 발생합니다.
- 토큰이 취소된 경우 또는 클라이언트가 유효하지 않은 토큰을 제출한 경우 HTTP 200 OK가 발생합니다.

saml2/idresponse 엔드포인트

SAML `/saml2/idpresponse` 어설션을 수신합니다. service-provider-initiated (SP에서 시작한) 로그인에서는 SAML 2.0 ID 공급자 (IdP) 가 SAML 응답과 함께 사용자를 이 엔드포인트로 리디렉션합니다. SP에서 시작한 로그인에서는 애플리케이션이 이 엔드포인트와 상호 작용하지 않습니다. ACS (어설션 소비자 서비스) URL의 경로를 사용하여 `saml2/idpresponse` IdP를 구성합니다. 세션 시작에 대한 자세한 내용은 을 참조하십시오. Amazon Cognito 사용자 풀에서 SAML 세션 시작

IdP 개시 로그인에서는 사용자가 자체 프로세스를 통해 IdP로 로그인하고 HTTPS를 통해 요청 본문에 SAML 어설션을 제출할 수 있습니다. `HTTP POST` 요청 본문은 파라미터와 파라미터여야 합니다.

POST SAMLResponse Relaystate 자세한 정보는 IDP에서 시작한 SAML 로그인 사용을 참조하세요.

POST **/saml2/idpresponse**

IdP가 시작한 로그인에서 /saml2/idpresponse 엔드포인트를 사용하려면 사용자 풀에 사용자 세션에 대한 정보를 제공하는 파라미터를 사용하여 POST 요청을 생성하십시오.

- 로그인하려는 앱 클라이언트.
- 사용자가 최종 목적지로 이동하려는 콜백 URL.
- 사용자의 액세스 토큰에서 요청하려는 OAuth 2.0 범위입니다.
- 로그인 요청을 시작한 IdP.

IdP에서 시작한 요청 본문 파라미터

SAML 응답

사용자 풀의 유효한 앱 클라이언트 및 IdP 구성과 연결된 IdP의 Base64로 인코딩된 SAML 어설션입니다.

RelayState

RelayState파라미터에는 엔드포인트로 전달해야 하는 요청 파라미터가 포함되어 있습니다. oauth2/authorize 이러한 파라미터에 대한 자세한 내용은 을 참조하십시오권한 부여 엔드포인트.

response_type

OAuth 2.0 부여 유형.

client_id

앱 클라이언트 ID입니다.

redirect_uri

Amazon Cognito가 사용자에게 권한을 부여한 후 인증 서버에서 브라우저를 리디렉션하는 URL입니다.

identity_provider

사용자를 리디렉션하려는 ID 제공자의 이름.

idp_identifier

사용자를 리디렉션하려는 ID 제공자의 식별자.

scope

사용자가 권한 부여 서버에 요청하기를 원하는 OAuth 2.0 범위입니다.

긍정적인 응답이 있는 요청의 예

예 — POST 요청

다음 요청은 앱 클라이언트의 MySAMLIdP IdP로부터 사용자에게 인증 코드를 부여하기 위한 요청입니다. 1example23456789 사용자는 인증 코드를 https://www.example.com 사용하여 리디렉션합니다. 인증 코드는 OAuth 2.0 범위, 및 의 액세스 토큰이 포함된 토큰으로 교환할 수 있습니다. openid email phone

```
POST /saml2/idpresponse HTTP/1.1
User-Agent: USER_AGENT
Accept: */*
Host: example.auth.us-east-1.amazoncognito.com
Content-Type: application/x-www-form-urlencoded

SAMLResponse=[Base64-encoded SAML assertion]&RelayState=identity_provider
%3DMySAMLIdP%26client_id%3D1example23456789%26redirect_uri%3Dhttps%3A%2F
%2Fwww.example.com%26response_type%3Dcode%26scope%3Demail%2Bopenid%2Bphone
```

예 — 응답

다음은 이전 요청에 대한 응답입니다.

```
HTTP/1.1 302 Found
Date: Wed, 06 Dec 2023 00:15:29 GMT
Content-Length: 0
x-amz-cognito-request-id: 8aba6eb5-fb54-4bc6-9368-c3878434f0fb
Location: https://www.example.com?code=[Authorization code]
```

OAuth 2.0 권한 부여

Amazon Cognito 사용자 풀 OAuth 2.0 권한 부여 서버는 세 가지 유형의 OAuth 2.0 권한 부여에 대한 응답으로 토큰을 발급합니다. 사용자 풀의 각 앱 클라이언트에 대해 지원되는 권한 부여 유형을 설정할 수 있습니다. 암시적 또는 인증 코드 권한 부여와 동일한 앱 클라이언트에서 클라이언트 보안 인증 권

한 부여를 활성화할 수 없습니다. 암시적 및 인증 코드 권한 부여에 대한 요청은 권한 부여 엔드포인트에서 시작되고, 클라이언트 보안 인증 권한 부여 요청은 Token 엔드포인트에서 시작됩니다.

인증 코드 권한 부여

성공적인 인증 요청에 대한 응답으로 권한 부여 서버는 콜백 URL에 code 파라미터의 인증 코드를 추가합니다. 그런 다음 ID, 액세스 및 새로 고침 토큰에 대한 코드를 Token 엔드포인트와 교환해야 합니다. 인증 코드 권한 부여를 요청하려면 요청에서 response_type을 code로 설정합니다. 요청 예시는 인증 코드 권한 부여 섹션을 참조하세요.

인증 코드 권한 부여는 가장 안전한 형태의 권한 부여입니다. 이는 토큰 내용을 사용자에게 직접 표시하지 않습니다. 대신 앱이 사용자 토큰을 검색하고 안전하게 저장하는 역할을 합니다. Amazon Cognito에서는 인증 코드 권한 부여가 권한 부여 서버에서 세 가지 토큰 유형(ID, 액세스 및 새로 고침)을 모두 얻는 유일한 방법입니다. Amazon Cognito 사용자 풀 API를 통한 인증에서도 세 가지 토큰 유형을 모두 얻을 수 있지만 API는 aws.cognito.signin.user.admin 이외의 범위로 액세스 토큰을 발급하지 않습니다.

암시적 허용

성공적인 인증 요청에 대한 응답으로 권한 부여 서버는 access_token 파라미터의 액세스 토큰과 id_token 파라미터의 ID 토큰을 콜백 URL에 추가합니다. 암시적 권한 부여에는 Token 엔드포인트와의 추가 상호 작용이 필요하지 않습니다. 암시적 권한 부여를 요청하려면 요청에서 response_type을 token으로 설정합니다. 암시적 권한 부여는 ID와 액세스 토큰만 생성합니다. 요청 예시는 openid 범위가 없는 토큰 부여 섹션을 참조하세요.

암시적 권한 부여는 레거시 권한 부여입니다. 인증 코드 권한 부여와 달리 사용자가 토큰을 가로채고 검사할 수 있습니다. 암시적 권한 부여를 통한 토큰 전달을 방지하려면 인증 코드 권한 부여만 지원하도록 앱 클라이언트를 구성합니다.

클라이언트 자격 증명

클라이언트 자격 증명은 권한 부여 전용 액세스 권한입니다. machine-to-machine 클라이언트 보안 인증 권한 부여를 받으려면 권한 부여 엔드포인트를 우회하고 Token 엔드포인트에 직접 요청을 생성합니다. 앱 클라이언트는 클라이언트 암호를 가져야 하며 클라이언트 보안 인증 권한 부여만 지원해야 합니다. 성공적인 요청에 대한 응답으로 권한 부여 서버는 액세스 토큰을 반환합니다.

클라이언트 보안 인증 권한 부여의 액세스 토큰은 OAuth 2.0 범위를 포함하는 권한 부여 메커니즘입니다. 일반적으로 토큰에는 액세스 보호된 API에 대한 HTTP 작업을 승인하는 사용자 지정 범위 클레임이 포함됩니다. 자세한 정보는 리소스 서버를 통한 범위, M2M 및 API 인증을 참조하세요.

클라이언트 자격 증명 부여는 청구서에 비용을 추가합니다. AWS 자세한 내용은 Amazon Cognito 요금을 참조하세요.

Amazon Cognito 사용자 풀과 함께 권한 부여 코드 부여에 PKCE 사용

Amazon Cognito는 인증 코드 부여에서 코드 교환을 위한 증명 키 (PKCE) 인증을 지원합니다. PKCE는 퍼블릭 클라이언트를 위한 OAuth 2.0 인증 코드 부여의 확장입니다. PKCE는 도용된 인증 코드의 상환을 방지합니다.

아마존 코그니토가 PKCE를 사용하는 방법

PKCE 인증을 시작하려면 애플리케이션에서 고유한 문자열 값을 생성해야 합니다. 이 문자열은 Amazon Cognito가 초기 권한 부여를 요청하는 클라이언트와 인증 코드를 토큰으로 교환하는 클라이언트를 비교하는 데 사용하는 비밀 값인 코드 검증자입니다.

앱은 코드 검증 문자열에 SHA256 해시를 적용하고 결과를 base64로 인코딩해야 합니다. 해시된 문자열을 요청 본문의 권한 부여 엔드포인트 `code_challenge` as매개변수로 전달하십시오. 앱이 인증 코드를 토큰으로 교환할 때는 요청 본문의 `code_verifier` 매개 변수로 코드 검증자 문자열을 일반 텍스트로 포함해야 합니다. Token 엔드포인트 Amazon Cognito는 코드 검증 도구에서 동일한 hash-and-encode 작업을 수행합니다. Amazon Cognito는 코드 검증자가 승인 요청에서 수신한 것과 동일한 코드 챌린지로 이어진다고 판단하는 경우에만 ID, 액세스 및 새로 고침 토큰을 반환합니다.

PKCE를 사용하여 권한 부여 흐름을 구현하려면

1. Amazon Cognito 콘솔을 엽니다. 메시지가 표시되면 자격 증명을 입력합니다. AWS
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다. 사용자 풀을 생성하는 경우 마법사 실행 중에 앱 클라이언트를 설정하고 호스팅된 UI를 구성하라는 메시지가 표시됩니다.

   a. 새 사용자 풀을 생성하는 경우 설정 안내에 따라 앱 클라이언트를 설정하고 호스팅된 UI를 구성하십시오.

   b. 기존 사용자 풀을 구성하는 경우 아직 추가하지 않았다면 도메인과 공용 앱 클라이언트를 추가하세요.
4. PKCE에 대한 코드 챌린지를 생성하려면 일반적으로 범용 고유 식별자 (UUID) 인 임의의 영숫자 문자열을 생성하십시오. 이 문자열은 요청 시 제출할 `code_verifier` 매개 변수의 값입니다. Token 엔드포인트
5. SHA256 알고리즘으로 `code_verifier` 문자열을 해시합니다. 해싱 작업의 결과를 base64로 인코딩합니다. 이 문자열은 요청 시 에 제출할 `code_challenge` 매개 변수의 값입니다. 권한 부여 엔드포인트

   다음 Python 예제에서는 a를 `code_verifier` 생성하고 다음을 계산합니다. `code_challenge`

```
#!/usr/bin/env python3

import random
from base64 import urlsafe_b64encode
from hashlib import sha256
from string import ascii_letters
from string import digits

# use a cryptographically strong random number generator source
rand = random.SystemRandom()

code_verifier = ''.join(rand.choices(ascii_letters + digits, k=128))
code_verifier_hash = sha256(code_verifier.encode()).digest()
code_challenge = urlsafe_b64encode(code_verifier_hash).decode().rstrip('=')

print(f"code challenge: {code_challenge}")
print(f"code verifier: {code_verifier}")
```

다음은 Python 스크립트의 예제 출력입니다.

```
code challenge: Eh0mg-OZv7BAyo-tdv_vYamx1boOYDulDklyXoMDtLg
code verifier: 9D-aW_iygXrgQcWJd0y0tNVMPSXSChIc2xceDhvYVdGLCBk-
JWFTmBNjvKSdOrjTTYazOFbUmrFERrjWx6oKtK2b6z_x4_gHBDlr4K1mRFGyE8yA-05-_v7Dxf3EIYJH
```

6. PKCE를 통한 인증 코드 부여 요청으로 호스팅된 UI 로그인을 완료하십시오. 다음은 예제 URL입니다.

```
https://mydomain.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&client_id=1example23456789&redirect_uri=https://
www.example.com&code_challenge=Eh0mg-OZv7BAyo-
tdv_vYamx1boOYDulDklyXoMDtLg&code_challenge_method=S256
```

7. 승인을 code 수집하여 토큰 엔드포인트가 있는 토큰으로 교환하십시오. 다음은 요청 예제입니다.

```
POST /oauth2/token HTTP/1.1
Host: mydomain.us-east-1.amazoncognito.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 296

redirect_uri=https%3A%2F%2Fwww.example.com&
client_id=1example23456789&
```

```
code=7378f445-c87f-400c-855e-0297d072ff03&
grant_type=authorization_code&
code_verifier=9D-aW_iygXrgQcWJd0y0tNVMPSXSChIc2xceDhvYVdGLCBk-
JWFTmBNjvKSdOrjTTYazOFbUmrFERrjWx6oKtK2b6z_x4_gHBDlr4K1mRFGyE8yA-05-_v7Dxf3EIYJH
```

8. 응답을 검토하세요. 여기에는 ID, 액세스 및 새로 고침 토큰이 포함됩니다. Amazon Cognito 사용자 풀 토큰 사용에 대한 자세한 내용은 을 참조하십시오. 사용자 풀에 토큰 사용

호스팅 UI 및 페더레이션 오류 응답

호스팅 UI 또는 페더레이션 로그인의 로그인 프로세스에서 오류가 반환될 수 있습니다. 인증이 오류와 함께 종료될 수 있는 몇 가지 조건은 다음과 같습니다.

- 사용자가 사용자 풀이 수행할 수 없는 작업을 수행합니다.
- Lambda 트리거가 예상 구문으로 응답하지 않습니다.
- ID 제공업체(idP)가 오류를 반환합니다.
- Amazon Cognito가 사용자가 제공한 속성 정보를 검증하지 못했습니다.
- IdP가 필수 속성에 매핑되는 클레임을 보내지 않았습니다.

Amazon Cognito는 오류가 발생하면 다음 방법 중 하나로 이를 알립니다.

1. Amazon Cognito가 요청 파라미터에 오류가 있는 리디렉션 URL을 전송합니다.
2. Amazon Cognito가 호스팅 UI에 오류를 표시합니다.

Amazon Cognito가 요청 파라미터에 추가하는 오류의 형식은 다음과 같습니다.

```
https://<Callback URL>/?error_description=error+description&error=error+name
```

작업을 수행할 수 없는 사용자의 오류 정보 제출을 도와주는 경우 페이지의 URL 및 텍스트 또는 스크린샷을 캡처하도록 요청하세요.

> **Note**
>
> Amazon Cognito 오류 설명은 고정된 문자열이 아니므로 고정된 패턴 또는 형식을 사용하는 로직을 사용해서는 안 됩니다.

OIDC 및 소셜 ID 제공업체 오류 메시지

ID 제공업체가 오류를 반환할 수 있습니다. OIDC 또는 OAuth 2.0 IdP가 표준을 준수하는 오류를 반환하면 Amazon Cognito는 사용자를 콜백 URL로 리디렉션하고 오류 요청 파라미터에 공급자 오류 응답을 추가합니다. Amazon Cognito는 기존 오류 문자열에 공급자 이름과 HTTP 오류 코드를 추가합니다.

다음 URL은 오류를 Amazon Cognito에 반환한 IdP에서의 리디렉션 예입니다.

```
https://www.amazon.com/?error_description=LoginWithAmazon+Error+-+400+invalid_request
+The+request+is+missing+a+required+parameter+%3A+client_secret&error=invalid_request
```

Amazon Cognito는 공급자로부터 받은 것만 반환하므로 사용자가 보는 정보는 이 정보의 일부일 수 있습니다.

IdP를 통한 사용자의 최초 로그인에 문제가 발생하면 IdP는 오류 메시지를 사용자에게 직접 전달합니다. Amazon Cognito는 IdP에 대한 사용자 세션 검증 요청을 생성할 때 사용자에게 오류 메시지를 전달합니다. Amazon Cognito는 다음 엔드포인트의 OAuth 및 OIDC IdP 오류 메시지를 전달합니다.

`/token`

Amazon Cognito는 IdP 권한 부여 코드를 액세스 토큰으로 교환합니다.

`/.well-known/openid-configuration`

Amazon Cognito는 발급자 엔드포인트 경로를 검색합니다.

`/.well-known/jwks.json`

Amazon Cognito는 사용자의 JSON 웹 토큰(JWT)을 확인하기 위해 IdP가 토큰에 서명하는 데 사용하는 JSON 웹 키(JWK)를 검색합니다.

Amazon Cognito는 HTTP 오류를 반환할 수 있는 SAML 2.0 공급자에 대한 아웃바운드 세션을 시작하지 않으므로 SAML 2.0 IdP 세션 중에 발생한 사용자 오류에는 이러한 형태의 공급자 오류 메시지가 포함되지 않습니다.

Amazon Cognito 사용자 풀 API 참조

Amazon Cognito 사용자 풀을 사용하면 웹과 모바일 앱을 이용해 사용자가 가입하고 로그인하게 할 수 있습니다. 인증 사용자에 대해 암호를 변경할 수 있고 미인증 사용자에 대해 잊어버린 암호 흐름을 시작할 수 있습니다. 자세한 내용은 사용자 풀 인증 흐름 및 사용자 풀에 토큰 사용 섹션을 참조하세요.

Amazon Cognito 사용자 풀 API에는 사용자 풀과 사용자를 확인 및 수정하고, 사용자 인증과 권한 부여를 수행하는 작업이 포함됩니다. Amazon Cognito 사용자 풀 API에 결합되는 API 작업 클래스에 대한 설명은 Amazon Cognito 사용자 풀 API 및 사용자 풀 엔드포인트 사용 섹션을 참조하세요.

Amazon Cognito 사용자 풀 API 작업 및 구문에 대한 자세한 목록은 Amazon Cognito 사용자 풀 API 참조를 참조하세요. Amazon Cognito 사용자 풀 API 참조의 각 페이지에서는 다양한 AWS SDK의 구문 및 예제를 제공하는 참조 자료를 확인할 수 있습니다.

Amazon Cognito 자격 증명 풀(페더레이션 자격 증명) API 참조

웹과 모바일 앱 사용자는 Amazon Cognito 자격 증명 풀로 권한이 제한된 임시 AWS 자격 증명을 얻어 다른 AWS 서비스에 액세스할 수 있습니다.

자격 증명 풀(페더레이션 자격 증명) API 참조의 전체 내용은 Amazon Cognito API 참조를 참조하세요.

Amazon Cognito Sync API 참조

Amazon Cognito Sync는 애플리케이션 관련 사용자 데이터의 교차 디바이스 동기화를 지원하는 클라이언트 라이브러리 및 AWS 서비스입니다.

Amazon Cognito Sync API 참조에 대한 자세한 내용은 Amazon Cognito Sync API 참조를 참조하세요.

Amazon Cognito의 문서 기록

다음 표에서는 Amazon Cognito 설명서에 대한 중요 추가 사항을 설명합니다. 또한 사용자로부터 받은 의견을 수렴하기 위해 설명서가 자주 업데이트됩니다. 피드백을 제출하려면 Amazon Cognito 설명서의 페이지 하단에서 Feedback(피드백) 링크를 찾으세요.

변경 사항	설명	날짜
프리 토큰 Lambda 트리거에서 복잡한 객체에 대한 지원이 추가되었습니다.	이제 ID 및 액세스 토큰 클레임에 어레이와 JSON 객체를 추가할 수 있습니다.	2024년 5월 30일
검증된 권한 및 Amazon Cognito에 대한 정보가 업데이트되었습니다.	아마존 검증 권한은 이제 Amazon Cognito와 더욱 직접적으로 통합됩니다.	2024년 5월 15일
다중 지역 Amazon SES 검증 ID.	Amazon SES가 AWS 리전 없는 일부 지역에서는 Amazon Cognito 사용자가 두 원격 지역 간에 이메일을 풀링하여 로드 밸런싱합니다.	2024년 5월 10일
M2M 승인 및 관리 비용에 대한 정보가 추가되었습니다.	Amazon Cognito 사용자 풀의 machine-to-machine (M2M) 사용 사례에 클라이언트 자격 증명 부여를 사용하는 방법을 알아보십시오.	2024년 5월 9일
Amazon Cognito는 이제 유럽 (스페인) 및 아시아 태평양 (하이데라바드) 에서 사용할 수 있습니다. AWS 리전	이제 유럽 (스페인) 및 아시아 태평양 (하이데라바드) 지역에서 Amazon Cognito 리소스를 생성할 수 있습니다.	2024년 4월 15일
Amazon Cognito는 이제 아시아 태평양 (멜버른) 에서 사용할 수 있습니다. AWS 리전	이제 아시아 태평양 (멜버른) 지역에서 Amazon Cognito 리소스를 생성할 수 있습니다.	2024년 4월 4일

Amazon Cognito 사용자 풀용 Flutter에 예제 안드로이드 앱을 추가했습니다.	의 Flutter 애플리케이션 예제에서 Amazon Cognito용 스타터 모바일 앱을 구축할 수 있습니다. GitHub	2024년 4월 4일
새로운 시작 콘텐츠	시작하기, 일반적인 시나리오, 멀티테넌트 모범 사례, 로그인 후 리소스 액세스를 위한 확장된 콘텐츠	2024년 4월 1일
Amazon Cognito는 이제 유럽 (취리히) 에서 사용할 수 있습니다. AWS 리전	이제 유럽 (취리히) 지역에서 Amazon Cognito 리소스를 생성할 수 있습니다.	2024년 3월 14일
Amazon Cognito는 이제 중동 (UAE) 에서 사용할 수 있습니다. AWS 리전	이제 중동 (UAE) 지역에서 Amazon Cognito 리소스를 생성할 수 있습니다.	2024년 3월 8일
새로운 SAML 기능 및 개선된 콘텐츠.	이제 SAML 요청에 서명하고, SAML 응답을 암호화하고, IdP에서 시작하는 SAML SSO를 설정할 수 있습니다.	2024년 2월 1일
할당량 인상이 가능합니다.	이제 Amazon Cognito 요청 속도 할당량을 위한 추가 용량을 구매할 수 있습니다.	2024년 1월 25일
Amazon Cognito 자격 증명 풀은 서비스 할당량의 요청 비율을 지원합니다.	이제 Service Quotas 콘솔에서 Amazon Cognito 자격 증명 풀의 requests-per-second (RPS) 할당량을 모니터링하고 증가를 요청할 수 있습니다.	2023년 12월 19일
액세스 토큰 콘텐츠 사용자 지정을 위한 새 기능이 추가되었습니다.	이제 사용자 풀 액세스 토큰에서 클레임 및 범위를 추가, 수정 및 제거할 수 있습니다.	2023년 12월 12일

앱 클라이언트 및 OAuth 범위에 대한 콘텐츠를 개선했습니다.	사용자 풀 앱 클라이언트 및 리소스 서버를 통한 범위, M2M 및 API 인증 관련 사항을 보다 명확하게 수정했습니다. 기존 콘솔 지침을 제거했습니다.	2023년 11월 14일
기기 및 기기 인증에 대한 콘텐츠를 개선했습니다.	디바이스 키 및 디바이스 SRP 인증 사용에 대한 새로운 콘텐츠.	2023년 10월 18일
AWS Management Console 지침이 업데이트되었습니다.	사용자 풀 콘솔 참조를 제거하고 관련 주제 내에서 주제를 재배치하고 Amazon Cognito 콘솔의 탭 기반 구성에 지침을 추가했습니다.	2023년 8월 30일
LOGIN 엔드포인트에 대한 직접 액세스는 더 이상 강조하지 않았습니다.	사용자 풀 Login 엔드포인트의 시각적 개요를 추가하고 권한 부여 엔드포인트를 통한 인증 시작을 강조했습니다.	2023년 8월 30일
Amazon Cognito는 이제 아시아 태평양 (오사카) 과 이스라엘 (텔아비브) 에서 사용할 수 있습니다. AWS 리전	이제 아시아 태평양 (오사카) 및 이스라엘 (텔아비브) 지역에서 Amazon Cognito 리소스를 생성할 수 있습니다.	2023년 8월 30일
Amazon 검증 권한을 사용한 Amazon Cognito의 권한 부여에 대한 정보를 소개했습니다.	앱에서 Verified Permissions API를 호출하여 중앙 기관으로부터 액세스 결정을 내릴 수 있습니다.	2023년 8월 1일
Amazon Logs에 사용자 풀의 세부 사용자 활동을 CloudWatch 기록하는 새로운 기능을 추가했습니다.	이제 이메일 및 SMS 메시지 전송 오류를 CloudWatch 로그 그룹에 기록할 수 있습니다.	2023년 8월 1일

자격 증명 풀 게스트 사용자의 AWS 관리형 정책에 대한 정보가 업데이트되었습니다.	이제 자격 증명 풀 게스트 사용자의 권한 범위 축소에 인라인 세션 정책과 관리형 세션 정책이 모두 포함됩니다. AWS	2023년 5월 16일
Amazon Cognito 자격 증명 풀에 대한 콘텐츠 개선 및 새로운 콘솔 지침	새로운 콘솔 경험을 반영하여 새로운 콘솔 안내를 추가하고 자격 증명 풀에 대한 코드 통합 세부 정보를 개선했습니다.	2023년 5월 16일
서비스 홈페이지 및 사용자 풀 홈페이지의 추가 및 개선	Amazon Cognito 및 사용자 풀의 개요 페이지가 업데이트되었습니다.	2023년 5월 16일
사용자 풀 토큰 설명서의 전반적인 개선 사항.	예시 토큰을 업데이트하고 토큰 검증에 대한 새로운 정보를 추가했습니다.	2023년 2월 16일
이제 Amazon Cognito 자격 증명 풀 데이터 이벤트를 기록할 수 있습니다. AWS CloudTrail	CloudTrail 데이터 이벤트를 기록하는 트레일에서 Amazon Cognito 자격 증명 풀의 대량 API 작업을 선택할 수 있도록 지원합니다.	2023년 2월 15일
Lambda 트리거 예제와 설명이 업데이트되었습니다.	Lambda 트리거 예제가 버전 3으로 업데이트되었습니다 JavaScript . 이제 Lambda 트리거를 API 작업과 직접 연관시킬 수 있습니다.	2023년 1월 31일
Amazon Cognito 자격 증명 풀은 인증되지 않은 AWS 세션에 관리형 정책을 적용합니다.	향상된 흐름을 사용하여 인증하는 자격 증명 풀 사용자는 이제 세션에 추가 AWS 관리형 정책을 적용할 수 있습니다.	2023년 1월 31일

코드 예제가 추가되었습니다.	이제 이 가이드에 Amazon Cognito 앱 예시 코드가 다양한 프로그래밍 언어로 포함되었습니다.	2023년 1월 23일
Amazon Cognito 사용자 풀을 사용한 API 모델 및 인증에 대한 정보가 추가되었습니다.	Amazon Cognito 사용자 풀에는 요청 권한 부여를 위한 여러 API 인터페이스와 형식이 있습니다.	2022년 12월 15일
Amazon Cognito는 이제 유럽(밀라노) 에서 사용할 수 있습니다. AWS 리전	이제 유럽(밀라노) 리전에서 Amazon Cognito 사용자 풀을 생성할 수 있습니다.	2022년 12월 6일
사용자 풀 삭제 보호에 대한 정보가 추가되었습니다.	를 사용하여 새 사용자 풀을 AWS Management Console만들면 이제 기본적으로 삭제되지 않도록 보호됩니다.	2022년 10월 20일
호스팅된 UI에 대한 사용 설명서와 호스팅된 UI에 TOTP MFA에 대한 정보가 추가되었습니다.	이제 사용자는 Amazon Cognito 호스팅 UI에서 TOTP MFA 디바이스를 등록할 수 있습니다. 이제 기본 호스팅 UI를 미리 볼 수 있습니다.	2022년 9월 8일
Amazon AWS WAF Cognito에 대한 정보가 추가되었습니다.	이제 AWS WAF 웹 ACL을 Amazon Cognito 사용자 풀과 연결할 수 있습니다.	2022년 8월 3일
더 많은 예제 AWS CloudTrail 이벤트가 추가되었습니다.	이제 Amazon Cognito가 페더레이션 및 호스팅 UI 요청을 추적에 기록합니다.	2022년 6월 15일
2단계 속성 확인에 대한 정보가 추가되었습니다.	이제 사용자가 새 이메일 주소 또는 전화번호를 인증해야 로그인할 수 있는지 여부를 선택할 수 있습니다.	2022년 6월 9일

페더레이션 설명서가 업데이트되었습니다. 새 IP 주소 전파 기능.	사용자 풀 소셜 설정을 위한 안내를 업데이트했습니다. IdPs 페더레이션 사용자 프로필 및 속성 매핑에 대한 정보가 추가되었습니다. 고급 보안을 위해 장치 지문에 대한 새로운 정보가 추가되었습니다.	2022년 5월 31일
호스팅된 UI와의 상호 작용 없이 페더레이션 사용자로 로그인할 수 있습니다.	Amazon Cognito가 자동으로 사용자를 페더레이션된 로그인으로 안내하도록 애플리케이션을 북마크하는 방법에 대한 새 페이지를 추가했습니다.	2022년 5월 29일
Amazon Cognito 사용자 풀을 위한 지역 내 SMS 및 이메일 메시징	이제 사용자 풀과 AWS 리전 동일하게 SMS 메시지에는 Amazon 심플 알림 서비스를, 이메일 메시지에는 Amazon 심플 이메일 서비스를 사용할 수 있습니다.	2022년 3월 14일
할당량 페이지 업데이트	리소스 및 요청률 할당량을 추가하고 명확히 했습니다.	2022년 1월 10일
새로운 Amazon Cognito 사용자 풀 콘솔 환경	업데이트된 Amazon Cognito 콘솔에서 사용자 풀을 생성하고 관리하는 지침을 업데이트했습니다.	2021년 11월 18일
RevokeToken API 및 해지 엔드포인트	RevokeToken 작업을 사용하여 사용자의 새로 고침 토큰을 취소할 수 있습니다.	2021년 6월 10일
멀티테넌트 모범 사례	멀티테넌트 애플리케이션에 대한 모범 사례가 추가되었습니다.	2021년 3월 4일

액세스 제어를 위한 속성	Amazon Cognito 자격 증명 풀은 고객이 사용자에게 리소스에 대한 액세스 권한을 부여하는 방법으로 액세스 제어(AFAC) 에 대한 속성을 제공합니다. AWS 사용자가 Amazon Cognito와 페더레이션하는 데 사용하는 자격 증명 공급자의 사용자 속성을 기반으로 권한 부여를 수행할 수 있습니다.	2021년 1월 15일
사용자 지정 SMS 발신자 Lambda 트리거 및 사용자 지정 이메일 발신자 Lambda 트리거	사용자 지정 SMS 발신자 Lambda 트리거와 사용자 지정 이메일 발신자 Lambda 트리거를 사용하면 서드 파티 공급자가 Lambda 함수 코드 내에서 사용자에게 이메일 및 SMS 알림을 보낼 수 있습니다.	2020년 11월 30일
아마존 코그니토 토큰 업데이트	업데이트된 만료 정보가 액세스, ID 및 새로 고침 토큰에 추가되었습니다.	2020년 10월 29일

아마존 코그니토 서비스 쿼터	Amazon Cognito 범주 할당량에 Service Quotas를 사용할 수 있습니다. Service Quotas 콘솔을 사용하여 할당량 사용량을 확인하고, 할당량 증가를 요청하고, 할당량 사용량을 모니터링하는 경보를 CloudWatch 생성할 수 있습니다. 이 변경의 일환으로 Amazon Cognito 사용자 풀의 사용 가능한 CloudWatch 지표 섹션이 새 정보를 반영하도록 업데이트되었습니다. 새 섹션 이름은 다음과 같습니다. 및 Service Quotas에서의 CloudWatch 할당량 및 사용량 추적	2020년 10월 29일
아마존 코그니토 쿼터 분류	할당량 범주를 사용하여 할당량 사용량을 모니터링하고 할당량 증가를 요청할 수 있습니다. 할당량은 일반적인 사용 사례에 따라 범주로 분류됩니다.	2020년 8월 17일
미국 고브클라우드에서 아마존 코그니토 지원 AWS	Amazon Cognito는 이제 AWS GovCloud (미국) 리전에서 지원됩니다.	2020년 5월 13일
아마존 Cognito Pinpoint 문서 업데이트	새 서비스 연결 역할이 추가되었습니다. ‘Amazon Cognito 사용자 풀과 함께 Amazon Pinpoint 분석 사용’의 지침이 업데이트되었습니다.	2020년 5월 13일

새로운 Amazon Cognito 전용 보안 챕터	보안 장을 통해 조직은 서비스의 기본 보안 및 구성 가능한 보안 모두에 대한 심층적인 정보를 얻을 수 있습니다. AWS 새로운 장에서는 클라우드 및 클라우드 내 보안에 대한 정보를 제공합니다.	2020년 4월 30일
Amazon Cognito 자격 증명 풀은 이제 Apple을 통한 로그인을 지원합니다.	Sign in with Apple은 cn-north-1 리전을 제외하고 Amazon Cognito를 사용할 수 있는 모든 리전에서 사용 가능합니다.	2020년 4월 7일
새로운 페이스북 API 버전 관리	Facebook API에 대한 버전 선택이 추가되었습니다.	2020년 4월 3일
사용자 이름 대소문자를 구분하지 않는 업데이트	사용자 풀을 생성하기 전에 사용자 이름의 대/소문자를 구분하지 않는 것에 대한 권장 사항이 추가되었습니다.	2020년 2월 11일
에 대한 새로운 정보 AWS Amplify	SDK 및 라이브러리를 AWS Amplify 사용하여 Amazon Cognito를 웹 또는 모바일 앱과 통합하는 방법에 대한 정보가 추가되었습니다. AWS Amplify 앞에 있는 Amazon Cognito SDK 사용에 대한 정보를 제거했습니다.	2019년 11월 22일

사용자 풀 트리거에 대한 새 속성	Amazon Cognito는 이제 대부분의 사용자 풀 트리거에 대해 AWS Lambda 함수에 전달하는 이벤트 정보에 `clientMetadata` 파라미터를 포함합니다. 이 파라미터를 사용하면 추가 데이터로 사용자 지정 인증 워크플로우를 개선할 수 있습니다.	2019년 10월 4일
한도가 업데이트되었습니다.	ListUsers API 작업에 대한 제한 제한이 업데이트되었습니다.	2019년 6월 25일
새 한도	사용자 풀의 소프트 제한에는 이제 사용자 수에 대한 제한이 포함됩니다.	2019년 6월 17일
Amazon Cognito 사용자 풀에 대한 Amazon SES 이메일 설정	Amazon SES 구성을 사용하여 Amazon Cognito에서 사용자에게 이메일을 보내도록 사용자 풀을 구성할 수 있습니다. 이 설정을 사용하면 그렇지 않은 경우보다 Amazon Cognito에서 더 많은 볼륨의 이메일을 전송할 수 있습니다.	2019년 4월 8일
태깅 지원	Amazon Cognito 리소스 태깅과 관련한 정보가 추가되었습니다.	2019년 3월 26일
사용자 지정 도메인의 인증서 변경	사용자 지정 도메인을 사용하여 Amazon Cognito 호스트된 UI를 호스트하는 경우 필요에 따라 이 도메인의 SSL 인증서를 변경할 수 있습니다.	2018년 12월 19일

새 한도	각 사용자가 소속될 수 있는 최대 그룹 수에 새로운 한도가 추가되었습니다.	2018년 12월 14일
업데이트된 한도	사용자 풀의 소프트 한도가 업데이트되었습니다.	2018년 12월 11일
이메일 주소 및 전화번호 확인을 위한 문서 업데이트	사용자가 앱에 가입할 때 이메일 또는 전화 확인을 요구하도록 사용자 풀을 구성하는 방법에 대한 정보를 추가했습니다.	2018년 11월 20일
테스트 이메일을 위한 문서 업데이트	앱을 테스트할 때 Amazon Cognito에서 이메일을 시작하는 방법에 대한 지침을 추가했습니다.	2018년 11월 13일
아마존 코그니토 어드밴스드 시큐리티	개발자가 앱과 사용자를 악성 봇으로부터 보호하고 손상된 자격 증명으로부터 사용자 계정을 보호하며 로그인 시도의 계산된 위험에 기초하여 로그인하는 데 필요한 횟수를 자동으로 조정할 수 있는 새로운 보안 기능을 추가했습니다.	2018년 6월 14일
Amazon Cognito 호스팅 UI용 사용자 지정 도메인	Amazon Cognito 사용자 풀의 호스트된 UI에 개발자의 사용자 지정 도메인을 사용하도록 허용합니다.	2018년 6월 4일
Amazon Cognito 사용자 풀 OIDC 자격 증명 공급자	추가된 사용자 풀이 Salesforce 또는 Ping Identity 같은 OpenID Connect(OIDC) 자격 증명 공급자를 통해 로그인합니다.	2018년 5월 17일

Amazon Cognito 람다 마이그레이션 트리거	Lambda 마이그레이션 트리거 기능을 다루는 페이지가 추가되었습니다.	2018년 4월 8일
Amazon Cognito 개발자 가이드 업데이트	최상위 'Amazon Cognito란' 및 'Amazon Cognito 시작하기'가 추가되었습니다. 또한 일반적인 시나리오가 추가되고 사용자 풀 TOC가 재구성되었습니다. 'Amazon Cognito 사용자 풀 시작하기' 섹션이 추가되었습니다.	2018년 4월 6일
아마존 Cognito 어드밴스드 시큐리티 베타	개발자가 앱과 사용자를 악성 봇으로부터 보호하고 인터넷에서 손상된 자격 증명으로부터 사용자 계정을 보호하며 로그인 시도의 계산된 위험에 기초하여 로그인하는 데 필요한 횟수를 자동으로 조정할 수 있는 새로운 보안 기능을 추가했습니다.	2017년 11월 28일
아마존 핀포인트 통합	Amazon Pinpoint를 사용하여 Amazon Cognito 사용자 풀 앱에 대한 분석을 제공하고 Amazon Pinpoint 캠페인에 대한 사용자 데이터를 보강하는 기능이 추가되었습니다.	2017년 9월 26일

Amazon Cognito 사용자 풀의 페더레이션 및 내장된 앱 UI 기능	사용자가 Facebook, Google, Login with Amazon 또는 SAML 자격 증명 공급자를 통해 사용자 풀에 로그인할 수 있도록 허용하는 기능이 추가되었습니다. 사용자 지정 클레임에서 사용자 지정이 가능한 기본 제공 앱 UI 및 OAuth 2.0 지원을 추가했습니다.	2017년 8월 10일
HIPAA 및 PCI 규정 준수 관련 기능 변경	전화 번호나 이메일 주소를 사용자 이름을 사용할 수 있도록 허용하는 기능이 추가되었습니다.	2017년 6월 7일
사용자 그룹 및 역할 기반 액세스 제어 기능	사용자 그룹을 생성하고 관리하기 위해 관리 기능이 추가되었습니다. 관리자는 그룹 멤버십과 관리자 생성 역할에 따라 사용자에게 IAM 역할을 할당할 수 있습니다.	2016년 12월 15일
문서 업데이트	사용자 풀에서 AWS Lambda 트리거를 사용하는 방법을 보여주는 예제가 업데이트되었습니다.	2016년 11월 27일
문서 업데이트	iOS 코드 예제가 업데이트되었습니다.	2016년 11월 18일
문서 업데이트	사용자 계정의 확인 흐름에 대한 정보가 추가되었습니다.	2016년 11월 9일
사용자 계정 생성 기능	Amazon Cognito 콘솔 및 API를 통해 사용자 계정을 생성하기 위해 관리 기능이 추가되었습니다.	2016년 10월 6일

사용자 가져오기 기능	Cognito 사용자 풀에 대한 대량 가져오기 기능이 추가되었습니다. 이 기능을 사용하여 기존 자격 증명 공급자에서 Amazon Cognito 사용자 풀로 사용자를 마이그레이션합니다.	2016년 9월 1일
Cognito 사용자 풀의 일반 가용성	Cognito 사용자 풀 기능이 추가되었습니다. 이 기능을 사용하여 사용자 디렉터리를 생성 및 유지 관리하고 사용자 풀을 통해 모바일 앱 또는 웹 애플리케이션에 가입 및 로그인을 추가합니다.	2016년 7월 28일
SAML 지원	SAML 2.0(Security Assertion Markup Language 2.0)을 통해 자격 증명 공급자의 인증에 대한 지원이 추가되었습니다.	2016년 6월 23일
CloudTrail 통합	와 통합이 추가되었습니다 AWS CloudTrail.	2016년 2월 18일
Lambda와 이벤트 통합	Amazon Cognito의 중요한 이벤트에 대한 응답으로 AWS Lambda 함수를 실행할 수 있습니다.	2015년 4월 9일
Amazon Kinesis로의 데이터 스트림	데이터 스트림에 제어 및 통찰력을 제공합니다.	2015년 3월 4일
OpenID Connect 지원	OpenID Connect 공급자에 대한 지원을 활성화합니다.	2014년 11월 23일
푸시 동기화	자동 푸시 동기화에 대한 지원을 활성화합니다.	2014년 11월 6일

개발자 인증 ID 지원 추가	Amazon Cognito에서 자체 인증 및 자격 증명 관리 시스템이 있는 개발자를 자격 증명 공급자로 처리할 수 있습니다.	2014년 9월 29일
아마존 코그니토 일반 가용성		2014년 7월 10일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.