
Amazon Cognito

개발자 가이드



Amazon Cognito: 개발자 가이드

Copyright © 2019 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

Amazon Cognito란 무엇입니까?	1
Amazon Cognito의 기능	2
Amazon Cognito 시작하기	2
리전별 가용성	3
Amazon Cognito 가격	3
Amazon Cognito 콘솔 사용	3
리전 데이터 고려 사항	4
Amazon Cognito 시작하기	5
일반적인 Amazon Cognito 시나리오	6
사용자 풀을 통한 인증	6
서버 측 리소스 액세스	6
API 게이트웨이 및 Lambda에서 리소스 액세스	7
사용자 풀 및 자격 증명 풀을 사용하여 AWS 서비스에 액세스	8
타사를 통한 인증 및 자격 증명 풀을 통한 AWS 서비스 액세스	8
Amazon Cognito를 사용한 AWS AppSync 리소스 액세스	9
자습서	10
사용자 풀 생성	10
관련 리소스	10
자격 증명 풀 생성	10
관련 리소스	11
AWS 리소스 정리	11
사용자 풀에 웹 또는 모바일 앱 추가	11
Amazon Cognito 사용자 풀	12
사용자 풀 시작하기	13
필수 조건: AWS 계정 가입	13
단계 1. 사용자 풀이 포함된 사용자 디렉터리 생성	13
단계 2. 호스팅 웹 UI를 활성화하는 앱 추가	14
단계 3. 사용자 풀에 소셜 로그인 추가(옵션)	15
단계 4. SAML 자격 증명 공급자를 사용한 사용자 풀 로그인 추가(옵션)	19
단계 5. Amazon Cognito 사용자 풀 SDK 설치	20
다음 단계	21
웹 또는 모바일 앱 추가	22
JavaScript 앱 추가	22
Android 앱 추가	37
iOS 앱 추가	57
호스팅 UI 사용	71
앱 클라이언트 추가(AWS Management 콘솔)	72
앱 클라이언트 구성	73
도메인 구성	77
내장 웹페이지 사용자 지정	82
리소스 서버 정의	86
타사를 통한 로그인 추가	88
소셜 자격 증명 공급자 추가	88
SAML 공급자 추가	92
OIDC 공급자 추가	99
속성 매핑 지정	104
보안 관리	107
멀티 팩터 인증(MFA) 추가	107
고급 보안 기능 추가	110
Lambda 트리거 사용	118
Lambda 트리거 사용	120
사용자 풀 Lambda 트리거 이벤트	120
사용자 풀 Lambda 트리거 공통 파라미터	120
Lambda 트리거 소스	121

사전 가입 Lambda 트리거	123
사후 확인 Lambda 트리거	128
사전 인증 Lambda 트리거	131
사후 인증 Lambda 트리거	133
문제 Lambda 트리거	136
사전 토큰 생성 Lambda 트리거	144
사용자 마이그레이션 Lambda 트리거	147
사용자 지정 메시지 Lambda 트리거	150
Amazon Pinpoint 분석 사용	155
Amazon Pinpoint 분석 사용	155
Amazon Pinpoint 분석 설정 지정(AWS CLI 및 AWS API)	156
사용자 관리	156
사용자 계정 가입 및 확인	156
관리자로 사용자 생성	163
사용자 풀에 그룹 추가	166
사용자 관리 및 검색	168
사용자 풀로 사용자 가져오기	171
이메일 설정	182
기본 이메일 기능	182
Amazon SES 이메일 구성	182
이메일 계정 구성	182
인증	185
사용자 풀 인증 흐름	187
토큰 사용	190
로그인 후 리소스 액세스	197
서버 측 리소스 액세스	6
API 게이트웨이 및 Lambda를 통해 리소스 액세스	198
자격 증명 풀을 사용하여 AWS 리소스 액세스	199
사용자 풀 콘솔 참조	201
사용자 풀 이름	202
사용자 및 그룹	202
속성	202
암호 요건	208
사용자 생성 관리 정책	208
이메일 또는 전화 확인	209
메시지 사용자 지정	209
태그	213
디바이스	213
앱 클라이언트	214
트리거	215
설정 검토	215
분석	215
앱 클라이언트 설정	216
도메인 이름	217
UI 사용자 지정	217
리소스 서버	218
자격 증명 공급자	218
속성 매핑	221
Amazon Cognito 자격 증명 풀	223
자격 증명 풀 시작하기	223
AWS 계정에 가입	224
Amazon Cognito에서 자격 증명 풀 만들기	224
Mobile 또는 JavaScript SDK 설치	224
자격 증명 공급자 통합	225
자격 증명 얻기	225
자격 증명 풀 사용	225
사용자 IAM 역할	226

인증된 자격 증명 및 인증되지 않은 자격 증명	226
미인증 자격 증명 활성화 또는 비활성화	226
자격 증명 유형과 연관된 역할 변경	226
인증 공급자 활성화 또는 편집	227
자격 증명 풀 삭제	227
자격 증명 풀에서 자격 증명 삭제	227
데이터 세트 관리	228
대량으로 데이터 게시	228
푸시 동기화 활성화	228
Amazon Cognito 스트림 설정	229
Amazon Cognito 이벤트 설정	229
자격 증명 풀 개념	229
자격 증명 풀 인증 흐름	229
IAM 역할	234
역할 신뢰 및 권한	238
역할 기반 액세스 제어	238
역할 매핑을 위한 역할 만들기	239
역할 전달 권한 부여	239
토큰을 사용하여 사용자에게 역할 할당	240
규칙 기반 매핑을 사용하여 사용자에게 역할 할당	240
규칙 기반 매핑에 사용할 토큰 클레임	241
역할 기반 액세스 제어를 위한 모범 사례	241
자격 증명 받기	242
Android	242
iOS - Objective-C	243
iOS - Swift	244
JavaScript	245
Unity	246
Xamarin	247
AWS 제품 액세스	248
Android	248
iOS - Objective-C	248
iOS - Swift	248
JavaScript	248
Unity	249
Xamarin	249
자격 증명 풀 외부 자격 증명 공급자	249
Facebook	249
Login with Amazon	254
Google	257
OpenID Connect 공급자	263
SAML 자격 증명 공급자	265
개발자 인증 자격 증명	267
인증 흐름 이해	267
개발자 공급자 이름 정의 및 해당 이름과 자격 증명 풀 연결	267
자격 증명 공급자 구현	268
로그인 맵 업데이트(Android 및 iOS 전용)	273
토큰 가져오기(서버 측)	274
기존 소셜 자격 증명에 연결	275
공급자 간 전환 지원	275
자격 증명 전환	277
Android	278
iOS - Objective-C	278
iOS - Swift	278
JavaScript	279
Unity	279
Xamarin	280

Amazon Cognito Sync	281
Amazon Cognito Sync 시작하기	281
AWS 계정에 가입	281
Amazon Cognito에서 자격 증명 풀 설정	282
데이터 저장 및 동기화	282
데이터 동기화	282
Amazon Cognito Sync 클라이언트 초기화	282
데이터 세트 이해	284
데이터 세트의 데이터 읽기 및 쓰기	285
로컬 데이터를 Sync 스토어와 동기화	287
콜백 처리	289
Android	289
iOS - Objective-C	290
iOS - Swift	293
JavaScript	295
Unity	297
Xamarin	299
푸시 동기화	300
Amazon Simple Notification Service(Amazon SNS) 앱 생성	301
Amazon Cognito 콘솔에서 푸시 동기화 활성화	301
앱에서 푸시 동기화 사용: Android	301
앱에서 푸시 동기화 사용: iOS - Objective-C	303
앱에서 푸시 동기화 사용: iOS - Swift	304
Amazon Cognito 스트림	306
Amazon Cognito 이벤트	308
제한	311
소프트 제한	311
하드 제한	312
API 참조	316
사용자 풀 API 참조	316
사용자 풀 Auth API 참조	316
권한 부여 엔드포인트	316
토큰 엔드포인트	320
USERINFO 엔드포인트	324
로그인 엔드포인트	325
로그아웃 엔드포인트	326
자격 증명 API 참조	327
Cognito 동기화 API 참조	327
AWS CloudTrail을 사용하여 Amazon Cognito API 호출 로깅	328
CloudTrail의 Amazon Cognito 정보	328
예제: Amazon Cognito 로그 파일 항목	329
리소스 태그 지정	331
지원되는 리소스	331
태그 제한	331
콘솔로 태그 관리	331
AWS CLI 예	332
태그 할당	332
태그 보기	333
태그 제거	333
리소스를 생성할 때 태그 적용	334
API 작업	334
사용자 풀 태그에 대한 API 작업	334
자격 증명 풀 태그에 대한 API 작업	334
리소스 권한	335
Amazon 리소스 이름(ARN)	335
정책 예제	335
관리형 정책	336

서명된 API와 서명되지 않은 API 비교	337
서비스 연결 역할 사용	338
Amazon Cognito에 대한 서비스 연결 역할 권한	338
Amazon Cognito에 대한 서비스 연결 역할 생성	339
Amazon Cognito에 대한 서비스 연결 역할 편집	339
Amazon Cognito에 대한 서비스 연결 역할 삭제	339
Amazon Cognito 서비스 연결 역할에 대해 지원되는 리전	340
문서 기록	341

Amazon Cognito란 무엇입니까?

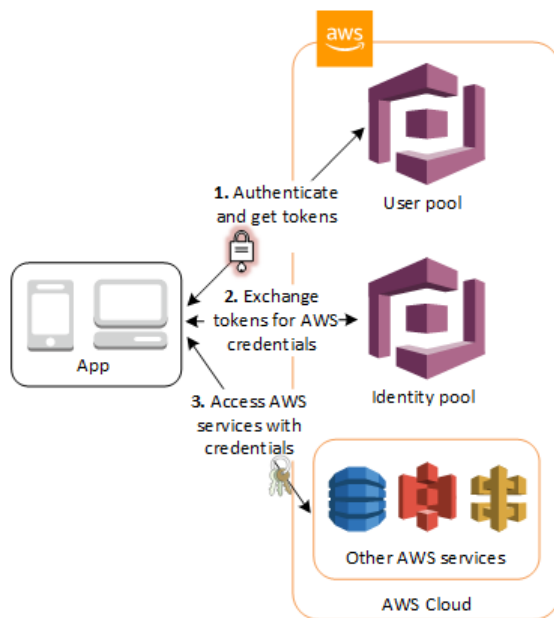
Amazon Cognito는 웹 및 모바일 앱에 대한 인증, 권한 부여 및 사용자 관리를 제공합니다. 사용자는 사용자 이름과 암호를 사용하여 직접 로그인하거나 Facebook, Amazon, 또는 Google 같은 타사를 통해 로그인할 수 있습니다.

Amazon Cognito의 두 가지 주요 구성 요소는 사용자 풀과 자격 증명 풀입니다. 사용자 풀은 앱 사용자의 가입 및 로그인 옵션을 제공하는 사용자 디렉터리입니다. 자격 증명 풀을 통해 기타 AWS 서비스에 대한 사용자 액세스 권한을 부여할 수 있습니다. 자격 증명 풀과 사용자 풀을 별도로 또는 함께 사용할 수 있습니다.

Amazon Cognito 사용자 풀 및 자격 증명 풀을 함께 사용

아래 일반적인 Amazon Cognito 시나리오 다이어그램을 참조하십시오. 목표는 사용자 인증 이후 다른 AWS 서비스에 대한 사용자 액세스 권한을 부여하는 것입니다.

1. 첫 번째 단계에서 앱 사용자는 사용자 풀을 통해 로그인하여 인증 성공 이후 사용자 풀 토큰을 받습니다.
2. 다음으로 앱은 자격 증명 풀을 통해 사용자 풀 토큰을 AWS 자격 증명으로 교환합니다.
3. 마지막으로 앱 사용자는 AWS 자격 증명을 사용하여 Amazon S3 또는 DynamoDB 등 기타 AWS 서비스에 액세스할 수 있습니다.



자격 증명 풀과 사용자 풀 사용에 대한 예제는 [일반적인 Amazon Cognito 시나리오 \(p. 6\)](#) 단원을 참조하십시오.

Amazon Cognito는 SOC 1-3, PCI DSS, ISO 27001을 준수하며, HIPAA-BAA 적격 서비스입니다. 자세한 내용은 [AWS 범위 내 서비스](#)를 참조하십시오. 또한 [리전 데이터 고려 사항 \(p. 4\)](#) 단원도 참조하십시오.

주제

- [Amazon Cognito의 기능 \(p. 2\)](#)

- [Amazon Cognito 시작하기 \(p. 2\)](#)
- [리전별 가용성 \(p. 3\)](#)
- [Amazon Cognito 가격 \(p. 3\)](#)
- [Amazon Cognito 콘솔 사용 \(p. 3\)](#)
- [리전 데이터 고려 사항 \(p. 4\)](#)

Amazon Cognito의 기능

사용자 풀

사용자 풀은 Amazon Cognito의 사용자 디렉터리입니다. 사용자 풀에서 사용자는 Amazon Cognito를 통해, 또는 타사 자격 증명 공급자(IdP)를 통해 연동하여 웹 또는 모바일 앱에 로그인할 수 있습니다. 사용자가 직접 또는 타사를 통해 로그인하는지 여부와 무관하게 사용자 풀의 모든 멤버는 디렉터리 프로필을 보유하며, SDK를 통해 이를 액세스할 수 있습니다.

사용자 풀 제공 사항:

- 가입 및 로그인 서비스.
- 사용자 로그인을 위한 내장 사용자 지정 웹 UI
- Facebook, Google, Login with Amazon을 통한 소셜 로그인 및 사용자 풀의 SAML 및 OIDC 자격 증명 공급자를 통한 로그인.
- 사용자 디렉터리 관리 및 사용자 프로필.
- 멀티 팩터 인증(MFA), 이상 있는 자격 증명 확인, 계정 탈취 보호, 전화 및 이메일 확인과 같은 보안 기능.
- AWS Lambda 트리거를 통한 사용자 지정 워크플로우 및 사용자 마이그레이션.

사용자 풀에 대한 자세한 내용은 [사용자 풀 시작하기 \(p. 13\)](#) 및 [Amazon Cognito 사용자 풀 API 참조](#) 단원을 참조하십시오.

자격 증명 풀

자격 증명 풀로 사용자는 임시 AWS 자격 증명을 얻어 Amazon S3 및 DynamoDB 등과 같은 다른 AWS 서비스에 액세스할 수 있습니다. 자격 증명 풀은 익명 게스트 사용자는 물론 자격 증명에 대한 사용자 인증에 사용할 수 있는 다음 자격 증명 공급자를 지원합니다.

- Amazon Cognito 사용자 풀
- Facebook, Google, Login with Amazon을 통한 소셜 로그인
- OpenID Connect(OIDC) 공급자
- SAML 자격 증명 공급자
- 개발자 인증 자격 증명

사용자 프로필 정보를 저장하려면 자격 증명 풀이 사용자 풀에 통합되어야 합니다.

자격 증명 풀에 대한 자세한 내용은 [Amazon Cognito 자격 증명 시작하기\(연동 자격 증명\) \(p. 223\)](#) 및 [Amazon Cognito 자격 증명 풀 API 참조](#) 단원을 참조하십시오.

Amazon Cognito 시작하기

주요 작업 및 시작 지점 가이드는 [Amazon Cognito 시작하기 \(p. 5\)](#) 단원을 참조하십시오.

비디오, 기사, 설명서 및 샘플 앱은 [Amazon Cognito 개발자 리소스](#)를 참조하십시오.

Amazon Cognito를 사용하려면 AWS 계정이 있어야 합니다. 자세한 내용은 [Amazon Cognito 콘솔 사용 \(p. 3\)](#) 단원을 참조하십시오.

리전별 가용성

Amazon Cognito는 전 세계의 여러 AWS 리전에서 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 영역은 아주 높은 수준의 가용성과 중복성을 제공하면서 지연 시간을 최소화하여 AWS가 Amazon Cognito를 포함한 여러 서비스를 제공할 수 있도록 합니다.

현재 Amazon Cognito를 사용할 수 있는 모든 리전의 목록을 보려면 Amazon Web Services 일반 참조의 [AWS 리전 및 엔드포인트](#)를 참조하십시오. 각 리전에서 사용할 수 있는 가용 영역의 수를 알아보려면 [AWS 글로벌 인프라](#)를 참조하십시오.

Amazon Cognito 가격

Amazon Cognito 요금에 대한 자세한 내용은 [Amazon Cognito 요금](#)을 참조하십시오.

Amazon Cognito 콘솔 사용

[Amazon Cognito 콘솔](#)을 사용하여 사용자 풀 및 자격 증명 풀을 생성 및 관리할 수 있습니다.

Amazon Cognito 콘솔을 사용하려면

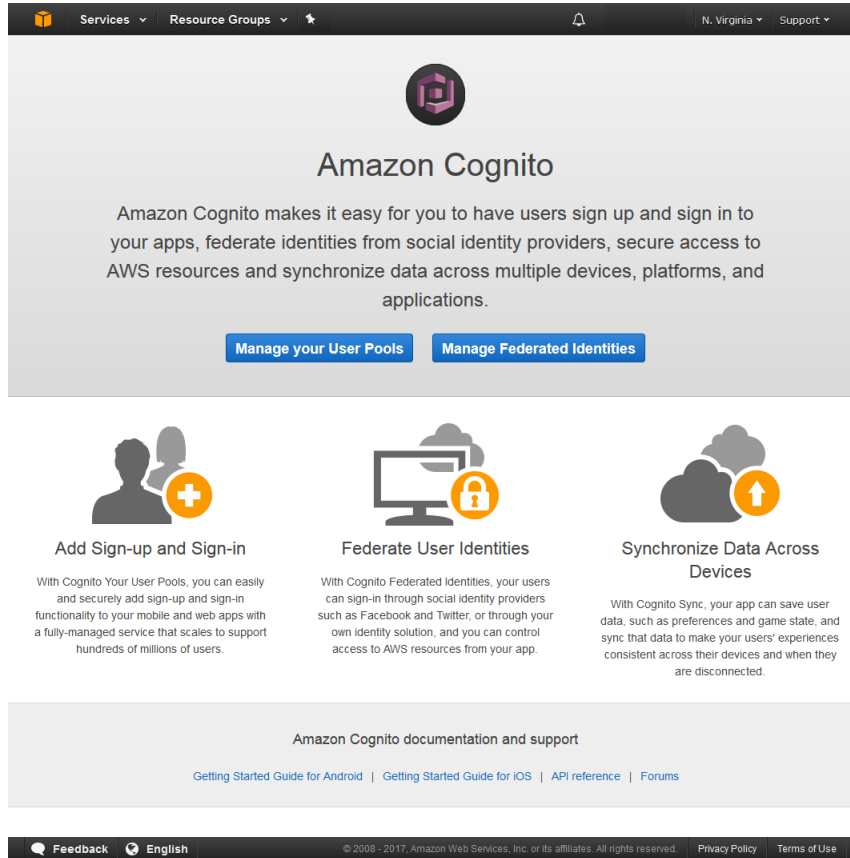
1. Amazon Cognito를 사용하려면 [AWS 계정에 가입](#)해야 합니다.
2. [Amazon Cognito 콘솔](#)로 이동합니다. AWS 자격 증명을 입력하라는 메시지가 나타날 수 있습니다.
3. 사용자 풀을 만들거나 편집하려면 사용자 풀 관리를 선택합니다.

자세한 내용은 [사용자 풀 시작하기 \(p. 13\)](#) 단원을 참조하십시오.

4. 자격 증명 풀을 생성 또는 편집하려면 연동 자격 증명 관리를 선택합니다.

자세한 내용은 [Amazon Cognito 자격 증명 시작하기\(연동 자격 증명\) \(p. 223\)](#) 단원을 참조하십시오.

Amazon Cognito 콘솔은 AWS Management 콘솔의 일부이며 계정 및 결제에 대한 정보를 제공합니다. 자세한 내용은 [AWS Management 콘솔 작업](#) 단원을 참조하십시오.



리전 데이터 고려 사항

Amazon Cognito 사용자 풀은 하나의 AWS 리전에 각각 생성되며, 해당 리전의 사용자 프로필 데이터만을 저장합니다. 옵션 기능의 구성 여부에 따라 다른 AWS 리전으로 사용자 데이터를 전송할 수 있습니다.

- Amazon Cognito 사용자 풀을 사용한 이메일 주소 라우팅 확인에 기본값인 `no-reply@verificationemail.com` 이메일 주소 설정이 사용된 경우 이메일은 미국 서부(오레곤) 리전을 통해 라우팅됩니다.
- Amazon Cognito 사용자 풀을 사용한 Amazon Simple Email Service(Amazon SES) 구성에 다른 이메일 주소가 사용된 경우 이메일 주소는 Amazon SES의 이메일 주소와 연결된 AWS 리전을 통해 라우팅됩니다.
- Amazon Cognito 사용자 풀의 SMS 메시지는 Amazon SNS 전 세계 SMS 전송의 지원 대상 중 가장 가까운 AWS 리전을 통해 라우팅됩니다.
- Amazon Cognito 사용자 풀과 함께 Amazon Pinpoint 분석이 사용되는 경우 이벤트 데이터는 미국 동부(버지니아 북부) 리전으로 라우팅됩니다.

Amazon Cognito 시작하기

이 단원에서는 주요 Amazon Cognito 작업과 시작 지점을 설명합니다. Amazon Cognito 개요는 [Amazon Cognito란 무엇입니까? \(p. 1\)](#) 단원을 참조하십시오.

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 AWS 자격 증명을 제공하여 기타 AWS 서비스에 대한 사용자 액세스 권한을 부여합니다. 사용자 풀과 자격 증명 풀을 별도로 또는 함께 사용할 수 있습니다.

주요 작업 및 시작 지점
<p>가입 및 로그인에 사용자 풀 추가</p> <ol style="list-style-type: none"> 1. 사용자 풀이 포함된 사용자 디렉터리 생성 2. 앱을 추가하여 호스팅 UI 활성화 3. 사용자 풀에 소셜 로그인 추가 4. 사용자 풀에 SAML 기반 자격 증명 공급자(IdP)를 통해 로그인을 추가합니다. 5. 사용자 풀에 OpenID Connect(OIDC) IdP를 통해 로그인을 추가합니다. 6. 사용자 풀 SDK 설치 7. 내장 호스팅 UI 로그인 및 가입 페이지 사용자 지정 8. 사용자 풀 보안 기능 구성 9. Lambda 트리거를 사용하여 사용자 풀 워크플로우 사용자 지정 10. Amazon Pinpoint 분석을 사용하여 데이터 및 대상 캠페인 수집
<p>사용자 풀에서 사용자 관리</p> <ul style="list-style-type: none"> • 가입 및 사용자 계정 확인 • 관리자로서 사용자 계정 생성 • 사용자 계정 관리 및 검색 • 사용자 풀에 그룹 추가 • 사용자 풀로 사용자 가져오기
<p>리소스 액세스</p> <p>일반적인 Amazon Cognito 시나리오:</p> <ul style="list-style-type: none"> • 사용자 풀을 통한 인증 • 사용자 풀을 통해 백엔드 리소스 액세스 • 사용자 풀을 통해 API 게이트웨이 및 Lambda 액세스 • 사용자 풀 및 자격 증명 풀을 사용하여 AWS 서비스에 액세스 • 타사 및 자격 증명 풀을 사용하여 AWS 서비스에 액세스 • 사용자 풀 또는 자격 증명 풀을 사용하여 AWS AppSync 리소스 액세스

일반적인 Amazon Cognito 시나리오

이 주제에서는 Amazon Cognito 사용에 대한 6가지 일반적인 시나리오를 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 AWS 자격 증명을 제공하여 기타 AWS 서비스에 대한 사용자 액세스 권한을 부여합니다.

사용자 풀은 Amazon Cognito의 사용자 디렉터리입니다. 앱 사용자는 사용자 풀을 통해 직접 로그인하거나 타사 자격 증명 공급자(IdP)를 통해 연동 로그인할 수 있습니다. 사용자 풀에서는 Facebook, Google 및 Amazon을 통한 소셜 로그인에서 반환된 토큰과 OpenID Connect(OIDC) 및 SAML IdP에서 반환된 토큰의 처리 작업을 관리합니다. 사용자가 직접 또는 타사를 통해 로그인하는지 여부와 무관하게 사용자 풀의 모든 멤버는 디렉터리 프로필을 보유하며, SDK를 통해 이를 액세스할 수 있습니다.

자격 증명 풀로 사용자는 임시 AWS 자격 증명을 얻어 Amazon S3 및 DynamoDB 같은 다른 AWS 서비스에 액세스할 수 있습니다. 자격 증명 풀은 익명의 게스트 사용자를 비롯해 타사 IdP를 통한 연동을 지원합니다.

주제

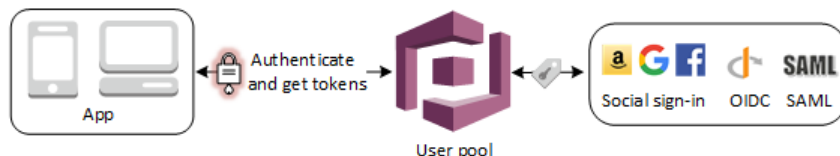
- [사용자 풀을 통한 인증 \(p. 6\)](#)
- [사용자 풀을 통해 서버 측 리소스 액세스 \(p. 6\)](#)
- [사용자 풀을 통해 API 게이트웨이 및 Lambda에서 리소스 액세스 \(p. 7\)](#)
- [사용자 풀 및 자격 증명 풀을 사용하여 AWS 서비스에 액세스 \(p. 8\)](#)
- [타사를 통한 인증 및 자격 증명 풀을 통한 AWS 서비스 액세스 \(p. 8\)](#)
- [Amazon Cognito를 사용한 AWS AppSync 리소스 액세스 \(p. 9\)](#)

사용자 풀을 통한 인증

사용자가 사용자 풀을 사용하여 인증하도록 할 수 있습니다. 앱 사용자는 사용자 풀을 통해 직접 로그인하거나 타사 자격 증명 공급자(IdP)를 통해 연동 로그인할 수 있습니다. 사용자 풀에서는 Facebook, Google 및 Amazon을 통한 소셜 로그인에서 반환된 토큰과 OpenID Connect(OIDC) 및 SAML IdP에서 반환된 토큰의 처리 작업을 관리합니다.

성공적인 인증 이후 웹 또는 모바일 앱은 Amazon Cognito로부터 사용자 풀 토큰을 받습니다. 이 토큰을 사용하여 앱이 기타 AWS 서비스에 액세스할 수 있도록 하는 AWS 자격 증명을 가져올 수 있습니다. 또는 이를 사용하여 서버 측 리소스 또는 Amazon API Gateway에 대한 액세스를 제어할 수 있습니다.

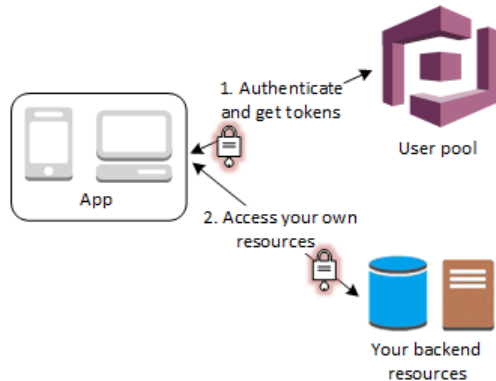
자세한 내용은 [사용자 풀 인증 흐름 \(p. 187\)](#) 및 [사용자 풀을 통해 토큰 사용 \(p. 190\)](#) 단원을 참조하십시오.



사용자 풀을 통해 서버 측 리소스 액세스

성공적인 사용자 풀 로그인 이후에 웹 또는 모바일 앱은 Amazon Cognito로부터 사용자 풀 토큰을 받습니다. 이 토큰을 사용하여 서버 측 리소스에 대한 액세스를 제어할 수 있습니다. 사용자 풀 그룹을 생성하여 권한을

관리하고 다른 유형의 사용자를 표시할 수도 있습니다. 그룹을 사용하여 리소스 액세스를 제어하는 방법은 [사용자 풀에 그룹 추가 \(p. 166\)](#) 단원을 참조하십시오.



사용자 풀에 대한 도메인이 구성되면 Amazon Cognito는 호스팅된 웹 UI를 프로비저닝하기 때문에 앱에 가입 및 로그인 페이지를 추가할 수 있습니다. 이러한 OAuth 2.0 파운데이션을 사용하여 사용자가 보호가 되는 리소스에 액세스하도록 자체 리소스 서버를 생성할 수 있습니다. 자세한 내용은 [사용자 풀의 리소스 서버 정의 \(p. 86\)](#) 단원을 참조하십시오.

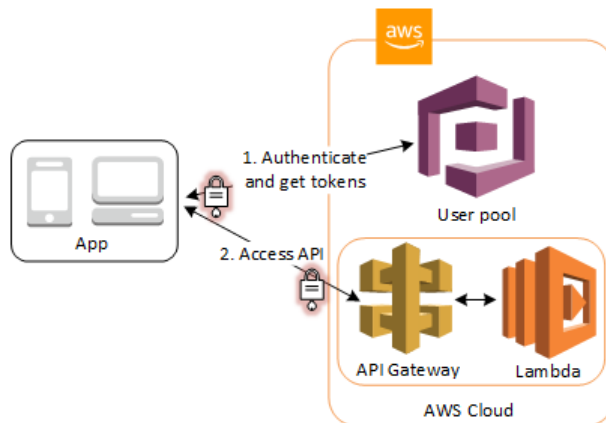
사용자 풀 인증에 대한 자세한 내용은 [사용자 풀 인증 흐름 \(p. 187\)](#) 및 [사용자 풀을 통해 토큰 사용 \(p. 190\)](#) 단원을 참조하십시오.

사용자 풀을 통해 API 게이트웨이 및 Lambda에서 리소스 액세스

사용자가 API 게이트웨이를 통해 API에 액세스하도록 할 수 있습니다. API 게이트웨이는 성공적인 사용자 풀 인증 이후 토큰을 확인하고, 이를 사용하여 Lambda 함수를 포함한 리소스 또는 고유 API에 대한 사용자 액세스 권한을 부여합니다.

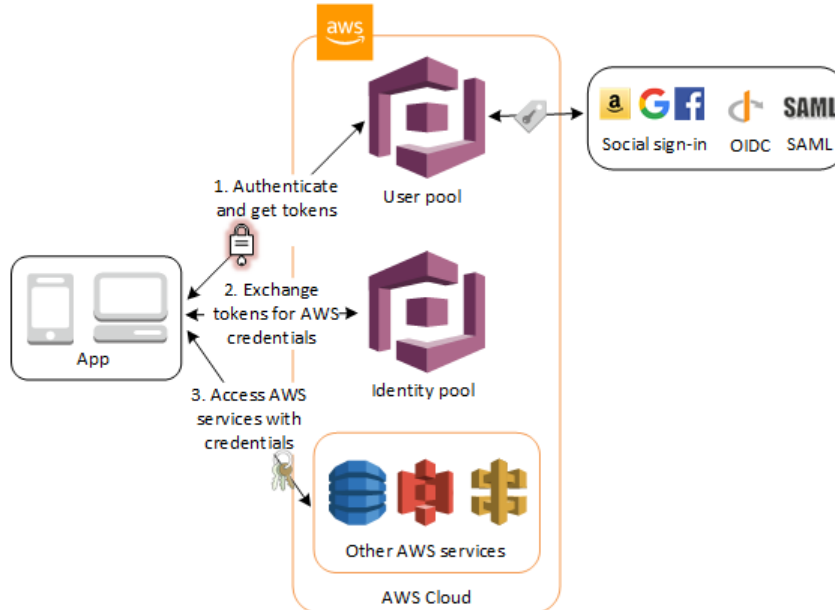
사용자 풀에 있는 그룹을 사용하여 IAM 역할로 그룹 멤버십을 매핑함으로써 API 게이트웨이를 통한 권한을 제어할 수 있습니다. 사용자가 속한 그룹은 앱 사용자가 로그인할 때 사용자 풀에서 제공한 ID 토큰에 포함됩니다. 사용자 풀 그룹에 대한 자세한 내용은 [사용자 풀에 그룹 추가 \(p. 166\)](#) 단원을 참조하십시오.

Amazon Cognito 권한 부여자 Lambda 함수에 의한 확인에 대한 API 게이트웨이로의 요청을 포함하여 사용자 풀 토큰을 제출할 수 있습니다. API 게이트웨이에 대한 자세한 내용은 [Amazon Cognito 사용자 풀에서 API 게이트웨이 사용](#)을 참조하십시오.



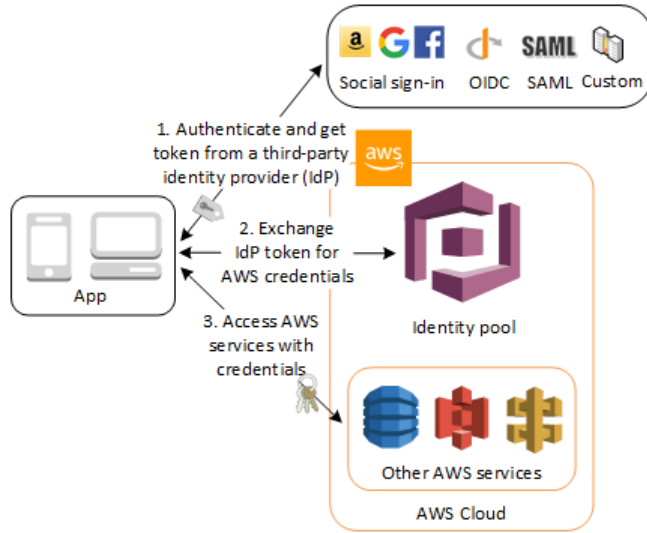
사용자 풀 및 자격 증명 풀을 사용하여 AWS 서비스에 액세스

성공적인 사용자 풀 인증 이후에 웹 또는 모바일 앱은 Amazon Cognito로부터 사용자 풀 토큰을 받습니다. 사용자는 자격 증명 풀을 사용하여 다른 AWS 서비스에 임시 액세스하도록 이 토큰을 교환할 수 있습니다. 자세한 내용은 [로그인 후 자격 증명 풀을 사용하여 AWS 서비스 액세스](#) (p. 199) 및 [Amazon Cognito 자격 증명 시작하기\(연동 자격 증명\)](#) (p. 223) 단원을 참조하십시오.



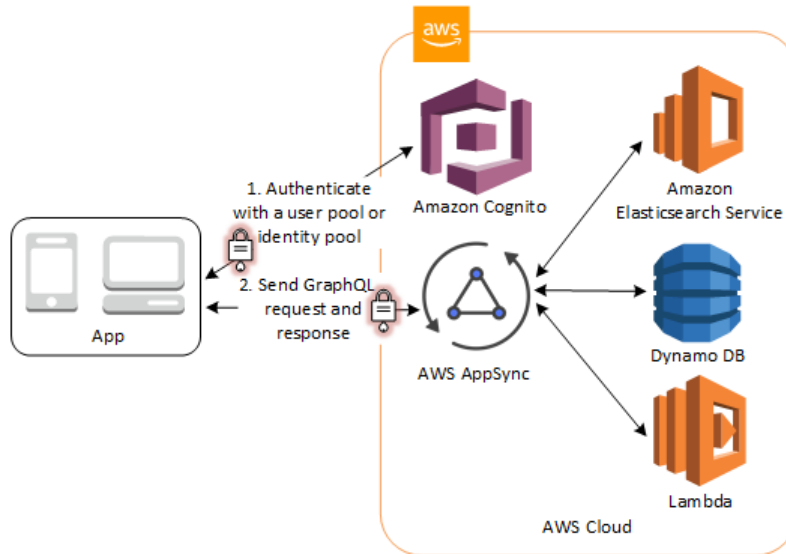
타사를 통한 인증 및 자격 증명 풀을 통한 AWS 서비스 액세스

사용자가 자격 증명 풀을 통해 AWS 서비스에 액세스하도록 할 수 있습니다. 자격 증명 풀에는 타사 자격 증명 공급자에 의해 인증된 사용자의 IdP 토큰이 필요합니다(또는 익명 게스트인 경우에는 아무 것도 필요하지 않음). 그 대신 자격 증명 풀은 일시 AWS 자격 증명을 부여하고, 이를 사용하여 기타 AWS 서비스에 액세스할 수 있습니다. 자세한 내용은 [Amazon Cognito 자격 증명 시작하기\(연동 자격 증명\)](#) (p. 223) 단원을 참조하십시오.



Amazon Cognito를 사용한 AWS AppSync 리소스 액세스

사용자 풀 또는 자격 증명 풀에서 성공적인 Amazon Cognito 인증 이후에 토큰을 사용하여 사용자에게 AWS AppSync 리소스에 대한 액세스 권한을 부여할 수 있습니다. 자세한 내용은 [사용자 풀 또는 연동 자격 증명을 사용하여 AWS AppSync 및 데이터 원본 액세스](#) 단원을 참조하십시오.



Amazon Cognito 자습서

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 AWS 자격 증명을 제공하여 기타 AWS 서비스에 대한 사용자 액세스 권한을 부여합니다.

주제

- [자습서: 사용자 풀 생성 \(p. 10\)](#)
- [자습서: 자격 증명 풀 생성 \(p. 10\)](#)
- [자습서: AWS 리소스 정리 \(p. 11\)](#)
- [자습서: 사용자 풀에 웹 또는 모바일 앱 추가 \(p. 11\)](#)

자습서: 사용자 풀 생성

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

사용자 풀을 생성하려면

1. [Amazon Cognito 콘솔](#)로 이동합니다. AWS 자격 증명을 입력하라는 메시지가 나타날 수 있습니다.
2. [Manage your User Pools]를 선택합니다.
3. 사용자 풀 생성을 선택합니다.
4. 사용자 풀 이름을 입력하고 기본값 검토를 선택하여 이름을 저장합니다.
5. 검토 페이지에서 풀 생성을 선택합니다.

관련 리소스

사용자 풀에 관한 자세한 내용은 [Amazon Cognito 사용자 풀 \(p. 12\)](#) 섹션을 참조하십시오.

[사용자 풀 인증 흐름 \(p. 187\)](#) 및 [사용자 풀을 통해 토큰 사용 \(p. 190\)](#)도 참조하세요.

자습서: 자격 증명 풀 생성

자격 증명 풀로 사용자는 임시 AWS 자격 증명을 얻어 Amazon S3 및 DynamoDB 등과 같은 다른 AWS 서비스에 액세스할 수 있습니다.

자격 증명 풀을 생성하려면

1. [Amazon Cognito 콘솔](#)로 이동합니다. AWS 자격 증명을 입력하라는 메시지가 나타날 수 있습니다.
2. 연동 자격 증명 관리를 선택합니다.
3. [Create new identity pool]을 선택합니다.
4. 자격 증명 풀의 이름을 입력합니다.
5. 인증되지 않은 자격 증명을 활성화하려면 인증되지 않은 자격 증명 축소 가능 섹션에서 인증되지 않은 자격 증명에 대한 액세스 활성화를 선택합니다.
6. [Create Pool]을 선택합니다.
7. AWS 리소스에 액세스하라는 메시지가 표시됩니다.

Allow(허용)를 선택하여 자격 증명 풀과 연결된 기본 역할 2개(인증되지 않은 사용자의 역할 1개, 인증된 사용자의 역할 1개)를 만듭니다. 이 기본 역할은 에 대한 자격 증명 풀 액세스를 제공합니다. IAM 콘솔에서 자격 증명 풀과 연결된 역할을 수정할 수 있습니다.

8. 자격 증명 풀 ID 번호를 메모합니다. 나중에 이것을 사용하여 앱 사용자들이 Amazon Simple Storage Service 또는 DynamoDB 같은 다른 AWS 서비스에 액세스하게 하는 정책을 설정합니다.

관련 리소스

자격 증명 풀에 관한 자세한 내용은 [Amazon Cognito 자격 증명 풀\(연동 자격 증명\)](#) (p. 223) 섹션을 참조하십시오.

자격 증명 풀을 사용하는 S3 예제는 [Uploading Photos to Amazon S3 from a Browser](#) 섹션을 참조하십시오.

자습서: AWS 리소스 정리

자격 증명 풀을 삭제하려면

1. [Amazon Cognito 콘솔](#)로 이동합니다. AWS 자격 증명을 입력하라는 메시지가 나타날 수 있습니다.
2. 연동 자격 증명 관리를 선택합니다.
3. 삭제할 자격 증명 풀 이름을 선택합니다. 자격 증명 풀에 대한 대시보드 페이지가 표시됩니다.
4. 대시보드 페이지의 우측 상단 모서리에서 자격 증명 풀 편집을 선택합니다. 자격 증명 풀 편집 페이지가 표시됩니다.
5. 아래로 스크롤한 후 자격 증명 풀 삭제를 선택하여 확장합니다.
6. 자격 증명 풀 삭제를 선택합니다.
7. 풀 삭제를 선택합니다.

사용자 풀을 삭제하려면

1. [Amazon Cognito 콘솔](#)로 이동합니다. AWS 자격 증명을 입력하라는 메시지가 나타날 수 있습니다.
2. [Manage your User Pools]를 선택합니다.
3. 이전 단계에서 생성한 사용자 풀을 선택합니다.
4. 앱 통합의 도메인 이름 페이지에서 도메인 삭제를 선택합니다.
5. 확인을 묻는 메시지가 표시되면 도메인 삭제를 선택합니다.
6. 일반 설정 페이지로 이동합니다.
7. 페이지 오른쪽 상단 모서리에서 풀 삭제를 선택합니다.
8. delete를 입력하고 확인을 묻는 메시지가 표시되면 풀 삭제를 선택합니다.

자습서: 사용자 풀에 웹 또는 모바일 앱 추가

JavaScript, Android 및 iOS용 자습서를 사용하여 사용자 풀에 웹 또는 모바일 앱을 추가하십시오.

- [자습서: JavaScript 애플리케이션 사용자 풀 통합](#) (p. 23)를 선택하십시오.
- [자습서: Android 앱을 위한 사용자 풀 통합](#) (p. 39)를 선택하십시오.
- [자습서: iOS 앱을 위한 사용자 풀 통합](#) (p. 58)를 선택하십시오.

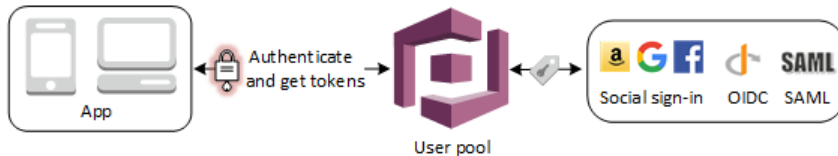
Amazon Cognito 사용자 풀

사용자 풀은 Amazon Cognito의 사용자 디렉터리입니다. 사용자 풀이 있으면 사용자는 Amazon Cognito를 통해 웹 또는 모바일 앱에 로그인할 수 있습니다. 사용자는 또한 Facebook 또는 Amazon과 같은 소셜 자격 증명 공급자를 통해, 그리고 SAML 자격 증명 공급자를 통해 로그인할 수 있습니다. 사용자가 직접 또는 타사를 통해 로그인하는지 여부와 무관하게 사용자 풀의 모든 멤버는 디렉터리 프로필을 보유하며, SDK를 통해 이를 액세스할 수 있습니다.

사용자 풀 제공 사항:

- 가입 및 로그인 서비스.
- 사용자 로그인을 위한 내장 사용자 지정 웹 UI
- Facebook, Google, Login with Amazon을 통한 소셜 로그인 및 사용자 풀의 SAML 자격 증명 공급자를 통한 로그인.
- 사용자 디렉터리 관리 및 사용자 프로필.
- 멀티 팩터 인증(MFA), 이상 있는 자격 증명 확인, 계정 탈취 보호, 전화 및 이메일 확인과 같은 보안 기능.
- AWS Lambda 트리거를 통한 사용자 지정 워크플로우 및 사용자 마이그레이션.

성공적으로 사용자를 인증한 이후 Amazon Cognito는 JSON 웹 토큰(JWT)을 발행하며, 이를 사용하여 고유 API에 대한 액세스 보안 및 자격 증명을 수행하거나 AWS 자격 증명으로 교환합니다.



Amazon Cognito는 JavaScript, Android 및 iOS용 Amazon Cognito 사용자 풀 Identity SDK를 통한 토큰 처리를 제공합니다. [사용자 풀 시작하기 \(p. 13\)](#) 및 [사용자 풀을 통해 토큰 사용 \(p. 190\)](#) 단원을 참조하십시오.

Amazon Cognito의 두 가지 주요 구성 요소는 사용자 풀과 자격 증명 풀입니다. 자격 증명 풀은 AWS 자격 증명을 제공하여 기타 AWS 서비스에 대한 사용자 액세스 권한을 부여합니다. 사용자 풀에 있는 사용자가 AWS 리소스에 액세스할 수 있도록 자격 증명 풀을 구성하여 사용자 풀 토큰을 AWS 자격 증명으로 교환할 수 있습니다. 자세한 내용은 [로그인 후 자격 증명 풀을 사용하여 AWS 서비스 액세스 \(p. 199\)](#) 및 [Amazon Cognito 자격 증명 시작하기\(연동 자격 증명\) \(p. 223\)](#) 단원을 참조하십시오.

주제

- [사용자 풀 시작하기 \(p. 13\)](#)
- [Amazon Cognito 사용자 풀에 웹 또는 모바일 앱 추가 \(p. 22\)](#)
- [Amazon Cognito 호스팅 UI를 사용하여 등록 및 로그인 \(p. 71\)](#)
- [타사를 통한 사용자 풀 로그인 추가 \(p. 88\)](#)
- [Amazon Cognito 사용자 풀 보안 관리 \(p. 107\)](#)
- [Lambda 트리거를 사용하여 사용자 풀 워크플로우 사용자 지정 \(p. 118\)](#)
- [Amazon Cognito 사용자 풀을 통해 Amazon Pinpoint 분석 사용 \(p. 155\)](#)
- [사용자 풀에서 사용자 관리 \(p. 156\)](#)
- [Amazon Cognito 사용자 풀에 대한 이메일 설정 \(p. 182\)](#)
- [사용자 풀을 사용한 인증 \(p. 185\)](#)
- [성공적인 사용자 풀 인증 이후 리소스 액세스 \(p. 197\)](#)
- [사용자 풀 참조\(AWS Management 콘솔\) \(p. 201\)](#)

사용자 풀 시작하기

이 단계는 [Amazon Cognito 콘솔](#)을 사용한 사용자 풀 설정 및 구성을 설명합니다. Amazon Cognito 시작 지침 가이드는 [Amazon Cognito 시작하기 \(p. 5\)](#) 단원을 참조하십시오.

주제

- 필수 조건: AWS 계정 가입 (p. 13)
- 단계 1. 사용자 풀이 포함된 사용자 디렉터리 생성 (p. 13)
- 단계 2. 호스팅 웹 UI를 활성화하는 앱 추가 (p. 14)
- 단계 3. 사용자 풀에 소셜 로그인 추가(옵션) (p. 15)
- 단계 4. SAML 자격 증명 공급자를 사용한 사용자 풀 로그인 추가(옵션) (p. 19)
- 단계 5. Amazon Cognito 사용자 풀 SDK 설치 (p. 20)
- 다음 단계 (p. 21)

필수 조건: AWS 계정 가입

Amazon Cognito를 사용하려면 AWS 계정이 있어야 합니다. 아직 계정이 없는 경우 다음 절차를 사용하여 가입하십시오.

AWS 계정에 가입하려면 다음을 수행합니다.

1. <https://aws.amazon.com/>을 열고 Create an AWS Account(AWS 계정 생성)를 선택합니다.

Note

전에 AWS 계정 루트 사용자 자격 증명을 사용하여 AWS Management 콘솔에 로그인한 적이 있는 경우 Sign in to a different account(다른 계정으로 로그인)를 선택합니다. 전에 IAM 자격 증명을 사용하여 콘솔에 로그인한 적이 있는 경우 Sign-in using root account credentials(루트 계정 자격 증명으로 로그인)를 선택합니다. 그런 다음 Create a new AWS account(새 AWS 계정 생성)를 선택합니다.

2. 온라인 지시 사항을 따릅니다.

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

다음 단계

[단계 1. 사용자 풀이 포함된 사용자 디렉터리 생성 \(p. 13\)](#)

단계 1. 사용자 풀이 포함된 사용자 디렉터리 생성

Amazon Cognito 사용자 풀을 사용하여 사용자 디렉터리를 생성 및 유지 관리하고 모바일 앱 또는 웹 애플리케이션에 가입 및 로그인을 추가할 수 있습니다.

사용자 풀을 생성하려면

1. [Amazon Cognito 콘솔](#)로 이동합니다. AWS 자격 증명을 입력하라는 메시지가 나타날 수 있습니다.
2. [Manage your User Pools]를 선택합니다.
3. 페이지 오른쪽 상단에서 사용자 풀 생성을 선택합니다.
4. 사용자 풀 이름을 입력하고 기본값 검토를 선택하여 이름을 저장합니다.
5. 속성 페이지에서 이메일 주소 또는 전화번호와 이메일 주소 허용을 선택합니다.

6. 페이지 하단에서 다음 단계를 선택하여 속성을 저장합니다.
7. 페이지 왼쪽에 있는 탐색 모음에서 검토를 선택합니다.
8. 검토 페이지 하단에서 풀 생성을 선택합니다.

다음 단계

단계 2. 호스팅 웹 UI를 활성화하는 앱 추가 (p. 14)

단계 2. 호스팅 웹 UI를 활성화하는 앱 추가

사용자 풀을 생성한 이후 앱을 생성하여 사용자 가입 및 로그인에 대해 내장 웹 페이지를 사용할 수 있습니다.

사용자 풀에 앱을 생성하려면

1. [Amazon Cognito 콘솔](#)로 이동합니다. AWS 자격 증명을 입력하라는 메시지가 나타날 수 있습니다.
2. [Manage your User Pools]를 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 [사용자 풀을 생성합니다](#).
4. 페이지 왼쪽에 있는 탐색 모음의 일반 설정에서 앱 클라이언트를 선택합니다.
5. [Add an app client]를 선택합니다.
6. 앱 이름을 지정합니다.
7. 이 시작하기 연습을 위한 클라이언트 보안키 생성 옵션을 지우십시오. 클라이언트 측 JavaScript를 사용하여 URL에서 전송할 경우 안전하지 않을 수 있습니다. 클라이언트 암호는 클라이언트 암호를 보호할 수 있는 서버 측 구성요소가 있는 애플리케이션에서 사용됩니다.
8. [Create app client]를 선택합니다.
9. 앱 클라이언트 ID를 메모합니다.
10. 풀 세부 정보로 돌아가기를 선택합니다.
11. 앱을 구성합니다.

- a. 콘솔 페이지 왼쪽에 있는 탐색 모음에서 앱 클라이언트 설정을 선택합니다.
- b. 활성화된 자격 증명 공급자 중 하나로 Cognito User Pool(Cognito 사용자 풀)을 선택합니다.

Note

Facebook, Amazon 및 Google 같은 외부 자격 증명 공급자(IdP)를 비롯해 OpenID Connect(OIDC) 또는 SAML IdP를 통해 로그인하도록 허용하려면 먼저 다음 단원에 설명된 대로 이들을 구성한 다음, 앱 클라이언트 설정으로 돌아가 이들을 활성화합니다.

- c. Amazon Cognito 권한 부여 서버에 대한 콜백 URL을 입력하여 사용자가 인증된 후 호출합니다. 웹 앱의 경우 URL이 `https://`로 시작해야 합니다(예: `https://www.example.com`).
- iOS 또는 Android 앱의 경우에는 `myapp://`와 같은 콜백 URL을 사용할 수 있습니다.
- d. Authorization code grant(권한 부여 코드 허용)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. 보안을 위해 모바일 앱에서는 [Proof Key for Code Exchange \(PKCE\)](#)와 함께 권한 부여 코드 허용 흐름을 사용하는 것이 좋습니다.

허용된 OAuth Flows에서 Implicit grant(암시적 허용)를 선택하여 Amazon Cognito로부터 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.

Authorization code grant(권한 부여 코드 허용) 및 Implicit code grant(암시적 코드 부여)를 모두 활성화하고 각각을 필요한 경우 사용할 수 있습니다.

특별히 하나를 제외하고 싶은 경우가 아니라면 허용된 OAuth Scopes 확인란을 모두 선택합니다.

Note

사용자를 대신해서가 아니라 자체적으로 액세스 토큰을 요청해야 하는 경우에만 Client credentials(클라이언트 자격 증명)를 선택합니다.

- e. [Save changes]를 선택합니다.
12. 사용자 풀 도메인을 구성합니다.
- a. 도메인 이름 페이지에서 사용 가능한 도메인 접두사를 입력합니다.
 - b. 전체 도메인 주소를 기록해 둡니다.
 - c. [Save changes]를 선택합니다.

로그인 페이지를 보려면

다음 URL을 사용하여 호스팅 UI 로그인 웹 페이지를 볼 수 있습니다. `response_type`을 기록합니다. 이 경우 `response_type=code`는 인증 코드 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=token`은 묵시적 코드 부여입니다. 로그인 성공 이후 는 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.

묵시적 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
```

AWS Lambda를 사용하여 사용자 풀 토큰을 디코딩 및 확인할 수 있습니다. AWS GitHub 웹 사이트에서 [Decode and verify Amazon Cognito JWT tokens](#)를 참조하십시오.

Amazon Cognito 사용자 풀 토큰은 RS256 알고리즘을 사용하여 서명됩니다.

콘솔 변경 사항이 표시되기 전까지 브라우저를 새로 고침하는 데 잠시 기다려야 할 수 있습니다.

도메인 이름 페이지에 도메인이 표시됩니다. 앱 클라이언트 설정 페이지에 앱 클라이언트 ID와 콜백 URL이 표시됩니다.

다음 단계

단계 3. 사용자 풀에 소셜 로그인 추가(옵션) (p. 15)

단계 3. 사용자 풀에 소셜 로그인 추가(옵션)

앱 사용자가 Facebook, Google 및 Login with Amazon과 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 건너뛰십시오.

1단계: 소셜 IdP로 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. My Apps(내 앱) 메뉴에서 Create New App(새 앱 생성)을 선택합니다.
4. Facebook 앱 이름을 지정한 다음 Create App ID(앱 ID 생성)를 선택합니다.
5. 왼쪽 탐색 모음에서 설정과 기본을 차례로 선택합니다.
6. App ID(앱 ID)와 앱 보안을 메모합니다. 다음 섹션에서 이 둘을 사용합니다.
7. 페이지 하단에서 + Add Platform(플랫폼 추가)을 선택합니다.
8. 웹 사이트를 선택합니다.
9. 웹 사이트에서 엔드포인트가 /oauth2/idpresponse인 사용자 풀 도메인을 Site URL(사이트 URL)에 입력합니다.

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

10. [Save changes]를 선택합니다.
11. App Domains(앱 도메인)에 사용자 풀 도메인을 입력합니다.

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

12. [Save changes]를 선택합니다.
13. 탐색 모음에서 제품을 선택하고 Facebook 로그인에서 Set up(설정)을 선택합니다.
14. 탐색 모음에서 Facebook 로그인과 설정을 차례로 선택합니다.

Valid OAuth Redirect URIs(유효 OAuth 리디렉션 URI)에 리디렉션 URL을 입력합니다. 엔드포인트가 /oauth2/idpresponse인 사용자 풀 도메인으로 이루어집니다.

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

15. [Save changes]를 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 Amazon에 로그인을 선택합니다.

4. Create a Security Profile(보안 프로필 생성)을 선택합니다.
5. Security Profile Name(보안 프로필 이름), Security Profile Description(보안 프로필 설명) 및 Consent Privacy Notice URL(개인 정보 보호 정책 URL 동의)에 입력합니다.

6. Save를 선택합니다.
7. 클라이언트 ID 및 클라이언트 암호를 선택하여 클라이언트 ID 및 암호를 표시합니다. 다음 섹션에서 이 둘을 사용합니다.
8. 기어를 마우스로 가리켜 Web Settings(웹 설정) 탭을 선택하고 편집을 선택합니다.
9. Allowed Origins(허용되는 오리진)에 사용자 풀 도메인을 입력합니다.

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

10. 엔드포인트가 /oauth2/idpresponse인 사용자 풀 도메인을 Allowed Return URLs(허용되는 반환 URL)에 입력합니다.

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

11. Save를 선택합니다.

Google로 앱을 등록하려면

1. Google에서 [개발자 계정을 생성](#)합니다.
2. Google 자격 증명으로 [로그인](#)합니다.
3. CONFIGURE A PROJECT(프로젝트 구성)를 선택합니다.
4. 프로젝트 이름을 입력하고 다음을 선택합니다.
5. 사용자의 프로젝트 이름을 입력하고 다음을 선택합니다.
6. Where are you calling from?(호출 위치는 어디입니까?) 드롭다운 목록에서 Web browser(웹 브라우저)를 선택합니다.
7. Authorized JavaScript origins(권한 있는 JavaScript 오리진)에 사용자 풀 도메인을 입력합니다.

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

8. CREATE를 선택합니다. 이 단계에서는 클라이언트 ID와 클라이언트 보안을 사용하지 않습니다.
9. DONE을 선택합니다.
10. Google 콘솔에 [로그인](#)합니다.
11. 왼쪽 탐색 모음에서 Credentials(자격 증명)를 선택합니다.
12. 자격 증명 생성 드롭다운 목록에서 OAuth client ID(OAuth 클라이언트 ID)를 선택하여 OAuth 2.0 자격 증명을 생성합니다.
13. Web application(웹 애플리케이션)을 선택합니다.
14. Authorized JavaScript origins(권한 있는 JavaScript 오리진)에 사용자 풀 도메인을 입력합니다.

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

15. /oauth2/idpresponse 엔드포인트가 있는 사용자 풀 도메인을 Authorized Redirect URIs(인증 리디렉션 URI)에 입력합니다.

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

16. Create(생성)를 두 번 선택합니다.
17. OAuth 클라이언트 ID와 클라이언트 암호를 메모합니다. 다음 섹션에서 이 둘이 필요합니다.
18. 확인을 선택합니다.

2단계: 사용자 풀에 소셜 IdP 추가

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

AWS Management 콘솔을 사용하여 사용자 풀 소셜 자격 증명 공급자를 구성하려면

1. [Amazon Cognito 콘솔](#)로 이동합니다. AWS 자격 증명을 입력하라는 메시지가 나타날 수 있습니다.
2. [Manage your User Pools]를 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 [사용자 풀을 생성합니다](#).
4. 왼쪽 탐색 모음에서 자격 증명 공급자를 선택합니다.
5. Facebook, Google, Login with Amazon 등 소셜 자격 증명 공급자를 선택합니다.
6. 이전 섹션의 소셜 자격 증명 공급자에게서 받은 앱 클라이언트 ID와 앱 클라이언트 암호를 입력합니다.
7. 승인하려는 범위의 이름을 입력합니다. 범위는 앱에서 액세스하고자 하는 사용자 속성(예: name 및 email)을 정의합니다. Facebook의 경우 심포로 구분해야 합니다. Google 및 Login with Amazon의 경우 공백으로 구분해야 합니다.

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

앱 사용자는 앱에 이러한 속성들을 제공한다는 데 동의하라는 요청 메시지를 받게 됩니다. 범위에 대한 자세한 내용은 Google, Facebook 및 Login with Amazon의 설명서를 참조하십시오.

8. 구성 중인 소셜 자격 증명 공급자에 대해 Enable(활성화)을 선택합니다.
9. 탐색 모음에서 앱 클라이언트 설정을 선택합니다.
10. 사용자 풀 앱의 활성화된 자격 증명 공급자 중 하나로 소셜 자격 증명 공급자를 선택합니다.
11. 사용자 풀 앱의 콜백 URL(여러 개 가능)에 콜백 URL을 입력합니다. 이것은 로그인 성공 후 사용자가 리디렉션되는 페이지의 URL입니다.

`https://www.example.com`

12. [Save changes]를 선택합니다.
13. 속성 매핑 탭에서 다음과 같이 최소한의 필수 속성(일반적으로 email)에 대한 매핑을 추가합니다.
 - a. 확인란을 선택하여 Facebook, Google 또는 Amazon 속성 이름을 선택합니다. Amazon Cognito 콘솔에 올라와 있지 않은 추가 속성의 이름을 입력할 수도 있습니다.
 - b. 드롭다운 목록에서 대상 사용자 풀 속성을 선택합니다.
 - c. [Save changes]를 선택합니다.
 - d. 요약본으로 이동을 선택합니다.

3단계: 소셜 IdP 구성 테스트

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

```
https://your_user_pool_domain/login?  
response_type=code&client_id=your_client_id&redirect_uri=https://www.example.com
```

사용자 풀 도메인 이름 콘솔 페이지에서 사용자의 도메인을 찾을 수 있습니다. client_id는 앱 클라이언트 설정 페이지에 있습니다. redirect_uri 파라미터용 콜백 URL을 사용합니다. 이 콜백 URL 주소는 사용자 풀 도메인과 다릅니다.

다음 단계

단계 4. SAML 자격 증명 공급자를 사용한 사용자 풀 로그인 추가(옵션) (p. 19)

단계 4. SAML 자격 증명 공급자를 사용한 사용자 풀 로그인 추가(옵션)

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 건너뛰십시오.

SAML 자격 증명 공급자를 업데이트하고 사용자 풀을 구성해야 합니다. SAML 2.0 자격 증명 공급자에 대한 신뢰 당사자 또는 애플리케이션으로서 사용자 풀을 추가하는 방법에 대한 내용은 SAML 자격 증명 공급자 설명서를 참조하십시오.

그리고 SAML 자격 증명 공급자에 어설션 소비자 엔드포인트를 제공해야 합니다. SAML 2.0 POST에 대해 이 엔드포인트를 구성하여 SAML 자격 증명 공급자를 바인딩합니다.

```
https://<yourDomainPrefix>.auth.<region>.amazoncognito.com/saml2/idpresponse
```

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

일부 SAML 자격 증명 공급자의 경우 SP urn /대상 URI / SP 개체 ID를 다음 양식으로 제공해야 합니다.

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

Amazon Cognito 콘솔의 앱 클라이언트 설정 탭에서 사용자 풀 ID를 찾을 수 있습니다.

사용자 풀에 필요한 속성에 대한 속성 값을 제공하려면 SAML 자격 증명 공급자도 구성해야 합니다. 일반적으로 email은 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 연동을 지원합니다. 사용자 풀이 사용자 에이전트를 통해 자격 증명 공급자로부터 직접 SAML 응답을 받으므로 앱에서 SAML 어설션 응답을 수신하거나 구문 분석을 할 필요가 없습니다.

사용자 풀에 SAML 2.0 자격 증명 공급자를 구성하려면

1. Amazon Cognito 콘솔로 이동합니다. AWS 자격 증명을 입력하라는 메시지가 나타날 수 있습니다.
2. [Manage your User Pools]를 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. 왼쪽 탐색 모음에서 자격 증명 공급자를 선택합니다.
5. SAML을 선택하여 SAML 대화상자를 엽니다.
6. 메타데이터 문서에서 사용자 SAML IdP의 메타데이터 문서를 업로드합니다. 메타데이터 문서를 가리키는 URL을 입력할 수도 있습니다. 자세한 내용은 타사 SAML 자격 증명 공급자와 Amazon Cognito 사용자 풀 통합 (p. 98) 단원을 참조하십시오.

Note

Amazon Cognito에서 메타데이터를 자동으로 새로 고칠 수 있으므로 퍼블릭 엔드포인트인 경우 파일을 업로드하지 않고 엔드포인트 URL을 제공하는 것이 좋습니다. 일반적으로 메타데이터 새로 고침은 6시간마다 또는 메타데이터가 만료되기 전 중 더 빠른 시간에 발생합니다.

7. 사용자의 SAML 공급자 이름을 입력합니다. SAML 명명에 관한 자세한 내용은 [SAML 자격 증명 공급자 이름 선택 \(p. 94\)](#) 섹션을 참조하십시오.
8. 사용할 SAML 식별자(선택사항)를 입력합니다.
9. Amazon Cognito에서 로그아웃할 때 SAML IdP에서 사용자를 로그아웃되도록 하려면 Enable IdP sign out flow(IdP 로그아웃 흐름 활성화)를 선택합니다.

이 흐름을 활성화하면 [로그아웃 엔드포인트 \(p. 326\)](#)가 호출될 때 서명된 로그아웃 요청이 SAML IdP로 전송됩니다.

IdP의 로그아웃 응답을 사용하도록 이 엔드포인트를 구성합니다. 이 엔드포인트는 사후 바인딩을 사용합니다.

```
https://<yourDomainPrefix>.auth.<region>.amazoncognito.com/saml2/logout
```

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.

10. 공급자 생성을 선택합니다.
11. 속성 매핑 탭에서 다음과 같이 최소한의 필수 속성(일반적으로 email)에 대한 매핑을 추가합니다.
 - a. 자격 증명 공급자의 SAML 어설션에 표시된 대로 SAML 속성 이름을 입력합니다. 자격 증명 공급자가 예제 SAML 어설션을 제공하는 경우 이름을 찾는 데 도움이 될 수 있습니다. 일부 자격 증명 공급자는 email과 같은 간단한 이름을 사용하지만 다른 자격 증명 공급자는 다음과 비슷한 이름을 사용합니다.

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

- b. 드롭다운 목록에서 대상 사용자 풀 속성을 선택합니다.
12. [Save changes]를 선택합니다.
 13. 요약본으로 이동을 선택합니다.

자세한 내용은 [사용자 풀에 SAML 자격 증명 공급자 추가 \(p. 92\)](#) 단원을 참조하십시오.

다음 단계

[단계 5. Amazon Cognito 사용자 풀 SDK 설치 \(p. 20\)](#)

단계 5. Amazon Cognito 사용자 풀 SDK 설치

Amazon Cognito는 상호 보완적인 두 가지 사용자 풀 SDK를 제공합니다.

Amazon Cognito Identity SDK는 코어 사용자 풀 라이브러리입니다. 이를 설치하면 API의 사용자 관리 및 인증 기능과 상호 작용합니다.

Amazon Cognito Auth SDK는 내장 호스팅 UI 웹 페이지를 활용합니다. 이를 설치하면 앱에 가입, 로그인, 확인, 멀티 팩터 인증(MFA) 및 로그아웃에 대한 웹 페이지를 추가합니다.

JavaScript

사용자 풀 SDK	샘플 코드	Documentation
JavaScript용 Identity SDK.	Identity SDK 샘플 AWS Mobile React 스타터 키트	Amazon Cognito 사용자 풀에 JavaScript 앱 추가 (p. 22) AWS Amplify 라이브러리
JavaScript용 Auth SDK	Auth SDK 샘플 앱	Amazon Cognito 사용자 풀 Auth API 참조 (p. 316)

Android

사용자 풀 SDK	샘플 코드	설명서
Android용 Identity SDK.	Identity SDK 샘플 Android용 Mobile SDK 샘플	Amazon Cognito 사용자 풀에 Android 앱 추가 (p. 37) Android용 Mobile SDK 설명서
Android용 Auth SDK.	Auth SDK 샘플 앱	Amazon Cognito 사용자 풀 Auth API 참조 (p. 316)

iOS

사용자 풀 SDK	샘플 코드	설명서
iOS용 Identity SDK.	Identity SDK 샘플 Mobile SDK for iOS 샘플	Amazon Cognito 사용자 풀에 iOS 앱 추가 (p. 57) AWS Mobile SDK for iOS 설명서
iOS용 Auth SDK에는 다음 항목이 포함됩니다. Mobile SDK for iOS.	Auth SDK 샘플 앱	Amazon Cognito 사용자 풀 Auth API 참조 (p. 316)

기타 AWS SDK

기타 SDK 및 설명서는 [AWS SDK](#) 단원을 참조하십시오.

다음 단계

이제 사용자 풀을 생성하고 SDK를 설치했습니다. 다음 작업은 다음과 같습니다.

JavaScript, Android 및 iOS 앱에 Amazon Cognito를 설정하는 방법을 알아봅니다.

- [Amazon Cognito 사용자 풀에 JavaScript 앱 추가 \(p. 22\)](#)
- [Amazon Cognito 사용자 풀에 Android 앱 추가 \(p. 37\)](#)
- [Amazon Cognito 사용자 풀에 iOS 앱 추가 \(p. 57\)](#)

사용자 풀 기능 자세히 알아보기:

- 내장 로그인 및 가입 웹페이지 사용자 지정 (p. 82)
- 사용자 풀에 멀티 팩터 인증(MFA) 추가 (p. 107)
- 사용자 풀에 고급 보안 기능 추가 (p. 110)
- Lambda 트리거를 사용하여 사용자 풀 워크플로우 사용자 지정 (p. 118)
- Amazon Cognito 사용자 풀을 통해 Amazon Pinpoint 분석 사용 (p. 155)

Amazon Cognito 인증 및 권한 부여 사례에 대한 개요는 [일반적인 Amazon Cognito 시나리오](#) (p. 6) 단원을 참조하십시오.

사용자 풀 인증 성공 이후 기타 AWS 서비스에 액세스하려면 [로그인 후 자격 증명 풀을 사용하여 AWS 서비스 액세스](#) (p. 199) 단원을 참조하십시오.

AWS Management 콘솔 및 이 단원에서 앞서 언급한 사용자 풀 SDK 외에도 [AWS Command Line Interface](#)를 사용하여 사용자 풀 프로필과 상호 작용할 수 있습니다.

Amazon Cognito 사용자 풀에 웹 또는 모바일 앱 추가

Amazon Cognito 사용자 풀로 웹과 모바일 앱 사용자들이 가입하고 로그인하도록 할 수 있습니다. 인증 사용자에게 대해 암호를 변경할 수 있고 미인증 사용자에게 대해 잊어버린 암호 흐름을 시작할 수 있습니다.

Amazon Cognito 사용자 풀은 JavaScript, Android 및 iOS용 Identity SDK를 제공합니다. 이를 사용하여 성공적인 사용자 풀 인증 이후에서 반환한 서명된 JSON 웹 토큰(JWT)을 저장 및 관리할 수 있습니다. 토큰에 대한 자세한 내용은 [사용자 풀을 통해 토큰 사용](#) (p. 190) 단원을 참조하십시오.

주제

- [Amazon Cognito 사용자 풀에 JavaScript 앱 추가](#) (p. 22)
- [Amazon Cognito 사용자 풀에 Android 앱 추가](#) (p. 37)
- [Amazon Cognito 사용자 풀에 iOS 앱 추가](#) (p. 57)

Amazon Cognito 사용자 풀에 JavaScript 앱 추가

Amazon Cognito 사용자 풀로 웹과 모바일 앱 사용자들이 가입하고 로그인하도록 할 수 있습니다. 인증 사용자에게 대해 암호를 변경할 수 있고 미인증 사용자에게 대해 잊어버린 암호 흐름을 시작할 수 있습니다. 다음 단원에서는 [Amazon Cognito 자격 증명 공급자](#)를 사용하여 사용자 풀 프로필을 검색 및 업데이트할 수 있도록 설정 정보와 예시를 제공합니다.

Amazon Cognito 자격 증명 공급자를 포함하는 웹용 AWS Amplify 라이브러리를 사용하여 JavaScript 기반 웹 애플리케이션에 사용자 풀을 추가합니다. 자세한 내용은 [웹용 AWS Amplify 라이브러리 설정](#)을 참조하십시오. 또한 [AWS Amplify Library 인증 가이드](#)를 참조하십시오.

Note

[JavaScript용 Amazon Cognito Identity SDK](#)는 이제 [AWS Amplify Library](#)의 일부입니다.

주제

- [자습서: JavaScript 앱용 사용자 풀 통합](#) (p. 23)
- [예제: JavaScript SDK 사용](#) (p. 26)
- [예제: SDK for JavaScript에서 AdminCreateUser API를 통해 생성한 사용자의 새 암호를 인증 및 설정](#) (p. 36)

- 예제: Lambda 트리거를 사용한 JavaScript 사용자 마이그레이션 (p. 36)

자습서: JavaScript 앱용 사용자 풀 통합

본 자습서는 JavaScript용 Amazon Cognito SDK를 사용하여 사용자 풀을 시작하는 데 도움이 됩니다.

주제

- 1단계: 콘솔을 사용하여 JavaScript 앱용 사용자 풀 생성 (p. 23)
- 2단계: 앱에서 사용자 풀 객체 생성 (p. 24)
- 3단계: 앱에 사용자 가입 (p. 24)
- 4단계: 앱에 대해 사용자 확인 (p. 24)
- 5단계: 앱에 사용자 로그인 (p. 25)
- 6단계: 사용자 세부 정보 가져오기 (p. 25)
- 7단계: 앱 사용자를 위해 AWS 리소스에 액세스하기 위한 자격 증명 가져오기 (p. 26)
- 다음 단계 (p. 26)

1단계: 콘솔을 사용하여 JavaScript 앱용 사용자 풀 생성

다음 절차에서는 사용자 풀을 생성하고 앱에서 사용자 풀을 사용하는 방법을 설명합니다. 이 절차에서는 사용자 풀 ID와 앱 클라이언트 ID를 만듭니다. 이 설정의 사용자 지정에 대한 자세한 내용은 [사용자 풀 참조 \(AWS Management 콘솔\)](#) (p. 201) 단원을 참조하십시오.

앱을 위한 사용자 풀을 생성하려면

1. [Amazon Cognito 콘솔](#)에 로그인합니다.
2. [Manage your User Pools]를 선택합니다.
3. 사용자 풀 생성을 선택합니다.
4. 풀 이름에서 풀에 대한 이름을 입력하고 기본값 검토를 선택합니다. 그러면 기본 설정을 사용해 풀이 생성됩니다.
5. 왼쪽 탐색 창에서 속성을 선택하여 필요한 속성과 별칭으로 사용할 속성을 지정합니다. 다음 속성을 설정한 후 풀의 사용자가 이메일 주소를 확인하면 사용자 이름이나 이메일 주소를 통해 로그인할 수 있습니다.
 - a. email의 경우 필수 사항 및 Alias(별칭)를 선택하십시오.
 - b. phone number의 경우 필수 사항 및 Alias(별칭)를 선택하십시오.
 - c. given name의 경우 필수 사항을 선택하십시오.
 - d. [Save changes]를 선택합니다.
6. 왼쪽 탐색 창에서 정책을 선택하여 암호 정책을 지정합니다. 이 자습서에서는 기본 설정을 사용합니다.
7. 왼쪽 탐색 창에서 메시지 사용자 지정을 선택합니다. 이 페이지에서는 확인 코드를 전달하기 위해 풀의 사용자에게 전송되는 메시지를 사용자 지정할 수 있습니다. 이 자습서에서는 기본 설정을 사용합니다.
8. 왼쪽 탐색 창에서 앱 클라이언트를 선택한 다음 앱 클라이언트 추가를 선택합니다. 사용자 풀에 여러 앱 클라이언트를 생성할 수 있습니다.
9. App name(앱 이름)의 경우 앱 이름을 입력합니다. 클라이언트 보안키 생성 확인란이 지워져 있는지 확인한 다음 속성 읽기 및 쓰기 권한 설정을 선택합니다. 쓰기 권한이 필요한 속성을 선택합니다. 필수 속성에는 항상 쓰기 권한이 있습니다.

Note

Amazon Cognito JavaScript SDK에서는 앱 클라이언트 비밀번호를 사용하지 않습니다. 앱 클라이언트 비밀번호를 사용하여 사용자 풀 앱 클라이언트를 구성할 경우 SDK에서 예외가 발생합니다.

10. Create app(앱 생성)과 Save changes(변경 사항 저장)를 차례대로 선택합니다.

11. 왼쪽 탐색 모음에서 검토와 풀 생성을 차례대로 선택합니다.
12. 풀 ID와 클라이언트 ID를 기록합니다. 왼쪽 탐색 모음에 있는 앱 클라이언트에서 앱 클라이언트 ID를 찾을 수 있습니다.

2단계: 앱에서 사용자 풀 객체 생성

사용자 풀 객체를 만들려면 1단계에서 얻은 클라이언트 ID와 사용자 풀 ID가 필요합니다. 다음 예에서는 CognitoUserPool 객체를 생성하는 방법을 보여줍니다. JavaScript SDK에서는 앱 클라이언트 비밀번호를 지원하지 않습니다. 앱 클라이언트 비밀번호를 사용하여 사용자 풀 앱 클라이언트를 구성할 경우 SDK에서 예외가 발생합니다.

```
var poolData = {
  UserPoolId : '...', // your user pool id here
  ClientId : '...' // your app client id here
};
var userPool =
new AmazonCognitoIdentity.CognitoUserPool(poolData);
var userData = {
  Username : '...', // your username here
  Pool : userPool
};
```

3단계: 앱에 사용자 가입

사용자 풀 객체를 만든 후 사용자는 앱에 가입할 수 있습니다. 사용자의 정보는 웹 UI를 통해 수집할 수 있으며 CognitoUserAttribute 호출에서 통과되는 signUp 객체를 채우는 데 사용할 수 있습니다.

```
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 attributeEmail =
new AmazonCognitoIdentity.CognitoUserAttribute(dataEmail);
var attributePhoneNumber =
new AmazonCognitoIdentity.CognitoUserAttribute(dataPhoneNumber);

attributeList.push(attributeEmail);
attributeList.push(attributePhoneNumber);

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());
});
```

4단계: 앱에 대해 사용자 확인

가입한 후 사용자는 사용자 풀 설정에 따라 SMS 또는 이메일을 통해 전송된 코드를 입력하여 가입을 확인합니다. 또는 PreSignUp AWS Lambda 기능을 사용하여 사용자를 자동으로 확인할 수 있습니다. 가입을 확인하려면 사용자가 받은 코드(다음 예에서는 '123456')를 수집하고 다음과 같이 해당 코드를 사용해야 합니다.

```
cognitoUser.confirmRegistration('123456', true, function(err, result) {
  if (err) {
    alert(err);
    return;
  }
  console.log('call result: ' + result);
});
```

resendConfirmationCode 객체의 cognitoUser 메서드를 사용하여 등록 코드를 재전송할 수 있습니다. 이는 미인증 호출이며 사용자 이름, 클라이언트 ID 및 사용자 풀 정보만 필요합니다.

5단계: 앱에 사용자 로그인

확인된 사용자는 로그인하여 세션을 얻습니다. 세션에는 사용자 클레임이 포함된 ID 토큰과 인증된 호출을 수행하기 위해 내부적으로 사용되는 액세스 토큰 및 각 시간이 만료된 후 세션을 새로 고치기 위해 내부적으로 사용되는 새로 고침 토큰이 있습니다. 토큰에 대한 자세한 내용은 [사용자 풀을 통해 토큰 사용 \(p. 190\)](#)을 참조하십시오. 성공적으로 로그인한 경우 onSuccess 콜백이 호출됩니다. 로그인하지 못한 경우 onFailure 콜백이 호출됩니다. 로그인할 때 MFA가 필요한 경우 mfaRequired 콜백이 호출되며 sendMFACode 객체에 대해 cognitoUser를 호출해야 합니다. 수신된 확인 코드가 전달된 다음 사용자가 로그인해야 합니다.

```
var authenticationData = {
  Username : '...', // your username here
  Password : '...', // your password here
};
var authenticationDetails =
new AmazonCognitoIdentity.AuthenticationDetails(authenticationData);

var cognitoUser =
new AmazonCognitoIdentity.CognitoUser(userData);
cognitoUser.authenticateUser(authenticationDetails, {
  onSuccess: function (result) {
    var accessToken = result.getAccessToken().getJwtToken();
  },

  onFailure: function(err) {
    alert(err);
  },
  mfaRequired: function(codeDeliveryDetails) {
    var verificationCode = prompt('Please input verification code' , '');
    cognitoUser.sendMFACode(verificationCode, this);
  }
});
```

6단계: 사용자 세부 정보 가져오기

로그인 후 사용자는 사용자 속성 검색, 사용자 속성 확인(예: 확인되지 않은 이메일 주소), 사용자 속성 삭제, 사용자 속성 업데이트, 사용자 암호 변경, 사용자 계정 삭제 등 권한 있는 작업을 수행할 수 있습니다. 선택 사항인 MFA 설정이 있는 사용자 풀의 경우 사용자는 MFA를 활성화 또는 비활성화할 수 있습니다. 앱에서 로그아웃하면 로컬 사용자 세션이 지워지며 새 세션을 설정하려면 사용자가 다시 로그인해야 합니다.

사용자가 암호를 잊어버린 경우 잊어버린 암호 흐름을 시작할 수 있습니다. 코드가 사용자에게 전송됩니다. 사용자는 새 암호와 함께 이 코드를 사용하여 흐름을 완료할 수 있습니다. 관련된 호출은 미인증된 forgotPassword 객체에 대한 cognitoUser이며 관련 콜백이 다음 예에 표시됩니다.

```
cognitoUser.forgotPassword({
  onSuccess: function (result) {
    console.log('call result: ' + result);
  },
  onFailure: function(err) {
```



```
        alert(err);
    },
    inputVerificationCode() {
        var verificationCode = prompt('Please input verification code ' , '');
        var newPassword = prompt('Enter new password ' , '');
        cognitoUser.confirmPassword(verificationCode, newPassword, this);
    }
});
```

7단계: 앱 사용자를 위해 AWS 리소스에 액세스하기 위한 자격 증명 가져오기

다른 AWS 제품을 작업하려면 먼저 Amazon Cognito [자격 증명 풀](#) (p. 225)을 생성해야 합니다. 이 자격 증명 풀을 생성한 후 사용자에게 로그인할 때 자격 증명 풀 ID와 ID 토큰(이전에 받음)을 전달하여 자격 증명을 가져올 수 있습니다. 다음 예는 IdentityPoolId를 채우고 Logins 맵을 통해 ID 토큰을 전달하는 방법을 보여줍니다.

```
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
    IdentityPoolId: 'us-east-1:XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX',
    Logins: {
        'cognito-idp.us-east-1.amazonaws.com/us-east-1_XXXXXXXX':
        result.getIdToken().getJwtToken()
    }
});

AWS.config.credentials.get(function(err){
    if (err) {
        alert(err);
    }
});
```

다음 단계

더 많은 예제 및 본 자습서에서 사용된 코드 개요는 [Amazon Cognito 자격 증명 JavaScript GitHub 리포지토리](#)를 참조하십시오.

예제: JavaScript SDK 사용

애플리케이션에 사용자 등록

CognitoUserPool 및 UserPoolId를 제공하고 사용자 이름, 암호, 속성 목록 및 확인 데이터를 통해 등록하여 ClientId 객체를 생성해야 합니다.

```
var poolData = { UserPoolId : 'us-east-1_TcoKgbf7n',
    ClientId : '4pe2usejqcdmhi0a25jp4b5sh3'
};
var userPool = new AmazonCognitoIdentity.CognitoUserPool(poolData);

var attributeList = [];

var dataEmail = {
    Name : 'email',
    Value : 'email@mydomain.com'
};
var dataPhoneNumber = {
    Name : 'phone_number',
    Value : '+15555555555'
};
var attributeEmail = new AmazonCognitoIdentity.CognitoUserAttribute(dataEmail);
var attributePhoneNumber = new
AmazonCognitoIdentity.CognitoUserAttribute(dataPhoneNumber);
```

```
attributeList.push(attributeEmail);
attributeList.push(attributePhoneNumber);

userPool.signUp('username', 'password', attributeList, null, function(err, result){
    if (err) {
        alert(err);
        return;
    }
    cognitoUser = result.user;
    console.log('user name is ' + cognitoUser.getUsername());
});
```

로컬 스토리지에서 현재 사용자 검색

```
var data = { UserPoolId : 'us-east-1_Iqc12345',
    ClientId : '12345du353sm7khj1q'
};
var userPool = new AmazonCognitoIdentity.CognitoUserPool(data);
var cognitoUser = userPool.getCurrentUser();

if (cognitoUser != null) {
    cognitoUser.getSession(function(err, session) {
        if (err) {
            alert(err);
            return;
        }
        console.log('session validity: ' + session.isValid());
    });
}
```

사용자 인증

다음은 Amazon Cognito 서비스를 통해 사용자를 인증하고 사용자 세션을 설정하는 예제입니다.

Note

콘솔에서 관리자가 사용자를 생성했지만 앞에 나온 JavaScript 예제를 통하지 않은 경우에는 [예제: Android용 Mobile SDK에서 AdminCreateUser API를 사용하여 생성된 사용자 처리 \(p. 55\)](#) 및 [관리자로서 사용자 계정 생성 \(p. 163\)](#) 단원을 참조하십시오.

```
var authenticationData = {
    Username : 'username',
    Password : 'password',
};
var authenticationDetails = new
AmazonCognitoIdentity.AuthenticationDetails(authenticationData);
var poolData = { UserPoolId : 'us-east-1_TcoKgbf7n',
    ClientId : '4pe2usejqcdmhi0a25jp4b5sh3'
};
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);  
    },  
});
```

사용자 풀에 대한 MFA 활성화

다음은 인증 사용자에게 대한 MFA 설정(선택 사항)이 있는 사용자 풀의 MFA(멀티 팩터 인증)를 활성화하는 예제입니다.

```
cognitoUser.enableMFA(function(err, result) {  
    if (err) {  
        alert(err);  
        return;  
    }  
    console.log('call result: ' + result);  
});
```

사용자 풀에 대한 MFA 비활성화

다음은 인증 사용자에게 대한 MFA 설정(선택 사항)이 있는 사용자 풀의 MFA(멀티 팩터 인증)를 비활성화하는 예제입니다.

```
cognitoUser.disableMFA(function(err, result) {  
    if (err) {  
        alert(err);  
        return;  
    }  
    console.log('call result: ' + result);  
});
```

사용자 풀 객체 생성

```
var data = { UserPoolId : 'us-east-1_q2Y6U8uuY',  
             ClientId : '224kjog470jnt9ov773erj7qn9'  
};  
  
var userPool = new AmazonCognitoIdentity.CognitoUserPool(data);
```

애플리케이션에 가입

```
var attribute = {  
    Name : 'email',  
    Value : 'email@mydomain.com'  
};  
  
var attributeEmail = new AmazonCognitoIdentity.CognitoUserAttribute(attribute);  
var attributeList = [];  
  
attributeList.push(attributeEmail);  
var cognitoUser;  
  
userPool.signUp('username', 'password', attributeList, null, function(err,  
result) {  
    if (err) {  
        alert(err);  
    }  
});
```

```
        return;
    }
    cognitoUser = result.user;
});
```

MFA를 활성화하여 로그인

```
var userData = {
    Username : 'username',
    Pool : userPool
};

cognitoUser = new AmazonCognitoIdentity.CognitoUser(userData);

var authenticationData = {
    Username : 'username',
    Password : 'password',
};

var authenticationDetails = new
AmazonCognitoIdentity.AuthenticationDetails(authenticationData);

cognitoUser.authenticateUser(authenticationDetails, {
    onSuccess: function (result) {
        alert('authentication successful!')
    },
    onFailure: function(err) {
        alert(err);
    },
    mfaRequired: function(codeDeliveryDetails) {
        var verificationCode = prompt('Please input verification code' , '');
        cognitoUser.sendMFACode(verificationCode, this);
    }
});
```

속성 업데이트

다음은 인증 사용자에게 대한 사용자 속성을 업데이트하는 예제입니다.

```
var attributeList = [];
var attribute = {
    Name : 'nickname',
    Value : 'joe'
};
var attribute = new AmazonCognitoIdentity.CognitoUserAttribute(attribute);
attributeList.push(attribute);

cognitoUser.updateAttributes(attributeList, function(err, result) {
    if (err) {
        alert(err);
        return;
    }
    console.log('call result: ' + result);
});
```

속성 삭제

다음은 인증 사용자에게 대한 사용자 속성을 삭제하는 예제입니다.

```
var attributeList = [];  
attributeList.push('nickname');  
  
cognitoUser.deleteAttributes(attributeList, function(err, result) {  
    if (err) {  
        alert(err);  
        return;  
    }  
    console.log('call result: ' + result);  
});
```

속성 확인

다음은 인증 사용자에게 대한 사용자 속성을 확인하는 예제입니다.

```
cognitoUser.getAttributeVerificationCode('email', {  
    onSuccess: function (result) {  
        console.log('call result: ' + result);  
    },  
    onFailure: function(err) {  
        alert(err);  
    },  
    inputVerificationCode: function() {  
        var verificationCode = prompt('Please input verification code: ' , '');  
        cognitoUser.verifyAttribute('email', verificationCode, this);  
    }  
});
```

속성 검색

다음은 인증 사용자에게 대한 사용자 속성을 검색하는 예제입니다.

```
cognitoUser.getUserAttributes(function(err, result) {  
    if (err) {  
        alert(err);  
        return;  
    }  
    for (i = 0; i < result.length; i++) {  
        console.log('attribute ' + result[i].getName() + ' has value ' +  
result[i].getValue());  
    }  
});
```

확인 코드 재전송

다음은 SMS를 통해 미인증 사용자에게 대한 등록을 확인하는 확인 코드를 재전송하는 예제입니다.

```
cognitoUser.resendConfirmationCode(function(err, result) {  
    if (err) {  
        alert(err);  
        return;  
    }  
    alert(result);  
});
```

등록 확인

```
cognitoUser.confirmRegistration('123456', true, function(err, result) {  
    if (err) {
```

```
        alert(err);  
        return;  
    }  
    alert(result);  
});
```

암호 변경

다음은 인증 사용자의 현재 암호를 변경하는 예제입니다.

```
cognitoUser.changePassword('oldPassword', 'newPassword', function(err, result) {  
    if (err) {  
        alert(err);  
        return;  
    }  
    console.log('call result: ' + result);  
});
```

잊어버린 암호 흐름

다음은 미인증 사용자에게 대한 잊어버린 암호 흐름을 시작하고 완료하는 예제입니다.

```
cognitoUser.forgotPassword({  
    onSuccess: function (result) {  
        console.log('call result: ' + result);  
    },  
    onFailure: function(err) {  
        alert(err);  
    },  
    inputVerificationCode() {  
        var verificationCode = prompt('Please input verification code ' , '');  
        var newPassword = prompt('Enter new password ' , '');  
        cognitoUser.confirmPassword(verificationCode, newPassword, this);  
    }  
});
```

사용자 삭제

다음은 인증 사용자를 삭제하는 예제입니다.

```
cognitoUser.deleteUser(function(err, result) {  
    if (err) {  
        alert(err);  
        return;  
    }  
    console.log('call result: ' + result);  
});
```

사용자 로그아웃

다음은 애플리케이션에서 현재 사용자를 로그아웃하는 예제입니다.

```
if (cognitoUser != null) {  
    cognitoUser.signOut();  
}
```

전역적으로 사용자 로그아웃

다음은 발급된 모든 토큰을 무효화하여 전역적으로 사용자를 로그아웃하는 예제입니다.

```
cognitoUser.globalSignOut();
```

현재 사용자 가져오기

다음은 로컬 스토리지에서 현재 사용자를 검색하는 예제입니다.

```
var data = {
  UserPoolId : '...', // Your user pool id here
  ClientId : '...' // Your client id here
};
var userPool = new AmazonCognitoIdentity.CognitoUserPool(data);
var cognitoUser = userPool.getCurrentUser();

if (cognitoUser != null) {
  cognitoUser.getSession(function(err, session) {
    if (err) {
      alert(err);
      return;
    }
    console.log('session validity: ' + session.isValid());

    AWS.config.credentials = new AWS.CognitoIdentityCredentials({
      IdentityPoolId : '...' // your identity pool id here
      Logins : {
        // Change the key below according to the specific region your user pool
        // is in.
        'cognito-idp.<region>.amazonaws.com/<YOUR_USER_POOL_ID>' :
        session.getIdToken().getJwtToken()
      }
    });

    // Instantiate aws sdk service objects now that the credentials have been
    // updated.
    // example: var s3 = new AWS.S3();

  });
}
```

사용자 풀의 사용자를 자격 증명 풀과 통합

다음은 사용자 풀의 현재 사용자를 지정된 자격 증명 풀과 통합하는 예제입니다.

```
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()
        }
      });
    }
  });
}

//call refresh method in order to authenticate user and get new temp credentials
```

```
AWS.config.credentials.refresh((error) => {
  if (error) {
    console.error(error);
  } else {
    console.log('Successfully logged!');
  }
});
```

사용자에 대한 모든 디바이스 나열

다음은 인증 사용자에 대한 모든 디바이스를 나열하는 예제입니다. 이 경우 한 번에 검색되는 디바이스 수에 대한 제한을 전달해야 합니다. 첫 번째 호출에서 페이지 매김 토큰은 null이어야 합니다. 첫 번째 호출은 모든 후속 호출에 전달되어야 하는 페이지 매김 토큰을 반환합니다.

```
cognitoUser.listDevices(limit, paginationToken, {
  onSuccess: function (result) {
    console.log('call result: ' + result);
  },
  onFailure: function(err) {
    alert(err);
  }
});
```

디바이스 정보 나열

다음은 현재 디바이스에 대한 정보를 나열하는 예제입니다.

```
cognitoUser.listDevices(limit, paginationToken, {
  onSuccess: function (result) {
    console.log('call result: ' + result);
  },
  onFailure: function(err) {
    alert(err);
  }
});
```

디바이스 기억

다음은 디바이스를 기억하는 예제입니다.

```
cognitoUser.setDeviceStatusRemembered({
  onSuccess: function (result) {
    console.log('call result: ' + result);
  },
  onFailure: function(err) {
    alert(err);
  }
});
```

디바이스를 기억하지 않음

다음은 디바이스를 기억하지 않음으로 표시하는 예제입니다.

```
cognitoUser.setDeviceStatusNotRemembered({
  onSuccess: function (result) {
    console.log('call result: ' + result);
  },
  onFailure: function(err) {
    alert(err);
  }
});
```



```
});
```

디바이스 잊기

다음은 현재 디바이스를 잊어버리는 예제입니다.

```
cognitoUser.forgetDevice({
  onSuccess: function (result) {
    console.log('call result: ' + result);
  },
  onFailure: function(err) {
    alert(err);
  }
});
```

등록된 미인증 사용자 확인

다음은 SMS 메시지를 통해 수신한 확인 코드를 사용하여 등록된 미인증 사용자를 확인하는 예제입니다.

```
var poolData = {
  UserPoolId : 'us-east-1_TcoKGbf7n',
  ClientId : '4pe2usejqcdmhi0a25jp4b5sh3'
};

var userPool = new AmazonCognitoIdentity.CognitoUserPool(poolData);
var userData = {
  Username : 'username',
  Pool : userPool
};

var cognitoUser = new AmazonCognitoIdentity.CognitoUser(userData);
cognitoUser.confirmRegistration('123456', true, function(err, result) {
  if (err) {
    alert(err);
    return;
  }
  console.log('call result: ' + result);
});
```

TOTP MFA를 사용하여 MFA 방법 선택 및 인증

다음 예에서는 TOTP를 사용하여 MFA 방법을 선택하고 인증합니다.

```
var authenticationData = {
  Username : 'username',
  Password : 'password',
};
var authenticationDetails = new
AmazonCognitoIdentity.AuthenticationDetails(authenticationData);
var poolData = {
  UserPoolId : '...', // Your user pool id here
  ClientId : '...' // Your client id here
};
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();
  },

  onFailure: function(err) {
    alert(err);
  },

  mfaSetup: function(challengeName, challengeParameters) {
    cognitoUser.associateSoftwareToken(this);
  },

  associateSecretCode : function(secretCode) {
    var challengeAnswer = prompt('Please input the TOTP code.' , '');
    cognitoUser.verifySoftwareToken(challengeAnswer, 'My TOTP device', this);
  },

  selectMFAType : function(challengeName, challengeParameters) {
    var mfaType = prompt('Please select the MFA method.', '');
    cognitoUser.sendMFASelectionAnswer(mfaType, this);
  },

  totpRequired : function(secretCode) {
    var challengeAnswer = prompt('Please input the TOTP code.' , '');
    cognitoUser.sendMFACode(challengeAnswer, this, 'SOFTWARE_TOKEN_MFA');
  }
});
```

SMS MFA를 활성화하고 사용자에게 대한 기본 MFA 방법으로 설정

다음 예에서는 SMS MFA를 활성화하고 사용자에게 대한 기본 MFA 방법으로 설정합니다.

```
smsMfaSettings = {
  PreferredMfa : true,
  Enabled : true
};
cognitoUser.setUserMfaPreference(smsMfaSettings, null, function(err, result) {
  if (err) {
    alert(err);
  }
  console.log('call result ' + result)
});
```

TOTP 소프트웨어 토큰 MFA를 활성화하고 사용자에게 대한 기본 MFA 방법으로 설정

다음 예에서는 TOTP 소프트웨어 토큰 MFA를 활성화하고 사용자에게 대한 기본 MFA 방법으로 설정합니다.

```
totpMfaSettings = {
  PreferredMfa : true,
  Enabled : true
};
cognitoUser.setUserMfaPreference(null, totpMfaSettings, function(err, result) {
  if (err) {
    alert(err);
  }
  console.log('call result ' + result)
});
```

예제: SDK for JavaScript에서 AdminCreateUser API를 통해 생성한 사용자의 새 암호를 인증 및 설정

관리자가 생성한 사용자에 대한 사용자 로그인 흐름을 지원하려면(AdminCreateUser API 사용) 사용자가 처음 로그인할 때 newPasswordRequired 콜백 메서드를 구현하여 새 암호를 설정합니다. 사용자는 초대에서 받은 임시 암호를 통해 처음 로그인하려고 시도하며 SDK가 newPasswordRequired 콜백을 호출합니다. 새 암호 및 필수 속성을 비롯하여 필요한 입력을 수집한 다음 completeNewPasswordChallenge 메서드를 호출합니다. 해당 메서드는 CognitoUser 클래스에서 사용할 수 있습니다.

newPasswordRequired 콜백은 userAttributes 및 requiredAttributes의 두 파라미터를 가져옵니다.

```
cognitoUser.authenticateUser(authenticationDetails, {
  onSuccess: function (result) {
    // User authentication was successful
  },

  onFailure: function(err) {
    // User authentication was not successful
  },

  mfaRequired: function(codeDeliveryDetails) {
    // MFA is required to complete user authentication.
    // Get the code from user and call
    cognitoUser.sendMFACode(mfaCode, this)
  },

  newPasswordRequired: function(userAttributes, requiredAttributes) {
    // User was signed up by an admin and must provide new
    // password and required attributes, if any, to complete
    // authentication.

    // userAttributes: object, which is the user's current profile. It will list
    // all attributes that are associated with the user.
    // Required attributes according to schema, which don't have any values yet,
    // will have blank values.
    // requiredAttributes: list of attributes that must be set by the user along
    // with new password to complete the sign-in.

    // Get these details and call
    // newPassword: password that user has given
    // attributesData: object with key as attribute name and value that the user
    // has given.
    cognitoUser.completeNewPasswordChallenge(newPassword, attributesData, this)
  }
});
```

예제: Lambda 트리거를 사용한 JavaScript 사용자 마이그레이션

사용자 마이그레이션 Lambda 트리거는 기존 사용자 관리 시스템에서 암호 재설정 없이 사용자 풀로의 손쉬운 마이그레이션을 허용합니다.

사용자 마이그레이션 Lambda 트리거 설정

JavaScript 앱을 변경하기 전에 사용자 풀에 대한 사용자 마이그레이션 Lambda를 설정합니다.

Lambda 트리거에 대해 자세히 알아보려면 [Lambda 트리거를 사용하여 사용자 풀 워크플로우 사용자 지정 \(p. 118\)](#) 단원을 참조하십시오.

Lambda 트리거를 사용한 사용자 마이그레이션에 대한 자세한 내용은 [사용자 마이그레이션 Lambda 트리거를 사용하여 사용자 풀로 사용자 가져오기 \(p. 171\)](#) 단원을 참조하십시오.

JavaScript 앱 변경

[AWSCognitoIdentityProvider JavaScript SDK](#)를 버전 2.0.2 이상으로 업데이트합니다.

사용자 마이그레이션 인증 흐름

기존 시스템에 대한 사용자를 인증하고 암호를 확인한 다음 프로필을 사용자 풀로 원활하게 마이그레이션할 수 있습니다. 하지만 암호 재설정을 회피하려면 서비스에 기존 암호가 필요합니다.

SDK의 기본 인증 흐름은 암호가 실제로 네트워크를 통해 전송되지 않는 보안 원격 암호(SRP) 프로토콜을 구현합니다. 앱에서 사용자 마이그레이션을 활성화하려면 `USER_PASSWORD_AUTH` 인증 흐름을 사용하여 인증 도중 암호화된 SSL 연결을 통해 서비스로 암호를 전송합니다. 사용자 마이그레이션 이후 기본 SRP 인증 흐름을 사용합니다.

인증 흐름 유형을 `USER_PASSWORD_AUTH`로 설정합니다.

```
cognitoUser.setAuthenticationFlowType('USER_PASSWORD_AUTH');

cognitoUser.authenticateUser(authenticationDetails, {
  onSuccess: function(result) {
    // User authentication was successful
  },
  onFailure: function(err) {
    // User authentication was not successful
  },
  mfaRequired: function (codeDeliveryDetails) {
    // MFA is required to complete user authentication.
    // Get the code from user and call
    cognitoUser.sendMFACode(verificationCode, this);
  }
});
```

Amazon Cognito 사용자 풀에 Android 앱 추가

Amazon Cognito 사용자 풀로 웹과 모바일 앱 사용자들이 가입하고 로그인하도록 할 수 있습니다. 인증 사용자에게 대해 암호를 변경할 수 있고 미인증 사용자에게 대해 잊어버린 암호 흐름을 시작할 수 있습니다. 다음 단원에서는 [Amazon Cognito 자격 증명 공급자](#)를 사용하여 사용자 풀 프로필을 검색 및 업데이트할 수 있도록 설정 정보와 예시를 제공합니다. 이러한 지침은 Android Studio에서 Android 애플리케이션 개발을 위해 작성되었습니다.

시작하려면 [Android용 AWS Mobile SDK](#) 및 [Android용 Mobile SDK 설명서](#)를 참조하십시오.

React Native 개발자인 경우 Amazon Cognito 자격 증명 공급자를 포함하는 React Native용 AWS Amplify 라이브러리를 사용하여 React Native 애플리케이션에 사용자 풀을 추가합니다. 자세한 내용은 [React Native용 AWS Amplify 라이브러리 설정](#)을 참조하십시오. 또한 [AWS Amplify Library 인증 가이드](#)를 참조하십시오.

Gradle 종속성

다음 종속성을 앱의 Gradle 파일에 추가합니다.

- AWS Android Core SDK(aws-android-sdk-core-x.x.x.jar): 최신 버전의 AWS Android Core를 aws-android-sdk-core-2.2.8.jar 형태로 프로젝트의 빌드 Gradle에 종속성 라이브러리로 추가합니다.
- AWS Cognito 자격 증명 공급자 Android SDK(aws-android-sdk-cognitoidentityprovider:2.3.8.jar): 앱의 빌드 Gradle에 최신 버전의 Cognito 자격 증명 공급자용 Android SDK를 추가합니다.

네트워크 권한

앱에서 네트워크 호출을 통해 AWS Cognito 자격 증명 공급자와 통신할 수 있도록 하려면 앱을 활성화하여 네트워크 작업을 수행해야 합니다.

다음 권한을 앱의 매니페스트 파일에 포함합니다.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

주제

- [Android용 Mobile SDK 사용 \(p. 38\)](#)
- [자습서: Android 앱을 위한 사용자 풀 통합 \(p. 39\)](#)
- [Android용 Mobile SDK를 통한 사용자 풀 생성 예제 \(p. 48\)](#)
- [예제: Android용 Mobile SDK에서 AdminCreateUser API를 사용하여 생성된 사용자 처리 \(p. 55\)](#)
- [예제: Lambda 트리거를 사용한 Android 사용자 마이그레이션 \(p. 56\)](#)

Android용 Mobile SDK 사용

이 단원에서는 앱에서 Android용 Mobile SDK를 통해 Amazon Cognito 사용자 풀을 사용하는 방법에 대해 알아보겠습니다. 이 SDK는 사용자 등록(가입), 확인, 및 인증 등을 수행하는 API를 제공합니다.

주요 개념

현재 스레드 또는 백그라운드 스레드에서 실행

네트워크 호출을 수행하여 Amazon Cognito 자격 증명 공급자 서비스와 상호 작용하는 모든 API에는 두 가지 메서드가 있습니다. 이러한 메서드 중 하나는 현재 스레드(예: `signUp()`)에서 작업과 네트워크 작업을 실행 하며, 다른 메서드(`InBackground`가 뒤에 표시됨, 예: `signUpInBackground()`)는 작업을 백그라운드 스레드에서 실행하지만 `InBackground` 메서드가 호출된 스레드에서 콜백 메서드를 호출합니다.

캐싱

Android용 Mobile SDK는 성공적으로 인증된 마지막 사용자 및 사용자의 토큰을 디바이스에 `SharedPreferences`로 로컬로 캐시합니다. 이 SDK는 성공적으로 인증된 마지막 사용자를 가져오는 메서드도 제공합니다.

앱 ID 및 앱 암호

앱 ID(클라이언트 ID라고도 함) 및 앱 암호(클라이언트 암호라고도 함)가 Amazon Cognito 사용자 풀 콘솔에서 생성됩니다. 앱 ID는 를 사용하는 데 필요합니다. 앱 암호는 선택 사항입니다. 그러나 앱 ID가 앱 암호와 연결된 경우 SDK에서 앱 ID와 앱 암호를 사용해야 합니다.

기본 클래스

CognitoUserPool

사용자 풀의 추상화를 나타냅니다. 새 사용자를 등록하고 이 풀에 속하는 사용자에 대한 새 인스턴스 `CognitoUser`를 생성하는 메서드를 제공합니다.

CognitoUser

사용자 풀의 단일 사용자를 나타냅니다. 이 클래스를 통해 인증(로그인), 사용자 속성과 설정 관리 등을 비롯하여 사용자에 대한 가능한 모든 작업을 수행할 수 있습니다. `CognitoUserPool` 객체에서 이 클래스의 인스턴스를 생성할 수 있습니다.

CognitoUserSession

Amazon Cognito에서 발급한 토큰(ID, 액세스 및 새로 고침 토큰)을 캡슐화하며 ID 및 액세스 토큰을 읽을 수 있는 메서드를 제공합니다.

CognitoUserDetails

CognitoUserAttributes 및 CognitoUserSettings를 캡슐화합니다.

CognitoUserAttributes

모든 사용자 속성을 캡슐화하고 속성을 읽고 쓸 수 있는 메서드를 제공합니다. 속성에 대한 자세한 내용은 [사용자 풀 속성 구성 \(p. 202\)](#)을 참조하십시오.

CognitoUserSettings

모든 사용자 설정을 캡슐화하고 속성을 읽고 쓸 수 있는 메서드를 제공합니다.

자습서: Android 앱을 위한 사용자 풀 통합

이 자습서에서는 Amazon Cognito 사용자 풀을 Android 애플리케이션과 통합하는 핵심 단계를 간략하게 설명합니다. 애플리케이션에서 사용자 풀을 사용하는 방법을 보여주는 전체 샘플 애플리케이션을 보려면 GitHub 웹 사이트의 [Amazon Cognito 사용자 풀 샘플](#)을 참조하십시오.

주제

- 1단계: 콘솔을 사용하여 앱을 위한 사용자 풀 생성 (p. 39)
- 2단계: 사용자 풀 인스턴스 생성 (p. 40)
- 3단계: 앱에 사용자 가입 (p. 40)
- 4단계: 앱에 대해 사용자 확인 (p. 41)
- 5단계: 별칭 값 충돌 해결 (p. 42)
- 6단계: 앱에 사용자 로그인 (p. 42)
- 7단계: 사용자 세부 정보 가져오기 (p. 43)
- 8단계: 앱 사용자를 위해 AWS 리소스에 액세스하기 위한 자격 증명 가져오기 (p. 43)
- 9단계: AWS 리소스에 액세스할 수 있도록 IAM 권한 설정 (p. 44)
- 10단계: 사용자를 위한 새로운 멀티 팩터 인증 메커니즘 설정 (p. 44)

1단계: 콘솔을 사용하여 앱을 위한 사용자 풀 생성

다음 절차에서는 사용자 풀을 생성하고 앱에서 사용자 풀을 사용하는 방법을 설명합니다. 이 절차는 기본 설정을 사용하여 풀 ID, 앱 클라이언트 ID 및 앱 클라이언트 암호를 생성합니다. 이 설정의 사용자 지정에 대한 자세한 내용은 [사용자 풀 참조\(AWS Management 콘솔\) \(p. 201\)](#) 단원을 참조하십시오.

앱을 위한 사용자 풀을 생성하려면

1. [Amazon Cognito 콘솔](#)에 로그인합니다.
2. [Manage your User Pools]를 선택합니다.
3. 사용자 풀 생성을 선택합니다.
4. 풀 이름에서 풀에 대한 이름을 입력하고 기본값 검토를 선택합니다. 그러면 기본 설정을 사용해 풀이 생성됩니다.
5. 왼쪽 탐색 창에서 속성을 선택하여 필요한 속성과 별칭으로 사용할 속성을 지정합니다. 다음 속성을 설정한 후 풀의 사용자가 이메일 주소를 확인하면 사용자 이름이나 이메일 주소를 통해 로그인할 수 있습니다.
 - a. email의 경우 필수 사항 및 Alias(별칭)를 선택하십시오.
 - b. phone number의 경우 필수 사항 및 Alias(별칭)를 선택하십시오.

- c. given name의 경우 필수 사항을 선택하십시오.
- d. [Save changes]를 선택합니다.
6. 왼쪽 탐색 창에서 정책을 선택하여 암호 정책을 지정합니다. 이 자습서에서는 기본 설정을 사용합니다.
7. 왼쪽 탐색 창에서 확인을 선택합니다. 이 페이지에서는 확인 코드를 전달하기 위해 풀의 사용자에게 전송되는 메시지를 사용자 지정할 수 있습니다. 이 자습서에서는 기본 설정을 사용합니다.
8. 왼쪽 탐색 창에서 앱 클라이언트를 선택한 다음 앱 클라이언트 추가를 선택합니다. 사용자 풀에 여러 앱 클라이언트를 생성할 수 있습니다.
9. App name(앱 이름)의 경우 앱 이름을 입력합니다. Generate app client secret(앱 클라이언트 보안키 생성)을 선택한 상태로 속성 읽기 및 쓰기 권한 설정을 선택합니다. 쓰기 권한이 필요한 속성을 선택합니다. 필수 속성에는 항상 쓰기 권한이 있습니다.
10. Create app(앱 생성)과 Save changes(변경 사항 저장)를 차례대로 선택합니다.
11. 왼쪽 탐색 모음에서 검토와 풀 생성을 차례대로 선택합니다.
12. 풀 ID, 풀 ARN, 앱 클라이언트 ID 및 앱 클라이언트 보안키를 기록해 둡니다. 왼쪽 탐색 모음의 앱 클라이언트 아래에서 앱 클라이언트 ID와 앱 클라이언트 보안키를 찾을 수 있습니다. 클라이언트 비밀번호를 보려면 세부 정보 표시를 선택합니다.

2단계: 사용자 풀 인스턴스 생성

사용자 풀 객체의 인스턴스를 만들려면 사용자 풀 ID, 클라이언트 ID, 클라이언트 암호 및 AWS 리전이 필요합니다. 다음 예에서는 CognitoUserPool 인스턴스를 생성하는 방법을 보여 줍니다. 이것은 애플리케이션에서 사용자 풀과 이루어지는 모든 상호 작용의 엔드포인트입니다. 샘플 애플리케이션에서는 userPool이 AppHelper.java에 생성됩니다.

리전 파라미터는 Android용 Mobile SDK enum [리전](#)의 유효한 AWS 리전입니다.

```
/* Create a CognitoUserPool instance */
CognitoUserPool userPool = new CognitoUserPool(context, userPoolId, clientId, clientSecret, cognitoRegion);
```

3단계: 앱에 사용자 가입

다음 단계에서는 사용자를 앱에 가입시키는 방법을 설명합니다.

사용자를 앱에 가입시키려면

1. 사용자에게서 다음 정보를 수집합니다.
 - 사용자 ID: 사용자가 로그인하는 데 사용되며 풀에서 고유해야 합니다.
 - 암호: 사용자의 암호입니다.
 - 사용자 속성: 풀의 필수 속성을 지정해야 합니다(이메일, 지정된 이름 및 전화 번호).
2. 풀 인스턴스를 사용하여 사용자를 가입시킵니다.

```
// Create a CognitoUserAttributes object and add user attributes
CognitoUserAttributes userAttributes = new CognitoUserAttributes();

// Add the user attributes. Attributes are added as key-value pairs
// Adding user's given name.
// Note that the key is "given_name" which is the OIDC claim for given name
userAttributes.addAttribute("given_name", userGivenName);

// Adding user's phone number
userAttributes.addAttribute("phone_number", phoneNumber);

// Adding user's email address
```

```
userAttributes.addAttribute("email", emailAddress);
```

3. 가입을 위한 콜백 핸들러를 만듭니다. 가입이 성공하면 onSuccess 메서드가 호출됩니다.

```
SignUpHandler signupCallback = new SignUpHandler() {  
  
    @Override  
    public void onSuccess(CognitoUser cognitoUser, boolean userConfirmed,  
        CognitoUserCodeDeliveryDetails cognitoUserCodeDeliveryDetails) {  
        // Sign-up was successful  
  
        // Check if this user (cognitoUser) needs to be confirmed  
        if(!userConfirmed) {  
            // This user must be confirmed and a confirmation code was sent to the user  
            // cognitoUserCodeDeliveryDetails will indicate where the confirmation code  
was sent  
            // Get the confirmation code from user  
        }  
        else {  
            // The user has already been confirmed  
        }  
    }  
  
    @Override  
    public void onFailure(Exception exception) {  
        // Sign-up failed, check exception for the cause  
    }  
};
```

4. 가입 API를 호출합니다.

```
userPool.signUpInBackground(userId, password, userAttributes, null, signupCallback);
```

4단계: 앱에 대해 사용자 확인

가입한 사용자가 로그인하려면 먼저 사용자를 확인해야 합니다. 이메일이나 전화를 통해 사용자를 확인할 수 있습니다. 가입이 성공한 후 사용자를 확인해야 할 경우 사용자 이메일 주소나 전화 번호로 확인 코드가 전송됩니다. Lambda 트리거를 사용하여 가입 후 자동으로 사용자를 확인할 수도 있습니다.

가입하는 동안 사용자가 이메일 주소나 전화 번호를 제공하고 사용자 풀에 대해 자동 확인이 선택되어 있으면 확인 코드가 사용자 전화 번호에 문자 메시지로 전송되거나 사용자 이메일 주소로 전송됩니다. 가입 성공 후 콜백 핸들러에 전달된 cognitoUserCodeDeliveryDetails 객체는 이 확인 코드가 전송된 대상을 나타냅니다. 이 객체를 사용하면 사용자가 확인 코드를 받을 방법을 알 수 있습니다.

다음 단계에서는 사용자가 앱에 로그인하기 전에 사용자 정보를 확인하는 방법을 설명합니다.

앱에 대해 사용자를 확인하려면

1. 사용자를 확인하기 위한 콜백 핸들러를 만듭니다. 이 콜백 핸들러는 확인 API 호출의 결과를 전달하기 위해 SDK에 사용됩니다.

```
// Callback handler for confirmSignUp API  
GenericHandler confirmationCallback = new GenericHandler() {  
  
    @Override  
    public void onSuccess() {  
        // User was successfully confirmed  
    }  
  
    @Override
```



```
public void onFailure(Exception exception) {
    // User confirmation failed. Check exception for the cause.
}
};
```

2. 새 사용자가 확인되면 확인 코드 전송에 사용된 사용자의 속성(이메일 주소 또는 전화 번호)이 확인됨으로 표시됩니다. 또한 이 속성을 별칭으로 사용하도록 설정한 경우 사용자 이름 대신 해당 속성(이메일 주소 또는 전화 번호)으로 사용자가 로그인할 수 있습니다.

5단계: 별칭 값 충돌 해결

별칭 값은 풀에서 고유해야 합니다. 새 사용자를 확인할 때 해당 사용자의 이메일 주소나 전화 번호가 별칭으로 사용되고 그 이메일이나 전화 번호가 이미 풀에 있는 기존 사용자에 사용되고 있으면 이 충돌을 해결해야 합니다. 고유성을 확인하기 위해 다음 중 하나를 수행할 수 있습니다.

- `forcedAliasCreation` 파라미터를 `false`로 설정합니다. 그러면 사용자 확인 실패를 허용하여 충돌이 해결됩니다. 속성은 기존 사용자에 대해 확인됨으로 유지되고 기존 사용자의 별칭으로 계속 사용됩니다. 새로운 사용자는 다음 예제에서와 같이 확인되지 않음으로 유지됩니다.

```
// This will cause confirmation to fail if the user attribute has been verified for
another user in the same pool
boolean forcedAliasCreation = false;

// Call API to confirm this user
cognitoUser.confirmSignUpInBackground(confirmationCode, forcedAliasCreation,
confirmationCallback);
```

- `forcedAliasCreation` 파라미터를 `true`로 설정하면 속성(이메일 또는 전화 번호)이 새 사용자에 대해 확인됨으로 표시되고 기존 사용자에 대해서는 확인되지 않음으로 표시되어 충돌이 해결됩니다. 이 속성은 더 이상 기존 사용자의 별칭이 아닙니다.

확인된 모든 사용자는 로그인할 수 있습니다. 로그인에 성공하면 액세스 및 ID 토큰이 반환됩니다. 이 토큰은 `CognitoUserSession` 객체에 있습니다.

6단계: 앱에 사용자 로그인

사용자를 앱에 로그인하도록 하려면 먼저 인증을 위한 콜백 핸들러를 만들어야 합니다. 다음 예제에서는 이 콜백 핸들러를 통해 SDK가 애플리케이션과 상호 작용하는 방식을 보여줍니다.

```
// Callback handler for the sign-in process
AuthenticationHandler authenticationHandler = new AuthenticationHandler() {

    @Override
    public void onSuccess(CognitoUserSession cognitoUserSession) {
        // Sign-in was successful, cognitoUserSession will contain tokens for the user
    }

    @Override
    public void getAuthenticationDetails(AuthenticationContinuation
authenticationContinuation, String userId) {
        // The API needs user sign-in credentials to continue
        AuthenticationDetails authenticationDetails = new AuthenticationDetails(userId,
password, null);

        // Pass the user sign-in credentials to the continuation
        authenticationContinuation.setAuthenticationDetails(authenticationDetails);
    }
};
```

```

        // Allow the sign-in to continue
        authenticationContinuation.continueTask();
    }

    @Override
    public void getMFACode(MultiFactorAuthenticationContinuation
multiFactorAuthenticationContinuation) {
        // Multi-factor authentication is required; get the verification code from user
        multiFactorAuthenticationContinuation.setMfaCode(mfaVerificationCode);
        // Allow the sign-in process to continue
        multiFactorAuthenticationContinuation.continueTask();
    }

    @Override
    public void onFailure(Exception exception) {
        // Sign-in failed, check exception for the cause
    }
};

// Sign in the user
cognitoUser.getSessionInBackground(authenticationHandler);

```

7단계: 사용자 세부 정보 가져오기

사용자를 인증한 후 다음 예제에서와 같이 사용자 풀에 있는 사용자에 대한 기타 정보를 가져올 수 있습니다.

```

// Implement callback handler for getting details
GetDetailsHandler getDetailsHandler = new GetDetailsHandler() {
    @Override
    public void onSuccess(CognitoUserDetails cognitoUserDetails) {
        // The user detail are in cognitoUserDetails
    }

    @Override
    public void onFailure(Exception exception) {
        // Fetch user details failed, check exception for the cause
    }
};

// Fetch the user details
cognitoUser.getDetailsInBackground(getDetailsHandler);

```

8단계: 앱 사용자를 위해 AWS 리소스에 액세스하기 위한 자격 증명 가져오기

사용자를 위해 AWS 리소스에 액세스할 자격 증명을 가져오려면 먼저 자격 증명 풀을 만들어 사용자 풀을 이 자격 증명 풀에 연결해야 합니다.

AWS 리소스에 액세스할 AWS 자격 증명을 가져오려면

1. [Amazon Cognito 콘솔](#)에 로그인합니다.
2. 연동 자격 증명 관리를 선택합니다.
3. [Create new identity pool]을 선택합니다. 자격 증명 풀 이름에 자격 증명 풀 이름을 입력합니다.
4. [Authentication providers] 섹션을 확장합니다. Cognito 탭에서 생성된 사용자 풀의 사용자 풀 ID 및 앱 클라이언트 ID를 입력합니다.
5. [Create Pool]을 선택합니다.
6. 애플리케이션 코드에서 다음과 같이 인증 성공 후 받은 ID 토큰을 자격 증명 공급자에 추가합니다.

```

// Get id token from CognitoUserSession.
String idToken = cognitoUserSession.getIdToken().getJWTToken();

```

```
// Create a credentials provider, or use the existing provider.
CognitoCachingCredentialsProvider credentialsProvider = new
CognitoCachingCredentialsProvider(context, IDENTITY_POOL_ID, REGION);

// Set up as a credentials provider.
Map<String, String> logins = new HashMap<String, String>();
logins.put("cognito-idp.us-east-1.amazonaws.com/us-east-1_123456678",
cognitoUserSession.getIdToken().getJWTToken());
credentialsProvider.setLogins(logins);
```

7. 다음과 같이 자격 증명 공급자를 사용하여 Amazon DynamoDB 테이블 등의 AWS 리소스에 액세스합니다.

```
AmazonDynamoDBClient ddbClient = new AmazonDynamoDBClient(credentialsProvider);
```

9단계: AWS 리소스에 액세스할 수 있도록 IAM 권한 설정

자격 증명 풀을 만들 때 Amazon Cognito에서 2개의 IAM 역할인 Cognito<identity pool name>Auth_Role 및 Cognito<identity pool name>Unauth_Role을 만듭니다. 기본적으로 이 역할은 Amazon Cognito 자격 증명 및 Amazon Cognito Sync에 대한 액세스만 허용합니다. 애플리케이션이 Amazon DynamoDB 등의 AWS 서비스에 액세스하도록 허용하려면 적절한 관리형 정책을 역할에 연결해야 합니다. 예를 들어, 애플리케이션이 DynamoDB 데이터베이스에 쓰고 읽으려면 다음 절차에서 설명하는 대로 AmazonDynamoDBFullAccess 관리형 정책을 역할에 연결해야 합니다.

AWS 리소스에 액세스할 수 있도록 IAM 권한을 설정하려면

1. AWS Management 콘솔에 로그인한 다음 <https://console.aws.amazon.com/iam/>에서 IAM 콘솔을 엽니다.
2. 정책에 연결할 인증된 역할을 역할 목록에서 선택한 후 정책 연결을 선택합니다.
3. 관리형 정책 목록에 대한 필수 정책(예: AmazonDynamoDBFullAccess)을 선택한 후 Attach Policy(정책 연결)를 선택합니다.

이제 애플리케이션이 DynamoDB에서 작업을 생성하고, 읽고, 업데이트하고, 삭제할 수 있습니다.

10단계: 사용자를 위한 새로운 멀티 팩터 인증 메커니즘 설정

Amazon Cognito SDK 버전 2.6.8로 시작하면 사용자를 위한 새로운 MFA 메커니즘을 설정할 수 있습니다. Amazon Cognito 사용자 풀 콘솔의 MFA and verifications(MFA 및 확인) 탭에서 사용자 풀 레벨 MFA 설정을 관리할 수 있습니다.

새로운 MFA 방법을 등록할 수 있으려면 먼저 사용자 풀에 대해 해당 방법을 활성화해야 합니다.

소프트웨어 토큰(TOTP) MFA 구성

Amazon Cognito에서는 시간 기반 일회용 암호 알고리즘(TOTP) 멀티 팩터 인증(MFA) 메커니즘을 지원합니다.

TOTP 구성 프로세스는 다음 두 단계로 구성됩니다.

1. 소프트웨어 토큰 연결: 사용자를 위한 TOTP MFA를 추가할 것을 Amazon Cognito 서비스에 요청함으로써 프로세스가 시작됩니다. 이것은 사용자에게 이미 설정된 TOTP MFA를 덮어씁니다. 이 요청에 대한 응답으로 해당 서비스는 보안 키를 생성하며, 이 키는 안전하게 저장되고 MFA 검증을 위한 TOTP 코드를 생성하는 데 사용되어야 합니다.
2. 소프트웨어 토큰 확인: 보안 키로부터 생성된 TOTP 코드를 사용하여 MFA 설정을 완료합니다. 이 단계를 완료하려면 Amazon Cognito SDK에 의해 반환된 `VerifyMfaContinuation` 객체를 사용합니다.

```
// Create a callback handler.
RegisterMfaHandler registerMFAHandler = new RegisterMfaHandler() {
    @Override
    public void onSuccess(final String sessionToken) {
        // Success, new MFA setup is complete.
    }

    @Override
    public void onVerify(VerifyMfaContinuation continuation) {
        // Get the secret key from Continuation.
        String secretKey = continuation.getParameters().get("secretKey");

        // Store the secret key in a TOTP code generator and verify using
        // the generated TOTP code.
        String verificationCode = storeAndGetTotpVerificationCode(secretKey);

        // Set a user friendly name to remember the TOTP generator.
        String friendlyName = "the best TOTP generator";

        // Complete the registration by verifying the TOTP code.
        continuation.setVerificationResponse(verificationCode, friendlyName);
        continuation.continueTask();
    }

    @Override
    public void onFailure(Exception exception) {
        closeWaitDialog();
        showDialogMessage("TOTP MFA registration failed",
            AppHelper.formatException(exception), false);
    }
};

// Use the CognitoUser to register a new Software Token MFA.
associateSoftwareTokenInBackground(sessionToken, registerMFAHandler);
```

Note

앱과의 TOTP 소프트웨어 토큰 MFA 연결은 사용자당 한 개만 허용됩니다. 새로운 MFA 소프트웨어 토큰을 연결하면 현재 연결을 덮어씁니다. TOTP MFA 소프트웨어 토큰을 연결하려면 해당 사용자가 유효한 액세스 토큰을 사용하여 인증을 받아야 합니다. 인증되지 않은 사용자가 새로운 TOTP MFA 소프트웨어 토큰을 연결해야 하는 경우, Amazon Cognito에서 생성된 sessionToken을 사용하여 새로운 TOTP MFA 소프트웨어 토큰을 연결할 수 있습니다.

세션 토큰으로 MFA 구성

이제 Amazon Cognito 사용자 풀에서는 여러 MFA 유형을 지원합니다. 콘솔의 사용자 풀에 허용되는 MFA 유형을 활성화할 수 있습니다. 또한 활성화된 MFA 유형을 선택 사항 또는 필수 사항으로 설정하도록 선택할 수 있습니다.

MFA를 선택 사항으로 설정하면 풀의 사용자가 해당 기본 설정에 따라 하나 이상의 MFA 방법을 등록하고 활성화하도록 선택할 수 있습니다.

사용자 풀에 대해 MFA를 필수 사항으로 설정하면 사용자는 적어도 한 개의 MFA 유형을 등록하고 활성화해야 합니다. MFA가 필수인 경우 적어도 한 개의 MFA 방법이 설정되지 않은 사용자에게 MFA_SETUP 챌린지가 표시됩니다. 사용자는 적어도 한 개의 MFA 유형을 설정해야 인증을 계속 진행할 수 있습니다.

이 단계에서 사용자가 새로운 MFA 방법을 등록할 수 있도록 허용하기 위해 Amazon Cognito에서는 해당 챌린지와 함께 세션 토큰을 반환합니다. SDK에 이 세션 토큰을 사용하여 MFA 유형을 등록한 다음 인증을 계속 진행할 수 있습니다. 세션 토큰은 새로운 MFA 유형을 등록하는 데만 사용할 수 있는 일회용 토큰입니다.

MFA_SETUP 챌린지

MFA_SETUP 챌린지가 표시되면 사용자는 MFA 방법을 설정해야 합니다. Amazon Cognito SDK는 RegisterMFAContinuation을 반환합니다. 이 연속 코드를 사용하여 사용자가 설정할 수 있는 MFA 유형을 가져온 다음 인증을 계속 진행할 수 있습니다.

```
AuthenticationHandler authHandler = new AuthenticationHandler() {

    @Override
    public void onSuccess(CognitoUserSession cognitoUserSession) {
        // ...
    }

    @Override
    public void getAuthenticationDetails(AuthenticationContinuation
authenticationContinuation, String userId) {
        // ...
    }

    @Override
    public void getMFACode(MultiFactorAuthenticationContinuation
multiFactorAuthenticationContinuation) {
        // ...
    }

    @Override
    public void authenticationChallenge(ChallengeContinuation continuation) {
        // This challenge is invoked for MFA_SETUP Challenge
        RegisterMFAContinuation regMFAContinuation =
        (RegisterMFAContinuation) continuation;

        // Register the new MFA.
        registerMfa(regMFAContinuation);
    }

    @Override
    public void onFailure(Exception exception) {
        // Sign-in failed, check exception for the cause
    }
};

// Register a new MFA.
public void registerMfa(RegisterMFAContinuation regMFAContinuation) {
    // Get the list of MFA's to setup.
    List<String> mfaOptions = continuation.getMfaOptions();

    // Get the session token to register an MFA.
    final String sessionToken = continuation.getSessionToken();

    // ...

    // Use the sessionToken to register MFA.
    associateSoftwareTokenInBackground(sessionToken, registerMFAHandler);
}

RegisterMfaHandler registerMFAHandler = new RegisterMfaHandler() {
    @Override
    public void onSuccess(final String sessionToken) {
        // Success, new MFA setup is complete.
        // Use the sessionToken to continue to authenticate.
        regMFAContinuation.setSessionToken(sessionToken);
    }

    @Override
    public void onVerify(VerifyMfaContinuation continuation) {
        // ...
    }
}
```

```
@Override
public void onFailure(Exception exception) {
    // ...
}
};
```

MFA 메커니즘 활성화 및 비활성화

API `setUserMfaSettings`를 사용하여 MFA 방법을 활성화하거나 비활성화하고 사용자에게 대해 MFA 유형을 기본으로 설정합니다.

```
GenericHandler updatesMFASettingsHandler = new GenericHandler() {
    @Override
    public void onSuccess() {
        // Update complete.
    }

    @Override
    public void onFailure(Exception exception) {
        // Failed update, check exception for details.
    }
};

// Enable SMS MFA and set preferred state
void enableSmsMfa(boolean preferred) {
    CognitoMfaSettings smsMfaSettings = new CognitoMfaSettings(CognitoMfaSettings.SMS_MFA);
    smsMfaSettings.setEnabled(true);
    smsMfaSettings.setPreferred(preferred);
    List<CognitoMfaSettings> settings = new ArrayList<CognitoMfaSettings>();
    settings.add(smsMfaSettings);
    cognitoUser.setUserMfaSettingsInBackground(settings, updateSettingHandler);
}
```

MFA 챌린지 선택

사용자에게 여러 MFA 유형이 활성화되어 있지만 기본으로 설정된 유형이 없는 경우 Amazon Cognito에서는 `SELECT_MFA_TYPE` 챌린지를 표시하며, 여기에서 사용자는 인증할 MFA 방법을 선택할 수 있습니다.

```
AuthenticationHandler authHandler = new AuthenticationHandler() {

    @Override
    public void onSuccess(CognitoUserSession cognitoUserSession) {
        // ...
    }

    @Override
    public void getAuthenticationDetails(AuthenticationContinuation
authenticationContinuation, String userId) {
        // ...
    }

    @Override
    public void getMFACode(MultiFactorAuthenticationContinuation
multiFactorAuthenticationContinuation) {
        // ...
    }

    @Override
```

```
public void authenticationChallenge(ChallengeContinuation continuation) {
    ChooseMfaContinuation mfaOptionsContinuation = (ChooseMfaContinuation)
continuation;
    // Get the list of MFA's to choose from
    List<String> mfaOptions = mfaOptionsContinuation.getMfaOptions();

    // ...

    // Set the MFA option and continue to authenticate.
    mfaOptionsContinuation.setMfaOption(option);
    mfaOptionsContinuation.continueTask();
}
@Override
public void onFailure(Exception exception) {
    // Sign-in failed, check exception for the cause
}
};
```

Android용 Mobile SDK를 통한 사용자 풀 생성 예제

이 주제에서는 Android용 Mobile SDK를 사용하여 기본 작업을 수행하는 코드 예제를 제공합니다. SDK에서 네트워크를 호출하므로 모든 API 호출은 비활동 스레드에서 수행되어야 합니다.

CognitoUserPool 생성

```
CognitoUserPool userPool = new CognitoUserPool(context, userPoolId, clientId,
clientSecret);

// user pool can also be created with client app configuration:
CognitoUserPool userPool = new CognitoUserPool(context, userPoolId, clientId, clientSecret,
clientConfiguration);
```

새 사용자 등록

```
// create a handler for registration
SignUpHandler handler = new SignUpHandler() {
    @Override
    public void onSuccess(CognitoUser user, CognitoUserCodeDeliveryDetails
codeDeliveryDetails) {
        // If the sign up was successful, "user" is a CognitoUser object of the user who
was signed up.
        // "codeDeliveryDetails" will contain details about where the confirmation codes
will be delivered.
    }

    @Override
    public void onFailure(Exception exception) {
        // Sign up failed, code check the exception for cause and perform remedial actions.
    }
}
```

캐시된 사용자 가져오기

```
CognitoUser user = userPool.getCurrentUser();
```

사용자 ID를 통해 사용자 객체 생성

```
CognitoUser user = userPool.getUser(userId);
```

사용자 확인

```
// create a callback handler for confirm
GenericHandler handler = new GenericHandler() {

    @Override
    public void onSuccess() {
        // User was successfully confirmed!
    }
    @Override
    public void onFailure(Exception exception) {
        // Confirmation failed, probe exception for details
    }
}

user.confirmSignUp(code, handler);
```

확인 코드 요청

```
// create a callback handler for the confirmation code request
GenericHandler handler = new GenericHandler() {

    @Override
    public void onSuccess() {
        // Confirmation code was successfully sent!
    }
    @Override
    public void onFailure(Exception exception) {
        // Confirmation code request failed, probe exception for details
    }
}

user.resendConfirmationCode(handler);
```

암호 찾기: 새 암호를 설정할 코드 가져오기

```
ForgotPasswordHandler handler = new ForgotPasswordHandler() {
    @Override
    public void onSuccess() {
        // Forgot password process completed successfully, new password has been
        // successfully set
    }

    @Override
    public void getResetCode(ForgotPasswordContinuation continuation) {
        // A code will be sent, use the "continuation" object to continue with the forgot
        // password process

        // This will indicate where the code was sent
        String codeSentHere = continuation.getParameters();

        // Code to get the code from the user - user dialogs etc.

        // If the program control has to exit this method, take the "continuation" object.
        // "continuation" is the only possible way to continue with the process

        // When the code is available

        // Set the new password
    }
}
```



```
        continuation.setPassword(newPassword);

        // Set the code to verify
        continuation.setVerificationCode(code);

        // Let the forgot password process proceed
        continuation.continueTask();

    }

    /**
     * This is called for all fatal errors encountered during the password reset process
     * Probe {@exception} for cause of this failure.
     * @param exception
     */
    public void onFailure(Exception exception) {
        // Forgot password processing failed, probe the exception for cause
    }
};

user.forgotPassword(handler);
```

인증 핸들러: 토큰 가져오기

```
// Implement authentication handler,
AuthenticationHandler handler = new AuthenticationHandler() {
    @Override
    public void onSuccess(CognitoUserSession userSession, CognitoDevice newDevice) {
        // Authentication was successful, the "userSession" will have the current valid
        tokens
        // Time to do awesome stuff
    }

    @Override
    public void getAuthenticationDetails(final AuthenticationContinuation continuation,
        final String userID) {
        // User authentication details, userID and password are required to continue.
        // Use the "continuation" object to pass the user authentication details

        // After the user authentication details are available, wrap them in an
        AuthenticationDetails class
        // Along with userID and password, parameters for user pools for Lambda can be
        passed here
        // The validation parameters "validationParameters" are passed in as a Map<String,
        String>
        AuthenticationDetails authDetails = new AuthenticationDetails(userID, password,
        validationParameters);

        // Now allow the authentication to continue
        continuation.setAuthenticationDetails(authDetails);
        continuation.continueTask();
    }

    @Override
    public void getMFACode(final MultiFactorAuthenticationContinuation continuation) {
        // Multi-factor authentication is required to authenticate
        // A code was sent to the user, use the code to continue with the authentication

        // Find where the code was sent to
        String codeSentHere = continuation.getParameter()[0];

        // When the verification code is available, continue to authenticate
        continuation.setMfaCode(code);
    }
};
```

```
        continuation.continueTask();
    }

    @Override
    public void authenticationChallenge(final ChallengeContinuation continuation) {
        // A custom challenge has to be solved to authenticate

        // Set the challenge responses

        // Call continueTask() method to respond to the challenge and continue with
        authentication.
    }

    @Override
    public void onFailure(final Exception exception) {
        // Authentication failed, probe exception for the cause
    }
};
user.getSession(handler);
```

사용자 세부 정보 가져오기

```
GetDetailsHandler handler = new GetDetailsHandler() {
    @Override
    public void onSuccess(final CognitoUserDetails list) {
        // Successfully retrieved user details
    }

    @Override
    public void onFailure(final Exception exception) {
        // Failed to retrieve the user details, probe exception for the cause
    }
};
user.getDetails(handler);
```

속성 확인 코드 가져오기

```
GenericHandler handler = new GenericHandler() {

    @Override
    public void onSuccess() {
        // Attribute verification code was successfully sent!
    }

    @Override
    public void onFailure(Exception exception) {
        // Attribute verification code request failed, probe exception for details
    }
};
user.getAttributeVerificationCode(attributeName, handler);
```

속성 확인

```
GenericHandler handler = new GenericHandler() {

    @Override
    public void onSuccess() {
        // Attribute verification was successful!
    }
};
```

```
    }

    @Override
    public void onFailure(Exception exception) {
        // Attribute verification failed, probe exception for details
    }
};
user.verifyAttribute(attributeName, code, handler);
```

속성 삭제

```
GenericHandler handler = new GenericHandler() {

    @Override
    public void onSuccess() {
        // Attribute deletion was successful!
    }

    @Override
    public void onFailure(Exception exception) {
        // Attribute deletion failed, probe exception for details
    }
};
user.deleteAttribute(attributeName, handler);
```

비밀번호 변경

```
GenericHandler handler = new GenericHandler() {

    @Override
    public void onSuccess() {
        // Password change was successful!
    }

    @Override
    public void onFailure(Exception exception) {
        // Password change failed, probe exception for details
    }
};
user.changePassword(oldPassword, newPassword, handler);
```

사용자 설정 변경 또는 지정

```
GenericHandler handler = new GenericHandler() {

    @Override
    public void onSuccess() {
        // Successfully changed settings!
    }

    @Override
    public void onFailure(Exception exception) {
        // Change failed, probe exception for details
    }
};

// userSettings is an object of the type CognitoUserSettings,
CognitoUserSettings userSettings = new CognitoUserSettings();

// Set the user settings
userSettings.setSettings(settingName, settingValue);
```

```
// Now update the new settings to the Amazon Cognito Identity Provider Service
user.setUserSettings(userSettings, handler);
```

사용자 삭제

```
GenericHandler handler = new GenericHandler() {

    @Override
    public void onSuccess() {
        // Delete was successful!
    }

    @Override
    public void onFailure(Exception exception) {
        // Delete failed, probe exception for details
    }
};
user.deleteUser(handler);
```

사용자 로그아웃

```
// This has cleared all tokens and this user will have to go through the authentication
process to get tokens.
user.signOut();
```

CognitoUserSession에서 액세스 및 ID 토큰 가져오기

```
// Session is an object of the type CognitoUserSession
String accessToken = session.getAccessToken().getJWT();
String idToken = session.getIdToken().getJWTToken();
```

사용자에 대한 모든 디바이스 나열

```
DevicesHandler devicesHandler = new DevicesHandler() {

    @Override
    public void onSuccess(List<CognitoDevice> devices) {
        // devices will contain a list of all remembered devices
    }

    @Override
    public void onFailure(Exception e) {
        // List devices failed, probe exception for details
    }
};
user.listDevicesInBackground(10, null, devicesHandler);
```

디바이스 기억

```
GenericHandler handler = new GenericHandler() {

    @Override
    public void onSuccess() {
        // Successful!
    }
};
```

```
@Override
public void onFailure(Exception exception) {
    // Failed, probe exception for details
}
};
cognitoDevice.rememberThisDeviceInBackground(handler)
```

디바이스를 기억하지 않음

```
GenericHandler handler = new GenericHandler() {

    @Override
    public void onSuccess() {
        // Successful!
    }

    @Override
    public void onFailure(Exception exception) {
        // Failed, probe exception for details
    }
};
cognitoDevice.doNotRememberThisDeviceInBackground(handler)
```

Amazon Pinpoint 분석

다음 절차에서는 안드로이드 Amazon Cognito 애플리케이션에 Amazon Pinpoint를 통합하는 방법을 설명합니다. 또한 Amazon Cognito 사용자 풀을 구성하여 이를 사용해야 합니다. Amazon Pinpoint 통합에 대한 자세한 정보는 [Amazon Cognito 사용자 풀을 통해 Amazon Pinpoint 분석 사용 \(p. 155\)](#) 단원을 참조하십시오. Amazon Pinpoint에 대한 자세한 내용은 [Amazon Pinpoint 설명서](#)를 참조하십시오.

CognitoUserPool 인스턴스 생성 시 Amazon Pinpoint 프로젝트 ID 포함

1. AWS Management 콘솔에 로그인한 다음 <https://console.aws.amazon.com/pinpoint/>에서 Amazon Pinpoint 콘솔을 엽니다.
2. Amazon Pinpoint 프로젝트가 없는 경우 하나를 생성하십시오. 프로젝트를 기록해 둡니다.

Note

Amazon Cognito 사용자 풀에 대해서도 동일한 Amazon Pinpoint 프로젝트 ID를 구성해야 합니다. 자세한 내용은 [사용자 풀 속성 분석 \(p. 215\)](#) 단원을 참조하십시오.

3. Android Amazon Cognito 앱에서 Amazon Cognito 사용자 풀 인스턴스를 인스턴스화하려면 `AWSCognitoIdentityUserPoolConfiguration`를 제공하고 이전 단계에서 Amazon Pinpoint 앱 ID를 사용하여 `pinpointAppId`를 설정합니다.

```
CognitoUserPool pool = new CognitoUserPool(context, userPoolId, clientId, clientSecret,
cognitoRegion, pinpointAppId);
```

Android 앱에서의 종속성

Android용 AWS Mobile SDK 자격 증명 제공자 버전 2.6.3 이상에서 Amazon Cognito를 사용하여 안드로이드 앱에서 분석을 활성화합니다.

Gradle 빌드를 사용 중인 경우 앱의 `build.gradle` 파일에 다음 라인을 추가합니다.

```
dependency {
    compile 'com.amazonaws:aws-android-sdk-cognitoidentityprovider:2.6.3'
}
```

이 설정을 완료하면 사용자 탭의 Amazon Pinpoint 프로젝트 콘솔에서 사용자 풀에 대한 분석을 사용할 수 있습니다.

예제: Android용 Mobile SDK에서 AdminCreateUser API를 사용하여 생성된 사용자 처리

Amazon Cognito 사용자 풀을 사용하면 관리자가 새 사용자를 생성하고 사용자를 로그인하도록 초대할 수 있습니다. 사용자는 처음 로그인할 때 암호를 설정해야 합니다. 또한 처음 로그인할 때 사용자는 아직 값이 없는 필수 속성에 대해 값을 제공해야 합니다.

Android용 Mobile SDK(버전 2.3.2 이상)는 이 기능을 지원합니다. 앱에서 이 기능을 지원하려면 AuthenticationChallenge 콜백 메서드를 구현해야 합니다. 이러한 사용자에게 대한 사용자 인증 프로세스가 변경되지 않았습니다. 그러나 초기 암호 확인 이후에는 SDK가 사용자가 새 암호를 읽도록 구현할 수 있는 AuthenticationChallenge 콜백을 호출합니다. 그런 다음 사용자는 필수 속성을 설정하고 관리자가 이미 설정한 사용자 속성을 변경할 수 있습니다.

AuthenticationChallenge 콜백 메서드에 전달된 연속 객체는 NewPasswordContinuation 유형입니다. NewPasswordContinuation 클래스는 ChallengeContinuation의 하위입니다. ChallengeContinuation 클래스는 챌린지 속성에 대한 손쉬운 액세스를 제공합니다.

사용자 인증 프로세스 중 AuthenticationChallenge 콜백이 호출되면 먼저 Challenge name을 확인하십시오. NEW_PASSWORD_REQUIRED라는 챌린지 이름은 관리자가 사용자의 계정을 생성한 후 사용자가 처음으로 로그인하려고 시도함을 나타냅니다. 챌린지 이름을 가져오려면 continuation.getChallengeName을 호출하십시오.

로그인 프로세스를 완료하려면 사용자는 사용자 풀이 생성되거나 업데이트될 경우 새 암호를 생성하고 필수로 표시된 사용자 속성에 누락된 값을 제공해야 합니다. 모든 필수 속성의 목록을 가져오려면 continuation.getRequiredAttributes를 호출하십시오. 관리자가 이미 설정한 속성과 값을 가져오려면 continuation.getCurrentUserAttributes를 호출하십시오.

사용자의 새 암호와 속성(필수 속성 포함)을 설정하려면 continuation.setPassword 및 continuation.setUserAttribute를 각각 호출하십시오.

continuation.continueTask를 호출하여 로그인 프로세스를 완료합니다.

```
@Override
public void authenticationChallenge(final ChallengeContinuation continuation) {
    // Check the challenge name
    if("NEW_PASSWORD_REQUIRED".equals(continuation.getChallengeName())) {
        // A new user is trying to sign in for the first time after
        // admin has created the user's account

        // Cast to NewPasswordContinuation for easier access to challenge parameters
        NewPasswordContinuation newPasswordContinuation = (NewPasswordContinuation) continuation;

        // Get the list of required parameters
        List<String> requiredAttributes = newPasswordContinuation.getRequiredAttributes()

        // Get the current user attributes
        Map<String, String> currUserAttributes =
        newPasswordContinuation.getCurrentUserAttributes();

        // Prompt user to set a new password and values for required attributes

        // Set new user password
        newPasswordContinuation.setPassword();

        // Set user attributes
        newPasswordContinuation.setUserAttribute(attributeName, attributeValue);
    }
}
```

```
// Set user attributes
newPasswordContinuation.setUserAttribute(anotherAttribute, valueOfAnotherAttribute);

// Allow the sign-in to complete
newPasswordContinuation.continueTask();
}

// Set the challenge responses

// Call continueTask() method to respond to the challenge and continue with
authentication.
}
```

예제: Lambda 트리거를 사용한 Android 사용자 마이그레이션

사용자 마이그레이션 Lambda 트리거는 기존 사용자 관리 시스템에서 암호 재설정 없이 사용자 풀로의 손쉬운 마이그레이션을 허용합니다.

사용자 마이그레이션 Lambda 트리거 설정

Android 앱을 변경하기 전에 사용자 풀에 대한 사용자 마이그레이션 Lambda를 설정합니다.

Lambda 트리거에 대해 자세히 알아보려면 [Lambda 트리거를 사용하여 사용자 풀 워크플로우 사용자 지정 \(p. 118\)](#) 단원을 참조하십시오.

Lambda 트리거를 사용한 사용자 마이그레이션에 대한 자세한 내용은 [사용자 마이그레이션 Lambda 트리거를 사용하여 사용자 풀로 사용자 가져오기 \(p. 171\)](#) 단원을 참조하십시오.

Android 앱 변경

AWSCognitoIdentityProvider Android SDK를 버전 2.6.15 이상으로 업데이트합니다.

사용자 마이그레이션 인증 흐름

기존 시스템에 대한 사용자를 인증하고 암호를 확인한 다음 프로필을 사용자 풀로 원활하게 마이그레이션할 수 있습니다. 하지만 암호 재설정을 회피하려면 서비스에 기존 암호가 필요합니다.

SDK의 기본 인증 흐름은 암호가 실제로 네트워크를 통해 전송되지 않는 보안 원격 암호(SRP) 프로토콜을 구현합니다. 앱에서 사용자 마이그레이션을 활성화하려면 USER_PASSWORD 인증 흐름을 사용하여 인증 도중 암호화된 SSL 연결을 통해 서비스로 암호를 전송합니다. 사용자 마이그레이션 이후 기본 SRP 인증 흐름을 사용합니다.

인증 유형을 USER_PASSWORD로 설정합니다.

```
authenticationDetails.setAuthenticationType("USER_PASSWORD");
```

getAuthenticationDetails() 콜백 핸들러에서 인증 유형이 설정됩니다.

```
AuthenticationHandler authenticationHandler = new AuthenticationHandler() {
    @Override
    public void onSuccess(CognitoUserSession cognitoUserSession, CognitoDevice device) {
        // Successful authentication.
    }

    @Override
    public void getAuthenticationDetails(AuthenticationContinuation authContinuation,
        String username) {
        // Get user password.
        String password = getUserPassword();
    }
}
```

```
// Add user credentials to Authentication Details.
AuthenticationDetails authenticationDetails = new AuthenticationDetails(username,
password, validationData);

// Set the authentication type to use USER_PASSWORD flow.
authenticationDetails.setAuthenticationType("USER_PASSWORD");

// Continue with authentication.
authContinuation.setAuthenticationDetails(authenticationDetails);
authContinuation.continueTask();
}

@Override
public void getMFACode(MultiFactorAuthenticationContinuation
multiFactorAuthenticationContinuation) {
    // ...
}

@Override
public void onFailure(Exception e) {
    // ...
}

@Override
public void authenticationChallenge(ChallengeContinuation continuation) {
    // ...
}
};
```

Amazon Cognito 사용자 풀에 iOS 앱 추가

Amazon Cognito 사용자 풀로 웹과 모바일 앱 사용자들이 가입하고 로그인하도록 할 수 있습니다. 인증 사용자에게 대해 암호를 변경할 수 있고 미인증 사용자에게 대해 잊어버린 암호 흐름을 시작할 수 있습니다. 다음 단원에서는 [Amazon Cognito 자격 증명 공급자](#)를 사용하여 사용자 풀 프로필을 검색 및 업데이트할 수 있도록 설정 정보와 예시를 제공합니다.

시작하려면 [AWS Mobile SDK for iOS](#) 및 [Mobile SDK for iOS 설명서](#)를 참조하십시오.

이러한 프레임워크를 사용하여 Mobile SDK for iOS를 기존 프로젝트로 가져올 수 있습니다.

- CocoaPods

CocoaPods 팟 `AWSCognitoIdentityProvider`를 PodSpec에 추가하고 `#import AWSCognitoIdentityProvider.h`를 사용할 클래스에 추가합니다.

[Mobile SDK for iOS CocoaPods 설정 지침](#)을 참조하십시오.

[CocoaPods](#)에 대해 자세히 알아보십시오.

- 프레임워크

프레임워크 `AWSCognitoIdentityProvider.framework`를 추가하고 `#import <AWSCognitoIdentityProvider/AWSCognitoIdentityProvider.h>`를 사용할 클래스에 추가합니다.

[Mobile SDK for iOS 프레임워크 설정 지침](#)을 참조하십시오.

[프레임워크](#)에 대해 자세히 알아보십시오.

- Carthage

[Mobile SDK for iOS Carthage 설정 지침](#)을 참조하십시오.

[Carthage](#)에 대해 자세히 알아보십시오.

- React Native

Amazon Cognito 자격 증명 공급자를 포함하는 React Native용 AWS Amplify 라이브러리를 사용하여 React Native 애플리케이션에 사용자 풀을 추가합니다.

자세한 내용은 [React Native용 AWS Amplify 라이브러리 설정](#)을 참조하십시오. 또한 [AWS Amplify Library 인증 가이드](#)를 참조하십시오.

[React Native](#)에 대해 자세히 알아보십시오.

이러한 환경 중 하나를 사용하여 AWS Mobile SDK를 가져오되 둘 이상은 가져올 수 없습니다. 다양한 방법으로 SDK를 가져오면 중복된 SDK 사본이 프로젝트에 로드되어 오류가 발생합니다.

주제

- [자습서: iOS 앱을 위한 사용자 풀 통합](#) (p. 58)
- [예제: iOS SDK를 통한 사용자 풀 사용](#) (p. 61)
- [예제: Lambda 트리거를 사용한 iOS 사용자 마이그레이션](#) (p. 70)

자습서: iOS 앱을 위한 사용자 풀 통합

이 자습서를 참고하여 사용자 풀을 시작할 수 있습니다.

주제

- [1단계: 콘솔을 사용하여 앱을 위한 사용자 풀 생성](#) (p. 58)
- [2단계: UserPool 객체 생성](#) (p. 59)
- [3단계: 앱에 사용자 가입](#) (p. 59)
- [4단계: 앱에 대해 사용자 확인](#) (p. 59)
- [5단계: 앱에 대해 사용자 인증](#) (p. 60)
- [6단계: 사용자 세부 정보 가져오기](#) (p. 60)
- [7단계: 앱 사용자를 위해 AWS 리소스에 액세스하기 위한 자격 증명 가져오기](#) (p. 61)
- [다음 단계](#) (p. 61)

1단계: 콘솔을 사용하여 앱을 위한 사용자 풀 생성

다음 절차에서는 사용자 풀을 생성하고 앱에서 사용자 풀을 사용하는 방법을 설명합니다. 이 절차는 기본 설정을 사용하여 풀 ID, 앱 클라이언트 ID 및 앱 클라이언트 암호를 생성합니다. 이 설정의 사용자 지정에 대한 자세한 내용은 [사용자 풀 참조\(AWS Management 콘솔\)](#) (p. 201) 단원을 참조하십시오.

앱을 위한 사용자 풀을 생성하려면

1. [Amazon Cognito 콘솔](#)에 로그인합니다.
2. [Manage your User Pools]를 선택합니다.
3. 사용자 풀 생성을 선택합니다.
4. 풀 이름에서 풀에 대한 이름을 입력하고 기본값 검토를 선택합니다.
5. 왼쪽 탐색 모음에서 앱 클라이언트와 Add an app(앱 추가)을 차례대로 선택합니다. 사용자 풀에 여러 앱 클라이언트를 생성할 수 있습니다.
6. App name(앱 이름)의 경우 앱 이름을 입력합니다. 클라이언트 보안키 생성을 선택한 상태로 Create app(앱 생성)을 선택한 후 변경 사항 저장을 선택합니다.

7. 왼쪽 탐색 모음에서 검토와 풀 생성을 차례대로 선택합니다.
8. 풀 ID를 기록합니다. 왼쪽 탐색 모음의 앱 클라이언트 아래에서 앱 클라이언트 ID와 앱 클라이언트 보안 키를 찾을 수 있습니다.

2단계: UserPool 객체 생성

1단계에서 받은 사용자 풀 ID, 앱 클라이언트 ID 및 앱 클라이언트 비밀번호를 사용하여 클라이언트 앱에 사용자 풀 개체를 만듭니다.

```
//setup service config
AWSServiceConfiguration *serviceConfiguration =
[[AWSServiceConfiguration alloc] initWithRegion:AWSRegionUSEast1 credentialsProvider:nil];

//create a pool
AWSCognitoIdentityUserPoolConfiguration *configuration =
[[AWSCognitoIdentityUserPoolConfiguration alloc] initWithClientId:@"CLIENT_ID"
                                                         clientSecret:@"CLIENT_SECRET"
                                                         poolId:@"USER_POOL_ID"];

[AWSCognitoIdentityUserPool
 registerCognitoIdentityUserPoolWithConfiguration:serviceConfiguration
               userPoolConfiguration:configuration forKey:@"UserPool"];
AWSCognitoIdentityUserPool *pool =
[AWSCognitoIdentityUserPool CognitoIdentityUserPoolForKey:@"UserPool"];
```

3단계: 앱에 사용자 가입

사용자를 가입시키려면 앱의 등록 UI가 사용자로부터 정보를 수집하고 signUp을 호출해야 합니다.

```
NSMutableArray * attributes = [NSMutableArray new];

//Set user attributes by retrieving them from your UI. These values are hardcoded for this
example
AWSCognitoIdentityUserAttributeType * phone = [AWSCognitoIdentityUserAttributeType new];

phone.name = @"phone_number";
//All phone numbers require +country code as a prefix
phone.value = @"+15555555555";

AWSCognitoIdentityUserAttributeType * email = [AWSCognitoIdentityUserAttributeType new];
email.name = @"email";
email.value = @"email@mydomain.com";

[attributes addObject:phone];
[attributes addObject:email];

//set username and password by retrieving them from your UI. They are hardcoded in this
example.
AWSCognitoIdentityUser *user = [[pool signUp:@"username" password:@"password"
userAttributes:attributes validationData:nil] continueWithSuccessBlock:^(id
_Nullable(AWSTask<AWSCognitoIdentityUser *> * _Nonnull task) {
    NSLog(@"Successfully registered user: %@",task.result.username);
    return nil;
}]];
```

4단계: 앱에 대해 사용자 확인

이메일 주소 또는 전화 번호가 확인되면 사용자가 확인됩니다. 다음 예제에서 사용자는 등록 흐름 중 이메일 주소 또는 모바일 전화의 SMS를 통해 확인 코드를 받으며, 가입을 완료하기 위해 이 코드를 입력해야 합니다. 최종 사용자로부터 확인 코드를 받은 후 confirmSignUp을 호출합니다.

```
//replace VERIFICATION_CODE with the value the user inputs
[[user confirmSignUp:@"VERIFICATION_CODE"] continueWithSuccessBlock:^(id
_Nullable(AWSTask<AWSCognitoIdentityProviderConfirmSignUpResponse *> * _Nonnull task) {
    NSLog(@"Successfully confirmed user: %@",user.username);
    return nil;
}]];
```

5단계: 앱에 대해 사용자 인증

확인된 사용자를 인증하려면 다음에 표시된 대로

AWSCognitoIdentityInteractiveAuthenticationDelegate 프로토콜을 구현하고 폴의 위임을 설정합니다. 이 프로토콜은 사용자 지정 로그인 UI를 관리하고 최종 사용자로부터 사용자 이름과 암호 정보를 수락합니다. 사용자가 인증하지 않았거나 로그아웃했거나 사용자의 새로 고침 토큰(30일간 유효)이 만료된 경우에만 프로토콜의 메서드가 호출됩니다.

```
//This code goes in your AppDelegate
pool.delegate = self;

-(id<AWSCognitoIdentityPasswordAuthentication>) startPasswordAuthentication{
    //implement code to instantiate and display login UI here
    //return something that implements the AWSCognitoIdentityPasswordAuthentication
    protocol
    return loginUI;
}

//This code goes in your Login UI
-(void) getPasswordAuthenticationDetails:
(AWSCognitoIdentityPasswordAuthenticationInput *) authenticationInput
passwordAuthenticationCompletionSource: (AWSTaskCompletionSource *)
passwordAuthenticationCompletionSource {
    //using inputs from login UI create an
    AWSCognitoIdentityPasswordAuthenticationDetails object.
    //These values are hardcoded for this example.
    AWSCognitoIdentityPasswordAuthenticationDetails * result =
    [[AWSCognitoIdentityPasswordAuthenticationDetails alloc] initWithUsername:@"USERNAME"
    password:@"PASSWORD"];
    //set the result to continue the sign-in process
    passwordAuthenticationDetails.result = result;
};

-(void) didCompletePasswordAuthenticationStepWithError:(NSError*) error {
    dispatch_async(dispatch_get_main_queue(), ^{
        //present error to end user
        if(error){
            [[UIAlertView alloc] initWithTitle:error.userInfo[@"__type"]
            message:error.userInfo[@"message"]
            delegate:nil
            cancelButtonTitle:nil
            otherButtonTitles:@"Ok", nil] show];
        }else{
            //dismiss view controller
            [self dismissViewControllerAnimated:YES completion:nil];
        }
    });
}
```

6단계: 사용자 세부 정보 가져오기

사용자 세부 정보를 가져오려면 다음에 표시된 대로 getDetails를 호출합니다.

```
[[user getDetails] continueWithSuccessBlock:^(id
_Nullable(AWSTask<AWSCognitoIdentityUserGetDetailsResponse *> * _Nonnull task) {
```

```
AWSCognitoIdentityUserGetDetailsResponse *response = task.result;
for (AWSCognitoIdentityUserAttributeType *attribute in response.userAttributes) {
    //print the user attributes
    NSLog(@"Attribute: %@ Value: %@", attribute.name, attribute.value);
}
return nil;
}];
```

7단계: 앱 사용자를 위해 AWS 리소스에 액세스하기 위한 자격 증명 가져오기

사용자를 위해 AWS 리소스에 액세스하기 위한 자격 증명을 가져오려면 먼저 사용자 풀을 자격 증명 풀에 연결한 후 `AWSCognitoIdentityUserPool`을 `AWSCognitoCredentialsProvider`에 제공합니다. 다음 절차에서는 자격 증명 풀을 가져오는 방법을 설명합니다.

자격 증명 풀을 생성하려면

1. [Amazon Cognito 콘솔](#)에 로그인합니다.
2. 연동 자격 증명 관리를 선택합니다.
3. [Create new identity pool]을 선택합니다. 자격 증명 풀 이름에 자격 증명 풀 이름을 입력합니다.
4. [Authentication providers] 섹션을 확장합니다.
5. Cognito 탭에서 사용자 풀 ID 및 앱 클라이언트 ID를 지정합니다.
6. 자격 증명 풀 연결을 구성한 후 `AWSCognitoIdentityUserPool`을 `AWSCognitoCredentialsProvider`에 제공하여 AWS 자격 증명을 앱으로 가져옵니다.

```
AWSCognitoCredentialsProvider *credentialsProvider = [[AWSCognitoCredentialsProvider
    alloc] initWithRegionType:AWSRegionUSEast1 identityPoolId:@"IDENTITY_POOL_ID"
    identityProviderManager:pool];
AWSServiceConfiguration *defaultServiceConfiguration = [[AWSServiceConfiguration alloc]
    initWithRegion:AWSRegionUSEast1
    credentialsProvider:credentialsProvider];
AWSServiceManager.defaultServiceManager.defaultServiceConfiguration =
    defaultServiceConfiguration;
```

다음 단계

이 자습서에 설명된 기능을 보여주는 실제 예제는 [Github의 Objective-C 샘플](#) 또는 [Github의 Swift 샘플](#)을 참조하십시오.

예제: iOS SDK를 통한 사용자 풀 사용

이 주제에서는 AWS Mobile SDK for iOS와 함께 사용자 풀을 사용할 때 사용자 등록, 확인 및 인증과 사용자 속성 가져오기에 대한 세부 정보를 제공합니다.

AWSCognitoIdentityUserPool 객체 생성

다음 절차는 상호 작용할 `AWSCognitoIdentityUserPool` 객체를 생성하는 방법을 설명합니다.

1. 서비스 구성 설정

Note

`credentialsProvider`가 `nil`로 설정됩니다. 이 서비스와 상호 작용하는 데 자격 증명 공급자 또는 AWS 자격 증명은 필요 없습니다.

```
//setup service config
```

```
AWSServiceConfiguration *serviceConfiguration = [[AWSServiceConfiguration alloc]
    initWithRegion:AWSRegionUSEast1 credentialsProvider:nil];
```

2. 사용자 풀 구성 생성

```
//create a pool
AWSCognitoIdentityUserPoolConfiguration *configuration =
    [[AWSCognitoIdentityUserPoolConfiguration alloc] initWithClientId:@"CLIENT_ID"

        clientSecret:@"CLIENT_SECRET"

        poolId:@"USER_POOL_ID"];
[AWSCognitoIdentityUserPool
    registerCognitoIdentityUserPoolWithConfiguration:serviceConfiguration
    userPoolConfiguration:configuration forKey:@"UserPool"];
AWSCognitoIdentityUserPool *pool = [AWSCognitoIdentityUserPool
    CognitoIdentityUserPoolForKey:@"UserPool"];
```

예제: 사용자 등록

pool.signUp:password:userAttributes:validationData를 사용하여 사용자를 등록합니다.

```
AWSCognitoIdentityUserAttributeType * phone = [AWSCognitoIdentityUserAttributeType new];
phone.name = @"phone_number";
//phone number must be prefixed by country code
phone.value = @"+15555555555";
AWSCognitoIdentityUserAttributeType * email = [AWSCognitoIdentityUserAttributeType new];
email.name = @"email";
email.value = @"email@mydomain.com";

//sign up the user
[[pool signUp:@"username" password:@"password" userAttributes:[email,phone]
    validationData:nil] continueWithBlock:^(id
    _Nullable(AWSTask<AWSCognitoIdentityUserPoolSignUpResponse *> * _Nonnull task) {
    dispatch_async(dispatch_get_main_queue(), ^{
        if(task.error){
            [[[UIAlertView alloc] initWithTitle:task.error.userInfo[@"__type"]
                message:task.error.userInfo[@"message"]
                delegate:self
                cancelButtonTitle:@"Ok"
                otherButtonTitles:nil] show];
        }else {
            AWSCognitoIdentityUserPoolSignUpResponse * response = task.result;
            if(!response.userConfirmed){
                //need to confirm user using user.confirmUser:
            }
        }
    });
    return nil;
}];
```

예제: 사용자 가져오기

풀에서 다음 메서드 중 하나를 사용하거나 등록하여 사용자를 가져올 수 있습니다.

```
//get the last logged in user
[pool currentUser];

//get a user without a username
[pool getUser];

//get a user with a specific username
```

```
[pool getUser:@"username"];
```

예제: 사용자 로그인

두 가지 로그인 방법이 있습니다. 명시적인 방법 또는 위임을 통해 사용자 자격 증명이 필요한 경우입니다.

명시적으로 로그인하려면 다음을 사용하십시오.

```
[[user getSession:@"username" password:@"password" validationData:nil scopes:nil]
continueWithSuccessBlock:^(id _Nullable(AWSCognitoIdentityUserSession *) * _Nonnull
task) {
    //success, task.result has user session
    return nil;
}];
```

위임을 구현하려면 풀에서 `AWSCognitoIdentityInteractiveAuthenticationDelegate`를 구현하고 위임을 설정합니다.

```
pool.delegate = self;
```

구현 시 인증 사용자 인터페이스가 생성되지 않아 표시되지 않는 경우 해당 인터페이스를 인스턴스화하는 코드를 작성합니다.

```
//set up password authentication ui to retrieve username and password from the user
-(id) startPasswordAuthentication {
    //write code to instantiate your sign in ui if it wasn't created here
    dispatch_async(dispatch_get_main_queue(), ^{
        //write code to display your ui
    });

    //return your sign in ui which implements the
    AWSCognitoIdentityPasswordAuthentication protocol
    return signInViewController;
}

//set up mfa ui to retrieve mfa code from end user
//this is optional and only necessary if you turn on multifactor authentication on your
pool
-(id) startMultiFactorAuthentication {
    //write code to instantiate your multifactor authentication ui if it wasn't created
    here
    dispatch_async(dispatch_get_main_queue(), ^{
        //write code to display your ui
    });

    //return your sign in ui which implements the
    AWSCognitoIdentityMultiFactorAuthentication protocol
    return mfaViewController;
}

//set up new password required ui to retrieve new password and any required user profile
from end user
//this is optional and only necessary if you use the AdminCreateUser feature on the pool
-(id) startNewPasswordRequired {
    //write code to instantiate your new password required ui if it wasn't created here
    dispatch_async(dispatch_get_main_queue(), ^{
        //write code to display your ui
    });

    //return your new password required ui which implements the
    AWSCognitoIdentityNewPasswordRequired protocol
```

```

        return newPasswordRequiredController;
    }

    //set up ui to prompt end user to setup a software MFA
    //this is optional and only necessary if you have software MFA enabled and MFA is required
    on your pool
    -(id<AWSCognitoIdentitySoftwareMfaSetupRequired>) startSoftwareMfaSetupRequired {
        //write code to instantiate your software token setup required ui if it wasn't
        created here
        dispatch_async(dispatch_get_main_queue(), ^{
            //write code to display your ui
        });

        //return your software mfa setup required ui which implements the
        AWSCognitoIdentitySoftwareMfaSetupRequired protocol
        return softwareMfaSetupController;
    }

    //set up ui to prompt end user to select which MFA they want for this authentication
    //this is optional and only necessary if you have users can have multiple MFAs setup on the
    pool
    -(id<AWSCognitoIdentitySelectMfa>) startSelectMfa {
        //write code to instantiate your select MFA ui if it wasn't created here
        dispatch_async(dispatch_get_main_queue(), ^{
            //write code to display your ui
        });

        //return your select MFA ui which implements the AWSCognitoIdentitySelectMfa protocol
        return selectMfaController;
    }

    //set up ui to drive a custom authentication flow
    //this is optional and only necessary if you have custom authentication lambdas configured
    on your pool
    -(id<AWSCognitoIdentityCustomAuthentication>) startCustomAuthentication {
        //write code to instantiate your custom authentication ui if it wasn't created here
        dispatch_async(dispatch_get_main_queue(), ^{
            //write code to display your ui
        });

        //return your custom authentication ui which implements the
        AWSCognitoIdentityCustomAuthentication protocol
        return customAuthenticationController;
    }
}

```

암호 인증 UI에서 AWSCognitoIdentityPasswordAuthentication 프로토콜을 구현합니다.

```

-(void) getPasswordAuthenticationDetails: (AWSCognitoIdentityPasswordAuthenticationInput
*) authenticationInput passwordAuthenticationCompletionSource:
(AWSTaskCompletionSource<AWSCognitoIdentityPasswordAuthenticationDetails *> *)
passwordAuthenticationCompletionSource {
    //keep a handle to the completion, you'll need it continue once you get the inputs from
    the end user
    self.passwordAuthenticationCompletion = passwordAuthenticationCompletionSource;
    //authenticationInput has details about the last known username if you need to use it
}

-(void) didCompletePasswordAuthenticationStepWithError:(NSError*) error {
    dispatch_async(dispatch_get_main_queue(), ^{
        //on completion, either display the error or dismiss the ui
        if(error){
            [[[UIAlertView alloc] initWithTitle:error.userInfo[@"__type"]
            message:error.userInfo[@"message"]
            delegate:nil

```

```

                                cancelButtonTitle:nil
                                otherButtonTitles:@"Retry", nil] show];
        }else{
            [self dismissViewControllerAnimated:YES completion:nil];
        }
    });
}

```

최종 사용자가 사용자 이름 및 암호를 입력한 경우 `passwordAuthenticationCompletion`에 대한 결과를 설정합니다.

```

self.passwordAuthenticationCompletion.result =
[[AWSCognitoIdentityPasswordAuthenticationDetails alloc] initWithUsername:@"username"
password:@"password"];

```

MFA(멀티 팩터 인증)가 지원되는 경우 `AWSCognitoIdentityMultiFactorAuthentication` 프로토콜을 구현할 수 있습니다.

```

-(void) getMultiFactorAuthenticationCode:
(AWSCognitoIdentityMultifactorAuthenticationInput )authenticationInput
mfaCodeCompletionSource: (AWSTaskCompletionSource<NSString > *) mfaCodeCompletionSource {
    //keep a handle to the completion, you'll need it continue once you get the inputs from
    the end user
    self.mfaCodeCompletion = mfaCodeCompletionSource;
    //authenticationInput has details about where the mfa code was sent if you need to
    display them in your ui
}
-(void) didCompleteMultifactorAuthenticationStepWithError:(NSError*) error {
dispatch_async(dispatch_get_main_queue(), ^{
    //on completion, either display the error or dismiss the ui
    if(error){
        [[UIAlertView alloc] initWithTitle:error.userInfo[@"__type"]
                                     message:error.userInfo[@"message"]
                                     delegate:nil
                                     cancelButtonTitle:nil
                                     otherButtonTitles:@"Retry", nil] show];
    }else{
        [self dismissViewControllerAnimated:YES completion:nil];
    }
});
}

```

최종 사용자가 코드를 입력한 경우 `mfaCodeCompletion`에 대한 결과를 설정합니다.

```

self.mfaCodeCompletion.result = @"mfaCodeFromUser";

```

`AdminCreateUser`를 통해 가입을 지원하는 경우 `AWSCognitoIdentityNewPasswordRequired` 프로토콜을 구현할 수 있습니다.

```

-(void) getNewPasswordDetails: (AWSCognitoIdentityNewPasswordRequiredInput
*) newPasswordRequiredInput
newPasswordRequiredCompletionSource:
(AWSTaskCompletionSource<AWSCognitoIdentityNewPasswordRequiredDetails *> *)
newPasswordRequiredCompletionSource {
    //keep a handle to the completion, you'll need it continue once you get the inputs
    from the end user
    self.newPasswordRequiredCompletionSource = newPasswordRequiredCompletionSource;
    //newPasswordRequiredInput has details about the existing user attributes and
    required fields if you need to display them in your ui
}

-(void) didCompleteNewPasswordStepWithError:(NSError* _Nullable) error {

```



```
dispatch_async(dispatch_get_main_queue(), ^{
//on completion, either display the error or dismiss the ui
if(error){
    [[[UIAlertView alloc] initWithTitle:error.userInfo[@"__type"]
                                message:error.userInfo[@"message"]
                                delegate:nil
                                cancelButtonTitle:nil
                                otherButtonTitles:@"Retry", nil] show];
}
}else{
    [self dismissViewControllerAnimated:YES completion:nil];
}});
```

최종 사용자가 제안된 암호 및 필수 속성을 입력한 경우 newPasswordRequiredCompletionSource에 대한 결과를 설정합니다.

```
NSDictionary<NSString *, NSString *> *userAttributes = @{@"name":@"My new name",
    @"email":@"mynewemail@myemail.com"};
AWSCognitoIdentityNewPasswordRequiredDetails *details =
    [[[AWSCognitoIdentityNewPasswordRequiredDetails alloc]
    initWithProposedPassword:@"newPassword" userAttributes:userAttributes];
self.newPasswordRequiredCompletionSource.result = details;
```

소프트웨어 MFA를 지원하는 경우, AWSCognitoIdentitySoftwareMfaSetupRequired 프로토콜을 구현할 수 있습니다.

```
-(void) getSoftwareMfaSetupDetails: (AWSCognitoIdentitySoftwareMfaSetupRequiredInput
*) softwareMfaSetupInput softwareMfaSetupRequiredDetails:
(AWSTaskCompletionSource<AWSCognitoIdentitySoftwareMfaSetupRequiredDetails *> *)
softwareMfaSetupRequiredCompletionSource{
    //keep a handle to the completion, you'll need it continue once you get the inputs
    from the end user
    self.softwareMfaSetupRequiredCompletionSource =
    softwareMfaSetupRequiredCompletionSource;
    // softwareMfaSetupInput has details about the secret code the end user needs to
    register with their TOTP application.
}

-(void) didCompleteMfaSetupStepWithError:(NSError* _Nullable) error {
    dispatch_async(dispatch_get_main_queue(), ^{
        //on completion, either display the error or dismiss the ui
        if(error){
            [[[UIAlertView alloc] initWithTitle:error.userInfo[@"__type"]
                                message:error.userInfo[@"message"]
                                delegate:nil
                                cancelButtonTitle:nil
                                otherButtonTitles:@"Retry", nil] show];
        }
        }else{
            [self dismissViewControllerAnimated:YES completion:nil];
        }
    });
}
```

최종 사용자에게 소프트웨어 토큰 앱으로부터의 현재 코드가 있는 경우, softwareMfaSetupRequiredCompletionSource에 대한 결과를 설정합니다.

```
AWSCognitoIdentitySoftwareMfaSetupRequiredDetails *details =
    [[[AWSCognitoIdentitySoftwareMfaSetupRequiredDetails alloc] initWithUserCode:@"User Code"
    friendlyDeviceName:@"Friendly Name"
    ]];
```

```
self.newPasswordRequiredCompletionSource.result = details;
```

여러 MFA 유형을 지원하는 경우, AWSCognitoIdentitySelectMfa 프로토콜을 구현할 수 있습니다.

```
-(void) getSelectMfaDetails: (AWSCognitoIdentitySelectMfaInput *) selectMfaInput
selectMfaCompletionSource: (AWSTaskCompletionSource<AWSCognitoIdentitySelectMfaDetails *>
*) selectMfaCompletionSource
{
    //keep a handle to the completion, you'll need it continue once you get the inputs
    from the end user
    self.selectMfaCompletionSource = selectMfaCompletionSource;
    // selectMfaInput has details about what MFAS are available to select.
}

-(void) didCompleteMfaSetupStepWithError:(NSError* _Nullable) error {
    dispatch_async(dispatch_get_main_queue(), ^{
        //on completion, either display the error or dismiss the ui
        if(error){
            [[UIAlertView alloc] initWithTitle:error.userInfo[@"__type"]
            message:error.userInfo[@"message"]
            delegate:nil
            cancelButtonTitle:nil
            otherButtonTitles:@"Retry", nil] show];
        }else{
            [self dismissViewControllerAnimated:YES completion:nil];
        }
    });
}
```

최종 사용자가 MFA를 선택한 경우, selectMfaCompletionSource에 대한 결과를 설정합니다.

```
AWSCognitoIdentitySoftwareMfaSetupRequiredDetails *details =
[[AWSCognitoIdentitySelectMfaDetails
alloc] initWithSelectedMfa: @"Selected MFA"
];
self.selectMfaCompletionSource.result = details;
```

예제: 암호 찾기

```
[[user forgotPassword] continueWithSuccessBlock:^(id
_Nullable(AWSTask<AWSCognitoIdentityUserForgotPasswordResponse*> * _Nonnull task) {
    //success
    return nil;
}];

[[user confirmForgotPassword:@"code" password:@"newPassword"] continueWithSuccessBlock:^(id
_Nullable(AWSTask<AWSCognitoIdentityUserConfirmForgotPasswordResponse *> * _Nonnull task)
{
    //success
    return nil;
}];
```

예제: Amazon Pinpoint 분석

다음 절차에서는 iOS Amazon Cognito 애플리케이션에 Amazon Pinpoint를 통합하는 방법을 설명합니다.

1. AWS Management 콘솔에 로그인한 다음 <https://console.aws.amazon.com/pinpoint/>에서 Amazon Pinpoint 콘솔을 엽니다.

2. Amazon Pinpoint 애플리케이션을 만듭니다. 앱 ID를 기록해 둡니다.
3. Amazon Cognito 앱에서 Amazon Cognito 사용자 풀 인스턴스를 인스턴스화하려면 `AWSCognitoIdentityUserPoolConfiguration`을 제공하고 이전 단계에서 Amazon Pinpoint 앱 ID를 사용하여 `pinpointAppId`를 설정합니다.

Objective-C:

```
AWSCognitoIdentityUserPoolConfiguration * poolConfiguration
= [[AWSCognitoIdentityUserPoolConfiguration alloc]
initWithClientId:@"YOUR_APP_CLIENT_ID"

               clientSecret:@"YOUR_OPTIONAL_APP_CLIENT_SECRET"

               poolId:@"YOUR_USER_POOL_ID"

               shouldProvideCognitoValidationData:YES

               pinpointAppId:@"YOUR_PINPOINT_APP_ID"];
```

Swift:

```
let poolConfiguration = AWSCognitoIdentityUserPoolConfiguration(clientId:
    "YOUR_APP_CLIENT_ID",
                                                                clientSecret:
    "YOUR_OPTIONAL_APP_CLIENT_SECRET",
                                                                poolId:
    "YOUR_USER_POOL_ID",
                                                                shouldProviderCognitoValidationData: YES,
                                                                pinpointAppId:
    "YOUR_PINPOINT_APP_ID")
```

인증 예제: 사용자 속성 가져오기

```
[[user getDetails] continueWithBlock:^(id
_Nullable(AWSTask<AWSCognitoIdentityUserGetDetailsResponse *> * _Nonnull task) {
    dispatch_async(dispatch_get_main_queue(), ^{
        if(task.error){
            [[[UIAlertView alloc] initWithTitle:task.error.userInfo[@"__type"]
                message:task.error.userInfo[@"message"]
                delegate:self
                cancelButtonTitle:nil
                otherButtonTitles:@"Retry", nil] show];
        }else{
            AWSCognitoIdentityUserGetDetailsResponse *response = task.result;
            //do something with response.userAttributes
        }
    });
    return nil;
}];
```

인증 예제: 사용자 속성 확인

```
[[user getAttributeVerificationCode:@"phone_number"] continueWithSuccessBlock:^(id
_Nullable(AWSTask<AWSCognitoIdentityUserGetAttributeVerificationCodeResponse *> * _Nonnull
task) {
    //success
    return nil;
}];
```

```
[[user verifyAttribute:@"phone_number"code:@"code"] continueWithSuccessBlock:^(id
_Nullable(AWSTask<AWSCognitoIdentityUserVerifyAttributeResponse *> * _Nonnull task) {
    //success
    return nil;
}];
```

인증 예제: 사용자 속성 업데이트

```
AWSCognitoIdentityUserAttributeType * attribute = [AWSCognitoIdentityUserAttributeType
new];
attribute.name = @"name";
attribute.value = @"John User";
[[user updateAttributes:@[attribute]] continueWithSuccessBlock:^(id
_Nullable(AWSTask<AWSCognitoIdentityUserUpdateAttributesResponse *> * _Nonnull task) {
    //success
    return nil;
}];
```

인증 예제: 암호 변경

```
[[user changePassword:@"currentPassword" proposedPassword:@"proposedPassword"]
continueWithSuccessBlock:^(id
_Nullable(AWSTask<AWSCognitoIdentityUserChangePasswordResponse *> * _Nonnull task) {
    //success
    return nil;
}];
```

인증 예제: SMS MFA 켜기

```
AWSCognitoIdentityUserSettings * settings = [AWSCognitoIdentityUserSettings new];
AWSCognitoIdentityUserMFAOption * mfaOptions = [AWSCognitoIdentityUserMFAOption new];
mfaOptions.attributeName = @"phone_number";
mfaOptions.deliveryMedium = AWSCognitoIdentityProviderDeliveryMediumTypeSms;
settings.mfaOptions = @[mfaOptions];
[[user setUserSettings:settings] continueWithSuccessBlock:^(id
_Nullable(AWSTask<AWSCognitoIdentityUserSetUserSettingsResponse *> * _Nonnull task) {
    //success
    return nil;
}];
```

인증 예제: 소프트웨어 MFA 켜기

```
//start by calling associateSoftwareToken to get a secret code for end user to register
[[self.user associateSoftwareToken] continueWithSuccessBlock:^(id
_Nullable(AWSTask<AWSCognitoIdentityUserAssociateSoftwareTokenResponse *> * _Nonnull t) {
    dispatch_async(dispatch_get_main_queue(), ^{
        UIAlertController *alert = [UIAlertController alertControllerWithTitle:@"Associate
Software MFA" message:t.result.secretCode preferredStyle:UIAlertControllerStyleAlert];
        [alert addAction:[UIAlertAction actionWithTitle:@"Verify"
style:UIAlertActionStyleDefault handler:^(UIAlertAction * _Nonnull action) {
            //verify the software token by having end user input current code
            [[self.user verifySoftwareToken:alert.textFields[0].text
friendlyDeviceName: @"My Software Token"] continueWithSuccessBlock:^(id
_Nullable(AWSTask<AWSCognitoIdentityUserVerifySoftwareTokenResponse *> * _Nonnull t) {
                //now the software token is configured, but not enabled, enable it
                AWSCognitoIdentityUserMfaPreferences * mfaPreferences =
                [AWSCognitoIdentityUserMfaPreferences new];
                mfaPreferences.softwareTokenMfa = [[AWSCognitoIdentityUserMfaType alloc]
initWithEnabled:YES preferred:NO];
```

```
[[self.user setUserMfaPreference:mfaPreferences]
continueWithSuccessBlock:^(id
_Nullable(AWSTask<AWSCognitoIdentityUserSetUserMfaPreferenceResponse *> * _Nonnull t) {
    return t;
}];
return nil;
}];
}]]];
[alert addTextFieldWithConfigurationHandler:^(UITextField *textField) {
    textField.placeholder = @"User Code:";
}];
[self presentViewController:alert animated:YES completion:nil];
});
return nil;
}];
```

예제: Lambda 트리거를 사용한 iOS 사용자 마이그레이션

사용자 마이그레이션 Lambda 트리거는 기존 사용자 관리 시스템에서 암호 재설정 없이 사용자 풀로의 손쉬운 마이그레이션을 허용합니다.

사용자 마이그레이션 Lambda 트리거 설정

iOS 앱을 변경하기 전에 사용자 풀에 대한 사용자 마이그레이션 Lambda를 설정합니다.

Lambda 트리거에 대해 자세히 알아보려면 [Lambda 트리거를 사용하여 사용자 풀 워크플로우 사용자 지정 \(p. 118\)](#) 단원을 참조하십시오.

Lambda 트리거를 사용한 사용자 마이그레이션에 대한 자세한 내용은 [사용자 마이그레이션 Lambda 트리거를 사용하여 사용자 풀로 사용자 가져오기 \(p. 171\)](#) 단원을 참조하십시오.

iOS 앱 변경

1. SDK 업데이트

AWSCognitoIdentityProvider iOS SDK를 버전 2.6.12 이상으로 업데이트합니다.

2. 마이그레이션 활성화

Info.plist를 사용하여 사용자 풀을 구성하는 경우:

값이 MigrationEnabled인 부울 YES 키를 추가합니다. Info.plist에서 Open As->Source Code를 실행하면 다음과 같은 모습이어야 합니다.

```
<key>AWS</key>
<dict>
    <key>CognitoUserPool</key>
    <dict>
        <key>Default</key>
        <dict>
            <key>AppClientId</key>
            <string>YOUR_APP_CLIENT_ID</string>
            <key>PoolId</key>
            <string>region_YOUR_USER_POOL_ID </string>
            <key>Region</key>
            <string>us-west-2</string>
            <key>MigrationEnabled</key>
            <true/>
        </dict>
    </dict>
```

```
</dict>
</dict>
```

사용자 마이그레이션 인증 흐름

기존 시스템에 대한 사용자를 인증하고 암호를 확인한 다음 프로필을 사용자 풀로 원활하게 마이그레이션할 수 있습니다. 하지만 암호 재설정을 회피하려면 서비스에 기존 암호가 필요합니다. 따라서 인증 흐름에서 명시적으로 활성화된 경우 SDK가 암호화된 SSL 연결을 통해 텍스트로 서비스에 사용자의 암호를 전송합니다.

AWSCognitoIdentityUserPoolConfiguration을 사용하여 사용자 풀을 구성하는 경우 이니셜라이저를 migrationEnabled 플래그를 지원하는 것으로 변경합니다.

Objective-C

```
AWSCognitoIdentityUserPoolConfiguration * poolConfiguration
= [[AWSCognitoIdentityUserPoolConfiguration alloc]
  initWithClientId:@"YOUR_APP_CLIENT_ID"

               clientSecret:@"YOUR_OPTIONAL_APP_CLIENT_SECRET"

               poolId:@"YOUR_USER_POOL_ID"

  shouldProvideCognitoValidationData:YES

               pinpointAppId:@"YOUR_OPTIONAL_PINPOINT_APP_ID"

               migrationEnabled:YES];
```

Swift

```
let poolConfiguration = AWSCognitoIdentityUserPoolConfiguration(clientId:
  "YOUR_APP_CLIENT_ID",
                                                                clientSecret:
  "YOUR_OPTIONAL_APP_CLIENT_SECRET",
                                                                poolId:
  "YOUR_USER_POOL_ID",
                                                                shouldProviderCognitoValidationData: YES,
                                                                pinpointAppId:
  "YOUR_OPTIONAL_PINPOINT_APP_ID",
                                                                migrationEnabled: YES)
```

Amazon Cognito 호스팅 UI를 사용하여 등록 및 로그인

사용자 풀에서 앱을 생성하여 사용자 가입 및 로그인용 내장 웹 페이지를 사용할 수 있습니다. Amazon Cognito의 호스팅된 UI는 OpenID Connect(OIDC) 및 SAML 자격 증명 공급자는 물론 Facebook, Amazon 및 Google을 통해서도 사용자 풀에 직접 로그인할 수 있는 기능을 포함해 다른 여러 기능을 위한 기반입니다.

주제

- 앱 클라이언트 추가(AWS Management 콘솔) (p. 72)
- 사용자 풀 앱 클라이언트 구성 (p. 73)
- 사용자 풀 도메인 구성 (p. 77)

- 내장 로그인 및 가입 웹페이지 사용자 지정 (p. 82)
- 사용자 풀의 리소스 서버 정의 (p. 86)

앱 클라이언트 추가(AWS Management 콘솔)

1. [Amazon Cognito 콘솔](#)로 이동합니다. AWS 자격 증명을 입력하라는 메시지가 나타날 수 있습니다.
2. [Manage your User Pools]를 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 [사용자 풀을 생성합니다](#).
4. 페이지 왼쪽에 있는 탐색 모음의 일반 설정에서 앱 클라이언트를 선택합니다.
5. [Add an app client]를 선택합니다.
6. 앱 이름을 지정합니다.
7. 인증 흐름에서 필요한 경우를 제외하고 옵션 클라이언트 보안키 생성을 지웁니다. 클라이언트 암호는 클라이언트 암호를 보호할 수 있는 서버 측 구성요소가 있는 애플리케이션에서 사용됩니다.
8. [Create app client]를 선택합니다.
9. 앱 클라이언트 ID를 메모합니다.
10. 풀 세부 정보로 돌아가기를 선택합니다.
11. 앱을 구성합니다.
 - a. 콘솔 페이지 왼쪽에 있는 탐색 모음에서 앱 클라이언트 설정을 선택합니다.
 - b. 활성화된 자격 증명 공급자 중 하나로 Cognito User Pool(Cognito 사용자 풀)을 선택합니다.

Note

Facebook, Amazon 및 Google 같은 외부 자격 증명 공급자(IdP)를 비롯해 OpenID Connect(OIDC) 또는 SAML IdP를 통해 로그인하도록 허용하려면 먼저 다음 단원에 설명된 대로 이들을 구성한 다음, 앱 클라이언트 설정으로 돌아가 이들을 활성화합니다.

- c. Amazon Cognito 권한 부여 서버에 대한 콜백 URL을 입력하여 사용자가 인증된 후 호출합니다. 웹 앱의 경우 URL이 `https://`로 시작해야 합니다(예: `https://www.example.com`).

iOS 또는 Android 앱의 경우에는 `myapp://`와 같은 콜백 URL을 사용할 수 있습니다.
- d. Authorization code grant(권한 부여 코드 허용)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. 보안을 위해 모바일 앱에서는 [Proof Key for Code Exchange \(PKCE\)](#)와 함께 권한 부여 코드 허용 흐름을 사용하는 것이 좋습니다.

허용된 OAuth Flows에서 Implicit grant(암시적 허용)를 선택하여 Amazon Cognito로부터 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.

Authorization code grant(권한 부여 코드 허용) 및 Implicit code grant(암시적 코드 부여)를 모두 활성화하고 각각을 필요한 경우 사용할 수 있습니다.

특별히 하나를 제외하고 싶은 경우가 아니라면 허용된 OAuth Scopes 확인란을 모두 선택합니다.

Note

사용자를 대신해서가 아니라 자체적으로 액세스 토큰을 요청해야 하는 경우에만 Client credentials(클라이언트 자격 증명)를 선택합니다.

- e. [Save changes]를 선택합니다.
12. 도메인을 구성합니다.
 - a. 도메인 이름 페이지에서 사용 가능한 도메인 접두사를 입력합니다.

- b. 전체 도메인 주소를 기록해 둡니다.
- c. [Save changes]를 선택합니다.

로그인 페이지를 보려면

다음 URL을 사용하여 호스팅 UI 로그인 웹 페이지를 볼 수 있습니다. `response_type`을 기록합니다. 이 경우 `response_type=code`는 인증 코드 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=token`은 묵시적 코드 부여입니다. 로그인 성공 이후는 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.

묵시적 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
```

AWS Lambda를 사용하여 사용자 풀 토큰을 디코딩 및 확인할 수 있습니다. AWS GitHub 웹 사이트에서 [Decode and verify Amazon Cognito JWT tokens](#)를 참조하십시오.

Amazon Cognito 사용자 풀 토큰은 RS256 알고리즘을 사용하여 서명됩니다.

콘솔 변경 사항이 표시되기 전까지 브라우저를 새로 고침하는 데 잠시 기다려야 할 수 있습니다.

도메인 이름 페이지에 도메인이 표시됩니다. 앱 클라이언트 설정 페이지에 앱 클라이언트 ID와 콜백 URL이 표시됩니다.

Note

Amazon Cognito 호스팅 로그인 웹 페이지는 사용자 지정 인증 흐름을 지원하지 않습니다.

사용자 풀 앱 클라이언트 구성

사용자 풀을 생성한 이후 앱 클라이언트를 생성하여 사용자 가입 및 로그인에 대해 내장 웹 페이지를 사용할 수 있습니다. AWS Management 콘솔을 사용하여 앱 클라이언트 및 Amazon Cognito 호스팅 도메인을 추가하는 방법은 [앱을 추가하여 호스팅 웹 UI 활성화](#)를 참조하십시오.

앱 클라이언트 설정 개요

활성화된 자격 증명 공급자

사용자를 인증할 자격 증명 공급자(IdP)를 선택할 수 있습니다. 이 서비스는 사용자 풀로 수행할 수 있고 또는 Facebook 같은 타사가 수행할 수도 있습니다. IdP를 사용하기 전에 이것을 활성화해야 합니다. 다

중 IdP를 활성화할 수도 있지만 최소한 한 개는 활성화해야 합니다. 외부 IdP 사용에 대한 자세한 내용은 [타사를 통한 사용자 풀 로그인 추가 \(p. 88\)](#) 섹션을 참조하십시오.

콜백 URL(여러 개 가능)

콜백 URL은 로그인 성공 후 사용자가 리디렉션되는 위치를 나타냅니다. 하나 이상의 콜백 URL을 다음과 같이 선택합니다.

- 절대 URI이어야 합니다.
- 클라이언트로 미리 등록되어야 합니다.
- 조각 구성요소가 없어야 합니다.

[OAuth 2.0 - Redirection Endpoint](#) 단원을 참조하십시오.

테스트 목적으로만 `http://localhost`를 사용하는 경우를 제외하고 Amazon Cognito에서는 HTTP가 아니라 HTTPS가 필요합니다.

`myapp://example` 같은 앱 콜백 URL도 지원됩니다.

로그아웃 URL(여러 개 가능)

로그아웃 URL은 로그아웃 후 사용자가 리디렉션되는 위치를 나타냅니다.

허용된 OAuth 흐름

권한 부여 코드 허용 흐름은 응답으로 권한 부여 코드를 제공하는 코드 허용 흐름을 시작합니다. 이 코드는 [토큰 엔드포인트 \(p. 320\)](#)를 통해 액세스 토큰과 교환할 수 있습니다. 토큰은 최종 사용자에게 직접 노출되지 않기 때문에 침해될 가능성이 낮습니다. 하지만 사용자 풀 토큰에 대한 인증 코드를 교환하려면 백엔드에 사용자 지정 애플리케이션이 필요합니다.

Note

보안상의 이유로 모바일 앱에서는 PKCE와 함께 권한 부여 코드 허용 흐름만 사용하는 것이 좋습니다.

클라이언트는 암시적 허용 흐름을 통해 [권한 부여 엔드포인트 \(p. 316\)](#)에서 직접 액세스 토큰(및 범위에 따라 선택 사항으로 ID 토큰)을 가져올 수 있습니다. 앱이 권한 부여 코드 허용 흐름을 시작할 수 없는 경우에는 이 흐름을 선택합니다. 자세한 내용은 [OAuth 2.0 사양](#)을 참조하십시오.

Authorization code grant(권한 부여 코드 허용) 및 Implicit code grant(암시적 코드 부여)를 모두 활성화하고 각각을 필요한 경우 사용할 수 있습니다.

Client credentials(클라이언트 자격 증명) 흐름은 기계 간 커뮤니케이션에 사용됩니다. 이를 통해 사용자 리소스에 액세스하기 위한 액세스 토큰을 요청할 수 있습니다. 앱이 사용자를 대신해서가 아니라 자체적으로 액세스 토큰을 요청하고 있을 때 이 흐름을 사용합니다.

Note

클라이언트 자격 증명 흐름은 사용자 대신 사용되지 않으므로, 사용자 지정 범위만 이 흐름과 함께 사용됩니다. 사용자 지정 범위는 자체 리소스 서버에 대해 정의할 수 있습니다. [사용자 풀의 리소스 서버 정의 \(p. 86\)](#) 단원을 참조하십시오.

허용된 OAuth 범위

다음 OAuth 범위에서 하나 이상을 선택하여 액세스 토큰에 대해 요청될 수 있는 액세스 권한을 지정합니다.

- `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.

- `openid` 범위는 ID 토큰에서 클라이언트가 읽을 수 있는 모든 사용자 속성을 반환합니다. 클라이언트가 `openid` 범위를 요청하지 않으면 ID 토큰이 반환되지 않습니다.
- `aws.cognito.signin.user.admin` 범위는 [UpdateUserAttributes](#) 및 [VerifyUserAttribute](#)와 같이 액세스 토큰이 필요한 [Amazon Cognito 사용자 풀 API 작업](#)에 대한 액세스 권한을 부여합니다.
- `profile` 범위는 클라이언트가 읽을 수 있는 모든 사용자 속성에 대한 액세스 권한을 부여합니다.

허용된 사용자 지정 범위

사용자 지정 범위는 리소스 서버에서 자체 리소스 서버에 대해 정의할 수 있습니다. 형식은 `resource-server-identifier/scope`입니다. [사용자 풀의 리소스 서버 정의 \(p. 86\)](#) 단원을 참조하십시오.

OAuth 범위에 대한 자세한 내용은 [표준 OIDC 범위 목록](#)을 참조하십시오.

앱 클라이언트를 구성하려면(AWS Management 콘솔)

1. [Amazon Cognito 콘솔](#)로 이동합니다. AWS 자격 증명을 입력하라는 메시지가 나타날 수 있습니다.
2. Manage User Pools(사용자 풀 관리)를 선택합니다.
3. 편집할 사용자 풀을 선택합니다.
4. 콘솔 페이지 왼쪽에 있는 탐색 모음에서 앱 클라이언트 설정을 선택합니다.
5. 활성화된 자격 증명 공급자 중 하나로 Cognito User Pool(Cognito 사용자 풀)을 선택하여 IdP에 포함합니다. 하나 이상의 IdP를 활성화하십시오.

Note

Facebook, Amazon 및 Google 같은 외부 자격 증명 공급자(IdP)를 비롯해 OpenID Connect(OIDC) 또는 SAML IdP를 통해 로그인하도록 허용하려면 먼저 이들을 구성한 다음, 앱 클라이언트 설정 페이지로 돌아가 이들을 활성화합니다. 자세한 내용은 [타사를 통한 사용자 풀 로그인 추가 \(p. 88\)](#) 단원을 참조하십시오.

6. 로그인 및 로그아웃 URL 섹션에서 콜백 URL을 쉼표로 구분하여 입력합니다.

웹 앱의 경우 URL이 `https://`로 시작해야 합니다(예: `https://www.example.com`).

iOS 또는 Android 앱의 경우에는 `myapp://`와 같은 콜백 URL을 사용할 수 있습니다.

Note

사용자 풀 앱 클라이언트에서 URL을 사용하려면 콘솔에서 콜백 및 로그아웃 URL을 등록하거나 CLI 또는 API를 사용하여 URL을 등록해야 합니다.

7. 선택 사항인 로그아웃 URL을 쉼표로 구분하여 입력합니다.
8. OAuth 2.0 옵션을 선택합니다.

- 권한 있는 코드 허용
- 암시적 허용
- 클라이언트 자격 증명

클라이언트 자격 증명을 선택하면 다른 선택 사항이 지워지고 표준 범위를 사용할 수 없습니다.

9. a. 코드 허용과 암시적 허용 선택 사항에서 허용된 OAuth Scopes에서 범위를 선택하십시오. 각 범위는 하나 이상의 표준 속성 집합입니다. 자세한 내용은 [앱 클라이언트 설정 개요 \(p. 73\)](#) 단원을 참조하십시오.
b. 클라이언트 자격 증명 흐름에서는 표준 범위를 사용할 수 없습니다. 클라이언트 자격 증명 흐름은 사용자 대신 사용되지 않으므로, 사용자 지정 범위만 사용됩니다. 사용자 지정 범위는 자체 리소스 서버에 대해 정의할 수 있습니다. [사용자 풀의 리소스 서버 정의 \(p. 86\)](#) 단원을 참조하십시오.
10. [Save changes]를 선택합니다.

앱 클라이언트를 구성하려면(AWS CLI 및 AWS API)

AWS CLI를 사용하여 사용자 풀 앱 클라이언트를 업데이트, 생성, 설명 및 삭제할 수 있습니다.

"MyUserPoolID" 및 "MyAppClientID"를 사용자 풀과 이 예제의 앱 클라이언트 ID 값으로 대체합니다. 마찬가지로 사용자의 파라미터 값이 이 예제에서 사용한 값과 다를 수 있습니다.

Note

CLI가 원격 파라미터 파일로 처리하지 않도록 콜백 및 로그아웃 URL에 JSON 형식을 사용합니다.

```
--callback-urls '["https://example.com"]'
```

```
--logout-urls '["https://example.com"]'
```

사용자 풀 앱 클라이언트 업데이트(AWS CLI 및 AWS API)

```
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 성공할 경우 AWS CLI가 확인을 반환합니다.

```
{  
  "UserPoolClient": {  
    "ClientId": "MyClientId",  
    "SupportedIdentityProviders": [  
      "LoginWithAmazon",  
      "MySAMLIdP"  
    ],  
    "CallbackURLs": [  
      "https://example.com"  
    ],  
    "AllowedOAuthScopes": [  
      "openid"  
    ],  
    "ClientName": "Example",  
    "AllowedOAuthFlows": [  
      "implicit",  
      "code"  
    ],  
    "RefreshTokenValidity": 30,  
    "CreationDate": 1524628110.29,  
    "AllowedOAuthFlowsUserPoolClient": true,  
    "UserPoolId": "MyUserPoolID",  
    "LastModifiedDate": 1530055177.553  
  }  
}
```

자세한 내용은 다음 AWS CLI 명령 레퍼런스를 참조하십시오. [update-user-pool-client](#).

AWS API: [UpdateUserPoolClient](#)

사용자 풀 앱 클라이언트 생성(AWS CLI 및 AWS API)

```
aws cognito-idp create-user-pool-client --user-pool-id MyUserPoolID --client-name myApp
```

자세한 내용은 다음 AWS CLI 명령 레퍼런스를 참조하십시오. [create-user-pool-client](#).

AWS API: [CreateUserPoolClient](#)

사용자 풀 앱 클라이언트에 대한 정보를 가져오기(AWS CLI 및 AWS API)

```
aws cognito-idp describe-user-pool-client --user-pool-id MyUserPoolID --client-id MyClientID
```

자세한 내용은 다음 AWS CLI 명령 레퍼런스를 참조하십시오. [describe-user-pool-client](#).

AWS API: [DescribeUserPoolClient](#)

사용자 풀에 모든 앱 클라이언트 정보 나열(AWS CLI 및 AWS API)

```
aws cognito-idp list-user-pool-clients --user-pool-id "MyUserPoolID" --max-results 3
```

자세한 내용은 다음 AWS CLI 명령 레퍼런스를 참조하십시오. [list-user-pool-clients](#).

AWS API: [ListUserPoolClients](#)

사용자 풀 앱 클라이언트 삭제(AWS CLI 및 AWS API)

```
aws cognito-idp delete-user-pool-client --user-pool-id "MyUserPoolID" --client-id "MyAppClientID"
```

자세한 내용은 다음 AWS CLI 명령 레퍼런스를 참조하십시오. [delete-user-pool-client](#).

AWS API: [DeleteUserPoolClient](#)

사용자 풀 도메인 구성

앱 클라이언트를 설정한 후에 등록 및 로그인 웹 페이지의 주소를 구성할 수 있습니다. Amazon Cognito 호스트 도메인을 사용하고 사용 가능한 도메인 접두사를 선택하거나 사용자 지정 도메인으로 자체 웹 주소를 사용할 수 있습니다.

AWS Management 콘솔을 사용하여 앱 클라이언트 및 Amazon Cognito 호스팅 도메인을 추가하는 방법은 [앱을 추가하여 호스팅 웹 UI 활성화](#)를 참조하십시오.

주제

- [호스팅 UI에 Amazon Cognito 도메인 사용 \(p. 77\)](#)
- [호스팅 UI에 자체 도메인 사용 \(p. 79\)](#)

호스팅 UI에 Amazon Cognito 도메인 사용

앱 클라이언트를 설정한 후에 등록 및 로그인 웹 페이지의 주소를 구성할 수 있습니다. 자체 도메인 접두사를 가진 호스팅된 Amazon Cognito 도메인을 사용할 수 있습니다.

AWS Management 콘솔을 사용하여 앱 클라이언트 및 Amazon Cognito 호스팅 도메인을 추가하는 방법은 [앱을 추가하여 호스팅 웹 UI 활성화](#)를 참조하십시오.

주제

- [사전 조건 \(p. 78\)](#)
- [1단계: 호스팅된 사용자 풀 도메인 구성 \(p. 78\)](#)
- [2단계: 로그인 페이지 확인 \(p. 78\)](#)

사전 조건

시작하려면 다음이 필요합니다.

- 앱 클라이언트가 포함된 사용자 풀. 자세한 내용은 [사용자 풀 시작하기 \(p. 13\)](#) 단원을 참조하십시오.

1단계: 호스팅된 사용자 풀 도메인 구성

호스팅된 사용자 풀 도메인을 구성하는 방법(AWS Management 콘솔)

AWS Management 콘솔을 사용하여 사용자 풀 도메인을 구성할 수 있습니다.

Amazon Cognito 호스팅 도메인을 구성하는 방법

1. [Amazon Cognito 콘솔](#)에 로그인합니다.
2. 탐색 창에서 사용자 풀 관리를 선택한 다음 편집할 사용자 풀을 선택합니다.
3. 도메인 이름 탭을 선택합니다.
4. Prefix domain name(접두사 도메인 이름) 상자에서 사용하려는 도메인 접두사를 입력합니다.
5. 가용성 확인을 선택하여 도메인 접두사가 사용 가능한지 확인합니다.
6. [Save changes]를 선택합니다.

호스팅된 사용자 풀 도메인을 구성하는 방법(AWS CLI 및 AWS API)

다음 명령을 사용하여 도메인 접두사를 생성하여 사용자 풀에 할당합니다.

사용자 풀 도메인을 구성하는 방법

- 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 정보 가져오기

- AWS CLI: `aws cognito-idp describe-user-pool-domain`

예: `aws cognito-idp describe-user-pool-domain --domain <domain_name>`

- AWS API: [DescribeUserPoolDomain](#)

도메인 삭제

- 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 호스팅 도메인에서 로그인 페이지가 사용 가능한지 확인합니다.

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

Amazon Cognito 콘솔의 도메인 이름 페이지에 도메인이 표시됩니다. 앱 클라이언트 설정 페이지에 앱 클라이언트 ID와 콜백 URL이 표시됩니다.

호스팅 UI에 자체 도메인 사용

앱 클라이언트를 설정한 다음 Amazon Cognito 호스팅 UI를 위한 사용자 지정 도메인으로 사용자 풀을 구성할 수 있습니다. 사용자 지정 도메인을 통해 사용자가 자체 웹 주소로 애플리케이션에 로그인하도록 할 수 있습니다.

주제

- 사용자 풀에 사용자 지정 도메인 추가 (p. 79)
- 사용자 지정 도메인의 SSL 인증서 변경 (p. 81)

사용자 풀에 사용자 지정 도메인 추가

사용자 풀에 사용자 지정 도메인을 추가하려면 Amazon Cognito 콘솔에서 도메인 이름을 지정하고 [AWS Certificate Manager\(ACM\)](#)로 관리하는 인증서를 제공해야 합니다. 도메인을 추가하고 나면 Amazon Cognito에서 별칭 대상을 제공하며, 이를 DNS 구성에 추가하면 됩니다.

사전 조건

시작하려면 다음이 필요합니다.

- 앱 클라이언트가 포함된 사용자 풀. 자세한 내용은 [사용자 풀 시작하기 \(p. 13\)](#) 단원을 참조하십시오.
- 사용자가 소유하고 있는 웹 도메인입니다. 도메인의 루트에는 DNS에 유효한 A 레코드가 있어야 합니다. 자세한 내용은 [도메인 이름](#)을 참조하십시오.
- 사용자 지정 도메인을 위한 하위 도메인을 만들 수 있습니다. 하위 도메인으로 auth를 사용하는 것이 좋습니다. 예를 들면 [auth.example.com](#)을 사용합니다.

Note

[와일드카드 인증서](#)가 없는 경우에는 사용자 지정 도메인의 하위 도메인에 대해 새 인증서를 얻어야 합니다.

- ACM으로 관리하는 Secure Sockets Layer(SSL) 인증서입니다. 인증서를 요청하거나 가져오기 전에 ACM 콘솔에서 AWS 리전을 미국 동부(버지니아 북부)로 변경해야 합니다.
- 사용자 지정 도메인 이름을 설정하거나 해당 인증서를 업데이트하려면 Amazon CloudFront 배포를 업데이트할 수 있는 권한이 있어야 합니다. 이를 위해 다음 IAM 정책 명령문을 AWS 계정의 IAM 사용자, 그룹 또는 역할에 연결할 수 있습니다.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "AllowCloudFrontUpdateDistribution",  
      "Effect": "Allow",  
      "Action": [  
        "cloudfront:updateDistribution"  
      ],  
      "Resource": [  
        "*"   
      ]  
    }  
  ]  
}
```

```
}  
    ]  
}
```

CloudFront에 대한 자격 증명 기반 정책(IAM 정책) 사용을 참조하십시오.

1단계: 사용자 지정 도메인 이름 입력

Amazon Cognito 콘솔 또는 API를 사용하여 사용자 풀에 도메인을 추가할 수 있습니다.

사용자 풀에 도메인을 할당하려면(Amazon Cognito 콘솔)

1. [Amazon Cognito 콘솔](#)에 로그인합니다. AWS 자격 증명을 입력하라는 메시지가 나타날 수 있습니다.
2. AWS Management 콘솔에 로그인한 다음 <https://console.aws.amazon.com/cognito/home>에서 Amazon Cognito 콘솔을 엽니다.
3. Manage User Pools(사용자 풀 관리)를 선택합니다.
4. 사용자 풀 페이지에서 도메인을 추가할 사용자 풀을 선택합니다.
5. 왼쪽의 탐색 메뉴에서 도메인 이름을 선택합니다.
6. Your own domain(자체 도메인)에서 Use your domain(자체 도메인 사용)을 선택합니다.
7. 도메인 이름에 사용자 지정 도메인 이름을 입력합니다. 도메인 이름에는 소문자, 숫자, 하이픈(-)만 사용할 수 있습니다. 첫 번째나 마지막 자리에 하이픈을 사용하지 마십시오. 마침표로 하위 도메인 이름을 구분하십시오.
8. AWS managed certificate(AWS 관리형 인증서)에서 해당 도메인에 사용할 SSL 인증서를 선택합니다. ACM로 관리하는 인증서 중 하나를 선택할 수 있습니다.

선택할 수 있는 인증서가 없는 경우, ACM를 사용하여 프로비저닝할 수 있습니다. 자세한 내용은 AWS Certificate Manager 사용 설명서의 [시작하기](#)를 참조하십시오.

9. [Save changes]를 선택합니다.
10. Alias target(별칭 대상)을 적어 둡니다. IP 주소나 도메인 이름 대신에 Alias target(별칭 대상)은 Amazon CloudFront 배포를 가리키는 별칭 리소스 레코드 집합입니다.

사용자 풀에 도메인을 할당하려면(Amazon Cognito API)

- [CreateUserPoolDomain](#) 작업을 사용합니다.

2단계: 별칭 대상 및 하위 도메인 추가

이 단계에서 이전 단계에서 나온 별칭 대상을 다시 가리키는 DNS(Domain Name Server) 서비스 공급자를 통해 별칭을 설정합니다. DNS 주소 확인을 위해 Amazon Route 53을 사용하고 있는 경우에는 Route 53을 사용하여 별칭 대상 및 하위 도메인을 추가하는 방법 단원을 참조하십시오.

현재 DNS 구성에 별칭 대상 및 하위 도메인을 추가하는 방법

- DNS 주소 확인에 Route 53을 사용하고 있지 않은 경우에는 DNS 서비스 공급자가 이전 단계에서 나온 별칭 대상을 사용자 풀 사용자 지정 도메인을 위한 별칭으로 추가할 수 있도록 허용해야 합니다. 또한 DNS 공급자는 사용자 지정 도메인에서 하위 도메인을 설정해야 합니다.

Route 53을 사용하여 별칭 대상 및 하위 도메인을 추가하는 방법

1. [Route 53 콘솔](#)에 로그인합니다. AWS 자격 증명을 입력하라는 메시지가 나타날 수 있습니다.
2. Route 53에 호스팅 영역이 없는 경우에는 영역을 설정합니다. 그렇지 않으면 이 단계를 건너뜁니다.
 - a. 호스팅 영역 만들기를 선택합니다.

- b. 도메인 이름 목록에서 사용자 지정 도메인을 선택합니다.
 - c. 선택에 따라 설명에서 호스팅 영역에 대한 설명을 입력합니다.
 - d. Create를 선택합니다.
3. Hosted Zones(호스팅 영역) 페이지에서 호스팅 영역의 이름을 선택합니다.
4. [Create Record Set]를 선택합니다.
5. 별칭 옵션에 대해 예를 선택합니다.
6. 이전 단계에서 적어둔 별칭 대상 이름을 Alias Target(별칭 대상)에 입력합니다.
7. Create를 선택합니다.

Note

새 레코드는 Route 53 DNS 서버로 전파되기까지 시간이 걸립니다. 현재 변경 사항의 전파 여부를 확인하는 유일한 방법은 Route 53 [GetChange](#) API 메서드를 사용하는 것입니다. 변경 사항은 일반적으로 60초 이내에 모든 Route 53 이름 서버로 전파됩니다.

8. 별칭 대상을 사용하여 Route 53에서 하위 도메인을 추가합니다.
 - a. Hosted Zones(호스팅 영역) 페이지에서 호스팅 영역의 이름을 선택합니다.
 - b. Create Record Set(레코드 집합 생성)을 선택하고 다음 값을 입력합니다.
 - i. 이름에서 기본 설정된 하위 도메인 이름을 입력합니다. 예를 들어 생성을 시도 중인 하위 도메인이 `auth.example.com`인 경우에는 `auth`를 입력합니다.
 - ii. [Type]에서 [A - IPv4 address]를 선택합니다.
 - iii. 별칭 옵션에 대해 예를 선택합니다.
 - iv. 이전 단계에서 적어둔 별칭 대상 이름을 Alias Target(별칭 대상)에 입력합니다.
 - c. Create를 선택합니다.

Note

또는 하위 도메인에 연결된 레코드를 저장할 호스팅 영역을 새로 생성합니다. 또한 하위 도메인 호스팅 영역으로 클라이언트를 위임하는 상위 호스팅 영역에 설정된 위임을 생성할 수 있습니다. 이는 호스팅 영역을 관리하고 있을 때 훨씬 유연한 방법입니다(예를 들어 영역을 편집할 수 있는 사용자 제한). 프라이빗 호스팅 영역에 NS 레코드를 추가하는 방법이 현재 지원되고 있지 않기 때문에 이 방법은 퍼블릭 호스팅 영역에서만 사용할 수 있습니다. 자세한 내용은 [Amazon Route 53을 통해 호스팅된 도메인에서 하위 도메인 생성](#)을 참조하십시오.

3단계: 로그인 페이지 확인

- 사용자 지정 도메인에서 로그인 페이지가 사용 가능한지 확인합니다.

이 주소를 브라우저에 입력하여 사용자 지정 도메인 및 하위 도메인에 로그인합니다. 이는 하위 도메인이 `auth`인 사용자 지정 도메인 `example.com`의 URL 예제입니다.

```
https://auth.example.com/login?
response_type=code&client_id=<your_app_client_id>&redirect_uri=<your_callback_url>
```

사용자 지정 도메인의 SSL 인증서 변경

필요한 경우, 하위 도메인에 적용한 인증서를 Amazon Cognito에서 변경할 수 있습니다.

대개 ACM로 정기 인증서 갱신을 한 뒤에는 이렇게 할 필요가 없습니다. ACM에서 기존 인증서를 갱신해도 해당 인증서의 ARN은 그대로 유지되며, 사용자 지정 도메인에서는 자동으로 새 인증서를 사용합니다.

그러나 기존 인증서를 새 인증서로 바꾸면 ACM가 새 인증서에 새 ARN을 부여합니다. 사용자 지정 도메인에 새 인증서를 적용하려면 이 ARN을 Amazon Cognito에 입력해야 합니다.

새 인증서를 입력한 뒤 Amazon Cognito에서 이를 사용자 지정 도메인에 배포할 때까지는 최대 1시간이 걸립니다.

시작하기 전에

Amazon Cognito에서 인증서를 변경하려면 먼저 인증서를 ACM에 추가해야 합니다. 자세한 내용은 AWS Certificate Manager 사용 설명서의 [시작하기](#)를 참조하십시오.
ACM에 인증서를 추가할 때 AWS 리전으로 미국 동부(버지니아 북부)를 선택해야 합니다.

Amazon Cognito 콘솔 또는 API를 사용하여 인증서를 변경할 수 있습니다.

인증서를 갱신하려면(Amazon Cognito 콘솔)

1. AWS Management 콘솔에 로그인한 다음 <https://console.aws.amazon.com/cognito/home>에서 Amazon Cognito 콘솔을 엽니다.
2. Manage User Pools(사용자 풀 관리)를 선택합니다.
3. 사용자 풀 페이지에서 도메인을 추가할 사용자 풀을 선택합니다.
4. 왼쪽의 탐색 메뉴에서 도메인 이름을 선택합니다.
5. Your own domain(자체 도메인)에서 AWS 관리형 인증서로 새 인증서를 선택합니다.
6. [Save changes]를 선택합니다.

인증서를 갱신하려면(Amazon Cognito API)

- [UpdateUserPoolDomain](#) 작업을 사용합니다.

내장 로그인 및 가입 웹페이지 사용자 지정

AWS Management 콘솔이나 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. 다수의 CSS 사용자 지정을 선택할 수도 있습니다.

단일 클라이언트(특정 `clientId`를 가진) 또는 모든 클라이언트(`clientId`를 ALL로 설정)에서 앱 UI 사용자 지정 설정을 지정할 수 있습니다. ALL을 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.

Note

이 기능을 사용하려면 사용자 풀에 기능과 연결된 도메인이 있어야 합니다.

앱의 사용자 지정 로고 지정

로고 이미지에 허용되는 최대 파일 크기는 100KB입니다.

앱의 CSS 사용자 지정 로고 지정

호스팅된 앱 페이지에서 CSS를 사용자 지정할 수 있지만, 다음과 같은 제한 사항이 있습니다.

- CSS 클래스 이름은 다음 목록에만 포함될 수 있습니다.
 - `background-customizable`

- banner-customizable
 - errorMessage-customizable
 - idpButton-customizable
 - idpButton-customizable: hover
 - inputField-customizable
 - inputField-customizable: focus
 - label-customizable
 - legalText-customizable
 - logo-customizable
 - submitButton-customizable
 - submitButton-customizable: hover
 - textDescription-customizable
- 속성 값에는 HTML, @import, @supports, @page, @media 문이나 Javascript가 포함될 수 없습니다.

다음 CSS 속성을 사용자 지정할 수 있습니다.

레이블

- font-weight는 100에서 900 사이의 100의 배수입니다.

입력 필드

- width는 포함 블록의 비율로 표시되는 너비입니다.
- height는 픽셀(px) 단위의 입력 필드 높이입니다.
- color는 텍스트 색상입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.
- background-color는 입력 필드의 배경색입니다. 표준 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 inline일 수 있습니다.
- font-size는 텍스트 설명의 글꼴 크기입니다.

제출 버튼

- font-size는 버튼 텍스트의 글꼴 크기입니다.
- font-weight는 버튼 텍스트의 글꼴 두께로 bold, italic 또는 normal일 수 있습니다.
- margin은 버튼의 위쪽, 오른쪽, 아래쪽 및 왼쪽 마진 크기를 나타내는 4자 문자열입니다.
- font-size는 텍스트 설명의 글꼴 크기입니다.
- 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-color`는 커서를 올려 놓았을 때 버튼의 배경색입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.

자격 증명 공급자 버튼 가리키기

- `color`는 커서를 올려 놓았을 때 버튼의 전경색입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.
- `background-color`는 커서를 올려 놓았을 때 버튼의 배경색입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.

암호 확인이 유효하지 않음

- `color`는 "Password check not valid" 메시지의 텍스트 색상입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.

배경

- `background-color`는 앱 창의 배경색입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.

오류 메시지

- `margin`은 위쪽, 오른쪽, 아래쪽 및 왼쪽 마진 크기를 나타내는 4자 문자열입니다.
- `padding`은 패딩 크기입니다.
- `font-size`는 글꼴 크기입니다.
- `width`는 포함 블록의 비율로 표시되는 오류 메시지의 너비입니다.
- `background-color`는 오류 메시지의 배경색입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.
- `border`는 경계의 너비, 투명도 및 색상을 지정하는 3자 문자열입니다.
- `color`는 오류 메시지 텍스트 색상입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.
- `box-sizing`은 크기 속성(너비 및 높이)에 포함시켜야 할 브라우저를 나타내는 데 사용됩니다.

자격 증명 공급자 버튼

- `height`는 픽셀(px) 단위의 버튼 높이입니다.
- `width`는 포함 블록의 비율로 표시되는 버튼 텍스트의 너비입니다.
- `text-align`은 텍스트 정렬 설정입니다. `left`, `right` 또는 `center`일 수 있습니다.
- `margin-bottom`은 아래쪽 마진 설정입니다.
- `color`는 버튼 텍스트 색상입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.
- `background-color`는 버튼의 배경색입니다. 표준 색상 값이면 무엇이든 가능합니다.
- `border-color`는 버튼 경계의 색상입니다. 표준 색상 값이면 무엇이든 가능합니다.

자격 증명 공급자 설명

- `padding-top`은 설명 위의 패딩 양입니다.
- `padding-bottom`은 설명 아래의 패딩 양입니다.
- `display`는 `block` 또는 `inline`일 수 있습니다.
- `font-size`는 설명의 글꼴 크기입니다.

법적 고지 텍스트

- `color`는 텍스트 색상입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.
- `font-size`는 글꼴 크기입니다.

로고

- `max-width`는 포함 블록의 비율로 표시되는 최대 너비입니다.
- `max-height`는 포함 블록의 비율로 표시되는 최대 높이입니다.

입력 필드 포커스

- `border-color`는 입력 필드의 색상입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.
- `outline`은 픽셀 단위의 입력 필드 경계 너비입니다.

소셜 버튼

- height는 픽셀(px) 단위의 버튼 높이입니다.
- text-align은 텍스트 정렬 설정입니다. left, right 또는 center일 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 지정 설정 지정(AWS Management 콘솔)

AWS Management 콘솔을 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_callback_url>`을 구성하고 이를 브라우저에 입력하면 사용자가 지정한 호스팅 UI를 볼 수 있습니다. 브라우저를 새로 고침하고 콘솔의 변경 내용이 나타나기까지 최대 1분 정도 기다려야 할 수도 있습니다. 도메인 이름 탭에 도메인이 나타납니다. 앱 클라이언트 설정 탭에 앱 클라이언트 ID와 콜백 URL이 나타납니다.

앱 UI 사용자 지정 설정 지정 방법

1. [Amazon Cognito 콘솔](#)에 로그인합니다.
2. 탐색 창에서 사용자 풀 관리를 선택한 다음 편집할 사용자 풀을 선택합니다.
3. UI 사용자 지정 탭을 선택합니다.
4. 사용자 지정할 앱 클라이언트로 가서 이전에 앱 클라이언트 탭에서 생성한 앱 클라이언트의 드롭다운 메뉴에서 사용자 지정을 원하는 앱을 선택합니다.
5. 자체의 로고 이미지 파일을 업로드하려면 파일 선택을 선택하거나 로고(선택 사항) 상자로 파일을 끌어옵니다.
6. CSS 사용자 지정(선택 사항)으로 가서 기본값에서 다양한 속성을 변경하여 앱의 모양을 사용자 지정할 수 있습니다.

사용자 풀에 대한 앱 UI 사용자 지정 설정 지정(AWS CLI 및 AWS API)

다음 명령을 사용하여 사용자 풀에 대한 앱 UI 사용자 지정 설정을 지정합니다.

사용자 풀의 내장 앱 UI에 대해 사용자 지정을 설정하려면

- AWS CLI: `aws cognito-idp get-ui-customization`
- AWS API: [GetUICustomization](#)

사용자 풀의 내장 앱 UI에 대해 사용자 지정을 설정하려면

- AWS CLI: `aws cognito-idp set-ui-customization --user-pool-id <your-user-pool-id> --client-id <your-app-client-id> --image-file <path-to-logo-image-file> --css ".label-customizable{ color: <color>;}"`

- AWS API: [SetUICustomization](#)

사용자 풀의 리소스 서버 정의

사용자 풀에 대한 도메인이 구성되면 Amazon Cognito 서비스는 호스팅된 웹 UI를 자동으로 프로비저닝합니다. 이를 사용하여 앱에 가입 및 로그인 페이지를 추가할 수 있습니다. 자세한 내용은 [단계 2. 호스팅 웹 UI를 활성화하는 앱 추가 \(p. 14\)](#) 단원을 참조하십시오.

리소스 서버는 액세스 보호가 되는 리소스의 서버로서, 액세스 토큰을 가진 앱에서 인증된 요청을 처리합니다. 일반적으로 리소스 서버는 이러한 액세스 요청을 위해 [CRUD](#) API를 제공합니다. 이 API는 Amazon API Gateway에 호스팅하거나 AWS 외부에 호스팅할 수 있습니다. 앱은 API 호출 시 액세스 토큰을 리소스 서버로 전달합니다. 앱은 액세스 요청 시 토큰을 전달할 때 액세스 토큰을 사용자가 볼 수 없게 처리해야 합니다. 리소스 서버는 액세스 토큰을 검사하여 액세스 권한을 부여할지 여부를 결정합니다.

Note

리소스 서버는 토큰 내에서 클레임을 처리하기 앞서 액세스 토큰 서명과 만료 날짜를 확인해야 합니다. 사용자 풀 토큰의 확인 및 사용에 대한 자세한 내용은 [이 블로그 게시물](#)을 참조하십시오. Amazon API Gateway는 액세스 토큰을 검사하여 리소스를 보호하기에 효과적인 옵션입니다. API 게이트웨이 사용자 지정 권한 부여자에 대한 자세한 내용은 [API 게이트웨이 사용자 지정 권한 부여자 사용](#)을 참조하십시오.

범위는 앱이 리소스에 요청할 수 있는 액세스의 수준입니다. 예를 들어 사진에 대한 리소스 서버를 가지고 있는 경우에는 두 개의 범위를 정의하게 되는데, 하나는 사진에 대한 읽기 액세스를 위한 범위이고 다른 하나는 쓰기/삭제 액세스를 위한 범위입니다. 앱이 액세스 요청을 위해 API 호출을 하고 액세스 토큰을 전달하면 토큰 내부에 하나 이상의 범위가 기본적으로 제공됩니다.

개요

Amazon Cognito는 앱 개발자가 자체 OAuth2.0 리소스 서버를 만들고 여기에서 사용자 지정 범위를 정의할 수 있도록 해줍니다. 사용자 지정 범위는 클라이언트에 연결될 수 있고, 클라이언트는 OAuth2.0 권한 부여 코드 부여 흐름, 암시적 흐름 및 클라이언트 자격 증명 흐름에서 이를 요청할 수 있습니다. 액세스 토큰의 scope 클레임에 사용자 지정 범위가 추가됩니다. 클라이언트는 리소스 서버에서 액세스 토큰을 사용하여 토큰에 포함된 범위를 토대로 권한 부여를 결정할 수 있습니다. 액세스 토큰 범위에 대한 자세한 내용은 [사용자 풀을 통해 토큰 사용 \(p. 190\)](#)을 참조하십시오.

Note

리소스 서버는 토큰 내에서 클레임을 처리하기 앞서 액세스 토큰 서명과 만료 날짜를 확인해야 합니다.

Note

앱 클라이언트는 클라이언트 암호를 가지고 있는 경우에만 클라이언트 자격 증명을 사용할 수 있습니다.

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.

예:

- Name: Weather API 또는 Photo API 같은 친숙한 리소스 서버 이름입니다.
- Identifier: 리소스 서버의 고유 식별자입니다. 리소스 서버가 위치한 HTTPS 엔드포인트가 될 수 있습니다. 예, <https://my-weather-api.example.com>
- Scope Name: 범위 이름입니다. 예, `weather.read`

- Scope Description: 범위에 대한 간략한 설명입니다. 예: Retrieve weather information.

클라이언트 앱이 OAuth2.0 흐름에서 사용자 지정 범위를 요청할 때는 범위 `resourceServerIdentifier/scopeName`에 대한 완전한 식별자를 요청해야 합니다. 예를 들어 리소스 서버 식별자가 `https://myphotosapi.example.com`이고 범위 이름이 `photos.read`이면 클라이언트 앱은 런타임 시 `https://myphotosapi.example.com/photos.read`를 요청해야 합니다.

리소스 서버에서 범위를 삭제해도 모든 클라이언트와의 연결이 삭제되지는 않습니다. 범위를 삭제하면 범위가 비활성화됩니다. 따라서 클라이언트 앱이 런타임 시 삭제된 범위를 요청하면 해당 범위가 무시되고 액세스 토큰에 포함되지 않습니다. 나중에 해당 범위를 다시 추가하면 액세스 토큰에 범위가 다시 포함됩니다.

클라이언트에서 범위가 삭제된 경우에는 클라이언트와 범위 간의 연결이 삭제됩니다. 클라이언트가 런타임 시 허용되지 않은 범위를 요청하면 오류가 발생하고 액세스 토큰이 발급되지 않습니다.

AWS Management 콘솔, API 및 CLI를 사용하여 사용자 풀의 리소스 서버와 범위를 정의할 수 있습니다.

사용자 풀의 리소스 서버 정의(AWS Management 콘솔)

AWS Management 콘솔을 사용하여 사용자 풀의 리소스 서버를 정의할 수 있습니다.

리소스 서버를 정의하려면

1. [Amazon Cognito 콘솔](#)에 로그인합니다.
2. 탐색 창에서 사용자 풀 관리를 선택한 다음 편집할 사용자 풀을 선택합니다.
3. 리소스 서버 탭을 선택합니다.
4. 리소스 서버 추가를 선택합니다.
5. 예를 들어 Photo Server와 같이 리소스 서버의 이름을 입력합니다.
6. 예를 들어 com.example.photos와 같이 리소스 서버의 ID를 입력합니다.
7. read 및 write와 같이 리소스에 대한 사용자 지정 범위의 이름을 입력합니다.
8. 범위 이름 각각에 대해 view your photos 및 update your photos와 같이 설명을 입력합니다.
9. [Save changes]를 선택합니다.

정의한 각각의 사용자 지정 범위가 OAuth2.0 허용된 사용자 지정 범위 아래 앱 클라이언트 설정 탭에 나타납니다(예: com.example.photos/read).

사용자 풀의 리소스 서버 정의(AWS CLI 및 AWS API)

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

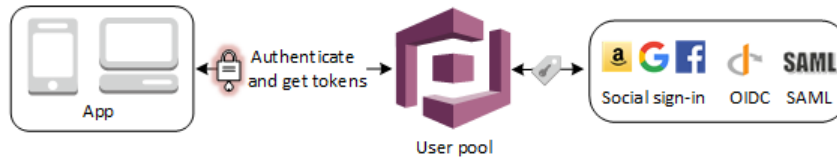
- AWS CLI: `aws cognito-idp delete-resource-server`
- AWS API: [DeleteResourceServer](#)

리소스 서버의 설정을 업데이트하려면

- AWS CLI: `aws cognito-idp update-resource-server`
- AWS API: [UpdateResourceServer](#)

타사를 통한 사용자 풀 로그인 추가

앱 사용자는 사용자 풀을 통해 직접 로그인하거나 타사 자격 증명 공급자(IdP)를 통해 연동 로그인할 수 있습니다. 사용자 풀에서는 Facebook, Google 및 Amazon을 통한 소셜 로그인에서 반환된 토큰과 OpenID Connect(OIDC) 및 SAML IdP에서 반환된 토큰의 처리 작업을 관리합니다. 내장된 호스트 웹 UI를 사용하여 Amazon Cognito가 모든 자격 증명 공급자의 인증 사용자에게 대한 토큰 처리 및 관리를 제공하기 때문에 사용자의 백엔드 시스템을 한 세트의 사용자 풀 토큰에서 표준화할 수 있습니다.



Note

타사(연동)를 통한 로그인을 Amazon Cognito 사용자 풀에서 사용할 수 있습니다. 이 기능은 자격 증명 풀(연동 자격 증명)을 통한 연동과 무관합니다.

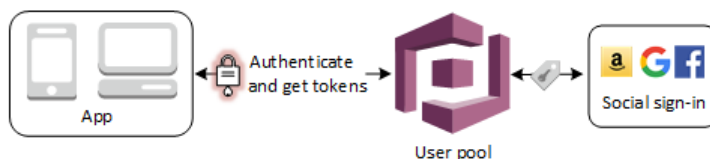
주제

- 사용자 풀에 소셜 자격 증명 공급자 추가 (p. 88)
- 사용자 풀에 SAML 자격 증명 공급자 추가 (p. 92)
- 사용자 풀에 OIDC 자격 증명 공급자 추가 (p. 99)
- 사용자 풀에 대한 자격 증명 공급자 속성 매핑 지정 (p. 104)

사용자 풀에 소셜 자격 증명 공급자 추가

웹 및 모바일 앱 사용자들이 Facebook, Google, Amazon 같은 소셜 자격 증명 공급자(IdP)를 통해 로그인할 수 있습니다. 내장된 호스트 웹 UI를 사용하여 Amazon Cognito가 모든 자격 증명 공급자의 모든 인증 사용자에게 대한 토큰 처리 및 관리를 제공하기 때문에 사용자의 백엔드 시스템을 한 세트의 사용자 풀 토큰에서 표준화할 수 있습니다.

AWS CLI 또는 Amazon Cognito API 호출을 사용하여 AWS Management 콘솔에서 소셜 자격 증명 공급자를 추가할 수 있습니다.



Note

타사(연동)를 통한 로그인을 Amazon Cognito 사용자 풀에서 사용할 수 있습니다. 이 기능은 자격 증명 풀(연동 자격 증명)을 통한 연동과 무관합니다.

주제

- [사전 조건](#) (p. 89)
- [1단계: 소셜 IdP로 등록](#) (p. 89)
- [2단계: 사용자 풀에 소셜 IdP 추가](#) (p. 91)
- [3단계: 소셜 IdP 구성 테스트](#) (p. 92)

사전 조건

시작하려면 다음이 필요합니다.

- 애플리케이션 클라이언트 및 사용자 풀 도메인이 있는 사용자 풀. 자세한 내용은 [사용자 풀 생성](#) 단원을 참조하십시오.
- 소셜 자격 증명 공급자.

1단계: 소셜 IdP로 등록

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

Facebook으로 앱을 등록하려면

1. Facebook에서 [개발자 계정을 생성](#)합니다.
2. Facebook 자격 증명으로 [로그인](#)합니다.
3. My Apps(내 앱) 메뉴에서 Create New App(새 앱 생성)을 선택합니다.
4. Facebook 앱 이름을 지정한 다음 Create App ID(앱 ID 생성)를 선택합니다.
5. 왼쪽 탐색 모음에서 설정과 기본을 차례로 선택합니다.
6. App ID(앱 ID)와 앱 보안을 메모합니다. 다음 섹션에서 이 둘을 사용합니다.
7. 페이지 하단에서 + Add Platform(플랫폼 추가)을 선택합니다.
8. 웹 사이트를 선택합니다.
9. 웹 사이트에서 엔드포인트가 /oauth2/idpresponse인 사용자 풀 도메인을 Site URL(사이트 URL)에 입력합니다.

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

10. [Save changes]를 선택합니다.
11. App Domains(앱 도메인)에 사용자 풀 도메인을 입력합니다.

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

12. [Save changes]를 선택합니다.
13. 탐색 모음에서 제품을 선택하고 Facebook 로그인에서 Set up(설정)을 선택합니다.
14. 탐색 모음에서 Facebook 로그인과 설정을 차례로 선택합니다.

Valid OAuth Redirect URIs(유효 OAuth 리디렉션 URI)에 리디렉션 URL을 입력합니다. 엔드포인트가 /oauth2/idpresponse인 사용자 풀 도메인으로 이루어집니다.

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

15. [Save changes]를 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 Amazon에 로그인을 선택합니다.
4. Create a Security Profile(보안 프로필 생성)을 선택합니다.
5. Security Profile Name(보안 프로필 이름), Security Profile Description(보안 프로필 설명) 및 Consent Privacy Notice URL(개인 정보 보호 정책 URL 동의)에 입력합니다.
6. Save를 선택합니다.
7. 클라이언트 ID 및 클라이언트 암호를 선택하여 클라이언트 ID 및 암호를 표시합니다. 다음 섹션에서 이 둘을 사용합니다.
8. 기어를 마우스로 가리켜 Web Settings(웹 설정) 탭을 선택하고 편집을 선택합니다.
9. Allowed Origins(허용되는 오리진)에 사용자 풀 도메인을 입력합니다.

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

10. 엔드포인트가 /oauth2/idpresponse인 사용자 풀 도메인을 Allowed Return URLs(허용되는 반환 URL)에 입력합니다.

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

11. Save를 선택합니다.

Google로 앱을 등록하려면

1. Google에서 [개발자 계정을 생성합니다](#).
2. Google 자격 증명으로 [로그인합니다](#).
3. CONFIGURE A PROJECT(프로젝트 구성)를 선택합니다.
4. 프로젝트 이름을 입력하고 다음을 선택합니다.
5. 사용자의 프로젝트 이름을 입력하고 다음을 선택합니다.
6. Where are you calling from?(호출 위치는 어디입니까?) 드롭다운 목록에서 Web browser(웹 브라우저)를 선택합니다.
7. Authorized JavaScript origins(권한 있는 JavaScript 오리진)에 사용자 풀 도메인을 입력합니다.

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

8. CREATE를 선택합니다. 이 단계에서는 클라이언트 ID와 클라이언트 보안을 사용하지 않습니다.

9. DONE을 선택합니다.
10. Google 콘솔에 [로그인합니다](#).
11. 왼쪽 탐색 모음에서 Credentials(자격 증명)를 선택합니다.
12. 자격 증명 생성 드롭다운 목록에서 OAuth client ID(OAuth 클라이언트 ID)를 선택하여 OAuth 2.0 자격 증명을 생성합니다.
13. Web application(웹 애플리케이션)을 선택합니다.
14. Authorized JavaScript origins(권한 있는 JavaScript 오리진)에 사용자 풀 도메인을 입력합니다.

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

15. /oauth2/idpresponse 엔드포인트가 있는 사용자 풀 도메인을 Authorized Redirect URIs(인증 리디렉션 URI)에 입력합니다.

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

16. Create(생성)를 두 번 선택합니다.
17. OAuth 클라이언트 ID와 클라이언트 암호를 메모합니다. 다음 섹션에서 이 둘이 필요합니다.
18. 확인을 선택합니다.

2단계: 사용자 풀에 소셜 IdP 추가

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

AWS Management 콘솔을 사용하여 사용자 풀 소셜 자격 증명 공급자를 구성하려면

1. [Amazon Cognito 콘솔](#)로 이동합니다. AWS 자격 증명을 입력하라는 메시지가 나타날 수 있습니다.
2. [Manage your User Pools]를 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 [사용자 풀을 생성합니다](#).
4. 왼쪽 탐색 모음에서 자격 증명 공급자를 선택합니다.
5. Facebook, Google, Login with Amazon 등 소셜 자격 증명 공급자를 선택합니다.
6. 이전 섹션의 소셜 자격 증명 공급자에게서 받은 앱 클라이언트 ID와 앱 클라이언트 암호를 입력합니다.
7. 승인하려는 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.

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

앱 사용자는 앱에 이러한 속성들을 제공한다는 데 동의하라는 요청 메시지를 받게 됩니다. 범위에 대한 자세한 내용은 Google, Facebook 및 Login with Amazon의 설명서를 참조하십시오.

8. 구성 중인 소셜 자격 증명 공급자에 대해 Enable(활성화)을 선택합니다.
9. 탐색 모음에서 앱 클라이언트 설정을 선택합니다.

10. 사용자 풀 앱의 활성화된 자격 증명 공급자 중 하나로 소셜 자격 증명 공급자를 선택합니다.
11. 사용자 풀 앱의 콜백 URL(여러 개 가능)에 콜백 URL을 입력합니다. 이것은 인증 성공 이후에 사용자가 리디렉션되는 페이지의 URL입니다.

```
https://www.example.com
```

12. [Save changes]를 선택합니다.
13. 속성 매핑 탭에서 다음과 같이 최소한의 필수 속성(일반적으로 email)에 대한 매핑을 추가합니다.
 - a. 확인란을 선택하여 Facebook, Google 또는 Amazon 속성 이름을 선택합니다. Amazon Cognito 콘솔에 올라와 있지 않은 추가 속성의 이름을 입력할 수도 있습니다.
 - b. 드롭다운 목록에서 대상 사용자 풀 속성을 선택합니다.
 - c. [Save changes]를 선택합니다.
 - d. 요약본으로 이동을 선택합니다.

3단계: 소셜 IdP 구성 테스트

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

```
https://<your_user_pool_domain>/login?
response_type=code&client_id=<your_client_id>&redirect_uri=https://www.example.com
```

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

사용자 풀에 SAML 자격 증명 공급자 추가

Microsoft Active Directory Federation Services (ADFS) 또는 Shibboleth 같은 SAML 자격 증명 공급자(IdP)를 통해 웹 및 모바일 앱 사용자가 로그인하게 할 수 있습니다. SAML 2.0 표준을 지원하는 SAML 자격 증명 공급자를 선택합니다.

내장된 호스트 웹 UI를 사용하여 Amazon Cognito가 모든 자격 증명 공급자의 모든 인증 사용자에게 대한 토큰 처리 및 관리를 제공하기 때문에 사용자의 백엔드 시스템을 한 세트의 사용자 풀 토큰에서 표준화할 수 있습니다. AWS Management 콘솔에서 AWS CLI 또는 Amazon Cognito API 호출을 사용하여 SAML IdP를 생성하고 관리할 수 있습니다. 콘솔로 시작하려면 [AWS Management 콘솔을 사용하여 SAML 기반 자격 증명 공급자를 통해 사용자 풀에 로그인 추가 단원을 참조하십시오](#).



Note

타사(연동)를 통한 로그인을 Amazon Cognito 사용자 풀에서 사용할 수 있습니다. 이 기능은 자격 증명 풀(연동 자격 증명)을 통한 연동과 무관합니다.

SAML 자격 증명 공급자를 업데이트하고, 이를 지원하도록 사용자 풀을 구성해야 합니다. SAML 2.0 자격 증명 공급자에 대한 신뢰 당사자 또는 애플리케이션으로서 사용자 풀을 추가하는 방법에 대한 내용은 SAML 자격 증명 공급자 설명서를 참조하십시오.

그리고 SAML 자격 증명 공급자에 어설션 소비자 엔드포인트를 제공해야 합니다. SAML 2.0 POST에 대해 이 엔드포인트를 구성하여 SAML 자격 증명 공급자를 바인딩합니다.

```
https://<yourDomainPrefix>.auth.<region>.amazoncognito.com/saml2/idpresponse
```

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

일부 SAML 자격 증명 공급자의 경우 SP urn /대상 URI / SP 개체 ID를 다음 양식으로 제공해야 합니다.

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

Amazon Cognito 콘솔의 앱 클라이언트 설정 탭에서 사용자 풀 ID를 찾을 수 있습니다.

사용자 풀에 필요한 속성에 대한 속성 값을 제공하려면 SAML 자격 증명 공급자도 구성해야 합니다. 일반적으로 email은 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 연동을 지원합니다. 사용자 풀이 사용자 에이전트를 통해 자격 증명 공급자로부터 직접 SAML 응답을 받으므로 앱에서 SAML 어설션 응답을 수신하거나 구문 분석을 할 필요가 없습니다. 사용자 풀은 애플리케이션을 대신하여 SP(서비스 공급자) 역할을 합니다. Amazon Cognito는 [SAML V2.0 Technical Overview](#)의 단원 5.1.2에 설명된 SP-initiated SSO(Single Sign-On)를 지원합니다.

주제

- [SAML 사용자 풀 IdP 인증 흐름 \(p. 93\)](#)
- [SAML 자격 증명 공급자 이름 선택 \(p. 94\)](#)
- [사용자 풀에 대한 SAML 자격 증명 공급자 생성 및 관리\(AWS Management 콘솔\) \(p. 95\)](#)
- [사용자 풀에 대한 SAML 자격 증명 공급자 생성 및 관리\(AWS CLI 및 AWS API\) \(p. 96\)](#)
- [타사 SAML 자격 증명 공급자와 Amazon Cognito 사용자 풀 통합 \(p. 98\)](#)

SAML 사용자 풀 IdP 인증 흐름

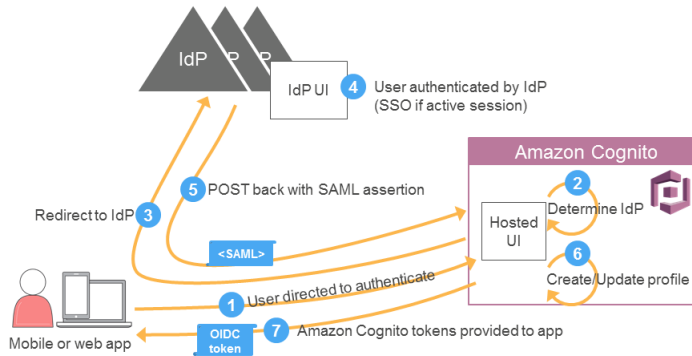
사용자 풀의 SAML 기반 IdP를 직접 통합할 수 있습니다.

1. 앱이 AWS에서 호스팅하는 UI로 사용자를 보내 가입 및 로그인 프로세스를 시작합니다. 모바일 앱이 웹 보기를 사용하여 에서 호스팅하는 페이지를 표시할 수 있습니다.
2. 일반적으로 사용자 풀은 사용자의 이메일 주소에서 사용자의 자격 증명 공급자를 결정합니다.

그 대신 사용자를 사용자 풀로 전달하기 전에 앱에서 정보를 수집한 경우 쿼리 파라미터를 통해 해당 정보를 Amazon Cognito에 제공합니다.
3. 사용자가 자격 증명 공급자로 리디렉션됩니다.
4. 필요한 경우 IdP가 사용자를 인증합니다. 사용자에게 활성 세션이 있음을 인식하면 IdP가 인증을 건너뛰어 SSO(Single Sign-On) 환경을 제공합니다.
5. IdP가 SAML 어설션을 Amazon Cognito 서비스에 게시합니다.
6. AWS Management 콘솔에서 AWS CLI 또는 Amazon Cognito API 호출을 사용하여 사용자의 프로파일을 생성 또는 업데이트할 수 있습니다.
7. SAML 어설션을 확인하고 어설션에서 사용자 속성(클레임)을 수집한 후 Amazon Cognito가 지금 로그인한 사용자를 위해 앱에 OIDC 토큰을 반환합니다.

다음 다이어그램은 이 프로세스의 인증 흐름을 보여줍니다.

SAML Federation



사용자가 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. ID 및 액세스 토큰은 한 시간 후에 만료되지만 앱이 새로 고침 토큰을 사용하여 사용자를 다시 인증하지 않고 새 토큰을 가져올 수 있습니다. 개발자는 새로 고침 토큰의 만료 시간과 사용자가 인증해야 하는 빈도를 선택할 수 있습니다. 사용자가 외부 IdP를 통해 인증한 경우(즉, 연동된 사용자) 앱이 계속 토큰을 새로 고침 토큰과 함께 사용하여 외부 IdP의 토큰이 만료되는 것과 관계없이 사용자가 재인증할 때까지의 기간을 결정합니다. ID 및 액세스 토큰이 만료되면 사용자 풀이 자동으로 새로 고침 토큰을 사용하여 새 ID 및 액세스 토큰을 가져옵니다. 새로 고침 토큰도 만료된 경우 서버가 에서 호스팅하는 앱의 페이지를 통해 인증을 자동으로 시작합니다.

SAML 자격 증명 공급자 이름 선택

SAML 공급자의 이름을 선택해야 합니다. 문자열 형식은 [w@s+.=.-]+이며 최대 40자일 수 있습니다.

선택적으로 SAML 공급자의 식별자를 선택할 수 있습니다. 식별자는 사용자 풀과 연결된 자격 증명 공급자를 고유하게 확인합니다. 일반적으로 각 식별자는 SAML IdP가 나타내는 회사에 속한 도메인에 해당합니다. 여러 회사에서 사용할 수 있는 다중 테넌트 앱의 경우 사용자를 정확한 IdP에 리디렉션하는 데 식별자를 사용할 수 있습니다. 같은 회사가 여러 도메인을 소유할 수 있으므로 여러 식별자를 제공할 수 있습니다.

각 SAML 공급자에 식별자를 50개까지 연결할 수 있습니다. 식별자는 자격 증명 공급자에서 고유해야 합니다.

예를 들어, 서로 다른 두 회사(A와 B)의 직원이 사용할 수 있는 앱을 빌드한 경우 A 회사는 domainA.com 및 domainA.co.uk를 소유하고 B 회사는 domainB.com을 소유합니다. 이때 회사마다 하나씩 IdP 2개를 설정했다고 가정해 보겠습니다.

- IdP A에 식별자 DomainA.com 및 DomainA.co.uk를 정의할 수 있습니다.
- IdP B에 식별자 DomainB.com을 정의할 수 있습니다.

애플리케이션에서 이메일 주소를 입력하라는 메시지를 사용자에게 표시할 수 있습니다. 이메일 주소에서 도메인을 파생시키면 IdPIdentifier 엔드포인트에 대한 호출의 /authorize에 도메인을 제공하여 사용자를 정확한 IdP에 리디렉션할 수 있습니다. 예를 들어, 사용자가 bob@domain1.co.uk를 입력하면 사용자가 IdP A로 리디렉션됩니다.

Amazon Cognito에서 호스팅하는 로그인 페이지는 이메일 주소를 자동으로 구문 분석하여 정보를 도출합니다. 그리고 이메일에서 이메일 도메인을 구문 분석하여 이를 IdPIdentifier 엔드포인트를 호출할 때 /authorize로 사용합니다.

- SAML IdP가 여러 개이고 이들 중 하나에 대해 IdPIdentifier 값을 지정한 경우에는 호스팅된 페이지에 이메일 주소를 입력하라는 상자가 나타납니다.
- SAML IdP가 여러 개이고 이들에 대해 IdPIdentifier 값을 지정하지 않은 경우에는 호스팅된 페이지에 IdP 목록이 나타납니다.

자체 UI를 빌드할 경우 도메인 이름을 구문 분석하여 해당 이름이 IdP 설정 중 제공된 `IdPIdentifiers`와 일치하도록 해야 합니다. IdP 설정에 대한 자세한 내용은 [사용자 풀에 자격 증명 공급자 구성 \(p. 218\)](#)을 참조하십시오.

사용자 풀에 대한 SAML 자격 증명 공급자 생성 및 관리(AWS Management 콘솔)

AWS Management 콘솔 콘솔을 사용하여 SAML 자격 증명 공급자를 만들고 삭제할 수 있습니다.

SAML 자격 증명 공급자를 생성하기 전에 타사 IdP(자격 증명 공급자)에서 가져온 SAML 메타데이터 문서가 필요합니다. 필요한 SAML 메타데이터 문서를 가져오거나 생성하는 방법에 대한 지침은 [타사 SAML 자격 증명 공급자와 Amazon Cognito 사용자 풀 통합 \(p. 98\)](#)을 참조하십시오.

사용자 풀에 SAML 2.0 자격 증명 공급자를 구성하려면

1. [Amazon Cognito 콘솔](#)로 이동합니다. AWS 자격 증명을 입력하라는 메시지가 나타날 수 있습니다.
2. [Manage your User Pools]를 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 [사용자 풀을 생성합니다](#).
4. 왼쪽 탐색 모음에서 자격 증명 공급자를 선택합니다.
5. SAML을 선택하여 SAML 대화상자를 엽니다.
6. 메타데이터 문서에서 사용자 SAML IdP의 메타데이터 문서를 업로드합니다. 메타데이터 문서를 가리키는 URL을 입력할 수도 있습니다. 자세한 내용은 [타사 SAML 자격 증명 공급자와 Amazon Cognito 사용자 풀 통합 \(p. 98\)](#) 단원을 참조하십시오.

Note

Amazon Cognito에서 메타데이터를 자동으로 새로 고칠 수 있으므로 퍼블릭 엔드포인트인 경우 파일을 업로드하지 않고 엔드포인트 URL을 제공하는 것이 좋습니다. 일반적으로 메타데이터 새로 고침은 6시간마다 또는 메타데이터가 만료되기 전 중 더 빠른 시간에 발생합니다.

7. 사용자의 SAML 공급자 이름을 입력합니다. SAML 명명에 관한 자세한 내용은 [SAML 자격 증명 공급자 이름 선택 \(p. 94\)](#) 섹션을 참조하십시오.
8. 사용할 SAML 식별자(선택사항)를 입력합니다.
9. Amazon Cognito에서 로그아웃할 때 SAML IdP에서 사용자를 로그아웃되도록 하려면 Enable IdP sign out flow(IdP 로그아웃 흐름 활성화)를 선택합니다.

이 흐름을 활성화하면 [로그아웃 엔드포인트 \(p. 326\)](#)가 호출될 때 서명된 로그아웃 요청이 SAML IdP로 전송됩니다.

IdP의 로그아웃 응답을 사용하도록 이 엔드포인트를 구성합니다. 이 엔드포인트는 사후 바인딩을 사용합니다.

```
https://<yourDomainPrefix>.auth.<region>.amazoncognito.com/saml2/logout
```

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.

10. 공급자 생성을 선택합니다.
11. 속성 매핑 탭에서 다음과 같이 최소한의 필수 속성(일반적으로 email)에 대한 매핑을 추가합니다.
 - a. 자격 증명 공급자의 SAML 어설션에 표시된 대로 SAML 속성 이름을 입력합니다. 자격 증명 공급자가 예제 SAML 어설션을 제공하는 경우 이름을 찾는 데 도움이 될 수 있습니다. 일부 자격 증명 공급

자는 email과 같은 간단한 이름을 사용하지만 다른 자격 증명 공급자는 다음과 비슷한 이름을 사용합니다.

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

- b. 드롭다운 목록에서 대상 사용자 풀 속성을 선택합니다.
12. [Save changes]를 선택합니다.
13. 요약본으로 이동을 선택합니다.

Note

HTTPS 메타데이터 엔드포인트 URL을 통해 SAML 자격 증명 공급자를 생성하는 동안 "<metadata endpoint>에서 메타데이터를 검색하는 동안 오류 발생"과 같은 `InvalidParameterException`이 표시되면 메타데이터 엔드포인트에 SSL이 올바르게 설정되어 있는지 그리고 이와 연관된 유효한 SSL 인증서가 있는지를 확인하십시오.

사용자 풀을 신뢰 당사자로 추가하도록 SAML IdP를 설정하려면

- 사용자 풀 서비스 공급자 URN은 `urn:amazon:cognito:sp:<user_pool_id>`입니다. 대상 제한이 있는 SAML 어설션을 이 URN에 발급하기 위해 Amazon Cognito는 `AuthnRequest`를 SAML IdP에 발급합니다. IdP는 IdP에서 SP로 보내는 응답 메시지에 대해 다음 POST 바인딩 엔드포인트를 사용합니다.
`https://<domain_prefix>.auth.<region>.amazoncognito.com/saml2/idpresponse`
- SAML IdP가 SAML 어설션의 사용자 풀에 대한 필수 속성과 NameID를 채우는지 확인합니다. NameID는 사용자 풀에서 SAML 연동 사용자를 고유하게 식별하는 데 사용됩니다. 지속 SAML 이름 ID 형식을 사용합니다.

서명 인증서를 추가하도록 SAML IdP를 설정하려면

- 서명된 로그인 요청을 확인하기 위해 자격 증명 공급자가 사용할 퍼블릭 키를 포함하는 인증서를 가져오려면 연동 콘솔 페이지의 자격 증명 공급자에서 SAML 대화 상자의 Active SAML Providers(활성 SAML 공급자)에 있는 Show signing certificate(서명 인증서 표시)를 선택합니다.

SAML 공급자를 삭제하려면

1. [Amazon Cognito 콘솔](#)에 로그인합니다.
2. 탐색 창에서 사용자 풀 관리를 선택한 다음 편집할 사용자 풀을 선택합니다.
3. 연동 콘솔 페이지에서 자격 증명 공급자를 선택합니다.
4. SAML을 선택하여 SAML 자격 증명 공급자를 표시합니다.
5. 삭제할 공급자 옆에 있는 확인란을 선택합니다.
6. Delete provider(공급자 삭제)를 선택합니다.

사용자 풀에 대한 SAML 자격 증명 공급자 생성 및 관리(AWS CLI 및 AWS API)

다음 명령을 사용해 SAML 공급자를 생성 및 관리합니다.

자격 증명 공급자를 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 <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.json에 다음 사항이 포함됩니다.

```
{
  "MetadataFile": "<SAML metadata XML>"
}
```

Note

<SAML metadata XML>에 따옴표(")가 있으면 이스케이프해야 합니다(\").

메타데이터 URL이 포함된 예제: `aws cognito-idp create-identity-provider --user-pool-id <user_pool_id> --provider-name=SAML_provider_1 --provider-type SAML --provider-details MetadataURL=<metadata_url> --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

- AWS API: [CreateIdentityProvider](#)

자격 증명 공급자의 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 <user_pool_id> --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.json에 다음 사항이 포함됩니다.

```
{
  "MetadataFile": "<SAML metadata XML>"
}
```

Note

<SAML metadata XML>에 따옴표(")가 있으면 이스케이프해야 합니다(\").

메타데이터 URL이 포함된 예제: `aws cognito-idp update-identity-provider --user-pool-id <user_pool_id> --provider-name=SAML_provider_1 --provider-details MetadataURL=<metadata_url> --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

- AWS API: [UpdateIdentityProvider](#)

특정 자격 증명 공급자에 대한 정보를 가져오려면

- AWS CLI: `aws cognito-idp describe-identity-provider`

```
aws cognito-idp describe-identity-provider --user-pool-id <user_pool_id> --provider-name=SAML_provider_1
```

- AWS API: [DescribeIdentityProvider](#)

모든 자격 증명 공급자에 대한 정보를 나열하려면

- AWS CLI: `aws cognito-idp list-identity-providers`


```
예: aws cognito-idp list-identity-providers --user-pool-id <user_pool_id> --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 <user_pool_id> --provider-name=SAML_provider_1
```

- AWS API: [DeleteIdentityProvider](#)

타사 SAML 자격 증명 공급자와 Amazon Cognito 사용자 풀 통합

Amazon Cognito 사용자 풀에서 자격 증명 연동을 사용할 수 있도록 타사 SAML 2.0 자격 증명 공급자 솔루션을 구성하려면 `https://<yourDomainPrefix>.auth.<region>.amazoncognito.com/saml2/idpresponse`라는 리디렉션 또는 로그인 URL을 입력해야 합니다. [Amazon Cognito 콘솔](#)의 도메인 이름 콘솔 페이지에서 사용자 풀의 도메인 접두사와 리전 값을 찾을 수 있습니다.

Note

2017년 8월 10일 이전의 공개 베타 버전을 사용하는 중에 사용자 풀에서 생성한 모든 SAML 자격 증명 공급자는 `https://<yourDomainPrefix>.auth.<region>.amazoncognito.com/login/redirect`라는 리디렉션 URL을 갖습니다. 사용자 풀에서 생성된 이러한 SAML 자격 증명 공급자 중 하나가 공개 베타 버전에서 나온 것이면 다음 조치 중 하나를 취해야 합니다.

- 신규 리디렉션 URL을 사용하는 SAML 자격 증명 공급자로 바꿉니다.
- 기존 및 신규 리디렉션 URL을 모두 수락하도록 SAML 자격 증명 공급자의 구성을 업데이트합니다.

Amazon Cognito의 모든 SAML 자격 증명 공급자가 새로운 URL로 전환되고 기존 URL은 2017년 10월 31일자로 사용이 중단됩니다.

일부 SAML 자격 증명 공급자의 경우 urn / 대상 URI / SP 개체 ID를 `urn:amazon:cognito:sp:<yourUserPoolID>` 양식으로 제공해야 합니다. Amazon Cognito 콘솔의 일반 설정 탭에서 사용자 풀 ID를 찾을 수 있습니다.

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

다음 링크를 참고하여 Amazon Cognito 사용자 풀에 대한 연동 작업을 수행하도록 타사 SAML 2.0 자격 증명 공급자 솔루션을 구성할 수 있습니다.

Note

자격 증명 공급자 지원은 Amazon Cognito에 기본적으로 제공되므로 다음 공급자 사이트로 이동하여 SAML 메타데이터 문서를 가져오면 됩니다. 공급자 웹 사이트에서 통합에 대한 자세한 지침을 확인할 수 있지만 꼭 필요한 것은 아닙니다.

솔루션	추가 정보
Microsoft AD FS(Active Directory Federation Services)	다음 주소에서 ADFS 연동 서버에 대한 SAML 메타데이터 문서를 다운로드할 수 있습니다. <code>https://<yourservername>/</code>

솔루션	추가 정보
	FederationMetadata/2007-06/ FederationMetadata.xml
Okta	Okta에서 애플리케이션으로 Amazon Cognito 사용자 풀을 구성한 경우 Okta 대시보드의 Admin 섹션에서 메타데이터 문서를 찾을 수 있습니다. 애플리케이션을 선택하고 Sign On(로그인) 섹션을 선택한 다음 Settings for SAML(SAML 설정) 아래를 확인합니다. URL은 <code>https://<app-domain>.oktapreview.com/app/<application-ID>/sso/saml/metadata</code> 와 같아야 합니다.
Auth0	Auth0 대시보드에서 메타데이터 다운로드 문서를 가져옵니다. Clients(클라이언트)를 선택한 다음 Settings(설정)를 선택합니다. 아래로 스크롤하고 Show Advanced Settings(고급 설정 표시)를 선택한 다음 SAML Metadata URL(SAML 메타데이터 URL)을 찾습니다. URL은 <code>https://<your-domain-prefix>.auth0.com/samlp/metadata/<your-Auth0-client-ID></code> 와 같아야 합니다.
Ping Identity	PingFederate의 경우 Provide general SAML metadata by file 에서 메타데이터 XML 파일 다운로드 지침을 확인할 수 있습니다.

사용자 풀에 OIDC 자격 증명 공급자 추가

OpenID Connect (OIDC) 자격 증명 공급자(IdP)(예: [Salesforce](#) 또는 [Ping Identity](#))의 계정이 있는 사용자들이 가입 단계를 건너뛰고 기존 계정을 사용하여 사용자의 애플리케이션에 로그인하게 할 수 있습니다. 내장된 호스트 웹 UI를 사용하여 Amazon Cognito가 모든 자격 증명 공급자의 모든 인증 사용자에게 대한 토큰 처리 및 관리를 제공하기 때문에 사용자의 백엔드 시스템을 한 세트의 사용자 풀 토큰에서 표준화할 수 있습니다.



Note

타사(연동)를 통한 로그인을 Amazon Cognito 사용자 풀에서 사용할 수 있습니다. 이 기능은 자격 증명 풀(연동 자격 증명)을 통한 연동과 무관합니다.

AWS Management Console에서 AWS CLI 또는 사용자 풀 API 메서드 [CreateIdentityProvider](#)를 사용하여 OIDC IdP를 사용자 풀에 추가할 수 있습니다.

주제

- [사전 조건](#) (p. 100)
- [1단계: OIDC IdP로 등록](#) (p. 100)
- [2단계: 사용자 풀에 OIDC IdP 추가](#) (p. 102)
- [3단계: OIDC IdP 구성 테스트](#) (p. 103)

- [OIDC 사용자 풀 IdP 인증 흐름 \(p. 104\)](#)

사전 조건

시작하려면 다음이 필요합니다.

- 애플리케이션 클라이언트 및 사용자 풀 도메인이 있는 사용자 풀. 자세한 내용은 [사용자 풀 생성](#) 단원을 참조하십시오.
- OIDC IdP.

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.

OIDC IdP 링크

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:// <i>Your Ping domain address</i> :9031/idp/userinfo.openid 예: https://pf.company.com:9031/idp/userinfo.openid
Okta	Okta 자격 증명 공급자 설치	https:// <i>Your Okta subdomain</i> .oktapreview.com 또는 https:// <i>Your Okta subdomain</i> .okta.com
Microsoft Azure Active Directory (Azure AD)	Microsoft Azure AD 자격 증명 공급자 설치	https://login.windows.net/common
Google	Google 자격 증명 공급자 설치	https://accounts.google.com Note Amazon Cognito는 통합 소셜 로그인 IdP로 Google을 제공합니다. 통합 IdP 사용을 권장합니다. 사용자 풀에 소셜 자격 증명 공급자 추가 (p. 88) 단원을 참조하십시오.

2. 엔드포인트가 /oauth2/idpresponse인 사용자 풀 도메인 URL을 OIDC IdP에 등록합니다. 그래야만 사용자를 인증할 때 나중에 Amazon Cognito에서 OIDC IdP를 수락합니다.

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

- OIDC IdP에 콜백 URL을 등록합니다. 이것은 인증 성공 이후에 사용자가 리디렉션되는 페이지의 URL입니다.

```
https://www.example.com
```

- 사용자의 **범위**를 선택합니다. 범위 openid는 필수 항목입니다. 이메일 범위가 있어야만 이메일 및 email_verified **클레임**에 대한 액세스 권한을 받을 수 있습니다.
- OIDC IdP에서 클라이언트 ID와 클라이언트 암호를 제공합니다. 사용자 풀에서 OIDC IdP를 설정할 때 이 둘을 사용하게 됩니다.

예: 사용자 풀에서 Salesforce를 OIDC IdP로 사용

OIDC 자격 증명 공급자는 Salesforce 같은 OIDC 호환 IdP와 사용자 풀 사이에 신뢰를 구축해야 할 때 사용합니다.

- Salesforce 개발자 웹사이트에서 **계정을 생성합니다**.
- 이전 단계에서 설정한 개발자 계정을 통해 로그인합니다**.
- Salesforce 페이지 상단을 살펴봅니다.
 - 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.
- 왼쪽 탐색 모음에서 Company Settings(회사 설정)를 선택합니다.
- 탐색 모음에서 도메인을 선택하고, 도메인을 입력한 후, 생성을 선택합니다.
- 왼쪽 탐색 모음에서 Platform Tools(플랫폼 도구)로 이동하여 Apps(앱)를 선택합니다.
- App Manager(앱 관리자)를 선택합니다.
- new connected app(새로 연결된 앱)을 선택합니다.
 - 필수 필드를 입력합니다.

Start URL(시작 URL)에서 엔드포인트가 /oauth2/idpresponse인 사용자 풀 도메인 URL을 입력합니다.

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

- OAuth settings(OAuth 설정)을 활성화하고 콜백 URL에 콜백 URL을 입력합니다. 이것은 로그인 성공 후 사용자가 리디렉션되는 페이지의 URL입니다.

```
https://www.example.com
```

- 사용자의 **범위**를 선택합니다. 범위 openid는 필수 항목입니다. 이메일 범위가 있어야만 이메일 및 email_verified **클레임**에 대한 액세스 권한을 받을 수 있습니다.

범위는 공백으로 구분됩니다.

- Create를 선택합니다.

Salesforce에서는 클라이언트 ID를 Consumer Key(사용자 키)로 부르고, 클라이언트 암호를 Consumer Secret(사용자 암호)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)

1. [Amazon Cognito 콘솔](#)로 이동합니다. AWS 자격 증명을 입력하라는 메시지가 나타날 수 있습니다.
2. [Manage your User Pools]를 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 [사용자 풀을 생성합니다](#).
4. 왼쪽 탐색 모음에서 자격 증명 공급자를 선택합니다.
5. OpenId Connect를 선택합니다.
6. Provider name(공급자 이름)에 고유한 이름을 입력합니다.
7. 이전 섹션의 OIDC IdP 클라이언트 ID를 클라이언트 ID에 입력합니다.
8. 이전 섹션의 클라이언트 암호를 클라이언트 암호에 입력합니다.
9. 드롭다운 목록에서, userinfo 엔드포인트에서 사용자 세부 정보를 Attributes request method(속성 요청 메서드)로 가져올 때 사용하는 HTTP 메서드(GET 또는 POST)를 선택합니다.
10. 승인하려는 범위의 이름을 입력합니다. 범위는 애플리케이션서 액세스하고자 하는 사용자 속성(예: name 및 email)을 정의합니다. 범위는 [OAuth 2.0](#) 사양에 따라 공백으로 구분됩니다.

애플리케이션 사용자는 앱에 이러한 속성들을 제공한다는 데 동의하라는 요청 메시지를 받게 됩니다.

11. IdP의 URL을 입력하고 검색 실행을 선택합니다.

예를 들어, Salesforce는 다음 URL을 사용합니다.

```
https://login.salesforce.com
```

Note

URL은 https://로 시작해야 하며, 슬래시 /로 끝나면 안 됩니다.

- 검색 실행에 실패한 경우에는 권한 부여 엔드포인트, 토큰 엔드포인트, Userinfo 엔드포인트 및 Jwks uri([JSON 웹 키](#)의 위치)를 제공해야 합니다.
12. 공급자 생성을 선택합니다.
 13. 왼쪽 탐색 모음에서 앱 클라이언트 설정을 선택합니다.
 14. 이전 단계에서 활성화된 자격 증명 공급자 중 하나로 설정한 OIDC 공급자를 선택합니다.
 15. Amazon Cognito 권한 부여 서버에 대한 콜백 URL을 입력하여 사용자가 인증된 후 호출합니다. 이것은 인증 성공 이후에 사용자가 리디렉션되는 페이지의 URL입니다.

```
https://www.example.com
```

16. 허용된 OAuth Flows에서 Authorization code grant(권한 부여 코드 허용)와 Implicit code grant(암시적 코드 허용)를 모두 활성화합니다.

특별히 하나를 제외하고 싶은 경우가 아니라면 허용된 OAuth Scopes 확인란을 모두 선택합니다.

17. [Save changes]를 선택합니다.
18. 왼쪽 탐색 모음의 속성 매핑 탭에서 OIDC 클레임 매핑을 사용자 풀 속성에 추가합니다.

- 기본적으로 OIDC 클레임 sub는 사용자 풀 속성 사용자 이름으로 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.
- 드롭다운 목록에서 대상 사용자 풀 속성을 선택합니다.
- [Save changes]를 선택합니다.
- 요약본으로 이동을 선택합니다.

OIDC IdP를 추가하려면(AWS CLI)

- [CreateIdentityProvider](#) API 메서드의 파라미터 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.

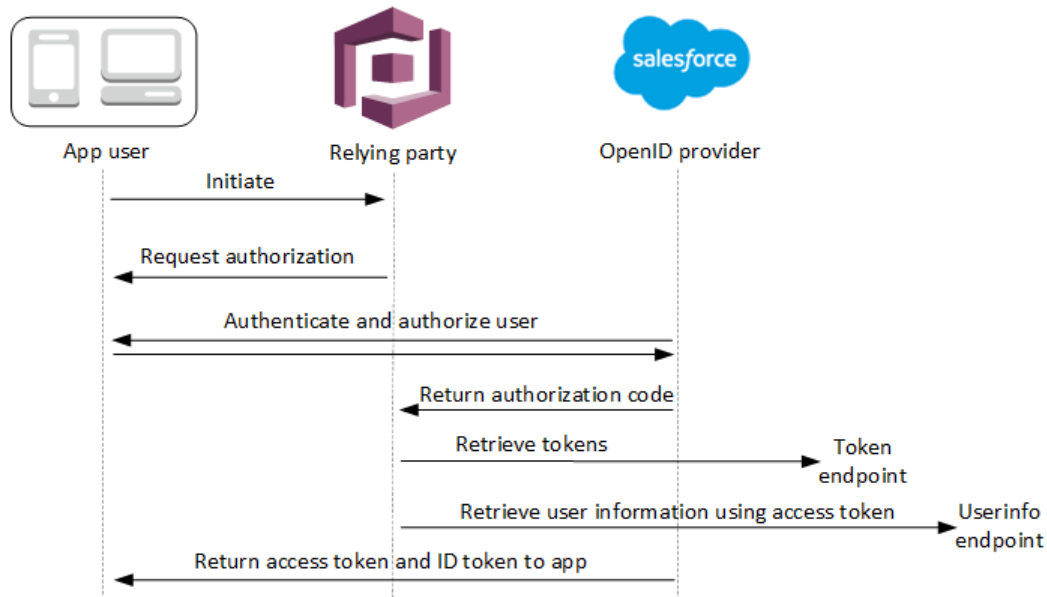
```
https://<your_user_pool_domain>/oauth2/authorize?
response_type=code&client_id=<your_client_id>&redirect_uri=https://www.example.com
```

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

OIDC 사용자 풀 IdP 인증 흐름

이것은 사용자가 OIDC IdP를 사용하여 애플리케이션에 로그인할 때의 인증 흐름입니다.

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.



OIDC는 OAuth 2.0에 있는 자격 증명 레이어로서, IdP가 OIDC 클라이언트 앱(신뢰 당사자)에 발행하는 JSON 형식(JWT) 자격 증명 토큰을 지정합니다. Amazon Cognito를 OIDC 신뢰 당사자로 추가하는 방법은 OIDC IdP 설명서를 참조하십시오.

사용자가 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 풀에 대한 자격 증명 공급자 속성 매핑 지정

AWS Management 콘솔이나 AWS CLI 또는 API를 사용하여 사용자 풀의 자격 증명 공급자에 대한 속성 매핑을 지정할 수 있습니다.

요구 사항

사용자 풀 속성을 자격 증명 풀 속성에 매핑할 때 다음 요구 사항을 기억하십시오.

- 사용자가 애플리케이션에 로그인할 때 필요한 각 사용자 풀 속성의 매핑이 있어야 합니다. 예를 들어 사용자 풀에 로그인용 이메일 속성이 필요한 경우 이 속성을 자격 증명 공급자와 동일한 해당 속성에 매핑합니다.
- 매핑된 각 사용자 풀 속성의 경우 최대 값 길이(2048 문자)는 Amazon Cognito가 자격 증명 공급자에서 가져오는 값에 대해 충분히 커야 합니다. 그렇지 않으면 사용자가 로그인을 할 때 Amazon Cognito에 오류가

발생합니다. 자격 증명 공급자 토큰에 사용자 지정 속성을 매핑할 경우, 길이를 최대 2,048자로 설정합니다.

- `username` 사용자 풀 속성은 다음 자격 증명 공급자의 특정 속성에만 매핑되어야 합니다.

ID 공급자	<code>username</code> 에 매핑할 속성
Facebook	<code>id</code>
Google	<code>sub</code>
Login with Amazon	<code>user_id</code>
OpenID Connect(OIDC) 공급자	<code>sub</code>

- 사용자가 애플리케이션에 로그인할 때 Amazon Cognito는 매핑된 사용자 풀 속성을 업데이트할 수 있어야 합니다. 사용자가 자격 증명 공급자를 통해 로그인하면 Amazon Cognito는 자격 증명 공급자의 최신 정보로 매핑된 속성을 업데이트합니다. 현재 값이 이미 최신 정보와 일치하더라도 Amazon Cognito는 매핑된 각 속성을 업데이트합니다. Amazon Cognito가 속성을 업데이트할 수 없는 경우 오류가 발생합니다. Amazon Cognito가 속성을 업데이트할 수 있는지 확인하려면 다음 요구 사항을 확인하십시오.
- 매핑된 사용자 풀 속성은 변경 가능해야 합니다. 변경 가능한 속성은 해당 속성에 대한 쓰기 권한이 있는 앱 클라이언트를 통해 업데이트될 수 있습니다. 사용자 풀 속성을 Amazon Cognito 콘솔에서 정의할 때 이를 변경 가능하도록 설정할 수 있습니다. 또는 [CreateUserPool](#) API 작업을 사용하여 사용자 풀을 생성하는 경우 이러한 각 속성에 대한 `Mutable` 파라미터를 `true`로 설정할 수 있습니다.
- 애플리케이션의 앱 클라이언트 설정에서 매핑된 속성은 쓰기 가능해야 합니다. Amazon Cognito 콘솔의 앱 클라이언트 페이지에서 쓰기 가능한 속성을 설정할 수 있습니다. 또는 [CreateUserPoolClient](#) API 작업을 사용하여 앱 클라이언트를 생성하는 경우 이러한 속성을 `WriteAttributes` 어레이에 추가할 수 있습니다.

사용자 풀에 대한 자격 증명 공급자 속성 매핑 지정(AWS Management 콘솔)

AWS Management 콘솔을 사용하여 사용자 풀의 자격 증명 공급자에 대한 속성 매핑을 지정할 수 있습니다.

소셜 자격 증명 공급자 속성 매핑을 지정하려면

1. [Amazon Cognito 콘솔](#)에 로그인합니다.
2. 탐색 창에서 사용자 풀 관리를 선택한 다음 편집할 사용자 풀을 선택합니다.
3. 속성 매핑 탭을 선택합니다.
4. Facebook, Google 또는 Amazon 탭을 선택합니다.
5. 매핑할 각 속성에 대해 다음 단계를 수행합니다.
 - a. 캡처 확인란을 선택합니다.
 - b. 사용자 풀 속성은 드롭다운 목록에서 소셜 자격 증명 공급자 속성으로 매핑할 사용자 풀 속성을 선택합니다.
 - c. 더 많은 속성이 필요할 경우 Facebook 속성 추가(또는 Google 속성 추가나 Add Amazon attribute(Amazon 속성 추가))를 선택하고 다음 단계를 수행합니다.
 - i. Facebook attribute(Facebook 속성)(또는 Google attribute(Google 속성)나 Amazon attribute(Amazon 속성)) 필드에서 매핑하려는 속성의 이름을 입력합니다.
 - ii. 사용자 풀 속성 필드의 드롭다운 목록에서 소셜 자격 증명 공급자 속성을 매핑할 사용자 풀 속성을 선택합니다.
 - d. [Save changes]를 선택합니다.

SAML 공급자 속성 매핑을 지정하려면

1. [Amazon Cognito 콘솔](#)에 로그인합니다.
2. 탐색 창에서 사용자 풀 관리를 선택한 다음 편집할 사용자 풀을 선택합니다.
3. 속성 매핑 탭을 선택합니다.
4. SAML 탭을 선택합니다.
5. 값을 캡처할 모든 속성의 캡처 상자를 선택합니다. 속성의 캡처 상자를 선택 취소하고 변경 사항을 저장하면 해당 속성의 매핑이 제거됩니다.
6. 드롭다운 목록에서 자격 증명 공급자를 선택합니다.
7. 매핑할 각 속성에 대해 다음 단계를 수행합니다.
 - a. SAML 속성 추가를 선택합니다.
 - b. SAML 속성 필드에서 매핑할 SAML 속성의 이름을 입력합니다.
 - c. 사용자 풀 속성 필드의 드롭다운 목록에서 SAML 속성을 매핑할 사용자 풀 속성을 선택합니다.
8. [Save changes]를 선택합니다.

사용자 풀에 대한 자격 증명 공급자 속성 매핑 지정(AWS CLI 및 AWS API)

다음 명령을 사용하여 사용자 풀에 대한 자격 증명 공급자 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.json에 다음 사항이 포함됩니다.

```
{
  "MetadataFile": "<SAML metadata XML>"
}
```

Note

<SAML metadata XML>에 따옴표(")가 있으면 이스케이프해야 합니다(\").

메타데이터 URL이 포함된 예제: `aws cognito-idp create-identity-provider --user-pool-id <user_pool_id> --provider-name=SAML_provider_1 --provider-type SAML --provider-details MetadataURL=<metadata_url> --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

- AWS API: [CreateIdentityProvider](#)

기존 자격 증명 공급자에 대한 속성 매핑을 지정하려면

- AWS CLI: `aws cognito-idp update-identity-provider`

예: `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 자격 증명 공급자에 대한 속성 매핑 정보를 가져오려면

- AWS CLI: `aws cognito-idp describe-identity-provider`

예: `aws cognito-idp describe-identity-provider --user-pool-id <user_pool_id> --provider-name <provider_name>`

- AWS API: [DescribeIdentityProvider](#)

Amazon Cognito 사용자 풀 보안 관리

멀티 팩터 인증(MFA)을 사용자 풀에 추가하여 사용자의 자격 증명을 보호할 수 있습니다. MFA는 사용자 이름 및 암호에만 의존하지 않는 두 번째 인증 방법을 추가합니다. 사용자 로그인에서의 두 번째 팩터로 SMS 문자 메시지 또는 일회용 암호(TOTP)를 선택하여 사용할 수 있습니다. 또한 위험 기반 모델에서 조정 인증을 사용하여 다른 인증 팩터가 필요할 때를 예측할 수 있습니다. 사용자 풀의 고급 보안 기능의 일부로, 손상된 자격 증명의 사용을 방지하는 기능도 포함되어 있습니다.

주제

- [사용자 풀에 멀티 팩터 인증\(MFA\) 추가 \(p. 107\)](#)
- [사용자 풀에 고급 보안 기능 추가 \(p. 110\)](#)

사용자 풀에 멀티 팩터 인증(MFA) 추가

멀티 팩터 인증(MFA)은 다른 인증 방법을 추가하여 사용자 이름과 암호에만 의존하지 않도록 함으로써 앱의 보안을 강화합니다. 사용자 로그인에서의 두 번째 팩터로 SMS 문자 메시지 또는 일회용 암호(TOTP)를 선택하여 사용할 수 있습니다.

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 \(p. 110\)](#) 단원을 참조하십시오.

주제

- [사전 조건 \(p. 107\)](#)
- [멀티 팩터 인증 구성 \(p. 108\)](#)
- [SMS 문자 메시지 MFA \(p. 108\)](#)
- [TOTP 소프트웨어 토큰 MFA \(p. 109\)](#)

사전 조건

시작하려면 다음이 필요합니다.

- 처음 사용자 풀을 생성할 때는 필수 사항으로만 MFA를 선택할 수 있습니다.
- MFA가 활성화되고 두 번째 팩터로 SMS 문자 메시지가 선택된 경우에는 전화 번호를 확인해야 합니다.
- 고급 보안 기능은 MFA를 활성화하고 Amazon Cognito 사용자 풀 콘솔에서 옵션으로 설정하도록 요구합니다. 자세한 내용은 [사용자 풀에 고급 보안 기능 추가 \(p. 110\)](#) 단원을 참조하십시오.

멀티 팩터 인증 구성

Amazon Cognito 콘솔에서 MFA를 구성하는 방법

1. 왼쪽 탐색 모음에서 MFA and verifications(MFA 및 확인)를 선택합니다.
2. 해제, 선택 사항 및 필수 사항에서 MFA 상태를 선택합니다.

Do you want to enable Multi-Factor Authentication?

Multi-Factor Authentication (MFA) increases security for your end users. If you choose 'optional', individual users can have MFA enabled for their account, and if you do, all users must use MFA. Phone numbers must be verified if MFA is enabled. You can configure adaptive MFA based on risk scoring of user sign in attempts. [Learn more about multi-factor authentication.](#)

Note: separate charges apply for sending text messages.

☐ Off ☒ Optional ☐ Required

Which second factors do you want to enable?

Your users will be able to configured and choose any of the factors you enabled. You must enable at least one factor.

☒ SMS text message

Important: In order to send SMS messages to users to verify phone numbers or for MFA you must request a spending limit.

Note: separate charges may apply for sending SMS text messages. See the [Worldwide SMS pricing page](#) for more details.

☒ Time-based One-time Password

3. 위험 기반 조정 인증을 사용하고 있는 경우에 선택 사항을 선택하여 사용자별로 MFA를 활성화합니다. 조정 인증에 대한 자세한 내용은 [사용자 풀에 고급 보안 기능 추가 \(p. 110\)](#) 단원을 참조하십시오.
4. SMS 문자 메시지 또는 Time-based One-time Password(시간 기반 일회용 암호)를 두 번째 팩터로 사용할 수 있습니다. 앱에서 지원할 두 번째 팩터를 선택합니다.
5. 두 번째 팩터로 SMS 문자 메시지를 사용 중이고 이 권한으로 IAM 역할을 정의하지 않은 경우에는 콘솔에서 역할을 생성할 수 있습니다. 역할 생성을 선택하여 Amazon Cognito가 대신 사용자에게 SMS 메시지를 전송하도록 허용하는 IAM 역할을 생성합니다. 자세한 내용은 [IAM 역할](#)을 참조하십시오.
6. [Save changes]를 선택합니다.

SMS 문자 메시지 MFA

사용자가 MFA를 컨 채로 로그인할 때는 먼저 사용자 이름과 암호를 입력하고 제출합니다. 권한 부여 코드가 전송된 위치를 나타내는 `getMFA` 응답이 클라이언트 앱에 수신됩니다. 클라이언트 앱에서는 코드를 찾을 수 있는 위치(예: 코드가 전송된 전화 번호)를 사용자에게 알리고, 코드를 입력할 양식을 제공하고, 로그인 프로세스를 완료하기 위해 코드를 제출해야 합니다. 대상이 표시됩니다(전화 번호의 마지막 4자리만 표시됨). 앱이 호스팅 UI를 사용하는 경우 사용자에게 MFA 코드를 입력하는 페이지를 표시합니다.

SMS 문자 메시지 권한 부여 코드는 3분 동안 유효합니다.

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 권한을 가진 관리자는 사용자의 전화 번호를 변경할 수 있지만, AWS CLI 또는 API를 통해서만 변경이 가능합니다.

사용자가 SMS 문자 메시지 MFA 흐름을 성공적으로 통과하면 해당 사용자의 전화 번호도 확인됨으로 표시됩니다.

Note

MFA를 위한 SMS 요금은 따로 부과됩니다. 이메일 주소로 확인 코드를 전송하는 것은 무료입니다. Amazon SNS 요금에 대한 자세한 내용은 [전 세계 SMS 요금](#)을 참조하십시오. 현재 SMS 메시징이 가능한 국가의 목록은 [지원되는 리전 및 국가](#)를 참조하십시오.

Important

전화 번호 및 SMS 문자 메시지 MFA를 확인하기 위해 SMS 메시지가 전송되도록 하려면 Amazon SNS의 증가된 지출 한도를 요청해야 합니다.

Amazon Cognito는 사용자에게 SMS 메시지를 전송하기 위해 Amazon SNS를 사용합니다. 가 전달하는 SMS 메시지 수에는 지출 한도가 적용됩니다. AWS 계정 및 개별 메시지에 지출 한도를 지정할 수 있으며, 이 한도는 SMS 메시지 전송 비용에만 적용됩니다.

계정당 기본 지출 한도는(지정되지 않은 경우) 매월 1.00USD입니다. 이 한도를 높이려면 AWS 지원 센터에 [SNS 한도 증가 사례](#)를 제출하십시오. 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 멀티 팩터 인증\(MFA\) 추가 \(p. 107\)](#) 단원을 참조하십시오.

TOTP 소프트웨어 토큰 MFA

사용자는 자신의 사용자 이름과 암호가 확인된 후 TOTP 소프트웨어 토큰 MFA가 활성화되면 시간 기반 일회용(TOTP) 암호를 사용하여 인증을 완료해야 합니다. 앱이 Amazon Cognito 호스팅 UI를 사용하여 사용자를 로그인하게 하는 경우, UI는 사용자가 사용자 이름과 암호를 제출한 후 TOTP 암호를 입력할 수 있는 두 번째 페이지를 표시합니다.

Amazon Cognito 콘솔, Amazon Cognito 호스팅 UI 또는 Amazon Cognito API를 통해 사용자 풀에 대해 TOTP MFA를 활성화할 수 있습니다. 사용자 풀 수준에서 [SetUserPoolMfaConfig](#)를 호출하여 MFA를 구성하고 TOTP MFA를 활성화할 수 있습니다.

Note

사용자 풀에서 TOTP 소프트웨어 토큰 MFA가 활성화되어 있지 않은 경우, 사용자는 토큰을 연결 또는 확인할 수 없습니다. 사용자는 "Software Token MFA has not been enabled by the userPool(사용자 풀에서 소프트웨어 토큰 MFA가 활성화되지 않음)" 같은 `SoftwareTokenMFANotFoundException` 예외 메시지를 수신합니다.

사용자에 대해 TOTP를 구성하는 것은 사용자가 일회용 암호를 입력하여 유효성을 검사하는 보안 코드를 받는 다단계 프로세스입니다. 그런 다음 사용자에게 대해 TOTP MFA를 활성화하거나 TOTP를 사용자에게 대한 기본 MFA 방법으로 설정할 수 있습니다.

사용자 풀에 MFA를 추가하는 방법은 [사용자 풀에 멀티 팩터 인증\(MFA\) 추가 \(p. 107\)](#) 단원을 참조하십시오.

TOTP 토큰 연결

1. 사용자가 TOTP 소프트웨어 토큰 MFA를 선택하면 [AssociateSoftwareToken](#)을 호출하여 해당 사용자 계정에 대해 생성된 고유한 공유 보안 키 코드를 반환합니다. 이 API 메서드 요청은 액세스 토큰 또는 세션 문자열 중 하나를 사용합니다. 편의상 보안 키를 QR(Quick Response) 코드로 배포할 수 있습니다.
2. 키 코드 또는 QR 코드는 앱에 표시되며, 사용자는 이를 Google Authenticator와 같은 TOTP 생성 앱에 입력해야 합니다.
3. 사용자는 TOTP 생성 앱에 키 코드를 입력하여 새 계정을 클라이언트 앱에 연결합니다.

TOTP 토큰 확인

1. 새 TOTP 계정이 앱에 연결되면 임시 암호가 생성됩니다.

2. 사용자는 앱에 임시 암호를 입력하고, [VerifySoftwareToken](#) 호출로 응답합니다. 서비스 서버에서 TOTP 코드가 생성되고 이를 사용자의 임시 암호와 비교합니다. 일치하면 서비스가 이를 확인된 것으로 표시합니다.
3. 코드가 올바르면 사용된 시간이 범위 및 최대 재시도 횟수 내에 있는지 확인합니다. 사용자가 모든 단계를 통과하면 확인이 완료됩니다.

그렇지 않고 코드가 잘못된 경우 확인을 완료할 수 없으며 사용자가 다시 시도하거나 취소할 수 있습니다. 사용자가 TOTP 생성 앱의 시간을 동기화하는 것이 좋습니다.

TOTP MFA로 로그인

1. 사용자는 사용자 이름과 암호를 입력하여 클라이언트 앱에 로그인합니다.
2. TOTP MFA 챌린지가 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. 각 로그인에 대해 [RespondToAuthChallenge](#)를 호출하여 새 TOTP 인증 챌린지에 대한 응답을 받아야 합니다.
5. Amazon Cognito에서 토큰을 확인하면 로그인이 성공하고 사용자가 인증 흐름을 계속합니다.

TOTP 토큰 제거

1. 앱은 사용자가 TOTP 토큰을 제거할 수 있도록 허용해야 합니다.
2. 클라이언트 앱은 사용자에게 암호를 입력하라고 요청해야 합니다.
3. 암호가 올바르면 TOTP 토큰을 제거합니다.

Note

TOTP 소프트웨어 토큰 삭제 작업은 현재 이 API에서 제공되지 않습니다. 이 기능은 향후 릴리스에서 제공될 예정입니다. [SetUserMFAPreference](#)를 사용하여 개별 사용자의 TOTP MFA를 비활성화합니다.

사용자 풀에 고급 보안 기능 추가

사용자 풀을 생성한 후에는 Amazon Cognito 콘솔의 탐색 모음에서 Advanced security(고급 보안)에 액세스할 수 있습니다. 사용자 풀 고급 보안 기능을 켜고 여러 위험에 대응하여 취하는 조치를 사용자 지정할 수 있습니다. 또는 감사 모드를 사용하여 조치를 취하지 않고 탐지된 위험에 대한 지표를 수집할 수 있습니다. 감사 모드에서 고급 보안 기능은 Amazon CloudWatch에 지표를 게시합니다. [고급 보안 측정치 보기 \(p. 116\)](#) 단원을 참조하십시오.

Note

Amazon Cognito 고급 보안 기능에는 추가 요금이 적용됩니다. [Amazon Cognito 요금 페이지](#)를 참조하십시오.

주제

- [사전 조건 \(p. 111\)](#)
- [고급 보안 기능 구성 \(p. 111\)](#)
- [손상된 자격 증명 확인 \(p. 112\)](#)
- [조정 인증 사용 \(p. 113\)](#)
- [고급 보안 측정치 보기 \(p. 116\)](#)
- [앱에서 사용자 풀 고급 보안 기능 활성화 \(p. 117\)](#)

사전 조건

시작하려면 다음이 필요합니다.

- 앱 클라이언트가 포함된 사용자 풀. 자세한 내용은 [사용자 풀 시작하기 \(p. 13\)](#) 단원을 참조하십시오.
- Amazon Cognito 콘솔에서 멀티 팩터 인증(MFA)을 선택 사항으로 설정하여 위험 기반 조정 인증 기능을 사용합니다. 자세한 내용은 [사용자 풀에 멀티 팩터 인증\(MFA\) 추가 \(p. 107\)](#) 단원을 참조하십시오.
- 알림을 위해 이메일을 사용하고 있는 경우에는 [Amazon SES 콘솔](#)로 이동하여 알림 이메일에서 사용할 이메일 주소나 도메인을 구성 및 확인합니다. Amazon SES에 대한 자세한 내용은 [Amazon SES에서 자격 증명 확인](#)을 참조하십시오.

고급 보안 기능 구성

사용자 풀에서 고급 보안 기능을 구성하는 방법

1. 왼쪽의 탐색 모음에서 Advanced security(고급 보안)를 선택합니다.
2. 이 사용자 풀의 고급 보안 기능을 활성화하시겠습니까?에서 예를 선택하여 고급 보안 기능을 활성화합니다. 또는 Audit only(감사 전용)를 선택하여 정보를 수집하고 사용자 풀 데이터를 Amazon CloudWatch에 전송합니다.

작업을 활성화하기 전에 고급 보안 기능을 2주 동안 감사 모드로 유지하는 것이 좋습니다. 덕분에 Amazon Cognito가 앱 사용자의 사용 패턴을 이해할 수 있습니다.

3. 드롭다운 목록에서 설정을 사용자 지정하려는 앱 클라이언트는 무엇입니까?를 선택합니다. 기본적으로 설정은 모든 앱 클라이언트를 위한 전역 값으로 남아 있습니다.
4. Which action do you want to take with the compromised credentials?(손상된 자격 증명에 대해 어떤 조치를 취하시겠습니까?)에서 허용 또는 Block use(사용 차단)를 선택합니다.
5. Customize when compromised credentials are blocked(손상된 자격 증명에 차단된 경우 사용자 지정)를 선택하여 어떤 이벤트에서 손상된 자격 증명 확인을 트리거해야 하는지 선택합니다.

- 로그인
- 가입
- 암호 변경


6. How do you want to use adaptive authentication for sign-in attempts rated as low, medium and high risk?(위험도 낮음, 중간, 높음 등급의 로그인 시도에 대해 조정 인증을 어떻게 사용하고 싶으십니까?)에서 악의적인 로그인 시도에 어떻게 대응할 것인지 선택합니다. 로그인 시도를 허용 또는 차단하거나 로그인을 허용하기 앞서 추가 챌린지를 요구할 수 있습니다.

비정상적인 로그인 시도가 탐지될 때 이메일 알림을 보내려면 Notify users(사용자 알림)를 선택합니다.

How do you want to use adaptive authentication for sign-in attempts rated as low, medium and high risk?

You can use adaptive authentication to add protections against sign-in attempts that are rated as higher risk such as coming from a new device or location. For more information, see [adaptive authentication](#).

	Allow	Optional MFA	Require MFA	Block	Notify users
Low risk	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="checkbox"/>
Medium risk	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="checkbox"/>
High risk	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input checked="" type="checkbox"/>

Choosing allow will still log all attempted sign-ins rated as higher risk to [AWS Cloudwatch](#). 

- 이전 단계에서 Notify users(사용자 알림)를 선택한 경우에는 알림 메시지 사용자 지정 양식을 사용하여 이메일 알림 메시지를 사용자 지정할 수 있습니다.

Notification message customization

Customize the email user notification messages sent to users after sign-in attempts with risks detected. Enter your settings below for SES.

SES Region

US East (Virginia)

Source ARN

arn:aws:ses:us-east-1:123456789012:identity/janedoe@example.com

You must verify your email address with Amazon SES before you can select it. [Verify an SES identity.](#)

FROM email address

example@example.com

REPLY-TO email address

example@example.com

Adaptive authentication notification messages [Customize...](#)

- 사용자 지정을 선택하여 HTML 및 일반 텍스트 이메일 버전 모두에서 조정 인증 알림을 사용자 지정합니다. 이메일 템플릿에 대한 자세한 내용은 [메시지 템플릿 \(p. 210\)](#) 단원을 참조하십시오.
- 고급 보안 위험 평가에 관계 없이 Always allow(항상 허용) 또는 Always block(항상 차단)을 적용하고 싶은 IP 주소를 입력합니다. [CIDR 표기법](#)(예: 192.168.100.0/24)으로 IP 주소 범위를 지정합니다.
- [Save changes]를 선택합니다.

손상된 자격 증명 확인

Amazon Cognito는 사용자의 자격 증명(사용자 이름 및 암호)이 다른 곳에서 손상되었는지 여부를 탐지할 수 있습니다. 사용자가 하나 이상의 사이트에서 자격 증명을 다시 사용하거나 추측하기 쉬운 암호를 사용할 때 이런 문제가 발생할 수 있습니다.

Amazon Cognito 콘솔의 Advanced security(고급 보안) 페이지에서 손상된 자격 증명이 탐지되는 경우 사용자를 허용할 것인지 차단할 것인지 선택할 수 있습니다. 차단이 되면 사용자는 다른 암호를 선택해야 합니다. 허용을 선택하면 손상된 자격 증명에 대한 모든 사용 시도가 Amazon CloudWatch에 게시됩니다. 자세한 내용은 [고급 보안 측정치 보기 \(p. 116\)](#) 단원을 참조하십시오.

또한 Amazon Cognito가 로그인, 가입 및 암호 변경 동안 손상된 자격 증명을 확인할 것인지 여부를 선택할 수 있습니다.

Which action do you want to take with the compromised credentials?

You can detect and protect your users from using credentials that have been exposed through breaches of other websites. [Learn more about compromised credentials](#)

☐ Allow ☒ Block use

Blocking use will require users to choose a different password. Choosing allow will still log all attempted uses of compromised credentials to [AWS CloudWatch](#). [↗](#)

Which events should trigger compromised credentials checks?

☒ Sign in ☒ Sign up ☒ Password change

At least one event must be selected.

Note

현재 Amazon Cognito는 로그인 중에 암호를 보내지 않는 SRP(Secure Remote Password) 흐름에서의 로그인 작업에 대해 손상된 자격 증명을 확인하지 않습니다. ADMIN_NO_SRP_AUTH 흐름에서 [AdminInitiateAuth](#) API를, USER_PASSWORD_AUTH 흐름에서 [InitiateAuth](#) API를 사용하는 로그인에는 자격 증명 손상이 없는지 확인됩니다.

사용자 풀에 손상된 자격 증명 차단을 추가하는 방법은 [사용자 풀에 고급 보안 기능 추가 \(p. 110\)](#) 단원을 참조하십시오.

조정 인증 사용

조정 인증을 통해 의심스러운 로그인을 차단하거나 위험 수준이 높아졌을 때 두 번째 팩터를 통한 인증을 요구하도록 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 새로운 장치, 사용자 위치 또는 IP 주소의 탐지 여부를 포함하여 다양한 요소를 기반으로 합니다.

Amazon Cognito는 Amazon CloudWatch에 로그인 시도, 해당 위험 수준, 실패한 챌린지를 게시합니다. 자세한 내용은 [고급 보안 측정치 보기 \(p. 116\)](#) 단원을 참조하십시오.

사용자 풀에 조정 인증을 추가하는 방법은 [사용자 풀에 고급 보안 기능 추가 \(p. 110\)](#) 단원을 참조하십시오.

주제

- [조정 인증 개요 \(p. 113\)](#)
- [장치 지문 생성 \(p. 114\)](#)
- [사용자 이벤트 기록 보기 \(p. 114\)](#)
- [이벤트 피드백 제공 \(p. 116\)](#)
- [알림 메시지 전송 \(p. 116\)](#)

조정 인증 개요

Amazon Cognito 콘솔의 Advanced security(고급 보안) 페이지에서 서로 다른 위험 수준에서 취할 조치 및 사용자에 대한 알림 메시지 사용자 지정을 비롯해 적응 인증을 위한 설정을 선택할 수 있습니다.

위험 수준별로 다음 옵션 중에서 선택할 수 있습니다.

옵션	작업
허용	추가 요소 없이 로그인 시도가 허용됩니다.
선택 사항 MFA	MFA가 구성된 사용자는 로그인을 위해 두 번째 팩터 챌린지를 완료해야 합니다. MFA가 구성되지 않

옵션	작업
	은 사용자는 추가 팩터 없이 로그인을 하는 것이 허용됩니다.
MFA 필요	MFA이 구성된 사용자는 로그인을 위해 두 번째 팩터 챌린지를 완료해야 합니다. MFA가 구성되지 않은 사용자는 로그인이 차단됩니다.
차단	해당 위험 수준의 모든 로그인 시도가 차단됩니다.

장치 지문 생성

서버에서 [AdminInitiateAuth](#) 또는 [AdminRespondToAuthChallenge](#) 같은 Amazon Cognito 인증 API를 호출할 때는 ContextData 파라미터에서 사용자의 소스 IP를 전달해야 합니다. 이 IP는 Amazon Cognito 컨텍스트 데이터 모음 라이브러리를 사용해 수집한 서버 이름, 서버 경로 및 인코딩된 장치 지문 데이터에 추가됩니다.

웹 또는 모바일 앱에서 고급 보안 기능을 활성화하는 방법은 [앱에서 사용자 풀 고급 보안 기능 활성화](#) (p. 117) 단원을 참조하십시오.

앱이 서버에서 Amazon Cognito를 호출 중일 때는 클라이언트 측에서 사용자 컨텍스트 데이터를 수집합니다. 다음은 JavaScript SDK 메서드 `getData`를 사용하는 예제입니다.

```
var encodedData = AmazonCognitoAdvancedSecurityData.getData(username, userPoolId,
  clientId);
```

Note

앱에 최신 Amazon Cognito SDK를 통합하는 것이 좋습니다. 이렇게 하면 적응 인증에서 장치 ID, 모델 및 시간대 같은 장치 지문 정보를 수집할 수 있습니다. Amazon Cognito SDK에 대한 자세한 내용은 [사용자 풀 SDK 설치](#)를 참조하십시오.

사용자 이벤트 기록 보기

사용자 및 그룹으로 이동하여 Amazon Cognito 콘솔에서 사용자를 선택하면 해당 사용자의 로그인 기록을 확인할 수 있습니다. 사용자 이벤트 기록은 Amazon Cognito에 2년 동안 보관됩니다.

Date (UTC)	Event	Result	Risk level	Risk decision	Challenge	IP	Device
Jan 23, 2018 11:43:05 PM	Sign In	Pass	-	No Risk	Password:Success	52.94.36.11	Chrome, Windows 10
Jan 23, 2018 11:42:14 PM	Sign In	Pass	-	No Risk	Password:Success	52.94.36.11	Chrome, Windows 10
Jan 18, 2018 9:21:21 PM	Sign In	Fail	High	Account Takeover	Password:Success	67.132.130.174	Chrome Mobile, Android Mobile
Jan 18, 2018 9:20:28 PM	Sign In	In Progress	High	Account Takeover	Password:Success	67.132.130.174	Chrome Mobile, Android Mobile
Jan 18, 2018 9:18:18 PM	Sign In	Pass	-	No Risk	Password:Success	67.132.130.174	Chrome Mobile, Android Mobile

각 로그인 이벤트에는 이벤트 ID, 위치 등의 컨텍스트 데이터, 디바이스 세부 정보, 그와 관련된 위험 탐지 결과가 있습니다.

Jan 23, 2018 11:42:14 PM Sign In event

Event ID	09da6736-0097-11e8-9495-0dba8a248f46	Event risk	-
Event type	Sign In	Risk decision	No Risk
Event date (UTC)	Jan 23, 2018 11:42:14 PM	Event response	Pass
IP address	52.94.36.11	Feedback	-
Device	Chrome, Windows 10	Feedback date	-
Time zone	-	Feedback provider	-
City	London		
Country	United Kingdom		
Challenge response	Password:Success		

Mark event as valid

Mark event as invalid

이벤트 ID를 발급된 토큰과도 관련지을 수 있습니다. ID 토큰 및 액세스 토큰 같이 발급된 토큰은 이 이벤트 ID를 페이로드로 전달합니다. 새로 고침 토큰을 사용하더라도 원래 이벤트 ID가 유지됩니다. 원래 이벤트 ID를 추적하여 Amazon Cognito 토큰을 발급한 로그인 이벤트의 이벤트 ID를 찾을 수 있습니다. 이를 통해 특정 인증 이벤트에 대한 시스템 내의 토큰 사용을 추적할 수 있습니다.

이벤트 피드백 제공

이벤트 피드백은 위험 평가에 실시간으로 영향을 주며, 위험 평가 알고리즘을 지속적으로 개선해 나갑니다. 또한 Amazon Cognito 콘솔 및 API를 통해 로그인 시도의 유효성에 대한 피드백을 제공할 수 있습니다.

콘솔의 사용자 및 그룹 탭에 로그인 기록이 나열되며, 항목을 클릭하여 해당 이벤트를 유효하거나 유효하지 않은 것으로 표시할 수 있습니다. 또한 사용자 풀 API 메서드 [AdminUpdateAuthEventFeedback](#)이나 AWS CLI 명령 [admin-update-auth-event-feedback](#)을 통해 피드백을 제공할 수 있습니다.

알림 메시지 전송

고급 보안 기능을 통해 Amazon Cognito는 사용자에게 로그인 시도를 알리고, 링크를 클릭하여 로그인이 유효한지 유효하지 않은지 나타내라는 메시지를 표시하고, 사용자의 피드백을 참조하여 사용자 풀에 대한 위험 탐지 정확도를 높일 수 있습니다.

How do you want to use adaptive authentication for sign-in attempts rated as low, medium and high risk?(위험도 낮음, 중간, 높음 등급의 로그인 시도에 대해 조정 인증을 어떻게 사용하고 싶으십니까?) 섹션에서 위험도의 낮음, 중간, 높음에 따라 Notify Users(사용자 알림)를 선택합니다.

How do you want to use adaptive authentication for sign-in attempts rated as low, medium and high risk?

You can use adaptive authentication to add protections against sign-in attempts that are rated as higher risk such as coming from an untrusted device or location. For more information, see [adaptive authentication](#).

	Allow	Optional MFA	Require MFA	Block	Notify users
Low risk	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="checkbox"/>
Medium risk	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="checkbox"/>
High risk	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input checked="" type="checkbox"/>

Choosing allow will still log all attempted sign-ins rated as higher risk to [AWS Cloudwatch](#). [↗](#)

알림 이메일을 사용자 지정하고, 일반 텍스트 및 HTML 버전을 모두 제공할 수 있습니다. 조정 인증 알림 메시지에서 사용자 지정을 선택하여 이메일 알림을 사용자 지정합니다. 이메일 템플릿에 대한 자세한 내용은 [메시지 템플릿 \(p. 210\)](#) 단원을 참조하십시오.

고급 보안 측정치 보기

Amazon Cognito는 Amazon CloudWatch의 계정에 고급 보안 기능의 측정치를 게시합니다. 고급 보안 측정치는 위험 수준 및 요청 수준별로 함께 그룹화됩니다.

CloudWatch 콘솔을 사용해 지표를 보는 방법

1. <https://console.aws.amazon.com/cloudwatch/>에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 [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.
 - 지표를 그래프로 표시하려면 지표 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
CompromisedCredentialsRisk	Amazon Cognito가 이상 있는 자격 증명을 탐지한 요청	작업: PasswordChange, signIn 또는 signUp 등의 작업 유형 UserPoolId: 사용자 풀의 식별자 RiskLevel: 높음(기본), 중간 또는 낮음
AccountTakeOverRisk	Amazon Cognito가 계정 탈취 위험을 탐지한 요청	RiskLevel: 높음, 중간 또는 낮음
OverrideBlock	Amazon Cognito가 개발자가 제공한 구성의 결과로 차단한 요청입니다.	RiskLevel: 높음(기본), 중간 또는 낮음
Risk	Amazon Cognito가 위험하다고 표시한 요청	작업: PasswordChange, signIn 또는 signUp 등의 작업 유형 UserPoolId: 사용자 풀의 식별자
NoRisk	Amazon Cognito가 어떤 위험도 식별하지 않은 요청	작업: PasswordChange, signIn 또는 signUp 등의 작업 유형 UserPoolId: 사용자 풀의 식별자

Amazon Cognito는 CloudWatch에서 분석할 수 있도록 사전 정의된 두 개의 측정치 그룹을 제공합니다. By Risk Classification(위험 분류별)은 Amazon Cognito가 위험하다고 식별한 요청에 대한 위험의 세부 수준을 식별하고, By Request Classification(요청 분류별)은 요청 수준별로 집계된 측정치를 반영합니다.

집계된 측정치 그룹	설명
위험 분류별	Amazon Cognito가 위험하다고 식별한 요청
요청 분류별	요청별로 집계된 측정치

앱에서 사용자 풀 고급 보안 기능 활성화

사용자 풀에 대한 고급 보안 기능을 구성한 후에는 웹이나 모바일 앱에서 이를 활성화해야 합니다.

Amazon Cognito에 웹 또는 모바일 앱을 추가하는 방법은 [Amazon Cognito 사용자 풀에 웹 또는 모바일 앱 추가 \(p. 22\)](#) 단원을 참조하십시오.

JavaScript에서 고급 보안 기능을 사용하는 방법

1. Amazon Cognito SDK를 최신 버전으로 업데이트해야 할 수 있습니다. Amazon Cognito SDK에 대한 자세한 내용은 [사용자 풀 SDK 설치](#)를 참조하십시오.
2. Auth SDK를 사용하여 호스팅된 UI를 활성화하는 방법은 [CognitoAuth JavaScript 샘플 앱](#)을 참조하십시오.

3. AdvancedSecurityDataCollectionFlag를 true로 설정합니다. 또한 UserPoolId를 사용자 풀 ID로 설정합니다.
4. 애플리케이션에서 <리전>을 us-east-1 같은 AWS 리전으로 대체하고 JavaScript 파일에 이 소스 참조를 추가합니다.

```
<script src="https://amazon-cognito-assets.<region>.amazoncognito.com/amazon-cognito-advanced-security-data.min.js"></script>
```

자세한 내용은 [JavaScript용 Amazon Cognito Auth SDK 샘플](#)을 참조하십시오.

Android에서 고급 보안 기능을 사용하는 방법

1. Amazon Cognito SDK를 최신 버전으로 업데이트해야 할 수 있습니다. Amazon Cognito SDK에 대한 자세한 내용은 [사용자 풀 SDK 설치](#)를 참조하십시오.
2. Auth SDK를 사용하여 호스팅된 UI를 활성화하는 방법은 [CognitoAuth Android 샘플 앱](#)을 참조하십시오.
3. Gradle의 Maven을 통해 aws-android-sdk-cognitoidentityprovider-asf를 가져오는 동안 { transitive = true; }를 사용합니다.

build.gradle 파일에 종속 요소로 이를 포함시킵니다.

```
compile "com.amazonaws:aws-android-sdk-cognitoidentityprovider-asf:1.0.0"
```

자세한 내용은 [Android용 AWS SDK - Amazon Cognito 자격 증명 공급자 ASF](#)를 참조하십시오.

iOS에서 고급 보안 기능을 사용하는 방법

1. Amazon Cognito SDK를 최신 버전으로 업데이트해야 할 수 있습니다. Amazon Cognito SDK에 대한 자세한 내용은 [사용자 풀 SDK 설치](#)를 참조하십시오.
2. Auth SDK를 사용하여 호스팅된 UI를 활성화하는 방법은 [CognitoAuth iOS 샘플 앱](#)을 참조하십시오.
3. Info.plist를 사용하여 Auth SDK를 구성하려면 PoolIdForEnablingASF 키를 Amazon Cognito 사용자 풀 구성에 추가하고 이를 사용자 풀 ID로 설정합니다.

AWSCognitoAuthConfiguration을 사용하여 Auth SDK를 구성하려면 [이 이니셜라이저](#)를 사용하고 userPoolIdForEnablingASF로서 사용자 풀 ID를 지정합니다.

자세한 내용은 [AWSCognitoIdentityProviderASF](#)를 참조하십시오.

Lambda 트리거를 사용하여 사용자 풀 워크플로우 사용자 지정

AWS Lambda 함수를 생성한 다음, 사용자 가입, 확인 및 로그인(인증) 같은 사용자 풀 작업 시 Lambda 트리거를 사용하여 해당 함수를 시작할 수 있습니다. 인증 문제 추가, 사용자 마이그레이션 및 확인 메시지 사용자 지정을 수행할 수 있습니다.

Important

Amazon Cognito는 Lambda 함수를 동기식으로 호출합니다. 호출된 함수는 5초 내에 응답을 해야 합니다. 그렇지 않으면 가 호출을 다시 시도합니다. 3번 연속 실패하면 함수 제한 시간을 초과하게 됩니다. 5초로 설정된 제한 시간 값은 변경이 불가능합니다. 자세한 내용은 [Lambda 프로그래밍 모델](#)을 참조하십시오.

Important

AWS Lambda 트리거를 삭제하면 사용자 풀의 해당 트리거를 업데이트해야 합니다. 예를 들어 사후 인증 트리거를 삭제하면 해당 사용자 풀의 사후 인증 트리거를 없음으로 설정해야 합니다.

다음 표는 가능한 사용자 지정 내용을 요약한 것입니다.

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.
	인증 문제 응답 확인	사용자 지정 인증 흐름에서 응답이 올바른지 결정합니다.
인증 이벤트	the section called “사전 인증 Lambda 트리거” (p. 131)	로그인 요청 승인 및 거절에 대한 사용자 지정 검증입니다.
	the section called “사후 인증 Lambda 트리거” (p. 133)	사용자 지정 분석에 대한 이벤트 로그입니다.
	the section called “사전 토큰 생성 Lambda 트리거” (p. 144)	토큰 클레임의 보강 및 제한
가입	the section called “사전 가입 Lambda 트리거” (p. 123)	가입 요청 승인 및 거절에 대한 사용자 지정 검증입니다.
	the section called “사후 확인 Lambda 트리거” (p. 128)	사용자 지정 분석에 사용할 사용자 지정 환영 메시지 또는 이벤트 로그입니다.
	the section called “사용자 마이그레이션 Lambda 트리거” (p. 147)	사용자를 기존 사용자 디렉터리에서 사용자 풀로 마이그레이션합니다.
메시지	the section called “사용자 지정 메시지 Lambda 트리거” (p. 150)	고급 사용자 지정 및 메시지 현지화
토큰 생성	the section called “사전 토큰 생성 Lambda 트리거” (p. 144)	ID 토큰 속성을 추가하거나 제거합니다.

주제

- [사용자 풀 Lambda 트리거 추가 \(p. 120\)](#)
- [사용자 풀 Lambda 트리거 이벤트 \(p. 120\)](#)
- [사용자 풀 Lambda 트리거 공통 파라미터 \(p. 120\)](#)
- [사용자 풀 Lambda 트리거 소스 \(p. 121\)](#)
- [사전 가입 Lambda 트리거 \(p. 123\)](#)
- [사후 확인 Lambda 트리거 \(p. 128\)](#)
- [사전 인증 Lambda 트리거 \(p. 131\)](#)
- [사후 인증 Lambda 트리거 \(p. 133\)](#)
- [사용자 지정 인증 문제 Lambda 트리거 \(p. 136\)](#)
- [사전 토큰 생성 Lambda 트리거 \(p. 144\)](#)
- [사용자 마이그레이션 Lambda 트리거 \(p. 147\)](#)

- [사용자 지정 메시지 Lambda 트리거 \(p. 150\)](#)

사용자 풀 Lambda 트리거 추가

콘솔을 사용하여 사용자 풀 Lambda 트리거를 추가하려면

1. [Lambda 콘솔](#)을 사용하여 Lambda 함수를 생성합니다. Lambda 함수에 관한 자세한 내용은 [AWS Lambda Developer Guide](#) 단원을 참조하십시오.
2. [Amazon Cognito 콘솔](#)로 이동하여, Manage User Pools(사용자 풀 관리)를 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 [사용자 풀을 생성합니다](#).
4. 사용자 풀의 탐색 모음에서 트리거 탭을 선택합니다.
5. 사전 가입 또는 사전 인증 같은 Lambda 트리거를 선택하고, Lambda 함수 드롭다운 목록에서 사용자의 Lambda 함수를 선택합니다.
6. [Save changes]를 선택합니다.
7. Lambda 콘솔에서 CloudWatch를 사용하여 사용자의 Lambda 함수를 로그할 수 있습니다. 자세한 내용은 [Lambda에 대한 CloudWatch Logs 액세스](#) 단원을 참조하십시오.

사용자 풀 Lambda 트리거 이벤트

Amazon Cognito가 Lambda 함수에 이벤트 정보를 보내면 응답 속의 모든 변경 사항과 함께 동일한 이벤트 객체를 Amazon Cognito로 반환합니다. 이 이벤트에는 다음과 같은 트리거 공통 파라미터가 표시됩니다.

JSON

```
{
  "version": number,
  "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 [사용자 풀 Lambda 트리거 소스 \(p. 121\)](#) 섹션을 참조하십시오.

리전

AWSRegion 인스턴스로서의 AWS 리전입니다.

userPoolId

사용자 풀의 사용자 풀 ID입니다.

사용자 이름

현재 사용자의 사용자 이름입니다.

callerContext

다음과 같이 구성되는 호출자 컨텍스트입니다.

awsSdkVersion

AWS SDK 버전 번호입니다.

clientId

사용자 풀과 연결된 클라이언트의 ID입니다.

요청

Amazon Cognito 서비스에서 보내는 요청입니다. 이 요청은 다음 항목을 포함해야 합니다.

userAttributes

하나 이상의 사용자 속성 이름 및 값 페어입니다. 각 페어는 "name": "value" 형식입니다.

응답

Lambda 트리거에서 보내는 응답입니다. 응답의 반환 파라미터는 트리거하는 이벤트에 따라 다릅니다.

사용자 풀 Lambda 트리거 소스

이 섹션에서는 각 Amazon Cognito Lambda triggerSource 파라미터와 그 트리거 이벤트를 설명합니다.

가입, 확인 및 로그인(인증) 트리거

트리거	triggerSource 값	트리거하는 이벤트
사전 가입	PreSignUp_SignUp	사전 가입.
사전 가입	PreSignUp_AdminCreateUser	관리자가 새 사용자를 만들면 사전 가입.
게시 확인	PostConfirmation_ConfirmSignUp	사후 가입 확인.
게시 확인	PostConfirmation_ConfirmForgotPassword	사후 암호 재설정 확인.
사전 인증	PreAuthentication_Authentication	사전 인증.
사후 인증	PostAuthentication_Authentication	사후 인증.

사용자 지정 인증 문제 트리거

트리거	triggerSource 값	트리거하는 이벤트
인증 문제 정의	DefineAuthChallenge_Authentication	인증 문제 정의.

트리거	triggerSource 값	트리거하는 이벤트
인증 문제 생성	CreateAuthChallenge_Authentication	인증 문제 생성.
인증 문제 확인	VerifyAuthChallengeResponse_Authentication	인증 문제 응답 확인.

사전 토큰 생성 트리거

트리거	triggerSource 값	트리거하는 이벤트
사전 토큰 생성	TokenGeneration_HostedAuthenticator	인증하는 동안 Amazon Cognito 호스팅 UI 로그인 페이지에서 호출됩니다.
사전 토큰 생성	TokenGeneration_Authentication	사용자 인증 흐름이 완료된 이후 호출됩니다.
사전 토큰 생성	TokenGeneration_NewPasswordChallenge	사용자가 관리자에 의해 생성된 후 호출됩니다. 이 흐름은 사용자가 임시 비밀번호를 변경해야 할 때 호출됩니다.
사전 토큰 생성	TokenGeneration_AuthenticationDevice	사용자 다중 인증 마지막에 호출됩니다.
사전 토큰 생성	TokenGeneration_RefreshToken	사용자가 자격 증명 및 액세스 토큰을 새로 고치려 할 때 호출됩니다.

사용자 트리거 마이그레이션

트리거	triggerSource 값	트리거하는 이벤트
사용자 마이그레이션	UserMigration_Authentication	로그인 시 사용자 마이그레이션.
사용자 마이그레이션	UserMigration_ForgotPassword	암호 찾기 흐름 도중 사용자 마이그레이션.

사용자 지정 메시지 트리거

트리거	triggerSource 값	트리거하는 이벤트
사용자 지정 메시지	CustomMessage_SignUp	사용자 지정 메시지 - 가입 후 확인 코드 전송.
사용자 지정 메시지	CustomMessage_AdminCreateUser	사용자 지정 메시지 - 새 사용자에게 임시 암호 전송.
사용자 지정 메시지	CustomMessage_ResendCode	사용자 지정 메시지 - 기존 사용자에게 확인 코드 재전송.
사용자 지정 메시지	CustomMessage_ForgotPassword	사용자 지정 메시지 - 암호 분실 요청을 위한 확인 코드 전송.
사용자 지정 메시지	CustomMessage_UpdateUserAttributes	사용자 지정 메시지 - 사용자의 이메일 또는 전화 번호가 변경되면 이 트리거가 사용자에게 자동으로

트리거	triggerSource 값	트리거하는 이벤트
		확인 코드를 전송합니다. 다른 속성에는 사용할 수 없습니다.
사용자 지정 메시지	CustomMessage_VerifyUserAttributes	사용자 지정 메시지 – 사용자가 새 이메일 또는 전화 번호에 대해 수동으로 요청하면 이 트리거가 사용자에게 확인 코드를 전송합니다.
사용자 지정 메시지	CustomMessage_Authentication	사용자 지정 메시지 – 인증하는 동안 MFA 코드 전송.

사전 가입 Lambda 트리거

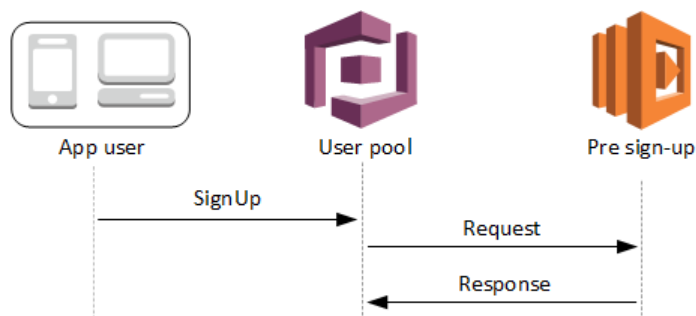
사전 가입 Lambda 함수는 Amazon Cognito가 새 사용자를 가입시키기 직전에 시작됩니다. 사용자 지정 확인을 수행하여 가입 프로세스의 일부로 등록 요청을 수락 또는 거부할 수 있습니다.

주제

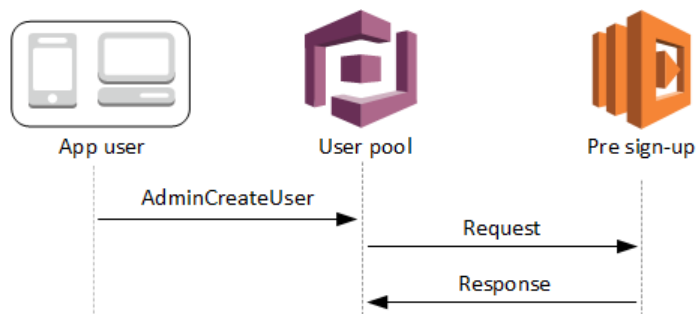
- [사전 가입 Lambda 흐름 \(p. 123\)](#)
- [사전 가입 Lambda 트리거 파라미터 \(p. 124\)](#)
- [가입 자습서 \(p. 125\)](#)
- [사전 가입 예제: 등록된 도메인의 사용자 자동 컨펌 \(p. 125\)](#)
- [사전 가입 예제: 모든 사용자 자동 컨펌 및 자동 확인 \(p. 127\)](#)

사전 가입 Lambda 흐름

클라이언트 가입 흐름



서버 가입 흐름



요청에는 사용자 풀 SignUp 및 AdminCreateUser API 메서드로 전달되는 `validationData` 값에 있는 클라이언트의 확인 데이터가 포함됩니다.

사전 가입 Lambda 트리거 파라미터

이것들은 [공통 파라미터](#) 외에 이 Lambda 함수에 필요한 파라미터입니다.

JSON

```
{
  "request": {
    "userAttributes": {
      "string": "string",
      ....
    },
    "validationData": [
      {
        "Name": "string",
        "Value": "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`

사용자 등록 요청에 확인 데이터를 포함하는 하나 이상의 이름-값 페어입니다. 사용자 등록 요청에서 확인 데이터가 설정되고 클라이언트로부터 전달됩니다.

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 전화번호를 자동으로 확인할 수 있습니다.

`autoConfirmUser`

사용자를 자동으로 확인하려면 `true`로 설정하고 그렇지 않으면 `false`로 설정합니다.

`autoVerifyEmail`

가입 중인 사용자의 이메일을 확인한 것으로 설정하려면 `true`로 설정하고 그렇지 않으면 `false`로 설정합니다. `autoVerifyEmail`이 `true`로 설정되면 `email` 속성에 `null`이 아닌 유효한 값이 있어야 합니다. 그렇지 않으면 오류가 발생하고 사용자가 가입을 완료할 수 없습니다.

`email` 속성이 별칭으로 선택될 경우 `autoVerifyEmail`이 설정되면 사용자 이메일의 별칭이 생성됩니다. 해당 이메일의 별칭이 이미 있으면 별칭이 새 사용자로 이동하고 이전 사용자의 이메일은 확인되지 않은 것으로 표시됩니다. 자세한 내용은 [별칭 개요 \(p. 204\)](#) 단원을 참조하십시오.

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 `null`이 아닌 유효한 값이 있어야 합니다. 그렇지 않으면 오류가 발생하고 사용자가 가입을 완료할 수 없습니다.

`phone_number` 속성이 별칭으로 선택될 경우 `autoVerifyPhone`이 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 [별칭 개요 \(p. 204\)](#) 단원을 참조하십시오.

가입 자습서

사전 가입 Lambda 함수는 사용자 가입 전에 트리거됩니다. JavaScript, Android 및 iOS용 가입 자습서를 참조하십시오.

플랫폼	자습서
JavaScript 자격 증명 SDK	JavaScript 사용자 가입
Android 자격 증명 SDK	Android 사용자 가입
iOS 자격 증명 SDK	iOS 사용자 가입

사전 가입 예제: 등록된 도메인의 사용자 자동 컨펌

사전 가입 Lambda 트리거를 통해 사용자 지정 로직을 추가하여 사용자 풀에 가입 중인 신규 사용자를 확인할 수 있습니다. 이것은 신규 사용자의 가입 방법을 보여주는 JavaScript 프로그램의 예제입니다. 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. 그러면 이 함수에서 응답에 변경 사항을 담아 동일한 이벤트 객체를 Amazon Cognito로 돌려보냅니다. 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 컨펌 및 자동 확인

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. 사용자 풀 [ListUsers API](#)를 사용하여 기존 사용자가 신규 사용자의 전화 번호나 이메일 주소를 이미 별칭으로 사용하고 있는지 확인하여 이를 방지할 수 있습니다.

Node.js

```
exports.handler = (event, context, callback) => {

    // 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 to Amazon Cognito
    callback(null, event);
};
```

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. 그러면 이 함수에서 응답에 변경 사항을 담아 동일한 이벤트 객체를 Amazon Cognito로 돌려보냅니다. 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": {}
}
```

사후 확인 Lambda 트리거

새로운 사용자가 확인된 후 Amazon Cognito가 이 트리거를 호출하여 사용자 지정 메시지를 보내거나 사용자 지정 로직을 추가할 수 있습니다. 예를 들어, 이 트리거를 사용하여 새로운 사용자 데이터를 수집할 수 있습니다.

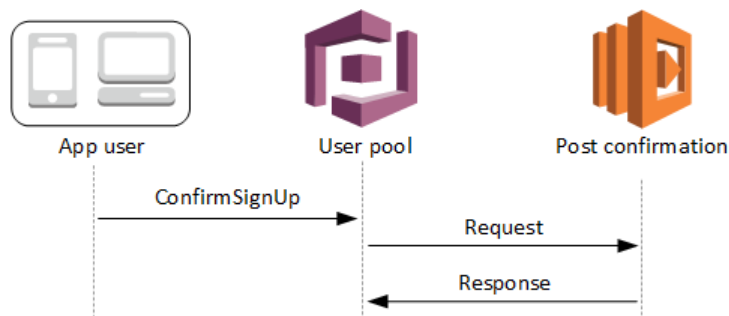
요청에는 확인된 사용자의 현재 속성이 포함됩니다.

주제

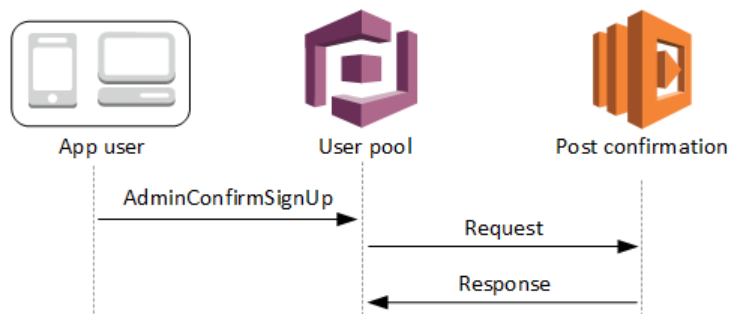
- [게시 확인 Lambda 흐름](#) (p. 128)
- [사후 확인 Lambda 트리거 파라미터](#) (p. 129)
- [사용자 확인 자습서](#) (p. 129)
- [사후 확인 예제](#) (p. 130)

게시 확인 Lambda 흐름

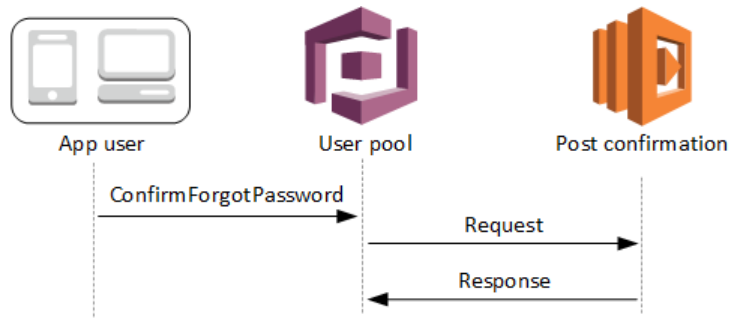
클라이언트 확인 가입 흐름



서버 확인 가입 흐름



암호 분실 확인 흐름



사후 확인 Lambda 트리거 파라미터

이것들은 **공통 파라미터** 외에 이 Lambda 함수에 필요한 파라미터입니다.

JSON

```

{
  "request": {
    "userAttributes": {
      "string": "string",
      ....
    },
    "response": {}
  }
}

```

사후 확인 요청 파라미터

userAttributes

사용자 속성을 나타내는 하나 이상의 이름-값 페어입니다.

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 참조하십시오.

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 예제에서 Lambda 함수는 Amazon SES를 사용하여 사용자에게 확인 이메일 메시지를 전송합니다. 자세한 내용은 [Amazon Simple Email Service 개발자 안내서](#) 단원을 참조하십시오.

Node.js

```
var aws = require('aws-sdk');

var ses = new aws.SES();

exports.handler = (event, context, callback) => {
    console.log(event);

    if (event.request.userAttributes.email) {
        sendEmail(event.request.userAttributes.email, "Congratulations " +
            event.userName + ", you have been confirmed: ", function(status) {

                // Return to Amazon Cognito
                callback(null, event);
            });
    } else {
        // Nothing to do, the user's email ID is unknown
        callback(null, event);
    }
};

function sendEmail(to, body, completedCallback) {
    var 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>"

};

var email = ses.sendEmail(eParams, function(err, data){
    if (err) {
        console.log(err);
    } else {
        console.log("===EMAIL SENT===");
    }
    completedCallback('Email sent');
});
console.log("EMAIL CODE END");
};
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그러면 이 함수에서 응답에 변경 사항을 담아 동일한 이벤트 객체를 Amazon Cognito로 돌려보냅니다. 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

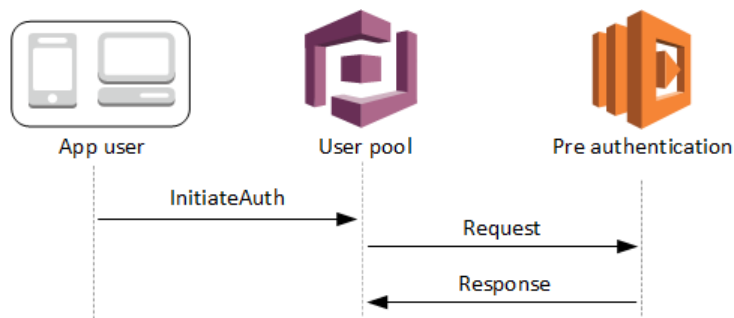
사용자가 가입을 시도하면 Amazon Cognito가 이 트리거를 호출하여 사용자 지정 검증을 통해 인증 요청을 수락 또는 거부할 수 있습니다.

주제

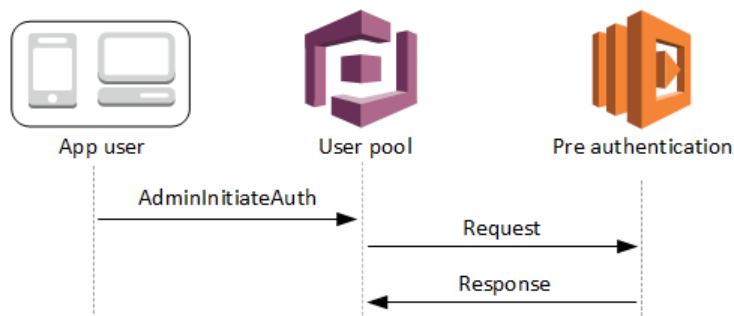
- [사전 인증 Lambda 흐름 \(p. 131\)](#)
- [사전 인증 Lambda 트리거 파라미터 \(p. 132\)](#)
- [인증 자습서 \(p. 132\)](#)
- [사전 인증 예제 \(p. 132\)](#)

사전 인증 Lambda 흐름

클라이언트 인증 흐름



서버 인증 흐름



요청에는 사용자 풀 InitiateAuth 및 AdminInitiateAuth API 메서드로 전달되는 ClientMetadata 값에 있는 클라이언트의 확인 데이터가 포함됩니다.

자세한 내용은 [사용자 풀 인증 흐름 \(p. 187\)](#) 단원을 참조하십시오.

사전 인증 Lambda 트리거 파라미터

이것들은 [공통 파라미터](#) 외에 이 Lambda 함수에 필요한 파라미터입니다.

JSON

```
{
  "request": {
    "userAttributes": {
      "string": "string",
      ....
    }
    "validationData": {
      "string": "string",
      "string": "string",
      ....
    }
  },
  "response": {}
}
```

사전 인증 요청 파라미터

userAttributes

사용자 속성을 나타내는 하나 이상의 이름-값 페어입니다.

validationData

사용자 로그인 요청에 확인 데이터를 포함하는 하나 이상의 키-값 페어입니다.

사전 인증 응답 파라미터

응답에는 추가적인 반환 정보가 없습니다.

인증 자습서

사전 인증 Lambda 함수는 Amazon Cognito가 새 사용자를 로그인시키기 직전에 시작됩니다. JavaScript, Android 및 iOS용 로그인 자습서를 참조하십시오.

플랫폼	자습서
JavaScript 자격 증명 SDK	JavaScript 사용자 로그인
Android 자격 증명 SDK	Android 사용자 로그인
iOS 자격 증명 SDK	iOS 사용자 로그인

사전 인증 예제

이 샘플 함수는 특정 사용자 풀 앱 클라이언트의 사용자가 사용자 풀에 로그인하지 못하도록 합니다.

Node.js

```
exports.handler = (event, context, callback) => {
  if (event.callerContext.clientId === "user-pool-app-client-id-to-be-blocked") {
    var error = new Error("Cannot authenticate users from this user pool app client");

    // Return error to Amazon Cognito
    callback(error, event);
  }

  // Return to Amazon Cognito
  callback(null, event);
};
```

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. 그러면 이 함수에서 응답에 변경 사항을 담아 동일한 이벤트 객체를 Amazon Cognito로 돌려보냅니다. 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

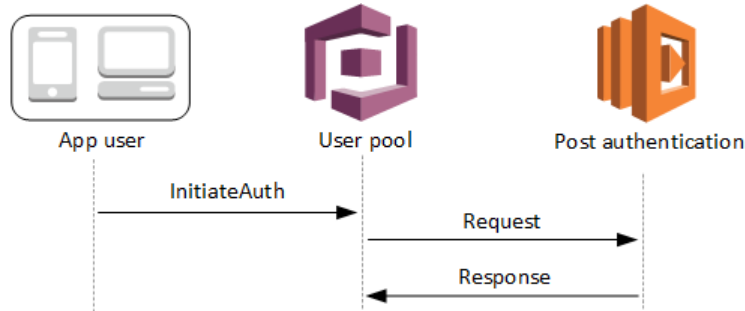
사용자 로그인 후 Amazon Cognito가 이 트리거를 호출하여 인증 후 사용자 지정 로직을 추가할 수 있습니다.

주제

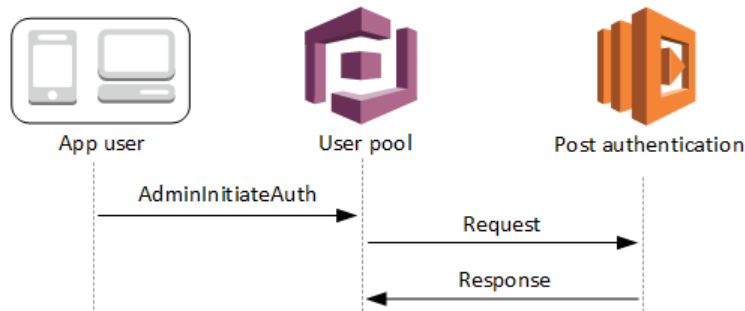
- [사후 인증 Lambda 흐름 \(p. 134\)](#)
- [사후 인증 Lambda 트리거 파라미터 \(p. 134\)](#)
- [인증 자습서 \(p. 135\)](#)
- [사후 인증 예제 \(p. 135\)](#)

사후 인증 Lambda 흐름

클라이언트 인증 흐름



서버 인증 흐름



자세한 내용은 [사용자 풀 인증 흐름 \(p. 187\)](#) 단원을 참조하십시오.

사후 인증 Lambda 트리거 파라미터

이것들은 [공통 파라미터](#) 외에 이 Lambda 함수에 필요한 파라미터입니다.

JSON

```
{
  "request": {
    "userAttributes": {
      "string": "string",
      ....
    },
    "newDeviceUsed": boolean
  },
  "response": {}
}
```

사후 인증 요청 파라미터

newDeviceUsed

이 플래그는 사용자가 새로운 디바이스에 로그인했는지 여부를 표시합니다. 이것은 기억된 사용자 풀의 디바이스 값이 Always 또는 User Opt-In으로 설정된 경우에만 설정됩니다.

userAttributes

사용자 속성을 나타내는 하나 이상의 이름-값 페어입니다.

사후 인증 응답 파라미터

응답에는 추가적인 반환 정보가 없습니다.

인증 자습서

사후 인증 Lambda 함수는 Amazon Cognito가 새 사용자를 로그인한 직후에 시작됩니다. JavaScript, Android 및 iOS용 로그인 자습서를 참조하십시오.

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

```
exports.handler = (event, context, callback) => {

    // Send post authentication data to 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 to Amazon Cognito
    callback(null, event);
};
```

Python

```
from __future__ import print_function
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'])

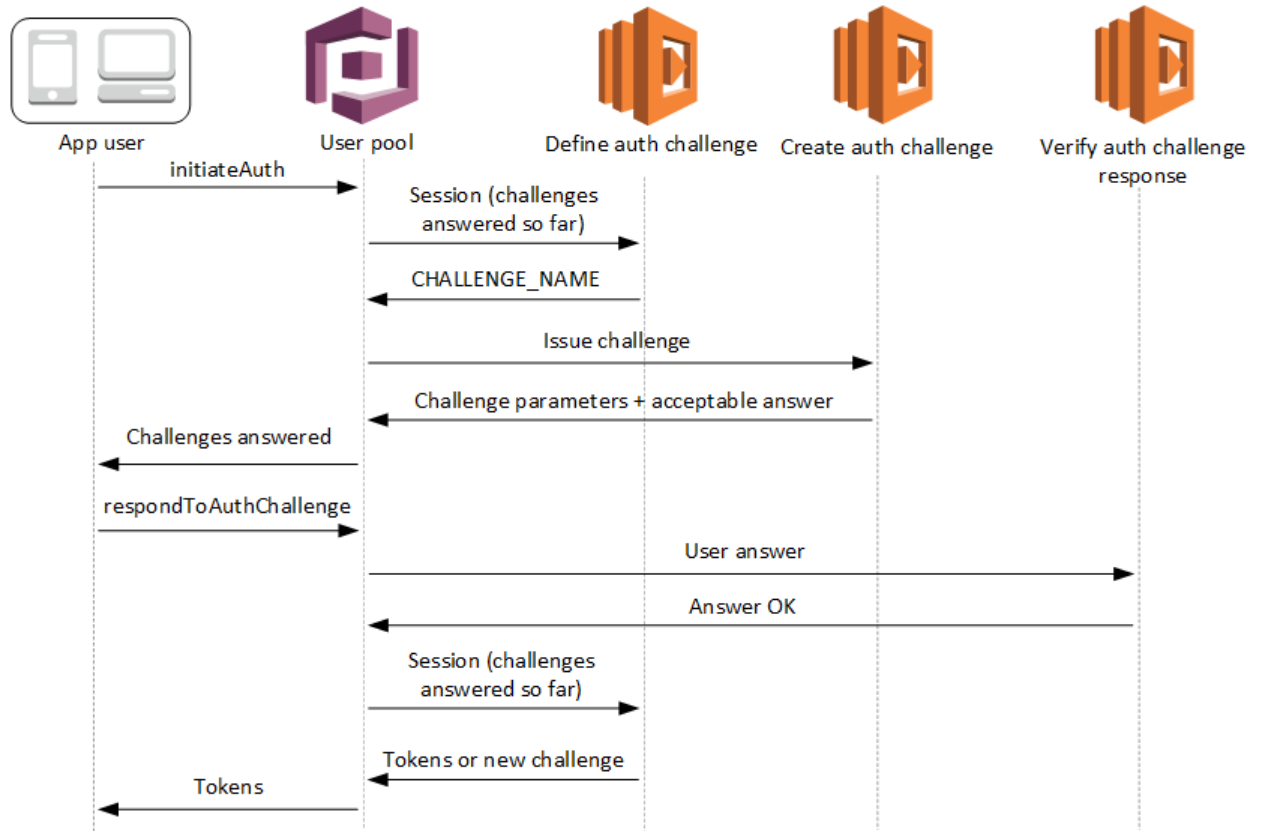
    # Return to Amazon Cognito
    return event
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그러면 이 함수에서 응답에 변경 사항을 담아 동일한 이벤트 객체를 Amazon Cognito로 돌려보냅니다. 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 지정 인증 문제 Lambda 트리거



이러한 Lambda 트리거는 자체 문제를 사용자 풀 **사용자 인증 흐름**의 일부로 확인하고 발행합니다.

인증 문제 정의

Amazon Cognito가 이 트리거를 호출하여 사용자 지정 인증 흐름을 시작합니다.

인증 문제 생성

Amazon Cognito는 이 트리거를 인증 문제 정의의 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.

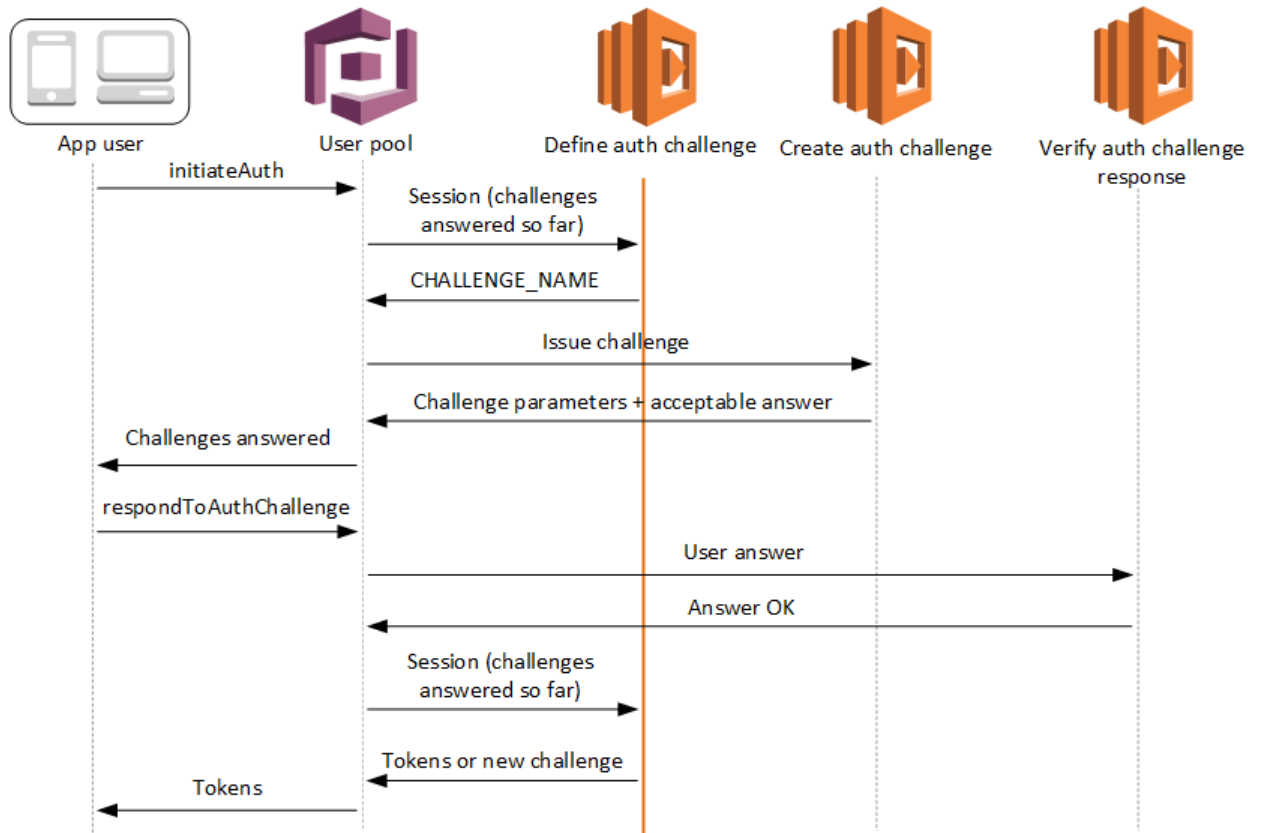
Note

Amazon Cognito 호스팅 로그인 웹 페이지는 사용자 지정 인증 흐름을 지원하지 않습니다.

주제

- 인증 문제 정의 Lambda 트리거 (p. 137)
- 인증 문제 생성 Lambda 트리거 (p. 140)
- 인증 문제 응답 확인 Lambda 트리거 (p. 142)

인증 문제 정의 Lambda 트리거



인증 문제 정의

Amazon Cognito가 이 트리거를 호출하여 사용자 지정 인증 흐름을 시작합니다.

요청에는 session이 포함되며, 이 항목은 진행 중인 인증 프로세스에서 사용자에게 표시되는 모든 챌린지와 해당하는 결과를 포함하는 어레이입니다. 챌린지 세부 정보(challengeResult)는 사용자에게 표시되는 첫 번째 챌린지를 나타내는 session과 함께 session[0] 어레이에 시간순으로 저장됩니다.

모든 챌린지에 응답할 때까지 챌린지 루프가 반복됩니다.

주제

- [인증 챌린지 정의 Lambda 트리거 파라미터 \(p. 138\)](#)
- [인증 챌린지 정의 예제 \(p. 139\)](#)

인증 챌린지 정의 Lambda 트리거 파라미터

이것들은 [공통 파라미터](#) 외에 이 Lambda 함수에 필요한 파라미터입니다.

JSON

```
{
  "request": {
    "userAttributes": {
      "string": "string",
      ....
    },
    "session": [
      ChallengeResult,
      ...
    ]
  },
  "response": {
    "challengeName": "string",
    "issueTokens": boolean,
    "failAuthentication": boolean
  }
}
```

인증 챌린지 정의 요청 파라미터

userAttributes

사용자 속성을 나타내는 하나 이상의 이름-값 페어입니다.

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 true로 설정하고 그렇지 않으면 false로 설정합니다.

challengeMetaData

사용자 지정 챌린지의 이름입니다. challengeName이 "CUSTOM_CHALLENGE"인 경우에만 사용 됩니다.

인증 문제 정의 응답 파라미터

응답에서 인증 프로세스의 다음 단계를 반환할 수 있습니다.

challengeName

다음 챌린지의 이름이 포함된 문자열입니다. 사용자에게 새로운 챌린지를 표시하려면 여기에 챌린지 이름을 지정하십시오.

issueTokens

사용자가 챌린지를 완료하여 충분히 인증되었다고 판단되면 true로 설정하고 그렇지 않으면 false로 설정합니다.

failAuthentication

현재 인증 프로세스를 종료하려면 true로 설정하고 그렇지 않으면 false로 설정합니다.

인증 챌린지 정의 예제

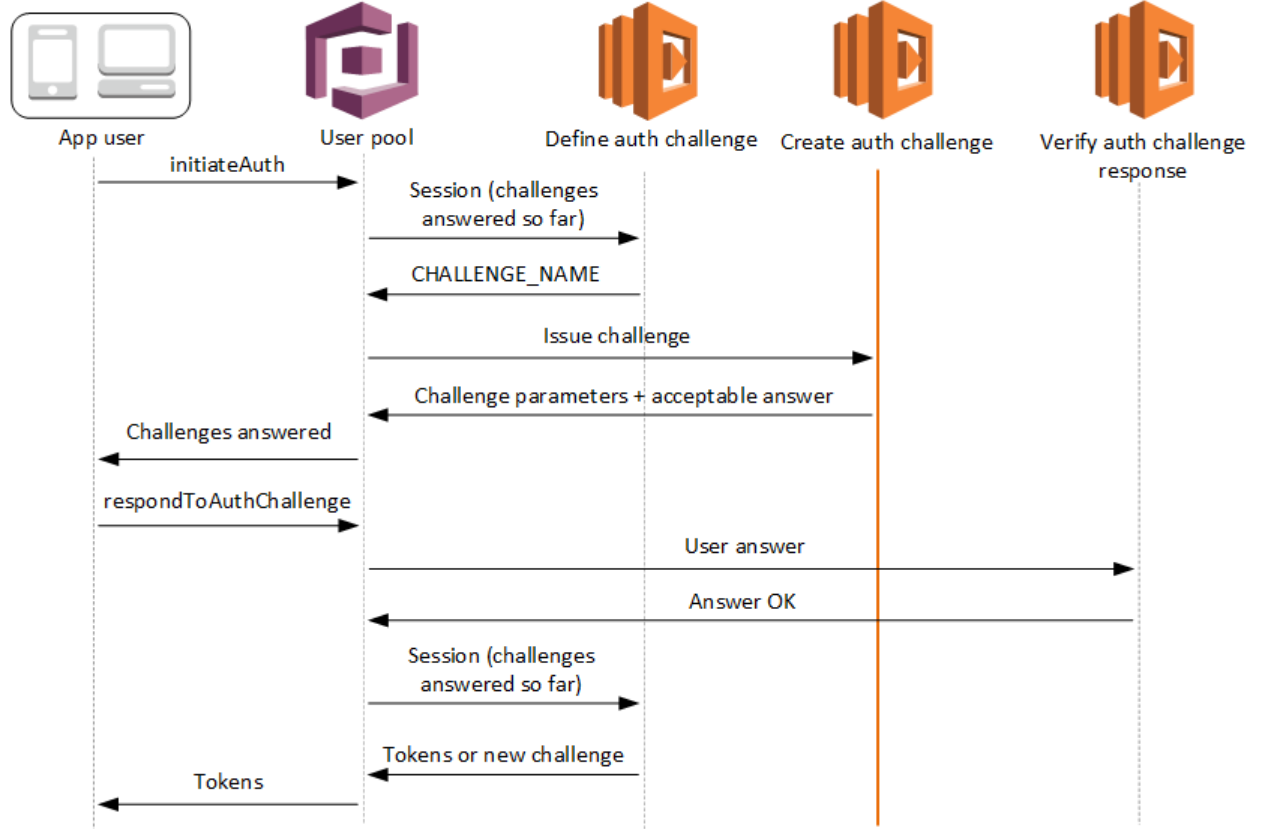
이 예제는 인증을 위한 일련의 챌린지를 정의하고 모든 챌린지가 성공적으로 완료된 경우에만 토큰을 발행합니다.

Node.js

```
exports.handler = (event, context, callback) => {
  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 = true;
    event.response.failAuthentication = false;
  } else {
    event.response.issueTokens = false;
    event.response.failAuthentication = true;
  }

  // Return to Amazon Cognito
  callback(null, event);
}
```

인증 문제 생성 Lambda 트리거



인증 문제 생성

사용자 지정 챌린지가 인증 문제 정의 트리거의 일부로 지정된 경우 Amazon Cognito가 인증 문제 정의 다음에 이 트리거를 호출합니다. [사용자 지정 인증 흐름](#)을 생성합니다.

사용자에게 표시할 챌린지를 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 챌린지 정의 트리거에서 이 속성의 값이 설정됩니다.

모든 챌린지에 응답할 때까지 챌린지 루프가 반복됩니다.

주제

- [인증 챌린지 생성 Lambda 트리거 파라미터 \(p. 140\)](#)
- [인증 챌린지 생성 예제 \(p. 142\)](#)

인증 챌린지 생성 Lambda 트리거 파라미터

이것들은 [공통 파라미터](#) 외에 이 Lambda 함수에 필요한 파라미터입니다.

JSON

```
{
  "request": {
    "userAttributes": {
```

```

        "string": "string",
        ....
    }
    "challengeName": "string",
    "session": [
        ChallengeResult,
        ...
    ]
},
"response": {
    "publicChallengeParameters": {
        "string": "string",
        ....
    },
    "privateChallengeParameters": {
        "string": "string",
        ....
    },
    "challengeMetadata": "string"
}
}

```

인증 챌린지 생성 요청 파라미터

userAttributes

사용자 속성을 나타내는 하나 이상의 이름-값 페어입니다.

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 true로 설정하고 그렇지 않으면 false로 설정합니다.

challengeMetadata

사용자 지정 챌린지의 이름입니다. challengeName이 "CUSTOM_CHALLENGE"인 경우에만 사용 됩니다.

인증 문제 생성 응답 파라미터

publicChallengeParameters

클라이언트 앱이 사용자에게 표시할 챌린지에 사용하기 위한 하나 이상의 키-값 페어입니다. 사용자에게 챌린지를 정확히 표시하는 데 필요한 모든 정보를 이 파라미터에 포함해야 합니다.

privateChallengeParameters

이 파라미터는 인증 챌린지 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 답을 포함합니다.

`challengeMetadata`

이 항목이 사용자 지정 챌린지일 경우 사용자 지정 챌린지의 이름입니다.

인증 챌린지 생성 예제

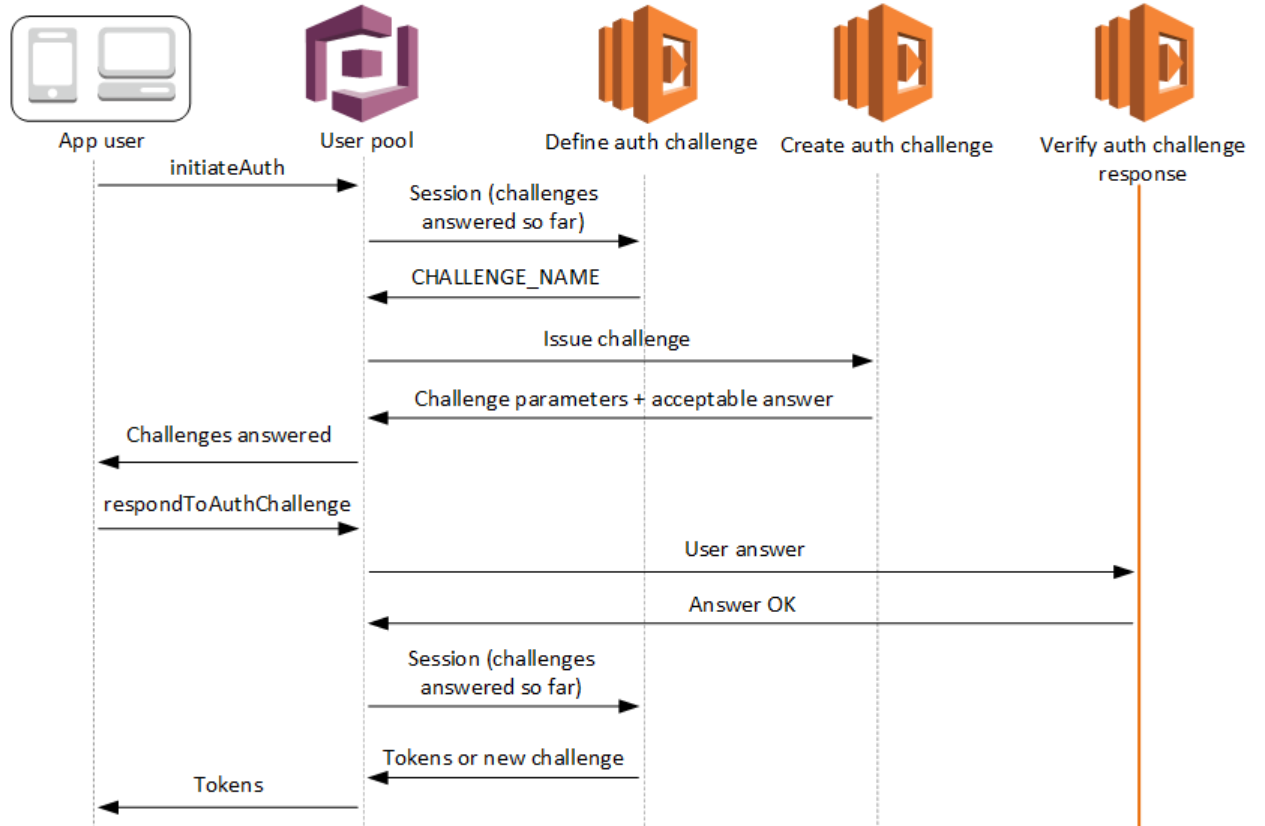
CAPTCHA가 사용자에게 대한 챌린지로 생성됩니다. CAPTCHA 이미지 URL이 퍼블릭 챌린지 파라미터에 "captchaUrl"로 추가되고 예상되는 답이 프라이빗 챌린지 파라미터에 추가됩니다.

Node.js

```
exports.handler = (event, context, callback) => {
  if (event.request.challengeName == 'CUSTOM_CHALLENGE') {
    event.response.publicChallengeParameters = {};
    event.response.publicChallengeParameters.captchaUrl = 'url/123.jpg';
    event.response.privateChallengeParameters = {};
    event.response.privateChallengeParameters.answer = '5';
    event.response.challengeMetadata = 'CAPTCHA_CHALLENGE';
  }

  Return to Amazon Cognito
  callback(null, event);
}
```

인증 문제 응답 확인 Lambda 트리거



인증 문제 응답 확인

Amazon Cognito는 이 트리거를 호출하여 사용자 지정 인증 챌린지에 대한 최종 사용자의 응답이 유효한지 여부를 확인합니다. 이것은 사용자 풀 [사용자 지정 인증 흐름](#)에 속합니다.

이 트리거에 대한 요청에는 `privateChallengeParameters` 및 `challengeAnswer` 파라미터가 포함됩니다. `privateChallengeParameters` 값은 인증 챌린지 생성 Lambda 트리거에 의해 반환되며 사용자의 예상 응답을 포함합니다. `challengeAnswer` 파라미터에는 챌린지에 대한 사용자의 응답이 포함됩니다.

응답에는 `answerCorrect` 속성이 포함되며 사용자가 챌린지를 성공적으로 완료하면 이 속성이 `true`로 설정되고 그렇지 않으면 `false`로 설정됩니다.

모든 챌린지에 응답할 때까지 챌린지 루프가 반복됩니다.

주제

- [인증 챌린지 확인 Lambda 트리거 파라미터 \(p. 143\)](#)
- [인증 챌린지 응답 확인 예제 \(p. 144\)](#)

인증 챌린지 확인 Lambda 트리거 파라미터

이것들은 [공통 파라미터](#) 외에 이 Lambda 함수에 필요한 파라미터입니다.

JSON

```
{
  "request": {
    "userAttributes": {
      "string": "string",
      ....
    },
    "privateChallengeParameters": {
      "string": "string",
      ....
    },
    "challengeAnswer": {
      "string": "string",
      ....
    }
  },
  "response": {
    "answerCorrect": boolean
  }
}
```

인증 문제 확인 요청 파라미터

userAttributes

사용자 속성을 나타내는 하나 이상의 이름-값 페어입니다.

privateChallengeParameters

이 파라미터는 '인증 문제 생성' 트리거에 속하며, 사용자의 `challengeAnswer`와 비교하여 사용자가 문제를 전달했는지 여부를 판단합니다.

이 파라미터는 인증 챌린지 응답 확인 Lambda 트리거에만 사용됩니다. 문제에 대한 사용자의 응답을 확인하는 데 필요한 모든 정보가 포함되어야 합니다. 사용자에게 제기된 질문이 있는 `publicChallengeParameters` 파라미터와 해당 문제에 대한 유효한 답이 있는 `privateChallengeParameters`가 포함되어 있습니다.

challengeAnswer

문제에 대한 사용자의 응답에 있는 답입니다.

인증 문제 확인 응답 파라미터

answerCorrect

사용자가 챌린지를 성공적으로 완료하면 true로 설정하고 그렇지 않으면 false로 설정합니다.

인증 챌린지 응답 확인 예제

이 예제에서 Lambda 함수가 챌린지에 대한 사용자의 응답이 예상 응답과 일치하는지 확인합니다. 사용자의 응답과 예상 응답이 일치하면 answerCorrect 파라미터가 true로 설정됩니다.

Node.js

```
exports.handler = (event, context, callback) => {
  if (event.request.privateChallengeParameters.answer ==
    event.request.challengeAnswer) {
    event.response.answerCorrect = true;
  } else {
    event.response.answerCorrect = false;
  }

  // Return to Amazon Cognito
  callback(null, event);
}
```

사전 토큰 생성 Lambda 트리거

토큰 생성 전에 Amazon Cognito에서 이 트리거가 호출되어 토큰 클레임을 사용자 지정할 수 있습니다.

이 Lambda 트리거를 통해 자격 증명이 생성되기 전에 사용자 정의할 수 있습니다. 이 트리거를 사용하여 자격 증명 토큰에서 새 클레임 추가, 클레임 업데이트 또는 억제가 가능합니다. 이 기능을 사용하려면 Amazon Cognito 사용자 풀 콘솔의 Lambda 함수를 연결하거나 AWS CLI를 통해 사용자 풀을 업데이트합니다.

수정할 수 없는 클레임이 있습니다. 이러한 클레임에는 acr, amr, aud, auth_time, azp, exp, iat, identities, iss, sub, token_use 및 cognito:username이 포함됩니다.

주제

- [사전 토큰 생성 Lambda 트리거 소스 \(p. 144\)](#)
- [사전 토큰 생성 Lambda 트리거 파라미터 \(p. 145\)](#)
- [사전 토큰 생성 예제: 새 클레임 추가 및 기존 클레임 억제 \(p. 146\)](#)
- [사전 토큰 생성 예제: 사용자의 그룹 멤버십 수정 \(p. 147\)](#)

사전 토큰 생성 Lambda 트리거 소스

triggerSource 값	트리거하는 이벤트
TokenGeneration_HostedAuth	인증하는 동안 Amazon Cognito 호스팅 UI 로그인 페이지에서 호출됩니다.

triggerSource 값	트리거하는 이벤트
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.
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

이것들은 [공통 파라미터](#) 외에 이 Lambda 함수에 필요한 파라미터입니다.

JSON

```
{
  "request": {
    "userAttributes": {
      "string": "string",
      ....
    }
    "groupConfiguration": {
      "groupsToOverride": ["string", ....],
      "iamRolesToOverride": ["string", ....],
      "preferredRole": "string"
    },
    "response": {
      "claimsOverrideDetails": {
        "claimsToAddOrOverride": {
          "string": "string",
          ....
        },
        "claimsToSuppress": ["string", ....],
        "groupOverrideDetails": {
          "groupsToOverride": ["string", ....],
          "iamRolesToOverride": ["string", ....],
          "preferredRole": "string"
        }
      }
    }
  }
}
```

사전 토큰 생성 요청 파라미터

groupConfiguration

현재 그룹 구성을 포함하는 입력 객체입니다. groupsToOverride, iamRolesToOverride 및 preferredRole을 포함합니다.

groupsToOverride

자격 증명 토큰이 발행된 사용자와 연결되어 있는 그룹 이름 목록입니다.

iamRolesToOverride

이러한 그룹과 연결된 현재 IAM 역할 목록입니다.

preferredRole

선호하는 IAM 역할을 나타내는 문자열입니다.

사전 토큰 생성 응답 파라미터

claimsToAddOrOverride

추가하거나 재정의하려는 하나 이상의 키-값 클레임 페어의 맵입니다. 클레임과 관련된 그룹인 경우 groupOverrideDetails를 대신 사용합니다.

claimsToSuppress

자격 증명 토큰의 억제될 클레임을 포함하는 목록입니다.

Note

값이 모두 억제되고 대체된 경우 값은 억제됩니다.

groupOverrideDetails

현재 그룹 구성을 포함하는 출력 객체입니다. groupsToOverride, iamRolesToOverride 및 preferredRole을 포함합니다.

groupOverrideDetails 객체는 사용자가 제공하는 구성으로 대체됩니다. 응답에서 빈 객체나 널 객체를 제공하면 그룹이 억제됩니다. 기존 그룹 구성을 그대로 두려면 요청의 groupConfiguration 객체 값을 대응하는 groupOverrideDetails 객체로 복사한 다음 서비스로 다시 전달합니다.

사전 토큰 생성 예제: 새 클레임 추가 및 기존 클레임 억제

이 예제에서는 사전 토큰 생성 Lambda를 사용하여 새 클레임을 추가하고 기존 클레임을 억제합니다.

Node.js

```
exports.handler = (event, context, callback) => {
  event.response = {
    "claimsOverrideDetails": {
      "claimsToAddOrOverride": {
        "attribute_key2": "attribute_value2",
        "attribute_key": "attribute_value"
      },
      "claimsToSuppress": ["email"]
    }
  };

  // Return to Amazon Cognito
  callback(null, event);
};
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그러면 이 함수에서 응답에 변경 사항을 담아 동일한 이벤트 객체를 Amazon Cognito로 돌려보냅니다. Lambda 콘솔에서 해당 Lambda 트리거와 관련 있는 데이터로 테스트 이벤트를 설정할 수 있습니다. 다음은 이 코드 샘플의 테스트 이벤트입니다.

JSON

```
{
  "request": {},
  "response": {}
}
```

```
}
```

사전 토큰 생성 예제: 사용자의 그룹 멤버십 수정

이 예제에서는 사전 토큰 생성 Lambda를 사용하여 사용자 그룹 멤버십을 수정합니다.

Node.js

```
exports.handler = (event, context, callback) => {
  event.response = {
    "claimsOverrideDetails": {
      "claimsToAddOrOverride": {
        "attribute_key2": "attribute_value2",
        "attribute_key": "attribute_value"
      },
      "claimsToSuppress": ["email"],
      "groupOverrideDetails": {
        "groupsToOverride": ["group-A", "group-B", "group-C"],
        "iamRolesToOverride": ["arn:aws:iam::XXXXXXXXXXXX:role/sns_callerA",
          "arn:aws:iam::XXXXXXXXXXXX:role/sns_callerB", "arn:aws:iam::XXXXXXXXXXXX:role/sns_callerC"],
        "preferredRole": "arn:aws:iam::XXXXXXXXXXXX:role/sns_caller"
      }
    }
  };

  // Return to Amazon Cognito
  callback(null, event);
};
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그러면 이 함수에서 응답에 변경 사항을 담아 동일한 이벤트 객체를 Amazon Cognito로 돌려보냅니다. 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

Amazon Cognito는 비밀번호로 로그인할 때 사용자가 사용자 풀에 존재하지 않거나 암호 찾기 흐름 시 이 트리거를 호출합니다. Lambda 함수가 성공적으로 반환된 이후 Amazon Cognito는 사용자를 사용자 풀에 추가합니다. 사용자 마이그레이션 Lambda 트리거를 사용한 인증 흐름에 대한 자세한 내용은 [사용자 마이그레이션 Lambda 트리거를 사용하여 사용자 풀로 사용자 가져오기 \(p. 171\)](#) 단원을 참조하십시오.

로그인 시 또는 Lambda 트리거를 사용한 암호 찾기 흐름 도중 기존 사용자 디렉터리의 사용자를 Amazon Cognito 사용자 풀로 마이그레이션할 수 있습니다.

주제

- [사용자 마이그레이션 Lambda 트리거 소스 \(p. 148\)](#)
- [사용자 마이그레이션 Lambda 트리거 파라미터 \(p. 148\)](#)
- [예제: 기존 암호를 사용한 사용자 마이그레이션 \(p. 149\)](#)

사용자 마이그레이션 Lambda 트리거 소스

triggerSource 값	트리거하는 이벤트
UserMigration_Authentication	로그인 시 사용자 마이그레이션.
UserMigration_ForgotPassword	암호 찾기 흐름 도중 사용자 마이그레이션.

사용자 마이그레이션 Lambda 트리거 파라미터

이것들은 **공통 파라미터** 외에 이 Lambda 함수에 필요한 파라미터입니다.

JSON

```
{
  "userName": "string",
  "request": {
    "password": "string"
  },
  "response": {
    "userAttributes": {
      "string": "string",
      ...
    },
    "finalUserStatus": "string",
    "messageAction": "string",
    "desiredDeliveryMediums": [ "string", ... ],
    "forceAliasCreation": boolean
  }
}
```

사용자 마이그레이션 요청 파라미터

사용자 이름

사용자가 입력한 사용자 이름.

암호

사용자가 로그인 시 입력한 암호. 암호 찾기 흐름에서는 설정되지 않습니다.

사용자 마이그레이션 응답 파라미터

userAttributes

이 필드는 필수입니다.

사용자 풀의 사용자 프로필에 저장된 사용자 속성을 나타내는 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 속성](#)을 참조하십시오.

Note

암호 찾기 흐름에서 사용자가 암호를 재설정하려면 확인된 이메일이나 확인된 전화 번호가 있어야 합니다. Amazon Cognito에서 사용자 속성에 있는 이메일이나 전화 번호로 암호 재설정 코드가 포함된 메시지를 보냅니다.

속성	요구 사항
사용자 풀을 만들 때 필수로 표시한 모든 속성	마이그레이션 중에 필수 속성이 누락된 경우 기본값이 사용됩니다.
username	로그인용 사용자 이름 외에 이메일 및/또는 preferred_username 별칭으로 사용자 풀을 구성하고, 사용자가 이메일 또는 전화번호를 입력하여 로그인할 때 필요합니다. 그렇지 않은 경우 이는 선택 사항이고 사용자가 입력한 사용자 이름 대신 사용자 이름으로 사용됩니다. Note 사용자 이름은 사용자 풀에서 고유해야 합니다.
cognito:mfa_enabled	사용자 풀에서 MFA가 선택 사항으로 구성되는 경우 필요합니다. 이 속성은 사용자에게 대해 MFA가 활성화되었는지 여부를 지정합니다.

finalUserStatus

로그인 도중 이 속성은 CONFIRMED로 설정되거나 설정되지 않아 사용자를 자동 확인하고 이전 암호를 사용한 로그인을 허용합니다. 이는 사용자에게 가장 간편한 환경입니다.

이 속성이 RESET_REQUIRED로 설정된 경우 사용자는 로그인 시 마이그레이션 직후 암호를 변경해야 하고, 클라이언트 앱은 인증 흐름 동안 PasswordResetRequiredException을 처리해야 합니다.

Note

새 사용자 풀에 대한 암호 정책은 기존 사용자 디렉터리의 암호 정책보다 강력하면 안 됩니다.

messageAction

이 속성은 "SUPPRESS"로 설정되어 Amazon Cognito가 새 사용자에게 주로 보내는 환영 인사 메시지를 억제할 수 있습니다. 이 속성이 반환되지 않는 경우 환영 인사 메시지가 전송됩니다.

desiredDeliveryMediums

이 속성은 "EMAIL"로 설정되어 환영 인사 메시지를 이메일로 전송하거나 "SMS"로 설정되어 환영 인사 메시지를 SMS로 전송할 수 있습니다. 이 속성이 반환되지 않는 경우 환영 인사 메시지가 SMS로 전송됩니다.

forceAliasCreation

이 파라미터가 "true"로 설정되고 UserAttributes 파라미터에 지정된 전화 번호 또는 이메일 주소가 다른 사용자의 별칭으로 이미 존재하는 경우 API 호출은 이전 사용자에서 새로 생성된 사용자로 별칭을 마이그레이션합니다. 이전 사용자는 더 이상 해당 별칭을 사용하여 로그인할 수 없습니다.

이 속성이 "false"로 설정되고 별칭이 존재하는 경우 사용자는 마이그레이션되지 않고 클라이언트 앱으로 오류가 반환됩니다.

이 속성이 반환되지 않는 경우 "false"로 간주됩니다.

예제: 기존 암호를 사용한 사용자 마이그레이션

이 예제 Lambda 함수는 기존 암호를 가진 사용자를 마이그레이션하고 Amazon Cognito의 환영 인사 메시지를 억제합니다.

Node.js

```
exports.handler = (event, context, callback) => {
```

```
var user;

if ( event.triggerSource == "UserMigration_Authentication" ) {

    // authenticate the user with your existing user directory service
    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";
        context.succeed(event);
    }
    else {
        // Return error to Amazon Cognito
        callback("Bad password");
    }
}
else if ( event.triggerSource == "UserMigration_ForgotPassword" ) {

    // Lookup the user in your existing user directory service
    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";
        context.succeed(event);
    }
    else {
        // Return error to Amazon Cognito
        callback("Bad password");
    }
}
else {
    // Return error to Amazon Cognito
    callback("Bad triggerSource " + event.triggerSource);
}
};
```

사용자 지정 메시지 Lambda 트리거

이메일이나 전화 확인 메시지 또는 멀티 팩터 인증(MFA) 코드를 보내기 전에 Amazon Cognito가 이 트리거를 호출하여 메시지를 동적으로 사용자 지정할 수 있습니다. 정적 사용자 지정 메시지는 [Amazon Cognito 콘솔](#)의 메시지 사용자 지정 탭에서 편집할 수 있습니다.

요청에는 사용자에게 전달되는 코드의 자리 표시자 역할을 하는 문자열인 `codeParameter`가 포함됩니다. 메시지 본문의 확인 코드 삽입 위치에 `codeParameter` 문자열을 삽입합니다. 이 응답을 받으면 Amazon Cognito 서비스가 `codeParameter` 문자열을 실제 확인 코드로 바꿉니다.

Note

사용자 지정 메시지 Lambda 함수와 `CustomMessage_AdminCreateUser` 트리거는 사용자 이름 및 확인 코드를 반환하므로 요청에 `request.usernameParameter` 및 `request.codeParameter`가 둘 다 포함되어야 합니다.

주제

- 사용자 지정 메시지 Lambda 트리거 소스 (p. 151)
- 사용자 지정 메시지 Lambda 트리거 파라미터 (p. 151)
- 가입용 사용자 지정 메시지 예 (p. 152)
- 사용자 생성 관리용 사용자 지정 메시지 예 (p. 153)

사용자 지정 메시지 Lambda 트리거 소스

triggerSource 값	트리거하는 이벤트
CustomMessage_SignUp	사용자 지정 메시지 – 가입 후 확인 코드 전송.
CustomMessage_AdminCreateUser	사용자 지정 메시지 – 새 사용자에게 임시 암호 전송.
CustomMessage_ResendCode	사용자 지정 메시지 – 기존 사용자에게 확인 코드 재 전송.
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 지정 메시지 Lambda 트리거 파라미터

이것들은 **공통 파라미터** 외에 이 Lambda 함수에 필요한 파라미터입니다.

JSON

```
{
  "request": {
    "userAttributes": {
      "string": "string",
      ....
    },
    "codeParameter": "####",
    "usernameParameter": "string"
  },
  "response": {
    "smsMessage": "string",
    "emailMessage": "string",
    "emailSubject": "string"
  }
}
```

사용자 지정 메시지 요청 파라미터

userAttributes

사용자 속성을 나타내는 하나 이상의 이름-값 페어입니다.

codeParameter

사용자 지정 메시지에서 확인 코드의 자리 표시자로 사용할 문자열입니다.

사용자 이름

사용자 이름 파라미터입니다. 사용자 생성 관리 흐름을 위한 필수 요청 파라미터입니다.

사용자 지정 메시지 응답 파라미터

응답에서는 사용자에게 보낼 메시지에서 사용할 사용자 지정 텍스트를 지정합니다.

smsMessage

사용자에게 보낼 사용자 지정 SMS 메시지입니다. 수신된 `codeParameter` 값을 요청에 포함해야 합니다.

emailMessage

사용자에게 보낼 사용자 지정 이메일 메시지입니다. 수신된 `codeParameter` 값을 요청에 포함해야 합니다.

emailSubject

사용자 지정 메시지의 제목 줄입니다.

가입용 사용자 지정 메시지 예

이 Lambda 함수는 서비스에서 확인 코드를 사용자에게 보내는 데 앱이 필요할 때 이메일이나 SMS 메시지를 사용자 지정하기 위해 호출됩니다.

Lambda 트리거는 사후 등록, 확인 코드 재전송, 암호 분실 또는 사용자 속성 확인 등 여러 지점에서 호출할 수 있습니다. SMS와 이메일 모두의 메시지가 응답에 포함됩니다. 사용자에게 전달되는 확인 코드의 자리 표시자인 코드 파라미터 "####"이 메시지에 포함되어야 합니다.

이메일의 메시지 최대 길이는 확인 코드를 포함하여 UTF-8 문자 20,000개입니다. 이 이메일에 HTML 태그를 사용할 수 있습니다.

SMS의 최대 길이는 확인 코드를 포함하여 UTF-8 문자 140개입니다.

Node.js

```
exports.handler = (event, context, callback) => {  
  //  
  if(event.userPoolId === "theSpecialUserPool") {  
    // Identify why was this function invoked  
    if(event.triggerSource === "CustomMessage_SignUp") {  
      // Ensure that your message contains event.request.codeParameter. This is  
      the placeholder for code that will be sent  
      event.response.smsMessage = "Welcome to the service. Your confirmation code  
is " + event.request.codeParameter;  
      event.response.emailSubject = "Welcome to the service";  
      event.response.emailMessage = "Thank you for signing up. " +  
event.request.codeParameter + " is your verification code";  
    }  
  }  
}
```

```
    }  
    // Create custom message for other events  
  }  
  // Customize messages for other user pools  
  
  // Return to Amazon Cognito  
  callback(null, event);  
};
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그러면 이 함수에서 응답에 변경 사항을 담아 동일한 이벤트 객체를 Amazon Cognito로 돌려보냅니다. 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>"  
  }  
}
```

사용자 생성 관리용 사용자 지정 메시지 예

사용자 지정 메시지 Lambda 함수와 CustomMessage_AdminCreateUser 트리거는 사용자 이름 및 확인 코드를 반환하므로 메시지 본문에 request.usernameParameter 및 request.codeParameter를 둘 다 포함해야 합니다.

코드 파라미터 값 "####"은 임시 암호용 자리 표시자이며 "username"은 사용자 이름으로 전달되는 사용자 이름의 자리 표시자입니다.

이메일의 메시지 최대 길이는 확인 코드를 포함하여 UTF-8 문자 20,000개입니다. 이 이메일에 HTML 태그를 사용할 수 있습니다. SMS의 최대 길이는 확인 코드를 포함하여 UTF-8 문자 140개입니다.

SMS와 이메일 모두의 메시지가 응답에 포함됩니다.

Node.js

```
exports.handler = (event, context, callback) => {
```



```
//
if(event.userPoolId === "theSpecialUserPool") {
  // Identify why was this function invoked
  if(event.triggerSource === "CustomMessage_AdminCreateUser") {
    // Ensure that your message contains event.request.codeParameter
    event.request.usernameParameter. This is the placeholder for the code and username
    that will be sent to your user.
    event.response.smsMessage = "Welcome to the service. Your user
    name is " + event.request.usernameParameter + " Your temporary password is " +
    event.request.codeParameter;
    event.response.emailSubject = "Welcome to the service";
    event.response.emailMessage = "Welcome to the service. Your user
    name is " + event.request.usernameParameter + " Your temporary password is " +
    event.request.codeParameter;
  }
  // Create custom message for other events
}
// Customize messages for other user pools

// Return to Amazon Cognito
callback(null, event);
};
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그러면 이 함수에서 응답에 변경 사항을 담아 동일한 이벤트 객체를 Amazon Cognito로 돌려보냅니다. 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>"
  }
}
```

Amazon Cognito 사용자 풀을 통해 Amazon Pinpoint 분석 사용

Amazon Cognito User Pools는 Amazon Pinpoint와 통합되어 Amazon Cognito 사용자 풀에 대해 분석을 제공하고 Amazon Pinpoint 캠페인에 대한 사용자 데이터를 보강합니다. Amazon Pinpoint에서 분석 및 대상 캠페인을 제공하여 푸시 알림을 사용하는 모바일 앱에서 사용자 참여를 유도할 수 있습니다. 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 로컬 및 앱 버전과 같은 다양한 날짜 범위 또는 속성에 대한 데이터를 자세히 확인할 수 있습니다.

또한 Android용 AWS Mobile SDK 또는 AWS Mobile SDK for iOS를 사용하여 앱에 고유한 사용자 속성을 설정할 수 있습니다. 그런 다음 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 선택하는 경우 사용자 이메일 주소 및 전화 번호에 대한 추가 엔드포인트가 생성됩니다.

Amazon Pinpoint 분석 설정 지정(AWS Management 콘솔)

분석 설정을 지정하려면

1. [Amazon Cognito 콘솔](#)에 로그인합니다.
2. 탐색 창에서 사용자 풀 관리를 선택한 다음 편집할 사용자 풀을 선택합니다.
3. Analytics(분석) 탭을 선택합니다.
4. Add analytics and campaigns(분석 및 캠페인 추가)를 선택합니다.
5. 목록에서 Cognito app client(Cognito 앱 클라이언트)를 선택합니다.
6. Amazon Cognito 앱을 Amazon Pinpoint 프로젝트에 매핑하려면 목록에서 Amazon Pinpoint 프로젝트를 선택합니다.

Note

Amazon Pinpoint 프로젝트 ID는 Amazon Pinpoint 프로젝트에 대해 고유한 32자 문자열입니다. 콘솔에 나열됩니다.

여러 Amazon Cognito 앱을 단일 Amazon Pinpoint 프로젝트에 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.

7. Amazon Cognito에서 사용자에 대한 추가 엔드포인트를 생성하기 위해 메일 주소 및 전화 번호를 Amazon Pinpoint에 전송하려면 Share user attribute data with Amazon Pinpoint(Amazon Pinpoint와 사용자 속성 데이터 공유)를 선택하십시오.

Note

엔드포인트는 Amazon Pinpoint를 사용하여 푸시 알림이 전송될 수 있는 사용자 디바이스를 고유하게 식별합니다. 엔드포인트에 대한 자세한 정보는 Amazon Pinpoint 개발자 안내서에서 [엔드포인트 추가](#)를 참조하십시오.

8. 이미 생성한 IAM 역할을 입력하거나 새 역할 생성을 선택하여 IAM 콘솔에서 새 역할을 생성합니다.
9. [Save changes]를 선택합니다.
10. 추가 앱 매핑을 지정하려면 Add another app mapping(다른 앱 매핑 추가)을 선택합니다.
11. [Save changes]를 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](#)

사용자 풀에서 사용자 관리

사용자 풀을 생성한 후 사용자 계정을 생성, 확인 및 관리할 수 있습니다. Amazon Cognito 사용자 풀 그룹이 있으면 IAM 역할을 그룹에 매핑함으로써 사용자와 리소스에 대한 액세스를 관리할 수 있습니다.

사용자 마이그레이션 Lambda 트리거를 사용하여 사용자를 사용자 풀로 가져올 수 있습니다. 이 접근 방식을 통해 사용자가 사용자 풀에 처음 로그인할 때 기존 사용자 디렉터리에서 사용자 풀로의 원활한 사용자 마이그레이션이 가능합니다.

주제

- [사용자 계정 가입 및 확인 \(p. 156\)](#)
- [관리자로서 사용자 계정 생성 \(p. 163\)](#)
- [사용자 풀에 그룹 추가 \(p. 166\)](#)
- [사용자 계정 관리 및 검색 \(p. 168\)](#)
- [사용자 풀로 사용자 가져오기 \(p. 171\)](#)

사용자 계정 가입 및 확인

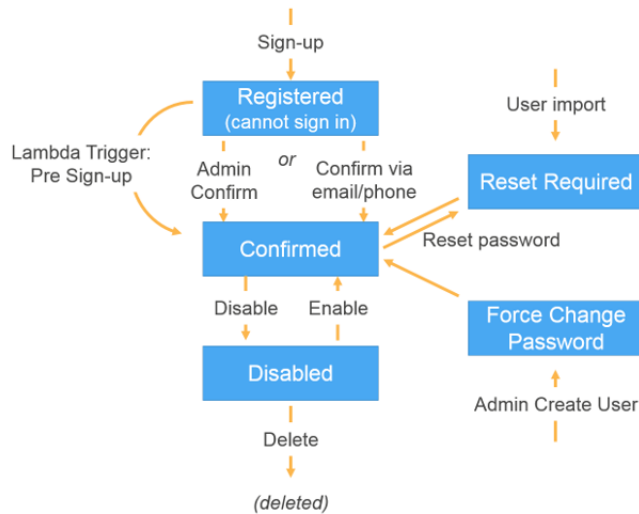
다음 방법 중 하나를 통해 사용자 계정이 사용자 풀에 추가됩니다.

- 사용자는 모바일 또는 웹 앱일 수 있는 사용자 풀의 클라이언트 앱에 가입합니다.
- 사용자의 계정을 사용자 풀로 가져올 수 있습니다. 자세한 내용은 [CSV 파일에서 사용자 풀로 사용자 가져오기 \(p. 172\)](#) 단원을 참조하십시오.
- 사용자 풀에서 사용자의 계정을 생성하고 로그인하도록 사용자를 초대할 수 있습니다. 자세한 내용은 [관리자로서 사용자 계정 생성 \(p. 163\)](#) 단원을 참조하십시오.

스스로 가입한 사용자가 로그인하려면 먼저 사용자를 확인해야 합니다. 가져온 사용자 및 생성된 사용자는 이미 확인되어 있지만 처음으로 로그인할 때 암호를 생성해야 합니다. 다음 단원에서는 확인 프로세스와 이메일 및 전화 확인에 대해 설명합니다.

사용자 계정 확인 개요

다음 그림은 확인 프로세스를 보여줍니다.



사용자 계정은 다음 상태 중 하나일 수 있습니다.

등록(확인되지 않음)

사용자가 가입에는 성공했지만 사용자 계정을 확인할 때까지 로그인할 수 없습니다. 이 상태에서는 사용자가 활성화되었지만 확인되지 않았습니다.

스스로 가입한 새 사용자는 이 상태에서 시작합니다.

확인됨

사용자 계정이 확인되었으며 사용자가 로그인할 수 있습니다. 사용자가 이메일 또는 전화(SMS)를 통해 받은 확인 코드(이메일의 경우 확인 링크)를 입력하여 사용자 계정을 확인한 경우, 해당 이메일 또는 전화 번호가 자동으로 확인됩니다. 코드 또는 링크는 24시간 동안 유효합니다.

관리자 또는 사전 가입 Lambda 트리거를 통해 사용자 계정이 확인된 경우 계정과 연관된 이메일 또는 전화 번호가 확인되지 않을 수 있습니다.

암호 재설정 필요

사용자 계정은 확인되었지만 사용자가 코드를 요청하고 로그인하기 전에 암호를 재설정해야 합니다.

관리자 또는 개발자가 가져온 사용자 계정은 이 상태로 시작합니다.

강제로 암호 변경

사용자 계정이 확인되었으며 사용자가 임시 암호를 사용하여 로그인할 수 있지만 처음 로그인할 때 사용자가 암호를 새 값으로 변경해야 다른 작업을 수행할 수 있습니다.

관리자 또는 개발자가 생성한 사용자 계정은 이 상태로 시작합니다.

Disabled

사용자 계정을 삭제하기 전에 먼저 비활성화해야 합니다.

가입 시 연락처 정보 확인

새 사용자가 앱에 가입할 때, 연락할 방법을 최소한 한 가지는 알고 싶을 것입니다. 사용자의 연락처 정보는 예를 들어 다음과 같이 사용할 수 있습니다.

- 사용자가 암호 재설정을 선택할 때 임시 암호를 보냅니다.

- 개인 정보나 금융 정보가 업데이트되면 사용자에게 알립니다.
- 특별 행사나 할인 등 프로모션 메시지를 보냅니다.
- 계정 요약 또는 청구 안내서를 보냅니다.

이러한 사용 사례에서는 확인된 대상 주소로 메시지를 보내는 것이 중요합니다. 그렇지 않으면 잘못된 이메일 주소나 잘못 입력된 전화번호로 메시지를 보내게 됩니다. 심지어 사용자인 척하는 악당들에게 중요 정보를 보낼 가능성도 있습니다.

정당한 개인에게만 메시지를 보내려면 사용자가 가입할 때 다음 정보를 제공해야 하도록 Amazon Cognito 사용자 풀을 구성하십시오.

- a. 이메일 주소 또는 전화번호
- b. Amazon Cognito가 해당 이메일 주소 또는 전화번호로 보내는 확인 코드

사용자는 이 확인 코드를 보냄으로써 코드를 받은 해당 메일박스 또는 전화에 대한 액세스 권한이 자신에게 있음을 입증하게 됩니다. 사용자가 이 코드를 입력하면 Amazon Cognito는 다음과 같이 사용자 풀에서 해당 사용자의 정보를 업데이트합니다.

- 사용자 상태를 `CONFIRMED`로 설정합니다.
- 이메일 주소 또는 전화번호가 확인되었음을 표시하여 사용자 속성을 업데이트합니다.

Amazon Cognito 콘솔에서 이 정보를 볼 수 있습니다. 또는 `AdminGetUser` API 작업, AWS CLI의 `admin-get-user` 명령, 아니면 AWS SDK 중 하나에서 해당 작업을 사용해도 됩니다.

연락 방법이 확인된 사용자가 암호 재설정을 요구하면 Amazon Cognito는 자동으로 그 사용자에게 메시지를 보냅니다.

이메일 또는 전화 확인을 요구하도록 사용자 풀을 구성하려면

이메일 또는 전화 확인을 요구함으로써 신뢰할 수 있는 사용자 연락 방법을 확보할 수 있습니다. 다음 단계에 따라 Amazon Cognito 콘솔에서 사용자 풀을 구성하십시오.

시작하기 전에

아직 계정에 사용자 풀이 없다면 하나 생성해야 합니다. 역할을 만들려면 [사용자 풀 시작하기 \(p. 13\)](#) 단원을 참조하십시오.

사용자 풀을 구성하려면

1. AWS Management 콘솔에 로그인한 다음 <https://console.aws.amazon.com/cognito>에서 Amazon Cognito 콘솔을 엽니다.
2. Manage User Pools(사용자 풀 관리)를 선택합니다.
3. 사용자 풀 페이지에서 구성할 사용자 풀을 선택합니다.
4. 왼쪽의 탐색 메뉴에서 MFA and verifications(MFA 및 확인)를 선택합니다.

Which attributes do you want to verify?(어떤 속성을 확인하시겠습니까?) 아래에 이메일 또는 전화 확인을 위한 옵션이 표시됩니다.

Which attributes do you want to verify?
Verification requires users to retrieve a code from their email or phone to confirm ownership. Verification of a phone or email is necessary to automatically confirm users and enable recovery from forgotten passwords. [Learn more about email and phone verification.](#)
☒ Email ☐ Phone number ☐ Email or phone number ☐ No verification

5. 다음 옵션 중 하나를 선택합니다.

이메일

이 옵션을 선택하면 사용자가 로그인할 때 Amazon Cognito에서 확인 코드를 이메일로 보냅니다. 보통 이메일을 통해 사용자와 소통하는 경우 이 옵션을 선택하십시오. 예를 들어 대금 청구서, 주문 요약, 특별 행사 등을 보낼 때 확인된 이메일 주소를 사용하게 됩니다.

전화번호

이 옵션을 선택하면 사용자가 로그인할 때 Amazon Cognito에서 SMS를 통해 확인 코드를 보냅니다. 보통 SMS를 통해 사용자와 소통하는 경우 이 옵션을 선택하십시오. 예를 들어 배송 통지, 약속 확인, 경보 등을 보낼 때 확인된 전화번호를 사용하게 됩니다.

이메일 또는 전화번호

모든 사용자의 확인된 연락 방법이 동일하지 않아도 되는 경우 이 옵션을 선택하십시오. 이 경우 앱의 로그인 페이지에서 사용자에게 선호하는 연락 방법만 확인해 달라고 요청할 수 있습니다. Amazon Cognito는 확인 코드를 보낼 때 앱의 signUp 요청에 명시된 연락 방법으로 코드를 보냅니다. 사용자가 이메일 주소와 전화번호를 둘 다 입력했고 앱의 signUp 요청에 두 가지 연락 방법이 모두 나와 있는 경우, Amazon Cognito는 전화번호로만 확인 코드를 보냅니다.

사용자에게 이메일 주소와 전화번호를 둘 다 확인해 달라고 요청하려면 이 옵션을 선택하십시오. 사용자가 가입할 때 Amazon Cognito가 한 가지 연락 방법을 확인하고, 사용자가 로그인한 뒤 앱에서 나머지 연락 방법을 확인해야 합니다. 자세한 내용은 [사용자에게 이메일 주소와 전화번호 둘 다 확인하도록 요구하는 경우 \(p. 160\)](#) 단원을 참조하십시오.

없음

이 옵션을 선택하면 사용자가 가입할 때 Amazon Cognito에서 확인 코드를 보내지 않습니다. Amazon Cognito의 확인 코드를 사용하지 않고 사용자 정의 인증 흐름을 통해 한 가지 이상의 연락 방법을 확인하는 경우 이 옵션을 선택하십시오. 예를 들어, 특정 도메인에 속하는 이메일 주소를 자동으로 확인해 주는 사전 가입 Lambda 트리거를 사용할 수 있습니다.

사용자의 연락처 정보를 확인하지 않으면 경우에 따라 해당 사용자가 앱을 사용하지 못하게 될 수도 있습니다. 사용자에게 확인된 연락처 정보가 필요한 이유를 명시하십시오.

- 암호 재설정 사용자가 forgotPassword API 작업을 호출하면 Amazon Cognito는 해당 사용자의 이메일 주소나 전화번호로 임시 암호를 보냅니다. Amazon Cognito는 그 사용자에게 확인된 연락 방법이 최소한 한 가지는 있어야 이 암호를 보내 줍니다.
- 이메일 주소나 전화번호를 별칭으로 사용하여 로그인합니다. 이러한 별칭을 허용하도록 사용자 풀을 구성하더라도, 사용자는 그 별칭이 확인된 경우에만 별칭으로 로그인할 수 있습니다. 자세한 내용은 [별칭 개요 \(p. 204\)](#) 단원을 참조하십시오.

6. [Save changes]를 선택합니다.

이메일 또는 전화 확인을 이용한 인증 흐름

사용자 풀에서 사용자에게 연락처 정보를 확인할 것을 요구하는 경우, 사용자가 가입할 때 앱에서 다음과 같은 흐름으로 진행되어야 합니다.

1. 사용자는 사용자 이름, 전화 번호 및/또는 이메일 주소와 가능한 경우 다른 속성을 입력하여 앱에 가입합니다.
2. Amazon Cognito 서비스는 앱으로부터 가입 요청을 받습니다. 요청에 가입에 필요한 모든 속성이 포함되어 있는지 확인한 후 서비스에서는 가입 프로세스를 완료하고 확인 코드를 사용자의 전화(SMS를 통해) 또는 이메일로 전송합니다. 코드는 24시간 동안 유효합니다.
3. 서비스가 가입이 완료되었으며 사용자 계정의 확인이 보류 중인 앱으로 반환됩니다. 응답에는 확인 코드가 전송된 위치에 대한 정보가 포함되어 있습니다. 이때 사용자의 계정은 미확인 상태이며, 사용자의 이메일 주소와 전화 번호가 확인되지 않습니다.
4. 이제 사용자에게 확인 코드를 입력하라는 메시지가 앱에 표시됩니다. 사용자가 코드를 즉시 입력할 필요는 없습니다. 그러나 사용자가 확인 코드를 입력해야 로그인할 수 있습니다.

5. 사용자가 앱에 확인 코드를 입력합니다.
6. 앱은 [ConfirmSignUp](#)을 호출하여 Amazon Cognito 서비스에 코드를 전송합니다. 그러면 이 서비스에서 코드를 확인하고, 코드가 올바르면 사용자의 계정을 확인됨 상태로 설정합니다. 사용자 계정을 성공적으로 확인한 후 서비스에서는 확인에 사용된 속성(이메일 또는 전화 번호)을 확인됨으로 자동으로 표시합니다. 이 속성 값이 변경되지 않은 한 사용자는 해당 속성 값을 다시 확인할 필요가 없습니다.
7. 이때 사용자의 계정은 확인 상태이며, 사용자가 로그인할 수 있습니다.

사용자에게 이메일 주소와 전화번호 둘 다 확인하도록 요구하는 경우

Amazon Cognito는 사용자가 가입할 때 한 가지 연락 방법만 확인합니다. Amazon Cognito가 이메일 주소 확인과 전화번호 확인 중 하나를 선택해야 할 때는 SMS를 통해 확인 코드를 보냄으로써 전화번호를 확인하게 됩니다. 예를 들어 사용자가 이메일 주소 또는 전화번호 중 하나를 확인하면 되도록 사용자 풀을 구성했는데 앱에서는 가입할 때 이 두 가지 속성을 모두 제공하는 경우, Amazon Cognito는 전화번호만 확인합니다. 사용자가 전화번호를 확인하면 Amazon Cognito는 해당 사용자의 상태를 `CONFIRMED`로 설정하고, 사용자는 앱에 로그인할 수 있게 됩니다.

사용자가 로그인한 뒤, 가입 단계에서 확인되지 않은 연락 방법을 확인할 수 있는 옵션이 앱에 표시됩니다. 이 두 번째 방법을 확인하기 위해 앱은 `VerifyUserAttribute` API 작업을 호출합니다. 이 작업에는 `AccessToken` 파라미터가 필요하고, Amazon Cognito는 인증된 사용자에게만 액세스 토큰을 제공한다는 점에 주의하십시오. 따라서 두 번째 연락 방법은 사용자가 로그인한 후에만 확인할 수 있습니다.

이메일 주소와 전화번호 둘 다 확인하도록 사용자에게 요구하려면 다음과 같이 하십시오.

1. 사용자가 이메일 주소 또는 전화번호를 확인할 수 있도록 사용자 풀을 구성합니다.
2. 앱 가입 흐름에서 이메일 주소와 전화번호 둘 다 제공할 것을 사용자에게 요구합니다. [SignUp](#) API 작업을 호출하고 `UserAttributes` 파라미터에 이메일 주소와 전화번호를 입력합니다. 이때 Amazon Cognito는 해당 사용자의 전화로 확인 코드를 보냅니다.
3. 앱 인터페이스에 사용자가 확인 코드를 입력할 수 있는 확인 페이지가 표시됩니다. [ConfirmSignUp](#) API 작업을 호출하여 사용자를 확인합니다. 이 시점에서 사용자 상태는 `CONFIRMED`가 되고 사용자의 전화번호가 확인되지만 이메일 주소는 확인되지 않았습니다.
4. 로그인 페이지를 열고 [InitiateAuth](#) API 작업을 호출하여 사용자를 인증합니다. 사용자가 인증되면 Amazon Cognito는 액세스 토큰을 앱에 반환합니다.
5. [GetUserAttributeVerificationCode](#) API 작업을 호출합니다. 요청에 다음 파라미터를 지정합니다.

- `AccessToken` – 사용자가 로그인할 때 Amazon Cognito가 반환한 액세스 토큰입니다.
- `AttributeName` – 속성 값으로 "email"을 지정합니다.

Amazon Cognito는 사용자의 이메일 주소로 확인 코드를 보냅니다.

6. 사용자가 확인 코드를 입력할 수 있는 확인 페이지를 표시합니다. 사용자가 코드를 제출하면 [VerifyUserAttribute](#) API 작업을 호출합니다. 요청에 다음 파라미터를 지정합니다.

- `AccessToken` – 사용자가 로그인할 때 Amazon Cognito가 반환한 액세스 토큰입니다.
- `AttributeName` – 속성 값으로 "email"을 지정합니다.
- `Code` – 사용자가 제공한 확인 코드입니다.

이때 이메일 주소가 확인됩니다.

사용자가 앱에 가입할 수 있지만 관리자로 확인됨

1. 사용자는 사용자 이름, 전화 번호 및/또는 이메일 주소와 가능한 경우 다른 속성을 입력하여 앱에 가입합니다.

2. Amazon Cognito 서비스는 앱으로부터 가입 요청을 받습니다. 요청에 가입에 필요한 모든 속성이 포함되어 있는지 확인한 후 서비스에서는 가입 프로세스를 완료하고, 가입이 완료되고 확인 보류 중인 앱으로 반환됩니다. 이때 사용자의 계정은 미확인 상태입니다. 사용자는 계정이 확인될 때까지 로그인할 수 없습니다.
3. 관리자는 Amazon Cognito 콘솔(사용자 탭에서 사용자 계정을 찾은 다음 확인 버튼 선택) 또는 CLI(admin-confirm-sign-up 명령 사용)에서 사용자의 계정을 확인합니다. 확인 버튼과 admin-confirm-sign-up 명령은 둘 다 [AdminConfirmSignUp](#) API를 사용하여 확인을 수행합니다.
4. 이때 사용자의 계정은 확인 상태이며, 사용자가 로그인할 수 있습니다.

SecretHash 값 계산

다음 Amazon Cognito 사용자 풀 API에는 SecretHash 파라미터가 있습니다.

- [ConfirmForgotPassword](#)
- [ConfirmSignUp](#)
- [ForgotPassword](#)
- [ResendConfirmationCode](#)
- [SignUp](#)

SecretHash 값은 메시지에서 사용자 풀 클라이언트의 보안 키와 사용자 이름 및 클라이언트 ID를 사용하여 계산된 Base 64 인코딩 키가 추가된 HMAC(해시 메시지 인증 코드)입니다. 다음 유사 코드는 이 값이 계산되는 방식을 보여줍니다. 이 유사 코드에서 +는 연결을 나타내고, HMAC_SHA256은 HmacSHA를 사용하여 HMAC 값을 생성하는 기능을 나타내며, Base64는 해시 출력의 Base-64 인코딩 버전을 생성하는 기능을 나타냅니다.

```
Base64 ( HMAC_SHA256 ( "Client Secret Key", "Username" + "Client Id" ) )
```

또는 서버 측 Java 애플리케이션 코드에서 다음 코드 예제를 사용할 수 있습니다.

```
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 ");
    }
}
```

이메일 또는 전화 번호를 확인하지 않고 사용자 계정 확인

가입 시 확인 코드 또는 이메일이나 전화 번호의 확인 없이 사용자 계정을 자동 확인하는 데 사전 가입 Lambda 트리거를 사용할 수 있습니다. 이러한 방법으로 확인된 사용자는 코드를 받지 않고 즉시 로그인할 수 있습니다.

이 트리거를 통해 사용자의 이메일 또는 전화 번호를 확인됨으로 표시할 수도 있습니다.

Note

이러한 접근 방식은 사용자가 시작할 때 편리하지만 최소한 하나의 이메일 또는 전화 번호를 자동 확인하는 것이 좋습니다. 그렇지 않으면 사용자가 암호를 잊어버렸을 경우 복구할 수 없게 됩니다.

가입 시 사용자가 확인 코드를 받거나 입력할 필요가 없는 경우 또는 사전 가입 Lambda 트리거에서 이메일이나 전화 번호를 자동으로 확인하지 않는 경우 해당 사용자 계정에 대해 확인된 이메일 주소나 전화 번호가 없는 위험이 있습니다. 사용자는 나중에 이메일 주소나 전화 번호를 확인할 수 있습니다. 그러나 사용자가 암호를 잊어버렸으며 확인된 이메일 주소나 전화 번호가 없는 경우 해당 계정에 대해 사용자가 잠깁니다. 암호 찾기 흐름에는 사용자에게 확인 코드를 전송하기 위한 확인된 이메일 또는 전화 번호가 필요하기 때문입니다.

사용자가 이메일 또는 전화 번호를 변경할 경우 확인

사용자가 앱에서 이메일 주소나 전화 번호를 변경하면 해당 속성이 확인되지 않음으로 표시됩니다. 업데이트되는 속성에 대해 자동 확인이 활성화되어 있는 경우 서비스에서는 즉시 사용자에게 확인 코드가 포함된 메시지를 전송하며, 사용자는 변경 사항을 확인하기 위해 이 코드를 입력해야 합니다. 사용자 지정 메시지 Lambda 트리거를 사용하여 이 메시지를 사용자 지정할 수 있습니다. 자세한 내용은 [Lambda 트리거를 사용하여 사용자 풀 워크플로우 사용자 지정](#) (p. 118) 단원을 참조하십시오. 사용자의 이메일 주소 또는 전화 번호가 확인되지 않는 경우에는 항상 앱에 확인되지 않음 상태가 표시되며 사용자가 새 이메일 또는 전화 번호를 확인할 수 있는 버튼이나 링크가 제공됩니다.

관리자 또는 개발자가 생성한 사용자 계정에 대한 확인 및 검증 프로세스

관리자 또는 개발자가 생성한 사용자 계정은 이미 확인된 상태이므로 사용자가 확인 코드를 입력할 필요가 없습니다. Amazon Cognito 서비스에서 이러한 사용자에게 전송한 초대 메시지에는 사용자 이름과 임시 암호가 있습니다. 사용자는 로그인하기 전에 암호를 변경해야 합니다. 자세한 내용은 [메시지 사용자 지정](#) (p. 165)의 관리자로서 사용자 계정 생성 (p. 163) 및 [Lambda 트리거를 사용하여 사용자 풀 워크플로우 사용자 지정](#) (p. 118)의 사용자 지정 메시지 트리거를 참조하십시오.

가져온 사용자 계정에 대한 확인 및 검증 프로세스

AWS Management 콘솔, CLI 또는 API([CSV 파일에서 사용자 풀로 사용자 가져오기](#) (p. 172) 참조)에서 사용자 가져오기 기능을 통해 생성한 사용자 계정은 이미 확인된 상태이므로 사용자가 확인 코드를 입력할 필요가 없습니다. 초대 메시지가 전송되지 않습니다. 그러나 가져온 사용자 계정에서는 사용자가 로그인하기 전에 `ForgotPassword` API를 호출하여 먼저 코드를 요청한 다음 `ConfirmForgotPassword` API를 호출하여 전달된 코드를 사용하여 암호를 생성해야 합니다. 자세한 내용은 [가져온 사용자에게 암호 재설정 요구](#) (p. 181) 단원을 참조하십시오.

사용자가 로그인할 때 확인이 필요하지 않도록 사용자 계정을 가져올 때 사용자의 이메일이나 전화 번호가 확인됨으로 표시되어야 합니다.

앱 테스트 중 이메일 보내기

Amazon Cognito는 사용자 풀에서 클라이언트 앱의 계정을 만들고 관리하는 사용자에게 이메일을 보냅니다. 이메일 확인을 요구하도록 사용자 풀을 구성하는 경우, Amazon Cognito는 다음 조건에서 이메일을 보냅니다.

- 사용자가 가입합니다.
- 사용자가 이메일 주소를 업데이트합니다.
- 사용자가 `ForgotPassword` API 작업을 호출하는 작업을 수행합니다.
- 관리자 역할을 할 사용자 계정을 생성합니다.

이메일을 시작한 작업이 무엇인지에 따라 이메일에는 확인 코드가 들어가거나 임시 암호가 들어갑니다. 사용자는 이러한 이메일을 받고 메시지 내용을 이해해야 합니다. 그렇지 않으면 로그인하여 앱을 사용할 수 없습니다.

이메일이 제대로 전송되었고 메시지는 제대로 보이는지 확인하기 위해, Amazon Cognito에서 이메일 배달을 시작하는 작업을 앱에서 테스트해 봅니다. 예를 들어, 테스트 이메일 주소로 가입한 다음 앱에서 가입 페이지를 사용하거나 `signUp` API 작업을 사용하여 이메일을 시작할 수 있습니다. 이런 방식으로 테스트할 때는 다음을 기억하십시오.

중요

Amazon Cognito에서 이메일을 시작하는 작업을 이메일 주소로 테스트할 때는 가짜 이메일 주소(메일박스가 없는 주소)를 사용하지 마십시오. 하드 바운스 없이 Amazon Cognito의 이메일을 받을 수 있는 실제 이메일 주소를 사용하십시오.

Amazon Cognito가 수신자의 메일박스로 이메일을 배달하지 못하면 하드 바운스가 발생합니다. 메일박스가 없으면 늘 그렇게 됩니다.

Amazon Cognito는 하드 바운스가 지속적으로 발생하는 AWS 계정의 이메일 전송 횟수를 제한합니다.

이메일을 시작하는 작업을 테스트할 때는 다음 이메일 주소 중 하나를 사용하여 하드 바운스를 방지하십시오.

- 사용자가 소유 중이고 테스트에 사용하는 이메일 계정의 주소입니다. 자체 이메일 주소를 사용하는 경우, Amazon Cognito가 보낸 이메일을 받게 됩니다. 앱에서 가입 경험을 테스트할 때 이 이메일로 확인 코드를 사용할 수 있습니다. 사용자 풀의 이메일 메시지를 사용자 지정한 경우, 수정한 내용이 잘 보이는지 확인할 수 있습니다.
- 메일박스 시뮬레이터 주소인 `success@simulator.amazonses.com`입니다. 시뮬레이터 주소를 사용하는 경우, Amazon Cognito에서 이메일을 보내도 볼 수는 없습니다. 이 옵션은 확인 코드를 사용할 필요가 없고 이메일 메시지를 확인하지 않아도 되는 경우에 유용합니다.
- 메일박스 시뮬레이터 주소에 `success+user1@simulator.amazonses.com` 또는 `success+user2@simulator.amazonses.com` 같은 임의의 레이블을 추가한 것입니다. Amazon Cognito는 이 주소로 이메일을 보냈지만 전송된 이메일을 볼 수는 없었습니다. 이 옵션은 각각 고유한 이메일 주소가 있는 테스트 사용자 여러 명을 사용자 풀에 추가하여 가입 프로세스를 테스트하고 싶을 때 유용합니다.

관리자로서 사용자 계정 생성

사용자 풀을 생성한 후 AWS Command Line Interface 또는 Amazon Cognito API 뿐만 아니라 AWS Management 콘솔을 사용하여 사용자를 생성할 수 있습니다. 사용자 풀에서 새 사용자에 대한 프로필을 생성하고, SMS 또는 이메일을 통해 사용자에게 가입 지침이 포함된 환영 메시지를 전송할 수 있습니다.

개발자 및 관리자는 다음 작업을 수행할 수 있습니다.

- AWS Management 콘솔을 사용하거나 `AdminCreateUser` API를 호출하여 새 사용자 프로필을 생성합니다.
- 임시 암호를 지정하거나 Amazon Cognito가 암호를 자동으로 생성하도록 허용합니다.
- 제공된 이메일 주소 및 전화 번호가 새 사용자에 대해 확인됨으로 표시되는지 여부를 지정합니다.
- AWS Management 콘솔 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 워크플로우 사용자 지정 \(p. 118\)](#) 단원을 참조하십시오.
- 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 콘솔을 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.

코드 예제의 경우 다음 주제를 참조하십시오.

- 예제: Android용 Mobile SDK에서 AdminCreateUser API를 사용하여 생성된 사용자 처리 (p. 55)
- 예제: iOS SDK를 통한 사용자 풀 사용 (p. 61)
- 예제: SDK for JavaScript에서 AdminCreateUser API를 통해 생성한 사용자의 새 암호를 인증 및 설정 (p. 36)

관리자 또는 개발자가 생성한 사용자에게 대한 인증 흐름

이러한 사용자에게 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 사용자에게 적용됩니다.

1. 사용자는 제공된 사용자 이름과 암호를 제출하여 처음으로 로그인을 시작합니다.
2. SDK는 InitiateAuth(Username, USER_SRP_AUTH)를 호출합니다.
3. Amazon Cognito는 솔트 및 암호 블록이 있는 PASSWORD_VERIFIER 챌린지를 반환합니다.
4. SDK는 SRP 계산을 수행하며 RespondToAuthChallenge(Username, *<SRP variables>*, PASSWORD_VERIFIER)를 호출합니다.
5. Amazon Cognito는 현재 및 필수 속성과 함께 NEW_PASSWORD_REQUIRED 챌린지를 반환합니다.
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 챌린지를 반환하고 코드가 제출됩니다.
9. 사용자가 암호와 선택 항목으로 제공된 속성 값 또는 완료된 MFA를 성공적으로 변경하면 사용자가 로그인되고 토큰이 발급됩니다.

사용자가 모든 챌린지에 대해 만족하면 Amazon Cognito 서비스는 사용자를 확인됨으로 표시하고 해당 사용자에게 ID, 액세스 및 새로 고침 토큰을 발급합니다. 자세한 내용은 [사용자 풀을 통해 토큰 사용 \(p. 190\)](#) 단원을 참조하십시오.

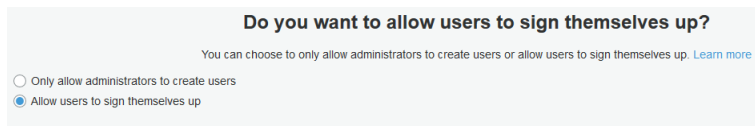
AWS Management 콘솔에서 새 사용자 생성

사용자 풀을 관리하기 위한 Amazon Cognito 콘솔은 다음에 표시된 대로 이 기능을 지원하기 위해 업데이트됩니다.

정책 탭

정책 탭에는 이와 관련된 설정이 있습니다.

- 사용자가 스스로 가입할 수 있도록 허용하는지 여부를 지정합니다. 이 옵션은 기본적으로 설정되어 있습니다.



Do you want to allow users to sign themselves up?

You can choose to only allow administrators to create users or allow users to sign themselves up. [Learn more](#)

☐ Only allow administrators to create users

☒ Allow users to sign themselves up

- 새 계정에 대한 사용자 계정 만료 시간 제한(일)을 지정합니다. 기본 설정은 7일이며, 사용자 계정이 생성된 날로부터 측정됩니다. 최대 설정은 90일입니다. 계정이 만료되면 사용자는 관리자가 사용자의 프로파일을 업데이트할 때까지 해당 계정에 로그인할 수 없습니다.

Note

사용자가 로그인하면 계정은 만료되지 않습니다.

How quickly should user accounts created by administrators expire if not used?

You can choose for how long until a user account created by an administrator expires if the account is not used.

Days to expire

7

메시지 사용자 지정 탭

메시지 사용자 지정 탭에는 사용자 지정 이메일 확인 메시지와 사용자 지정 사용자 초대 메시지를 지정하기 위한 템플릿이 포함되어 있습니다.

이메일(확인 메시지 또는 사용자 초대 메시지)의 경우 메시지의 최대 길이는 확인 코드 또는 임시 암호를 포함하여 2048자(UTF-8)입니다. SMS의 경우 최대 길이는 확인 코드 또는 임시 암호를 포함하여 140자(UTF-8)입니다.

확인 코드는 24시간 동안 유효합니다.

Do you want to customize your email verification messages?

Email subject

Your verification code

Email message

Your verification code is {#####}.

You can customize the message above, but it must include the "{#####}" placeholder, which will be replaced with the code.

Do you want to customize your user invitation messages?

SMS message

Your username is {username} and temporary password is {#####}.

You can customize the message above, but it must include the "{username}" and "{#####}" placeholder, which will be replaced with the username and temporary password respectively.

Email subject

Your temporary password

Email message

Your username is {username} and temporary password is {#####}.

You can customize the message above, but it must include the "{username}" and "{#####}" placeholder, which will be replaced with the username and temporary password respectively.

사용자 탭

사용자 및 그룹 탭의 사용자 탭에는 사용자 생성 버튼이 있습니다.

Users **Groups**

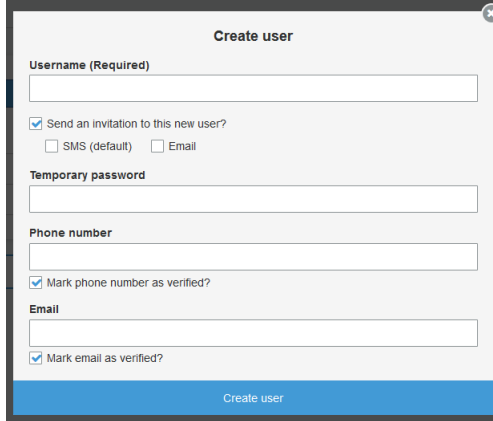
Import users Create user User name Search for value...

Username	Enabled	Account status	Email verified	Phone number verified	Updated	Created
7c0dc801ee66	Enabled	CONFIRMED	true	-	Nov 19, 2018 7:44:48 PM	Nov 19, 2018 7:44:48 PM
75f3de26d2b5	Enabled	CONFIRMED	true	true	Nov 7, 2018 8:59:35 PM	Nov 7, 2018 8:59:35 PM

사용자 생성을 선택하면 사용자 생성 양식이 표시됩니다. 이 양식을 사용하여 새 사용자에 대한 정보를 입력할 수 있습니다. 사용자 이름 필드만 필수입니다.

Note

AWS Management 콘솔에서 사용자 생성 양식을 사용하여 생성하는 사용자 계정의 경우 양식에 표시된 속성만 AWS Management 콘솔에서 설정할 수 있습니다. 다른 속성은 필수 속성으로 표시한 경우에도 AWS Command Line Interface 또는 Amazon Cognito API를 사용하여 설정해야 합니다.



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. AWS Identity and Access Management(IAM) 역할을 그룹에 할당하여 그룹 구성원에 대한 권한을 정의할 수 있습니다.

그룹을 사용하여 사용자 풀에서 사용자 모음을 생성할 수 있습니다. 이 작업은 해당 사용자에게 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 콘솔, API 및 CLI에서 사용자 풀의 그룹을 생성하고 관리할 수 있습니다. 개발자(자격 증명 사용)는 사용자 풀에 대한 그룹을 생성, 읽기, 업데이트, 삭제 및 나열할 수 있습니다. 그룹에서 사용자를 추가하고 제거할 수도 있습니다.

사용자 풀 내에서 그룹 사용에 대한 추가 비용은 없습니다. 자세한 내용은 [Amazon Cognito 요금](#)을 참조하십시오.

[SpaceFinder](#) 참조 앱에서 사용된 이 기능을 확인할 수 있습니다.

그룹에 IAM 역할 할당

그룹을 통해 그룹 내 사용자에게 대해 IAM 역할을 할당하여 AWS의 리소스에 액세스할 수 있는 권한을 제어할 수 있습니다. 그룹을 생성하면 그룹에 대한 역할 ARN을 제공하여 해당 그룹의 사용자에게 대해 IAM 역할을 지정할 수 있습니다. IAM 역할에는 사용자에게 대해 허용되고 거부되는 작업 및 리소스를 정의하는 연결된 정책이 있습니다. IAM 역할 및 해당 권한은 인증 사용자에게 대해 Amazon Cognito 자격 증명 풀이 제공하는 임시 AWS 자격 증명에 연결되어 있습니다. Amazon Cognito 연동 자격 증명이 토큰으로부터 역할 선택 옵션을 사용하여 AWS 자격 증명을 제공할 경우 그룹의 사용자가 그룹에 대한 IAM 역할에 자동으로 할당됩니다.

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.
- 자격 증명 풀을 통해 사용자에 대한 AWS 자격 증명을 요청할 경우 [GetCredentialsForIdentityCustomRoleARN](#) 파라미터에서 역할 ARN을 지정하여 사용 가능한 역할 중에서 앱을 선택할 수도 있습니다. 지정된 역할은 사용자에 대해 사용 가능한 역할과 일치해야 합니다.

그룹에 우선 순위 값 할당

사용자는 둘 이상의 그룹에 속할 수 있습니다. 사용자의 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. 이러한 우선 순위는 음수가 아닌 숫자이며 사용자 풀에서 사용자가 속할 수 있는 다른 그룹에 상대적으로 이 그룹의 우선 순위를 지정합니다. 0은 가장 높은 우선 순위 값입니다. 우선 순위 값이 낮은 그룹은 우선 순위 값이 높거나 null인 그룹보다 우선합니다. 사용자가 둘 이상의 그룹에 속할 경우 가장 낮은 우선 순위 값이 있는 그룹의 IAM 역할이 사용자의 ID 토큰에 있는 `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 사용자에게 대한 ID 토큰의 `cognito:preferred_role` 클레임에 사용됩니다. 두 그룹의 역할 ARN이 서로 다른 경우 `cognito:preferred_role` 클레임은 사용자의 ID 토큰에서 설정되지 않습니다.

Amazon API Gateway를 통해 그룹을 사용하여 권한 제어

Amazon API Gateway를 통해 사용자 풀에서 그룹을 사용하여 권한을 제어할 수 있습니다. 사용자가 속한 그룹은 사용자가 로그인할 때 사용자 풀에서 제공한 ID 토큰에 포함됩니다. 요청을 통해 이러한 ID 토큰을 Amazon API Gateway에 제출하고, 사용자 지정 권한 부여자 Lambda 기능을 사용하여 토큰을 확인한 다음 사용자가 속한 그룹을 검사할 수 있습니다. Amazon API Gateway 사용자 지정 권한 부여자로 사용자 풀 토큰을 사용하는 예제는 이 [블로그 게시물](#)을 참조하십시오.

그룹에 대한 제한 사항

사용자 그룹에 다음 제한이 적용됩니다.

- 생성할 수 있는 그룹 수는 [Amazon Cognito 서비스 제한 \(p. 311\)](#)에 따라 제한됩니다.
- 그룹은 중첩될 수 없습니다.
- 그룹에서 사용자를 검색할 수 없습니다.
- 이름으로 그룹을 검색할 수 없지만 그룹을 나열할 수 있습니다.
- 구성원이 없는 그룹만 삭제할 수 있습니다.

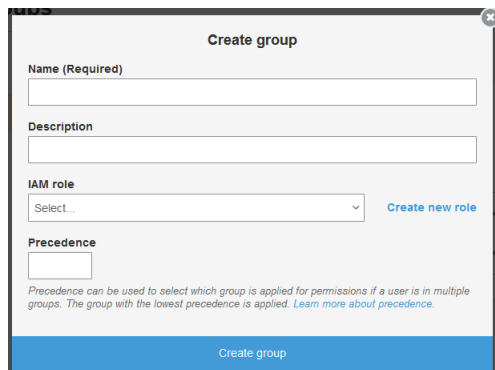
AWS Management 콘솔에서 새 그룹 생성

사용자 및 그룹 탭의 그룹 탭에는 그룹 생성 버튼이 있습니다.



그룹 생성을 선택한 경우 그룹 생성 양식이 표시됩니다. 새 그룹에 대한 정보를 이 그룹에 입력합니다. 이름 필드만 필수입니다. 사용자 풀을 자격 증명 풀과 통합한 경우 IAM 역할 설정은 자격 증명 풀이 토큰에서 역할

을 선택하도록 구성된 경우 사용자의 ID 토큰에 할당될 역할을 결정합니다. 정의된 역할이 아직 없는 경우 새 역할 생성을 선택합니다. 둘 이상의 그룹이 있으며 사용자가 둘 이상의 그룹에 할당될 수 있는 경우 각 그룹에 대해 우선 순위 값을 설정할 수 있습니다. 우선 순위 값은 음수가 아닌 정수일 수 있습니다. 0은 가장 높은 우선 순위 값입니다.



사용자 계정 관리 및 검색

사용자 풀을 생성하면 AWS Command Line Interface 또는 Amazon Cognito API 뿐만 아니라 AWS Management 콘솔을 사용하여 사용자를 보고 관리할 수 있습니다. 이 주제에서는 을 사용하여 사용자를 보고 검색하는 방법에 대해 설명합니다.

사용자 속성 보기

AWS Management 콘솔에서 수행할 수 있는 여러 작업이 있습니다.

- 풀 세부 정보를 보고 사용자 풀 속성, 암호 정책, MFA 설정, 앱 및 트리거를 편집할 수 있습니다. 자세한 내용은 [사용자 풀 참조\(AWS Management 콘솔\)](#) (p. 201) 단원을 참조하십시오.
- 사용자 풀에서 사용자를 보고 자세하게 살펴 볼 수 있습니다.
- 또한 사용자 풀에서 개별 사용자에 대한 세부 정보를 볼 수 있습니다.
- 사용자 풀에서 사용자를 검색할 수도 있습니다.

AWS Management 콘솔을 사용하여 사용자 풀을 관리하는 방법

1. AWS Management 콘솔의 Amazon Cognito 홈 페이지에서 사용자 자격 증명 관리를 선택합니다.
2. 사용자 풀 페이지에서 사용자 풀을 선택합니다.
3. User and Groups(사용자 및 그룹)를 선택하여 사용자 정보를 확인합니다.
4. 개별 사용자에 대한 자세한 내용을 표시하려면 사용자 이름을 선택하십시오. 이 화면에서 다음 작업 중 하나를 수행할 수 있습니다.
 - 그룹에 사용자 추가
 - 사용자 암호 재설정
 - 사용자 확인
 - MFA 활성화 또는 비활성화
 - 사용자 삭제

사용자 암호 재설정 작업으로 인해 사용자에게 확인 코드가 즉시 전송되며 사용자 상태가 RESET_REQUIRED로 변경되어 사용자의 현재 암호가 비활성화됩니다. MFA 활성화 작업으로 인해 사용자가 로그인을 시도할 때 확인 코드가 사용자에게 전송됩니다. 사용자 암호 재설정 코드는 1시간 동안 유효합니다. MFA 코드는 3분 동안 유효합니다.

사용자 속성 검색

사용자 풀을 이미 생성한 경우 AWS Management 콘솔의 사용자 패널에서 검색할 수 있습니다. Filter 파라미터를 수락하는 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

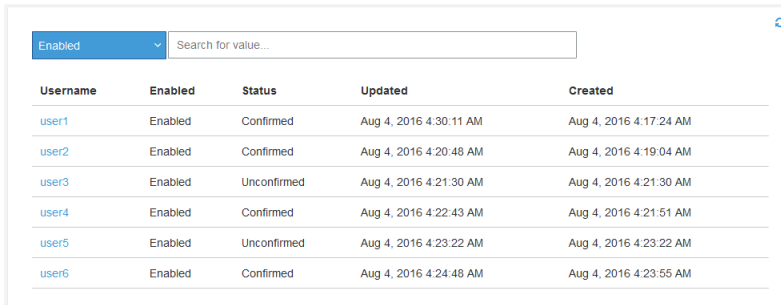
AWS Management 콘솔을 사용하여 사용자 검색

사용자 풀을 이미 생성한 경우 AWS Management 콘솔의 사용자 패널에서 검색할 수 있습니다.

AWS Management 콘솔 검색은 항상 접두사("다음으로 시작") 검색입니다.

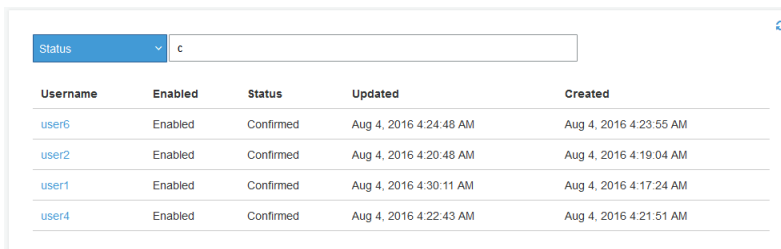
다음의 모든 예제는 동일한 사용자 풀을 사용합니다.

예를 들어, 모든 사용자를 나열하려면 검색 상자를 비워 두십시오.



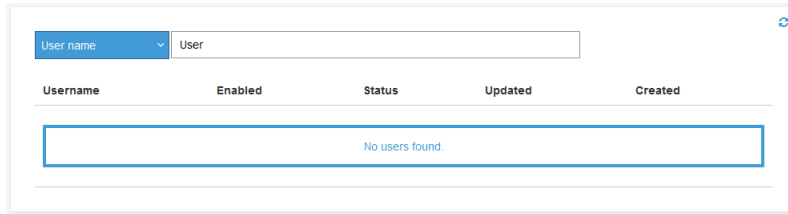
Username	Enabled	Status	Updated	Created
user1	Enabled	Confirmed	Aug 4, 2016 4:30:11 AM	Aug 4, 2016 4:17:24 AM
user2	Enabled	Confirmed	Aug 4, 2016 4:20:48 AM	Aug 4, 2016 4:19:04 AM
user3	Enabled	Unconfirmed	Aug 4, 2016 4:21:30 AM	Aug 4, 2016 4:21:30 AM
user4	Enabled	Confirmed	Aug 4, 2016 4:22:43 AM	Aug 4, 2016 4:21:51 AM
user5	Enabled	Unconfirmed	Aug 4, 2016 4:23:22 AM	Aug 4, 2016 4:23:22 AM
user6	Enabled	Confirmed	Aug 4, 2016 4:24:48 AM	Aug 4, 2016 4:23:55 AM

확인된 모든 사용자를 검색하려면 드롭다운 메뉴에서 상태를 선택합니다. 검색 상자에 "확인됨"이라는 단어의 첫 번째 글자를 입력합니다.



Username	Enabled	Status	Updated	Created
user6	Enabled	Confirmed	Aug 4, 2016 4:24:48 AM	Aug 4, 2016 4:23:55 AM
user2	Enabled	Confirmed	Aug 4, 2016 4:20:48 AM	Aug 4, 2016 4:19:04 AM
user1	Enabled	Confirmed	Aug 4, 2016 4:30:11 AM	Aug 4, 2016 4:17:24 AM
user4	Enabled	Confirmed	Aug 4, 2016 4:22:43 AM	Aug 4, 2016 4:21:51 AM

일부 속성 값은 User name과 같이 대/소문자를 구분합니다.



ListUsers API를 사용하여 사용자 검색

앱에서 사용자를 검색하려면 Amazon Cognito [ListUsers API](#)를 사용하십시오. 이 API는 다음 파라미터를 사용합니다.

- **AttributesToGet**: 문자열의 어레이입니다. 여기에서 각 문자열은 검색 결과에서 각 사용자에 대해 반환될 사용자 속성의 이름입니다. 어레이가 비어 있는 경우 모든 속성이 반환됩니다.
- **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 ListUsers는 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 =과 같이 given_name = "Jon"를 사용하십시오. 접두사("다음으로 시작")가 일치해야 하는 경우 ^=과 같이 given_name ^= "Jon"를 사용하십시오.
- **AttributeValue**: 각 사용자에 대해 일치해야 하는 속성 값입니다.
- **Limit**: 반환할 최대 사용자 수입니다.
- **PaginationToken**: 이전 검색에서 더 많은 결과를 가져올 토큰입니다.
- **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": [],
  "Filter": "",
  "Limit": 10,
  "UserPoolId": "us-east-1_samplepool"
}
```

다음은 전화 번호가 "+1312"로 시작하는 모든 사용자를 반환하고 모든 속성을 포함하는 예제입니다.

```
{
  "AttributesToGet": [],
  "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 풀로 사용자 가져오기

기존 사용자 디렉터리 또는 사용자 데이터베이스에서 사용자를 가져오거나 Amazon Cognito 사용자 풀로 마이그레이션할 수 있는 두 가지 방법이 있습니다. 사용자가 처음으로 Amazon Cognito를 사용하여 로그인할 때 사용자 마이그레이션 트리거를 사용하여 사용자를 마이그레이션할 수 있습니다. 이러한 접근 방식을 통해 사용자는 계속해서 기존 암호를 사용할 수 있고 사용자 풀로의 마이그레이션 이후 암호를 재설정할 필요가 없습니다. 또는 모든 사용자에 대한 사용자 프로필 속성을 포함한 CSV 파일을 업로드하여 일괄적으로 사용자를 마이그레이션할 수 있습니다. 다음 단원에서는 이 두 가지 접근 방식을 모두 설명합니다.

주제

- [사용자 마이그레이션 Lambda 트리거를 사용하여 사용자 풀로 사용자 가져오기 \(p. 171\)](#)
- [CSV 파일에서 사용자 풀로 사용자 가져오기 \(p. 172\)](#)

사용자 마이그레이션 Lambda 트리거를 사용하여 사용자 풀로 사용자 가져오기

이 접근 방식을 통해 사용자가 최초 로그인 도중 또는 암호 찾기 프로세스 도중 새로운 Amazon Cognito 활성화 앱을 처음 사용할 때 기존 사용자 디렉터리의 사용자를 사용자 풀로 원활하게 마이그레이션할 수 있습니다. 이 마이그레이션은 사용자 풀에서 구성해야 하는 사용자 마이그레이션 Lambda 함수로 활성화됩니다. 요청 및 응답 파라미터를 포함한 Lambda 트리거와 예제 코드에 대한 자세한 내용은 [사용자 마이그레이션 Lambda 트리거 파라미터 \(p. 148\)](#) 단원을 참조하십시오.

사용자 마이그레이션 프로세스를 시작하기 전에 AWS 계정에서 사용자 마이그레이션 Lambda 함수를 생성하고, 사용자 풀을 구성하여 사용자 마이그레이션 트리거에 대한 Lambda 함수 ARN을 사용합니다. Lambda 함수에 인증 정책을 추가하여 Amazon Cognito 서비스 계정 보안 주체("cognito-idp.amazonaws.com") 및 사용자 풀의 SourceARN에서만 이를 호출하도록 활성화하고, 다른 AWS 고객의 사용자 풀의 Lambda 함수 호출을 방지합니다. 자세한 내용은 [AWS Lambda에 대한 리소스 기반 정책 사용\(Lambda 함수 정책\)](#) 단원을 참조하십시오.

로그인 도중 단계

1. 사용자가 앱을 열고 [AWS Mobile SDK](#)로부터 Cognito Identity Provider API를 사용하여 앱의 기본 로그인 UI 화면을 통해 로그인합니다. 또는 [Amazon Cognito Auth SDK](#)를 사용하여 Amazon Cognito에서 제공된 호스팅된 로그인 UI를 사용합니다.
2. 앱에서 사용자 이름과 암호를 Amazon Cognito로 전송합니다. 앱에 기본 로그인 UI가 있고 Cognito Identity Provider SDK를 사용한 경우 앱에서 `USER_PASSWORD_AUTH` 흐름을 사용해야 합니다. 여기서 SDK는 암호를 서버로 전송합니다(앱에서 기본 `USER_SRP_AUTH` 흐름을 사용해서는 안 됩니다. SDK가 SRP 인증 흐름에서 암호를 서버로 전송하지 않기 때문입니다). `USER_PASSWORD_AUTH` 흐름은 `AuthenticationDetails.authenticationType`을 "USER_PASSWORD"로 설정함으로써 활성화됩니다.
3. Amazon Cognito는 사용자의 이메일, 전화번호 또는 기본 설정 사용자 이름에 대한 별칭을 포함하여 사용자 풀에 사용자 이름이 존재하는지 확인합니다. 사용자가 존재하지 않는 경우 Amazon Cognito가 사용자 이름 및 암호를 포함한 파라미터를 포함한 사용자 마이그레이션 Lambda 함수를 호출하며, 이는 [사용자 마이그레이션 Lambda 트리거 파라미터 \(p. 148\)](#) 단원에 상세히 설명되어 있습니다.
4. 사용자 마이그레이션 Lambda 함수는 기존 로그인 서비스를 호출함으로써 기존 사용자 디렉터리 또는 사용자 데이터베이스를 통해 사용자를 인증합니다. 그리고 사용자 풀의 사용자 프로필에 저장된 사용자 속성이 반환되어야 합니다. 사용자가 기존 암호를 계속 사용하고자 하는 경우 Lambda 응답에서 `finalUserStatus = "CONFIRMED"` 속성을 설정합니다. 응답에 대해 필요한 속성은 [사용자 마이그레이션 Lambda 트리거 파라미터 \(p. 148\)](#) 단원을 참조하십시오.

Important

사용자 마이그레이션 Lambda 코드의 전체 요청 이벤트 객체를 로그하지 마십시오 (CloudWatch 로그로 전송된 로그 레코드의 암호를 포함하고 있기 때문). 그리고 암호가 기록되지 않도록 로그를 삭제해야 합니다.

5. Amazon Cognito는 사용자 풀에 사용자 프로필을 생성하고 앱 클라이언트로 토큰을 반환합니다.
6. 앱 클라이언트는 이제 로그인 이후 정상 기능으로 진행할 수 있습니다.

사용자가 마이그레이션된 이후 기본 모바일 앱이 로그인에 대해 `USER_SRP_AUTH` 흐름을 사용하는 것이 좋습니다. 이는 네트워크를 통해 암호를 전송할 필요 없이 보안 원격 암호(SRP) 프로토콜을 사용하여 사용자를 인증합니다. 그리고 마이그레이션 동안 사용된 `USER_PASSWORD_AUTH` 흐름에 대한 보안 이점을 제공합니다.

마이그레이션 도중 클라이언트 디바이스 또는 네트워크 문제를 포함한 오류가 발생한 경우 앱은 Amazon Cognito 사용자 풀 API로부터 오류 응답을 받고, 사용자 계정은 사용자 풀에 생성 또는 생성되지 않았을 수 있습니다. 사용자는 로그인을 다시 시도하고 반복적으로 실패하는 경우 앱의 암호 찾기 흐름을 사용하여 암호 재설정을 시도합니다.

암호 찾기 흐름은 유사한 방식으로 작동합니다. 단, 사용자 마이그레이션 Lambda 함수에 어떠한 암호도 제공되지 않고, 함수는 기존 사용자 디렉터리의 사용자만을 조회하며, 사용자 풀에 저장된 속성을 반환합니다. 이는 이메일 또는 SMS로 암호 재설정 코드를 사용자에게 전송하고, 그런 다음 사용자는 앱에 새 암호를 설정할 수 있습니다. 앱에 기본 UI가 있는 경우 암호 찾기 흐름에 대한 화면을 제공해야 합니다. 앱에서 호스팅된 UI를 사용하는 경우 UI 화면이 제공됩니다.

CSV 파일에서 사용자 풀로 사용자 가져오기

Amazon Cognito 사용자 풀로 사용자를 가져올 수 있습니다. .csv 형식의 특수한 파일에서 사용자 정보를 가져옵니다. 가져오기 프로세스에서 암호를 제외한 모든 사용자 속성의 값이 설정됩니다. 보안 모범 사례에 따라 암호를 일반 텍스트로 사용할 수 없고 가져오기 해시를 지원하지 않으므로 암호 가져오기는 지원되지 않습니다. 즉, 사용자가 처음 로그인할 때 암호를 변경해야 합니다. 따라서 이 메서드를 사용하여 가져오기하면 사용자들이 `RESET_REQUIRED` 상태가 됩니다.

Note

각 사용자의 생성일은 사용자를 사용자 풀로 가져온 시간입니다. 생성일은 가져온 속성 중 하나가 아닙니다.

기본 단계는 다음과 같습니다.

1. AWS Identity and Access Management(IAM) 콘솔에서 Amazon CloudWatch Logs 역할을 만듭니다.
2. 사용자 가져오기 .csv 파일을 만듭니다.
3. 사용자 가져오기 작업을 만들고 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 만들기 \(p. 173\)](#)
- [사용자 가져오기 .csv 파일 만들기 \(p. 174\)](#)
- [Amazon Cognito 사용자 풀 가져오기 작업 생성 및 실행 \(p. 176\)](#)
- [CloudWatch 콘솔에서 사용자 풀 가져오기 결과 보기 \(p. 180\)](#)
- [가져온 사용자에게 암호 재설정 요구 \(p. 181\)](#)

CloudWatch Logs IAM 역할 만들기

Amazon Cognito CLI 또는 API를 사용하는 경우 CloudWatch IAM 역할을 만들어야 합니다. 다음 절차에서는 Amazon Cognito가 사용자 풀 가져오기 작업에 대해 CloudWatch Logs에 정보를 기록하도록 허용하는 방법을 설명합니다.

Note

[Amazon Cognito 콘솔](#)을 사용하는 경우 콘솔에서 귀하를 위한 역할을 생성하므로 이 절차를 사용할 필요가 없습니다.

사용자 풀 가져오기를 위해 CloudWatch Logs IAM 역할을 만들려면

1. AWS Management 콘솔에 로그인한 다음 <https://console.aws.amazon.com/iam/>에서 IAM 콘솔을 엽니다.
2. [Roles]를 선택합니다.
3. 새 역할 생성을 선택합니다.
4. 역할 이름을 입력하고 다음 단계를 선택합니다.
5. Select Role Type(역할 유형 선택)에서 Amazon EC2를 선택합니다. 아무 역할 유형이나 선택할 수 있습니다. 나중에 나오는 단계에서 이 설정을 변경합니다. 처음부터 IAM 역할을 만들 수는 없고 기존의 IAM 역할을 템플릿으로 사용해서만 여기에 덮어써 필요한 역할을 만들어야 하기 때문입니다.
6. [Attach Policy]에서 [Next Step]을 선택합니다.
7. 검토에서 역할 생성을 선택합니다.
8. Roles(역할)에서 방금 만든 역할을 선택합니다.
9. Summary(요약)에서 Permissions(권한)를 선택합니다.
10. Permissions(권한) 탭에서 Inline Policies(인라인 정책)를 선택한 후 [click here](#)(여기 클릭)를 선택합니다.
11. Set Permissions(권한 설정)에서 custom policy(사용자 지정 정책)를 선택한 후 select(선택)를 선택합니다.
12. Review Policy(정책 검토)에 공백 없이 정책 이름을 입력하고 다음 텍스트를 복사하여 기존 텍스트 대신 역할 액세스 정책으로 붙여 넣습니다.

```
{
```

```
"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/*"
    ]
  }
]
```

13. [Apply Policy]를 선택합니다.
14. Summary(요약)에서 Trust Relationships(신뢰 관계) 탭을 선택합니다.
15. Edit Trust Relationship(신뢰 관계 편집)을 선택합니다.
16. 다음 신뢰 관계 텍스트를 복사하여 기존 텍스트 대신 Policy Document(정책 문서) 텍스트 상자에 붙여 넣습니다.

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

17. [Update Trust Policy(신뢰 정책 업데이트)]를 선택합니다. 이제 역할 만들기가 끝났습니다.
18. 역할 ARN을 기록해 둡니다. 나중에 가져오기 작업을 만들 때 이 역할 ARN이 필요합니다.

사용자 가져오기 .csv 파일 만들기

기존의 사용자를 사용자 풀로 가져오려면 먼저 입력으로 사용할 .csv 파일을 만들어야 합니다. 그러려면 사용자 가져오기 .csv 헤더 정보를 다운로드하여 [.csv 파일 형식 지정 \(p. 175\)](#)에 요약된 형식 요구 사항에 맞게 파일을 편집합니다.

AWS Management 콘솔을 사용하여 .csv 파일 헤더 다운로드

1. [Amazon Cognito 콘솔](#)로 이동하여 Manage User Pools(사용자 풀 관리)를 선택한 후 사용자를 가져올 사용자 풀을 선택합니다.
2. [Users] 탭을 선택합니다.
3. 사용자 가져오기를 선택합니다.
4. CSV 헤더 다운로드를 선택하여 .csv 파일에 포함할 헤더 행이 있는 .csv 파일을 가져옵니다.

CLI를 사용하여 .csv 파일 헤더 다운로드

정확한 헤더 목록을 얻으려면 다음 CLI 명령을 실행하십시오. 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 다음과 같습니다.

```
cognito:username,name,given_name,family_name,middle_name,nickname,preferred_username,profile,picture,we
```

이 헤더와 사용자의 속성 값을 포함하고 다음 규칙에 따라 형식이 지정되도록 .csv 파일을 편집해야 합니다.

Note

적절한 전화 번호 형식 등의 속성 값에 대한 자세한 내용은 [사용자 풀 속성 구성 \(p. 202\)](#) 단원을 참조하십시오.

- 파일 첫째 줄은 사용자 속성 이름을 포함한 다운로드된 헤더 행입니다.
- .csv 파일의 열 순서는 중요하지 않습니다.
- 첫째 줄 다음의 각 줄에는 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
 - email(email_verified가 true일 경우)

- phone_number(phone_number_verified가 true일 경우)
- 사용자 풀을 만들 때 필수로 표시한 모든 속성
- 사용자 풀에는 자동 확인 속성이 email_verified와 phone_number_verified 중 적어도 하나는 있어야 합니다. 각 사용자에게 대해 자동 확인 속성 중 적어도 하나는 true여야 합니다. 사용자 풀에 자동 확인 속성이 없으면 가져오기 작업이 시작되지 않습니다. 사용자 풀에 자동 확인 속성이 하나뿐이면 사용자마다 해당 속성을 확인해야 합니다. 예를 들어, 사용자 풀에 자동 확인 속성이 phone_number만 있으면 사용자마다 phone_number_verified 값이 true여야 합니다.

Note

사용자가 암호를 재설정하려면 확인된 이메일이나 전화 번호가 있어야 합니다. Amazon Cognito에서 .csv 파일에 지정된 이메일이나 전화 번호로 암호 재설정 코드가 포함된 메시지를 보냅니다. 전화 번호로 메시지를 보낼 경우 메시지가 SMS를 통해 전송됩니다.

- 문자열로 이루어진 속성 값을 인용 부호 안에 포함하면 안 됩니다.
- 속성 값에 쉼표가 있으면 쉼표 앞에 백슬래시(\)를 넣어야 합니다. .csv 파일에 있는 필드가 쉼표로 분리되기 때문입니다.
- .csv 파일 콘텐츠는 바이트 순서 표시가 없는 UTF-8 형식이어야 합니다.
- cognito:username 필드는 필수이며 사용자 풀에서 고유해야 합니다. 아무 유니코드 문자열이나 가능하지만 공백이나 탭을 포함할 수 없습니다.
- birthdate 값이 있을 경우 이 값은 mm/dd/yyyy 형식이어야 합니다. 예를 들어, 1985년 2월 1일인 birthdate는 02/01/1985로 인코딩되어야 합니다.
- cognito:mfa_enabled 필드는 필수입니다. 사용자 풀에서 MFA(멀티 팩터 인증)를 요구하도록 설정한 경우 이 필드는 모든 사용자에게 대해 true여야 합니다. MFA를 해제하도록 설정한 경우 이 필드는 모든 사용자에게 대해 false여야 합니다. MFA를 선택 사항으로 설정한 경우에는 이 필드가 true 또는 false일 수 있지만 비워 두면 안 됩니다.
- 최대 줄 길이는 16,000자입니다.
- 최대 .csv 파일 크기는 100MB입니다.
- 파일의 최대 줄 수(사용자)는 헤더를 제외하고 500,000개입니다.
- updated_at 필드 값은 초 단위 epoch 시간(예: 1471453471)이어야 합니다.
- 속성값에 있는 선행 또는 후행 공백은 잘립니다.

전체 샘플 사용자 가져오기 .csv 파일은 다음과 같습니다.

```
cognito:username,name,given_name,family_name,middle_name,nickname,preferred_username,profile,picture,we
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 단원에서는 Amazon Cognito 콘솔과 AWS Command Line Interface를 사용하여 사용자 풀 가져오기 작업을 만들고 실행하는 방법을 설명합니다.

주제

- [Amazon Cognito 콘솔을 사용하여 .csv 파일에서 사용자 가져오기 \(p. 176\)](#)
- [AWS CLI를 사용하여 사용자 가져오기 \(p. 177\)](#)

Amazon Cognito 콘솔을 사용하여 .csv 파일에서 사용자 가져오기

다음 절차에서는 .csv 파일에서 사용자를 가져오는 방법을 설명합니다.

Amazon Cognito 콘솔을 사용하여 .csv 파일에서 사용자를 가져오려면

1. 가져오기 작업 생성을 선택합니다.
2. 작업 이름을 입력합니다. 작업 이름에는 대/소문자(a-z, A-Z), 숫자(0-9) 및 특수 문자(+ = , . @ 및 -)를 포함할 수 있습니다.
3. 처음으로 사용자 가져오기 작업을 만드는 경우 AWS Management 콘솔에서 자동으로 IAM 역할을 만듭니다. 그렇지 않다면 IAM 역할 목록에서 기존 역할을 선택하거나 AWS Management 콘솔을 통해 새 역할을 만들 수 있습니다.
4. CSV 업로드를 선택하고 사용자를 가져올 .csv 파일을 선택합니다.
5. [Create job]을 선택합니다.
6. 작업을 시작하려면 시작을 선택합니다.

AWS CLI를 사용하여 사용자 가져오기

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

이 명령의 명령줄 옵션 목록을 가져오려면 help 명령줄 옵션을 사용하십시오. 예:

```
aws cognito-idp get-csv-header help
```

사용자 가져오기 작업 생성

.csv 파일을 만든 후 다음 CLI 명령을 사용하여 사용자 가져오기 작업을 만들 수 있습니다. 여기서 **JOB_NAME**은 작업을 위해 선택하는 이름이고 **USER_POOL_ID**는 전과 동일한 사용자 풀 ID이며 **ROLE_ARN**은 CloudWatch Logs IAM 역할 만들기 (p. 173)에서 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 반환된 **PRE_SIGNED_URL**은 15분간 유효합니다. 그 후에는 만료되므로 새로운 사용자 가져오기 작업을 만들어 새 URL을 가져와야 합니다.

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 응답에서 다음 Status 값 중 하나가 표시됩니다.

- "Created" - 작업이 생성되었지만 시작되지 않았습니다.
- "Pending" - 전환 상태입니다. 작업을 시작했지만 사용자 가져오기를 아직 시작하지 않았습니다.
- "InProgress" - 작업이 시작되었고 사용자를 가져오고 있습니다.
- "Stopping" - 작업을 중지했지만 작업이 사용자 가져오기를 아직 중지하지 않았습니다.
- "Stopped" - 작업을 중지했고 작업이 사용자 가져오기를 중지했습니다.
- "Succeeded" - 작업이 성공적으로 완료되었습니다.
- "Failed" - 오류로 인해 작업이 중지되었습니다.
- "Expired" - 작업을 만들었지만 24-48시간 안에 작업을 시작하지 않았습니다. 작업에 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 업로드하십시오.

```
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 찾아 보십시오. 이 문구는 파일이 성공적으로 업로드되었음을 나타냅니다.

사용자 가져오기 작업 설명

사용자 가져오기 작업의 설명을 가져오려면 다음 명령을 사용하십시오. 여기서 **USER_POOL_ID**는 사용자 풀 ID이고 **JOB_ID**는 사용자 가져오기 작업을 만들 때 반환된 작업 ID입니다.

```
aws cognito-idp describe-user-import-job --user-pool-id "USER_POOL_ID" --job-id "JOB_ID"
```

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  
  }  
}
```

이전의 샘플 출력에서 **PRE_SIGNED_URL**은 .csv 파일을 업로드한 URL입니다. **ROLE_ARN**은 역할을 만들 때 받은 CloudWatch Logs 역할 ARN입니다.

사용자 가져오기 작업 나열

사용자 가져오기 작업을 나열하려면 다음 명령을 사용하십시오.

```
aws cognito-idp list-user-import-jobs --user-pool-id "USER_POOL_ID" --max-results 2
```

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"
}
```

마지막에 만든 작업부터 처음 만든 작업까지 시간 순서대로 작업이 나열됩니다. 두 번째 작업 뒤의 **PAGINATION_TOKEN** 문자열은 이 목록 명령의 다른 결과가 더 있음을 나타냅니다. 결과를 더 나열하려면 다음과 같이 --pagination-token 옵션을 사용하십시오.

```
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 사용하십시오.

```
aws cognito-idp start-user-import-job --user-pool-id "USER_POOL_ID" --job-id "JOB_ID"
```

제공된 사용자 풀에 대해 한 번에 하나만 사용자 가져오기 작업을 활성화할 수 있습니다.

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 사용하십시오. 작업을 중지한 후에 다시 시작할 수 없습니다.

```
aws cognito-idp stop-user-import-job --user-pool-id "USER_POOL_ID" --job-id "JOB_ID"
```

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 \(p. 180\)](#)
- [결과 해석 \(p. 181\)](#)

결과 보기

다음 단계에서는 사용자 풀 가져오기 결과를 보는 방법을 설명합니다.

사용자 풀 가져오기 결과를 보려면

1. AWS Management 콘솔에 로그인한 다음 <https://console.aws.amazon.com/cloudwatch/>에서 CloudWatch 콘솔을 엽니다.

2. [Logs]를 선택합니다.
3. 사용자 풀 가져오기 작업의 로그 그룹을 선택합니다. 로그 그룹 이름은 `/aws/cognito/userpools/USER_POOL_ID/USER_POOL_NAME` 형식입니다.
4. 방금 실행한 사용자 가져오기 작업의 로그를 선택합니다. 로그 이름은 `JOB_ID/JOB_NAME` 형식입니다. 로그 안의 결과는 줄 번호에 따른 사용자를 나타냅니다. 사용자 데이터는 로그에 기록되지 않습니다. 사용자마다 다음과 유사한 줄이 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 사용자의 상태는 "PasswordReset"으로 설정됩니다.

다음과 같은 경우 사용자를 가져오지 않지만 가져오기 작업은 계속됩니다.

- 자동 확인 속성이 true로 설정되어 있지 않습니다.
- 사용자 데이터가 스키마와 일치하지 않습니다.
- 내부 오류로 인해 사용자를 가져올 수 없습니다.

다음과 같은 경우 가져오기 작업이 실패합니다.

- Amazon CloudWatch Logs 역할을 위임할 수 없으며 이 역할에 정확한 액세스 정책이 없거나 역할이 삭제되었습니다.
- 사용자 풀이 삭제되었습니다.
- Amazon Cognito에서 .csv 파일을 구문 분석할 수 없습니다.

가져온 사용자에게 암호 재설정 요구

가져온 사용자가 각각 처음으로 로그인할 때 다음과 같이 새로운 암호를 입력하도록 요구합니다.

가져온 사용자에게 암호 재설정 요구

1. 사용자 이름과 암호를 제공하여 사용자가 로그인을 시도합니다(GetAuthenticationDetails 또는 InitiateAuth를 통해).
2. Amazon Cognito에서 PasswordResetRequiredException을 반환합니다.
3. 다음 절차에서 설명하는 대로 앱에서 사용자를 ForgotPassword 흐름으로 안내합니다.
 - a. 앱에서 ForgotPassword(**username**)를 호출합니다.
 - b. Amazon Cognito에서 확인된 이메일이나 전화 번호(해당 사용자에 대해 csv 파일에 제공한 내용에 따라)로 코드를 전송하고 ForgotPassword 요청에 응답하여 코드가 전송된 위치를 앱에 알립니다.

Note

암호 재설정 코드를 보내려면 사용자 풀에 전화 번호나 이메일 확인이 있어야 합니다.

- c. 앱은 코드가 전송되었다는 사실과 코드가 전송된 위치를 사용자에게 알리고 코드와 새 암호를 입력하기 위한 UI를 제공합니다.
- d. 사용자가 코드와 새 암호를 앱에 입력합니다.
- e. 앱에서 ConfirmForgotPassword(**code**, **password**)를 호출합니다. 성공하면 새 암호가 설정됩니다.

- f. 이제 앱이 사용자를 로그인 페이지로 연결합니다.

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 내장되어 있는 [기본 이메일 기능 \(p. 182\)](#)
- [Amazon SES 구성 \(p. 182\)](#)

이러한 설정은 되돌릴 수 있습니다. 필요한 경우 사용자 풀을 업데이트하여 둘 사이를 전환할 수 있습니다.

기본 이메일 기능

서비스와 함께 제공되는 기본 이메일 기능을 사용하여 Amazon Cognito에서 이메일 전송을 처리하도록 허용할 수 있습니다. 기본 옵션을 사용하면 Amazon Cognito는 사용자 풀에 대해 매일 제한된 수의 이메일만 허용합니다. 특정 제한 정보는 [Amazon Cognito의 제한 값 \(p. 311\)](#) 항목을 참조하십시오. 일반적인 프로덕션 환경의 경우 기본 이메일 제한은 필수 전송 볼륨보다 적습니다. 더 많은 전송량을 활성화하려면 Amazon SES 이메일 구성을 사용해야 합니다.

기본 옵션을 사용하면 다음 이메일 주소 중 하나를 발신 주소로 사용할 수 있습니다.

- 기본 이메일 주소는 no-reply@verificationemail.com입니다.
- 본인이 소유한 사용자 지정 이메일 주소. 자신의 이메일 주소를 사용하려면 먼저 Amazon SES로 확인해야 하며 Amazon Cognito 권한을 부여하여 사용해야 합니다.

Amazon SES 이메일 구성

애플리케이션에 기본 옵션으로 제공되는 것보다 더 많은 전송 볼륨이 필요할 수도 있습니다. 더 많은 전송 볼륨을 활성화하려면 Amazon SES 구성을 사용하여 사용자에게 이메일을 보내도록 사용자 풀을 구성합니다.

Amazon SES 구성을 사용하려면 먼저 Amazon SES로 하나 이상의 이메일 주소를 확인해야 합니다. 확인된 이메일 주소를 사용자 풀에 할당한 발신 이메일 주소로 사용합니다. 그런 다음 Amazon Cognito에서 이메일을 보내면 사용자 대신 Amazon SES를 호출하여 이메일 주소를 사용합니다.

Amazon SES 구성을 사용하면 사용자 풀에 대한 이 이메일 전송 제한은 AWS 계정에서 확인된 Amazon SES 이메일 주소에 적용되는 것과 동일한 제한입니다.

사용자 풀에 대한 이메일 구성

사용자 풀에 대한 이메일 설정을 구성하려면 다음 단계를 완료하십시오. 사용하려는 설정에 따라 Amazon SES, AWS Identity and Access Management(IAM), Amazon Cognito로 단계를 완료해야 할 수도 있습니다.

Note

이러한 단계에서 생성하는 리소스는 AWS 계정에서 공유할 수 없습니다. 예를 들어 다른 계정에 있는 Amazon SES 이메일 주소로 한 계정에서 사용자 풀을 구성할 수 없습니다. 따라서 여러 계정에 Amazon Cognito를 사용하는 경우 각 계정에서 이러한 단계를 반복해야 합니다.

1단계: Amazon SES로 이메일 주소 확인

사용자 풀을 구성하기 전에 다음 중 하나를 수행하려면 Amazon SES로 하나 이상의 이메일 주소를 확인해야 합니다.

- 자신의 이메일 주소를 발신 주소로 사용합니다.
- Amazon SES 구성을 사용하여 이메일 전송을 처리합니다.

이메일 주소를 확인함으로써 이메일 주소를 소유하고 있음을 확인하면 무단 사용을 방지하는 데 도움이 됩니다.

이 절차의 단계는 Amazon Simple Email Service 개발자 안내서에서 [이메일 주소 확인](#) 항목을 참조하십시오.

2단계: Amazon SES 샌드박스에서 계정 이동

처음에 Amazon SES를 사용하기 시작하면 AWS 계정이 Amazon SES 샌드박스에 배치됩니다. Amazon SES는 사기 및 침해 방지하기 위해 샌드박스를 사용합니다. Amazon SES 구성을 사용하여 이메일 전송을 처리하는 경우 Amazon Cognito에서 사용자에게 이메일을 보내기 전에 AWS 계정을 샌드박스 밖으로 이동해야 합니다.

기본 Amazon Cognito 이메일 기능을 사용하는 경우 이 단계를 건너뛸 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. AWS 계정은 샌드박스에 있지만 프로덕션 환경에 있는 애플리케이션에는 Amazon SES 구성을 사용하지 마십시오. 이 경우 Amazon Cognito에서 사용자의 이메일 주소로 메시지를 보낼 수 없습니다.

샌드박스 밖으로 이동하는 단계는 Amazon Simple Email Service 개발자 안내서에서 [Amazon SES 샌드박스 밖으로 이동](#) 항목을 참조하십시오.

3단계: Amazon Cognito에 이메일 권한 부여

사용자에게 이메일을 보내기 전에 특정 권한을 Amazon Cognito에 부여해야 할 수도 있습니다. 부여하는 권한과 부여하는 프로세스는 기본 이메일 기능을 사용하는지 아니면 Amazon SES 구성을 사용하는지 여부에 따라 다릅니다.

기본 이메일 기능을 사용하기 위해 사용 권한을 부여하는 방법

기본 이메일 기능을 사용하도록 사용자 풀을 구성한 경우 다음 주소 중 하나를 Amazon Cognito 사용자에게 이메일을 보내는 발신 주소로 사용합니다.

- 기본 주소
- Amazon SES에서 확인된 주소이어야 하는 사용자 지정 주소

기본 이메일 주소를 사용하는 경우 Amazon Cognito에 추가 권한이 필요하지 않으므로 이 단계를 건너뛸 수 있습니다.

사용자 지정 주소를 사용하는 경우 Amazon Cognito에 추가 권한이 있어야 이 주소를 이메일 주소로 사용할 수 있습니다. 이러한 권한은 Amazon SES의 주소에 연결되는 전송 권한 부여 정책에 의해 부여됩니다. Amazon Cognito 콘솔을 사용하여 사용자 풀에 사용자 지정 주소를 추가하면 자동으로 정책이 연결됩니다. 그러나 예를 들어 AWS CLI 또는 Amazon Cognito API를 사용하여 콘솔 외부에서 사용자 풀을 구성하는 경우 정책을 직접 연결해야 합니다.

자세한 내용은 Amazon Simple Email Service 개발자 안내서의 [Amazon SES에서 전송 권한 부여 사용](#)을 참조하십시오.

Example 전송 권한 부여 정책

다음 예는 Amazon Cognito에서 Amazon SES로 확인된 이메일 주소를 사용하여 이메일 전송을 허용하는 전송 권한 부여 정책입니다.

```
{
  "Version": "2008-10-17",
  "Statement": [
    {
      "Sid": "stmt1234567891234",
      "Effect": "Allow",
      "Principal": {
        "Service": "cognito-idp.amazonaws.com"
      },
      "Action": [
        "ses:SendEmail",
        "ses:SendRawEmail"
      ],
      "Resource": "<your SES identity ARN>"
    }
  ]
}
```

이 예제에서 "Sid" 값은 명령문을 고유하게 식별하는 임의의 문자열입니다.

정책 구문에 대한 자세한 내용은 Amazon Simple Email Service 개발자 안내서의 [Amazon SES 전송 권한 부여 정책](#)을 참조하십시오.

자세한 내용은 Amazon Simple Email Service 개발자 안내서의 [Amazon SES 전송 권한 부여 정책 예제](#)를 참조하십시오.

Amazon SES 구성을 사용하기 위해 사용 권한을 부여하는 방법

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 권한 부여는 IAM 서비스와 함께 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 Cognito에서 이 역할을 생성할 수 있기 전에 사용자 풀을 설정하는 데 사용하는 IAM 권한은 iam:CreateServiceLinkedRole 작업을 포함해야 합니다. IAM에서 권한을 업데이트하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 [IAM 사용자의 권한 변경](#) 항목을 참조하십시오.

Amazon Cognito에서 생성하는 서비스 연결 역할에 대한 자세한 내용은 [Amazon Cognito에 서비스 연결 역할 사용 \(p. 338\)](#) 항목을 참조하십시오.

4단계: 사용자 풀 구성

다음 중 하나를 사용하여 사용자 풀을 구성하려면 다음 단계를 완료하십시오.

- 이메일 발신자로 표시되는 사용자 지정 발신 주소
- 사용자가 발신 주소로 보낸 메시지를 받는 사용자 지정 회신 주소
- Amazon SES 구성

기본 Amazon Cognito 이메일 기능 및 주소를 사용하는 경우 이 절차를 완료할 필요가 없습니다.

이메일 주소 구성 방법

1. AWS Management 콘솔에 로그인한 다음 <https://console.aws.amazon.com/cognito>에서 Amazon Cognito 콘솔을 엽니다.
2. Manage User Pools(사용자 풀 관리)를 선택합니다.
3. 사용자 풀 페이지에서 구성할 사용자 풀을 선택합니다.
4. 왼쪽의 탐색 메뉴에서 메시지 사용자 지정을 선택합니다.
5. 사용자 지정 발신 주소를 사용하는 경우 사용자 지정 발신 주소 추가를 선택하고 다음을 수행하십시오.
 - a. SES 리전에서 확인된 이메일 주소가 포함된 리전을 선택합니다.
 - b. 발신 이메일 주소에서 이메일 주소를 선택합니다. Amazon Cognito 콘솔을 사용하면 선택한 리전에서 Amazon SES로 확인한 이메일 주소만 선택할 수 있습니다.
6. Amazon SES 구성을 통해 이메일을 보내시겠습니까? 아래에서 예 - Amazon SES를 사용합시다 또는 아니오 - Cognito(기본값)을 사용합시다를 선택합니다.

Amazon SES을 사용하도록 선택하면 변경 사항을 저장 한 후에 Amazon Cognito에서 서비스 연결 역할을 생성합니다.

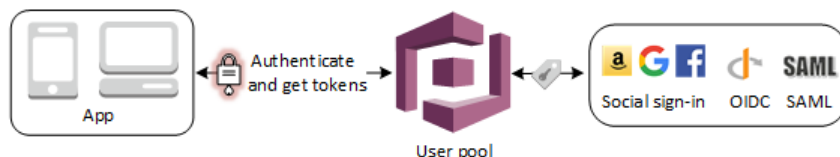
7. 사용자 지정 회신 주소를 사용하는 경우 사용자 지정 회신 주소 추가를 선택합니다. 그런 다음 사용자가 발신 주소로 보내는 메시지를 받으려는 이메일 주소를 지정합니다.
8. 이메일 계정 옵션 설정을 마치면 변경 사항 저장을 선택합니다.

또한 메시지 사용자 지정 페이지에서는 [확인 메시지 사용자 지정 \(p. 211\)](#) 및 [초청 메시지 사용자 지정 \(p. 211\)](#) 옵션을 제공합니다.

사용자 풀을 사용한 인증

앱 사용자는 사용자 풀을 통해 직접 로그인하거나 타사 자격 증명 공급자(IdP)를 통해 연동 로그인할 수 있습니다. 사용자 풀에서는 Facebook, Google 및 Amazon을 통한 소셜 로그인에서 반환된 토큰과 OpenID Connect(OIDC) 및 SAML IdP에서 반환된 토큰의 처리 작업을 관리합니다.

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 이를 AWS 자격 증명으로 교환하여 기타 AWS 서비스에 액세스할 수 있습니다.



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 Android용 Mobile SDK는 ID 및 액세스 토큰을 자동으로 새로 고칩니다. SDK와 JavaScript, Android 및 iOS용 샘플 코드에 대한 자세한 내용은 [Amazon Cognito 사용자 풀 SDK](#)를 참조하십시오.

앱 사용자가 성공적으로 로그인하고 나면 Amazon Cognito는 인증된 사용자에게 세션을 생성하고 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

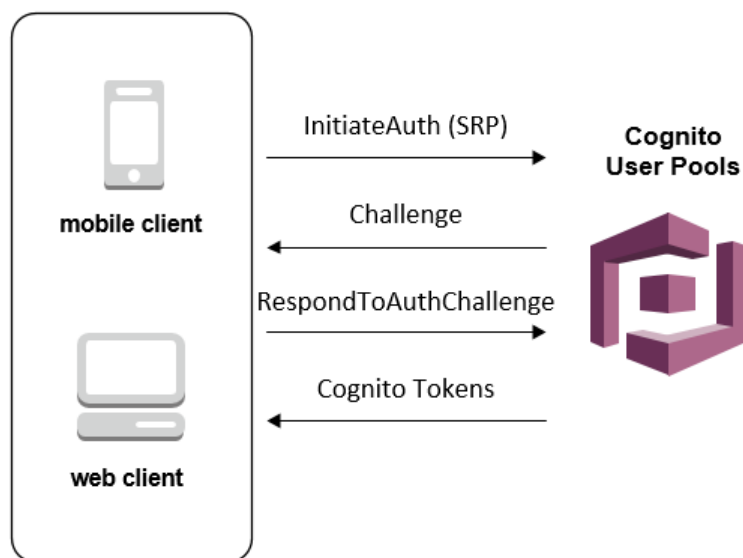
- [사용자 풀 인증 흐름 \(p. 187\)](#)

- 사용자 풀을 통해 토큰 사용 (p. 190)

사용자 풀 인증 흐름

현대식 인증 흐름에는 사용자의 자격 증명을 확인하기 위해 암호 외에도 새로운 챌린지 유형이 통합되어 있습니다. 인증은 두 가지 공통 단계로 일반화되며, 이러한 단계는 두 API인 `InitiateAuth` 및 `RespondToAuthChallenge`를 통해 구현됩니다.

인증이 실패하거나 사용자에게 토큰이 발행될 때까지 사용자는 연속 챌린지에 응답하여 인증합니다. Amazon에서는 서로 다른 챌린지를 포함하기 위해 반복 실행되는 이러한 두 단계를 통해 사용자 지정 인증 흐름을 지원할 수 있습니다.



AWS Lambda 트리거를 사용하여 인증 흐름을 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 마이그레이션이 가능합니다.

주제

- 클라이언트 측 인증 흐름 (p. 187)
- 서버 측 인증 흐름 (p. 188)
- 사용자 지정 인증 흐름 (p. 188)
- 관리 인증 흐름 (p. 189)
- 사용자 마이그레이션 인증 흐름 (p. 190)

클라이언트 측 인증 흐름

Android용 [AWS Mobile SDK](#), [AWS Mobile SDK for iOS](#), 또는 JavaScript용 [AWS SDK for JavaScript](#)를 사용하여 생성된 최종 사용자 클라이언트 측 앱에서 사용자 풀 인증의 작동 방식은 다음과 같습니다.

1. 사용자가 사용자 이름 및 암호를 앱에 입력합니다.
2. 앱이 사용자의 사용자 이름 및 SRP 세부 정보를 사용하여 `InitiateAuth` 메서드를 호출합니다.

이 메서드는 인증 파라미터를 반환합니다.

Note

앱이 Android, iOS 및 JavaScript SDK에서 Amazon Cognito SRP 지원을 사용하여 SRP 세부 정보를 생성합니다.

3. 앱에서 RespondToAuthChallenge 메서드를 호출합니다. 메서드 호출에 성공하면 메서드가 사용자 토큰을 반환하고 인증 흐름이 완료됩니다.

또 다른 챌린지가 필요한 경우 토큰이 반환되지 않습니다. 대신, RespondToAuthChallenge를 호출하면 세션이 반환됩니다.

4. RespondToAuthChallenge가 세션을 반환하는 경우 이번에는 앱이 세션 및 챌린지 응답(예: MFA 코드)과 함께 RespondToAuthChallenge를 다시 호출합니다.

서버 측 인증 흐름

최종 사용자 앱이 없지만, 대신 Java, Ruby 또는 Node.js 보안 백엔드 또는 서버 측 앱을 사용 중인 경우 Amazon Cognito 사용자 풀에 대해 인증된 서버 측 API를 사용할 수 있습니다.

서버 측 앱의 경우, 사용자 풀 인증은 클라이언트 측 앱과 비슷합니다. 단, 다음은 예외입니다.

- 서버 측 앱은 AdminInitiateAuth API(InitiateAuth 대신)를 호출합니다. 이 메서드에는 AWS 관리자 자격 증명도 필요합니다. 이 메서드는 인증 파라미터를 반환합니다.
- 인증 파라미터를 지정하고 나면 앱에서 AdminRespondToAuthChallenge API(RespondToAuthChallenge 대신)를 호출하는데, 이때는 AWS 관리자 자격 증명도 필요합니다.

AdminInitiateAuth 및 AdminRespondToAuthChallenge API는 다음 중 하나를 수행하여 명시적으로 활성화하지 않는 경우 관리자 로그인에 대해 사용자 이름 및 암호 사용자 자격 증명을 허용할 수 없습니다.

- 서버 측 앱의 CreateUserPoolClient 또는 UpdateUserPoolClient 호출에서 ExplicitAuthFlow 파라미터에 ADMIN_NO_SRP_AUTHENTICATION를 전달합니다.
- 사용자 풀 생성의 앱 클라이언트에서 서버 기반 인증용 로그인 API 활성화 (ADMIN_NO_SRP_AUTH)를 선택합니다. 자세한 내용은 [사용자 풀 앱 클라이언트 구성 \(p. 214\)](#) 단원을 참조하십시오.

사용자 지정 인증 흐름

Amazon Cognito 사용자 풀은 서비스는 사용자 지정 인증 흐름도 활성화하는데, 이 인증 흐름을 통해 AWS Lambda 트리거를 사용하여 챌린지 응답 기반 인증 모델을 쉽게 생성할 수 있습니다.

사용자 지정 인증 흐름은 여러 요구 사항에 맞추어 사용자 지정할 수 있는 일련의 챌린지 및 응답 사이클을 허용하도록 고안되었습니다. 이 흐름은 사용할 인증 유형을 지시하고 초기 인증 파라미터를 제공하는 InitiateAuth API를 호출하면서 시작됩니다. Amazon Cognito는 다음 중 하나를 사용하여 InitiateAuth 호출에 응답합니다.

- 사용자가 로그인한 경우 ID, 액세스 및 새로 고침 토큰
- 세션 및 파라미터와 함께 사용자에게 대한 챌린지
- 사용자가 인증에 실패할 경우 오류

Amazon Cognito가 챌린지를 통해 InitiateAuth 호출에 응답할 경우 앱은 입력을 더 수집하고 RespondToAuthChallenge API를 호출하여 챌린지 응답을 제공하고 세션을 다시 전달합니다. Amazon Cognito는 InitiateAuth 호출과 비슷하게 RespondToAuthChallenge 호출에 응답하여 사용자가 로그인한 경우 토큰을 제공하거나 또 다른 챌린지나 오류를 제공합니다. 또 다른 챌린지가 반환되면 사용자가 로그인하거나 오류가 반환될 때까지 RespondToAuthChallenge를 호출하여 앱에서 시퀀스가 반복됩니다. 자세한 내용은 InitiateAuth 및 RespondToAuthChallenge API의 API 설명서에 나와 있습니다.

Amazon Cognito에는 표준 인증을 위한 AuthFlow 및 ChallengeName 값이 기본으로 제공되어 SRP(Secure Remote Password) 프로토콜을 통해 사용자 이름과 암호를 확인합니다. 이 흐름은 iOS, Android 및 JavaScript SDK에 내장되어 있습니다. 상위 수준에서는 AuthParameters에 있는 USERNAME 및 SRP_A 값과 함께 USER_SRP_AUTH를 AuthFlow로 InitiateAuth에 보내면서 플로우가 시작됩니다. 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. RespondToAuthChallenge 호출이 성공하고 사용자가 로그인하면 토큰이 반환됩니다. 사용자에게 대해 MFA(멀티 팩터 인증)가 활성화된 경우 SMS_MFA의 ChallengeName이 반환되고 앱은 RespondToAuthChallenge를 또 호출하여 필요한 코드를 제공할 수 있습니다.

앱은 CUSTOM_AUTH와 함께 InitiateAuth를 Authflow로 호출하여 사용자 지정 인증 흐름을 시작할 수 있습니다. 사용자 지정 인증 흐름을 사용하면 트리거를 통해 챌린지 및 응답의 확인이 제어됩니다. DefineAuthChallenge Lambda 트리거는 이전 챌린지와 응답의 세션 어레이를 입력으로 사용하고, 사용자가 인증되거나(권한이 부여된 토큰이어야 함) 인증이 실패한 경우 다음 챌린지 이름과 부울을 출력합니다. 이 트리거는 챌린지를 통해 사용자의 경로를 제어하는 상태 시스템입니다. CreateAuthChallenge Lambda 트리거는 챌린지 이름을 입력으로 사용하고 챌린지 및 파라미터를 생성하여 응답을 평가합니다. CreateAuthChallenge는 DefineAuthChallenge가 CUSTOM_CHALLENGE를 다음 챌린지로 반환되고 다음 챌린지 유형이 챌린지 메타데이터 파라미터에 전달될 때 호출됩니다. VerifyAuthChallengeResponse Lambda 함수가 응답을 평가하고 부울을 반환하여 응답이 유효한지 여부를 나타냅니다.

또한 사용자 지정 인증 흐름은 SRP 암호 및 SMS를 통한 MFA와 같은 기본 제공 챌린지와 CAPTCHA 또는 본인 확인 지문과 같은 사용자 지정 챌린지를 함께 사용할 수 있습니다. 사용자 지정 인증 흐름에 SRP를 포함하려면 SRP를 시작해야 합니다. SRP 암호 확인을 시작하기 위해 DefineAuthChallenge Lambda 트리거가 챌린지 이름으로 SRP_A를 반환하고 인증 파라미터 맵의 SRP_A를 반환합니다. 암호가 확인되면 DefineAuthChallenge Lambda 트리거가 이전 챌린지 어레이의 PASSWORD_VERIFIER로 다시 호출됩니다. 사용자에게 대해 활성화된 경우 MFA가 자동으로 수행됩니다.

Lambda 트리거에 대한 자세한 내용은 [Lambda 트리거를 사용하여 사용자 풀 워크플로우 사용자 지정 \(p. 118\)](#) 단원을 참조하십시오.

Note

Amazon Cognito 호스팅 로그인 웹 페이지는 사용자 지정 인증 흐름을 지원하지 않습니다.

관리 인증 흐름

[사용자 지정 인증 흐름 \(p. 188\)](#)에서 설명한 API는 암호 확인을 위해 SRP가 사용되며 권장되는 인증 방식입니다. iOS, Android 및 JavaScript SDK는 이 방식을 기반으로 하며 SRP를 쉽게 사용하도록 해줍니다. 그러나 SRP 계산을 원치 않는 경우 보안 백엔드 서버용으로 고안된 관리 API를 대신 사용할 수 있습니다. 이 백엔드 관리 구현에는 InitiateAuth 대신 AdminInitiateAuth가 사용되고 RespondToAuthChallenge 대신 AdminRespondToAuthChallenge가 사용됩니다. 이 API를 사용할 때는 암호를 일반 텍스트로 제출할 수 있어 SRP 계산을 필요 없습니다. 예,

```
AdminInitiateAuth Request {
  "AuthFlow": "ADMIN_NO_SRP_AUTH",
  "AuthParameters": {
    "USERNAME": "<username>",
    "PASSWORD": "<password>"
  },
  "ClientId": "<clientId>",
  "UserPoolId": "<userPoolId>"
}
```

이 관리 인증 API에는 개발자 자격 증명이 필요하며 AWS 서명 버전 4(SigV4) 서명 프로세스가 사용됩니다. 이 API는 Node.js를 포함하여 Lambda 함수에 편리하게 쓸 수 있는 표준 AWS SDK에서 사용할 수 있습니다. 이 API를 사용하고 일반 텍스트로 된 암호를 허용하도록 하려면 콘솔을 사용하거나 CreateUserPoolClient 또는 UpdateUserPoolClient 호출에서 ExplicitAuthFlow 파라미터에 대해 ADMIN_NO_SRP_AUTH를

전달하여 앱에서 사용할 수 있게 설정해야 합니다. `InitiateAuth` 및 `RespondToAuthChallenge` API에는 `ADMIN_NO_SRP_AUTH` AuthFlow가 허용되지 않습니다.

`AdminInitiateAuth` 응답 `ChallengeParameters`에 `USER_ID_FOR_SRP` 속성이 있으면 별칭(이메일 주소 또는 전화 번호)이 아닌 사용자의 실제 사용자 이름이 이 속성에 포함됩니다. `AdminRespondToAuthChallenge`를 호출할 때 `ChallengeResponses`에서 `USERNAME` 파라미터에 있는 이 사용자 이름을 전달해야 합니다.

Note

관리 인증 흐름은 백엔드 관리 구현을 위한 것이므로 장치 추적을 지원하지 않습니다. 장치 추적이 활성화되면 관리 인증이 성공하지만 액세스 토큰을 새로 고치기 위한 호출은 실패합니다.

사용자 마이그레이션 인증 흐름

사용자 마이그레이션 Lambda 트리거는 기존 사용자 관리 시스템에서 사용자 풀로의 손쉬운 마이그레이션을 허용합니다. 사용자 마이그레이션 도중 암호 재설정을 피하려면 `USER_PASSWORD_AUTH` 인증 흐름을 선택하여 인증 도중 암호화된 SSL 연결을 통해 서비스로 사용자 암호를 전송합니다.

모든 사용자 마이그레이션을 완료했을 때 네트워크를 통해 어떠한 암호도 전송하지 않는 더욱 안전한 보안 SRP 흐름으로 흐름을 전환하는 것이 좋습니다.

Lambda 트리거에 대해 자세히 알아보려면 [Lambda 트리거를 사용하여 사용자 풀 워크플로우 사용자 지정 \(p. 118\)](#) 단원을 참조하십시오.

Lambda 트리거를 사용한 사용자 마이그레이션에 대한 자세한 내용은 [사용자 마이그레이션 Lambda 트리거를 사용하여 사용자 풀로 사용자 가져오기 \(p. 171\)](#) 단원을 참조하십시오.

사용자 풀을 통해 토큰 사용

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 임시 AWS 자격 증명으로 기타 AWS 서비스에 액세스할 수 있도록 이 토큰을 교환할 수 있습니다. [일반적인 Amazon Cognito 시나리오 \(p. 6\)](#) 단원을 참조하십시오.



웹 또는 모바일 앱에서의 사용자 풀 토큰 처리 및 관리는 Amazon Cognito SDK를 통해 클라이언트 측에서 제공됩니다. SDK와 JavaScript, Android 및 iOS용 샘플 코드에 대한 자세한 내용은 [Amazon Cognito 사용자 풀 SDK](#)를 참조하십시오. 서버 측 API 처리를 위해 토큰을 수동으로 처리해야 하거나 다른 프로그래밍 언어를 사용하고 있는 경우, 다양한 라이브러리를 활용하여 JWT를 디코딩 및 확인할 수 있습니다. [JWT 토큰 작업을 위한 OpenID Foundation 라이브러리 목록](#)을 참조하십시오.

OpenID Connect(OIDC) 개방형 표준에 정의된 바와 같이 Amazon Cognito 사용자 풀은 ID, 액세스 및 새로 고침 토큰을 구현합니다.

- **ID 토큰**에는 `name`, `email` 및 `phone_number` 같이 인증된 사용자의 자격 증명에 대한 클레임이 포함되어 있습니다.
- **액세스 토큰**은 인증된 리소스에 대한 액세스 권한을 부여합니다.
- **새로 고침 토큰**에는 새 ID나 액세스 토큰을 얻는 데 필요한 정보가 포함되어 있습니다.

Important

전송 중인 모든 토큰을 보호하고 애플리케이션의 컨텍스트에 저장하는 것이 좋습니다.

OIDC에 대한 자세한 내용은 [OpenID Connect\(OIDC\) 사양](#)을 참조하십시오.

주제

- [ID 토큰 사용](#) (p. 191)
- [액세스 토큰 사용](#) (p. 192)
- [새로 고침 토큰 사용](#) (p. 194)
- [사용자에 대한 모든 토큰 취소](#) (p. 194)
- [JSON 웹 토큰 확인](#) (p. 195)

ID 토큰 사용

ID 토큰은 name, email, phone_number 같이 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 대해 사용자를 인증하는 경우에도 ID 토큰을 사용할 수 있습니다. 사용자의 웹 API에 대해 ID 토큰이 애플리케이션의 외부에서 사용될 경우 ID 토큰의 서명을 확인해야 ID 토큰 내의 모든 클레임을 신뢰할 수 있습니다. [JSON 웹 토큰 확인](#) (p. 195) 단원을 참조하십시오.

ID 토큰은 사용자가 인증하고 한 시간이 지나면 만료됩니다. 만료된 후에는 클라이언트 또는 웹 API에서 ID 토큰을 처리해서는 안 됩니다.

OIDC 표준 클레임에 대한 자세한 내용은 [OIDC 표준 클레임](#)을 참조하십시오.

ID 토큰 헤더

헤더에는 키 ID(kid) 및 알고리즘(alg) 등 두 가지 정보가 포함되어 있습니다.

```
{
  "kid" : "1234example="
  "alg" : "RS256",
}
```

키 ID(kid)

kid 파라미터는 토큰의 JSON 웹 서명(JWS)을 보호하는 데 어떤 키가 사용되었는지 나타내는 힌트입니다.

kid 파라미터에 대한 자세한 내용은 [키 식별자\(kid\) 헤더 파라미터](#)를 참조하십시오.

알고리즘(alg)

alg 파라미터는 ID 토큰을 보호하는 데 사용되는 암호화 알고리즘을 나타냅니다. 사용자 풀은 SHA-256에서의 RSA 서명인 RS256 암호화 알고리즘을 사용합니다.

alg 파라미터에 대한 자세한 내용은 [알고리즘\(alg\) 헤더 파라미터](#)를 참조하십시오.

ID 토큰 페이로드

이는 ID 토큰에서 나온 샘플 페이로드입니다. 여기에는 인증된 사용자에 대한 클레임이 포함되어 있습니다. OIDC 표준 클레임에 대한 자세한 내용은 [OIDC 표준 클레임](#)을 참조하십시오.

```
{
  "sub": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
  "aud": "xxxxxxxxxxxxexample",
  "email_verified": true,
  "token_use": "id",
}
```

```
{
  "auth_time": 1500009400,
  "iss": "https://cognito-idp.us-east-1.amazonaws.com/us-east-1_example",
  "cognito:username": "janedoe",
  "exp": 1500013000,
  "given_name": "Jane",
  "iat": 1500009400,
  "email": "janedoe@example.com"
}
```

제목(sub)

sub 클레임은 인증된 사용자의 고유 식별자(UUID)입니다. 사용자 이름과 동일하지 않으므로 고유하지 않을 수도 있습니다.

발행자(iss)

iss 클레임은 다음과 같은 형식을 가집니다.

`https://cognito-idp.{region}.amazonaws.com/{userPoolId}`.

예를 들어 us-east-1 리전에서 사용자 풀을 생성했고 사용자 풀 ID가 u123456인 경우에는 사용자 풀의 사용자에게 발급된 ID 토큰에서 iss 클레임의 값이

`https://cognito-idp.us-east-1.amazonaws.com/u123456`입니다.

대상(aud)

aud 클레임에는 사용자 인증에 사용되는 client_id를 포함되어 있습니다.

토큰 사용(token_use)

The token_use 클레임은 이 토큰의 용도를 설명합니다. 이 값은 ID 토큰의 경우 항상 id입니다.

인증 시간(auth_time)

auth_time 클레임에는 인증이 발생했던 시간이 포함되어 있습니다. 이 값은 1970-01-01T0:0:0Z부터 해당 날짜/시간까지 UTC 형식으로 측정된 초 시간을 나타내는 JSON 번호입니다. 새로 고침 시, 이 값은 토큰이 발행된 시간이 아니라 원래 인증이 발생했던 시간을 나타냅니다.

ID 토큰에는 [OIDC 표준 클레임](#)에 정의된 OpenID Connect(OIDC) 표준 클레임이 포함될 수 있습니다. 또한 사용자 풀에 정의된 사용자 지정 속성도 포함될 수 있습니다.

Note

사용자 풀의 사용자 지정 속성은 항상 custom: 접두사로 시작합니다.

ID 토큰 서명

ID 토큰의 서명은 JWT 토큰의 헤더와 페이로드에 따라 계산됩니다. 웹 API에 있는 애플리케이션의 외부에서 사용될 경우 해당 토큰을 수락하기 전에 항상 이 서명을 확인해야 합니다. [JSON 웹 토큰 확인 \(p. 195\)](#) 단원을 참조하십시오.

액세스 토큰 사용

이 사용자 풀 액세스 토큰에는 인증된 사용자에 대한 클레임이 포함되어 있지만, ID 토큰과 다르게 자격 증명 정보는 포함되어 있지 않습니다. 액세스 토큰의 주 목적은 사용자 풀의 사용자 컨텍스트에서 API 작업을 승인하는 것입니다. 예를 들어 액세스 토큰을 사용하여 사용자 속성을 추가, 변경 또는 삭제하기 위한 액세스 권한을 사용자에게 부여할 수 있습니다. 액세스 토큰은 액세스 제어를 결정하고 사용자의 작업을 승인하기 위해 웹 API와 함께 사용될 수도 있습니다.

액세스 토큰도 JWT(JSON Web Token)로 표시됩니다. 액세스 토큰의 헤더는 ID 토큰과 구조가 동일하지만, ID 토큰과 액세스 토큰을 서명하는 데 다른 키가 사용되므로 키 ID(kid)는 다릅니다. ID 토큰에서와 마찬가지로

지로 먼저 웹 API에서 액세스 토큰의 서명을 확인해야 모든 클레임을 신뢰할 수 있습니다. [JSON 웹 토큰 확인 \(p. 195\)](#) 단원을 참조하십시오.

액세스 토큰은 사용자가 성공적으로 인증을 하고 한 시간이 지나면 만료됩니다. 만료된 후에는 이 토큰을 사용해서는 안 됩니다.

액세스 토큰 헤더

헤더에는 키 ID(kid) 및 알고리즘(alg) 등 두 가지 정보가 포함되어 있습니다.

```
{
  "kid" : "1234example="
  "alg" : "RS256",
}
```

키 ID(kid)

kid 파라미터는 토큰의 JSON 웹 서명(JWS)을 보호하는 데 어떤 키가 사용되었는지 나타내는 힌트입니다.

kid 파라미터에 대한 자세한 내용은 [키 식별자\(kid\) 헤더 파라미터](#)를 참조하십시오.

알고리즘(alg)

alg 파라미터는 액세스 토큰을 보호하는 데 사용되는 암호화 알고리즘을 나타냅니다. 사용자 풀은 SHA-256에서의 RSA 서명인 RS256 암호화 알고리즘을 사용합니다.

alg 파라미터에 대한 자세한 내용은 [알고리즘\(alg\) 헤더 파라미터](#)를 참조하십시오.

액세스 토큰 페이로드

액세스 토큰에서 나온 샘플 페이로드입니다. 자세한 내용은 [JWT 클레임](#)을 참조하십시오.

```
{
  "auth_time": 1500009400,
  "exp": 1500013000,
  "iat": 1500009400,
  "iss": "https://cognito-idp.us-east-1.amazonaws.com/us-east-1_example",
  "scope": "aws.cognito.signin.user.admin",
  "sub": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
  "token_use": "access",
  "username": "janedoe@example.com"
}
```

제목(sub)

sub 클레임은 인증된 사용자의 고유 식별자(UUID)입니다. 사용자 이름과 동일하지 않으므로 고유하지 않을 수도 있습니다.

발행자(iss)

iss 클레임은 다음과 같은 형식을 가집니다.

`https://cognito-idp.{region}.amazonaws.com/{userPoolId}`.

토큰 사용(token_use)

The token_use 클레임은 이 토큰의 용도를 설명합니다. 이 값은 액세스 토큰의 경우 항상 access입니다.

인증 시간(auth_time)

auth_time 클레임에는 인증이 발생했던 시간이 포함되어 있습니다. 이 값은 1970-01-01T0:0:0Z부터 해당 날짜/시간까지 UTC 형식으로 측정된 초 시간을 나타내는 JSON 번호입니다. 새로 고침 시, 이 값은 토큰이 발행된 시간이 아니라 원래 인증이 발생했던 시간을 나타냅니다.

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. [JSON 웹 토큰 확인\(p. 195\)](#) 단원을 참조하십시오.

새로 고침 토큰 사용

새로 고침 토큰을 사용하여 새 ID와 액세스 토큰을 검색할 수 있습니다.

기본적으로 새로 고침 토큰은 앱 사용자가 사용자 풀에 로그인하고 30일이 지나면 만료됩니다. 사용자 풀에 대한 앱을 생성할 경우 앱의 새로 고침 토큰 만료(일수)를 1에서 3650 사이의 값으로 설정할 수 있습니다.

유효한(기간이 만료되지 않은) 새로 고침 토큰이 존재하고 ID 및 액세스 토큰의 유효 기간이 최소 5분 남아 있는 경우, Mobile SDK for iOS 및 Android용 Mobile SDK는 ID 및 액세스 토큰을 자동으로 새로 고칩니다. 새로 고침 토큰의 기간이 만료되면 앱 사용자가 사용자 풀에 다시 로그인하여 재인증을 해야 합니다.

Note

Android용 Mobile SDK는 ID 및 액세스 토큰의 최소 유효 기간을 0 ~ 30분 사이의 값으로 변경할 수 있는 옵션을 제공합니다. Android용 AWS Mobile SDK API 참조의 [CognitoIdentityProviderClientConfig](#)의 setRefreshThreshold() 메서드를 참조하십시오.

새 계정에 대한 UnusedAccountValidityDays 시간 제한 전에 사용자가 한 번 이상 로그인하는 한, 사용자 계정 자체는 만료되지 않습니다.

새로 고침 토큰을 사용하여 사용자 풀 API에서 새 ID와 액세스 토큰을 얻으려면 [AdminInitiateAuth](#) 또는 [InitiateAuth](#) 메서드를 사용하십시오. AuthFlow 파라미터에 대해 REFRESH_TOKEN_AUTH를 전달합니다. 권한 부여 파라미터 AuthParameters는 키-값 맵으로, 여기서 키는 "REFRESH_TOKEN"이고 값은 실제 새로 고침 토큰입니다. Amazon Cognito는 새 ID와 액세스 토큰으로 응답합니다.

사용자에 대한 모든 토큰 취소

사용자는 GlobalSignOut 및 AdminUserGlobalSignOut API를 사용하여 사용자의 모든 토큰을 취소할 경우 현재 로그인되어 있는 모든 장치에서 로그아웃할 수 있습니다. 사용자가 로그아웃하면 다음 사항이 발생합니다.

- 사용자의 새로 고침 토큰을 사용하여 사용자에 대한 새 토큰을 가져올 수 없습니다.
- 사용자 풀 서비스에 대해 사용자의 액세스 토큰을 사용할 수 없습니다.
- 새 토큰을 가져오려면 사용자가 다시 인증해야 합니다.

앱에서 GlobalSignOut API를 사용하면 개별 사용자가 모든 장치에서 로그아웃할 수 있습니다. 일반적으로 앱에서는 이 옵션이 Sign out from all devices(모든 디바이스에서 로그아웃)와 같이 선택 사항으로 제공됩니다. 앱에서는 사용자의 유효하고, 만료되지 않았으며, 취소할 수 없는 액세스 토큰을 사용하여 이 메서드를 호출해야 합니다. 사용자는 이 메서드를 사용하여 다른 사용자를 로그아웃할 수 없습니다.

관리자 앱에서 AdminUserGlobalSignOut API를 사용하면 관리자가 모든 장치에서 사용자를 로그아웃할 수 있습니다. 관리자 앱에서는 AWS 개발자 자격 증명을 사용하여 이 메서드를 호출하고 사용자 풀 ID와 사용자 이름을 파라미터로 전달해야 합니다. AdminUserGlobalSignOut API를 사용하면 사용자 풀의 모든 사용자를 로그아웃할 수 있습니다.

JSON 웹 토큰 확인

이러한 단계들은 사용자 풀 JSON 웹 토큰(JWT)의 확인 방법을 설명합니다.

주제

- 사전 조건 (p. 195)
- 1단계: JWT 구조 확인 (p. 195)
- 2단계: JWT 서명 검증 (p. 195)
- 3단계: 클레임 확인 (p. 197)

사전 조건

- 이 단원에서는 라이브러리, SDK 또는 소프트웨어 프레임워크에서 이미 처리되었을 수 있는 작업을 설명합니다. 예를 들어 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 Android용 Mobile SDK는 ID 및 액세스 토큰을 자동으로 새로 고칩니다. SDK와 JavaScript, Android 및 iOS용 샘플 코드에 대한 자세한 내용은 [Amazon Cognito 사용자 풀 SDK](#)를 참조하십시오.
- 서버 측 API 처리를 위해 토큰을 수동으로 처리해야 하거나 다른 프로그래밍 언어를 사용하고 있는 경우, 다양한 라이브러리를 활용하여 JWT를 디코딩 및 확인할 수 있습니다. [JWT 토큰 작업을 위한 OpenID Foundation 라이브러리 목록](#)을 참조하십시오.

1단계: JWT 구조 확인

JSON 웹 토큰(JWT)은 다음 3개의 섹션이 포함되어 있습니다.

1. 헤더
2. Payload
3. Signature

```
11111111111.2222222222.33333333333
```

JWT는 Base64url 문자열로 암호화되며 점(".") 문자로 구분됩니다. JWT가 이 구조를 확인하지 않은 경우에는 이를 유효하지 않은 것으로 간주하고 수락하지 않도록 합니다.

2단계: JWT 서명 검증

JWT 서명은 헤더와 페이로드의 해시 조합입니다. Amazon Cognito는 각 사용자 풀에서 RSA 암호화 키를 두 쌍 생성합니다. 프라이빗 키 중 하나는 토큰 서명에 사용됩니다.

JWT 토큰의 서명을 확인하는 방법

1. ID 토큰을 디코딩합니다.
 - a. AWS Lambda를 사용하여 사용자 풀 JWT를 디코딩할 수 있습니다. 자세한 내용은 [Lambda를 사용하여 Amazon Cognito JWT 토큰을 디코딩 및 확인](#)을 참조하십시오.
 - b. 또한 OpenID Foundation에는 [JWT 토큰 작업을 위한 라이브러리 목록](#)이 포함되어 있습니다.
2. 로컬 키 ID(kid)를 퍼블릭 키와 비교합니다.
 - a. 사용자 풀에 해당되는 퍼블릭 JSON 웹 키(JWK)를 다운로드하고 저장합니다. 이 키는 JSON 웹 키 집합(JWKS)의 일부로 사용이 가능합니다. <https://cognito-idp>.

{region}.amazonaws.com/{userPoolId}/.well-known/jwks.json에서 찾을 수 있습니다.

JWK 및 JWK 집합에 대한 자세한 내용은 [JSON 웹 키\(JWK\)](#)를 참조하십시오.

Note

웹 API에서 토큰을 처리할 수 있으려면 이 단계를 한 번 거쳐야 합니다. 이제 웹 API에서 ID 토큰과 액세스 토큰을 사용할 때마다 다음 단계를 수행할 수 있습니다.

jwks.json 파일 샘플입니다.

```
{
  "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 정보는 [RSA 암호화](#)를 참조하십시오.

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. 퍼블릭 키를 사용하여 JWT 라이브러리를 이용한 서명을 확인합니다. 먼저 형식을 JWK에서 PEM로 변환해야 합니다. 이 예제에서는 JWT 및 JWK를 가져와서 Node.js 라이브러리인 jsonwebtoken을 사용하여 JWT 서명을 확인하고 있습니다.

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) {
});
```

3단계: 클레임 확인

JWT 클레임을 확인하는 방법

1. 토큰이 만료되지 않았는지 확인합니다.
2. 대상(aud) 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>`를 선택하십시오.

4. token_use 클레임을 확인합니다.
 - 웹 API에서 액세스 토큰만 허용하는 경우 해당 값은 access여야 합니다.
 - ID 토큰만 사용하고 있는 경우 해당 값은 id여야 합니다.
 - ID와 액세스 토큰을 모두 사용하고 있는 경우에는 token_use 클레임이 id 또는 access여야 합니다.

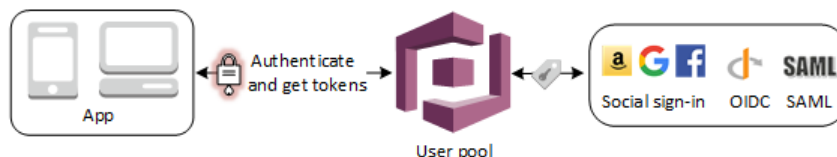
이제 토큰 내에 있는 클레임을 신뢰할 수 있습니다.

성공적인 사용자 풀 인증 이후 리소스 액세스

앱 사용자는 사용자 풀을 통해 직접 로그인하거나 타사 자격 증명 공급자(IdP)를 통해 연동 로그인할 수 있습니다. 사용자 풀에서는 Facebook, Google 및 Amazon을 통한 소셜 로그인에서 반환된 토큰과 OpenID Connect(OIDC) 및 SAML IdP에서 반환된 토큰의 처리 작업을 관리합니다. 자세한 정보는 [사용자 풀을 통해 토큰 사용](#) (p. 190) 단원을 참조하십시오.

인증 성공 이후에 앱은 Amazon Cognito로부터 사용자 풀 토큰을 받습니다. 이 토큰을 사용하여 앱이 기타 AWS 서비스에 액세스할 수 있도록 허용하는 AWS 자격 증명을 가져올 수 있습니다. 또는 이를 사용하여 서버 측 자체 리소스 또는 Amazon API Gateway에 대한 액세스를 제어할 수 있습니다.

자세한 내용은 [사용자 풀 인증 흐름](#) (p. 187) 및 [사용자 풀을 통해 토큰 사용](#) (p. 190) 단원을 참조하십시오.



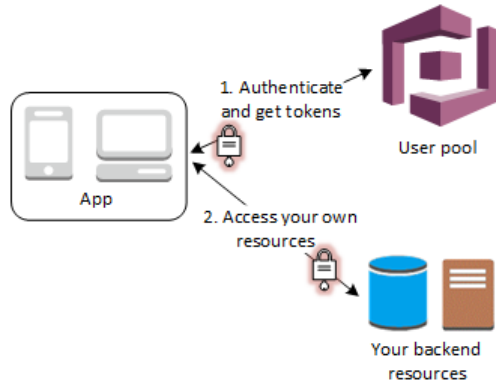
주제

- [로그인 후 서버 측 리소스 액세스](#) (p. 6)
- [로그인 후 API 게이트웨이 및 Lambda를 통해 리소스 액세스](#) (p. 198)

- 로그인 후 자격 증명 풀을 사용하여 AWS 서비스 액세스 (p. 199)

로그인 후 서버 측 리소스 액세스

성공적인 인증 이후 웹 또는 모바일 앱은 Amazon Cognito로부터 사용자 토큰을 받습니다. 이 토큰을 사용하여 서버 측 리소스에 대한 액세스를 제어할 수 있습니다. 사용자 토큰 그룹을 생성하여 권한을 관리하고 다른 유형의 사용자를 표시할 수도 있습니다. 그룹을 사용하여 리소스 액세스를 제어하는 방법은 [사용자 토큰 그룹 추가 \(p. 166\)](#) 단원을 참조하십시오.

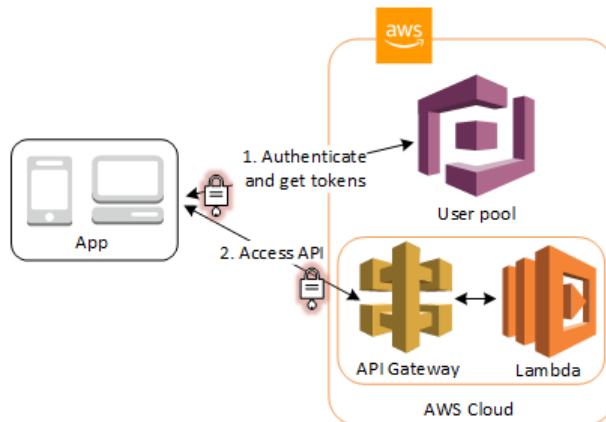


사용자 토큰에 대한 도메인이 구성되면 Amazon Cognito는 호스팅된 웹 UI를 프로비저닝하기 때문에 앱에 가입 및 로그인 페이지를 추가할 수 있습니다. 이러한 OAuth 2.0 파운데이션을 사용하여 사용자가 보호가 되는 리소스에 액세스하도록 자체 리소스 서버를 생성할 수 있습니다. 자세한 내용은 [사용자 토큰의 리소스 서버 정의 \(p. 86\)](#) 단원을 참조하십시오.

사용자 토큰 인증에 대한 자세한 내용은 [사용자 토큰 인증 흐름 \(p. 187\)](#) 및 [사용자 토큰을 통해 토큰 사용 \(p. 190\)](#) 단원을 참조하십시오.

로그인 후 API 게이트웨이 및 Lambda를 통해 리소스 액세스

사용자가 API 게이트웨이를 통해 API에 액세스하도록 할 수 있습니다. API 게이트웨이는 성공적인 사용자 토큰 인증 이후 토큰을 확인하고, 이를 사용하여 Lambda 함수를 포함한 리소스 또는 고유 API에 대한 사용자 액세스 권한을 부여합니다.



사용자 토큰에 있는 그룹을 사용하여 IAM 역할로 그룹 멤버십을 매핑함으로써 API 게이트웨이를 통한 권한을 제어할 수 있습니다. 사용자가 속한 그룹은 웹 또는 모바일 앱 사용자가 로그인할 때 사용자 토큰에서 제공한

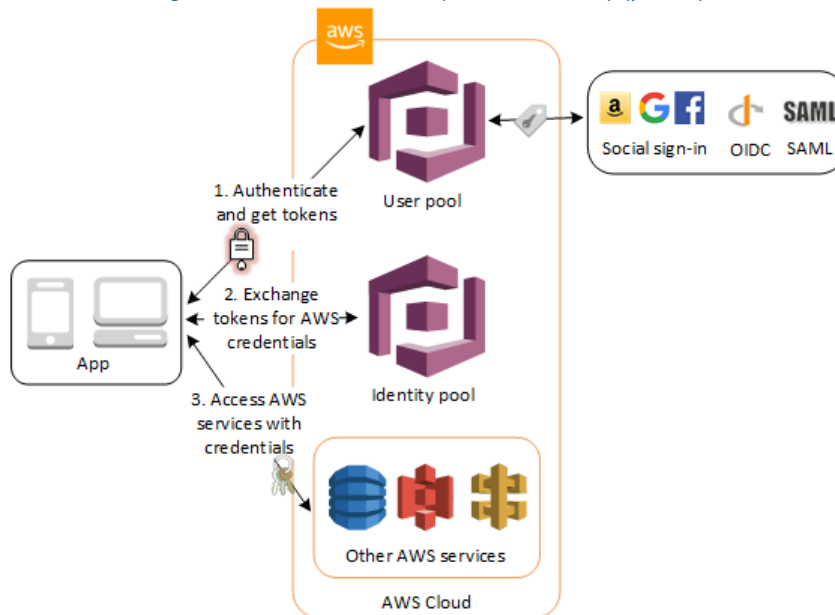
ID 토큰에 포함됩니다. 사용자 풀 그룹에 대한 자세한 내용은 [사용자 풀에 그룹 추가 \(p. 166\)](#) 단원을 참조하십시오.

Amazon Cognito 권한 부여자 Lambda 함수에 의한 확인에 대한 API 게이트웨이로의 요청을 포함하여 사용자 풀 토큰을 제출할 수 있습니다. API 게이트웨이에 대한 자세한 내용은 [Amazon Cognito 사용자 풀에서 API 게이트웨이 사용](#)을 참조하십시오.

로그인 후 자격 증명 풀을 사용하여 AWS 서비스 액세스

사용자 풀을 사용하여 로그인을 한 다음, 자격 증명을 사용하여 AWS 서비스에 액세스하도록 허용할 수 있습니다.

성공적인 인증 이후 웹 또는 모바일 앱은 Amazon Cognito로부터 사용자 풀 토큰을 받습니다. 이 토큰을 사용하여 앱이 기타 AWS 서비스에 액세스할 수 있도록 하는 AWS 자격 증명을 가져올 수 있습니다. 자세한 내용은 [Amazon Cognito 자격 증명 시작하기\(연동 자격 증명\) \(p. 223\)](#) 단원을 참조하십시오.



자격 증명 풀을 AWS 리소스 액세스를 제어하기 위한 사용자 풀 그룹과 함께 사용하는 방법에 대한 자세한 내용은 [사용자 풀에 그룹 추가 \(p. 166\)](#) 및 [역할 기반 액세스 제어 \(p. 238\)](#) 단원을 참조하십시오. 자격 증명 풀 및 [자격 증명 풀 개념\(연동 자격 증명\) \(p. 229\)](#)에 대한 자세한 내용은 AWS Identity and Access Management 단원을 참조하십시오.

AWS Management 콘솔을 통해 사용자 풀 설정

Amazon Cognito 사용자 풀을 생성하고 각 클라이언트 앱의 사용자 풀 ID 및 앱 클라이언트 ID를 기록합니다. 사용자 풀 생성에 대한 자세한 내용은 [사용자 풀 시작하기 \(p. 13\)](#)을 참조하십시오.

AWS Management 콘솔을 통해 자격 증명 풀 설정

다음 절차는 AWS Management 콘솔을 사용하여 자격 증명 풀을 하나 이상의 사용자 풀과 클라이언트 앱에 통합하는 방법을 설명합니다.

자격 증명 풀을 구성하려면

1. [Amazon Cognito 콘솔](#)을 엽니다.

2. Manage Identity Pools(자격 증명 풀 관리)를 선택합니다.
3. Amazon Cognito 사용자 풀을 공급자로 활성화할 자격 증명 풀의 이름을 선택합니다.
4. 대시보드 페이지에서[자격 증명 풀 편집]을 선택합니다.
5. [Authentication providers] 섹션을 확장합니다.
6. Cognito를 선택합니다.
7. 사용자 풀 ID를 입력합니다.
8. 앱 클라이언트 ID를 입력합니다. 이 ID는 Amazon Cognito에 대한 AWS Management 콘솔의 사용자 풀 섹션에서 앱을 생성할 때 받은 클라이언트 앱 ID와 동일해야 합니다.
9. 추가 앱이나 사용자 풀이 있는 경우 다른 공급자 추가를 선택하고 각 사용자 풀에 있는 각 앱에 대해 사용자 풀 ID 및 앱 클라이언트 ID를 입력합니다.
10. 추가할 앱이나 사용자 풀이 없는 경우 변경 사항 저장을 선택합니다.

성공하면 대시보드 페이지에 Changes saved successfully.(변경 사항이 저장되었습니다.)가 표시됩니다.

자격 증명 풀에 사용자 풀 통합

앱 사용자를 인증한 후 해당 사용자의 자격 증명 토큰을 자격 증명 공급자의 로그인 맵에 추가합니다. 공급자 이름은 Amazon Cognito 사용자 풀 ID에 따라 결정됩니다. 이름의 구조는 다음과 같습니다.

```
cognito-idp.<region>.amazonaws.com/<YOUR_USER_POOL_ID>
```

<region>의 값은 사용자 풀 ID의 리전과 동일합니다. 예: cognito-idp.us-east-1.amazonaws.com/us-east-1_123456789.

JavaScript

```
var cognitoUser = userPool.getCurrentUser();

if (cognitoUser != null) {
  cognitoUser.getSession(function(err, result) {
    if (result) {
      console.log('You are now logged in.');
```

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(serviceConfiguration,  
userPoolConfiguration: userPoolConfiguration, forKey: "UserPool")  
let pool = AWSCognitoIdentityUserPool(forKey: "UserPool")  
let credentialsProvider = AWSCognitoCredentialsProvider(regionType: .USEast1,  
identityPoolId: "YOUR_IDENTITY_POOL_ID", identityProviderManager:pool)
```

사용자 풀 참조(AWS Management 콘솔)

앱의 요구 사항에 따라 사용자 풀 설정을 사용자 지정할 수 있습니다. 이 단원에서는 설정의 각 범주를 설명하며 속성, 정책, 이메일 및 전화 확인, 멀티 팩터 인증, 앱, 트리거 및 신뢰할 수 있는 디바이스에 대해 자세히 설명합니다.

주제

- 사용자 풀 이름 추가 (p. 202)
- 사용자 및 그룹 가져오기 및 생성 (p. 202)
- 사용자 풀 속성 구성 (p. 202)
- 사용자 풀 암호 요건 추가 (p. 208)
- 사용자 생성 관리 정책 구성 (p. 208)
- 이메일 또는 전화 확인 구성 (p. 209)
- SMS 및 이메일 확인 메시지와 사용자 초대 메시지 구성 (p. 209)
- 사용자 풀에 비용 할당 태그 추가 (p. 213)
- 사용자 풀 디바이스 추적 설정 지정 (p. 213)
- 사용자 풀 앱 클라이언트 구성 (p. 214)
- 사용자 풀 Lambda 트리거 구성 (p. 215)
- 사용자 풀 생성 설정 검토 (p. 215)

- 사용자 풀 속성 분석 (p. 215)
- 앱 클라이언트 설정 구성 (p. 216)
- 사용자 풀의 도메인 이름 추가 (p. 217)
- 기본 제공 앱 UI를 사용자 지정하여 사용자 가입 및 로그인 (p. 217)
- 사용자 풀의 리소스 서버 추가 (p. 218)
- 사용자 풀에 자격 증명 공급자 구성 (p. 218)
- 사용자 풀의 속성 매핑 구성 (p. 221)

사용자 풀 이름 추가

AWS Management 콘솔에서 Amazon Cognito 사용자 풀의 풀 이름을 지정해야 합니다. 사용자 풀을 생성한 후에는 이름을 변경할 수 없습니다.

풀 이름은 1~128자 이내로 작성해야 합니다. 이 이름에는 대/소문자(a-z, A-Z), 숫자(0-9) 및 특수 문자(+ = , . @ 및 -).

사용자 및 그룹 가져오기 및 생성

사용자 및 그룹 탭에서 사용자를 가져오고, 사용자를 생성하고, 그룹을 생성하고, 사용자를 그룹에 할당하고, 사용자를 검색할 수 있습니다.

자세한 내용은 다음을 참조하십시오.

- CSV 파일에서 사용자 풀로 사용자 가져오기 (p. 172)
- 사용자 풀에 그룹 추가 (p. 166)
- 관리자로서 사용자 계정 생성 (p. 163)

사용자 풀 속성 구성

모든 사용자 풀에는 "표준 속성"이라는 여러 기본 속성이 주어집니다. AWS Management 콘솔에서 사용자 풀 정의에 사용자 지정 속성을 추가할 수도 있습니다. 이 단원에서는 이러한 속성을 자세히 설명하며 사용자 풀을 설정하는 방법에 대한 팁을 제공합니다.

속성은 이름, 이메일, 전화 번호 등 개별 사용자를 식별하는 데 도움이 되는 정보입니다.

사용자에 대한 일부 정보는 속성에 저장되지 않습니다. 예를 들어, 사용 통계나 게임 점수와 같이 자주 변경되는 사용자 데이터는 Amazon Cognito Sync 또는 Amazon DynamoDB와 같은 별도의 데이터 스토어에 저장해야 합니다.

표준 속성

다음은 사용자 풀에 있는 모든 사용자의 표준 속성입니다. 이러한 속성은 [OpenID Connect 사양](#)에 따라 구현됩니다.

- address
- birthdate
- email
- family_name
- gender

- given_name
- locale
- middle_name
- name
- nickname
- phone_number
- picture
- preferred_username
- profile
- timezone
- updated_at
- website

이러한 속성은 모든 사용자의 선택적 속성으로 사용할 수 있습니다. 속성을 필수로 설정하려면 속성 옆에 있는 확인란을 선택합니다.

Note

표준 속성이 필수로 표시되면 해당 속성의 값을 제공해야 사용자가 등록할 수 있습니다. 관리자는 필수 속성의 값을 제공하지 않고 [AdminCreateUser](#) API를 사용하여 사용자를 생성할 수 있습니다. 사용자 풀이 생성된 후에는 필수 속성을 필수가 아닌 속성으로, 필수가 아닌 속성을 필수 속성으로 전환할 수 없습니다.

표준 및 사용자 지정 속성 값은 기본적으로 최대 2,048자의 문자열일 수 있지만 `updated_at`과 같은 일부 속성 값에는 형식 제한이 있습니다. email 및 phone만 확인할 수 있습니다.

Note

사양에서는 속성을 멤버라고 합니다.

다음은 위의 일부 필드와 관련된 몇 가지 추가 정보입니다.

email

이메일 주소 값을 확인할 수 있습니다.

적절한 AWS 계정 권한을 가진 관리자가 사용자의 이메일을 변경하고 확인됨으로 표시할 수 있습니다. [AdminUpdateUserAttributes](#) API 또는 [admin-update-user-attributes](#) CLI 명령을 통해 `email_verified` 속성을 `true`로 변경하여 이를 수행할 수 있습니다.

phone

SMS MFA(멀티 팩터 인증)가 활성화되면 전화 번호가 필수 사항이 됩니다. 자세한 내용은 [사용자 풀에 멀티 팩터 인증\(MFA\) 추가 \(p. 107\)](#) 단원을 참조하십시오.

전화 번호 값을 확인할 수 있습니다.

적절한 AWS 계정 권한을 가진 관리자가 사용자의 전화 번호를 변경하고 확인됨으로 표시할 수 있습니다. [AdminUpdateUserAttributes](#) API 또는 [admin-update-user-attributes](#) CLI 명령을 통해 `phone_number_verified` 속성을 `true`로 변경하여 이를 수행할 수 있습니다.

Important

형식 규칙에 따라 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 문자를 제거해야 합니다. 예를 들어, 미국 기반 전화 번호는 **+14325551212** 형식을 따라야 합니다.

preferred_username

preferred_username을 필수 항목이자 별칭으로 선택할 수 없습니다. preferred_username이 별칭이면 사용자가 [UpdateUserAttributes](#) API를 사용하여 확인된 후 속성 값을 추가할 수 있습니다.

표준 속성을 편집하려면

1. 속성 탭에서 사용자 등록에 필요한 속성을 선택합니다. 속성이 필수이고 사용자가 필수 속성을 제공하지 않으면 등록할 수 없습니다.

Important

사용자 풀이 생성된 후에는 이 요구 사항을 변경할 수 없습니다. 자세한 내용은 [사용자 풀 속성 구성 \(p. 202\)](#) 단원을 참조하십시오.

2. 이메일, 전화 번호, 주소 또는 선호하는 사용자 이름에 대한 별칭을 생성하려면 Alias(별칭)를 선택합니다. 별칭에 대한 자세한 내용은 [별칭 개요 \(p. 204\)](#)를 참조하십시오.
3. 이동하여 [사용자 지정 속성 \(p. 207\)](#)을 생성합니다.

사용자 이름 및 선호하는 사용자 이름

username 값은 별도의 속성이며 name 속성과 동일하지 않습니다. username은 사용자를 등록하는 데 항상 필요하며, 사용자가 생성된 후에는 변경할 수 없습니다.

개발자는 preferred_username 속성을 사용하여 변경할 수 있는 사용자 이름을 사용자에게 제공할 수 있습니다. 자세한 내용은 [별칭 개요 \(p. 204\)](#) 단원을 참조하십시오.

애플리케이션에서 사용자 이름이 필요하지 않은 경우 사용자 이름을 제공하도록 사용자에게 요청할 필요가 없습니다. 앱이 백그라운드에서 고유한 사용자 이름을 생성할 수 있습니다. 예를 들어, 사용자가 이메일 주소와 암호를 사용하여 등록 및 로그인하게 하려는 경우 유용한 기능입니다. 자세한 내용은 [별칭 개요 \(p. 204\)](#) 단원을 참조하십시오.

username은 사용자 풀에서 고유해야 합니다. username은 삭제되고 더 이상 사용되지 않는 경우에만 재사용할 수 있습니다.

별칭 개요

최종 사용자가 별칭을 사용하여 여러 ID로 로그인하도록 허용할 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 선호하는 사용자 이름 속성은 별칭으로 표시할 수 있습니다.

예를 들어, 사용자 풀에 대해 이메일 및 전화를 별칭으로 선택하면 해당 사용자 풀의 사용자는 사용자 이름, 이메일 주소 또는 전화 번호를 암호와 함께 사용하여 로그인할 수 있습니다.

이메일을 별칭으로 선택하면 사용자 이름이 유효한 이메일 형식에 맞지 않습니다. 마찬가지로 전화 번호를 별칭으로 선택하면 해당 사용자 풀의 서비스가 유효한 전화 번호 패턴에 맞는 사용자 이름을 수락하지 않습니다.

Note

별칭 값은 사용자 풀에서 고유해야 합니다. 이메일 주소나 전화 번호의 별칭이 구성되면 제공된 값이 계정 하나에서만 확인됨 상태가 될 수 있습니다. 가입 중 이미 사용된 다른 계정에서 이메일 주소나 전화 번호가 별칭으로 제공되면 등록이 성공합니다. 그러나 사용자가 이 이메일(또는 전화 번호)로 계정 확인을 시도하고 유효한 코드를 입력하면 `AliasExistsException` 오류가 발생합니다. 이 오류는 이 이메일(또는 전화 번호)이 있는 계정이 이미 있음을 사용자에게 나타냅니다. 이때 사용자는 새 계정 생성을 중단하고 이전 계정의 암호를 재설정해볼 수 있습니다. 사용자가 계속해서 새 계정을 생성하면 앱이 `ConfirmSignUp` 옵션으로 `forceAliasCreation` API를 호출해야 합니다. 그러면 이전 계정에서 새로 생성된 계정으로 별칭이 이동하며, 이전 계정에서 해당 속성이 확인되지 않음으로 표시됩니다.

전화 번호와 이메일 주소가 확인된 경우에만 전화 번호와 이메일 주소가 사용자의 활성 별칭이 됩니다. 따라서 이메일 주소와 전화 번호를 별칭으로 사용할 경우 이메일 주소와 전화 번호의 자동 확인을 선택하는 것이 좋습니다. 사용자의 실제 사용자 이름 값을 변경할 수 없는 경우 사용자가 자신의 사용자 이름을 변경할 수 있도록 `preferred_username` 속성이 포함됩니다.

사용자가 사용자 이름을 변경할 수 있도록 허용하려면 새 `username` 값을 `preferred_username`으로 제출하고 별칭으로 `preferred_username`을 선택합니다. 그러면 입력한 새 값으로 로그인할 수 있습니다.

`preferred_username`을 별칭으로 선택하면 계정이 확인되어야 값을 입력할 수 있습니다. 등록 중에는 값을 입력할 수 없습니다.

별칭을 사용하여 사용자 가입 및 로그인 간소화

속성 콘솔 탭에서 사용자가 이메일 주소나 전화 번호를 사용자 이름으로 사용하여 로그인할 수 있도록 허용할지 여부를 선택할 수 있습니다.

Note

사용자 풀이 생성되고 나면 이 설정을 변경할 수 없습니다.

주제

- 옵션 1: 사용자가 사용자 이름을 사용하여 가입하고 사용자 이름이나 별칭을 사용하여 로그인 (p. 205)
- 옵션 2: 사용자가 사용자 이름 대신 이메일 또는 전화 번호를 사용하여 가입 및 로그인 (p. 206)

옵션 1: 사용자가 사용자 이름을 사용하여 가입하고 사용자 이름이나 별칭을 사용하여 로그인

이 경우, 사용자가 사용자 이름으로 가입합니다. 또한 선택에 따라 사용자가 다음 별칭 중 하나 이상을 사용하여 로그인할 수 있도록 허용할 수 있습니다.

- 확인된 이메일 주소
- 확인된 전화 번호
- 기본 설정 사용자 이름

사용자가 가입한 후에 이들 별칭을 변경할 수 있습니다.

다음 단계에 따라 별칭을 사용하여 로그인할 수 있도록 콘솔에서 사용자 풀을 구성합니다.

별칭을 사용한 로그인이 가능하도록 사용자 풀을 구성하려면

1. 속성 탭의 최종 사용자 로그인 방법을 어떻게 설정하시겠습니까?에서 사용자 이름을 선택합니다.
2. 다음 옵션 중 하나를 선택합니다.

- 확인된 이메일 주소로 로그인 허용: 사용자가 이메일 주소를 사용하여 로그인할 수 있습니다.
- 확인된 전화 번호로 로그인 허용: 사용자가 전화 번호를 사용하여 로그인할 수 있습니다.
- 기본 설정 사용자 이름으로 로그인 허용: 사용자가 기본 설정 사용자 이름을 사용하여 로그인할 수 있습니다. 기본 설정 사용자 이름은 사용자가 변경할 수 있는 사용자 이름입니다.

옵션 2: 사용자가 사용자 이름 대신 이메일 또는 전화 번호를 사용하여 가입 및 로그인

이 경우, 사용자는 이메일 주소나 전화 번호를 사용자 이름으로 사용하여 가입할 수 있습니다. 사용자가 이메일 주소만, 전화 번호만, 또는 둘 중 하나를 사용하여 가입할 수 있도록 허용할지 여부를 선택할 수 있습니다.

이메일이나 전화 번호는 고유해야 하며 다른 사용자가 이미 사용하고 있어서는 안 됩니다. 확인을 받을 필요는 없습니다. 이메일이나 전화 번호를 사용하여 가입을 하고 나면 사용자는 같은 이메일이나 전화 번호로 새 계정을 생성할 수 없습니다. 기존 계정은 재사용만 가능합니다(필요할 경우 암호 재설정). 그러나 이메일이나 전화 번호를 새로 변경할 수는 있습니다. 이미 사용 중인 것이 아니라면 변경된 이메일이나 전화 번호가 새 사용자 이름이 됩니다.

Note

사용자가 이메일 주소를 사용자 이름으로 사용하여 가입한 경우에는 사용자 이름을 다른 이메일 주소로 변경할 수 있으나 전화 번호로 변경할 수는 없습니다. 사용자가 전화 번호를 사용하여 가입한 경우에는 사용자 이름을 다른 전화 번호로 변경할 수 있으나 이메일 주소로 변경할 수는 없습니다.

다음 단계에 따라 이메일이나 전화 번호를 사용하여 가입 및 로그인을 할 수 있도록 콘솔에서 사용자 풀을 구성합니다.

이메일이나 전화 번호를 사용하여 가입 및 로그인을 할 수 있도록 사용자 풀을 구성하려면

1. 속성 탭의 최종 사용자 로그인 방법을 어떻게 설정하시겠습니까?에서 이메일 주소 또는 전화 번호를 선택합니다.
2. 다음 옵션 중 하나를 선택합니다.
 - 이메일 주소 허용: 사용자가 이메일을 사용자 이름으로 사용하여 로그인할 수 있습니다.
 - 전화 번호 허용: 사용자가 전화 번호를 사용자 이름으로 사용하여 로그인할 수 있습니다.
 - 이메일 주소 및 전화 번호 허용(사용자가 하나 선택 가능): 사용자가 가입 시 이메일 주소나 전화 번호를 사용자 이름으로 사용할 수 있습니다.

Note

이메일이나 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 형식이면 사용자 풀에서 사용자의 `email` 속성이 `username` 값으로 자동으로 채워집니다.
 - `username` 문자열이 유효한 전화 번호 형식이면 사용자 풀에서 사용자의 `phone_number` 속성이 `username` 값으로 자동으로 채워집니다.
 - `username` 문자열 형식이 이메일이나 전화 번호가 아니면 `SignUp` API에서 예외가 발생합니다.

- `SignUp` API는 사용자에게 대해 영구적인 UUID를 생성하고 내부적으로 이를 변경이 불가능한 사용자 이름 속성으로 사용합니다. 이 UUID는 사용자 ID 토큰의 `sub` 클레임과 동일한 값을 갖습니다.
- `username` 문자열에 이미 사용 중인 이메일 주소나 전화 번호가 포함되어 있으면 `SignUp` API에서 예외가 발생합니다.

`ListUsers` API를 제외하고 모든 API에서 사용자 이름 자리에 별칭으로 이메일 주소나 전화 번호를 사용할 수 있습니다. `ListUsers`를 호출하면 `email` 또는 `phone_number` 속성으로 검색을 할 수 있습니다. `username`으로 검색을 하는 경우에는 별칭이 아니라 실제 사용자 이름을 입력해야 합니다.

사용자 지정 속성

사용자 풀에 사용자 지정 속성 25개까지 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 속성:

- 문자열 또는 숫자로 정의할 수 있습니다.
- 필수 항목이 될 수 없습니다.
- 사용자 풀에 추가된 후에는 삭제 또는 변경할 수 없습니다.
- Amazon Cognito 허용 한도 내에 있는 문자 길이의 이름을 가질 수 있습니다. 자세한 내용은 [Amazon Cognito의 제한 값 \(p. 311\)](#) 단원을 참조하십시오.

Note

[역할 기반 액세스 제어 \(p. 238\)](#)에 대한 규칙 설정 및 코드에서는 표준 속성과 구분하기 위해 사용자 지정 속성에 `custom:` 접두사가 필요합니다.

콘솔을 사용하여 사용자 지정 속성을 추가하려면

1. 왼쪽의 탐색 모음에서 속성을 선택합니다.
2. 새로운 각 속성에 대해
 - a. 사용자 지정 속성을 추가하시겠습니까? 에서 다른 속성 추가를 선택합니다.
 - b. 데이터 유형(문자열 또는 숫자), 이름, 최소 길이, 최대 길이 등 각 사용자 지정 속성의 속성을 선택합니다.
 - c. 사용자가 사용자 지정 속성의 값을 제공한 후 해당 값을 변경할 수 있도록 하려면 변경 가능을 선택합니다.

속성 권한 및 범위

각 사용자 속성의 앱당 읽기 및 쓰기 권한을 설정할 수 있습니다. 권한을 설정하면 사용자에게 대해 저장된 각 속성을 확인 및/또는 수정할 수 있는 애플리케이션을 제어할 수 있습니다. 예를 들어, 사용자가 유료 고객인지 여부를 나타내는 사용자 지정 속성이 있을 수 있습니다. 앱에서는 이 속성을 볼 수 있지만 직접 수정할 수 없습니다. 대신 관리 도구 또는 백그라운드 프로세스를 사용하여 이 속성을 업데이트합니다. 사용자 속성에 대한 권한은 Amazon Cognito 콘솔, API 또는 CLI에서 설정할 수 있습니다. 기본적으로 새로운 사용자 지정 속성은 사용자가 그에 대한 읽기 및 쓰기 권한을 설정한 이후에만 사용할 수 있습니다.

콘솔을 사용하여 속성 권한을 설정 또는 변경하려면

1. 왼쪽의 탐색 모음에서 앱 클라이언트를 선택합니다.
2. 업데이트할 앱 클라이언트의 세부 정보 표시를 선택합니다.
3. Set the attribute read and write permissions for each attribute(각 속성에 대해 속성 읽기 및 쓰기 권한 설정)를 선택합니다.

4. 앱 클라이언트 변경 사항 저장을 선택합니다.

사용자 지정 속성을 사용하여 각 앱마다 이러한 단계를 반복합니다.

각 앱에 대해 속성을 읽기 가능 또는 쓰기 가능으로 표시할 수 있습니다. 표준 속성과 사용자 지정 속성에 모두 해당하는 사항입니다. 앱에서 읽기 가능으로 표시된 속성을 읽을 수 있으며, 쓰기 가능으로 표시된 속성을 쓸 수 있습니다. 앱에서 쓰기가 가능하지 않은 속성을 업데이트하려고 시도할 경우 `NotAuthorizedException` 예외가 발생합니다. `GetUser`를 호출하는 앱은 해당 앱에 대해 읽기 가능한 속성만 받습니다. 사후 인증으로 발급된 ID 토큰에는 읽기 가능 속성에 해당하는 클레임만 포함됩니다. 사용자 풀에 대한 필수 속성은 항상 쓸 수 있습니다. CLI 또는 관리자 API를 사용하여 쓰기 가능 속성을 설정하고 필수 속성을 제공하지 않는 경우 `InvalidParameterException` 예외가 발생합니다.

사용자 풀을 생성한 후 속성 권한과 범위를 변경할 수 있습니다.

사용자 풀 암호 요건 추가

대문자, 숫자 및 특수 문자 요구를 비롯하여 8자 이상의 최소 암호 길이를 지정하면 앱 사용자의 강력한 암호가 생성됩니다. 복잡한 암호는 알아맞추기 어려우므로 보안의 모범 사례로 복잡한 암호를 사용하는 것이 좋습니다.

다음 문자는 암호에 사용할 수 있습니다.

- 대문자와 소문자
- 숫자
- 등호 기호 "="
- 더하기 기호 "+"
- 다음 섹션에 나열된 특수 문자

등호 기호 "=" 및 더하기 기호 "+"는 특수 문자로 취급되지 않습니다.

암호 정책 생성

AWS Management 콘솔 콘솔에서 다음과 같은 암호 요구 사항을 지정할 수 있습니다.

- 최소 길이 - 6자 이상, 99자 미만이어야 합니다.
- 숫자 요구
- 이 세트에서 특수 문자 필요:

`^ $ * . [] { } () ? - " ! @ # % & / \ , > < ' : ; | _ ~ ``

- 대문자 요구
- 소문자 요구

사용자 생성 관리 정책 구성

사용자 생성 관리에 다음 정책을 지정할 수 있습니다.

- 사용자가 스스로 가입할 수 있도록 허용하는지 여부를 지정합니다. 이 옵션은 기본적으로 설정되어 있습니다. 설정이 되어 있지 않으면 이 풀에서 관리자만 사용자를 생성할 수 있고 `NotAuthorizedException` 오류 코드와 함께 `SignUp` API에 대한 호출이 실패합니다.
- 새 계정에 사용자 계정 만료 시간 제한(일)을 지정합니다. 기본 설정은 7일이며, 사용자 계정이 생성된 날로부터 측정됩니다. 최대 설정은 90일입니다. 계정이 만료되면 관리자가 속성을 업데이트하거나 사용자에게 암호를 재전송하여 사용자의 프로파일을 업데이트할 때까지 사용자가 해당 계정에 로그인할 수 없습니다.

Note

사용자가 로그인하면 계정은 만료되지 않습니다.

이메일 또는 전화 확인 구성

MFA and verifications(MFA 및 확인) 탭에서 이메일 또는 전화 확인 설정을 선택할 수 있습니다.

Amazon Cognito는 확인 코드를 보내거나 이메일의 경우 확인 링크를 보내서 이메일 주소 또는 휴대 전화 번호를 자동으로 확인할 수 있습니다. 이메일 주소의 경우 코드나 링크가 이메일 메시지로 전송됩니다. 전화 번호의 경우 코드가 SMS 문자 메시지로 전송됩니다.

사용자를 자동으로 확인하고 분실한 암호 복구를 활성화하려면 전화나 이메일 확인이 필요합니다. 또는 사전 가입 Lambda 트리거 또는 [AdminConfirmSignUp](#) API를 사용하여 사용자를 자동으로 확인할 수 있습니다. 자세한 내용은 [사용자 계정 가입 및 확인 \(p. 156\)](#) 단원을 참조하십시오.

확인 코드나 링크는 24시간 동안 유효합니다.

이메일이나 전화 확인을 필수 항목으로 선택하면 사용자가 가입할 때 확인 코드나 링크가 자동으로 전송됩니다.

참고

- 전화 번호를 확인하기 위한 Amazon SNS 문자 메시지 사용은 Amazon SNS에서 요금이 별도로 청구됩니다. 이메일 주소로 확인 코드를 전송하는 것은 무료입니다. Amazon SNS 요금에 대한 자세한 내용은 [전 세계 SMS 요금](#)을 참조하십시오. 현재 SMS 메시징이 가능한 국가의 목록은 [지원되는 리전 및 국가](#)를 참조하십시오.
- Amazon Cognito에서 이메일을 시작하는 작업을 앱에서 테스트할 때는 하드 바운스가 발생하지 않고 Amazon Cognito에서 보낼 수 있는 이메일 주소를 사용하십시오. 자세한 내용은 [the section called “앱 테스트 중 이메일 보내기” \(p. 162\)](#) 단원을 참조하십시오.
- 암호를 잊어버린 경우 사용자의 이메일이나 전화 번호를 확인하도록 요구합니다.

Important

사용자가 전화 번호 및 이메일 주소 둘 다를 사용하여 가입했으며 사용자 풀 설정에서 두 속성의 확인이 필요한 경우 확인 코드가 SMS를 통해 전화로 전송됩니다. 이메일 주소는 확인이 되지 않기 때문에 앱에서 [GetUser](#)를 호출하여 이메일 주소가 확인을 기다리는지 여부를 확인할 수 있습니다. 기다릴 경우 앱은 [GetUserAttributeVerificationCode](#)를 호출하여 이메일 확인 흐름을 시작한 다음 [VerifyUserAttribute](#)를 호출하여 확인 코드를 제출해야 합니다.

AWS 계정 및 개별 메시지에 지출 한도를 지정할 수 있으며, 이 한도는 SMS 메시지 전송 비용에만 적용됩니다. 자세한 내용은 <https://aws.amazon.com/sns/faqs/>를 참조하십시오.

자동으로 SMS 메시지를 전송하도록 Amazon Cognito에 권한 부여

Amazon Cognito에서 자동으로 사용자에게 SMS 메시지를 보내려면 권한이 필요합니다. 해당 권한을 부여하려면 Amazon Cognito 콘솔의 MFA and verifications(MFA 및 확인) 탭에서 역할 생성을 선택하여 AWS Identity and Access Management(IAM) 역할을 만들면 됩니다.

SMS 및 이메일 확인 메시지와 사용자 초대 메시지 구성

메시지 사용자 지정 탭에서 다음을 사용자 지정할 수 있습니다.

- SMS 텍스트 메시지 MFA 메시지
- SMS 및 이메일 확인 메시지
- 이메일 확인 유형(코드 또는 링크)
- 사용자 초대 메시지
- 사용자 풀을 통해 전송되는 이메일의 발신 및 회신 이메일 주소

Note

확인 탭에서 전화 번호 및 이메일 확인을 요구하도록 선택해야 SMS 및 이메일 확인 메시지 템플릿이 표시됩니다. 마찬가지로 MFA 설정이 REQUIRED 또는 OPTIONAL이어야 SMS MFA 메시지 템플릿이 표시됩니다.

주제

- [메시지 템플릿 \(p. 210\)](#)
- [SMS 메시지 사용자 지정 \(p. 211\)](#)
- [이메일 확인 메시지 사용자 지정 \(p. 211\)](#)
- [사용자 초대 메시지 사용자 지정 \(p. 211\)](#)
- [이메일 주소 사용자 지정 \(p. 212\)](#)
- [자동으로 Amazon SES 이메일을 전송하도록 Amazon Cognito에 권한 부여\(사용자 지정 발신 이메일 주소 사용\) \(p. 212\)](#)

메시지 템플릿

메시지 템플릿에서 해당 값으로 대체될 자리 표시자를 사용하여 메시지에 필드를 삽입할 수 있습니다.

템플릿 자리 표시자

설명	Token
확인 코드	{####}
임시 암호	{####}
사용자 이름	{사용자 이름}

고급 보안 템플릿 자리 표시자를 사용하여 다음을 수행할 수 있습니다.

- Amazon Cognito 고급 보안 기능을 통해 분석할 수 있도록 IP 주소, 도시, 로그인 시간 및 디바이스 이름 등 이벤트에 대한 특정 세부 정보를 포함시킬 수 있습니다.
- 원클릭 링크가 유효한지 확인할 수 있습니다.
- 이벤트 ID, 피드백 토큰, 사용자 이름을 사용하여 자체 원클릭 링크를 빌드할 수 있습니다.

고급 보안 템플릿 자리 표시자

설명	Token
IP 주소	{ip-address}
구/군/시	{시}
국가	{국가}

설명	Token
로그인 시간	{login-time}
디바이스 이름	{device-name}
원클릭 링크 유효	{one-click-link-valid}
원클릭 링크 잘못된	{one-click-link-invalid}
이벤트 ID	{event-id}
피드백 토큰	{feedback-token}

SMS 메시지 사용자 지정

Do you want to customize your SMS messages?(SMS 메시지를 사용자 지정하시겠습니까?) 머리글 아래의 템플릿을 편집하여 MFA 인증을 위한 SMS 메시지를 사용자 지정할 수 있습니다.

Important

사용자 지정 메시지에는 {####} 자리 표시자가 있어야 하며, 이 자리 표시자는 메시지를 전송하기 전에 인증 코드로 대체됩니다.

메시지의 최대 길이는 인증 코드를 포함하여 UTF-8 문자 140개입니다.

SMS 확인 메시지 사용자 지정

SMS 확인 메시지를 사용자 지정하시겠습니까? 머리글 아래의 템플릿을 편집하여 전화 번호 확인을 위한 SMS 메시지를 사용자 지정할 수 있습니다.

Important

사용자 지정 메시지에는 {####} 자리 표시자가 있어야 하며, 이 자리 표시자는 메시지를 전송하기 전에 확인 코드로 대체됩니다.

메시지의 최대 길이는 확인 코드를 포함하여 UTF-8 문자 140개입니다.

이메일 확인 메시지 사용자 지정

이메일 확인 유형(코드 또는 링크)을 선택할 수 있습니다.

이메일 확인 메시지를 사용자 지정하시겠습니까? 아래의 템플릿을 편집하여 이메일 주소 확인을 위한 이메일 제목 및 메시지를 사용자 지정할 수 있습니다.

Important

확인 유형으로 코드를 선택한 경우에는 사용자 지정 메시지에 {####} 자리 표시자가 포함되어야 하며, 이 자리 표시자는 메시지를 전송하기 전에 확인 코드로 대체됩니다.

메시지의 최대 길이는 확인 코드를 포함하여(있을 경우) UTF-8 문자 20,000개입니다. 이 이메일에 HTML 태그를 사용할 수 있습니다.

사용자 초대 메시지 사용자 지정

사용자 초청 메시지를 사용자 지정하시겠습니까? 머리글 아래의 템플릿을 편집하여 Amazon Cognito가 SMS 또는 이메일을 통해 새 사용자에게 전송하는 사용자 초대 메시지를 사용자 지정할 수 있습니다.

Important

사용자 지정 메시지는 {username} 및 {####} 자리 표시자가 있어야 하며, 이 자리 표시자는 메시지를 전송하기 전에 사용자의 사용자 이름과 암호로 대체됩니다.

SMS의 최대 길이는 확인 코드를 포함하여 UTF-8 문자 140개입니다. 이메일의 메시지 최대 길이는 확인 코드를 포함하여 UTF-8 문자 20,000개입니다. 이 이메일에 HTML 태그를 사용할 수 있습니다.

이메일 주소 사용자 지정

기본적으로 Amazon Cognito가 사용자 풀의 사용자에게 보내는 이메일 메시지는 no-reply@verificationemail.com에서 전송됩니다. no-reply@verificationemail.com 대신 사용자 지정 발신 이메일 주소와 회신 이메일 주소를 사용하도록 지정할 수 있습니다.

발신 이메일 주소를 사용자 지정하려면 사용자 지정 발신 주소 추가를 선택하고 지침에 따라 Amazon Simple Email Service 자격 증명을 확인합니다. AWS 리전과 Amazon SES 확인 자격 증명을 선택합니다. 자세한 내용은 Amazon Simple Email Service 개발자 안내서의 [Amazon SES에서 이메일 주소 및 도메인 확인](#)을 참조하십시오.

회신 이메일 주소를 사용자 지정하려면 사용자 지정 회신 주소 추가를 선택하고 유효한 이메일 주소를 입력합니다.

자동으로 Amazon SES 이메일을 전송하도록 Amazon Cognito에 권한 부여(사용자 지정 발신 이메일 주소 사용)

기본값 대신 사용자 지정 발신 이메일 주소에서 이메일을 전송하려면 Amazon SES 확인 자격 증명 대신 사용자에게 이메일 메시지를 전송할 수 있는 권한이 Amazon Cognito에 필요합니다. 해당 권한을 부여하려면 전송 권한 부여 정책을 생성하십시오. 자세한 내용은 Amazon Simple Email Service 개발자 안내서의 [Amazon SES에서 전송 권한 부여 사용](#)을 참조하십시오.

다음은 Amazon Cognito 사용자 풀에 대한 Amazon SES 전송 권한 부여 정책의 예제입니다. 자세한 내용은 Amazon Simple Email Service 개발자 안내서의 [Amazon SES 전송 권한 부여 정책 예제](#)를 참조하십시오.

Note

이 예제에서 "Sid" 값은 명령문을 고유하게 식별하는 임의의 문자열입니다. 정책 구문에 대한 자세한 내용은 Amazon Simple Email Service 개발자 안내서의 [Amazon SES 전송 권한 부여 정책](#)을 참조하십시오.

```
{
  "Version": "2008-10-17",
  "Statement": [
    {
      "Sid": "stmnt1234567891234",
      "Effect": "Allow",
      "Principal": {
        "Service": "cognito-idp.amazonaws.com"
      },
      "Action": [
        "ses:SendEmail",
        "ses:SendRawEmail"
      ],
      "Resource": "<your SES identity ARN>"
    }
  ]
}
```

드롭다운 메뉴에서 Amazon SES 자격 증명을 선택할 때 Amazon Cognito 콘솔이 백그라운드에서 이 정책을 추가합니다. CLI 또는 API를 사용하여 사용자 풀을 구성할 경우 이 정책을 자격 증명에 연결해야 합니다.

사용자 풀에 비용 할당 태그 추가

태그 탭에서 비용 할당 태그를 추가하여 AWS 비용을 분류하고 추적할 수 있습니다. AWS 리소스(예: Amazon Cognito 사용자 풀)에 태그를 적용할 때 태그별로 집계된 사용 내역 및 비용이 AWS 비용 할당 보고서에 포함됩니다. 비즈니스 범주를 나타내는 태그(예: 비용 센터, 애플리케이션 이름 및 소유자)를 적용하여 여러 서비스의 비용을 정리할 수 있습니다. 자세한 내용은 AWS Billing and Cost Management 사용 설명서의 [비용 할당 태그 사용](#)을 참조하십시오.

태그를 추가하려면 태그 추가를 선택합니다. **태그 제한**에 나열된 제한에 따라 태그 키 및 태그 값을 지정합니다. 변경 사항 저장을 선택하여 태그를 저장합니다.

Important

결제 보고서에 적용되는 태그를 표시하려면 결제 콘솔에서 태그를 활성화해야 합니다. 자세한 내용은 AWS Billing and Cost Management 사용 설명서의 [사용자 정의 비용 할당 태그 활성화](#)를 참조하십시오.

사용자 풀 디바이스 추적 설정 지정

보안 강화를 위한 방편으로, 사용자가 로그인한 디바이스를 추적할 수 있습니다. 이 단원에서는 AWS Management 콘솔에서 Amazon Cognito 사용자 풀에 디바이스 추적을 추가하는 방법을 설명합니다.

기억된 디바이스 설정

Amazon Cognito 사용자 풀을 사용하면 Amazon Cognito가 애플리케이션에 액세스하는 데 사용된 디바이스를 기억하고 기억된 디바이스를 사용자 풀의 애플리케이션 사용자에게 연결할 수 있습니다. MFA(멀티 팩터 인증)를 설정한 경우 기억된 디바이스를 사용하여 사용자에게 코드를 전송하는 것을 중지할 수도 있습니다.

Amazon Cognito 콘솔을 통해 기억된 디바이스 기능을 설정할 때 항상, 사용자 옵트인 및 아니오의 3가지 옵션이 제공됩니다.

- 아니오(기본값) – 디바이스가 기억되지 않습니다.
- 항상 – 애플리케이션의 사용자가 사용한 모든 디바이스가 기억됩니다.
- 사용자 옵트인 – 사용자가 디바이스를 기억하도록 선택한 경우에만 사용자의 디바이스가 기억됩니다.

항상 또는 사용자 옵트인이 선택되면 사용자가 처음으로 디바이스에 로그인할 때 각 디바이스에 디바이스 식별자(키 및 암호)가 할당됩니다. 이 키는 디바이스 식별을 위해서만 사용되지만 서비스에 의해 추적됩니다.

항상을 선택할 경우 Amazon Cognito는 사용자 인증 흐름의 일환으로 사용자가 디바이스에 로그인할 때마다 디바이스 식별자(키 및 암호)를 사용하여 디바이스를 인증합니다.

사용자 옵트인을 선택할 경우 애플리케이션의 사용자가 기억하도록 선택한 경우에만 디바이스를 기억할 수 있습니다. 사용자가 새 디바이스에 로그인할 때 추적 시작 요청의 응답은 디바이스를 기억할 것인지에 대해 묻는 메시지를 사용자에게 표시할지 여부를 나타냅니다. 사용자에게 묻는 사용자 인터페이스를 생성해야 합니다. 사용자가 디바이스를 기억하도록 선택하면 디바이스 상태가 '기억됨' 상태로 업데이트됩니다.

AWS Mobile SDK에는 기억된 디바이스를 확인하고([ListDevices](#), [GetDevice](#)), 디바이스를 기억됨 또는 기억되지 않음으로 표시하며([UpdateDeviceStatus](#)), 디바이스 추적을 중지([ForgetDevice](#))하기 위한 추가 API가 있습니다. REST API에는 승격된 권한이 있으며 모든 사용자에게 대해 작동하는 API의 추가 관리자 버전이 있습니다. 이 API에는 [AdminListDevices](#), [AdminGetDevice](#) 등의 API 이름이 있습니다. 이 API는 SDK를 통해 노출되지 않습니다.

기억된 디바이스를 사용하여 MFA(멀티 팩터 인증) 억제

항상 또는 사용자 옵트인을 선택한 경우 애플리케이션의 사용자에게 대해 기억된 디바이스에서 MFA 챌린지를 억제할 수도 있습니다. 이 기능을 사용하려면 사용자 풀에 대해 MFA를 활성화해야 합니다. 자세한 내용은 [사용자 풀에 멀티 팩터 인증\(MFA\) 추가](#) (p. 107) 단원을 참조하십시오.

Note

디바이스 기억 기능이 항상으로, 멀티 팩터 인증(MFA) 중에 기억한 디바이스를 이용해 두 번째 요소를 억제하시겠습니까?가 예로 설정되어 있으면, 위험 기반 MFA에서 위험도 중간/높음에 대한 MFA 설정이 무시됩니다.

사용자 풀 앱 클라이언트 구성

앱은 사용자 풀 내에 있으며 등록, 로그인 및 잊어버린 암호 처리를 위한 API 등 미인증 API(인증된 사용자가 없는 API)를 호출하는 권한이 있는 개체입니다. 이 API를 호출하려면 앱 클라이언트 ID와 클라이언트 암호(선택 사항)가 필요합니다. 개발자는 권한이 있는 클라이언트 앱만 이러한 미인증 API를 호출할 수 있도록 앱 클라이언트 ID 또는 암호를 보호할 책임이 있습니다.

사용자 풀에 대해 여러 앱을 생성할 수 있으며, 일반적으로 앱은 앱의 플랫폼에 해당합니다. 예를 들어, 서버 측 애플리케이션에 대한 앱과 다른 Android 앱을 생성할 수 있습니다. 각 앱에는 고유한 앱 클라이언트 ID가 있습니다.

앱을 생성할 때 필요한 경우 해당 앱에 대한 암호를 생성하도록 선택할 수 있습니다. 앱에 대해 암호가 생성되면 앱을 사용하기 위해 해당 암호를 제공해야 합니다. JavaScript로 작성된 브라우저 기반 애플리케이션에는 암호가 있는 앱이 필요하지 않을 수 있습니다.

앱을 생성한 후에는 암호를 변경할 수 없습니다. 사용하고 있는 암호를 교체하려는 경우 새 암호가 있는 새 앱을 생성할 수 있습니다. 앱을 삭제하여 해당 앱 클라이언트 ID를 사용하는 앱의 액세스를 차단할 수도 있습니다.

앱을 생성하려면

1. 사용자 풀 생성의 Apps(앱) 탭에서 앱 클라이언트 추가를 선택합니다.
2. 앱 클라이언트 이름을 지정합니다.
3. 앱의 갱신 토큰 만료(일수)를 지정합니다. 기본값은 30입니다. 이 값을 1~3650 사이의 값으로 변경할 수 있습니다.
4. 기본적으로 사용자 풀은 앱의 클라이언트 암호를 생성합니다. 클라이언트 암호를 생성하지 않으려면 클라이언트 보안키 생성을 선택 취소합니다.
5. 앱이 개발자 자격 증명(서명 버전 4 사용)이 필요한 서버 앱이고 SRP(보안 원격 프로토콜) 인증을 사용하지 않는 경우 서버 기반 인증용 로그인 API 활성화 (ADMIN_NO_SRP_AUTH)를 선택하여 서버 측 인증을 활성화합니다. 자세한 내용은 [관리 인증 흐름 \(p. 189\)](#) 단원을 참조하십시오.
6. 기본적으로 사용자 풀은 앱에서 모든 속성을 읽고 쓰도록 허용합니다. 앱에 다른 권한을 설정하려면 다음 단계를 수행하십시오.
 - a. 속성 읽기 및 쓰기 권한 설정을 선택합니다.
 - b. 다음 두 가지 방법을 모두 사용해 읽기 및 쓰기 권한을 설정할 수 있습니다.
 - 하나 이상의 범위를 선택합니다. 각 범위는 표준 속성 집합입니다. 자세한 내용은 [표준 OIDC 범위 목록](#)을 참조하십시오.
 - 개별 표준 또는 사용자 지정 속성을 선택합니다.

Note

앱의 쓰기 권한에서 필수 속성을 제거할 수 없습니다.

7. [Create app client]를 선택합니다.
8. 다른 앱을 생성하려면 Add an app(앱 추가)을 선택합니다.
9. 원하는 앱을 모두 생성한 경우 변경 사항 저장을 선택합니다.

CLI 명령 `create-user-pool-client` 및 `update-user-pool-client`와 API, [CreateUserPoolClient](#) 및 [UpdateUserPoolClient](#)를 사용하여 사용자 풀에서 앱 클라이언트를 생성하고 업데이트할 수도 있습니다.

사용자 풀 Lambda 트리거 구성

AWS Lambda 트리거를 사용하여 Amazon Cognito를 통해 워크플로우와 사용자 경험을 사용자 지정할 수 있습니다. 사전 가입, 사전 인증, 사용자 지정 메시지, 사후 인증, 사후 확인, 인증 챌린지 정의, 인증 챌린지 생성, 인증 챌린지 응답 확인 및 사용자 마이그레이션과 같은 Lambda 트리거를 생성할 수 있습니다.

사용자 마이그레이션 Lambda 트리거는 기존 사용자 관리 시스템에서 사용자 풀로의 순쉬운 마이그레이션을 허용합니다.

각 Lambda 트리거의 예제는 [Lambda 트리거를 사용하여 사용자 풀 워크플로우 사용자 지정 \(p. 118\)](#) 단원을 참조하십시오.

Note

사용자 지정 메시지 AWS Lambda 트리거는 이메일 및 SMS 메시지를 사용자 지정하는 향상된 방법입니다. 자세한 내용은 [Lambda 트리거를 사용하여 사용자 풀 워크플로우 사용자 지정 \(p. 118\)](#) 단원을 참조하십시오.

사용자 풀 생성 설정 검토

사용자 풀을 생성하기 전에 AWS Management 콘솔 콘솔에서 여러 설정을 검토하고 편집할 수 있습니다. Amazon Cognito는 사용자 풀 설정을 확인하고 변경할 사항이 있으면 경고합니다. 예:

Warning

이 사용자 풀에는 Amazon Cognito에서 SMS 메시지를 전송할 수 있도록 정의된 IAM 역할이 없으므로 2016년 8월 31일 이후 전화 번호 확인 또는 MFA 확인이 불가능합니다. 확인 패널에서 역할을 선택하여 IAM 역할을 정의할 수 있습니다.

메시지가 표시되면 풀 생성을 선택하기 전에 지침에 따라 수정하십시오.

사용자 풀 속성 분석

Note

기존 사용자 풀을 편집할 때에만 Analytics(분석) 탭이 나타납니다.

Amazon Pinpoint 분석을 이용하여 Amazon Cognito 사용자 풀 가입, 로그인, 인증 실패, 일 실사용자(DAU) 및 월 실사용자(MAU)를 추적할 수 있습니다. 또한 Android용 AWS Mobile SDK 또는 AWS Mobile SDK for iOS를 사용하여 앱에 고유한 사용자 속성을 설정할 수 있습니다. 그런 다음 사용자를 세분화하고 대상 푸시 알림에 전송하는데 이를 사용할 수 있습니다.

Analytics(분석) 탭에서 Amazon Cognito 앱 클라이언트에 대해 Amazon Pinpoint 프로젝트를 지정할 수 있습니다. 자세한 내용은 Amazon Cognito 사용자 풀을 통해 [Amazon Pinpoint 분석 사용 \(p. 155\)](#)을 참조하십시오.

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. 나열된 콘솔입니다.

여러 Amazon Cognito 앱을 단일 Amazon Pinpoint 프로젝트에 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. Amazon Cognito에서 사용자에게 추가 엔드포인트를 생성하기 위해 메일 주소 및 전화 번호를 Amazon Pinpoint에 전송하려면 Share user attribute data with Amazon Pinpoint(Amazon Pinpoint와 사용자 속성 데이터 공유)를 선택하십시오.

Note

엔드포인트는 Amazon Pinpoint를 사용하여 푸시 알림이 전송될 수 있는 사용자 디바이스를 고유하게 식별합니다. 엔드포인트에 대한 자세한 내용은, Amazon Pinpoint 개발자 안내서에서 [엔드포인트 추가](#)를 참조하십시오.

5. 이미 생성한 IAM 역할을 입력하거나 새 역할 생성을 선택하여 IAM 콘솔에서 새 역할을 생성합니다.
6. [Save changes]를 선택합니다.
7. 추가 앱 매핑을 지정하려면 Add app mapping(앱 매핑 추가)을 선택합니다.
8. [Save changes]를 선택합니다.

앱 클라이언트 설정 구성

Note

기존 사용자 풀을 편집할 때에만 앱 클라이언트 설정 탭이 나타납니다.

사용자를 가입 또는 로그인하기 위해 기본 제공 호스팅 페이지를 사용할 경우 또는 OAuth2.0 흐름을 사용할 경우 앱 클라이언트 설정 탭에서 앱의 IdP(자격 증명 공급자)를 하나 이상 구성해야 합니다. 자세한 내용은 [사용자 풀 앱 클라이언트 구성 \(p. 73\)](#) 단원을 참조하십시오.

사용자 풀의 앱 클라이언트 설정을 지정하려면

1. 활성화된 자격 증명 공급자에서 앱 클라이언트 탭에서 구성한 앱의 자격 증명 공급자를 선택합니다.
2. 원하는 콜백 URL을 쉼표로 구분하여 입력합니다. 이 URL은 선택한 모든 자격 증명 공급자에 적용됩니다.

Note

앱에서 URL을 사용하려면 콘솔에서 URL을 등록하거나 CLI 또는 API를 사용하여 URL을 등록해야 합니다.

3. 원하는 로그인 URL을 쉼표로 구분하여 입력합니다.

Note

앱에서 URL을 사용하려면 콘솔에서 URL을 등록하거나 CLI 또는 API를 사용하여 URL을 등록해야 합니다.

4. OAuth 2.0에서 다음 중 원하는 옵션을 선택합니다. 자세한 내용은 [앱 클라이언트 설정 개요 \(p. 73\)](#) 및 [OAuth 2.0 사양](#)을 참조하십시오.
 - 허용된 OAuth Flows에서 Authorized code grant(권한 있는 코드 허용)와 Implicit grant(암시적 허용)를 선택합니다. 사용자를 대신해서가 아니라 자체적으로 액세스 토큰을 요청해야 하는 경우에만 Client credentials(클라이언트 자격 증명)를 선택합니다.
 - 허용된 OAuth Scopes에서 원하는 범위를 선택합니다. 각 범위는 하나 이상의 표준 속성 집합입니다.
 - 허용된 사용자 지정 범위에서 정의한 사용자 지정 범위에서 원하는 범위를 선택합니다. 사용자 지정 범위는 리소스 서버 탭에서 정의됩니다. 자세한 내용은 [사용자 풀의 리소스 서버 정의 \(p. 86\)](#) 단원을 참조하십시오.

사용자 풀의 도메인 이름 추가

Note

기존 사용자 풀을 편집할 때에만 도메인 이름 탭이 나타납니다.

도메인 이름 탭에서 고유한 접두사 도메인 이름을 입력할 수 있습니다. 앱의 도메인은 `https://<domain_prefix>.auth.<region>.amazoncognito.com`입니다.

앱의 전체 URL은 `https://example.auth.us-east-1.amazoncognito.com/login?redirect_uri=https://www.google.com&response_type=code&client_id=<client_id_value>`와 같습니다.

자세한 내용은 [사용자 풀 도메인 구성](#) (p. 77) 단원을 참조하십시오.

Important

앱의 URL에 액세스하려면 콜백 및 리디렉션 URL과 같은 앱 클라이언트 설정을 지정해야 합니다. 자세한 내용은 [앱 클라이언트 설정 구성](#) (p. 216) 단원을 참조하십시오.

사용자 풀에 도메인 이름을 지정하려면

1. Prefix domain name(접두사 도메인 이름) 상자에 원하는 도메인 이름을 입력합니다.
2. 필요에 따라 Check availability(가용성 확인)를 선택합니다.
3. [Save changes]를 선택합니다.

기본 제공 앱 UI를 사용자 지정하여 사용자 가입 및 로그인

Note

기존 사용자 풀을 편집할 때에만 UI 사용자 지정 탭이 나타납니다.

UI 사용자 지정 탭에서 기본 앱 UI에 고유한 사용자 지정을 추가할 수 있습니다.

사용자 지정 필드에 대한 자세한 내용은 [내장 로그인 및 가입 웹페이지 사용자 지정](#) (p. 82) 단원을 참조하십시오.

Note

사용자 풀에 대한 세부 사항을 포함시켜 URL `https://<your_domain>/login?response_type=code&client_id=<your_app_client_id>&redirect_uri=<your_callback_url>`을 구성하고 이를 브라우저에 입력하면 사용자가 지정한 호스팅 UI를 볼 수 있습니다. 브라우저를 새로 고침하고 콘솔의 변경 내용이 나타나기까지 최대 1분 정도 기다려야 할 수도 있습니다. 도메인 이름 탭에 도메인이 나타납니다. 앱 클라이언트 설정 탭에 앱 클라이언트 ID와 콜백 URL이 나타납니다.

기본 제공 앱 UI를 사용자 지정하려면

1. 사용자 지정할 앱 클라이언트로 가서 이전에 앱 클라이언트 탭에서 생성한 앱 클라이언트의 드롭다운 메뉴에서 사용자 지정을 원하는 앱을 선택합니다.
2. 기본 앱 UI에 로고를 추가하려면 파일 선택을 선택하거나 로고 상자로 파일을 끌어 옵니다.
3. CSS 사용자 지정(선택 사항)으로 가서 기본값에서 다양한 속성을 변경하여 앱의 모양을 사용자 지정할 수 있습니다.

4. [Save changes]를 선택합니다.

사용자 풀의 리소스 서버 추가

Note

기존 사용자 풀을 편집할 때만 리소스 서버 탭이 나타납니다.

리소스 서버는 액세스 보호가 되는 리소스의 서버로서, 액세스 토큰을 가진 앱에서 인증된 요청을 처리합니다. 범위는 앱이 리소스에 요청할 수 있는 액세스의 수준입니다.

리소스 서버 탭에서 사용자 풀에 대한 사용자 지정 리소스 서버와 범위를 정의할 수 있습니다. 자세한 내용은 [사용자 풀의 리소스 서버 정의 \(p. 86\)](#) 단원을 참조하십시오.

사용자 지정 리소스 서버를 정의하려면

1. 리소스 서버 추가를 선택합니다.
2. 예를 들어 Photo Server와 같이 리소스 서버의 이름을 입력합니다.
3. 예를 들어 com.example.photos와 같이 리소스 서버의 ID를 입력합니다.
4. read 및 write와 같이 리소스에 대한 사용자 지정 범위의 이름을 입력합니다.
5. 범위 이름 각각에 대해 view your photos 및 update your photos와 같이 설명을 입력합니다.

정의한 각각의 사용자 지정 범위가 OAuth2.0 허용된 사용자 지정 범위 아래 앱 클라이언트 설정 탭에 나타납니다(예: com.example.photos/read).

사용자 풀에 자격 증명 공급자 구성

Note

기존 사용자 풀을 편집할 때에만 자격 증명 공급자 탭이 나타납니다.

자격 증명 공급자 탭에서 사용자 풀의 IdP(자격 증명 공급자)를 지정할 수 있습니다. 자세한 내용은 [타사를 통한 사용자 풀 로그인 추가 \(p. 88\)](#) 단원을 참조하십시오.

주제

- 사용자가 소셜 자격 증명 공급자를 사용하여 로그인할 수 있도록 허용 (p. 218)
- 사용자가 OpenID Connect (OIDC) 자격 증명 공급자를 사용하여 로그인할 수 있도록 허용 (p. 219)
- 사용자에게 SAML을 사용한 로그인 허용 (p. 220)

사용자가 소셜 자격 증명 공급자를 사용하여 로그인할 수 있도록 허용

Amazon Cognito 사용자 풀에 연동을 사용하여 Facebook, Google, Login with Amazon 등등 소셜 자격 증명 공급자를 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.

아래는 소셜 자격 증명 공급자의 이용에 도움이 되는 몇 가지 기본 용어들입니다.

- [Google 자격 증명 플랫폼](#)
- [개발자를 위한 Facebook](#)

- [Login with Amazon](#)

사용자가 소셜 자격 증명 공급자를 이용하여 로그인할 수 있도록 허용하려면

1. Facebook, Google, Login with Amazon 등 소셜 자격 증명 공급자를 선택합니다.
 2. Facebook(또는 Google이나 Amazon) 앱 ID로 Facebook, Google 또는 Login with Amazon 클라이언트 앱을 생성할 때 받았던 앱 ID를 입력합니다.
 3. 앱 보안에 클라이언트 앱을 생성할 때 받았던 앱 암호를 입력합니다.
 4. Authorize scopes(승인 범위)에 사용자 풀 속성에 매핑하고자 하는 소셜 자격 증명 공급자 범위의 이름을 입력합니다. 범위는 앱에서 액세스하고자 하는 사용자 속성(예: name 및 email)을 정의합니다. Facebook의 경우 쉼표로 구분해야 합니다(예: public_profile, email). Google 및 Login with Amazon의 경우 공백으로 구분해야 합니다(예: Google의 경우 profile email openid, Login with Amazon의 경우 profile postal_code).
- 최종 사용자는 앱에 이러한 속성들을 제공한다는 데 동의하라는 요청 메시지를 받게 됩니다. 범위에 대한 자세한 내용은 Google, Facebook 및 Login with Amazon의 설명서를 참조하십시오.
5. Facebook 활성화(또는 Google 활성화나 Amazon을 사용한 로그인 활성화)를 선택합니다.

소셜 IdP에 관한 자세한 내용은 [사용자 풀에 소셜 자격 증명 공급자 추가 \(p. 88\)](#) 섹션을 참조하십시오.

사용자가 OpenID Connect (OIDC) 자격 증명 공급자를 사용하여 로그인할 수 있도록 허용

사용자들이 Salesforce 또는 Ping Identity 같은 OIDC 자격 증명 공급자(IdP)를 통해 로그인하게 할 수 있습니다.

1. [Amazon Cognito 콘솔](#)로 이동합니다. AWS 자격 증명을 입력하라는 메시지가 나타날 수 있습니다.
2. [Manage your User Pools]를 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 [사용자 풀을 생성합니다](#).
4. 왼쪽 탐색 모음에서 자격 증명 공급자를 선택합니다.
5. OpenId Connect를 선택합니다.
6. Provider name(공급자 이름)에 고유한 이름을 입력합니다.
7. OIDC IdP의 클라이언트 ID를 클라이언트 ID에 입력합니다.
8. OIDC IdP의 클라이언트 암호를 클라이언트 암호에 입력합니다.
9. 드롭다운 목록에서, userinfo 엔드포인트에서 사용자 세부 정보를 Attributes request method(속성 요청 메서드)로 가져올 때 사용하는 HTTP 메서드(GET 또는 POST)를 선택합니다.
10. 승인하려는 범위의 이름을 입력합니다. 범위는 애플리케이션서 액세스하고자 하는 사용자 속성(예: name 및 email)을 정의합니다. 범위는 [OAuth 2.0](#) 사양에 따라 공백으로 구분됩니다.

애플리케이션 사용자는 앱에 이러한 속성들을 제공한다는 데 동의하라는 요청 메시지를 받게 됩니다.

11. IdP의 URL을 입력하고 검색 실행을 선택합니다.

예를 들어, Salesforce는 다음 URL을 사용합니다.

`https://login.salesforce.com`

Note

URL은 `https://`로 시작해야 하며, 슬래시 /로 끝나면 안 됩니다.

- 검색 실행에 실패한 경우에는 권한 부여 엔드포인트, 토큰 엔드포인트, Userinfo 엔드포인트 및 Jwks uri([JSON 웹 키](#)의 위치)를 제공해야 합니다.
12. 공급자 생성을 선택합니다.

13. 왼쪽 탐색 모음에서 앱 클라이언트 설정을 선택합니다.
14. OIDC 공급자를 활성화된 자격 증명 공급자 중 하나로 선택합니다.
15. Amazon Cognito 권한 부여 서버에 대한 콜백 URL을 입력하여 사용자가 인증된 후 호출합니다. 이것은 로그인 성공 후 사용자가 리디렉션되는 페이지의 URL입니다.

```
https://www.example.com
```

16. 허용된 OAuth Flows에서 Authorization code grant(권한 부여 코드 허용)와 Implicit code grant(암시적 코드 허용)를 모두 활성화합니다.

특별히 하나를 제외하고 싶은 경우가 아니라면 허용된 OAuth Scopes 확인란을 모두 선택합니다.
17. [Save changes]를 선택합니다.
18. 왼쪽 탐색 모음의 속성 매핑 탭에서 OIDC 클레임 매핑을 사용자 풀 속성에 추가합니다.
 - a. 기본적으로 OIDC 클레임 sub는 사용자 풀 속성 사용자 이름으로 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.
 - b. 드롭다운 목록에서 대상 사용자 풀 속성을 선택합니다.
 - c. [Save changes]를 선택합니다.
 - d. 요약본으로 이동을 선택합니다.

OIDC IdP에 관한 자세한 내용은 [사용자 풀에 OIDC 자격 증명 공급자 추가 \(p. 99\)](#) 섹션을 참조하십시오.

사용자에게 SAML을 사용한 로그인 허용

Amazon Cognito 사용자 풀에 연동을 사용하여 SAML IdP(자격 증명 공급자)와 통합할 수 있습니다. 파일을 업로드하거나 메타데이터 문서 엔드포인트 URL을 입력하여 메타데이터 문서를 제공해야 합니다. 타사 SAML IdP의 메타데이터 문서를 얻는 방법에 대한 자세한 내용은 [타사 SAML 자격 증명 공급자와 Amazon Cognito 사용자 풀 통합 \(p. 98\)](#)을 참조하십시오.

사용자에게 SAML을 사용한 로그인을 허용하려면

1. SAML을 선택하여 SAML 자격 증명 공급자 옵션을 표시합니다.
2. 메타데이터 문서를 업로드하려면 파일 선택을 선택하거나 메타데이터 문서 엔드포인트 URL을 입력합니다. 메타데이터 문서는 유효한 XML 파일이어야 합니다.
3. 예를 들어 "SAML_provider_1"과 같이 SAML 공급자 이름 및 원하는 식별자를 입력합니다. 공급자 이름은 필수 사항이고 식별자는 선택 사항입니다. 자세한 내용은 [사용자 풀에 SAML 자격 증명 공급자 추가 \(p. 92\)](#) 단원을 참조하십시오.
4. Amazon Cognito에서 로그아웃할 때 SAML IdP에서 사용자를 로그아웃되도록 하려면 Enable IdP sign out flow(IdP 로그아웃 흐름 활성화)를 선택합니다.

이 흐름을 활성화하면 [로그아웃 엔드포인트 \(p. 326\)](#)가 호출될 때 서명된 로그아웃 요청이 SAML IdP로 전송됩니다.

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.

5. 공급자 생성을 선택합니다.
6. 추가 공급자를 생성하려면 이전 단계를 반복합니다.

Note

HTTPS 메타데이터 엔드포인트 URL을 통해 SAML 자격 증명 공급자를 생성하는 동안 "<metadata endpoint>에서 메타데이터를 검색하는 동안 오류 발생"과 같은 `InvalidParameterException`이 표시되면 메타데이터 엔드포인트에 SSL이 올바르게 설정되어 있는지 그리고 이와 연관된 유효한 SSL 인증서가 있는지를 확인하십시오.

서명 인증서를 추가하도록 SAML IdP를 설정하려면

- 서명된 로그아웃 요청을 확인하기 위해 자격 증명 공급자가 사용할 퍼블릭 키를 포함하는 인증서를 가져오려면 연동 콘솔 페이지의 자격 증명 공급자에서 SAML 대화 상자의 Active SAML Providers(활성 SAML 공급자)에 있는 Show signing certificate(서명 인증서 표시)를 선택합니다.

SAML IdP에 관한 자세한 내용은 [사용자 풀에 SAML 자격 증명 공급자 추가 \(p. 92\)](#) 섹션을 참조하십시오.

사용자 풀의 속성 매핑 구성

Note

기존 사용자 풀을 편집할 때에만 속성 매핑 탭이 나타납니다.

속성 매핑 탭에서 사용자 풀 속성에 IdP(자격 증명 공급자) 속성 또는 어설션을 매핑할 수 있습니다. 자세한 내용은 [사용자 풀에 대한 자격 증명 공급자 속성 매핑 지정 \(p. 104\)](#) 단원을 참조하십시오.

Note

현재로는 Facebook id, Google sub 및 Login with Amazon user_id 속성만 Amazon Cognito 사용자 풀의 username 속성에 매핑할 수 있습니다.

Note

사용자 풀의 속성은 매핑된 자격 증명 공급자 속성의 값에 맞게 충분히 커야 합니다. 그렇지 않으면 사용자가 로그인을 할 때 오류가 발생합니다. 자격 증명 공급자 토큰에 매핑할 경우, 사용자 지정 속성은 최대 2,048자로 설정해야 합니다.

사용자 풀에 필요한 모든 속성에 대한 매핑을 만들어야 합니다.

사용자 풀에 대해 소셜 자격 증명 공급자 속성 매핑을 지정하려면

1. Facebook, Google 또는 Amazon 탭을 선택합니다.
2. 매핑할 각 속성에 대해 다음 단계를 수행합니다.
 - a. 캡처 확인란을 선택합니다.
 - b. 사용자 풀 속성 필드의 드롭다운 목록에서 소셜 자격 증명 공급자 속성을 매핑할 사용자 풀 속성을 선택합니다.
 - c. 더 많은 속성이 필요할 경우 Facebook 속성 추가(또는 Google 속성 추가나 Add Amazon attribute(Amazon 속성 추가))를 선택하고 다음 단계를 수행합니다.
 - i. Facebook attribute(Facebook 속성)(또는 Google attribute(Google 속성)나 Amazon attribute(Amazon 속성)) 필드에서 매핑하려는 속성의 이름을 입력합니다.
 - ii. 사용자 풀 속성의 드롭다운 목록에서 소셜 자격 증명 공급자 속성에 매핑할 사용자 풀 속성을 선택합니다.
 - d. [Save changes]를 선택합니다.

사용자 풀에 대해 SAML 공급자 속성 매핑을 지정하려면

1. SAML 탭을 선택합니다.

2. 매핑할 각 속성에 대해 다음 단계를 수행합니다.
 - a. SAML 속성 추가를 선택합니다.
 - b. SAML 속성 필드에서 매핑할 SAML 속성의 이름을 입력합니다.
 - c. 사용자 풀 속성의 드롭다운 목록에서 SAML 속성에 매핑할 사용자 풀 속성을 선택합니다
 - d. [Save changes]를 선택합니다.

Amazon Cognito 자격 증명 풀(연동 자격 증명)

Amazon Cognito 자격 증명 풀(연동 자격 증명)을 사용하여 사용자의 고유한 자격 증명을 만들고 자격 증명 공급자와 연동할 수 있습니다. 자격 증명 풀로 권한이 제한된 임시 AWS 자격 증명을 얻어 다른 AWS 서비스에 액세스할 수 있습니다. Amazon Cognito 자격 증명 풀은 다음 자격 증명 공급자를 지원합니다.

- 퍼블릭 공급자: [Login with Amazon\(자격 증명 풀\)](#) (p. 254), [Facebook\(자격 증명 풀\)](#) (p. 249), [Google\(자격 증명 풀\)](#) (p. 257).
- [OpenID Connect 공급자\(자격 증명 풀\)](#) (p. 263)
- [SAML 자격 증명 공급자\(자격 증명 풀\)](#) (p. 265)
- [개발자 인증 자격 증명\(자격 증명 풀\)](#) (p. 267)

Amazon Cognito 자격 증명 풀 리전 가용성에 대한 자세한 내용은 [AWS 서비스 리전 가용성](#) 단원을 참조하십시오.

Amazon Cognito 자격 증명 풀에 대한 자세한 내용은 다음 주제를 참조하십시오.

주제

- [Amazon Cognito 자격 증명 시작하기\(연동 자격 증명\)](#) (p. 223)
- [자격 증명 풀 사용\(연동 자격 증명\)](#) (p. 225)
- [자격 증명 풀 개념\(연동 자격 증명\)](#) (p. 229)
- [역할 기반 액세스 제어](#) (p. 238)
- [자격 증명 받기](#) (p. 242)
- [AWS 제품 액세스](#) (p. 248)
- [자격 증명 풀\(연동 자격 증명\) 외부 자격 증명 공급자](#) (p. 249)
- [개발자 인증 자격 증명\(자격 증명 풀\)](#) (p. 267)
- [인증되지 않은 사용자를 인증된 사용자로 전환\(자격 증명 풀\)](#) (p. 277)

Amazon Cognito 자격 증명 시작하기(연동 자격 증명)

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

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

자격 증명 풀을 사용하면 다른 AWS 제품에 직접 액세스하거나 Amazon API Gateway를 통해 리소스에 액세스하도록 정의하는 권한을 가진 임시 AWS 자격 증명을 얻을 수 있습니다.

주제

- [AWS 계정에 가입](#) (p. 224)

- [Amazon Cognito에서 자격 증명 풀 만들기 \(p. 224\)](#)
- [Mobile 또는 JavaScript SDK 설치 \(p. 224\)](#)
- [자격 증명 공급자 통합 \(p. 225\)](#)
- [자격 증명 얻기 \(p. 225\)](#)

AWS 계정에 가입

Amazon Cognito 자격 증명 풀을 사용하려면 AWS 계정이 있어야 합니다. 아직 계정이 없는 경우 다음 절차를 사용하여 가입하십시오.

AWS 계정에 가입하려면 다음을 수행합니다.

1. <https://aws.amazon.com/>을 열고 Create an AWS Account(AWS 계정 생성)를 선택합니다.

Note

전에 AWS 계정 루트 사용자 자격 증명을 사용하여 AWS Management 콘솔에 로그인한 적이 있는 경우 Sign in to a different account(다른 계정으로 로그인)를 선택합니다. 전에 IAM 자격 증명을 사용하여 콘솔에 로그인한 적이 있는 경우 Sign-in using root account credentials(루트 계정 자격 증명으로 로그인)를 선택합니다. 그런 다음 Create a new AWS account(새 AWS 계정 생성)를 선택합니다.

2. 온라인 지시 사항을 따릅니다.

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

Amazon Cognito에서 자격 증명 풀 만들기

Amazon Cognito 콘솔을 통해 자격 증명 풀을 만들거나 AWS 명령줄 인터페이스(CLI) 또는 Amazon Cognito API를 사용할 수 있습니다.

콘솔에서 새 자격 증명 풀을 만들려면

1. [Amazon Cognito 콘솔](#)에 로그인하고 연동 자격 증명 관리와 새 자격 증명 풀 만들기를 차례대로 선택합니다.
2. 자격 증명 풀의 이름을 입력합니다.
3. 인증되지 않은 자격 증명을 활성화하려면 인증되지 않은 자격 증명 축소 가능 섹션에서 인증되지 않은 자격 증명에 대한 액세스 활성화를 선택합니다.
4. 필요한 경우 인증 공급자 섹션에서 인증 공급자를 구성합니다.
5. [Create Pool]을 선택합니다.

Note

유효한 자격 증명 풀을 만들려면 하나 이상의 자격 증명이 필요합니다.

6. AWS 리소스에 액세스하라는 메시지가 표시됩니다.

Allow(허용)를 선택하여 자격 증명 풀과 연결된 기본 역할 2개(인증되지 않은 사용자의 역할 1개, 인증된 사용자의 역할 1개)를 만듭니다. 이 기본 역할은 에 대한 자격 증명 풀 액세스를 제공합니다. IAM 콘솔에서 자격 증명 풀과 연결된 역할을 수정할 수 있습니다.

Mobile 또는 JavaScript SDK 설치

Amazon Cognito 자격 증명 풀을 사용하려면 AWS Mobile 또는 JavaScript SDK를 설치하고 구성해야 합니다. 자세한 내용은 다음 주제를 참조하십시오.

- [AWS Mobile SDK for Android 설정](#)
- [AWS Mobile SDK for iOS 설정](#)
- [AWS SDK for JavaScript 설정](#)
- [AWS Mobile SDK for Unity 설정](#)
- [AWS Mobile SDK for .NET 및 Xamarin 설정](#)

자격 증명 공급자 통합

Amazon Cognito 자격 증명 풀(연동 자격 증명)은 인증되지 않은 자격 증명뿐 아니라 Amazon Cognito 사용자 풀, 연동된 자격 증명 공급자(Amazon, Facebook, Google 및 SAML 자격 증명 공급자 포함)를 통한 사용자 인증을 지원합니다. 이 기능은 자체 백엔드 인증 프로세스를 통해 사용자를 등록하고 인증할 수 있는 [개발자 인증 자격 증명\(자격 증명 풀\)](#) (p. 267)도 지원합니다.

Amazon Cognito 사용자 풀을 사용하여 사용자 디렉터리를 만드는 데 대한 자세한 내용은 [Amazon Cognito 사용자 풀](#) (p. 12) 및 [로그인 후 자격 증명 풀을 사용하여 AWS 서비스 액세스](#) (p. 199) 단원을 참조하십시오.

외부 자격 증명 공급자 사용에 대한 자세한 내용은 [자격 증명 풀\(연동 자격 증명\) 외부 자격 증명 공급자](#) (p. 249) 단원을 참조하십시오.

자체 백엔드 인증 프로세스 통합에 대한 자세한 내용은 [개발자 인증 자격 증명\(자격 증명 풀\)](#) (p. 267) 단원을 참조하십시오.

자격 증명 얻기

Amazon Cognito 자격 증명 풀은 게스트(미인증) 및 토큰을 인증하고 수신한 사용자의 임시 AWS 자격 증명을 제공합니다. 이 AWS 자격 증명을 사용하여 앱을 통해 AWS 내부나 외부의 백엔드에 안전하게 액세스할 수 있습니다. [자격 증명 받기](#) (p. 242) 단원을 참조하십시오.

자격 증명 풀 사용(연동 자격 증명)

Amazon Cognito 자격 증명 풀은 게스트(미인증) 및 토큰을 인증하고 수신한 사용자의 임시 AWS 자격 증명을 제공합니다. 자격 증명 풀은 계정에 관련된 사용자 자격 증명 데이터의 저장소입니다.

콘솔에서 새 자격 증명 풀을 만들려면

1. [Amazon Cognito 콘솔](#)에 로그인하고 연동 자격 증명 관리와 새 자격 증명 풀 만들기를 차례대로 선택합니다.
2. 자격 증명 풀의 이름을 입력합니다.
3. 인증되지 않은 자격 증명을 활성화하려면 인증되지 않은 자격 증명 축소 가능 섹션에서 인증되지 않은 자격 증명에 대한 액세스 활성화를 선택합니다.
4. 필요한 경우 인증 공급자 섹션에서 인증 공급자를 구성합니다.
5. [Create Pool]을 선택합니다.

Note

유효한 자격 증명 풀을 만들려면 하나 이상의 자격 증명이 필요합니다.

6. AWS 리소스에 액세스하라는 메시지가 표시됩니다.

Allow(허용)를 선택하여 자격 증명 풀과 연결된 기본 역할 2개(인증되지 않은 사용자의 역할 1개, 인증된 사용자의 역할 1개)를 만듭니다. 이 기본 역할은 에 대한 자격 증명 풀 액세스를 제공합니다. IAM 콘솔에서 자격 증명 풀과 연결된 역할을 수정할 수 있습니다. Amazon Cognito 콘솔 작업에 대한 추가 지침은 [Amazon Cognito 콘솔 사용](#) (p. 3) 단원을 참조하십시오.

사용자 IAM 역할

IAM 역할은 사용자가 [Amazon Cognito Sync \(p. 281\)](#)와 같은 AWS 리소스에 액세스할 수 있는 권한을 정의합니다. 생성된 역할을 애플리케이션 사용자가 수입합니다. 인증된 사용자와 인증되지 않은 사용자에게 다른 역할을 지정할 수 있습니다. IAM 역할에 대한 자세한 내용은 [IAM 역할 \(p. 234\)](#) 단원을 참조하십시오.

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\(연동 자격 증명\) 외부 자격 증명 공급자 \(p. 249\)](#) 단원을 참조하십시오.
- 자체 백엔드 인증 프로세스를 구성하려면 [개발자 인증 자격 증명\(자격 증명 풀\) \(p. 267\)](#) 단원을 참조하십시오.

미인증 자격 증명 활성화 또는 비활성화

Amazon Cognito 자격 증명 풀은 자격 증명 공급자를 통해 인증하지 않는 사용자에게 대해 고유한 식별자 및 AWS 자격 증명을 제공하여 미인증 자격 증명을 지원할 수 있습니다. 애플리케이션에서 로그인하지 않은 사용자를 허용하는 경우 미인증 자격 증명에 대한 액세스를 활성화할 수 있습니다. 자세한 내용은 [Amazon Cognito 자격 증명 시작하기\(연동 자격 증명\) \(p. 223\)](#) 단원을 참조하십시오.

Amazon Cognito 콘솔에서 연동 자격 증명 관리를 선택합니다.

1. 미인증 자격 증명을 활성화 또는 비활성화할 자격 증명 풀 이름을 클릭합니다. 자격 증명 풀에 대한 대시보드 페이지가 표시됩니다.
2. 대시보드 페이지의 우측 상단 모서리에서 자격 증명 풀 편집을 클릭합니다. 자격 증명 풀 편집 페이지가 표시됩니다.
3. 아래로 스크롤하고 인증되지 않은 자격 증명을 클릭하여 확장합니다.
4. 확인란을 선택하여 미인증 자격 증명에 대한 액세스를 활성화 또는 비활성화합니다.
5. 변경 사항 저장을 클릭합니다.

자격 증명 유형과 연관된 역할 변경

자격 증명 풀에서는 인증 및 미인증의 두 가지 자격 증명 유형을 정의합니다. 자격 증명 풀의 모든 자격 증명은 인증 또는 미인증입니다. 인증 자격 증명은 퍼블릭 로그인 공급자(Amazon Cognito 사용자 풀, Facebook, Amazon, Google, SAML 또는 OpenID Connect 공급자) 또는 개발자 공급자(자체 백엔드 인증 프로세스)에 의해 인증된 사용자에게 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 AWS 제품을 나타내는 정책이 첨부되어 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 자격 증명 유형에 다른 역할을 할당하여 자격 증명 유형이 액세스할 수 있는 AWS 제품을 제어할 수 있습니다. 자격 증명 풀에서 역할과 연관된 정책을 보거나 수정하려면 [AWS IAM 콘솔](#)을 참조하십시오.

Amazon Cognito 자격 증명 풀(연동 자격 증명) 콘솔을 사용하면 자격 증명 유형과 연관된 역할을 변경할 수 있습니다. [Amazon Cognito 콘솔](#)에서 연동 자격 증명 관리를 선택합니다.

1. 역할을 수정할 자격 증명 풀 이름을 클릭합니다. 자격 증명 풀에 대한 대시보드 페이지가 표시됩니다.
2. 대시보드 페이지의 우측 상단 모서리에서 자격 증명 풀 편집을 클릭합니다. 자격 증명 풀 편집 페이지가 표시됩니다.

3. 인증되지 않은 역할 및 인증된 역할 옆에 있는 드롭다운 메뉴를 사용하여 역할을 변경합니다. [AWS IAM 콘솔](#)에서 새 역할 생성을 클릭하여 각 자격 증명 유형과 연관된 역할을 생성하거나 수정합니다. 자세한 내용은 [IAM 역할](#)을 참조하십시오.

인증 공급자 활성화 또는 편집

사용자가 퍼블릭 자격 증명 공급자(예: Amazon Cognito 사용자 풀, Facebook, Amazon)를 이용하여 인증할 수 있도록 하는 경우 Amazon Cognito 자격 증명 풀(연동 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 \(p. 238\)](#) 단원을 참조하십시오.

Warning

자격 증명 풀과 연결된 애플리케이션 ID를 변경하면 기존 사용자가 자격 증명 풀을 이용하여 인증할 수 없게 됩니다. [자격 증명 풀\(연동 자격 증명\) 외부 자격 증명 공급자 \(p. 249\)](#) 단원에 대해 자세히 알아보십시오.

[Amazon Cognito 콘솔](#)에서 연동 자격 증명 관리를 선택합니다.

1. 외부 공급자를 활성화할 자격 증명 풀 이름을 클릭합니다. 자격 증명 풀에 대한 대시보드 페이지가 표시됩니다.
2. 대시보드 페이지의 우측 상단 모서리에서 자격 증명 풀 편집을 클릭합니다. 자격 증명 풀 편집 페이지가 표시됩니다.
3. 아래로 스크롤하고 인증 공급자를 클릭하여 확장합니다.
4. 해당하는 공급자에 대한 탭을 클릭하고 인증 공급자와 연관된 필수 정보를 입력합니다.

자격 증명 풀 삭제

[Amazon Cognito 콘솔](#)에서 연동 자격 증명 관리를 선택합니다.

1. 삭제할 자격 증명 풀 이름을 클릭합니다. 자격 증명 풀에 대한 대시보드 페이지가 표시됩니다.
2. 대시보드 페이지의 우측 상단 모서리에서 자격 증명 풀 편집을 클릭합니다. 자격 증명 풀 편집 페이지가 표시됩니다.
3. 아래로 스크롤하고 자격 증명 풀 삭제를 클릭하여 확장합니다.
4. 자격 증명 풀 삭제를 클릭합니다.
5. 풀 삭제를 클릭합니다.

Warning

삭제 버튼을 클릭하면 자격 증명 풀과 여기에 포함된 모든 사용자 데이터를 영구적으로 삭제합니다. 자격 증명 풀을 삭제할 경우 자격 증명 풀을 사용하는 애플리케이션과 다른 서비스를 더 이상 사용할 수 없습니다.

자격 증명 풀에서 자격 증명 삭제

[Amazon Cognito 콘솔](#)에서 연동 자격 증명 관리를 선택합니다.

1. 삭제할 자격 증명이 포함되어 있는 자격 증명 풀 이름을 클릭합니다. 자격 증명 풀에 대한 대시보드 페이지가 표시됩니다.
2. 대시보드 페이지의 왼쪽 탐색에서 자격 증명 브라우저를 클릭합니다. 자격 증명 페이지가 표시됩니다.

3. 자격 증명 페이지에서 삭제할 자격 증명 ID를 입력한 다음 검색을 클릭합니다.
4. 자격 증명 세부 정보 페이지에서 자격 증명 삭제 버튼을 클릭한 다음 Delete(삭제)를 클릭합니다.

데이터 세트 관리

애플리케이션에서 Amazon Cognito Sync 기능을 구현한 경우 Amazon Cognito 자격 증명 풀 콘솔에서는 개별 자격 증명에 대한 데이터 세트와 레코드를 수동으로 생성하고 삭제할 수 있습니다. 자격 증명 풀 콘솔에서 자격 증명의 데이터 세트 또는 레코드에 수행된 변경 사항은 콘솔에서 '동기화'를 클릭해야 저장되며 자격 증명에서 동기화를 호출해야 최종 사용자에게 표시됩니다. 개별 자격 증명에 대해 다른 장치에서 동기화되고 있는 데이터는 특정 자격 증명에 대한 목록 데이터 세트 페이지를 새로 고칠 경우 표시됩니다.

자격 증명에 대한 데이터 세트 생성

Amazon Cognito 자격 증명 풀 [Amazon Cognito 콘솔](#)에서 연동 자격 증명 관리를 선택합니다.

1. 데이터 세트를 생성할 자격 증명이 포함되어 있는 자격 증명 풀 이름을 클릭합니다. 자격 증명 풀에 대한 대시보드 페이지가 표시됩니다.
2. 대시보드 페이지의 왼쪽 탐색에서 자격 증명 브라우저를 클릭합니다. 자격 증명 페이지가 표시됩니다.
3. 자격 증명 페이지에서 데이터 세트를 생성할 자격 증명 ID를 입력한 다음 검색을 클릭합니다.
4. 해당 자격 증명에 대한 자격 증명 세부 정보 페이지에서 데이터 세트 생성 버튼을 클릭하고, 데이터 세트 이름을 입력한 다음 Create and edit dataset(데이터 세트 생성 및 편집)를 클릭합니다.
5. 현재 데이터 세트 페이지에서 레코드 생성을 클릭하여 해당 데이터 세트에 저장할 레코드를 생성합니다.
6. 해당 데이터 세트에 대한 키, 유효한 JSON 값 또는 저장할 값을 입력한 다음 JSON 형식 지정을 클릭하여 입력한 값을 꾸미고 올바른 JSON 형식인지 확인합니다. 작업을 마친 후 변경 사항 저장을 클릭합니다.
7. 동기화를 클릭하여 데이터 세트를 동기화합니다. 변경 사항은 [Synchronize]를 클릭해야 저장되며 자격 증명에서 동기화를 호출해야 사용자에게 표시됩니다. 동기화되지 않은 변경 사항을 무시하려면 무시할 변경 사항을 선택한 다음 Discard changes(변경 사항 폐기)를 클릭합니다.

자격 증명과 연관된 데이터 세트 삭제

[Amazon Cognito 콘솔](#)에서 연동 자격 증명 관리를 선택합니다.

1. 데이터 세트를 삭제할 자격 증명이 포함되어 있는 자격 증명 풀 이름을 클릭합니다. 자격 증명 풀에 대한 대시보드 페이지가 표시됩니다.
2. 대시보드 페이지의 왼쪽 탐색에서 자격 증명 브라우저를 클릭합니다. 자격 증명 페이지가 표시됩니다.
3. 자격 증명 페이지에서 삭제할 데이터 세트가 포함된 자격 증명 ID를 입력한 다음 검색을 클릭합니다.
4. 자격 증명 세부 정보 페이지에서 삭제할 데이터 세트 옆에 있는 확인란을 선택하고 삭제 선택됨과 Delete(삭제)를 차례대로 클릭합니다.

대량으로 데이터 게시

Amazon Cognito Sync 스토어에 이미 저장되어 있는 데이터를 Kinesis 스트림으로 내보내는 데 대량 게시를 사용할 수 있습니다. 모든 스트림을 대량 게시하는 방법에 대한 자세한 내용은 [Amazon Cognito 스트림 \(p. 306\)](#)을 참조하십시오.

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. 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.

Amazon Cognito 콘솔을 통해 푸시 동기화를 활성화할 수 있습니다. [Amazon Cognito 콘솔](#)에서 연동 자격 증명 관리를 선택합니다.

1. 푸시 동기화를 활성화할 자격 증명 풀 이름을 클릭합니다. 자격 증명 풀에 대한 대시보드 페이지가 표시됩니다.
2. 대시보드 페이지의 우측 상단 모서리에서 자격 증명 풀 편집을 클릭합니다. 자격 증명 풀 편집 페이지가 표시됩니다.
3. 아래로 스크롤하고 푸시 동기화를 클릭하여 확장합니다.
4. 서비스 역할 드롭다운 메뉴에서 SNS 알림을 전송할 권한을 Amazon Cognito에 부여하는 IAM 역할을 선택합니다. [AWS IAM 콘솔](#)에서 역할 생성을 클릭하여 자격 증명 풀과 연관된 역할을 생성하거나 수정합니다.
5. 플랫폼 애플리케이션을 선택한 다음 변경 사항 저장을 클릭합니다.

Amazon Cognito 스트림 설정

Amazon Cognito 스트림은 개발자에게 Amazon Cognito Sync에 저장된 데이터에 대한 제어와 통찰력을 제공합니다. 개발자는 이제 데이터가 업데이트 및 동기화될 때 이벤트를 수신하도록 Kinesis 스트림을 구성할 수 있습니다. Amazon Cognito는 각 데이터 세트 변경 사항을 사용자가 소유하는 Kinesis 스트림으로 실시간으로 푸시합니다. Amazon Cognito 콘솔에서 Amazon Cognito 스트림을 설정하는 방법에 대한 자세한 내용은 [Amazon Cognito 스트림 \(p. 306\)](#) 단원을 참조하십시오.

Amazon Cognito 이벤트 설정

Amazon Cognito 이벤트를 통해 Amazon Cognito Sync에서 중요한 이벤트에 반응하여 AWS Lambda 함수를 실행할 수 있습니다. 데이터 세트가 동기화될 경우 Amazon Cognito Sync에서는 동기화 트리거 이벤트가 발생합니다. 사용자가 데이터를 업데이트할 때 동기화 트리거 이벤트를 사용하여 작업을 수행할 수 있습니다. 콘솔에서 Amazon Cognito 이벤트를 설정하는 방법에 대한 자세한 내용은 [Amazon Cognito 이벤트 \(p. 308\)](#) 단원을 참조하십시오.

AWS Lambda에 대한 자세한 내용은 [AWS Lambda](#) 단원을 참조하십시오.

자격 증명 풀 개념(연동 자격 증명)

Amazon Cognito 자격 증명 풀을 사용하여 사용자의 고유한 자격 증명을 만들고 자격 증명 공급자를 사용하여 인증할 수 있습니다. 자격 증명으로 권한이 제한된 임시 AWS 자격 증명을 얻어 다른 AWS 서비스에 액세스할 수 있습니다. Amazon Cognito 자격 증명 풀은 퍼블릭 자격 증명 공급자(Amazon, 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 가용성](#) 단원을 참조하십시오. 자격 증명 풀 개념에 대한 자세한 내용은 다음 주제를 참조하십시오.

주제

- [자격 증명 풀\(연동 자격 증명\) 인증 흐름 \(p. 229\)](#)
- [IAM 역할 \(p. 234\)](#)
- [역할 신뢰 및 권한 \(p. 238\)](#)

자격 증명 풀(연동 자격 증명) 인증 흐름

Amazon Cognito는 디바이스와 플랫폼에서 일관성을 유지하는 최종 사용자에게 고유한 식별자 생성을 지원합니다. Amazon Cognito는 AWS 리소스에 액세스하기 위해 권한이 제한된 임시 자격 증명도 애플리케이션에

제공합니다. 이 페이지에서는 의 기본 인증 방법을 다루며, 자격 증명 풀 내에서 자격 증명의 수명 주기에 대해 설명합니다.

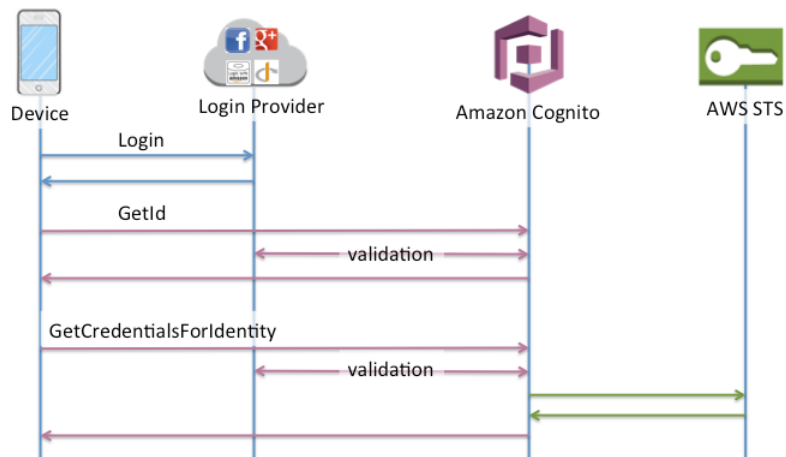
외부 공급자 인증 흐름

Amazon Cognito를 사용하여 인증하는 사용자는 자격 증명을 부트스트랩하기 위해 다단계 프로세스를 진행합니다. Amazon Cognito에는 퍼블릭 공급자와의 인증에 대해 향상된 및 기본의 두 가지 흐름이 있습니다.

이러한 흐름 중 하나를 완료하면 역할의 액세스 정책에 의해 정의된 대로 다른 AWS 제품에 액세스할 수 있습니다. 기본적으로 [Amazon Cognito 콘솔](#)은 Amazon Cognito Sync 스토어 및 Amazon Mobile Analytics에 액세스할 수 있는 역할을 생성합니다. 추가 액세스 권한을 부여하는 방법에 대한 자세한 내용은 [IAM 역할 \(p. 234\)](#)을 참조하십시오.

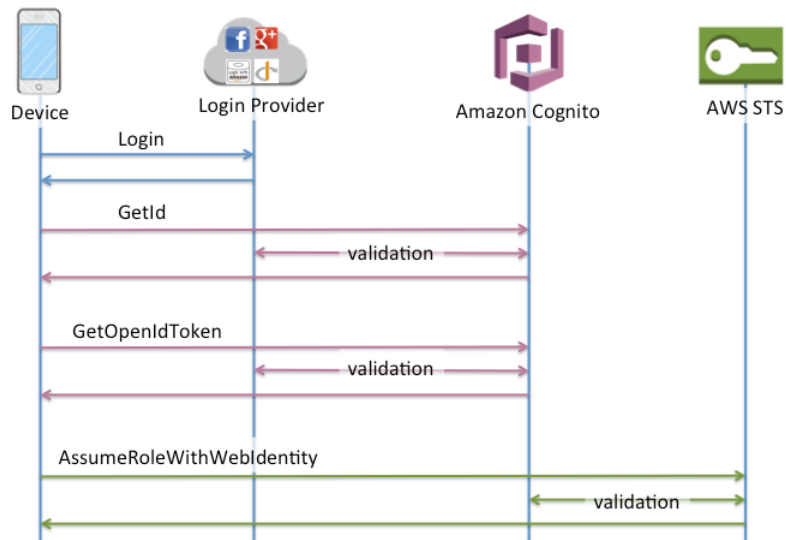
향상된(간소화된) 인증 흐름

1. `GetId`
2. `GetCredentialsForIdentity`



기본(클래식) 인증 흐름

1. `GetId`
2. `GetOpenIdToken`
3. `AssumeRoleWithWebIdentity`

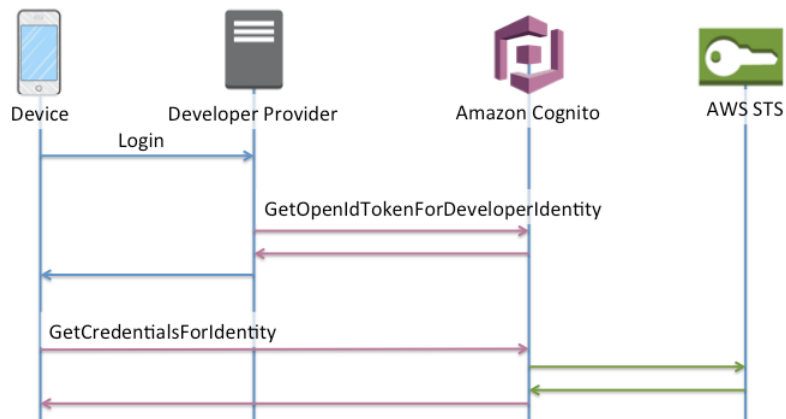


개발자 인증 자격 증명 인증 흐름

개발자 인증 자격 증명(자격 증명 풀) (p. 267)을 사용할 때 클라이언트는 Amazon Cognito에 속하지 않는 코드가 포함된 다른 인증 흐름을 사용하여 자체 인증 시스템에서 사용자를 검증합니다. 에 속하지 않는 코드는 다음과 같이 나타납니다.

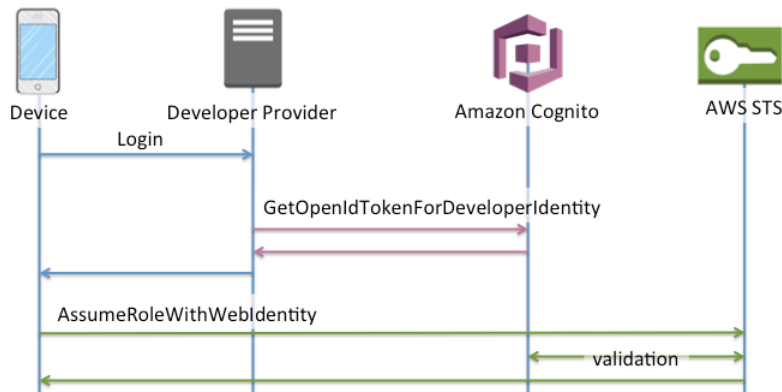
향상된 인증 흐름

1. 개발자 공급자를 통해 로그인(Amazon Cognito에 속하지 않는 코드)
2. 사용자 로그인 검증(Amazon Cognito에 속하지 않는 코드)
3. [GetOpenIdTokenForDeveloperIdentity](#)
4. [GetCredentialsForIdentity](#)



기본 인증 흐름

1. 개발자 공급자를 통해 로그인(Amazon Cognito에 속하지 않는 코드)
2. 사용자 로그인 검증(Amazon Cognito에 속하지 않는 코드)
3. [GetOpenIdTokenForDeveloperIdentity](#)
4. [AssumeRoleWithWebIdentity](#)



어떤 인증 흐름을 사용해야 하나요?

다음과 같이 향상된 흐름이 기본 흐름보다 더 많은 이점을 제공하므로 대부분의 고객의 경우 향상된 흐름이 올바른 선택입니다.

- 디바이스에서 자격 증명을 가져오기 위한 네트워크 호출 하나가 줄어듭니다.
- 모든 호출이 Amazon Cognito에 전송되므로 네트워크 연결 하나가 줄어듭니다.
- 더 이상 역할이 애플리케이션에 포함될 필요가 없으며 자격 증명을 부트스트래핑하기 위해 자격 증명 풀 ID와 리전만 필요합니다.

2015년 2월부터 [Amazon Cognito 콘솔](#)은 향상된 흐름을 사용한 예제 코드를 표시합니다. 또한 이 콘솔은 자격 증명 풀에 향상된 흐름을 사용하기 위해 필요한 역할 연결이 없는 경우 알림을 표시합니다.

다음은 향상된 흐름이 지원되는 최소 SDK 버전입니다.

SDK(최소 버전)

- iOS용 AWS SDK(2.0.14)
- Android용 AWS SDK(2.1.8)
- JavaScript용 AWS SDK(2.1.7)
- Unity용 AWS SDK(1.0.3)
- Xamarin용 AWS SDK(3.0.0.5)

콘솔에서 새 자격 증명 풀을 생성할 때 구성된 두 개의 기본 역할보다 더 많은 것을 원하는 경우 계속 기본 흐름을 사용하려고 할 수 있습니다.

API 요약

GetId

GetId API 호출은 Amazon Cognito에서 새 자격 증명을 설정하는 데 필요한 첫 번째 호출입니다.

미인증 액세스

Amazon Cognito는 애플리케이션에서 미인증 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 를 호출하기 위해 이 자격 증명 ID를 캐시해야 합니다. 브라우저에서 AWS Mobile SDK 및 JavaScript용 AWS SDK에는 사용자를 위해 이 캐싱을 처리하는 자격 증명 공급자가 있습니다.

인증 액세스

퍼블릭 로그인 공급자(Facebook, Google+, Login with Amazon)의 지원을 통해 애플리케이션을 구성한 경우 사용자는 해당 공급자에서 사용자를 식별하는 토큰(OAuth 또는 OpenID Connect)을 제공할 수도 있습니다. GetId 호출에 사용된 경우 Amazon Cognito는 새 인증 자격 증명을 생성하거나 이미 특정 로그인에 연관되어 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 생성한 애플리케이션 식별자와 일치함(예: Facebook 앱 ID)
- 토큰이 사용자 식별자와 일치함

GetCredentialsForIdentity

자격 증명 ID를 설정한 후 GetCredentialsForIdentity API를 호출할 수 있습니다. 이 API는 GetOpenIdToken과 AssumeRoleWithWebIdentity를 차례대로 호출하는 것과 기능적으로 동일합니다.

사용자 대신 Amazon Cognito에서 AssumeRoleWithWebIdentity를 호출하기 위해 자격 증명 풀에는 이와 연관된 IAM 역할이 있어야 합니다. 콘솔을 통하거나 수동으로 SetIdentityPoolRoles 작업을 통해 이를 수행할 수 있습니다(API 참조 확인).

GetOpenIdToken

자격 증명 ID를 설정한 후 GetOpenIdToken API 호출이 취소됩니다. 캐시된 자격 증명 ID가 있는 경우 앱 세션 중 수행한 첫 번째 호출이 될 수 있습니다.

미인증 액세스

미인증 자격 증명에 대한 토큰을 얻으려면 자격 증명 ID 자체만 필요합니다. 인증된 또는 비활성화된 자격 증명에 대한 미인증 토큰은 가져올 수 없습니다.

인증 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 전달한 자격 증명 ID가 반환된 자격 증명 ID일 수 없기 때문입니다.

로그인 연결

아직 자격 증명과 연관되지 않은 로그인에 대한 토큰을 전달하면 해당 로그인이 연관된 자격 증명에 "연결됨"으로 간주됩니다. 퍼블릭 공급자당 로그인 하나만 연결할 수 있습니다. 퍼블릭 공급자에 로그인을 둘 이상 연결하려고 시도하면 ResourceConflictException이 발생합니다. 로그인이 기존 자격 증명에 연결만 되어 있는 경우 GetOpenIdToken에서 반환된 자격 증명 ID는 전달된 것과 같습니다.

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 반환됩니다. 이 값이 다를 경우 로컬 캐시를 업데이트해야 합니다(브라우저에서 AWS Mobile SDK 또는 JavaScript용 AWS SDK의 공급자를 사용할 경우 처리됨).

GetOpenIdTokenForDeveloperIdentity

개발자 인증 자격 증명을 사용할 경우 GetOpenIdTokenForDeveloperIdentity API는 디바이스에서 GetId 및 GetOpenIdToken의 사용을 대체합니다. 이 API 호출은 AWS 자격 증명에 의해 서명되므로 Amazon Cognito는 해당 API에서 제공된 사용자 식별자가 유효함을 신뢰할 수 있습니다. 이 호출은 가 외부 공급자를 통해 수행하는 토큰 확인을 대체합니다.

이 API에 대한 페이로드에는 개발자 공급자의 키가 있어야 하는 로그인 맵과 시스템에서 사용자에게 대한 식별자 값이 포함되어 있습니다. 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 경우는 기존 자격 증명 ID와 OpenId Connect 토큰을 반환합니다.

로그인 연결

외부 공급자와 마찬가지로 아직 자격 증명과 연관되지 않은 추가 로그인을 제공하면 이러한 로그인과 해당 자격 증명이 암시적으로 연결됩니다. 외부 공급자 로그인을 자격 증명에 연결하면 사용자는 해당 공급자를 통한 외부 공급자 인증 흐름을 사용할 수 있지만 GetId 또는 GetOpenIdToken을 호출할 때 로그인 맵에 개발자 공급자 이름을 사용할 수 없음을 유의하십시오.

자격 증명 병합

개발자 인증 자격 증명을 사용하면 Amazon Cognito는 MergeDeveloperIdentities API를 통해 암시적 병합과 명시적 병합을 모두 지원합니다. 이러한 명시적 병합을 통해 시스템에서 사용자 식별자가 있는 두 개의 자격 증명을 단일 자격 증명으로 표시할 수 있습니다. 원본과 대상 사용자 식별자를 제공하기만 하면 에서는 이를 병합합니다. 다음에 사용자 식별자에 대한 OpenId Connect token을 요청할 때 동일한 자격 증명 ID가 반환됩니다.

AssumeRoleWithWebIdentity

OpenId Connect 토큰이 있으면 AWS STS(Security Token Service)의 AssumeRoleWithWebIdentity API 호출을 통해 이 토큰을 주고 임시 AWS 자격 증명을 얻을 수 있습니다. 다른 퍼블릭 공급자 중 하나에서 토큰 대신 Amazon Cognito 토큰을 전달하는 점을 제외하면 이 호출은 Facebook, Google+ 또는 Login with Amazon을 직접 사용하는 경우와 다르지 않습니다.

생성할 수 있는 자격 증명의 수에 대한 제한이 없으므로 사용자에게 부여되는 권한을 이해하는 것이 중요합니다. 애플리케이션에 대해 미인증 사용자에게 대한 역할과 인증 사용자에게 대한 역할 두 개가 있는 것이 좋습니다. Amazon Cognito 콘솔은 처음으로 자격 증명 풀을 설정할 때 기본적으로 이러한 역할을 생성합니다. 이러한 두 역할에 대한 액세스 정책은 정확하게 동일합니다. 해당 정책은 사용자에게 Amazon Cognito Sync에 대한 액세스 권한을 부여하고 이벤트를 Amazon Mobile Analytics에 제출합니다. 이러한 역할을 사용자의 요구에 맞게 수정할 수 있습니다.

[역할 신뢰 및 권한 \(p. 238\)](#) 단원에 대해 자세히 알아봅니다.

IAM 역할

자격 증명 풀 생성 중에 사용자가 담당할 IAM 역할을 업데이트하라는 메시지가 표시됩니다. 사용자가 앱에 로그인할 때 Amazon Cognito는 사용자에게 대해 임시 AWS 자격 증명을 생성합니다. IAM 역할은 이런 식으로 작동합니다. 이러한 임시 자격 증명은 특정 IAM 역할에 연결됩니다. IAM 역할을 통해 AWS 리소스에 액세스할 수 있는 일련의 권한을 정의할 수 있습니다.

인증 및 미인증 사용자에게 대해 기본 IAM 역할을 지정할 수 있습니다. 또한 사용자의 ID 토큰에 있는 클레임에 따라 각 사용자에게 대한 역할을 선택하는 규칙을 정의할 수 있습니다. 자세한 내용은 [역할 기반 액세스 제어 \(p. 238\)](#) 단원을 참조하십시오.

기본적으로 Amazon Cognito 콘솔은 Amazon Mobile Analytics 및 Amazon Cognito Sync에 액세스할 수 있는 IAM 역할을 생성합니다. 또는 기존 IAM 역할을 사용하도록 선택할 수 있습니다.

다른 서비스에 대한 액세스를 허용하거나 제한하도록 IAM 역할을 수정하려면 [IAM 콘솔에 로그인](#)합니다. 그런 다음 [Roles]를 클릭하고 역할을 선택합니다. 선택한 역할에 연결된 정책은 [Permissions] 탭에 나열됩니다. 해당하는 [Manage Policy] 링크를 클릭하여 액세스 정책을 사용자 지정할 수 있습니다. 정책 사용 및 정의에 대한 자세한 내용은 [IAM 정책 개요](#)를 참조하십시오.

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 설명서의 [최소 권한 부여](#)를 참조하십시오.

인증되지 않은 자격 증명은 앱에 로그인하지 않은 사용자에게 의해 수임된다는 점을 유의하십시오. 일반적으로, 인증되지 않은 자격 증명에 할당한 권한은 인증된 자격 증명의 권한보다 더 제한적이어야 합니다.

Amazon Cognito가 작동하도록 하려면 다음 예제와 같이 IAM 정책이 최소한 각 자격 증명의 Amazon Cognito 스토어에 대한 액세스를 활성화해야 합니다.

```
{
  "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}/*" ]
  }]
}
```

다음 정책은 전체 Amazon Cognito Sync 스토어에 액세스할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "cognito-sync:*",
    "Resource": [ "arn:aws:cognito-sync:us-east-1:123456789012:identitypool/*" ]
  }]
}
```

역할 신뢰 및 권한

Amazon Cognito는 IAM 역할을 이용하여 애플리케이션의 사용자에게 대한 임시 자격 증명을 생성합니다. 권한에 대한 액세스는 역할의 신뢰 관계에 의해 제어됩니다. [역할 신뢰 및 권한 \(p. 238\)](#) 단원에 대해 자세히 알아보십시오.

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 증명에 대한 액세스 제한

특정 앱 사용자 집합에 제한된 정책을 생성하려면 `cognito-identity.amazonaws.com:sub` 값을 선택하십시오.

```
"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 공급자에 대한 액세스 제한

특정 공급자(고유한 로그인 공급자)를 통해 로그인한 사용자에게 제한된 정책을 생성하려면 `cognito-identity.amazonaws.com:amr` 값을 선택하십시오.

```
"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 연결된 권한은 해당 역할을 담당하는 모든 사용자에게 적용됩니다. 사용자의 액세스를 파티셔닝하려는 경우 정책 변수를 통해 이를 수행할 수 있습니다. 특히 미인증 자격 증명에 대해 액세스 정책에 사용자의 자격 증명 ID를 포함시킬 경우 사용자가 로그인하도록 선택하면 이러한 사항이 변경될 수 있음을 유의하십시오.

추가적인 보안을 위해 Amazon Cognito는 `GetCredentialForIdentity`가 제공한 자격 증명에 범위가 축소된 정책을 적용하여 미인증 사용자에게 대해 다음 이외의 서비스에 대한 액세스를 금지합니다.

- AWS AppSync
- CloudWatch
- Amazon Cognito 자격 증명
- Amazon Cognito Sync
- Amazon Comprehend
- DynamoDB
- Amazon Kinesis Data Firehose
- Amazon Kinesis Data Streams
- GameLift
- AWS IoT
- AWS KMS
- AWS Lambda
- Amazon Lex
- Amazon Machine Learning
- Amazon Mobile Analytics
- Amazon Pinpoint
- Amazon Polly
- Amazon Rekognition
- Amazon Sumerian
- Amazon S3
- Amazon SimpleDB
- Amazon SES
- Amazon SNS
- Amazon SQS
- Amazon Transcribe

- Amazon Translate

미인증 사용자에게 대해 이러한 서비스 이외의 다른 사항에 액세스해야 할 경우 기본 인증 흐름을 사용해야 합니다. 이는 `NotAuthorizedException`을 가져오고 미인증 역할 정책에서 서비스에 대한 액세스를 활성화 했기 때문일 수 있습니다.

S3Prefix

`${cognito-identity.amazonaws.com:sub}` 변수에 접두사를 매핑하여 S3 버킷에서 사용자에게 특정 접두사 "폴더"를 제공할 수 있습니다.

```
{
  "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}/*"]
    }
  ]
}
```

Amazon DynamoDB에 대한 세분화된 액세스

Amazon Cognito 변수를 사용하여 Amazon DynamoDB 리소스에 대해 세분화된 액세스 제어를 제공할 수 있습니다. 자격 증명 ID를 기준으로 DynamoDB의 항목에 액세스를 부여하면 됩니다.

```
{
  "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}"]
        }
      }
    }
  ]
}
```

역할 신뢰 및 권한

이러한 역할이 서로 다른 방식은 신뢰 관계에 있습니다. 미인증 역할에 대한 예제 신뢰 정책을 살펴보겠습니다.

```
{
  "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-dead-beef-cafe-123456790ab"
        },
        "ForAnyValue:StringLike": {
          "cognito-identity.amazonaws.com:amr": "unauthenticated"
        }
      }
    }
  ]
}
```

이 정책은 `cognito-identity.amazonaws.com`의 연동 사용자(OpenID Connect 토큰의 발급자)가 이 역할을 담당할 수 있음을 허용합니다. 또한 토큰의 대상(이 경우 자격 증명 풀 ID)이 자격 증명 풀과 일치한다는 제한을 설정합니다. 마지막으로 토큰의 `amr`에 미인증된 값이 있다고 지정합니다.

Amazon Cognito가 토큰을 생성하면 토큰의 `amr`을 "미인증" 또는 "인증"으로 설정합니다. 인증의 경우 인증 중 사용된 공급자가 포함됩니다. 즉, `amr` 절을 다음과 같이 변경하면 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 기울이십시오. 역할이 자격 증명 풀을 신뢰하도록 올바르게 구성되지 않은 경우 다음과 같이 STS의 예외가 표시됩니다.

```
AccessDenied -- Not authorized to perform sts:AssumeRoleWithWebIdentity
```

이와 같은 사항이 표시되면 자격 증명 풀과 인증 유형에 대해 적절한 역할을 사용하고 있는지 다시 점검하십시오.

역할 기반 액세스 제어

Amazon Cognito 자격 증명 풀은 인증된 사용자가 AWS 리소스에 액세스할 수 있는 권한이 제한된 임시 자격 증명을 할당합니다. 각 사용자의 권한은 생성된 **IAM 역할**을 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 역할을 정의할 수도 있습니다.

역할 매핑을 위한 역할 만들기

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-dead-beef-
cafe-123456790ab"
        },
        "ForAnyValue:StringLike": {
          "cognito-identity.amazonaws.com:amr": "authenticated"
        }
      }
    }
  ]
}
```

이 정책에 의해 `cognito-identity.amazonaws.com`의 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 이 정책은 토큰의 `amr`에 `authenticated` 값이 포함되도록 지정합니다.

역할 전달 권한 부여

IAM 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 부여해야 합니다. 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 ARN을 사용하여 지정됩니다. 또한 이 정책을 사용자 또는 사용자가 속한 역할에 연결해야 합니다. 자세한 내용은 [관리형 정책 작업](#) 섹션을 참조하십시오.

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 사용](#) 단원을 참조하십시오.

토큰을 사용하여 사용자에게 역할 할당

Amazon Cognito 사용자 풀을 통해 로그인 하는 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(연동 자격 증명)은 다음과 같이 역할을 선택합니다.

- `GetCredentialsForIdentity` CustomRoleArn 파라미터가 설정되어있고 `cognito:roles` 클레임의 역할과 일치할 경우 이 파라미터를 사용합니다. 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 설정되며 `GetCredentialsForIdentity`에 대한 호출에 CustomRoleArn이 지정되지 않은 경우 콘솔의 역할 해결 설정 또는 AmbiguousRoleResolution 필드(`SetIdentityPoolRoles` API의 RoleMappings 파라미터에 있음)를 사용하여 할당할 역할을 결정합니다.

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 정합니다. 이때 일치 유형은 Equals, NotEqual, StartsWith 또는 Contains가 될 수 있습니다. 사용자가 클레임에 대해 일치하는 값을 가진 경우 자격 증명을 얻으면 해당 역할을 수임할 수 있습니다. 예를 들어, Sales의 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 CustomRoleArn이 순서를 무시하도록 지정하지 않으면 첫 번째 일치 규칙의 IAM 역할이 사용됩니다. Amazon Cognito 사용자 풀의 사용자 속성에 대한 자세한 내용은 [사용자 풀 속성 구성 \(p. 202\)](#) 단원을 참조하십시오.

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 NotEqual이고 클레임이 없으면 규칙이 평가되지 않습니다. 일치하는 규칙이 없으면 역할 해결 설정이 기본 인증된 역할 사용 또는 거부로 적용됩니다.

API 및 CLI에서 [SetIdentityPoolRoles](#) API의 `RoleMappings` 파라미터에 지정된 [RoleMapping](#)의 `AmbiguousRoleResolution` 필드에 일치하는 규칙이 없을 경우 할당할 역할을 지정할 수 있습니다.

자격 증명 풀에 구성된 각 사용자 풀 또는 기타 인증 공급자에 대해 규칙을 25개까지 만들 수 있습니다. 공급자를 위한 규칙이 25개 이상 필요한 경우 [Service Limit Increase](#) 지원 사례를 여십시오.

규칙 기반 매핑에 사용할 토큰 클레임

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 사양](#)을 참조하십시오. 다음은 표준 클레임 외에 와 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 사양](#)의 표준 클레임이 포함됩니다. OpenID 토큰의 모든 클레임은 규칙 기반 매핑에 사용할 수 있습니다. Google 토큰에서 사용할 수 있는 클레임에 대한 자세한 내용은 [OpenID Connect](#)를 참조하십시오.

OpenID

Open ID 토큰의 모든 클레임은 규칙 기반 매핑에 사용할 수 있습니다. 표준 클레임에 대한 자세한 내용은 [OpenID Connect 사양](#)을 참조하십시오. 사용 가능한 추가 클레임에 대한 내용은 OpenID 공급자 설명서를 참조하십시오.

SAML

수신된 SAML 어설션에서 클레임을 구문 분석합니다. SAML 어설션에서 사용할 수 있는 모든 클레임은 규칙 기반 매핑에서 사용할 수 있습니다.

역할 기반 액세스 제어를 위한 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 역

할에만 매핑하십시오. Amazon Cognito 사용자 풀에서 각 사용자 속성에 대해 애플리케이션이 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 해결 설정 및 [SetIdentityPoolRoles](#) API의 `RoleMappings` 파라미터를 사용하여 토큰에서 올바른 역할을 결정할 수 없을 때의 기본 동작을 지정할 수 있습니다.

자격 증명 받기

Amazon Cognito를 사용하여 애플리케이션에 권한이 제한된 임시 자격 증명을 전달하면 사용자가 AWS 리소스에 액세스할 수 있습니다. 이 단원에서는 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 확인하십시오.

Amazon Cognito ID는 자격 증명이 아닙니다. AWS Security Token Service(AWS STS)에서 웹 자격 증명 연동 지원을 사용하여 자격 증명으로 교환되는 것입니다. 앱 사용자를 위해 AWS 자격 증명을 얻는 권장 방법은 `AWS.CognitoIdentityCredentials`를 사용하는 것입니다. 그러면 자격 증명 객체의 ID가 AWS STS를 사용하여 자격 증명용으로 교환됩니다.

Note

2015년 2월 이전에 자격 증명 풀을 만든 경우 파라미터 역할 없이 `AWS.CognitoIdentityCredentials` 생성자를 사용하려면 역할을 작업 증명 풀과 다시 연결해야 합니다. 이렇게 하려면 [Amazon Cognito 콘솔](#)을 열고, 연동 자격 증명 관리를 선택하고, 자격 증명 풀을 선택합니다. 자격 증명 풀 편집을 선택하고, 인증된 역할과 인증되지 않은 역할을 지정하고 변경 사항을 저장하십시오.

Android

Amazon Cognito를 사용하여 애플리케이션에 권한이 제한된 임시 자격 증명을 전달하면 사용자가 AWS 리소스에 액세스할 수 있습니다. Amazon Cognito는 인증된 자격 증명과 인증되지 않은 자격 증명을 모두 지원합니다. 앱에 AWS 자격 증명을 제공하려면 아래의 단계를 수행하십시오.

1. [Amazon Cognito 콘솔](#)에서 연동 자격 증명 관리를 선택하여 자격 증명 풀을 생성하고 시작 코드 조각을 복사합니다.
2. 아직 Android용 AWS Mobile SDK를 프로젝트에 추가하지 않았다면 지금 추가합니다. 자세한 내용은 [Android용 Mobile SDK 설정](#)을 참조하십시오.
3. 다음 import 문을 포함합니다.

```
import com.amazonaws.auth.CognitoCachingCredentialsProvider;
import com.amazonaws.regions.Regions;
```

4. Amazon Cognito 콘솔에서 생성된 코드 조각을 사용하여 Amazon Cognito 자격 증명 공급자를 초기화합니다. `IDENTITY_POOL_ID` 값은 계정에 따라 다릅니다.

```
CognitoCachingCredentialsProvider credentialsProvider = new
CognitoCachingCredentialsProvider(
```

```
getApplicationContext(), // Context
"IDENTITY_POOL_ID", // Identity Pool ID
Regions.US_EAST_1 // Region
);
```

5. 초기화된 Amazon Cognito 자격 증명 공급자를 사용할 AWS 클라이언트 생성자에 전달합니다. 필요한 코드는 초기화할 서비스에 따라 다릅니다. 클라이언트가 이 공급자를 사용하여 AWS 리소스에 액세스할 수 있는 자격 증명을 가져옵니다.

Note

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 콘솔](#)을 열고, 연동 자격 증명 관리를 선택하고, 자격 증명 풀을 선택합니다. 그리고 자격 증명 풀 편집을 선택하고, 인증된 역할과 인증되지 않은 역할을 지정하고 변경 사항을 저장하십시오.

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 마십시오. Android 3.0(API 레벨 11)을 기준으로 기본 애플리케이션 스레드에서 네트워크 I/O를 수행하는 경우 앱이 자동으로 실패하고 [NetworkOnMainThreadException](#)이 발생합니다. `AsyncTask`를 사용하여 코드를 백그라운드 스레드로 이동해야 합니다. 자세한 내용은 [Android 설명서](#)를 참조하십시오. 이미 로컬에 캐시된 경우에만 `getCachedIdentityId()`를 호출하여 ID를 검색할 수 있습니다. 그렇지 않으면 메서드가 null을 반환합니다.

iOS - Objective-C

Amazon Cognito를 사용하여 애플리케이션에 권한이 제한된 임시 자격 증명을 전달하면 사용자가 AWS 리소스에 액세스할 수 있습니다. Amazon Cognito 자격 증명 풀은 인증된 자격 증명과 인증되지 않은 자격 증명을 모두 지원합니다. 앱에 AWS 자격 증명을 제공하려면 아래의 단계를 수행하십시오.

1. [Amazon Cognito 콘솔](#)에서 연동 자격 증명 관리를 선택하여 자격 증명 풀을 생성하고 시작 코드 조각을 복사합니다.
2. 아직 AWS Mobile SDK for iOS를 프로젝트에 추가하지 않았다면 지금 추가합니다. 자세한 내용은 [Mobile SDK for iOS 설정](#)을 참조하십시오.
3. 소스 코드에 AWSCore 헤더를 포함합니다.

```
#import <AWSCore/AWSCore.h>
```

4. Amazon Cognito 콘솔에서 생성된 코드 조각을 사용하여 Amazon Cognito 자격 증명 공급자를 초기화합니다. `IDENTITY_POOL_ID` 값은 계정에 따라 다릅니다.

```
AWSCognitoCredentialsProvider *credentialsProvider = [[AWSCognitoCredentialsProvider
alloc]
initWithRegionType:AWSRegionUSEast1 identityPoolId:@"IDENTITY_POOL_ID"];
AWSServiceConfiguration *configuration = [[AWSServiceConfiguration alloc]
initWithRegion:AWSRegionUSEast1 credentialsProvider:credentialsProvider];
```

```
AWSServiceManager.defaultServiceManager.defaultServiceConfiguration = configuration;
```

Note

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 콘솔](#)을 열고, 연동 자격 증명 관리를 선택하고, 자격 증명 풀을 선택합니다. 자격 증명 풀 편집을 선택하고, 인증된 역할과 인증되지 않은 역할을 지정하고 변경 사항을 저장하십시오.

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

getIdentityId는 비동기식 호출입니다. 자격 증명 ID가 이미 공급자에 설정된 경우 credentialsProvider.identityId를 호출하여 해당 자격 증명을 검색할 수 있으며 이 자격 증명은 로컬에 캐시됩니다. 그러나 자격 증명 ID가 공급자에 설정되지 않은 경우 credentialsProvider.identityId를 호출하면 nil이 반환됩니다. 자세한 내용은 [Mobile SDK for iOS API 참조](#)를 참조하십시오.

iOS - Swift

Amazon Cognito를 사용하여 애플리케이션에 권한이 제한된 임시 자격 증명을 전달하면 사용자가 AWS 리소스에 액세스할 수 있습니다. Amazon Cognito는 인증된 자격 증명과 인증되지 않은 자격 증명을 모두 지원합니다. 앱에 AWS 자격 증명을 제공하려면 아래의 단계를 수행하십시오.

1. [Amazon Cognito 콘솔](#)에서 연동 자격 증명 관리를 선택하여 자격 증명 풀을 생성하고 시작 코드 조각을 복사합니다.
2. 아직 Mobile SDK for iOS를 프로젝트에 추가하지 않았다면 지금 추가합니다. 자세한 내용은 [SDK for iOS 설정](#)을 참조하십시오.
3. 소스 코드에 AWSCore 헤더를 포함합니다.

```
import AWSCore
```

4. Amazon Cognito 콘솔에서 생성된 코드 조각을 사용하여 Amazon Cognito 자격 증명 공급자를 초기화합니다. IDENTITY_POOL_ID 값은 계정에 따라 다릅니다.

```
let credentialsProvider = AWSCognitoCredentialsProvider(regionType: .USEast1,
    identityPoolId: "IDENTITY_POOL_ID")
let configuration = AWSServiceConfiguration(region: .USEast1, credentialsProvider:
    credentialsProvider)
AWSServiceManager.default().defaultServiceConfiguration = configuration
```

Note

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 콘솔](#)을 열고, 연동 자격 증명 관리를 선택하고, 자격 증명 풀을 선택합니다. 자격 증명 풀 편집을 선택하고, 인증된 역할과 인증되지 않은 역할을 지정하고 변경 사항을 저장하십시오.

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

getIdentityId는 비동기식 호출입니다. 자격 증명 ID가 이미 공급자에 설정된 경우 credentialsProvider.identityId를 호출하여 해당 자격 증명을 검색할 수 있으며 이 자격 증명은 로컬에 캐시됩니다. 그러나 자격 증명 ID가 공급자에 설정되지 않은 경우 credentialsProvider.identityId를 호출하면 nil이 반환됩니다. 자세한 내용은 [Mobile SDK for iOS API 참조](#)를 참조하십시오.

JavaScript

아직 생성하지 않은 경우 AWS.CognitoIdentityCredentials 사용 전에 [Amazon Cognito 콘솔](#)에서 자격 증명 풀을 생성하십시오.

자격 증명 제공자로 자격 증명 풀을 구성한 후에는 AWS.CognitoIdentityCredentials를 사용하여 사용자를 인증할 수 있습니다. AWS.CognitoIdentityCredentials를 사용하도록 애플리케이션 자격 증명을 구성하려면, credentials 또는 서비스당 구성의 AWS.Config 속성을 설정하십시오. 다음 예에는 AWS.Config가 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'
    }
});

// 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 SDK](#)에서 FB.login 함수를 사용하여 자격 증명 공급자 토큰을 얻는 것입니다.

```
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 애플리케이션에 권한이 제한된 임시 자격 증명을 전달하면 사용자가 AWS 리소스에 액세스할 수 있습니다. Amazon Cognito는 인증된 자격 증명과 인증되지 않은 자격 증명을 모두 지원합니다. 앱에 AWS 자격 증명을 제공하려면 아래의 단계를 수행하십시오.

1. [Amazon Cognito 콘솔](#)에서 연동 자격 증명 관리를 선택하여 자격 증명 풀을 생성하고 시작 코드 조각을 복사합니다.
2. 아직 [AWS Mobile SDK for Unity](#) 패키지를 다운로드하지 않았다면 다운로드 후 프로젝트로 가져옵니다. Assets> Import Package> Custom Package 메뉴에서 수행할 수 있습니다.
3. Amazon Cognito를 호출할 스크립트에 콘솔의 시작 코드 조각을 붙여 넣습니다. IDENTITY_POOL_ID 값은 계정에 따라 다릅니다.

```
CognitoAWSCredentials credentials = new CognitoAWSCredentials (
    "IDENTITY_POOL_ID",    // Cognito Identity Pool ID
    RegionEndpoint.USEast1 // Region
);
```

4. 초기화된 Amazon Cognito 자격 증명을 사용할 AWS 클라이언트 생성자에 전달합니다. 필요한 코드는 초기화할 서비스에 따라 다릅니다. 클라이언트가 이 공급자를 사용하여 AWS 리소스에 액세스할 수 있는 자격 증명을 가져옵니다.

Note

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 콘솔](#)을 열고, 연동 자격 증명 관리를 선택하고, 자격 증명 풀을 선택합니다. 자격 증명 풀 편집을 선택하고, 인증된 역할과 인증되지 않은 역할을 지정하고 변경 사항을 저장하십시오.

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 애플리케이션에 권한이 제한된 임시 자격 증명을 전달하면 사용자가 AWS 리소스에 액세스할 수 있습니다. Amazon Cognito는 인증된 자격 증명과 인증되지 않은 자격 증명을 모두 지원합니다. 앱에 AWS 자격 증명을 제공하려면 아래의 단계를 수행하십시오.

1. [Amazon Cognito 콘솔](#)에서 연동 자격 증명 관리를 선택하여 자격 증명 풀을 생성하고 시작 코드 조각을 복사합니다.
2. 아직 AWS Mobile SDK for Xamarin을 프로젝트에 추가하지 않았다면 지금 추가합니다. 자세한 내용은 [SDK for Xamarin 설정](#) 단원을 참조하십시오.
3. 다음 using 문을 포함합니다.

```
using Amazon.CognitoIdentity;
```

4. Amazon Cognito를 호출할 스크립트에 콘솔의 시작 코드 조각을 붙여 넣습니다. IDENTITY_POOL_ID 값은 계정에 따라 다릅니다.

```
CognitoAWSCredentials credentials = new CognitoAWSCredentials (  
    "IDENTITY_POOL_ID",    // Cognito Identity Pool ID  
    RegionEndpoint.USEast1 // Region  
);
```

5. 초기화된 Amazon Cognito 자격 증명을 사용할 AWS 클라이언트 생성자에 전달합니다. 필요한 코드는 초기화할 서비스에 따라 다릅니다. 클라이언트가 이 공급자를 사용하여 AWS 리소스에 액세스할 수 있는 자격 증명을 가져옵니다.

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 콘솔](#)을 열고, 연동 자격 증명 관리를 선택하고, 자격 증명 풀을 선택합니다. 자격 증명 풀 편집을 선택하고, 인증된 역할과 인증되지 않은 역할을 지정하고 변경 사항을 저장하십시오.

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();
```

AWS 제품 액세스

Amazon Cognito 자격 증명 공급자를 초기화하고 새로 고치면 AWS 클라이언트의 이니셜라이저에 직접 전달할 수 있습니다. 예를 들어, 다음 조각은 클라이언트를 초기화합니다.

Android

```
// Create a service client with the provider
AmazonDynamoDB client = new AmazonDynamoDBClient(credentialsProvider);
```

자격 증명 공급자는 Amazon Cognito와 통신하여 인증된 사용자와 인증되지 않은 사용자의 고유 식별자는 물론 AWS Mobile SDK의 권한이 제한된 임시 SDK 자격 증명을 검색합니다. 검색된 자격 증명은 한시간 동안 유효하며 만료된 경우 공급자는 자격 증명을 새로 고칩니다.

iOS - Objective-C

```
// 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 사용자와 인증되지 않은 사용자의 고유 식별자는 물론 AWS Mobile SDK의 권한이 제한된 임시 SDK 자격 증명을 검색합니다. 검색된 자격 증명은 한시간 동안 유효하며 만료된 경우 공급자는 자격 증명을 새로 고칩니다.

iOS - Swift

```
// 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 사용자와 인증되지 않은 사용자의 고유 식별자는 물론 AWS Mobile SDK의 권한이 제한된 임시 SDK 자격 증명을 검색합니다. 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 사용자와 인증되지 않은 사용자의 고유 식별자는 물론 AWS Mobile SDK의 권한이 제한된 임시 SDK 자격 증명을 검색합니다. 검색된 자격 증명은 한시간 동안 유효하며 만료된 경우 공급자는 자격 증명을 새로 고칩니다.

Unity

```
// create a service client that uses credentials provided by Cognito
AmazonDynamoDBClient client = new AmazonDynamoDBClient(credentials, REGION);
```

자격 증명 공급자는 Amazon Cognito와 통신하여 인증된 사용자와 인증되지 않은 사용자의 고유 식별자는 물론 AWS Mobile SDK의 권한이 제한된 임시 SDK 자격 증명을 검색합니다. 검색된 자격 증명은 한시간 동안 유효하며 만료된 경우 공급자는 자격 증명을 새로 고칩니다.

Xamarin

```
// create a service client that uses credentials provided by Cognito
var client = new AmazonDynamoDBClient(credentials, REGION)
```

자격 증명 공급자는 Amazon Cognito와 통신하여 인증된 사용자와 인증되지 않은 사용자의 고유 식별자는 물론 AWS Mobile SDK의 권한이 제한된 임시 SDK 자격 증명을 검색합니다. 검색된 자격 증명은 한시간 동안 유효하며 만료된 경우 공급자는 자격 증명을 새로 고칩니다.

자격 증명 풀(연동 자격 증명) 외부 자격 증명 공급자

logins 속성을 사용하여 자격 증명 공급자에게서 받은 자격 증명을 설정할 수 있습니다. 또한 자격 증명 풀을 여러 자격 증명 공급자와 연결할 수 있습니다. 예를 들어, logins 속성에 Facebook 토큰과 Google 토큰을 둘 다 설정할 수 있어 고유한 Amazon Cognito 자격 증명이 두 자격 증명 공급자 로그인과 연결됩니다. 최종 사용자가 인증에 어느 계정을 사용해도 에서 같은 사용자 식별자를 반환합니다.

아래 지침은 Amazon Cognito 자격 증명 풀에서 지원하는 자격 증명 공급자를 통한 인증을 안내합니다.

주제

- [Facebook\(자격 증명 풀\) \(p. 249\)](#)
- [Login with Amazon\(자격 증명 풀\) \(p. 254\)](#)
- [Google\(자격 증명 풀\) \(p. 257\)](#)
- [OpenID Connect 공급자\(자격 증명 풀\) \(p. 263\)](#)
- [SAML 자격 증명 공급자\(자격 증명 풀\) \(p. 265\)](#)

Facebook(자격 증명 풀)

Amazon Cognito 자격 증명 풀은 Facebook과 통합되어 모바일 애플리케이션 사용자를 위해 연동된 인증을 제공합니다. 이 단원에서는 Facebook을 자격 증명 공급자로 사용하여 애플리케이션을 등록하고 설정하는 방법을 설명합니다.

Facebook 설정

Facebook 사용자 인증 및 Facebook API와의 상호 작용을 시작하려면 먼저 Facebook을 통해 애플리케이션을 등록해야 합니다.

[Facebook 개발자 포털](#)에서 애플리케이션 설정 프로세스에 연결됩니다. 이 프로세스를 완료해야 Identity Pool에서 Facebook을 통합할 수 있습니다.

Facebook을 설정하려면

1. [Facebook 개발자 포털](#)에서 Facebook 자격 증명으로 로그인합니다.

2. 앱 메뉴에서 새 앱 추가를 선택합니다.
3. 플랫폼을 선택하고 빠른 시작 프로세스를 완료합니다.

Android

Facebook 로그인 통합에 대한 자세한 내용이 [Facebook 시작 안내서](#)에 나와 있습니다.

iOS - Objective-C

Facebook 로그인 통합에 대한 자세한 내용이 [Facebook 시작 안내서](#)에 나와 있습니다.

iOS - Swift

Facebook 로그인 통합에 대한 자세한 내용이 [Facebook 시작 안내서](#)에 나와 있습니다.

JavaScript

Facebook 로그인 통합에 대한 자세한 내용이 [Facebook 시작 안내서](#)에 나와 있습니다.

Unity

Facebook 로그인 통합에 대한 자세한 내용이 [Facebook 시작 안내서](#)에 나와 있습니다.

Xamarin

Facebook 인증을 제공하려면 먼저 아래의 적합한 흐름에 따라 애플리케이션에 Facebook SDK를 포함하고 설정합니다. Amazon Cognito 자격 증명 풀은 Facebook 액세스 토큰을 사용하여 Cognito 자격 증명에 연결된 고유한 식별자를 생성합니다.

- [Xamarin이 제공하는 Facebook iOS SDK](#)
- [Xamarin이 제공하는 Facebook Android SDK](#)

Amazon Cognito 연동 자격 증명 콘솔에서 외부 공급자 구성

[Amazon Cognito 콘솔](#) 홈 페이지에서 연동 자격 증명 관리를 선택합니다.

1. 연동 자격 증명 관리를 선택합니다.
2. Facebook을 외부 공급자로 허용할 자격 증명 풀의 이름을 선택합니다. 자격 증명 풀에 대한 대시보드 페이지가 표시됩니다.
3. 대시보드 페이지의 우측 상단 모서리에서 자격 증명 풀 편집을 선택합니다. 자격 증명 풀 편집 페이지가 표시됩니다.
4. 아래로 스크롤하고 인증 공급자를 선택하여 확장합니다.
5. Facebook 탭을 선택합니다.
6. 잠금 해제를 선택합니다.
7. Facebook에서 가져온 Facebook 앱 ID를 입력하고 변경 사항 저장을 선택합니다.

Facebook 사용

Android

Facebook 인증을 제공하려면 먼저 [Facebook 안내서](#)에 따라 해당 애플리케이션에 SDK를 포함하십시오. 그런 다음 ["Facebook으로 로그인" 버튼](#)을 Android 사용자 인터페이스에 추가합니다. Facebook SDK에서 세션

객체를 사용하여 그 상태를 추적합니다. Amazon Cognito는 이 세션 객체에서 받은 액세스 토큰을 사용하여 사용자를 인증하고 고유한 식별자를 생성하며 필요할 경우 사용자에게 다른 AWS 리소스에 대한 액세스 권한을 허용합니다.

Facebook SDK로 사용자를 인증한 후에는 Amazon Cognito 자격 증명 공급자에 세션 토큰을 추가하십시오.

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 singleton 세션을 초기화합니다. Facebook 세션 객체에는 인증된 최종 사용자의 AWS 자격 증명을 생성하기 위해 Amazon Cognito에 사용되는 OAuth 토큰이 포함되어 있습니다. Amazon Cognito에서는 토큰을 사용하여 이 특정한 Facebook 자격 증명과 일치하는 사용자가 있는지 사용자 데이터베이스를 확인하기도 합니다. 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 실제로 AWS 자격 증명을 가져와야 합니다.

iOS - Objective-C

Facebook 인증을 추가하려면 먼저 [Facebook 안내서](#)를 따라 Facebook SDK를 애플리케이션에 통합하십시오. 그런 다음 [Facebook으로 로그인 버튼](#)을 귀하의 사용자 인터페이스에 추가합니다. Facebook SDK에서 세션 객체를 사용하여 그 상태를 추적합니다. Amazon Cognito는 이 세션 객체에서 받은 액세스 토큰을 사용하여 사용자를 인증하고 고유한 Amazon Cognito 자격 증명 풀(연동 자격 증명)에 바인딩합니다.

Amazon Cognito에 Facebook 액세스 토큰을 제공하려면 [AWSIdentityProviderManager](#) 프로토콜을 구현하십시오.

`logins` 메서드를 구현할 때 다음 코드 예제와 같이 `AWSIdentityProviderFacebook`이 키로, 인증된 Facebook 사용자에게서 받은 현재 액세스 토큰이 값으로 포함된 딕셔너리를 반환하십시오.

```
- (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 클래스

스를 전달하십시오. 자세한 내용을 보려면 [AWSCognitoCredentialsProvider](#) 참조 페이지에서 `initWithRegionType:identityPoolId:identityProviderManager`를 선택하십시오.

iOS - Swift

Facebook 인증을 추가하려면 먼저 [Facebook 안내서](#)를 따라 Facebook SDK를 애플리케이션에 통합하십시오. 그런 다음 [Facebook으로 로그인 버튼](#)을 귀하의 사용자 인터페이스에 추가합니다. Facebook SDK에서 세션 객체를 사용하여 그 상태를 추적합니다. Amazon Cognito는 이 세션 객체에서 받은 액세스 토큰을 사용하여 사용자를 인증하고 고유한 Amazon Cognito 자격 증명 풀(연동 자격 증명)에 바인딩합니다.

Amazon Cognito에 Facebook 액세스 토큰을 제공하려면 [AWSIdentityProviderManager](#) 프로토콜을 구현하십시오.

logins 메서드를 구현할 때 다음 코드 예제와 같이 `AWSIdentityProviderFacebook`이 키로, 인증된 Facebook 사용자에게서 받은 현재 액세스 토큰이 값으로 포함된 딕셔너리를 반환하십시오.

```
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 [AWSCognitoCredentialsProvider](#) 참조 페이지에서 `initWithRegionType:identityPoolId:identityProviderManager`를 선택하십시오.

JavaScript

Facebook 인증을 제공하려면 [웹용 Facebook 로그인](#)에 따라 웹 사이트에 "Facebook으로 로그인" 버튼을 추가하십시오. Facebook SDK에서 세션 객체를 사용하여 그 상태를 추적합니다. Amazon Cognito는 이 세션 객체에서 받은 액세스 토큰을 사용하여 사용자를 인증하고 고유한 식별자를 생성하며 필요할 경우 사용자에게 다른 AWS 리소스에 대한 액세스 권한을 허용합니다.

Facebook SDK로 사용자를 인증한 후에는 Amazon Cognito 자격 증명 공급자에 세션 토큰을 추가하십시오.

```
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 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 인증된 최종 사용자의 AWS 자격 증명을 생성하기 위해 Amazon Cognito에 사용되는 OAuth 토큰을 가져옵니다. Amazon Cognito에서는 토큰을 사용하여 이 특정한 Facebook 자격 증명과 일치하는 사용자가 있는지 사용자 데이터베이스를 확인하기도 합니다. 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 AWS 자격 증명을 가져와야 합니다. 코드 예제는 [JavaScript README 파일](#)의 "Use Case 17, Integrating User Pools with Cognito Identity"를 참조하십시오.

Unity

Facebook 인증을 제공하려면 먼저 [Facebook 안내서](#)에 따라 애플리케이션에 해당 SDK를 포함하고 설정하십시오. Amazon Cognito는 'FB' 객체에서 받은 Facebook 액세스 토큰을 사용하여 Cognito 자격 증명에 연결된 고유한 사용자 식별자를 생성합니다.

Facebook SDK로 사용자를 인증한 후에는 Amazon Cognito 자격 증명 공급자에 세션 토큰을 추가하십시오.

```
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);  
}
```

반드시 `FB.Login()`을 호출해야 합니다. `FB.IsLoggedIn`을 사용하기 전에는 `FB.AccessToken`이 `true`입니다.

Xamarin

Android용 Xamarin:

```
public void InitializeFacebook() {
    FacebookSdk.SdkInitialize(this.ApplicationContext);
    callbackManager = CallbackManagerFactory.Create();
    LoginManager.Instance.RegisterCallback(callbackManager, new FacebookCallback <LoginResult> () {
        HandleSuccess = loginResult = <LoginResult> {
            var accessToken = loginResult.AccessToken;
            credentials.AddLogin("graph.facebook.com", accessToken.Token);
            //open new activity
        },
        HandleCancel = () = <LoginResult> {
            //throw error message
        },
        HandleError = loginError = <LoginError> {
            //throw error message
        }
    });
    LoginManager.Instance.LogInWithReadPermissions(this, new List <string> {
        "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
        }
    });
}
```

Login with Amazon(자격 증명 풀)

Amazon Cognito는 Login with Amazon과 통합되어 모바일 애플리케이션 및 웹 애플리케이션 사용자를 위해 연동된 인증을 제공합니다. 이 단원에서는 Login with Amazon을 자격 증명 공급자로 사용하여 애플리케이션을 등록하고 설정하는 방법을 설명합니다.

Amazon Cognito와 연동하도록 Login with Amazon을 설정하는 두 가지 방법이 있습니다. 어떤 방법을 사용할지 잘 모르거나 둘 다 사용해야 할 경우 [Login with Amazon FAQ](#)의 "Setting Up Login with Amazon"을 참조하십시오.

- [Amazon 개발자 포털](#)을 통해. 최종 사용자가 Login with Amazon으로 인증하도록 허용하려는 경우 Seller Central 계정이 없으면 이 방법을 사용하십시오.
- <http://login.amazon.com/>에서 Seller Central을 통해. Seller Central을 사용하는 소매상은 이 방법을 사용합니다.

Note

Xamarin의 경우 [Xamarin 시작 안내서](#)에 따라 Login with Amazon을 Xamarin 애플리케이션과 통합하십시오.

Note

Unity 플랫폼에서는 Login with Amazon 통합이 기본적으로 지원되지 않습니다. 현재로서는 웹 보기에서 브라우저 로그인 흐름을 통해 통합해야 합니다.

Login with Amazon 설정

Login with Amazon을 구현하려면 다음 중 하나를 수행하십시오.

- [Amazon 개발자 포털](#)에서 애플리케이션의 보안 프로파일 ID를 만듭니다. 최종 사용자가 Amazon으로 인증하도록 허용하려는 경우 Seller Central 계정이 없으면 이 방법을 사용하십시오. 개발자 포털 [Login with Amazon](#) 설명서에서는 애플리케이션에서 Login with Amazon을 설정하고 클라이언트 SDK를 다운로드하며 Amazon 개발자 플랫폼에서 애플리케이션을 선언하는 프로세스를 안내합니다. [자격 증명 받기](#)에서 설명한 대로 Amazon Cognito 자격 증명 풀을 만들 때 보안 프로파일 ID를 Amazon 앱 ID로 입력해야 하므로 보안 프로파일 ID를 기록해 두십시오.
- <http://login.amazon.com/>에서 Seller Central을 통해 애플리케이션의 애플리케이션 ID를 만듭니다. Seller Central을 사용하는 소매상은 이 방법을 사용하십시오. Seller Central [Login with Amazon](#) 설명서에서는 애플리케이션에서 Login with Amazon을 설정하고 클라이언트 SDK를 다운로드하며 Amazon 개발자 플랫폼에서 애플리케이션을 선언하는 프로세스를 안내합니다. [자격 증명 받기](#)에서 설명한 대로 Amazon Cognito 자격 증명 풀을 만들 때 애플리케이션 ID를 Amazon 앱 ID로 입력해야 하므로 애플리케이션 ID를 기록해 두십시오.

Amazon Cognito 콘솔에서 외부 공급자 구성

[Amazon Cognito 콘솔](#) 홈 페이지에서 연동 자격 증명 관리를 선택합니다.

1. Login with Amazon을 외부 공급자로 허용할 자격 증명 풀의 이름을 클릭합니다. 자격 증명 풀에 대한 대시보드 페이지가 표시됩니다.
2. 대시보드 페이지의 우측 상단 모서리에서 자격 증명 풀 편집을 클릭합니다. [Edit identity pool] 페이지가 표시됩니다.
3. 아래로 스크롤하고 인증 공급자를 클릭하여 확장합니다.
4. Amazon 탭을 선택합니다.
5. 잠금 해제를 선택합니다.
6. Amazon에서 가져온 Amazon 앱 ID를 입력하고 변경 사항 저장을 선택합니다.

Login with Amazon 사용: Android

Amazon 로그인을 구현한 후 TokenListener 인터페이스의 onSuccess 메서드에서 Amazon Cognito 자격 증명 공급자에 토큰을 전달할 수 있습니다. 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 구현한 후 AMZNAccessTokenDelegate의 requestDidSucceed 메서드에서 Amazon Cognito 자격 증명 공급자에 토큰을 전달할 수 있습니다.

```
- (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 구현한 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]
    }
}
```

Login with Amazon 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 전달하십시오.

```
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.cs에 다음 내용을 삽입하십시오.

```
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;
}
```

ViewController.cs에서 다음을 수행하십시오.

```
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 : IIAAuthenticationDelegate
{
    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());
    }
}
```

Google(자격 증명 풀)

Amazon Cognito는 Google과 통합되어 모바일 애플리케이션 사용자를 위해 연동된 인증을 제공합니다. 이 단원에서는 Google을 자격 증명 공급자로 사용하여 애플리케이션을 등록하고 설정하는 방법을 설명합니다.

Android

Note

앱에서 Google을 사용하고 이후에 여러 모바일 플랫폼에서 이 앱을 사용하려면 [OpenID Connect 공급자](#)로 구성해 생성된 모든 클라이언트 ID를 추가 대상으로 추가하여 원활한 통합이 이루어 지도록 해야 합니다. Google 크로스 클라이언트 자격 증명 모델에 대한 자세한 내용은 [Cross-client Identity](#)를 참조하십시오.

Google 설정

Android용 Google 로그인을 사용하도록 설정하려면 애플리케이션을 위한 Google 개발자 콘솔 프로젝트를 만들어야 합니다.

1. [Google 개발자 콘솔](#)로 이동하여 새 프로젝트를 만듭니다.
2. API 및 인증 > API > 소셜 API에서 Google API를 사용 설정합니다.
3. API 및 인증 > 자격 증명 > OAuth 동의 화면에서 앱이 개인 데이터에 대한 액세스를 요청할 때 사용자에게 표시되는 대화 상자를 만듭니다.
4. 자격 증명 > 자격 증명 추가에서 Android용 OAuth 2.0 클라이언트 ID를 만듭니다. 개발하려는 각 플랫폼(웹, iOS, Android 등)의 클라이언트 ID가 필요합니다.
5. [자격 증명] > [자격 증명 추가]에서 서비스 계정을 만듭니다. 새 공개/비공개 키가 생성되었다는 알림이 콘솔에 표시됩니다.

Google 개발자 콘솔 사용에 대한 자세한 내용은 [개발자 콘솔에서 프로젝트 관리](#)를 참조하십시오.

Google을 Android 앱에 통합하는 데 대한 자세한 정보는 [Android용 Google 설명서](#)를 참조하십시오.

Amazon Cognito 콘솔에서 외부 공급자 구성

[Amazon Cognito 콘솔 홈 페이지](#)에서 연동 자격 증명 관리를 선택합니다.

1. Google을 외부 공급자로 허용할 자격 증명 풀의 이름을 클릭합니다. 자격 증명 풀에 대한 대시보드 페이지가 표시됩니다.
2. [Dashboard] 페이지의 우측 상단 모서리에서 [Edit identity pool]을 클릭합니다. [Edit identity pool] 페이지가 표시됩니다.
3. 아래로 스크롤하고 [Authentication providers]를 클릭하여 확장합니다.
4. [Google] 탭을 클릭합니다.
5. [Unlock]을 클릭합니다.
6. Google에서 가져온 Google 클라이언트 ID를 입력하고 [Save Changes]를 선택합니다.

Google 사용

애플리케이션에서 Google로 로그인하도록 허용하려면 [Android용 Google 설명서](#)를 따르십시오. 인증에 성공하면 OpenID Connect 인증 토큰을 가져오고 이를 사용하여 사용자를 인증하고 고유 식별자를 생성합니다.

다음 샘플 코드는 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

앱에서 Google을 사용하고 이후에 여러 모바일 플랫폼에서 이 앱을 사용하려면 [OpenID Connect 공급자](#)로 구성해 생성된 모든 클라이언트 ID를 추가 대상 값으로 추가하여 원활한 통합이 이루어지도록 해야 합니다. Google 크로스 클라이언트 자격 증명 모델에 대한 자세한 내용은 [Cross-client Identity](#)를 참조하십시오.

iOS용 Google 로그인을 사용하도록 설정하려면 애플리케이션을 위한 Google 개발자 콘솔 프로젝트를 만들어야 합니다.

Google 설정

1. [Google 개발자 콘솔](#)로 이동하여 새 프로젝트를 만듭니다.
2. API 및 인증 > API > 소셜 API에서 Google API를 사용 설정합니다.
3. API 및 인증 > 자격 증명 > OAuth 등의 화면에서 앱이 개인 데이터에 대한 액세스를 요청할 때 사용자에게 표시되는 대화 상자를 만듭니다.
4. 자격 증명 > 자격 증명 추가에서 iOS용 OAuth 2.0 클라이언트 ID를 만듭니다. 개발하려는 각 플랫폼(웹, iOS, Android 등)의 클라이언트 ID가 필요합니다.
5. [자격 증명] > [자격 증명 추가]에서 서비스 계정을 만듭니다. 새 공개/비공개 키가 생성되었다는 알림이 콘솔에 표시됩니다.

Google 개발자 콘솔 사용에 대한 자세한 내용은 [개발자 콘솔에서 프로젝트 관리](#)를 참조하십시오.

Google을 iOS 앱에 통합하는 데 대한 자세한 정보는 [iOS용 Google 설명서](#)를 참조하십시오.

[Amazon Cognito 콘솔 홈 페이지](#)에서 연동 자격 증명 관리를 선택합니다.

Amazon Cognito 콘솔에서 외부 공급자 구성

1. Google을 외부 공급자로 허용할 자격 증명 풀의 이름을 클릭합니다. 자격 증명 풀에 대한 대시보드 페이지가 표시됩니다.
2. [Dashboard] 페이지의 우측 상단 모서리에서 [Edit identity pool]을 클릭합니다. [Edit identity pool] 페이지가 표시됩니다.
3. 아래로 스크롤하고 [Authentication providers]를 클릭하여 확장합니다.
4. [Google] 탭을 클릭합니다.
5. [Unlock]을 클릭합니다.
6. Google에서 가져온 Google 클라이언트 ID를 입력하고 [Save Changes]를 선택합니다.

Google 사용

애플리케이션에서 Google로 로그인하도록 허용하려면 [iOS용 Google 설명서](#)를 따르십시오. 인증에 성공하면 OpenID Connect 인증 토큰을 가져오고 이를 사용하여 사용자를 인증하고 고유 식별자를 생성합니다.

인증에 성공하면 id_token이 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

앱에서 Google을 사용하고 이후에 여러 모바일 플랫폼에서 이 앱을 사용하려면 [OpenID Connect 공급자](#)로 구성해 생성된 모든 클라이언트 ID를 추가 대상 값으로 추가하여 원활한 통합이 이루어지도록 해야 합니다. Google 크로스 클라이언트 자격 증명 모델에 대한 자세한 내용은 [Cross-client Identity](#)를 참조하십시오.

iOS용 Google 로그인을 사용하도록 설정하려면 애플리케이션을 위한 Google 개발자 콘솔 프로젝트를 만들어야 합니다.

Google 설정

1. [Google 개발자 콘솔](#)로 이동하여 새 프로젝트를 만듭니다.
2. API 및 인증 > API > 소셜 API에서 Google API를 사용 설정합니다.
3. API 및 인증 > 자격 증명 > OAuth 동의 화면에서 앱이 개인 데이터에 대한 액세스를 요청할 때 사용자에게 표시되는 대화 상자를 만듭니다.
4. 자격 증명 > 자격 증명 추가에서 iOS용 OAuth 2.0 클라이언트 ID를 만듭니다. 개발하려는 각 플랫폼(웹, iOS, Android 등)의 클라이언트 ID가 필요합니다.
5. [자격 증명] > [자격 증명 추가]에서 서비스 계정을 만듭니다. 새 공개/비공개 키가 생성되었다는 알림이 콘솔에 표시됩니다.

Google 개발자 콘솔 사용에 대한 자세한 내용은 [개발자 콘솔에서 프로젝트 관리](#)를 참조하십시오.

Google을 iOS 앱에 통합하는 데 대한 자세한 정보는 [iOS용 Google 설명서](#)를 참조하십시오.

[Amazon Cognito 콘솔 홈 페이지](#)에서 연동 자격 증명 관리를 선택합니다.

Amazon Cognito 콘솔에서 외부 공급자 구성

1. Google을 외부 공급자로 허용할 자격 증명 풀의 이름을 클릭합니다. 자격 증명 풀에 대한 대시보드 페이지가 표시됩니다.
2. [Dashboard] 페이지의 우측 상단 모서리에서 [Edit identity pool]을 클릭합니다. [Edit identity pool] 페이지가 표시됩니다.
3. 아래로 스크롤하고 [Authentication providers]를 클릭하여 확장합니다.
4. [Google] 탭을 클릭합니다.
5. [Unlock]을 클릭합니다.
6. Google에서 가져온 Google 클라이언트 ID를 입력하고 [Save Changes]를 선택합니다.

Google 사용

애플리케이션에서 Google로 로그인하도록 허용하려면 [iOS용 Google 설명서](#)를 따르십시오. 인증에 성공하면 OpenID Connect 인증 토큰을 가져오고 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.

```
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

앱에서 Google을 사용하고 이후에 여러 모바일 플랫폼에서 이 앱을 사용하려면 [OpenID Connect 공급자](#)로 구성해 생성된 모든 클라이언트 ID를 추가 대상 값으로 추가하여 원활한 통합이 이루어

지도록 해야 합니다. Google 크로스 클라이언트 자격 증명 모델에 대한 자세한 내용은 [Cross-client Identity](#)를 참조하십시오.

Google 설정

웹 애플리케이션에 Google 로그인을 사용하도록 설정하려면 애플리케이션을 위한 Google 개발자 콘솔 프로젝트를 만들어야 합니다.

1. [Google 개발자 콘솔](#)로 이동하여 새 프로젝트를 만듭니다.
2. API 및 인증 > API > 소셜 API에서 Google API를 사용 설정합니다.
3. API 및 인증 > 자격 증명 > OAuth 동의 화면에서 앱이 개인 데이터에 대한 액세스를 요청할 때 사용자에게 표시되는 대화 상자를 만듭니다.
4. 자격 증명 > 자격 증명 추가에서 웹 애플리케이션을 위한 OAuth 2.0 클라이언트 ID를 만듭니다. 개발하려는 각 플랫폼(웹, iOS, Android 등)의 클라이언트 ID가 필요합니다.
5. [자격 증명] > [자격 증명 추가]에서 서비스 계정을 만듭니다. 새 공개/비공개 키가 생성되었다는 알림이 콘솔에 표시됩니다.

Google 개발자 콘솔 사용에 대한 자세한 내용은 [개발자 콘솔에서 프로젝트 관리](#)를 참조하십시오.

Amazon Cognito 콘솔에서 외부 공급자 구성

[Amazon Cognito 콘솔 홈 페이지](#)에서 연동 자격 증명 관리를 선택합니다.

1. Google을 외부 공급자로 허용할 자격 증명 풀의 이름을 클릭합니다. 자격 증명 풀에 대한 대시보드 페이지가 표시됩니다.
2. [Dashboard] 페이지의 우측 상단 모서리에서 [Edit identity pool]을 클릭합니다. [Edit identity pool] 페이지가 표시됩니다.
3. 아래로 스크롤하고 [Authentication providers]를 클릭하여 확장합니다.
4. [Google] 탭을 클릭합니다.
5. [Unlock]을 클릭합니다.
6. Google에서 가져온 Google 클라이언트 ID를 입력하고 [Save Changes]를 선택합니다.

Google 사용

애플리케이션에서 Google로 로그인하도록 허용하려면 [웹용 Google 설명서](#)를 따르십시오.

인증에 성공하면 `id_token`이 포함된 응답 객체를 가져오고 Amazon Cognito에서 이를 사용하여 사용자를 인증하고 고유 식별자를 생성합니다.

```
function signinCallback(authResult) {
    if (authResult['status']['signed_in']) {

        // Add the Google access token to the 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

웹 애플리케이션에 Google 로그인을 사용하도록 설정하려면 애플리케이션을 위한 Google 개발자 콘솔 프로젝트를 만들어야 합니다.

1. [Google 개발자 콘솔](#)로 이동하여 새 프로젝트를 만듭니다.
2. API 및 인증 > API > 소셜 API에서 Google API를 사용 설정합니다.
3. API 및 인증 > 자격 증명 > OAuth 등의 화면에서 앱이 개인 데이터에 대한 액세스를 요청할 때 사용자에게 표시되는 대화 상자를 만듭니다.
4. Unity의 경우 총 3개의 ID를 만들어야 합니다. 2개는 Android용이고 하나는 iOS용입니다. 자격 증명 > 자격 증명 추가에서 다음을 수행합니다.
 - Android: Android용 OAuth 2.0 클라이언트 ID와 웹 애플리케이션용 OAuth 2.0 클라이언트 ID를 만듭니다.
 - iOS: iOS용 OAuth 2.0 클라이언트 ID를 만듭니다.
5. [자격 증명] > [자격 증명 추가]에서 서비스 계정을 만듭니다. 새 공개/비공개 키가 생성되었다는 알림이 콘솔에 표시됩니다.

IAM 콘솔에서 OpenID 공급자 만들기

1. 그런 다음 IAM 콘솔에서 OpenID 공급자를 만들어야 합니다. OpenID 공급자를 설정하는 방법에 대한 지침은 [OpenID Connect 자격 증명 공급자 사용](#)을 참조하십시오.
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. 공급자를 만든 후 공급자 이름을 입력하고 나머지 클라이언트 ID 2개를 제공하여 대상 2개를 더 추가합니다.

Amazon Cognito 콘솔에서 외부 공급자 구성

[Amazon Cognito 콘솔 홈 페이지](#)에서 연동 자격 증명 관리를 선택합니다.

1. Google을 외부 공급자로 허용할 자격 증명 풀의 이름을 클릭합니다. 자격 증명 풀에 대한 대시보드 페이지가 표시됩니다.
2. [Dashboard] 페이지의 우측 상단 모서리에서 [Edit identity pool]을 클릭합니다. [Edit identity pool] 페이지가 표시됩니다.
3. 아래로 스크롤하고 [Authentication providers]를 클릭하여 확장합니다.
4. [Google] 탭을 클릭합니다.
5. [Unlock]을 클릭합니다.
6. Google에서 가져온 Google 클라이언트 ID를 입력하고 [Save Changes]를 선택합니다.

Unity Google 플러그인 설치

1. [Unity용 Google Play Games 플러그인](#)을 Unity 프로젝트에 추가합니다.
2. Unity의 Windows 메뉴에서 Android 및 iOS 플랫폼을 위한 ID 3개를 사용하여 플러그인을 구성합니다.

Google 사용

다음 샘플 코드는 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

Xamarin 플랫폼에서는 Google 통합이 기본적으로 지원되지 않습니다. 현재로서는 웹 보기에서 브라우저 로그인 흐름을 통해 통합해야 합니다. Google 통합이 그 밖의 SDK와 연동하는 방법을 알아 보려면 다른 플랫폼을 선택하십시오.

애플리케이션에서 Google로 로그인하도록 허용하려면 사용자를 인증하고 사용자에게서 OpenID Connect 토큰을 받아야 합니다. Amazon Cognito는 이 토큰을 사용하여 Cognito 자격 증명에 연결된 고유한 사용자 식별자를 생성합니다. 그러나 Xamarin용 Google SDK에서는 OpenID Connect 토큰 가져오기를 지원하지 않으므로 다른 클라이언트나 웹 보기의 웹 흐름을 사용해야 합니다.

토큰을 받으면 CognitoAWSCredentials에서 토큰을 설정할 수 있습니다.

```
credentials.AddLogin("accounts.google.com", token);
```

Note

앱에서 Google을 사용하고 이후에 여러 모바일 플랫폼에서 이 앱을 사용하려면 [OpenID Connect 공급자](#)로 구성해 생성된 모든 클라이언트 ID를 추가 대상 값으로 추가하여 원활한 통합이 이루어 지도록 해야 합니다. Google 크로스 클라이언트 자격 증명 모델에 대한 자세한 내용은 [Cross-client Identity](#)를 참조하십시오.

OpenID Connect 공급자(자격 증명 풀)

[OpenID Connect](#)는 여러 로그인 공급자가 지원하는 공개 인증 표준입니다. Amazon Cognito에서는 [AWS Identity and Access Management](#)를 통해 구성되는 OpenID Connect 공급자와 자격 증명의 연결을 지원합니다.

OpenID Connect 공급자 추가

OpenID Connect 공급자를 생성하는 방법에 대한 자세한 내용은 [IAM 설명서](#)를 참조하십시오.

Amazon Cognito에 공급자 연결

IAM 콘솔에서 OpenID Connect 공급자를 만든 후에는 자격 증명 풀에 이 공급자를 연결할 수 있습니다. 구성된 모든 공급자는 Amazon Cognito 콘솔의 자격 증명 풀 편집 화면에서 OpenID Connect Providers 헤더 밑에 표시됩니다.

▼ OpenID Connect providers ⓘ

Amazon Cognito can authenticate users through any OpenID Connect provider. Once a provider has been configured with IAM, you can select the provider from the list below. [Learn more about using OpenID Connect providers.](#)

✓ accounts.google.com

✓ login.salesforce.com

여러 OpenID Connect 공급자를 단일 자격 증명 풀에 연결할 수 있습니다.

OpenID Connect 사용

로그인하고 ID 토큰을 받는 방법은 공급자 설명서를 참조하십시오.

토큰을 받은 후에는 공급자의 URI를 키로 사용하여 로그인 맵에 토큰을 추가하십시오.

OpenID Connect 토큰 확인

Amazon Cognito와 처음 통합할 때 `InvalidToken` 예외를 받을 수 있습니다. 각 OpenID Connect 토큰을 확인하는 방법을 이해해야 합니다.

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. 퍼블릭 키를 호스팅하는 자격 증명의 지문이 OpenID Connect 공급자에서 구성된 것과 일치해야 합니다.
4. azp 파라미터가 있으면 OpenID Connect 공급자에서 나열된 클라이언트 ID와 비교하여 이 값을 확인합니다.
5. azp 파라미터가 없으면 OpenID Connect 공급자에서 나열된 클라이언트 ID와 비교하여 aud 파라미터를 확인합니다.

웹 사이트 jwt.io를 참고하여 토큰을 디코딩해 이 값을 확인할 수 있습니다.

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 `AWSCognitoIdentityProviderManager` 프로토콜을 구현하십시오.

로그인 메서드를 구현할 때 다음 코드 예제와 같이, 구성된 OIDC 공급자 이름이 키로 포함되고 인증된 사용자에게서 받은 현재 ID 토큰이 값으로 포함된 딕셔너리를 반환하십시오.

```
class OIDCProvider: NSObject, AWSCognitoIdentityProviderManager {
    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 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")
}
}
```

AWSCognitoCredentialsProvider를 인스턴스화할 때 생성자에서 identityProviderManager의 값으로 AWSIdentityProviderManager를 구현하는 클래스를 전달하십시오. 자세한 내용을 보려면 [AWSCognitoCredentialsProvider](#) 참조 페이지에서 `initWithRegionType:identityPoolId:identityProviderManager`를 선택하십시오.

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 자격 증명 공급자(자격 증명 풀)

Amazon Cognito에서는 SAML 2.0(Security Assertion Markup Language 2.0)을 통해 자격 증명 공급자로 인증하도록 지원합니다. 와 함께 SAML을 지원하는 자격 증명 공급자를 사용하여 사용자를 위한 간단한 온보딩 흐름을 제공할 수 있습니다. SAML을 지원하는 자격 증명 공급자는 여러 사용자에게 서로 다른 권한 세트를 부여할 수 있도록 사용자가 위임할 수 있는 IAM 역할을 지정합니다.

SAML 공급자에 대해 자격 증명 풀 구성

다음 단계에서는 SAML 기반 공급자를 사용하도록 자격 증명 풀을 구성하는 방법을 설명합니다.

Note

SAML 공급자를 지원하도록 자격 증명 풀을 구성하기 전에 먼저 [IAM 콘솔](#)에서 SAML 자격 증명 공급자를 구성해야 합니다. 자세한 내용은 IAM 사용 설명서의 [타사 SAML 솔루션 공급자를 AWS와 통합](#)을 참조하십시오.

SAML 공급자를 지원하도록 자격 증명 풀을 구성하려면

1. [Amazon Cognito 콘솔](#)에 로그인하여 연동 자격 증명 관리를 선택하고 새 자격 증명 풀 만들기를 선택합니다.
2. 인증 공급자 섹션에서 SAML 탭을 선택합니다.
3. SAML 공급자의 ARN을 선택한 후 풀 생성을 선택합니다.

SAML 자격 증명 공급자 구성

SAML 공급자를 만든 후에는 SAML 자격 증명 공급자를 구성하여 자격 증명 공급자와 AWS 간에 신뢰 당사자 신뢰를 추가하십시오. 신뢰 당사자 정보와 인증서가 포함된 XML 문서를 자격 증명 공급자가 읽을 수 있는 URL을 지정하도록 허용하는 자격 증명 공급자가 많습니다. AWS에는 <https://signin.aws.amazon.com/static/saml-metadata.xml>을 사용할 수 있습니다. 다음 단계에서는 에 필요한 클레임을 채우도록 자격 증명 공급자의 SAML 어설션 응답을 구성합니다. 클레임 구성에 대한 자세한 내용은 [인증 응답을 위한 SAML 어설션 구성](#)을 참조하십시오.

SAML로 사용자 역할 사용자 지정

Amazon Cognito 자격 증명과 함께 SAML을 사용하면 최종 사용자에게 맞게 역할을 사용자 지정할 수 있습니다. SAML 기반 자격 증명 공급자에는 [고급 흐름 \(p. 229\)](#)만 지원됩니다. 자격 증명 풀에 SAML 기반 자격 증명 공급자를 사용하기 위해 인증되거나 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. 이 쌍이 사용자가 위임할 수 있는 역할입니다. SAML 자격 증명 공급자를 구성하여 자격 증명 공급자에서 얻을 수 있는 사용자 속성 정보를 기반으로 역할 속성을 채울 수 있습니다. SAML 어설션에서 여러 역할을 받으면 customRoleArn를 호출하는 동안 getCredentialsForIdentity 파라미터를 채워야 합니다. 파라미터에 수신된 입력 역할이 SAML 어설션의 클레임에 있는 역할과 일치하면 사용자가 입력 역할을 위임합니다.

SAML 자격 증명 공급자로 사용자 인증

SAML 기반 자격 증명 풀과 연동하려면 로그인을 시작하는 데 사용하는 URL을 결정해야 합니다. AWS 연동에는 IdP 시작 로그인이 사용됩니다. AD FS 2.0에서 URL 형식은 <https://<fqdn>/adfs/ls/IdpInitiatedSignOn.aspx?loginToRp=urn:amazon:webservices>입니다.

Amazon Cognito에서 SAML 자격 증명 공급자에 대한 지원을 추가하려면 먼저 iOS나 Android 앱에서 SAML 자격 증명 공급자로 사용자를 인증해야 합니다. SAML 자격 증명 공급자와 통합하고 이를 통해 인증하기 위한 코드는 SAML 공급자에 따라 다릅니다. 사용자가 인증된 후에 Amazon Cognito API를 사용하여 Amazon Cognito 자격 증명에 결과 SAML 어설션을 제공할 수 있습니다.

Android

Android SDK를 사용하는 경우 다음과 같이 SAML 어설션으로 로그인 맵을 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 Cognito service to get the credentials.  
credentialsProvider.getCredentials();
```

iOS

iOS SDK를 사용하는 경우 다음과 같이 `AWSSignInProviderManager`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 [Facebook\(자격 증명 풀\)](#) (p. 249), [Google\(자격 증명 풀\)](#) (p. 257) 및 [Login with Amazon\(자격 증명 풀\)](#) (p. 254)을 통한 웹 자격 증명 연동 이외에 개발자 인증 자격 증명을 지원합니다. 개발자 인증 자격 증명을 사용하면 를 사용하여 사용자 데이터를 동기화하고 AWS 리소스에 액세스하면서도 자신의 기존 인증 프로세스를 통해 사용자를 등록 및 인증할 수 있습니다. 개발자 인증 자격 증명의 사용에는 최종 사용자 장치, 인증을 위한 백엔드 및 간의 상호 작용이 포함됩니다. 자세한 내용은 AWS의 [블로그](#)를 읽어 보십시오.

인증 흐름 이해

개발자 인증 자격 증명 인증 흐름과 해당 인증 흐름이 외부 공급자 인증 흐름과 어떻게 다른지에 대한 자세한 내용은 [자격 증명 풀\(연동 자격 증명\) 인증 흐름](#) (p. 229)을 참조하십시오.

개발자 공급자 이름 정의 및 해당 이름과 자격 증명 풀 연결

개발자 인증 자격 증명을 사용하려면 개발자 공급자에 연결된 자격 증명 풀이 필요합니다. 이렇게 하려면 다음 단계를 따르십시오.

1. [Amazon Cognito 콘솔](#)에 로그인합니다.
2. 프로세스의 일환으로 새 자격 증명 풀을 만들고, 인증 공급자의 사용자 지정 탭에서 개발자 공급자 이름을 정의합니다.
3. 또는 기존 자격 증명 풀을 편집하고 인증 공급자의 사용자 지정 탭에서 개발자 공급자 이름을 정의합니다.

참고: 공급자 이름을 설정하면 해당 이름을 변경할 수 없습니다.

Amazon Cognito 콘솔 작업에 대한 추가 지침은 [Amazon Cognito 콘솔 사용](#) (p. 3) 단원을 참조하십시오.

자격 증명 공급자 구현

Android

개발자 인증 자격 증명을 사용하려면 `AWSAbstractCognitoIdentityProvider`를 확장하는 고유한 자격 증명 공급자 클래스를 구현합니다. 자격 증명 공급자 클래스는 토큰이 속성으로 포함된 응답 객체를 반환해야 합니다.

다음은 [샘플 앱](#)에 사용되는 자격 증명 공급자의 간단한 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 `CognitoCachingCredentialsProvider`에 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 [AWSCognitoCredentialsProviderHelper](#)를 확장하는 고유한 자격 증명 공급자 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 증명 및 개발자 인증 자격 증명을 둘 다 지원하려면 `logins` 구현에서 `AWSCognitoCredentialsProviderHelper` 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 증명과 소셜 공급자를 지원하려면 logins의 `AWSCognitoCredentialsProviderHelper` 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 `AWSCognitoCredentialsProviderHelper`를 확장하는 고유한 자격 증명 공급자 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 `AWSCognitoCredentialsProvider`에 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 증명 및 개발자 인증 자격 증명을 둘 다 지원하려면 logins 구현에서 `AWSCognitoCredentialsProviderHelper` 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 증명과 소셜 공급자를 지원하려면 logins의 AWSIdentityProviderHelper 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 접속하는 자격 증명 공급자의 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 접속하는 자격 증명 공급자의 간단한 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 인증 시스템에서 사용자를 고유하게 식별하는 영속자 문자열인 개발자 공급자 이름 및 개발자 사용자 식별자로 로그인 맵을 업데이트합니다. refresh가 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(예: 최종 사용자가 미인증 상태에서 인증 상태로 변경되었으며 인증된 사용자에게 대한 자격 증명을 원하는 경우) clearCredentials에 대해 credentialsProvider를 호출하십시오.

```
[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 사용자에게 대한 자격 증명을 원하는 경우) clearCredentials에 대해 credentialsProvider를 호출하십시오.


```
credentialsProvider.clearCredentials()
```

토큰 가져오기(서버 측)

`GetOpenIdTokenForDeveloperIdentity`를 호출하여 토큰을 받습니다. AWS 개발자 자격 증명을 사용하여 백 엔드에서 이 API를 호출해야 합니다. 클라이언트 SDK에서 호출해서는 안 됩니다. API는 Cognito 자격 증명 풀 ID, 자격 증명 공급자 이름이 키로 식별자가 값으로 포함된 로그인 맵 및 필요할 경우 Cognito 자격 증명 ID를 받습니다(즉, 미인증 사용자가 인증된 사용자로 변경됨). 식별자는 사용자의 사용자 이름, 이메일 주소 또는 숫자 값일 수 있습니다. API는 사용자에게 대한 고유한 Cognito ID와 최종 사용자에게 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 주의하십시오. 공격자가 이 토큰을 얻은 경우 공격자는 토큰 지속 시간 동안 최종 사용자의 AWS 자격 증명을 위해 해당 토큰을 교환할 수 있습니다.

다음 Java 조각은 Amazon Cognito 클라이언트를 시작하고 개발자 인증 자격 증명에 대한 토큰을 검색하는 방법을 보여줍니다.

```
// 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 * 151);
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
//...
```

위의 단계에 따라 앱에 개발자 인증 자격 증명을 통합할 수 있어야 합니다. 문제나 질문이 있는 경우 자유롭게 [포럼](#)에 게시해 주십시오.

기존 소셜 자격 증명에 연결

개발자 인증 자격 증명을 사용할 때 백엔드에서 모든 공급자 연결이 수행되어야 합니다. 사용자의 소셜 자격 증명(Facebook, Google 또는 Amazon)에 사용자 지정 자격 증명을 연결하려면 [GetOpenIdTokenForDeveloperIdentity](#)를 호출할 때 로그인 맵에 자격 증명 공급자 토큰을 추가하십시오. 이렇게 하려면 클라이언트 SDK에서 백엔드를 호출하여 최종 사용자를 인증할 때 최종 사용자의 소셜 공급자 토큰을 추가로 전달하십시오.

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_ACCESTOKEN");
```

공급자 간 전환 지원

Android

애플리케이션에서는 개발자 인증 자격 증명과 함께 퍼블릭 공급자(Amazon, Facebook 또는 Google을 통한 로그인)를 사용하여 미인증 자격 증명 또는 인증 자격 증명을 지원하는 것이 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. 따라서 모바일 애플리케이션은 앱 사용자가 선택한 사항에 따라 두 개의 뚜렷한 흐름을 지원할 수 있어야 합니다. 이를 위해 사용자 지정 자격 증명 공급자를 일부 변경해야 합니다.

refresh 메서드가 로그인 맵이 비어 있지 않은지 및 개발자 공급자 이름의 키가 있는지 확인한 다음 백엔드를 호출해야 합니다. 그렇지 않으면 getIdentityId 메서드를 호출하고 null이 반환됩니다.

```
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

애플리케이션에서는 개발자 인증 자격 증명과 함께 퍼블릭 공급자(Amazon, Facebook 또는 Google 을 통한 로그인)를 사용하여 미인증 자격 증명 또는 인증 자격 증명을 지원하는 것이 필요할 수 있습니다. 이렇게 하려면 현재 자격 증명 공급자에 따라 정확한 로그인 맵을 반환할 수 있도록 [AWSCognitoCredentialsProviderHelper](#) logins 메서드를 무시하십시오. 이 예는 미인증, Facebook 및 개발자 인증 간 피벗할 수 있는 방법을 보여줍니다.

```
- (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(즉, 로그인 디렉터리에서 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

애플리케이션에서는 개발자 인증 자격 증명과 함께 퍼블릭 공급자(Amazon, Facebook 또는 Google 을 통한 로그인)를 사용하여 미인증 자격 증명 또는 인증 자격 증명을 지원하는 것이 필요할 수

있습니다. 이렇게 하려면 현재 자격 증명 공급자에 따라 정확한 로그인 맵을 반환할 수 있도록 [AWSCognitoCredentialsProviderHelper](#) logins 메서드를 무시하십시오. 이 예는 미인증, Facebook 및 개발자 인증 간 피벗할 수 있는 방법을 보여줍니다.

```
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 디렉터리에서 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

애플리케이션에서는 개발자 인증 자격 증명과 함께 퍼블릭 공급자(Amazon, Facebook 또는 Google을 통한 로그인)를 사용하여 미인증 자격 증명 또는 인증 자격 증명을 지원하는 것이 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. 따라서 모바일 애플리케이션은 앱 사용자가 선택한 사항에 따라 두 개의 뚜렷한 흐름을 지원할 수 있어야 합니다. 이를 위해 사용자 지정 자격 증명 공급자를 일부 변경해야 합니다.

Unity에서 이를 수행하기 위한 권장 방법은 `AbstractCognitoIdentityProvider` 대신 `AmazonCognitoEnhancedIdentityProvider`에서 자격 증명 공급자를 확장하고, 사용자가 고유한 백엔드로 인증되지 않은 경우 고유한 메서드 대신 상위 `RefreshAsync` 메서드를 호출하는 것입니다. 사용자가 인증 상태이면 이전에 설명한 것과 동일한 흐름을 사용할 수 있습니다.

Xamarin

애플리케이션에서는 개발자 인증 자격 증명과 함께 퍼블릭 공급자(Amazon, Facebook 또는 Google을 통한 로그인)를 사용하여 미인증 자격 증명 또는 인증 자격 증명을 지원하는 것이 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. 따라서 모바일 애플리케이션은 앱 사용자가 선택한 사항에 따라 두 개의 뚜렷한 흐름을 지원할 수 있어야 합니다. 이를 위해 사용자 지정 자격 증명 공급자를 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 않은 경우에도 AWS 리소스에 대한 액세스 권한을 받습니다.

이 액세스 권한 등급은 로그인하기 전에 사용자에게 콘텐츠를 표시하는 데 유용합니다. 인증되지 않은 사용자는 개별적으로 로그인되지 않았으며 인증되지 않은 경우에도 자격 증명 풀에 고유한 자격 증명이 있습니다.

이 단원에서는 사용자가 인증되지 않은 자격 증명으로 로그인하는 방식에서 인증된 자격 증명으로 로그인하는 방식으로 전환하도록 선택하는 사례에 대해 설명합니다.

Android

사용자는 애플리케이션에 인증되지 않은 게스트로 로그인할 수 있습니다. 결국에는 지원되는 IdP 중 하나를 사용하여 로그인하도록 결정할 수 있습니다. Amazon Cognito는 기존 자격 증명이 새 자격 증명과 동일한 고유 식별자를 유지하도록 하며, 프로필 데이터가 자동으로 병합되도록 합니다.

애플리케이션은 `IdentityChangedListener` 인터페이스를 통한 프로필 병합 정보를 받습니다. 다음 메시지를 수신하기 위해 인터페이스에서 `identityChanged` 메서드를 구현합니다.

```
@Override
public void identityChanged(String oldIdentityId, String newIdentityId) {
    // handle the change
}
```

iOS - Objective-C

사용자는 애플리케이션에 인증되지 않은 게스트로 로그인할 수 있습니다. 결국에는 지원되는 IdP 중 하나를 사용하여 로그인하도록 결정할 수 있습니다. Amazon Cognito는 기존 자격 증명에 새 자격 증명과 동일한 고유 식별자를 유지하도록 하며, 프로필 데이터가 자동으로 병합되도록 합니다.

`NSNotificationCenter`는 애플리케이션에 프로필 병합을 알려줍니다.

```
[[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 애플리케이션에 인증되지 않은 게스트로 로그인할 수 있습니다. 결국에는 지원되는 IdP 중 하나를 사용하여 로그인하도록 결정할 수 있습니다. Amazon Cognito는 기존 자격 증명에 새 자격 증명과 동일한 고유 식별자를 유지하도록 하며, 프로필 데이터가 자동으로 병합되도록 합니다.

`NSNotificationCenter`는 애플리케이션에 프로필 병합을 알려줍니다.

```
[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. 이러한 역할의 경우, 로그인 속성 없이 구성 객체의 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, 자격 증명 객체를 업데이트하고 로그인 토큰을 추가하는 사용자 정의 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. 해당 객체를 생성하면 업데이트된 자격 증명 구성 정보를 반영하기 위해 모든 기존 서비스 객체의 자격 증명 속성을 재설정해야 합니다. [글로벌 구성 객체 사용하기](#)를 참조하십시오.

`CognitoIdentityCredentials` 객체에 관한 자세한 내용은 AWS SDK for JavaScript API 참조의 [AWS.CognitoIdentityCredentials](#)를 참조하십시오.

Unity

사용자는 애플리케이션에 인증되지 않은 게스트로 로그인할 수 있습니다. 결국에는 지원되는 IdP 중 하나를 사용하여 로그인하도록 결정할 수 있습니다. Amazon Cognito는 기존 자격 증명이 새 자격 증명과 동일한 고유 식별자를 유지하도록 하며, 프로필 데이터가 자동으로 병합되도록 합니다.

프로파일 병합에 대한 알림을 받도록 `IdentityChangedEvent`를 구독할 수 있습니다.

```
credentialsProvider.IdentityChangedEvent += delegate(object sender,  
    CognitoAWSCredentials.IdentityChangedEventArgs e)  
{  
    // handle the change  
    Debug.log("Identity changed from " + e.OldIdentityId + " to " + e.NewIdentityId);  
};
```

Xamarin

사용자는 애플리케이션에 인증되지 않은 게스트로 로그인할 수 있습니다. 결국에는 지원되는 IdP 중 하나를 사용하여 로그인하도록 결정할 수 있습니다. Amazon Cognito는 기존 자격 증명이 새 자격 증명과 동일한 고유 식별자를 유지하도록 하며, 프로필 데이터가 자동으로 병합되도록 합니다.

```
credentialsProvider.IdentityChangedEvent += delegate(object sender,
    CognitoAWSCredentials.IdentityChangedEventArgs e){
    // handle the change
    Console.WriteLine("Identity changed from " + e.OldIdentityId + " to " +
        e.NewIdentityId);
};
```

Amazon Cognito Sync

참고

Amazon Cognito Sync를 처음 접하는 사용자는 [AWS AppSync](#)를 이용하십시오. Amazon Cognito Sync처럼, 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 활성화하는 클라이언트 라이브러리 및 AWS 제품입니다. 를 사용하여 자체의 백엔드 없이 모바일 디바이스와 웹에서 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 Cognito 자격 증명 리전 가용성에 대한 자세한 내용은 [AWS 서비스 리전 가용성](#)을 참조하십시오.

Amazon Cognito Sync에 대한 자세한 내용은 다음 주제를 참조하십시오.

주제

- [Amazon Cognito Sync 시작하기](#) (p. 281)
- [데이터 동기화](#) (p. 282)
- [콜백 처리](#) (p. 289)
- [푸시 동기화](#) (p. 300)
- [Amazon Cognito 스트림](#) (p. 306)
- [Amazon Cognito 이벤트](#) (p. 308)

Amazon Cognito Sync 시작하기

참고

Amazon Cognito Sync를 처음 접하는 사용자는 [AWS AppSync](#)를 이용하십시오. Amazon Cognito Sync처럼, 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 활성화하는 클라이언트 라이브러리 및 AWS 제품입니다. 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.

AWS 계정에 가입

Amazon Cognito Sync를 사용하려면 AWS 계정이 있어야 합니다. 아직 계정이 없는 경우 다음 절차를 사용하여 가입하십시오.

AWS 계정에 가입하려면 다음을 수행합니다.

1. <https://aws.amazon.com/>을 열고 Create an AWS Account(AWS 계정 생성)를 선택합니다.

Note

전에 AWS 계정 루트 사용자 자격 증명을 사용하여 AWS Management 콘솔에 로그인한 적이 있는 경우 Sign in to a different account(다른 계정으로 로그인)를 선택합니다. 전에 IAM 자격 증명을 사용하여 콘솔에 로그인한 적이 있는 경우 Sign-in using root account credentials(루트 계정 자격 증명으로 로그인)를 선택합니다. 그런 다음 Create a new AWS account(새 AWS 계정 생성)를 선택합니다.

2. 온라인 지시 사항을 따릅니다.

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

Amazon Cognito에서 자격 증명 풀 설정

Amazon Cognito Sync에서는 사용자 자격 증명을 제공하기 위해 Amazon Cognito 자격 증명 풀이 필요합니다. 따라서 먼저 자격 증명 풀을 설정해야 사용할 수 있습니다. [Amazon Cognito 자격 증명 시작하기\(연동 자격 증명\)](#) (p. 223) 가이드를 따라 자격 증명 풀을 생성하고 SDK를 설치하십시오.

데이터 저장 및 동기화

자격 증명 풀을 설정하고 SDK를 설치한 경우 디바이스 간에 데이터를 저장하고 동기화할 수 있습니다. 자세한 내용은 [데이터 동기화](#) (p. 282)를 참조하십시오.

데이터 동기화

참고

Amazon Cognito Sync를 처음 접하는 사용자는 [AWS AppSync](#)를 이용하십시오. Amazon Cognito Sync처럼, 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 최종 사용자 데이터를 저장할 수 있습니다. 이 데이터는 자격 증명과 연관되어 로그인 및 디바이스에서 액세스할 수 있습니다. 서비스와 최종 사용자의 디바이스 간에 이 데이터를 동기화하려면 동기화 메서드를 호출하십시오. 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 하면 오프라인 상태인 경우에도 디바이스에서 수행된 모든 변경 사항을 즉시 디바이스에서 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 리소스를 취득합니다. 필수 헤더 파일도 가져와야 합니다. 다음 단계를 사용하여 클라이언트를 초기화합니다.

Android

1. [자격 증명 받기 \(p. 242\)](#)의 지침에 따라 자격 증명 공급자를 생성합니다.
2. Amazon Cognito 패키지를 가져옵니다. `import com.amazonaws.mobileconnectors.cognito.*;`
3. Amazon Cognito Sync를 초기화하여 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 받기 \(p. 242\)](#)의 지침에 따라 자격 증명 공급자를 생성합니다.
2. `AWSCore` 및 `Cognito`를 가져오고 `AWSCognito`를 초기화합니다.

```
#import <AWSSDKv2/AWSCore.h>  
#import <AWSCognitoSync/Cognito.h>  
  
AWSCognito *syncClient = [AWSCognito defaultCognito];
```

3. CocoaPods를 사용하는 경우 `<AWSSDKv2/AWSCore.h>`를 `AWSCore.h`로 대체하고 Amazon Cognito 가져오기와 동일한 구문을 따릅니다.

iOS - Swift

1. [자격 증명 받기 \(p. 242\)](#)의 지침에 따라 자격 증명 공급자를 생성합니다.
2. `AWSCognito`를 가져오고 초기화합니다.

```
import AWSCognito  
let syncClient = AWSCognito.default()!
```

JavaScript

1. JavaScript용 [Amazon Cognito Sync 관리자](#)를 다운로드합니다.
2. 프로젝트에 Sync Manager 라이브러리를 포함합니다.
3. [자격 증명 받기 \(p. 242\)](#)의 지침에 따라 자격 증명 공급자를 생성합니다.
4. Sync Manager를 초기화합니다.

```
var syncManager = new AWS.CognitoSyncManager();
```

Unity

1. `CognitoAWSCredentials`의 지침에 따라 먼저 [자격 증명 받기 \(p. 242\)](#)의 인스턴스를 생성해야 합니다.
2. `CognitoSyncManager`의 인스턴스를 생성하여 `CognitoAwsCredentials` 객체와 `AmazonCognitoSyncConfig`를 최소한 리전 집합과 함께 전달합니다.

```
AmazonCognitoSyncConfig clientConfig = new AmazonCognitoSyncConfig { RegionEndpoint =  
    REGION };  
CognitoSyncManager syncManager = new CognitoSyncManager(credentials, clientConfig);
```

Xamarin

1. CognitoAWSCredentials의 지침에 따라 먼저 [자격 증명 받기 \(p. 242\)](#)의 인스턴스를 생성해야 합니다.
2. CognitoSyncManager의 인스턴스를 생성하여 CognitoAwsCredentials 객체와 AmazonCognitoSyncConfig를 최소한 리전 집합과 함께 전달합니다.

```
AmazonCognitoSyncConfig clientConfig = new AmazonCognitoSyncConfig { RegionEndpoint =  
    REGION };  
CognitoSyncManager syncManager = new CognitoSyncManager(credentials, clientConfig);
```

데이터 세트 이해

Amazon Cognito를 사용하면 최종 사용자 프로파일 데이터가 데이터 세트로 구성됩니다. 각 데이터 세트에는 최대 1MB의 데이터가 키 값 쌍의 양식으로 포함될 수 있습니다. 데이터 세트는 동기화 작업을 수행할 수 있는 가장 세분화된 개체입니다. 데이터 세트에서 수행된 읽기 및 쓰기 작업은 동기화 메서드가 호출될 때까지 로컬 스토어에만 영향을 줍니다. 데이터 세트는 고유한 문자열로 식별됩니다. 다음에 표시된 대로 새 데이터 세트를 생성하거나 기존 데이터 세트를 열 수 있습니다.

Android

```
Dataset dataset = client.openOrCreateDataset("datasetname");
```

데이터 세트를 삭제하려면 먼저 메서드를 호출하여 로컬 스토리지에서 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 먼저 메서드를 호출하여 로컬 스토리지에서 제거한 다음 synchronize 메서드를 호출하여 Amazon Cognito에서 데이터 세트를 삭제합니다.

```
[dataset clear];  
[dataset synchronize];
```

iOS - Swift

```
let dataset = syncClient.openOrCreateDataset("myDataSet")!
```

데이터 세트를 삭제하려면 먼저 메서드를 호출하여 로컬 스토리지에서 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");
```

Remove를 사용하여 데이터 세트에서 키를 삭제할 수 있습니다.

```
dataset.Remove("myKey");
```

Xamarin

```
Dataset dataset = syncManager.OpenOrCreateDataset("myDatasetName");
```

데이터 세트를 삭제하려면 먼저 메서드를 호출하여 로컬 스토리지에서 제거한 다음 synchronize 메서드를 호출하여 Amazon Cognito에서 데이터 세트를 삭제합니다.

```
dataset.Delete();  
dataset.SynchronizeAsync();
```

데이터 세트의 데이터 읽기 및 쓰기

Amazon Cognito 데이터 세트는 키를 통해 값에 액세스할 수 있는 사전으로 작동합니다. 데이터 세트의 키와 값은 데이터 세트가 사전인 것처럼 읽거나, 추가하거나, 수정할 수 있습니다. 다음은 그 한 예입니다.

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

remove 메서드를 사용하여 데이터 세트에서 키를 제거할 수 있습니다.

```
dataset.remove("myKey");
```

iOS - Objective-C

removeObjectForKey를 사용하여 데이터 세트에서 키를 삭제할 수 있습니다.

```
[dataset removeObjectForKey:@"myKey"];
```

iOS - Swift

removeObjectForKey를 사용하여 데이터 세트에서 키를 삭제할 수 있습니다.

```
dataset.removeObjectForKey("myKey")
```

Unity

Remove를 사용하여 데이터 세트에서 키를 삭제할 수 있습니다.

```
dataset.Remove("myKey");
```

Xamarin

Remove를 사용하여 데이터 세트에서 키를 삭제할 수 있습니다.

```
dataset.Remove("myKey");
```

로컬 데이터를 Sync 스토어와 동기화

Android

synchronize 메서드는 로컬로 캐시된 데이터를 Amazon Cognito Sync 스토어에 저장된 데이터와 비교합니다. 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 \(p. 289\)](#)를 참조하십시오.

iOS - Objective-C

synchronize 메서드는 로컬로 캐시된 데이터를 Amazon Cognito Sync 스토어에 저장된 데이터와 비교합니다. 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 `synchronizeOnConnectivity`는 1) 다음에 디바이스가 온라인 상태가 될 때 동기화를 예약하며 2) nil 결과와 함께 `AWSTask`를 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 `AWSCognito`에 있는 알림의 관찰자를 추가해야 합니다.

데이터 세트 동기화 및 다른 콜백에 대한 자세한 내용은 [콜백 처리 \(p. 289\)](#)를 참조하십시오.

iOS - Swift

`synchronize` 메서드는 로컬로 캐시된 데이터를 Amazon Cognito Sync 스토어에 저장된 데이터와 비교합니다. 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 `synchronizeOnConnectivity`는 연결을 확인하며, 디바이스가 온라인 상태인 경우 즉시 `synchronize`를 호출하고 시도와 연관된 `AWSTask` 객체를 반환합니다.

디바이스가 오프라인인 경우 `synchronizeOnConnectivity`는 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 `AWSCognito`에 있는 알림의 관찰자를 추가해야 합니다.

데이터 세트 동기화 및 다른 콜백에 대한 자세한 내용은 [콜백 처리 \(p. 289\)](#)를 참조하십시오.

JavaScript

`synchronize` 메서드는 로컬로 캐시된 데이터를 Amazon Cognito Sync 스토어에 저장된 데이터와 비교합니다. 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 \(p. 289\)](#)를 참조하십시오.

Unity

`synchronize` 메서드는 로컬로 캐시된 데이터를 Amazon Cognito Sync 스토어에 저장된 데이터와 비교합니다. 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 \(p. 289\)](#)를 참조하십시오.

Xamarin

synchronize 메서드는 로컬로 캐시된 데이터를 Amazon Cognito Sync 스토어에 저장된 데이터와 비교합니다. 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 \(p. 289\)](#)를 참조하십시오.

콜백 처리

참고

Amazon Cognito Sync를 처음 접하는 사용자는 [AWS AppSync](#)를 이용하십시오. Amazon Cognito Sync처럼, 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 단원에서는 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 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 캐시된 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 `NSArray`가 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 `NSArray`가 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.

`AWSCognitoRecordConflictHandler`를 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. `AWSCognitoConflict`를 사용하면 로컬 레코드 [`conflict resolveWithLocalRecord`]와 원격 레코드 [`conflict resolveWithRemoteRecord`] 또는 새로운 값인 [`conflict resolveWithValue:value`]를 사용하여 충돌을 해결할 수 있습니다. 이 메서드에서 `nil`을 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 를 사용하여 `AWSCognitoDatasetDeletedHandler`로컬로 캐시된 데이터 세트의 사본도 삭제해야 하는지 여부를 확인합니다. `AWSCognitoDatasetDeletedHandler`가 구현되지 않으면 로컬 데이터가 자동으로 제거됩니다. 제거하기 전에 로컬 데이터의 사본을 유지하거나 로컬 데이터를 유지하려면 `AWSCognitoDatasetDeletedHandler`를 구현하십시오.

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 `DatasetMergeHandler`를 통해 병합에 대한 알림을 받습니다. 핸들러는 루트 데이터 세트의 이름과 루트 데이터 세트의 병합으로 표시된 데이터 세트 이름의 배열을 받습니다.

`DatasetMergeHandler`가 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 `NSNotificationCenter`를 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. `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 `NSArray`가 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 `NSArray`가 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.

`AWSCognitoRecordConflictHandler`를 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. `AWSCognitoConflict`를 사용하면 로컬 레코드 [`conflict resolveWithLocalRecord`]와 원격 레코드 [`conflict resolveWithRemoteRecord`] 또는 새로운 값인 [`conflict resolveWithValue:value`]를 사용하여 충돌을 해결할 수 있습니다. 이 메서드에서 `nil`을 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 를 사용하여 `AWSCognitoDatasetDeletedHandler`로컬로 캐시된 데이터 세트의 사본도 삭제해야 하는지 여부를 확인합니다. `AWSCognitoDatasetDeletedHandler`가 구현되지 않으면 로컬 데이터가 자동으로 제거됩니다. 제거하기 전에 로컬 데이터의 사본을 유지하거나 로컬 데이터를 유지하려면 `AWSCognitoDatasetDeletedHandler`를 구현하십시오.

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 세트 병합 핸들러

이전에 연결되지 않은 두 자격 증명이 서로 연결되면 해당 데이터 세트가 모두 병합됩니다. 애플리케이션은 `DatasetMergeHandler`를 통해 병합에 대한 알림을 받습니다. 핸들러는 루트 데이터 세트의 이름과 루트 데이터 세트의 병합으로 표시된 데이터 세트 이름의 배열을 받습니다.

`DatasetMergeHandler`가 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.');
```

```
}

```

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(); //Asynchronously 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

참고

Amazon Cognito Sync를 처음 접하는 사용자는 [AWS AppSync](#)를 이용하십시오. Amazon Cognito Sync처럼, 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 안내서](#)에 설명된 대로 지원되는 플랫폼에 대해 Amazon SNS 앱을 생성 및 구성합니다.

Amazon Cognito 콘솔에서 푸시 동기화 활성화

Amazon Cognito 콘솔을 통해 푸시 동기화를 활성화할 수 있습니다. [콘솔 홈 페이지](#)에서 다음을 수행합니다.

1. 푸시 동기화를 활성화할 자격 증명 풀 이름을 클릭합니다. 자격 증명 풀에 대한 대시보드 페이지가 표시됩니다.
2. 대시보드 페이지의 우측 상단 모서리에서 연동 자격 증명 관리를 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 콘솔](#)에서 역할 생성을 클릭하여 자격 증명 풀과 연관된 역할을 생성하거나 수정합니다.
5. 플랫폼 애플리케이션을 선택한 다음 변경 사항 저장을 클릭합니다.
6. 애플리케이션에 SNS 액세스 권한 부여

[IAM 콘솔](#)에서 전체 SNS 액세스 권한을 보유하도록 IAM 역할을 구성하거나, cognito-sync를 신뢰하고 전체 SNS 액세스 권한이 있는 새 역할을 생성합니다. IAM 역할에 대한 자세한 내용은 [역할\(위임 및 연동\)](#)을 참조하십시오.

앱에서 푸시 동기화 사용: Android

애플리케이션에서는 Google Play 서비스를 가져와야 합니다. [Android SDK Manager](#)를 통해 최신 버전의 Google Play SDK를 다운로드할 수 있습니다. [GCM 클라이언트 구현](#)의 Android 설명서에 따라 앱을 등록하고 GCM으로부터 등록 ID를 받습니다. 등록 ID가 있는 경우 아래 조각에 표시된 대로 를 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 사용하십시오.

```
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 따르십시오. 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 사용하십시오.

```
[[[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 `subscribeAll`을 호출하십시오.

```
[[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 따르십시오. 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 사용하십시오.

```
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 호출하십시오.

```
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 호출하십시오.

```
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

참고

Amazon Cognito Sync를 처음 접하는 사용자는 [AWS AppSync](#)를 이용하십시오. Amazon Cognito Sync처럼, 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.

니다. Amazon Cognito는 각 데이터 세트 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 시작하기](#)를 참조하십시오.

스트림 구성

Amazon Cognito 콘솔에서 Amazon Cognito 스트림을 설정할 수 있습니다. Amazon Cognito 콘솔에서 Amazon Cognito 스트림을 활성화하려면 게시할 Kinesis 스트림과 선택한 스트림에 이벤트를 게시할 Amazon Cognito 권한을 부여하는 IAM 역할을 선택해야 합니다.

[콘솔 홈 페이지](#)에서 다음을 수행합니다.

1. Amazon Cognito 스트림을 설정할 자격 증명 풀 이름을 클릭합니다. 자격 증명 풀에 대한 대시보드 페이지가 표시됩니다.
2. 대시보드 페이지의 우측 상단 모서리에서 연동 자격 증명 관리를 클릭합니다. [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 안내서](#)를 참조하십시오.
5. 게시 역할 드롭다운 메뉴에서 스트림을 게시할 Amazon Cognito 권한을 부여하는 IAM 역할을 선택합니다. [AWS IAM 콘솔](#)에서 역할 생성을 클릭하여 자격 증명 풀과 연관된 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
}
```

50KB의 Kinesis 최대 페이로드 크기보다 큰 업데이트의 경우 업데이트의 전체 내용이 포함된 미리 서명된 Amazon S3 URL이 포함됩니다.

Amazon Cognito 스트림을 구성한 후 Kinesis 스트림을 삭제하거나 Amazon Cognito 동기화에서 더 이상 담당할 수 없도록 역할 신뢰 권한을 변경할 경우 Amazon Cognito 스트림이 비활성화됩니다. 스트림을 다시 생성하거나 역할을 수정한 다음 스트림을 다시 활성화해야 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 는 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 [Start bulk publish]를 클릭합니다. 대량 게시 작업은 지정된 시간에 한 번으로 제한되며 대량 게시 요청은 24시간마다 한 번으로 제한됩니다.

Amazon Cognito 이벤트

참고

Amazon Cognito Sync를 처음 접하는 사용자는 [AWS AppSync](#)를 이용하십시오. Amazon Cognito Sync처럼, 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 통해 Amazon Cognito에서 중요한 이벤트에 반응하여 AWS Lambda 함수를 실행할 수 있습니다. 데이터 세트가 동기화될 경우 Amazon Cognito에서는 동기화 트리거 이벤트가 발생합니다. 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 플레이어의 새 레벨에 도달할 때 상 제공 등 수신 데이터에 따라 데이터 세트의 다른 값을 업데이트하는 데 유용합니다.

아래 단계는 Amazon Cognito 데이터 세트가 동기화될 때마다 실행하는 Lambda 함수를 설정하는 방법을 보여줍니다.

Note

Amazon Cognito 이벤트를 사용할 때 Amazon Cognito 자격 증명에서 받은 자격 증명만 사용할 수 있습니다. 연결된 Lambda 함수가 있지만 AWS 계정 자격 증명(개발자 자격 증명)을 사용하여 UpdateRecords를 호출하는 경우 Lambda 함수가 호출되지 않습니다.

AWS Lambda에서 함수 생성

Lambda를 Amazon Cognito와 통합하려면 먼저 Lambda에서 함수를 생성해야 합니다. 그렇게 하려면 다음을 수행하십시오.

Amazon Cognito에서 Lambda 함수 선택

1. Lambda 콘솔을 엽니다.
2. Create a Lambda function(Lambda 함수 생성)을 클릭합니다.
3. [Select blueprint] 화면에서 "cognito-sync-trigger"를 검색하고 선택합니다.

4. [Configure event sources] 화면에서 이벤트 소스 유형을 "Cognito Sync 트리거"로 설정해 두고 자격 증명 풀을 선택합니다. [Next]를 클릭합니다.
5. [Configure function] 화면에서 함수 이름과 설명을 입력합니다. [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 참조하십시오. [Advanced] 설정을 변경하지 않고 그대로 둡니다. [Next]를 클릭합니다.
6. [Review] 화면에서 세부 정보를 검토하고 [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. [Dashboard] 페이지의 우측 상단 모서리에서 [Manage Federated Identities]를 클릭합니다. [Manage Federated Identities] 페이지가 나타납니다.
3. 아래로 스크롤하고 [Cognito Events]를 클릭하여 확장합니다.
4. 동기화 트리거 드롭다운 메뉴에서 동기화 이벤트가 발생할 때 트리거할 Lambda 함수를 선택합니다.
5. [Save Changes]를 클릭합니다.

이제 데이터 세트가 동기화될 때마다 Lambda 함수가 실행됩니다. 다음 단원에서는 함수에서 데이터가 동기화될 때 이 데이터를 읽고 수정하는 방법에 대해 설명합니다.

동기화 트리거에 대한 Lambda 함수 작성

동기화 트리거는 서비스 제공자 인터페이스 프로그래밍 패러다임을 따릅니다. Amazon Cognito는 다음 JSON 형식으로 Lambda 함수에 입력을 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. 전체 예제는 아래에 나와 있습니다.

동기화 트리거 이벤트에 대한 함수를 작성할 때 염두에 두어야 할 몇 가지 핵심 요지는 다음과 같습니다.

- UpdateRecords 중 Lambda 함수가 호출될 때 5초 이내에 응답해야 합니다. 그렇지 않으면 Amazon Cognito Sync 서비스에서는 `LambdaSocketTimeoutException` 예외가 발생합니다. 이 제한 시간 값은 늘릴 수 없습니다.
- `LambdaThrottledException` 예외가 발생할 경우 동기화 작업(레코드 업데이트)을 다시 시도해야 합니다.
- Amazon Cognito는 데이터 세트에 있는 모든 레코드를 함수에 대한 입력으로 제공합니다.
- 앱 사용자가 업데이트한 레코드에서는 'op' 필드가 "replace"로 설정되고 삭제된 레코드에서는 'op' 필드가 "remove"로 설정됩니다.
- 앱 사용자가 업데이트하지 않은 경우에도 레코드를 수정할 수 있습니다.
- `datasetRecords`를 제외한 모든 필드는 읽기 전용이며 변경할 수 없습니다. 이러한 필드를 변경하면 레코드 업데이트가 실패합니다.
- 레코드 값을 수정하려면 값을 업데이트하고 'op'를 "replace"로 설정하면 됩니다.
- 레코드를 제거하려면 'op'를 "remove"로 설정하거나 값을 null로 설정하십시오.
- 레코드를 추가하려면 `datasetRecords` 어레이에 새 레코드를 추가하면 됩니다.
- 응답에서 생략된 레코드는 업데이트가 무시됩니다.

예제 Lambda 함수

다음은 데이터에 액세스하고 데이터를 수정 및 제거하는 방법을 보여주는 예제 Lambda 함수입니다.

```
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 Cognito의 제한 값

이 단원에서는 Amazon Cognito 제한을 설명합니다.

Note

`accessToken`을 입력으로 사용하는 작업은 사용자에게 따라 조절됩니다. 이 조절은 사용자 풀 조절에 추가됩니다. 아래에 언급된 기본 제한은 모든 사용자에게 대한 전체 제한을 나타냅니다.

주제

- [소프트 제한 \(p. 311\)](#)
- [하드 제한 \(p. 312\)](#)

소프트 제한

다음 표는 Amazon Cognito에 대한 소프트(기본값) 제한을 제공합니다. 소프트 제한은 변경될 수 있는 제한입니다. 이러한 제한과 이를 변경하는 방법에 대한 자세한 내용은 [AWS 서비스 제한](#)을 참조하십시오.

Amazon Cognito 사용자 풀 리소스의 소프트 제한

리소스	기본 한도
사용자 풀당 최대 앱 수	1000
계정당 최대 사용자 풀 수	1000
사용자 풀당 최대 사용자 가져오기 작업 수	1000
사용자 풀당 자격 증명 공급자 최대 수	300
사용자 풀당 리소스 서버 최대 수	25
사용자 풀당 매일 보낸 이메일 수	50

Amazon Cognito 사용자 풀 API의 소프트 제한

API 제한(초당 요청 수)	기본 한도
<code>AdminInitiateAuth</code>	20
<code>AdminRespondToAuthChallenge</code>	20
<ul style="list-style-type: none">• <code>AdminAddUserToGroup</code>• <code>AdminRemoveUserFromGroup</code>• <code>AdminListGroupsForUser</code>	10
<code>SignUp</code> , <code>InitiateAuth</code> (sign in) 및 <code>ForgotPassword</code> 와 같은 사용자 인증 작업.	10
다음 API를 생성, 업데이트 및 삭제합니다. <ul style="list-style-type: none">• <code>UserPool</code>• <code>UserPoolClient</code>• <code>UserImportJob</code>• <code>UserPoolDomain</code>	1

API 제한(초당 요청 수)	기본 한도
<ul style="list-style-type: none"> IdentityProvider ResourceServer 	
다음 API를 설명합니다. <ul style="list-style-type: none"> UserPool UserPoolClient UserImportJob UserPoolDomain 	5
다음 API의 목록을 작성합니다. <ul style="list-style-type: none"> UserPool UserPoolClients UserImportJobs 사용자 	5
SetUICustomization, AddCustomAttributes	1
Admin API는 위에 나열되지 않습니다.	5

Amazon Cognito 자격 증명 풀 소프트웨어 제한(연동 자격 증명)

리소스	기본 한도
계정당 최대 자격 증명 풀 수	1000
자격 증명 풀당 최대 Amazon Cognito 사용자 풀 공 급자	10
역할 기반 액세스 제어(RBAC)용 규칙 최대 수	25

Amazon Cognito Sync의 소프트웨어 제한

리소스	기본 한도
자격 증명당 최대 데이터 세트 수	20
데이터 세트당 최대 레코드 수	1024
단일 데이터 세트의 최대 크기	1MB

하드 제한

다음 표는 Amazon Cognito 하드 제한을 설명합니다. 하드 제한은 변경될 수 없는 제한입니다.

Amazon Cognito 사용자 풀의 하드 제한

리소스	제한
사용자 풀당 최대 사용자 지정 속성 수	25

리소스	제한
속성당 최대 문자	2048 bytes
사용자 지정 속성 이름에 대한 최대 문자 길이	20
최소/최대 암호 정책 길이	6~99(경계값 포함)
이메일 제목의 최대 문자	140
이메일 메시지의 최대 문자	20,000건
SMS 확인 메시지의 최대 문자	140
암호의 최대 문자	256
자격 증명 공급자 이름에 대한 최대 문자 길이	40
자격 증명 공급자당 최대 식별자	50
자격 증명 공급자당 최대 콜백 URL	100
자격 증명 공급자당 최대 로그아웃 URL	100
리소스 서버당 범위의 최대 수	60
계정당 사용자 지정 도메인의 최대 수	4
각 사용자가 소속될 수 있는 최대 그룹 수	25
사용자 풀당 최대 그룹 수	500

Amazon Cognito 사용자 풀의 토큰 유효성에 대한 하드 제한

리소스	제한
ID 토큰	1시간
새로 고침 토큰	1일~3650일 사이(경계값 포함)

Amazon Cognito 사용자 풀의 하드 제한

리소스	제한
사용자 풀당 최대 사용자 지정 속성 수	25
속성당 최대 문자	2048 bytes
사용자 지정 속성 이름에 대한 최대 문자 길이	20
최소/최대 암호 정책 길이	6~99(경계값 포함)
사용자 풀당 매일 보낸 이메일 수. 이 한도는 Amazon Cognito 사용자 풀에 기본 이메일 기능을 사용하는 경우에만 적용됩니다. 더 많은 이메일 전송 볼륨을 활성화하려면 Amazon SES 이메일 구성을 사용하도록 사용자 풀을 구성함	50

리소스	제한
니다. 자세한 내용은 Amazon Cognito 사용자 풀에 대한 이메일 설정 (p. 182) 단원을 참조하십시오.	
이메일 제목의 최대 문자	140
이메일 메시지의 최대 문자	20,000건
SMS 확인 메시지의 최대 문자	140
암호의 최대 문자	256
자격 증명 공급자 이름에 대한 최대 문자 길이	40
자격 증명 공급자당 최대 식별자	50
자격 증명 공급자당 최대 콜백 URL	100
자격 증명 공급자당 최대 로그아웃 URL	100
리소스 서버당 범위의 최대 수	60
계정당 사용자 지정 도메인의 최대 수	4
각 사용자가 소속될 수 있는 최대 그룹 수	25

Amazon Cognito 사용자 풀의 코드 유효성에 대한 하드 제한

리소스	제한
가입 확인 코드	24시간
사용자 속성 확인 코드 유효성	24시간
멀티 팩터 인증 코드	3 minutes
암호 찾기 코드	1시간

Amazon Cognito 자격 증명 풀 하드 제한(연동 자격 증명)

리소스	제한
자격 증명 풀당 최대 자격 증명 수	무제한
자격 증명 풀 이름에 대한 최대 문자 길이	128 bytes
로그인 공급자 이름에 대한 최대 문자 길이	2048 bytes
단일 List/Lookup API 호출의 최대 결과 수	60

Amazon Cognito Sync의 하드 제한

리소스	제한
데이터 세트 이름에 대한 최대 문자 길이	128 bytes

리소스	제한
요청 성공 후 대량 게시를 위한 최소 대기 시간	24시간

Amazon Cognito API 참조

Amazon Cognito API 참조에 대한 자세한 내용은 다음 주제를 참조하십시오.

주제

- [Amazon Cognito 사용자 풀 API 참조 \(p. 316\)](#)
- [Amazon Cognito 사용자 풀 Auth API 참조 \(p. 316\)](#)
- [Amazon Cognito 자격 증명 풀\(연동 자격 증명\) API 참조 \(p. 327\)](#)
- [Amazon Cognito 동기화 API 참조 \(p. 327\)](#)

Amazon Cognito 사용자 풀 API 참조

Amazon Cognito 사용자 풀로 웹과 모바일 앱 사용자들이 가입하고 로그인하도록 할 수 있습니다. 인증 사용자에게 대해 암호를 변경할 수 있고 미인증 사용자에게 대해 잊어버린 암호 흐름을 시작할 수 있습니다. 자세한 내용은 [사용자 풀 인증 흐름 \(p. 187\)](#) 및 [사용자 풀을 통해 토큰 사용 \(p. 190\)](#) 단원을 참조하십시오.

사용자 풀 API 참조의 전체 내용은 [Amazon Cognito 사용자 풀 API 참조](#) 단원을 참조하십시오.

Amazon Cognito 사용자 풀 Auth API 참조

사용자 풀의 도메인이 구성된 이후에는 가입 및 로그인 웹페이지를 앱에 추가할 수 있는 인증 서버를 Amazon Cognito가 호스팅합니다. 자세한 내용은 [호스팅 UI를 활성화하는 앱 추가](#) 단원을 참조하십시오.

이 단원에는 예제 요청 및 응답을 비롯하여 사용자 풀 클라이언트의 Amazon Cognito 인증 서버에 대한 HTTPS 계약이 포함되어 있습니다. 여기에서는 긍정적 및 부정적 조건에 대해 인증 서버의 예상되는 동작을 설명합니다.

서버 계약 REST API 외에도 Amazon Cognito는 간편하게 요청을 작성하고 서버와 상호 작용할 수 있도록 Android, iOS 및 JavaScript용 Auth SDK도 제공합니다. [Amazon Cognito SDK](#)에 대해 자세히 알아보십시오.

사양에 대한 자세한 내용은 [OpenID Connect 1.0](#) 및 [OAuth 2.0](#)을 참조하십시오.

주제

- [권한 부여 엔드포인트 \(p. 316\)](#)
- [토큰 엔드포인트 \(p. 320\)](#)
- [USERINFO 엔드포인트 \(p. 324\)](#)
- [로그인 엔드포인트 \(p. 325\)](#)
- [로그아웃 엔드포인트 \(p. 326\)](#)

권한 부여 엔드포인트

`/oauth2/authorize` 엔드포인트는 사용자를 로그인합니다.

GET /oauth2/authorize

`/oauth2/authorize` 엔드포인트는 HTTPS GET만 지원합니다. 사용자 풀 클라이언트는 보통 브라우저를 통해 이 요청을 합니다. 웹 브라우저에는 Chrome 또는 Firefox가 포함됩니다. Android 브라우저에는 Custom Chrome Tab이 포함됩니다. iOS 브라우저에는 Safari View Control이 포함됩니다.

권한 부여 서버는 권한 부여 엔드포인트에 액세스할 때 프로토콜로 HTTP가 아니라 HTTPS가 필요합니다. 사양에 대한 자세한 내용은 [권한 부여 엔드포인트](#)를 참조하십시오.

요청 파라미터

response_type

응답 유형이며 code 또는 token이어야 합니다. 클라이언트가 최종 사용자용 권한 부여 코드(권한 부여 코드 허용 흐름)를 원하는지 아니면 최종 사용자에게 직접 토큰을 발급하는지(암시적 흐름) 여부를 나타냅니다.

필수

client_id

클라이언트 ID입니다.

사용자 풀에서 미리 등록된 클라이언트여야 하며 연동에 대해 활성화되어야 합니다.

필수

redirect_uri

사용자가 권한 부여를 허용한 후 인증 서버에서 브라우저를 리디렉션하는 URL입니다.

리디렉션 URI는 다음을 충족해야 합니다.

- 절대 URI이어야 합니다.
- 클라이언트로 미리 등록되어야 합니다.
- 조각 구성요소가 없어야 합니다.

[OAuth 2.0 - Redirection Endpoint](#) 단원을 참조하십시오.

테스트 목적으로만 `http://localhost`를 사용하는 경우를 제외하고 Amazon Cognito에서는 HTTP가 아니라 HTTPS가 필요합니다.

`myapp://example` 같은 앱 콜백 URL도 지원됩니다.

필수

state

클라이언트가 최초 요청에 추가할 불투명 값입니다. 클라이언트로 다시 리디렉션될 때 권한 부여 서버에 이 값이 포함됩니다.

[CSRF](#) 공격을 방지하려면 클라이언트가 이 값을 사용해야 합니다.

이는 선택 사항이지만 매우 권장됩니다.

identity_provider

개발자가 특정 공급자를 통해 직접 인증하는 데 사용됩니다.

- 소셜 로그인에서 유효한 값은 Facebook, Google 및 LoginWithAmazon입니다.
- Amazon Cognito 사용자 풀에서 유효한 값은 COGNITO입니다.
- 다른 자격 증명 공급자의 경우, 사용자 풀에서 IdP에 할당한 이름이 이 값이 됩니다.

선택

idp_identifier

개발자가 공급자 이름을 노출시키지 않고 공급자 이름을 매핑하는 데 사용됩니다.

선택

scope

시스템에 예약된 범위나 클라이언트와 연결된 사용자 지정 범위를 조합하여 사용할 수 있습니다. 범위는 공백으로 구분해야 합니다. 시스템에 예약된 범위로는 `openid`, `email`, `phone`, `profile` 및 `aws.cognito.signin.user.admin`이 있습니다. 사용된 범위는 클라이언트와 미리 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.

선택

code_challenge_method

챌린지를 생성하는 데 사용된 메서드입니다. [PKCE RFC](#)는 S256 및 일반의 두 가지 메서드를 정의하지만 Amazon Cognito 인증 서버는 S256만 지원합니다.

선택 사항

code_challenge

`code_verifier`에서 생성된 챌린지입니다.

`code_challenge_method`가 지정된 경우에만 필수입니다.

긍정 응답을 통한 요청의 예

권한 부여 코드 허용

예제 요청

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&
client_id=ad398u21ijw3s9w3939&
redirect_uri=https://YOUR_APP/redirect_uri&
state=STATE&
scope=openid+profile+aws.cognito.signin.user.admin
```

예제 응답

Amazon Cognito 인증 서버는 권한 부여 코드 및 상태를 통해 앱으로 다시 리디렉션합니다. 조각이 아닌 쿼리 문자열 파라미터에 코드 및 상태가 반환되어야 합니다. 쿼리 문자열은 '?' 문자 다음에 나오는 웹 요청의 일부이며 이 문자열은 '&' 문자로 구분된 하나 이상의 파라미터를 포함할 수 있습니다. 조각은 '#' 문자 다음에 나오는 웹 요청의 일부이며 문서의 하위 섹션을 지정합니다.

```
HTTP/1.1 302 Found
Location: https://YOUR_APP/redirect_uri?code=AUTHORIZATION_CODE&state=STATE
```

PKCE를 통한 권한 부여 코드 허용

예제 요청

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&
```

```
client_id=ad398u21ijw3s9w3939&
redirect_uri=https://YOUR_APP/redirect_uri&
state=STATE&
scope=aws.cognito.signin.user.admin&
code_challenge_method=S256&
code_challenge=CODE_CHALLENGE
```

예제 응답

인증 서버는 권한 부여 코드 및 상태를 통해 앱으로 다시 리디렉션합니다. 조각이 아닌 쿼리 문자열 파라미터에 코드 및 상태가 반환되어야 합니다.

```
HTTP/1.1 302 Found
Location: https://YOUR_APP/redirect_uri?code=AUTHORIZATION_CODE&state=STATE
```

openid 범위가 없는 토큰 부여

예제 요청

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=token&
client_id=ad398u21ijw3s9w3939&
redirect_uri=https://YOUR_APP/redirect_uri&
state=STATE&
scope=aws.cognito.signin.user.admin
```

예제 응답

Amazon Cognito 권한 부여 서버는 액세스 토큰을 통해 앱으로 다시 리디렉션합니다. openid 범위가 요청되지 않았으므로 ID 토큰이 반환되지 않습니다. 이 흐름에서 새로 고침 토큰은 반환되지 않습니다. 토큰과 상태는 쿼리 문자열이 아닌 조각에 반환됩니다.

```
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

예제 요청

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/
authorize?
response_type=token&
client_id=ad398u21ijw3s9w3939&
redirect_uri=https://YOUR_APP/redirect_uri&
state=STATE&
scope=aws.cognito.signin.user.admin+openid+profile
```

예제 응답

권한 부여 서버는 액세스 토큰 및 ID 토큰을 통해 앱으로 다시 리디렉션합니다(openid 범위가 포함되어 있기 때문).

```
HTTP/1.1 302 Found
Location: https://YOUR_APP/
redirect_ur#id_token=ID_TOKEN&access_token=ACCESS_TOKEN&token_type=bearer&expires_in=3600&state=STATE
```

부정 응답의 예

다음은 부정 응답의 예입니다.

- `client_id` 및 `redirect_uri`가 유효하지만 요청 파라미터에 다른 문제가 있는 경우(예: `response_type`이 포함되지 않은 경우, `code_challenge`가 제공되지만 `code_challenge_method`가 제공되지 않은 경우 또는 `code_challenge_method`가 'S256'이 아닌 경우) 인증 서버는 클라이언트의 `redirect_uri`에 해당 오류를 리디렉션합니다.

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?
error=invalid_request
```

- 클라이언트가 `response_type`에 '코드' 또는 '토큰'을 요청하지만 이러한 요청에 대한 권한이 없는 경우 Amazon Cognito 권한 부여 서버는 다음과 같이 클라이언트의 `redirect_uri`에 `unauthorized_client`를 반환해야 합니다.

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?
error=unauthorized_client
```

- 클라이언트가 유효하지 않고, 알 수 없으며, 잘못된 범위를 요청한 경우 Amazon Cognito 권한 부여 서버는 다음과 같이 클라이언트의 `redirect_uri`에 `invalid_scope`를 반환해야 합니다.

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?error=invalid_scope
```

- 서버에 예상치 못한 오류가 있는 경우 인증 서버는 클라이언트의 `server_error`에 `redirect_uri`를 반환해야 합니다. 이 오류는 클라이언트에 전송되지 않으므로 브라우저의 최종 사용자에게 표시되는 HTTP 500 오류가 아니어야 합니다. 다음 오류가 반환되어야 합니다.

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?error=server_error
```

토큰 엔드포인트

`/oauth2/token` 엔드포인트는 사용자의 토큰을 가져옵니다.

POST /oauth2/token

`/oauth2/token` 엔드포인트는 HTTPS POST만 지원합니다. 사용자 풀 클라이언트는 시스템 브라우저를 통해서가 아닌 직접 이 엔드포인트를 요청합니다.

사양에 대한 자세한 내용은 [토큰 엔드포인트](#)를 참조하십시오.

헤더의 요청 파라미터

승인

클라이언트에 보안이 발급된 경우 클라이언트는 기본 HTTP 권한 부여를 통해 권한 부여 헤더에서 해당 `client_id` 및 `client_secret`을 전달해야 합니다. 보안은 `기본 Base64Encode(client_id:client_secret)`입니다.

Content-Type

항상 `'application/x-www-form-urlencoded'`여야 합니다.

본문의 요청 파라미터

`grant_type`

허용 유형입니다.

`authorization_code`, `refresh_token` 또는 `client_credentials`가 있습니다.

필수

client_id

클라이언트 ID입니다.

사용자 풀에 미리 등록된 클라이언트여야 합니다. 클라이언트는 Amazon Cognito 연동에 대해 활성화되어 있어야 합니다.

클라이언트가 공개되어 있어 암호가 필요하지 않을 경우 필수 항목입니다.

scope

클라이언트와 연결된 사용자 지정 범위를 조합하여 사용할 수 있습니다. 요청된 범위는 클라이언트와 미리 연결이 되어 있어야 합니다. 그렇지 않으면 런타임 시 무시됩니다. 클라이언트가 범위를 요청하지 않은 경우 인증 서버에서는 클라이언트와 연결된 모든 사용자 지정 범위를 사용합니다.

선택. grant_type이 client_credentials인 경우에만 사용됩니다.

redirect_uri

/oauth2/authorize에서 redirect_uri를 얻기 위해 사용했던 authorization_code와 같아야 합니다.

grant_type이 authorization_code인 경우에만 필수입니다.

refresh_token

새로 고침 토큰입니다.

Note

새로 고침 토큰은 사양에 정의되어 있지만, [토큰 엔드포인트](#)에서 반환되도록 현재 구현되지 않습니다.

code

grant_type이 authorization_code인 경우 필수입니다.

code_verifier

증명 키입니다.

grant_type이 authorization_code이며 PKCE를 통해 권한 부여 코드가 요청된 경우 필수입니다.

긍정 응답을 통한 요청의 예

토큰에 대한 권한 부여 코드 교환

예제 요청

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token&
Content-Type='application/x-www-form-urlencoded'&
Authorization=Basic aSdx892iujendek328uedj

grant_type=authorization_code&
client_id=djc98u3jiedmi283eu928&
code=AUTHORIZATION_CODE&
redirect_uri=com.myclientapp://myclient/redirect
```

샘플 응답

```
HTTP/1.1 200 OK
Content-Type: application/json
```



```
{
  "access_token": "eyJz9sdfsd fsdfsd",
  "refresh_token": "dn43ud8uj32nk2je",
  "id_token": "dmcxd329ujdmkemkd349r",
  "token_type": "Bearer",
  "expires_in": 3600
}
```

Note

새로고침 토큰은 사양에 정의되어 있지만, [토큰 엔드포인트](#)에서 반환되도록 현재 구현되지 않습니다.

액세스 토큰에서 클라이언트 자격 증명 교환

예제 요청

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token >
Content-Type='application/x-www-form-urlencoded'&
Authorization=Basic aSdx d892iujendek328uedj

grant_type=client_credentials&
scope={resourceServerIdentifier1}/{scope1} {resourceServerIdentifier2}/{scope2}
```

샘플 응답

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "access_token": "eyJz9sdfsd fsdfsd",
  "token_type": "Bearer",
  "expires_in": 3600
}
```

PKCE를 통해 부여된 토큰에 대한 권한 부여 코드 교환

예제 요청

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token
Content-Type='application/x-www-form-urlencoded'&
Authorization=Basic aSdx d892iujendek328uedj

grant_type=authorization_code&
client_id=djc98u3jiedmi283eu928&
code=AUTHORIZATION_CODE&
code_verifier=CODE_VERIFIER&
redirect_uri=com.myclientapp://myclient/redirect
```

샘플 응답

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "access_token": "eyJz9sdfsd fsdfsd",
  "refresh_token": "dn43ud8uj32nk2je",
  "id_token": "dmcxd329ujdmkemkd349r",
  "token_type": "Bearer",
  "expires_in": 3600
}
```

```
}
```

Note

새로고침 토큰은 사양에 정의되어 있지만, [토큰 엔드포인트](#)에서 반환되도록 현재 구현되지 않습니다.

토큰에 대한 새로 고침 토큰 교환

예제 요청

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token >
Content-Type='application/x-www-form-urlencoded'
Authorization=Basic aSdx892iujendek328uedj

grant_type=refresh_token&
client_id=djc98u3jiedmi283eu928&
refresh_token=REFRESH_TOKEN
```

예제 응답

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "access_token": "eyJz9sdfsdfsdfsdfsd",
  "refresh_token": "dn43ud8uj32nk2je",
  "id_token": "dmcxd329ujdmkemkd349r",
  "token_type": "Bearer",
  "expires_in": 3600
}
```

Note

새로고침 토큰은 사양에 정의되어 있지만, [토큰 엔드포인트](#)에서 반환되도록 현재 구현되지 않습니다.

부정 응답의 예

샘플 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.

unauthorized_client

클라이언트가 코드 부여 흐름이나 토큰 새로 고침에 대해 허용되지 않습니다.

unsupported_grant_type

grant_type이 authorization_code 또는 refresh_token 이외의 것인 경우 반환됩니다.

USERINFO 엔드포인트

/oauth2/userInfo 엔드포인트는 인증된 사용자에게 대한 정보를 반환합니다.

GET /oauth2/userInfo

사용자 풀 클라이언트는 브라우저를 통해서가 아닌 직접 이 엔드포인트를 요청합니다.

자세한 내용은 OpenID Connect(OIDC) 사양의 [UserInfo 엔드포인트](#)를 참조하십시오.

주제

- [헤더의 요청 파라미터](#) (p. 324)
- [예제 요청](#) (p. 324)
- [긍정 응답 예](#) (p. 324)
- [부정 응답의 예](#) (p. 325)

헤더의 요청 파라미터

권한 부여: 보유자 `<access_token>`

권한 부여 헤더 필드를 사용하여 액세스 토큰을 전달합니다.

필수

예제 요청

```
GET https://<your-user-pool-domain>/oauth2/userInfo
Authorization: Bearer <access_token>
```

긍정 응답 예

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
  "sub": "248289761001",
  "name": "Jane Doe",
  "given_name": "Jane",
  "family_name": "Doe",
  "preferred_username": "j.doe",
  "email": "janedoe@example.com"
}
```

OIDC 클레임 목록은 [표준 클레임](#)을 참조하십시오.

부정 응답의 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 파라미터가 누락되거나, 지원되지 않는 파라미터 값이 포함되거나, 요청의 형식이 잘못되었습니다.

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 만료되었거나, 취소되었거나, 형식이 잘못되었거나, 유효하지 않습니다.

로그인 엔드포인트

/login 엔드포인트는 사용자를 로그인합니다. 로그인 페이지를 로드하고 클라이언트에 대해 구성된 인증 옵션을 사용자에게 표시합니다.

/login 획득

/login 엔드포인트는 HTTPS GET만 지원합니다. 사용자 풀 클라이언트는 시스템 브라우저를 통해 이 요청을 합니다. JavaScript용 시스템 브라우저에는 Chrome 또는 Firefox가 포함됩니다. Android 브라우저에는 Custom Chrome Tab이 포함됩니다. iOS 브라우저에는 Safari View Control이 포함됩니다.

요청 파라미터

client_id

앱에 대한 앱 클라이언트 ID입니다. 앱 클라이언트 ID를 얻으려면 앱을 사용자 풀에 등록해야 합니다. 자세한 내용은 [사용자 풀 앱 클라이언트 구성 \(p. 214\)](#) 단원을 참조하십시오.

필수

redirect_uri

인증에 성공한 후에 사용자가 리디렉션된 URI입니다. 지정된 response_type의 client_id에서 구성되어야 합니다.

필수

response_type

코드 부여 흐름에 대해 code가 되고 암시적 흐름에 대해 token이 될 수 있는 OAuth 응답 유형입니다.

필수

state

클라이언트가 최초 요청에 추가할 불투명 값입니다. 값은 리디렉션에 대한 클라이언트에 다시 반환됩니다.

[CSRF](#) 공격을 방지하려면 클라이언트가 이 값을 사용해야 합니다.

이는 선택 사항이지만 매우 권장됩니다.

scope

시스템에 예약된 범위나 클라이언트와 연결된 사용자 지정 범위를 조합하여 사용할 수 있습니다. 범위는 공백으로 구분해야 합니다. 시스템에 예약된 범위로는 `openid`, `email`, `phone`, `profile` 및 `aws.cognito.signin.user.admin`이 있습니다. 사용된 범위는 클라이언트와 미리 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.

선택

예제 요청: 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
```

로그아웃 엔드포인트

`/logout` 엔드포인트는 사용자를 로그아웃합니다.

GET /logout

`/logout` 엔드포인트는 HTTPS GET만 지원합니다. 사용자 풀 클라이언트는 일반적으로 시스템 브라우저를 통해 이 요청을 수행합니다. 해당 브라우저는 일반적으로 Android의 Custom Chrome Tab 및 iOS의 Safari View Control입니다.

요청 파라미터

client_id

앱에 대한 앱 클라이언트 ID입니다. 앱 클라이언트 ID를 얻으려면 앱을 사용자 풀에 등록해야 합니다. 자세한 내용은 [사용자 풀 앱 클라이언트 구성 \(p. 214\)](#) 단원을 참조하십시오.

선택

logout_uri

클라이언트 앱에 등록된 로그아웃 URL입니다. 자세한 내용은 [사용자 풀 앱 클라이언트 구성 \(p. 73\)](#) 단원을 참조하십시오.

선택

예제 요청

예제 #1: 로그아웃 및 클라이언트로 다시 리디렉션

이 예제는 기존 세션을 지우고 클라이언트로 다시 리디렉션합니다. 두 파라미터는 모두 필수입니다.

```
GET https://mydomain.auth.us-east-1.amazonaws.com/logout?
client_id=ad398u21ijw3s9w3939&
logout_uri=com.myclientapp://myclient/logout
```

예제 #2: 로그아웃 및 사용자에게 다른 사용자로 로그인하라는 메시지 표시

이 예제는 기존 세션을 지우고 GET /oauth2/authorize와 동일한 파라미터를 사용하여 로그인 화면을 표시합니다.

```
GET https://mydomain.auth.us-east-1.amazonaws.com/logout?
response_type=code&
client_id=ad398u21ijw3s9w3939&
redirect_uri=https://YOUR_APP/redirect_uri&
state=STATE&
scope=openid+profile+aws.cognito.signin.user.admin
```

Amazon Cognito 자격 증명 풀(연동 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(연동 자격 증명) API 참조의 전체 내용은 [Amazon Cognito API Reference](#) 단원을 참조하십시오.

Amazon Cognito 동기화 API 참조

[Amazon Cognito Sync API Reference](#)

AWS CloudTrail을 사용하여 Amazon Cognito API 호출 로깅

Amazon Cognito는 Amazon Cognito에서 사용자, 역할 또는 AWS 서비스가 수행한 작업의 레코드를 제공하는 서비스인 AWS CloudTrail과 통합됩니다. CloudTrail은 Amazon Cognito 콘솔의 호출 및 Amazon Cognito API에 대한 코드 호출의 호출을 포함하여 Amazon Cognito에 대한 API 호출의 하위 세트를 이벤트로 캡처합니다. 추적을 생성하면 CloudTrail 이벤트를 비롯하여 Amazon Cognito 이벤트를 Amazon S3 버킷으로 지속적으로 배포할 수 있습니다. 추적을 구성하지 않은 경우 이벤트 기록에서 CloudTrail 콘솔의 최신 이벤트를 볼 수도 있습니다. CloudTrail에서 수집하는 정보를 사용하여 Amazon Cognito에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

그 구성 및 활성화 방법을 포함하여 CloudTrail에 대한 자세한 내용은 [AWS CloudTrail User Guide](#)를 참조하십시오.

특정 CloudTrail 이벤트에 대해 Amazon CloudWatch 경보를 만들 수도 있습니다. 예를 들어, 자격 증명 풀 구성이 변경되면 경보를 트리거하도록 설정할 수 있습니다. 자세한 내용은 [CloudTrail 이벤트에 대한 CloudWatch 경보 생성: 예](#)를 참조하십시오.

CloudTrail의 Amazon Cognito 정보

CloudTrail은 계정 생성 시 AWS 계정에서 활성화됩니다. 지원되는 이벤트 활동이 Amazon Cognito에서 이루어지면 해당 활동이 이벤트 이력의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. AWS 계정에서 최신 이벤트를 확인, 검색 및 다운로드할 수 있습니다. 자세한 정보는 [CloudTrail 이벤트 기록에서 이벤트 보기](#)를 참조하십시오.

Amazon Cognito 이벤트를 포함하여 AWS 계정에 이벤트를 지속적으로 기록하려는 경우 추적을 생성합니다. 추적은 CloudTrail이 Amazon S3 버킷으로 로그 파일을 전송할 수 있도록 합니다. 콘솔에서 추적을 생성하면 기본적으로 모든 리전에 추적이 적용됩니다. 추적은 AWS 파티션에 있는 모든 리전의 이벤트를 로깅하고 지정된 Amazon S3 버킷으로 로그 파일을 전송합니다. 또는 CloudTrail 로그에서 수집된 이벤트 데이터를 추가 분석 및 처리하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음을 참조하십시오.

- [추적 생성 개요](#)
- [CloudTrail 지원 서비스 및 통합](#)
- [CloudTrail에 대한 Amazon SNS 알림 구성](#)
- [여러 리전에서 CloudTrail 로그 파일 받기 및 여러 계정에서 CloudTrail 로그 파일 받기](#)

Amazon Cognito는 CloudTrail 로그 파일의 이벤트로 다음 작업의 로깅을 지원합니다.

Amazon Cognito 사용자 풀

- [AddCustomAttributes](#)
- [CreateUserImportJob](#)
- [CreateUserPool](#)
- [CreateUserPoolClient](#)
- [DeleteUserPool](#)
- [DeleteUserPoolClient](#)
- [DescribeUserImportJob](#)

- [DescribeUserPool](#)
- [DescribeUserPoolClient](#)
- [GetCSVHeader](#)
- [ListUserImportJobs](#)
- [ListUserPoolClients](#)
- [ListUserPools](#)
- [StartUserImportJob](#)
- [StopUserImportJob](#)
- [UpdateUserPool](#)
- [UpdateUserPoolClient](#)

Amazon Cognito 연동 자격 증명

- [CreateIdentityPool](#)
- [DeleteIdentityPool](#)
- [DescribeIdentityPool](#)
- [GetIdentityPoolRoles](#)
- [ListIdentityPools](#)
- [SetIdentityPoolRoles](#)
- [UpdateIdentityPool](#)

Amazon Cognito Sync

- [BulkPublish](#)
- [DescribeIdentityPoolUsage](#)
- [GetBulkPublishDetails](#)
- [GetCognitoEvents](#)
- [GetIdentityPoolConfiguration](#)
- [ListIdentityPoolUsage](#)
- [SetCognitoEvents](#)
- [SetIdentityPoolConfiguration](#)

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. 자격 증명 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 아니면 IAM 사용자 자격 증명으로 했는지 여부
- 역할 또는 연합된 사용자에 대한 임시 보안 자격 증명을 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청했는지 여부.

자세한 내용은 [CloudTrail userIdentity 요소](#)를 참조하십시오.

예제: Amazon Cognito 로그 파일 항목

추적은 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 제공할 수 있도록 해 주는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함됩니다. 이벤트는 특정 소스의 단일 요청을 나타내며 요청된 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보가 들어 있습니다. CloudTrail 로그 파일은 퍼블릭 API 호출의 순서가 지정된 스택 추적이 아니기 때문에 특정 순서로 표시되지 않습니다.

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": "0000000000000000"
    }
  },
  "responseElements": {
    "identityPoolName": "TestPool",
    "identityPoolId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE",
    "allowUnauthenticatedIdentities": true,
    "supportedLoginProviders": {
      "graph.facebook.com": "0000000000000000"
    }
  },
  "requestID": "15cc73a1-0780-460c-91e8-e12ef034e116",
  "eventID": "f1d47f93-c708-495b-bff1-cb935a6064b2",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

Amazon Cognito 리소스에 태그 지정

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 제품이 태그 지정을 지원합니다. 따라서 서로 다른 서비스의 리소스에 같은 태그를 할당하여 리소스가 관련이 있음을 나타낼 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 이러한 태그를 활성화합니다. AWS는 태그를 사용하여 비용을 분류하고 월별 비용 할당 보고서를 제공합니다. 자세한 내용은 AWS Billing and Cost Management 사용 설명서의 [비용 할당 태그 사용](#) 항목을 참조하십시오.
- 할당된 태그를 기반으로 리소스에 대한 액세스를 제어합니다. AWS Identity and Access Management(IAM) 정책 조건에서 태그 키와 값을 지정하여 액세스를 제어합니다. 예를 들어 IAM 사용자에게 사용자 풀 업데이트를 허용할 수 있지만 사용자 풀의 해당 사용자 이름의 값에 `owner` 태그가 있는 경우에만 가능합니다. 자세한 내용은 IAM 사용 설명서에서 [태그를 사용하여 액세스 제어](#) 항목을 참조하십시오.

AWS CLI 또는 Amazon Cognito API를 사용하여 사용자 풀 및 자격 증명 풀에 태그를 추가, 편집 또는 삭제할 수 있습니다. 특히 사용자 풀의 경우 Amazon Cognito 콘솔을 사용하여 태그를 관리할 수 있습니다.

태그 사용에 대한 팁은 AWS Answers 블로그의 게시글 [AWS Tagging Strategies](#)를 참조하십시오.

다음 섹션에서는 Amazon Cognito의 태그에 대한 추가 정보를 제공합니다.

Amazon Cognito에서 지원되는 리소스

Amazon Cognito의 다음 리소스는 태그 지정을 지원합니다.

- 사용자 풀
- 자격 증명 풀

태그 제한

Amazon Cognito 리소스의 태그에 다음과 같은 기본 제한 사항이 적용됩니다.

- 리소스에 할당할 수 있는 최대 태그 수 - 50
- 최대 키 길이 - 유니코드 128자
- 최대 값 길이 - 유니코드 256자
- 키 및 값에 사용할 수 있는 문자 - a-z, A-Z, 0-9, 공백 및 `_`, `.`, `/`, `=`, `+`, `-`, `@` 문자
- 키와 값은 대/소문자를 구분합니다
- 키 접두사로 `aws:`를 사용하지 마세요. AWS 전용입니다.

Amazon Cognito 콘솔 사용으로 태그 관리

Amazon Cognito 콘솔을 사용하여 사용자 풀에 할당된 태그를 관리할 수 있습니다.

자격 증명 풀의 경우 콘솔에 태그 지정 기능이 없으므로 프로그래밍 방식으로 태그를 관리해야 합니다. 예를 들어 AWS CLI를 사용할 수 있습니다.

사용자 풀에 태그를 추가하는 방법

1. AWS Management 콘솔에 로그인한 다음 <https://console.aws.amazon.com/cognito>에서 Amazon Cognito 콘솔을 엽니다.
2. Manage User Pools(사용자 풀 관리)를 선택합니다.
3. 사용자 풀 페이지에서 태그를 추가할 사용자 풀을 선택합니다.
4. 왼쪽 탐색 메뉴에서 태그를 선택합니다.
5. 사용자 풀에 이미 태그가 없으면 태그 추가를 선택하여 첫 번째 태그를 추가합니다.
6. 태그 키 및 태그 값에 값을 지정합니다.
7. 추가하려는 각 추가 태그에 다른 태그 추가를 선택합니다.
8. 태그 추가가 완료되면 변경 사항 저장을 선택합니다.

태그 페이지에서 기존 태그의 키와 값을 편집할 수도 있습니다. 태그를 제거하려면 태그의 오른쪽 상단에 있는 x를 선택합니다.

AWS CLI 예

AWS CLI는 Amazon Cognito 사용자 풀 및 자격 증명 풀에 할당하는 태그를 관리하는 데 사용할 수 있는 명령을 제공합니다.

태그 할당

다음 명령을 사용하여 기존 사용자 풀 및 자격 증명 풀에 태그를 할당합니다.

Example **tag-resource** 사용자 풀 명령

cognito-idp 명령 집합에서 **tag-resource**를 사용하여 태그를 사용자 풀에 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 탭에서 풀 ARN 값을 확인합니다.
- **tags** – 태그의 키-값 쌍입니다.

한 번에 여러 태그를 할당하려면 쉼표로 구분된 목록에 태그를 지정합니다.

```
$ aws cognito-idp tag-resource \  
> --resource-arn user-pool-arn \  
> --tags Stage=Test, CostCenter=80432, Owner=SysEng
```

Example **tag-resource** 자격 증명 풀 명령

cognito-identity 명령 집합에서 **tag-resource**를 사용하여 태그를 자격 증명 풀에 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 편집을 선택합니다. 그런 다음 자격 증명 풀 ID에서 Show ARN(ARN 표시)을 선택합니다.
- **tags** - 태그의 키-값 쌍입니다.

한 번에 여러 태그를 할당하려면 쉼표로 구분된 목록에 태그를 지정합니다.

```
$ aws cognito-identity tag-resource \
> --resource-arn identity-pool-arn \
> --tags Stage=Test, CostCenter=80432, Owner=SysEng
```

태그 보기

사용자 풀 및 자격 증명 풀에 할당된 태그를 보려면 다음 명령을 사용합니다.

Example list-tags-for-resource 사용자 풀 명령

cognito-idp 명령 집합에서 **list-tags-for-resource**를 사용하여 사용자 풀에 할당된 태그를 봅니다.

```
$ aws cognito-idp list-tags-for-resource --resource-arn user-pool-arn
```

Example list-tags-for-resource 자격 증명 풀 명령

cognito-identity 명령 집합에서 **list-tags-for-resource**를 사용하여 자격 증명 풀에 할당된 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 untag-resource 사용자 풀 명령

cognito-idp 명령 집합에서 **untag-resource**를 사용하여 사용자 풀에서 태그를 제거합니다.

```
$ aws cognito-idp untag-resource \
> --resource-arn user-pool-arn \
> --tag-keys Stage CostCenter Owner
```

--tag-keys 파라미터의 경우 하나 이상의 태그 키를 지정하고 태그 값은 포함하지 마십시오.

Example untag-resource 자격 증명 풀 명령

cognito-identity 명령 집합에서 **untag-resource**를 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 하나 이상의 태그 키를 지정하고 태그 값은 포함하지 마십시오.

리소스를 생성할 때 태그 적용

다음 명령을 사용하여 사용자 풀 또는 자격 증명 풀을 생성할 때 태그를 할당합니다.

Example **create-user-pool** 태그가 있는 명령

create-user-pool 명령을 사용하여 사용자 풀을 생성할 때 --user-pool-tags 파라미터에 태그를 지정할 수 있습니다.

```
$ aws cognito-idp create-user-pool \  
> --pool-name user-pool-name \  
> --user-pool-tags Stage=Test, CostCenter=80432, Owner=SysEng
```

Example **create-identity-pool** 태그가 있는 명령

create-identity-pool 명령을 사용하여 자격 증명 풀을 생성할 때 --identity-pool-tags 파라미터에 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
```

Amazon Cognito API를 사용하여 태그 관리

Amazon Cognito API에서 다음 작업을 사용하여 사용자 풀 및 자격 증명 풀에 대한 태그를 관리할 수 있습니다.

사용자 풀 태그에 대한 API 작업

다음 API 작업을 사용하여 사용자 풀에 대한 태그 할당, 보기 및 제거를 수행합니다.

- [TagResource](#)
- [ListTagsForResource](#)
- [UntagResource](#)
- [CreateUserPool](#)

자격 증명 풀 태그에 대한 API 작업

다음 API 작업을 사용하여 자격 증명 풀에 대한 태그 할당, 보기 및 제거를 수행합니다.

- [TagResource](#)
- [ListTagsForResource](#)
- [UntagResource](#)
- [CreateIdentityPool](#)

리소스 권한

이 단원에서는 AWS Identity and Access Management(IAM)을 통한 Amazon Cognito 리소스 액세스 제한을 다룹니다. 자격 증명 풀을 AWS 리소스 액세스를 제어하기 위한 사용자 풀 그룹과 함께 사용하는 방법에 대한 자세한 내용은 [사용자 풀에 그룹 추가](#) (p. 166) 및 [역할 기반 액세스 제어](#) (p. 238) 단원을 참조하십시오. 자격 증명 풀 및 IAM에 대한 자세한 내용은 [자격 증명 풀 개념\(연동 자격 증명\)](#) (p. 229) 단원을 참조하십시오.

Amazon 리소스 이름(ARN)

Amazon Cognito 연동 자격 증명을 위한 ARN

Amazon Cognito 자격 증명 풀(연동 자격 증명)에서는 다음 예제와 같이 Amazon 리소스 이름(ARN) 형식을 사용하여 IAM 사용자의 특정 자격 증명 풀에 대한 액세스를 제한할 수 있습니다. ARN에 대한 자세한 내용은 [IAM 식별자](#)를 참조하십시오.

```
arn:aws:cognito-identity:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID
```

Amazon Cognito Sync를 위한 ARN

Amazon Cognito Sync에서는 고객이 자격 증명 풀 ID, 자격 증명 ID 및 데이터 세트 이름으로 액세스를 제한할 수도 있습니다.

작업 증명 풀에서 작동하는 API의 경우 작업 증명 풀 ARN 형식은 Amazon Cognito 연동 작업 증명과 동일하지만 서비스 이름이 cognito-identity가 아닌 cognito-sync인 것만 다릅니다.

```
arn:aws:cognito-sync:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID
```

RegisterDevice와 같이 단일 자격 증명에서 작동하는 API의 경우 다음의 ARN 형식으로 개별 자격 증명을 참조할 수 있습니다.

```
arn:aws:cognito-sync:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID/identity/IDENTITY_ID
```

UpdateRecords 및 ListRecords와 같이 데이터 세트에서 작동하는 API의 경우 다음의 ARN 형식으로 개별 데이터 세트를 참조할 수 있습니다.

```
arn:aws:cognito-sync:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID/identity/IDENTITY_ID/dataset/DATASET_NAME
```

Amazon Cognito 사용자 풀을 위한 ARN

Amazon Cognito 사용자 풀에서는 다음 ARN 형식을 사용하여 특정 사용자 풀에 대한 IAM 사용자의 액세스를 제한할 수 있습니다.

```
arn:aws:cognito-idp:REGION:ACCOUNT_ID:userpool/USER_POOL_ID
```

정책 예제

특정 자격 증명 풀에 대한 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"
    }
  ]
}
```

폴에 있는 모든 자격 증명의 특정 데이터 세트에 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"
    }
  ]
}
```

관리형 정책

고객이 Amazon Cognito에 대한 액세스 권한을 부여하는 데 사용할 수 있는 IAM 콘솔을 통해 다양한 정책이 제공됩니다.

- AmazonCognitoPowerUser - 자격 증명 풀의 모든 측면을 액세스하고 관리할 수 있는 권한
- AmazonCognitoPowerUser - 자격 증명 풀에 읽기 전용으로 액세스할 수 있는 권한
- AmazonCognitoDeveloperAuthenticatedIdentities - 인증 시스템이 Amazon Cognito와 통합하기 위한 권한

Amazon Cognito 팀에서 이 정책을 유지하므로 새로운 API가 추가되더라도 IAM 사용자는 계속해서 동일한 수준의 액세스 권한을 갖습니다.

Note

새로운 자격 증명 풀을 만들려면 IAM 역할도 만들어야 하기 때문에 새 자격 증명 풀을 생성하려는 IAM 사용자는 관리 정책을 적용해야 합니다.

서명된 API와 서명되지 않은 API 비교

AWS 자격 증명으로 서명된 API는 IAM 정책을 통해 제한할 수 있습니다. 다음 Cognito API는 서명되지 않았으므로 IAM 정책을 통해 제한할 수 없습니다.

Amazon Cognito 연동 자격 증명

- `GetId`
- `GetOpenIdToken`
- `GetCredentialsForIdentity`
- `UnlinkIdentity`

Amazon Cognito 사용자 풀

- `ChangePassword`
- `ConfirmDevice`
- `ConfirmForgotPassword`
- `ConfirmSignUp`
- `DeleteUser`
- `DeleteUserAttributes`
- `ForgetDevice`
- `ForgotPassword`
- `GetDevice`
- `GetUser`
- `GetUserAttributeVerificationCode`
- `GlobalSignOut`
- `InitiateAuth`
- `ListDevices`
- `ResendConfirmationCode`
- `RespondToAuthChallenge`
- `SetUserSettings`
- `SignUp`
- `UpdateDeviceStatus`
- `UpdateUserAttributes`
- `VerifyUserAttribute`

Amazon Cognito에 서비스 연결 역할 사용

Amazon Cognito의 경우 AWS Identity and Access Management(IAM) [서비스 연결 역할](#)을 사용합니다. 서비스 연결 역할은 Amazon Cognito에 직접 연결된 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 Amazon Cognito에서 사전 정의하며 서비스에서 다른 AWS 서비스를 자동으로 호출하기 위해 필요한 모든 권한을 포함합니다.

서비스 연결 역할을 통해 Amazon Cognito 설정이 쉬워지는데 필요한 권한을 수동으로 추가할 필요가 없기 때문입니다. Amazon Cognito에서 서비스 연결 역할 권한을 정의하므로, 달리 정의되지 않은 한 Amazon Cognito에서만 해당 역할을 맡을 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며, 이 권한 정책은 다른 IAM 개체에 연결할 수 없습니다.

먼저 관련 리소스를 삭제한 후에만 서비스 연결 역할을 삭제할 수 있습니다. 이렇게 하면 리소스에 대한 액세스 권한을 부주의로 삭제할 수 없기 때문에 Amazon Cognito 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용은 [IAM으로 작업하는 AWS 서비스](#)를 참조하고 서비스 연결 역할 옆에 예가 있는 서비스를 찾으십시오. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

Amazon Cognito에 대한 서비스 연결 역할 권한

Amazon Cognito에서는 다음 서비스 연결 역할을 사용합니다.

- AmazonCognitoIdpEmailService – Allows Amazon Cognito User Pools service to use your SES identities for email sending.

AmazonCognitoIdpEmailService 서비스 연결 역할은 역할을 위임하기 위해 다음 서비스를 신뢰합니다.

- `email.cognito-idp.amazonaws.com`

역할 권한 정책은 Amazon Cognito가 지정된 리소스에서 다음 작업을 완료하도록 허용합니다.

허용된 작업

- 작업: `ses:SendEmail` and `ses:SendRawEmail`
- Resource: *

정책은 지정된 리소스에 대해 다음 작업을 완료하는 Amazon Cognito 기능을 거부합니다.

거부된 작업

- 작업: `ses:List*`
- Resource: *

이러한 권한을 사용하여 Amazon Cognito에서는 Amazon SES에서 확인 이메일 주소를 사용하여 사용자에게 이메일을 보낼 수 있습니다. Amazon Cognito에서는 사용자 풀 클라이언트 앱에서 암호 등록 또는 재설정과 같은 특정 작업을 수행할 때 사용자에게 이메일을 보냅니다.

IAM 개체(사용자, 그룹, 역할 등)가 서비스 연결 역할을 작성하고 편집하거나 삭제할 수 있도록 권한을 구성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 [서비스 연결 역할 권한](#)을 참조하십시오.

Amazon Cognito에 대한 서비스 연결 역할 생성

서비스 연결 역할은 수동으로 생성할 필요가 없습니다. AWS Management 콘솔, AWS CLI 또는 Amazon Cognito API에서 configure a user pool to use your Amazon SES configuration to handle email delivery을 수행한 후에는 Amazon Cognito에서 서비스 연결 역할을 생성합니다.

이 서비스 연결 역할을 삭제한 다음 다시 생성해야 하는 경우 동일한 프로세스를 사용하여 계정에서 역할을 다시 생성할 수 있습니다. configure a user pool to use your Amazon SES configuration to handle email delivery 시 Amazon Cognito에서 서비스 연결 역할을 다시 생성합니다.

Amazon Cognito에서 이 역할을 생성할 수 있기 전에 사용자 풀을 설정하는 데 사용하는 IAM 권한은 iam:CreateServiceLinkedRole 작업을 포함해야 합니다. IAM에서 권한을 업데이트하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 [IAM 사용자의 권한 변경](#) 항목을 참조하십시오.

Amazon Cognito에 대한 서비스 연결 역할 편집

Amazon Cognito에서는 AmazonCognitoIdpEmailService 서비스 연결 역할을 편집하도록 허용하지 않습니다. 서비스 연결 역할을 생성한 후에는 다양한 개체가 역할을 참조할 수 있기 때문에 역할 이름을 변경할 수 없습니다. 그러나 IAM를 사용하여 역할의 설명을 편집할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 [서비스 연결 역할 편집](#)을 참조하십시오.

Amazon Cognito에 대한 서비스 연결 역할 삭제

서비스 연결 역할이 필요한 기능 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제할 것을 권장합니다. 따라서 적극적으로 모니터링하거나 유지하지 않는 미사용 개체가 없도록 합니다. AmazonCognitoIdpEmailService 역할을 삭제하기 전에 이 역할을 사용하는 각 사용자 풀에 대해 다음 중 하나를 수행해야 합니다.

- 사용자 풀을 삭제합니다.
- 기본 이메일 기능을 사용하도록 사용자 풀의 이메일 설정을 업데이트합니다. 기본 설정은 서비스 연결 역할을 사용하지 않습니다.

이 역할을 사용하는 사용자 풀을 포함하는 각 AWS 리전에서 이러한 작업을 수행해야 합니다.

Note

리소스를 삭제하려 할 때 Amazon Cognito 서비스가 역할을 사용 중이면 삭제에 실패할 수 있습니다. 이 문제가 발생하면 몇 분 기다렸다가 작업을 다시 시도하십시오.

Amazon Cognito 사용자 풀 삭제 방법

1. AWS Management 콘솔에 로그인한 다음 <https://console.aws.amazon.com/cognito>에서 Amazon Cognito 콘솔을 엽니다.
2. Manage User Pools(사용자 풀 관리)를 선택합니다.
3. 사용자 풀 페이지에서 삭제할 사용자 풀을 선택합니다.
4. 풀 삭제를 선택합니다.
5. 사용자 풀 삭제 창에서 **delete**를 입력하고 풀 삭제를 선택합니다.

기본 이메일 기능을 사용하기 위한 Amazon Cognito 사용자 풀 업데이트 방법

1. AWS Management 콘솔에 로그인한 다음 <https://console.aws.amazon.com/cognito>에서 Amazon Cognito 콘솔을 엽니다.
2. Manage User Pools(사용자 풀 관리)를 선택합니다.
3. 사용자 풀 페이지에서 업데이트할 사용자 풀을 선택합니다.
4. 왼쪽의 탐색 메뉴에서 메시지 사용자 지정을 선택합니다.
5. Amazon SES 구성을 통해 이메일을 보내시겠습니까? 아래에서 아니오 - Cognito (기본값)을 선택합니다.
6. 이메일 계정 옵션 설정을 마치면 변경 사항 저장을 선택합니다.

IAM을 사용하여 서비스 연결 역할을 수동으로 삭제하려면

IAM 콘솔, AWS CLI 또는 AWS API를 사용하여 AmazonCognitoIdpEmailService 서비스 연결 역할을 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 [서비스에 연결 역할 삭제](#) 단원을 참조하십시오.

Amazon Cognito 서비스 연결 역할에 대해 지원되는 리전

Amazon Cognito는 서비스가 제공되는 모든 리전에서 서비스 연결 역할을 사용하도록 지원합니다. 자세한 내용은 [AWS Regions and Endpoints](#) 단원을 참조하십시오.

Amazon Cognito 문서 기록

다음 표에서는 본 Amazon Cognito 릴리스 관련 문서에 대해 설명합니다.

- 원래 API 버전:

Amazon Cognito 사용자 풀: 2016년 4월 18일

Amazon Cognito 연동 자격 증명: 2014년 6월 30일

Amazon Cognito 동기화: 2014년 6월 30일

- 최신 설명서 업데이트: 2019년 4월 8일

변경 사항	설명	날짜
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. 자세한 내용은 Amazon Cognito 사용자 풀에 대한 이메일 설정 (p. 182) 단원을 참조하십시오.	2019년 4월 8일
태그 지정 지원	Amazon Cognito 리소스 태그 지정 (p. 331) 에 관한 정보가 추가됨.	2019년 3월 26일
사용자 지정 도메인의 인증서 변경	사용자 지정 도메인을 사용하여 Amazon Cognito 호스팅 UI를 호스팅하는 경우 필요에 따라 이 도메인의 SSL 인증서를 변경할 수 있습니다. 자세한 내용은 사용자 지정 도메인의 SSL 인증서 변경 (p. 81) 단원을 참조하십시오.	2018년 12월 19일
신규 한도	각 사용자가 소속될 수 있는 최대 그룹 수에 새로운 한도가 추가되었습니다. 자세한 내용은 Amazon Cognito의 제한 값 (p. 311) 단원을 참조하십시오.	2018년 12월 14일
업데이트된 한도	사용자 풀의 소프트 한도가 업데이트되었습니다. 자세한 내용은 Amazon Cognito의 제한 값 (p. 311) 단원을 참조하십시오.	2018년 12월 11일
이메일 주소 및 전화번호 확인에 대한 설명서 업데이트	사용자가 앱에 가입할 때 이메일 또는 전화 확인을 요구하도록 사용자 풀을 구성하는 방법에 대한 정보를 추가했습니다. 자세한 내용은 가입 시 연락처 정보 확인 (p. 157) 단원을 참조하십시오.	2018년 11월 20일

변경 사항	설명	날짜
이메일 테스트에 대한 설명서 업데이트	앱을 테스트할 때 Amazon Cognito에서 이메일을 시작하는 방법에 대한 지침을 추가했습니다. 자세한 내용은 앱 테스트 중 이메일 보내기 (p. 162) 단원을 참조하십시오.	2018년 11월 13일
Amazon Cognito 고급 보안	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 UI에 대한 사용자 지정 도메인	Amazon Cognito 사용자 풀에서 호스팅한 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 개발자 안내서 업데이트	최상위 "Amazon Cognito란" 및 "Amazon Cognito 시작하기"가 추가되었습니다. 또한 일반적인 시나리오가 추가되고 사용자 풀 TOC가 재구성되었습니다. 새로운 "시작하기" 단원이 추가되었습니다.	2018년 6월 4일
Amazon Cognito Lambda 마이그레이션 트리거	Lambda 마이그레이션 트리거 기능을 다루는 페이지가 추가되었습니다.	2018년 2월 8일
Amazon Cognito 고급 보안 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
Amazon Pinpoint 통합	Amazon Pinpoint를 사용하도록 기능을 추가하여 Amazon Cognito 사용자 풀 앱에 대한 분석을 제공하고 Amazon Pinpoint 캠페인에 대한 사용자 데이터를 보강합니다. 자세한 내용은 Amazon Cognito 사용자 풀을 통해 Amazon Pinpoint 분석 사용 (p. 155) 단원을 참조하십시오.	2017년 9월 26일

변경 사항	설명	날짜
Amazon Cognito 사용자 풀의 자격 증명 연동 및 기본 제공 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 준수와 관련된 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. 자세한 내용은 사용자 풀에 그룹 추가 (p. 166) 및 역할 기반 액세스 제어 (p. 238) 단원을 참조하십시오.	2016년 12월 15일
설명서 업데이트	개발자 인증 자격 증명(자격 증명 풀) (p. 267) 에서 iOS 코드 예제가 업데이트되었습니다.	2016년 11월 18일
설명서 업데이트	사용자 계정의 확인 흐름에 대한 정보가 추가되었습니다. 자세한 내용은 사용자 계정 가입 및 확인 (p. 156) 단원을 참조하십시오.	2016년 11월 9일
사용자 계정 생성 기능	Amazon Cognito 콘솔 및 API를 통해 사용자 계정을 생성하기 위해 관리 기능이 추가되었습니다. 자세한 내용은 관리자로서 사용자 계정 생성 (p. 163) 단원을 참조하십시오.	2016년 10월 6일
설명서 업데이트	사용자 풀로 AWS Lambda 트리거를 사용하는 방법을 보여주는 예제가 업데이트되었습니다. 자세한 내용은 Lambda 트리거를 사용하여 사용자 풀 워크플로우 사용자 지정 (p. 118) 단원을 참조하십시오.	2016년 9월 27일
사용자 가져오기 기능	Cognito 사용자 풀에 대한 대량 가져오기 기능이 추가되었습니다. 이 기능을 사용하여 기존 자격 증명 공급자에서 Amazon Cognito 사용자 풀로 사용자를 마이그레이션합니다. 자세한 내용은 CSV 파일에서 사용자 풀로 사용자 가져오기 (p. 172) 단원을 참조하십시오.	2016년 9월 1일

변경 사항	설명	날짜
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. 자세한 내용은 Amazon Cognito 사용자 풀 (p. 12) 단원을 참조하십시오.	2016년 7월 28일
SAML 지원	SAML 2.0(Security Assertion Markup Language 2.0)을 통해 자격 증명 공급자의 인증에 대한 지원이 추가되었습니다. 자세한 내용은 SAML 자격 증명 공급자(자격 증명 풀) (p. 265) 단원을 참조하십시오.	2016년 6월 23일
CloudTrail 통합	AWS CloudTrail와의 통합이 추가되었습니다. 자세한 내용은 AWS CloudTrail을 사용하여 Amazon Cognito API 호출 로깅 (p. 328) 단원을 참조하십시오.	2016년 2월 18일
Lambda와 이벤트 통합	Amazon Cognito에서 중요한 이벤트에 반응하여 AWS Lambda 함수를 실행할 수 있습니다. 자세한 내용은 Amazon Cognito 이벤트 (p. 308) 단원을 참조하십시오.	2015년 4월 9일
Amazon Kinesis에 대한 데이터 스트림	데이터 스트림에 제어 및 통찰력을 제공합니다. 자세한 내용은 Amazon Cognito 스트림 (p. 306) 단원을 참조하십시오.	2015년 3월 4일
푸시 동기화	자동 푸시 동기화에 대한 지원을 활성화합니다. 자세한 내용은 Amazon Cognito Sync (p. 281) 단원을 참조하십시오.	2014년 11월 6일
OpenID Connect 지원	OpenID Connect 공급자에 대한 지원을 활성화합니다. 자세한 내용은 자격 증명 풀(연동 자격 증명) 외부 자격 증명 공급자 (p. 249) 단원을 참조하십시오.	2014년 10월 23일
개발자 인증 자격 증명 지원이 추가됨	Amazon Cognito에서 자체 인증 및 자격 증명 관리 시스템이 있는 개발자를 자격 증명 공급자로 처리할 수 있습니다. 자세한 내용은 개발자 인증 자격 증명(자격 증명 풀) (p. 267) 단원을 참조하십시오.	2014년 9월 29일
Amazon Cognito 일반 가용성		2014년 7월 10일