aws

개발자 가이드

Amazon Cognito



Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Cognito: 개발자 가이드

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon의 상표 및 트레이드 드레스는 Amazon 외 제품 또는 서비스와 함께, Amazon 브랜드 이미지를 떨어뜨리거나 고객에게 혼동을 일으킬 수 있는 방식으로 사용할 수 없습니다. Amazon이 소유하지 않은 기타 모든 상표는 Amazon과 제휴 관계이거나 관련이 있거나 후원 관계 여부에 관계없이 해당 소유자의 자산입니다.

Table of Contents

Amazon Cognito란 무엇입니까? 1
Amazon Cognito의 어떤 부분이 필요합니까? 1
Amazon Cognito로 빌드할 수 있는 항목 2
사용자 풀 3
ID 풀 4
Amazon Cognito의 기능 5
사용자 풀 5
ID 풀 8
Amazon Cognito 사용자 풀과 자격 증명 풀 비교 9
Amazon Cognito 시작하기 13
리전별 가용성 14
Amazon Cognito 요금 14
용어 및 개념 14
일반 15
사용자 풀 17
ID 풀 21
시작하기 AWS 22
에 가입 AWS 계정 22
시작하기 23
첫 번째 로그인 24
첫 번째 앱 및 사용자 풀 25
기타 애플리케이션 옵션 27
React SPA 예시 28
Flutter 모바일 앱 예시 31
다음 단계 34
소셜 제공업체 추가 35
SAML IdP 추가 42
추가 시작 옵션 45
설계 계획 47
접근 방식 선택 47
일반적인 Amazon Cognito 시나리오 49
사용자 풀로 인증 49
서버 측 리소스에 액세스 50
API Gateway 및 Lambda를 사용하여 리소스 액세스 51

사용자 풀 및 자격 증명 풀을 사용하여 AWS 서비스에 액세스 51
타사로 인증하고 자격 증명 풀을 사용하여 AWS 서비스에 액세스 52
Amazon Cognito를 사용하여 AWS AppSync 리소스에 액세스 53
Machine-to-machine(M2M) 54
단계적 인증 54
로그인 표면 설정 56
관리형 로그인 56
관리형 로그인 현지화 58
약관 문서 59
Amazon Cognito 콘솔을 사용하여 관리형 로그인 설정 60
를 사용하여 관리형 로그인 설정 AWS Amplify 61
로그인 페이지 보기 62
인증 페이지 사용자 지정 63
관리형 로그인 및 호스팅 UI에 대해 알아야 할 사항 64
도메인 구성 66
브랜딩 및 사용자 지정 81
관리형 로그인으로 마이그레이션 93
사용자 인증 95
특성 95
가입 95
로그인 97
관리형 로그인 98
위협 방지 99
사용자 지정 사용자 경험 99
모니터링 및 분석 99
Amazon Cognito 자격 증명 풀 통합 99
사용자 풀 기능 플랜 100
기능 플랜 선택 101
플랜별 기능 102
Essentials 플랜 기능 105
Plus 플랜 기능 109
부적격 기능 끄기 112
인증 113
앱 클라이언트 114
인증 흐름 구현 124
알아야 할 것들 126

인증 흐름 예제 ... 129
관리형 로그인 인증 ... 132
SDK 인증 ... 135
인증 흐름 ... 141
SDK 권한 부여 모델 ... 165
타사 IdP 로그인 ... 179
리소스 액세스 ... 248
Verified Permissions를 사용하여 리소스에 액세스 ... 249
API Gateway 리소스 액세스 ... 251
자격 증명 풀을 사용하여 AWS 리소스에 액세스 ... 252
사용자 풀 토큰 ... 258
ID 토큰 ... 259
ID 토큰 헤더 ... 260
ID 토큰 기본 페이로드 ... 261
ID 토큰 서명 ... 268
액세스 토큰 ... 268
액세스 토큰 헤더 ... 269
액세스 토큰 기본 페이로드 ... 270
Machine-to-machine 액세스 토큰 페이로드 ... 272
액세스 토큰 서명 ... 272
토큰을 안전하게 구문 분석 ... 273
새로 고침 토큰 ... 274
토큰 새로 고침 ... 274
새로 고침 토큰 교체 ... 275
API 및 SDK 토큰 새로 고침 ... 277
OAuth 토큰 새로 고침 ... 278
새로 고침 토큰 철회 ... 278
토큰 철회 ... 280
토큰 취소 사용 ... 281
토큰 취소 ... 281
JSON 웹 토큰 확인 ... 282
사전 조건 ... 283
aws-jwt-verify를 사용한 토큰 검증 ... 283
토큰 이해 및 검사 ... 284
캐싱 토큰 ... 289
Amazon API Gateway를 통해 머신 간 액세스 토큰 캐싱 ... 290

M2M 및 범위 293
API 권한 부여 293
기계 간(M2M) 인증 294
범위에 대한 정보 297
리소스 서버에 대한 정보 299
리소스 바인딩 303
사용자 관리 305
사용자 관리 305
사용자 가입 허용 306
사용자 계정 가입 및 확인 309
관리자로 사용자 생성 334
사용자 풀에 그룹 추가 342
사용자 관리 및 검색 344
암호 352
사용자 풀로 사용자 가져오기 357
속성 383
사용자 풀 및 앱 클라이언트 업데이트 400
변경할 수 없는 설정 400
SMS 구성 402
AWS SDK AWS CDK또는 REST API를 사용하여 사용자 풀 업데이트 402
사용자 풀 사용자 지정 405
Lambda 트리거 405
알아야 할 것들 407
트리거 설정 410
사용자 풀 Lambda 트리거 이벤트 410
사용자 풀 Lambda 트리거 공통 파라미터 411
클라이언트 메타데이터 412
작업별 소스 트리거 415
함수별 트리거 소스 422
사전 가입 427
게시 확인 434
사전 인증 437
사후 인증 441
인바운드 페더레이션 445
사용자 지정 문제 455
사전 토큰 생성 476

사용자 마이그레이션 499
사용자 지정 메시지 505
사용자 지정 발신자 513
메시징 및 분석 528
이메일 설정 529
SMS 메시지 설정 543
Amazon Pinpoint 분석 사용 558
사용자 풀 보호 564
보안 모범 사례 565
네트워크 수준에서 사용자 풀 보호 565
SMS 메시지 침해로부터 보호 566
퍼블릭 인증 이해 566
클라이언트 보안 암호로 기밀 클라이언트 보호 569
기타 보안 암호 보호 569
사용자 풀 관리 최소 권한 570
토큰 보안 및 확인 573
신뢰할 수 있는 ID 제공업체 확인 573
범위가 사용자 프로필에 대한 액세스에 미치는 영향 이해 574
사용자 속성에 대한 입력 삭제 574
MFA 추가 575
알아야 할 것들 576
사용자 MFA 기본 설정 579
MFA 로직 581
MFA 구성 585
SMS 및 이메일 MFA 587
TOTP 소프트웨어 토큰 MFA 588
ACR 및 AMR 클레임을 사용한 인증 수준 595
인증 수준 및 방법 596
ACR 레벨 이름 구성 598
기능 계획 요구 사항 600
단계적 인증에 대한 책임 600
디바이스 작업 601
기억된 디바이스 설정 601
디바이스 키 가져오기 603
디바이스로 로그인 604
디바이스 보기, 업데이트 및 지우기 605

위협 방지 606
위협 방지를 위한 고려 사항 및 제한 사항 609
사용자 풀에서 위협 방지 켜기 610
위협 방지 시행 개념 610
위협 방지 옵션 611
사전 조건 612
위협 방지 구성 612
손상된 보안 인증 618
적응형 인증 620
데이터 수집 630
AWS WAF 웹 ACLs 634
AWS WAF 웹 ACLs 및 Amazon Cognito에 대해 알아야 할 사항 634
사용자 지정 블록 응답에 대한 Content-Security-Policy 636
사용자 지정 블록 응답의 상태 코드 637
웹 ACL을 사용자 풀과 연결 637
AWS WAF 웹 ACLs 테스트 및 로깅 640
대소문자 구분 640
삭제 방지 642
사용자 공개 관리 643
인증 및 사용자 생성 작업 644
암호 재설정 작업 648
확인 작업 649
다중 테넌시 모범 사례 650
테넌트당 사용자 풀 652
테넌트당 앱 클라이언트 653
테넌트당 사용자 풀 그룹 655
테넌트당 사용자 지정 속성 657
테넌트별 사용자 지정 범위 659
리소스 예시 662
멀티 테넌시 보안 권장 사항 663
앱과 통합 664
를 사용한 인증 AWS Amplify 665
Amplify로 사용자 인터페이스(UI) 생성 666
AWS SDKs 사용한 인증 667
인증 작동 방식 667
관리형 로그인 인증 668

SDK 인증 670
타사 ID 제공업체 인증 673
ID 풀 인증 676
AWS SDKs 작업 680
Amazon Verified Permissions를 통한 권한 부여 681
Verified Permissions를 사용한 API 권한 부여 682
Amazon Cognito 사용자에 대한 정책 예시 685
사용자 풀 엔드포인트 참조 688
관리형 로그인 엔드포인트 689
Login 엔드포인트 691
Logout 엔드포인트 693
페더레이션 엔드포인트 697
OIDC 발급자 699
권한 부여 엔드포인트 701
Token 엔드포인트 713
userInfo 엔드포인트 725
취소 엔드포인트 728
SAML idpresponse 엔드포인트 730
OAuth 2.0 권한 부여 733
PKCE 사용 734
Amazon Cognito가 PKCE를 사용하는 방법 734
관리형 로그인 및 페더레이션 오류 응답 736
배포 및 운영 739
할당량 739
API 요청률 할당량 이해 740
API 요청률 할당량 관리 743
사용자 풀 요청률 할당량 750
ID 풀 요청률 할당량 764
리소스 수 및 크기에 대한 할당량 766
다중 리전 복제 772
다중 리전 복제에 대해 알아야 할 사항 772
다중 리전 복제의 제한 사항 773
다중 리전 복제 구성 773
보조 리전에서 지원되는 API 작업 776
다중 리전 사용자 풀의 장애 조치 778
리소스에 태그 지정 781

지원되는 리소스 781
태그 제한 사항 782
콘솔을 사용하여 태그 관리 782
AWS CLI 예제 782
API 작업 786
CloudFormation 태그 지정 786
문제 해결 787
사용자 지정 도메인 구성 오류 787
Invalid refresh token 오류 790
페더레이션의 잘못된 SAML 응답 오류 790
관리형 로그인 사용자는 MFA 요소를 선택할 수 없습니다. 793
이메일/SMS를 통해 암호 재설정 코드를 수신할 수 없음 794
확인되지 않은 복구 속성으로 암호 재설정이 실패합니다. Could not reset password for the account, please contact support or try again 795
SECRET_HASH 오류 795
Amazon Cognito 콘솔에서 새 사용자 풀에 대한 기본 구성 선택 796
추가 문제 해결 리소스 797
코드 예제 798
Amazon Cognito 자격 증명 800
기본 사항 800
시나리오 825
Amazon Cognito 자격 증명 공급자 827
기본 사항 829
시나리오 996
Amazon Cognito Sync 1183
기본 사항 1184
AWS 자격 증명의 자격 증명 교환 1186
Amazon Cognito 자격 증명 풀 1186
ID 풀 구성 1188
자격 증명 풀 인증 흐름 1195
IAM 역할 1205
보안 모범 사례 1222
액세스 제어에 속성 사용 1226
역할 기반 액세스 제어 사용 1234
자격 증명 얻기 1241
보안 인증 사용 1246

타사 ID 제공업체 1249
개발자 인증 자격 증명 1279
자격 증명 전환 1296
자격 증명 풀 시작하기 1299
Amazon Cognito에서 자격 증명 풀 생성 1299
SDK 설정 1301
자격 증명 공급자 통합 1301
자격 증명 얻기 1302
애플리케이션 예시 1302
보안 1335
데이터 보호 1336
데이터 암호화 1336
전송 중 암호화 1346
ID 및 액세스 관리 1346
대상 1347
ID를 통한 인증 1347
정책을 사용하여 액세스 관리 1348
Amazon Cognito에서 IAM을 사용하는 방법 1350
ID 기반 정책 예시 1357
문제 해결 1362
서비스 연결 역할 사용 1364
로깅 및 모니터링 1368
비용 모니터링 1369
사용자 풀 로그 내보내기 1371
할당량 및 사용량 모니터링 1382
CloudTrail 로그 1398
AWS PrivateLink 1425
AWS PrivateLink 통합을 위한 인증 흐름 1426
의 운영 모드 AWS PrivateLink 1427
고려 사항 1428
리소스 제어 정책을 사용하여 액세스 제어 1433
인터페이스 엔드포인트 생성 1434
엔드포인트 정책을 생성 1434
자격 증명 기반 정책 생성 1436
규정 준수 확인 1438
복원력 1439

리전 데이터 고려 사항 1439
인프라 보안 1440
구성 및 취약성 분석 1440
AWS 관리형 정책 1440
정책 업데이트 1442
레거시 1445
Amazon Cognito Sync 1445
Amazon Cognito Sync 가용성 변경 1446
Amazon Cognito Sync 시작하기 1447
클라이언트 간 데이터 동기화 1448
이벤트 콜백 처리 1458
푸시 동기화 구현 1475
Amazon Cognito Sync 스트림 구현 1484
Amazon Cognito Events를 사용하여 워크플로 사용자 지정 1486
Amazon Cognito Sync를 자격 증명 풀과 함께 사용 1491
데이터 집합 관리 1491
데이터 대량 게시 1492
푸시 동기화 활성화 1493
Amazon Cognito 스트림 설정 1493
Amazon Cognito 이벤트 설정 1494
호스팅 UI(클래식) 브랜딩 1494
클래식 브랜딩으로 사용자 지정 로고 지정 1494
클래식 브랜딩에서 CSS 사용자 지정 항목 지정 1495
에서 클래식 브랜딩을 사용하여 호스팅 UI 사용자 지정 AWS Management Console 1499
사용자 풀 API 및에서 클래식 브랜딩을 사용하여 호스팅 UI 사용자 지정 AWS CLI 1500
문서 기록 1501
..... mdxxiii

Amazon Cognito란 무엇입니까?

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

주제

- Amazon Cognito의 어떤 부분이 필요합니까?
- Amazon Cognito로 빌드할 수 있는 항목
- 사용자 풀
- ID 풀
- Amazon Cognito의 기능
- Amazon Cognito 사용자 풀과 자격 증명 풀 비교
- Amazon Cognito 시작하기
- 리전별 가용성
- Amazon Cognito 요금
- 일반적인 Amazon Cognito 용어 및 개념
- 시작하기 AWS

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

Amazon Cognito의 어떤 부분이 필요합니까?

Amazon Cognito는 두 가지 질문에 답합니다. 애플리케이션이 수행해야 하는 작업을 기반으로를 선택합니다.

이 사용자는 누구입니까? (인증)

> 사용자 풀을 사용합니다. 사용자 풀은 OpenID Connect(OIDC) 표준에서 가져와 사용자를 인증하고 앱 또는 API에 JSON 웹 토큰(JWTs)을 발급하는 사용자 디렉터리입니다. 사용자는 Google 및 Apple과 같은 소셜 공급자 또는 엔터프라이즈 SAML 및 OIDC 공급자를 통해 직접 로그인할 수 있습니다. 가입 및 로그인이 필요한 대부분의 애플리케이션은 여기에서 시작됩니다.

이 사용자는 어떤 AWS 리소스에 도달할 수 있나요? (권한 부여)

자격 증명 풀을 사용합니다. 자격 증명 풀은 앱이 사용자를 대신하여 Amazon S3 및 Amazon DynamoDB와 같은 서비스를 호출할 수 있도록 신뢰할 수 있는 토큰(사용자 풀, 소셜 공급자 또는 SAML 또는 OIDC 공급자)을 임시의 제한된 권한 자격 AWS 증명으로 교환합니다. 자격 증명 풀은 토큰 없이 로그인하지 않은 게스트 사용자에게 제한된 권한의 자격 증명을 발급할 수도 있습니다.

둘 다 필요합니까?

앱이 사용자를 로그인하고 사용자에게 직접 AWS 리소스 액세스 권한을 부여해야 할 때 함께 사용합니다. 사용자 풀로 인증한 다음 사용자 풀 토큰을 자격 증명 풀과 교환하여 자격 증명을 얻 AWS 습니다.

두 구성 요소 모두 다른 구성 요소가 필요하지 않습니다. 사용자 풀은 자격 증명 풀 없이 자체 백엔드에 토큰을 발급할 수 있습니다. 자격 증명 풀은 사용자 풀 없이도 이미 보유한 공급자의 자격 증명을 브로커링할 수 있습니다. 두 구성 요소의 차이점에 대한 자세한 내용은 섹션을 참조하세요Amazon Cognito 사용자 풀과 자격 증명 풀 비교. 빌드를 시작하려면 섹션을 참조하세요Amazon Cognito 시작하기.

Amazon Cognito로 빌드할 수 있는 항목

대부분의 Amazon Cognito 애플리케이션은 몇 가지 일반적인 형태에 속합니다. 이 목록을 사용하여 현재 구축 중인 것과 가장 가까운 목록을 찾은 다음 전체 연습의 링크를 따르세요.

앱에 사용자 로그인

사용자 풀은 사용자를 인증하고 앱 또는 API가 확인하는 JSON 웹 토큰(JWTs)을 발급합니다. 이는 가입 및 로그인이 필요한 대부분의 앱의 시작점입니다. 사용자 풀로 인증 섹션을 참조하세요.

자체 API 또는 백엔드 리소스 보호

사용자 풀의 액세스 토큰을 사용하여 서버 측 리소스에 대한 요청을 승인하거나 Amazon API Gateway가 Lambda 함수 또는 자체 API에 대한 토큰 및 게이트 액세스를 검증하도록 합니다. 사용자 풀 토큰을 사용하여 백엔드 리소스에 액세스 및 사용자 풀로 API Gateway 및 Lambda를 사용하여 리소스 액세스 섹션을 참조하세요.

사용자에게 AWS 서비스에 대한 임시 액세스 권한 부여

앱이 사용자를 대신하여 Amazon S3 및 Amazon DynamoDB와 같은 서비스를 호출할 수 있도록 사용자 풀 토큰을 자격 증명 풀과 임시로 제한된 권한의 AWS 자격 증명으로 교환합니다. 사용자 풀 및 자격 증명 풀을 사용하여 AWS 서비스에 액세스 섹션을 참조하세요.

사용자가 Google, Apple 또는 엔터프라이즈 디렉터리로 로그인하도록 허용

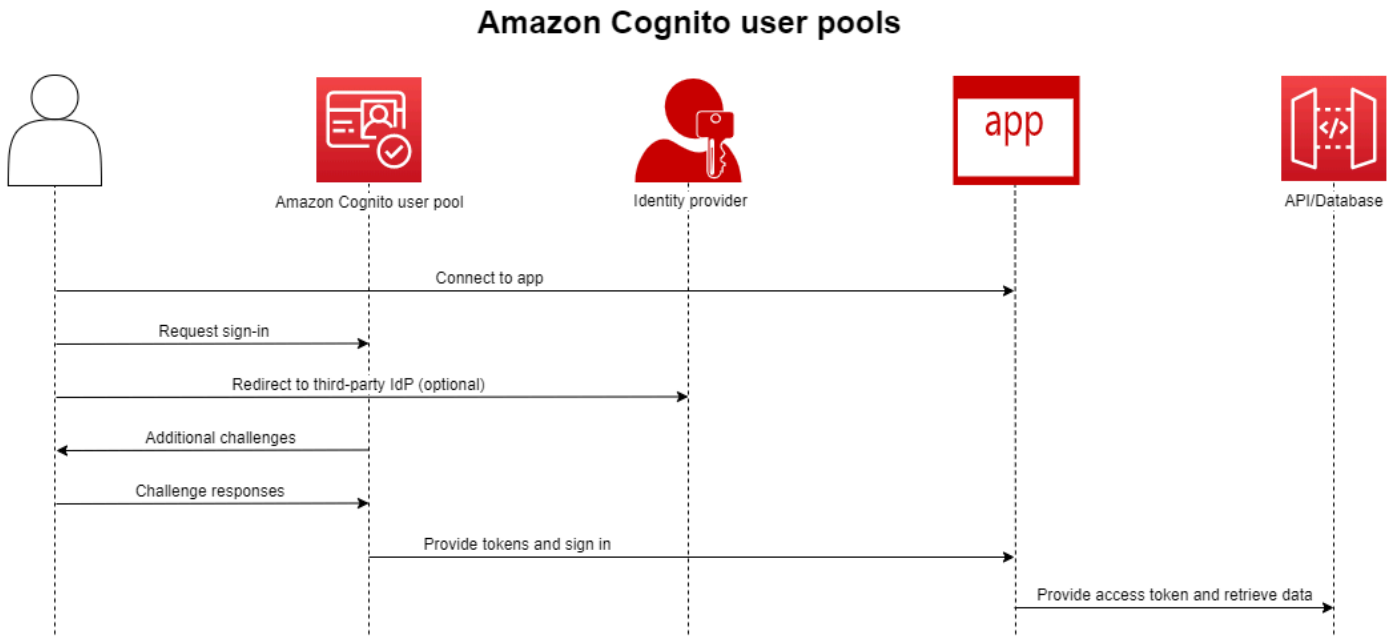
소셜 또는 SAML/OIDC 공급자를 페더레이션합니다. 사용자 풀은 로그인을 브로커링하고 앱에 표준 토큰 세트를 발급하거나 자격 증명 풀이 외부 토큰을 자격 AWS 증명으로 직접 교환합니다. 타사로 인증하고 자격 증명 풀을 사용하여 AWS 서비스에 액세스 섹션을 참조하세요.

인간 사용자 없이 애플리케이션 또는 디바이스 권한 부여

M2M(MachineMachine-to-machine) 권한 부여는 루프에 사람이 없는 상태에서 `client_credentials` 권한 부여를 통해 자체적으로 인증하는 앱, 서비스 또는 디바이스에 액세스 토큰을 발급합니다. M2M(Machinemachine-to-machine) 애플리케이션 권한 부여 섹션을 참조하세요.

아키텍처 다이어그램이 포함된 전체 시나리오 세트는 섹션을 참조하세요일반적인 Amazon Cognito 시나리오.

사용자 풀

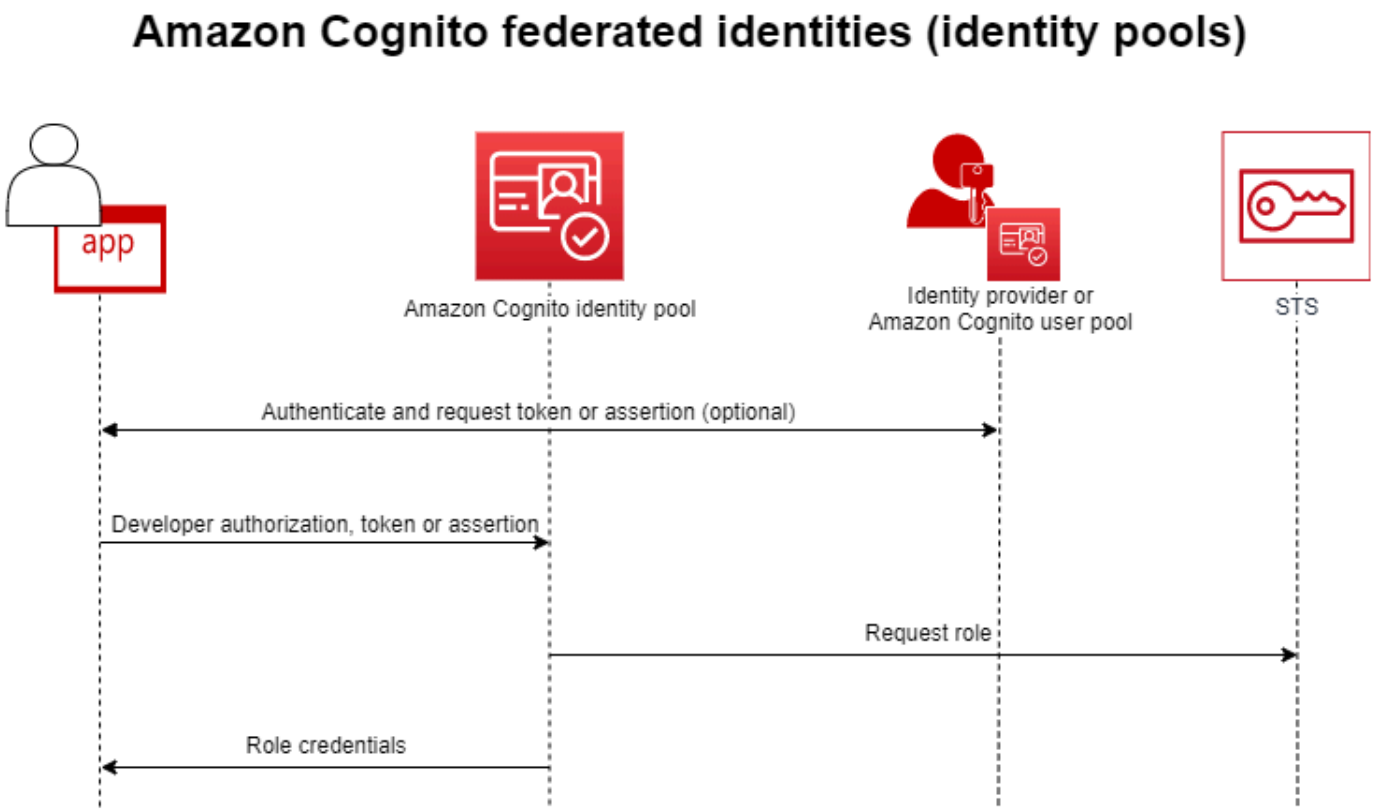


앱 또는 API에 사용자를 인증하고 권한을 부여하려는 경우 사용자 풀을 생성하세요. 사용자 풀은 셀프 서비스 및 관리자 기반 사용자 생성, 관리, 인증을 모두 갖춘 사용자 디렉터리입니다. 사용자 풀은 독립 디렉터리 및 OIDC ID 제공업체(idP)일 수 있고, 직원 및 고객 자격 증명 서드 파티 공급자에 대한 중간 서비스 공급자(SP)일 수 있습니다. 사용자 풀을 갖춘 SAML 2.0 및 OIDC IdP에 있는 조직의 직원 ID에 대한 Single Sign-On(SSO)을 앱에 제공할 수 있습니다. 퍼블릭 OAuth 2.0 ID 스토어인 Amazon,

Google, Apple, Facebook의 조직 고객 ID를 위한 SSO를 앱에 제공할 수도 있습니다. 고객 ID 및 액세스 관리(CIAM)에 대한 자세한 내용은 CIAM이란 무엇인가요? 섹션을 참조하십시오.

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

ID 풀



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

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

Amazon Cognito 사용자 풀과 자격 증명 풀을 함께 사용

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

1. 앱 사용자는 사용자 풀을 통해 로그인하고 OAuth 2.0 토큰을 받습니다.
2. 앱은 사용자 풀 토큰을 자격 증명 풀과 교환하여 AWS APIs 및 AWS Command Line Interface ()와 함께 사용할 수 있는 임시 AWS 자격 증명을 찾습니다AWS CLI.
3. 앱은 사용자에게 자격 증명 세션을 할당하고 Amazon S3 및 Amazon DynamoDB와 AWS 서비스 같은에 대한 권한 있는 액세스를 제공합니다.

자격 증명 풀 및 사용자 풀을 사용하는 추가 예제는 섹션을 참조하세요일반적인 Amazon Cognito 시나리오.

Amazon Cognito에서 공동 책임 모델의 클라우드 자체 보안 의무는 SOC 1-3, PCI DSS, ISO 27001을 준수하며, HIPA-BAA 적격입니다. Amazon Cognito의 클라우드에서 보안을 SOC 1-3, ISO 27001 및 HIPAA-BAA를 준수하도록 설계할 수 있지만 PCI DSS를 준수하도록 설계할 수는 없습니다. 자세한 내용은 범위 내AWS 서비스를 참조하세요. 또한 리전 데이터 고려 사항 섹션도 참조하세요.

Amazon Cognito의 기능

사용자 풀

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

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

Amazon Cognito 사용자 풀은 타사 IdPs에서 토큰과 어설션을 수락하고 사용자 속성을 앱에 발급하는 JWT로 수집합니다. Amazon Cognito가 IdP와의 상호 작용을 처리하여 해당 클레임을 중앙 토큰 형식에 매핑하는 동안 하나의 JWT 세트에서 앱을 표준화할 수 있습니다.

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

- Amazon Cognito 사용자 풀 API를 호출하여 사용자를 인증, 권한 부여, 관리하는 자체 웹 프런트엔드를 구현합니다.
- 사용자 다중 인증(MFA)을 설정합니다. Amazon Cognito는 시간 기반 일회용 암호(TOTP) 및 SMS 메시지 MFA를 지원합니다.
- 악의적으로 제어되는 사용자 계정의 액세스로부터 보호합니다.
- 자체 사용자 지정 다단계 인증 흐름을 생성합니다.
- 다른 디렉터리에서 사용자를 찾아 Amazon Cognito로 마이그레이션합니다.

또한 Amazon Cognito 사용자 풀은 IdP에 대한 서비스 공급자(SP)이자 앱에 대한 IdP로서 이중의 역할을 수행할 수 있습니다. Amazon Cognito 사용자 풀은 Facebook과 Google 같은 소비자 IdP나 Okta와 Active Directory Federation Services(ADFS) 같은 직원 IdP에 연결할 수 있습니다.

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

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

Amazon Cognito 사용자 풀의 기능

기능	설명
OIDC identity provider	Issue ID tokens to authenticate users
Authorization server	Issue access tokens to authorize user access to APIs
SAML 2.0 service provider	Transform SAML assertions into ID and access tokens
OIDC relying party	Transform OIDC tokens into ID and access tokens

Social provider relying party	Transform ID tokens from Apple, Facebook, Amazon, or Google to your own ID and access tokens
Authentication frontend service	Sign up, manage, and authenticate users with managed login
API support for your own UI	Create, manage and authenticate users through authentication API requests in supported AWS SDKs[1]
Multi-factor authentication	Use SMS messages, TOTPs, or your user's device as an additional authentication factor[1]
Security monitoring & response	Secure against malicious activity and insecure passwords[1]
Customize authentication flows	Build your own authentication mechanism, or add custom steps to existing flows[2]
Groups	Create logical groupings of users, and a hierarchy of IAM role claims when you pass tokens to identity pools
Customize tokens	Customize your ID and access tokens with new, modified, and suppressed claims
Customize user attributes	Assign values to user attributes and add your own custom attributes

[1] 페더레이션 사용자는 기능을 사용할 수 없습니다.

[2] 페더레이션 및 관리형 로그인 사용자는 기능을 사용할 수 없습니다.

사용자 풀에 대한 자세한 내용은 Amazon Cognito 콘솔에서 새 애플리케이션 생성 및 Amazon Cognito 사용자 풀 API 참조를 참조하세요.

ID 풀

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

인증된 자격 증명을 보완하기 위해 IdP 인증 없이 AWS 액세스를 승인하도록 자격 증명 풀을 구성할 수도 있습니다. 개발자 인증 자격 증명을 사용하여 사용자 지정 인증 증명을 제공할 수 있습니다. 인증되지 않은 ID를 사용하여 게스트 사용자에게 임시 AWS 자격 증명을 부여할 수 있습니다.

ID 풀을 사용하면 AWS 계정에서 두 가지 방법으로 IAM 정책과 통합할 수 있습니다. 이 두 기능은 함께 사용하거나 개별적으로 사용할 수 있습니다.

역할 기반 액세스 제어

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

액세스 제어용 속성

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

Amazon Cognito 자격 증명 풀의 기능

기능	설명
Amazon Cognito user pool relying party	Exchange an ID token from your user pool for web identity credentials from AWS STS

SAML 2.0 service provider	Exchange SAML assertions for web identity credentials from AWS STS
OIDC relying party	Exchange OIDC tokens for web identity credentials from AWS STS
Social provider relying party	Exchange OAuth tokens from Amazon, Facebook, Google, Apple, and Twitter for web identity credentials from AWS STS
Custom relying party	With AWS credentials, exchange claims in any format for web identity credentials from AWS STS
Unauthenticated access	Issue limited-access web identity credentials from AWS STS without authentication
Role-based access control	Choose an IAM role for your authenticated user based on their claims, and configure your roles to only be assumed in the context of your identity pool
Attribute-based access control	Convert claims into principal tags for your AWS STS temporary session, and use IAM policies to filter resource access based on principal tags

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

Amazon Cognito 사용자 풀과 자격 증명 풀 비교

기능	설명	사용자 풀	ID 풀
OIDC identity provider	Issue OIDC ID tokens to authenticate app users	✓	

User directory	Store user profiles for authentication	✓	
Authorize API access	Issue access tokens to authorize user access to APIs (including user profile self-service API operations), databases, and other resources that accept OAuth scopes	✓	
IAM web identity authorization	Generate tokens that you can exchange with AWS STS for temporary AWS credentials		✓
SAML 2.0 service provider & OIDC identity provider	Issue customized OIDC tokens based on claims from a SAML 2.0 identity provider	✓	
OIDC relying party & OIDC identity provider	Issue customized OIDC tokens based on claims from an OIDC identity provider	✓	
OAuth 2.0 relying party & OIDC identity provider	Issue customized OIDC tokens based on scopes from OAuth 2.0 social providers like Apple and Google	✓	

SAML 2.0 service provider & credentials broker	Issue temporary AWS credentials based on claims from a SAML 2.0 identity provider		✓
OIDC relying party & credentials broker	Issue temporary AWS credentials based on claims from an OIDC identity provider		✓
Social provider relying party & credentials broker	Issue temporary AWS credentials based on JSON web tokens from developer applications with social providers like Apple and Google		✓
Amazon Cognito user pool relying party & credentials broker	Issue temporary AWS credentials based on JSON web tokens from Amazon Cognito user pools		✓
Custom relying party & credentials broker	Issue temporary AWS credentials to arbitrary identities, authorize d by developer IAM credentials		✓
Authentication frontend service	Sign up, manage, and authenticate users with managed login	✓	

API support for your own authentication UI	Create, manage and authenticate users through API requests in supported AWS SDKs[1]	✓
MFA	Use SMS messages, TOTPs, or your user's device as an additional authentication factor[1]	✓
Security monitoring & response	Protect against malicious activity and insecure passwords[1]	✓
Customize authentication flows	Build your own authentication mechanism, or add custom steps to existing flows[1]	✓
User groups	Create logical groupings of users, and a hierarchy of IAM role claims when you pass tokens to identity pools	✓
Customize tokens	Customize your ID and access tokens with new, modified, and suppressed claims and scopes	✓
AWS WAF web ACLs	Monitor and control requests to your authentication front end with AWS WAF	✓

Customize user attributes	Assign values to user attributes and add your own custom attributes	✓		
Unauthenticated access	Issue limited-access web identity credentials from AWS STS without authentication			✓
Role-based access control	Choose an IAM role for your authenticated user based on their claims, and configure your role trust to limit access to web identity users			✓
Attribute-based access control	Transform user claims into principal tags for your AWS STS temporary session, and use IAM policies to filter resource access based on principal tags			✓

[1] 페더레이션 사용자는 기능을 사용할 수 없습니다.

Amazon Cognito 시작하기

사용자 풀 애플리케이션의 예는 Amazon Cognito 콘솔에서 새 애플리케이션 생성 섹션을 참조하세요.

ID 풀에 대한 소개는 Amazon Cognito ID 풀 시작 섹션을 참조하세요.

사용자 풀 및 ID 풀을 사용한 설정 환경 안내에 대한 링크는 Amazon Cognito용 설정 옵션 안내 섹션을 참조하세요.

AWS SDK를 시작하려면 AWS 개발자 도구를 참조하세요. Amazon Cognito와 관련된 개발자 리소스는 Amazon Cognito 개발자 리소스를 참조하세요.

Amazon Cognito를 사용하려면 AWS 계정이 있어야 합니다. 자세한 내용은 시작하기 AWS 섹션을 참조하세요.

리전별 가용성

Amazon Cognito는 전 세계 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 영역을 통해는 Amazon Cognito를 비롯한 서비스에 매우 높은 수준의 가용성과 중복성을 제공하는 동시에 지연 시간을 최소화 AWS 할 수 있습니다.

Amazon Cognito를 현재에서 사용할 수 있는지 확인하려면 AWS 리전별 서비스를 AWS 리전참조하세요.

리전 API 서비스 엔드포인트에 대한 자세한 내용은의 AWS 리전 및 엔드포인트를 참조하세요Amazon Web Services 일반 참조.

각 리전에서 사용할 수 있는 가용 영역 수에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

Amazon Cognito 요금

Amazon Cognito 요금에 대한 자세한 내용은 Amazon Cognito 요금을 참조하세요.

일반적인 Amazon Cognito 용어 및 개념

Amazon Cognito는 웹 및 모바일 앱에 대한 자격 증명을 제공합니다. ID 및 액세스 관리에서 일반적인 용어를 사용하고 이를 기반으로 구축됩니다. 범용 ID 및 액세스 조건에 대한 많은 가이드를 사용할 수 있습니다. 다음은 몇 가지 예시입니다.

- IDPro 지식 본문의 용어
- AWS 자격 증명 서비스
- NIST CSRC의 용어집

다음 목록은 Amazon Cognito에 고유하거나 Amazon Cognito에 특정 컨텍스트가 있는 용어를 설명합니다.

주제

- 일반
- 사용자 풀
- ID 풀

일반

이 목록의 용어는 Amazon Cognito에만 국한되지 않으며 ID 및 액세스 관리 실무자 사이에서 널리 인식됩니다. 다음은 용어의 전체 목록이 아니며 이 안내서의 특정 Amazon Cognito 컨텍스트에 대한 안내서입니다.

액세스 토큰

정보 시스템에 액세스하기 위한 엔터티의 권한 부여에 대한 정보가 포함된 JSON 웹 토큰(JWT)입니다.

앱, 애플리케이션

일반적으로 모바일 애플리케이션입니다. 이 가이드에서 앱은 Amazon Cognito에 연결하는 웹 애플리케이션 또는 모바일 앱의 약어인 경우가 많습니다.

ABAC(속성 기반 액세스 제어)

앱이 사용자의 직함 또는 부서 등 사용자 속성을 기반으로 리소스에 대한 액세스를 결정하는 모델입니다. ABAC를 적용하는 Amazon Cognito 도구에는 사용자 풀의 ID 토큰과 ID 풀의 보안 주체 태그가 포함됩니다.

인증

정보 시스템에 액세스하기 위해 실제 ID를 설정하는 프로세스입니다. Amazon Cognito는 타사 ID 공급자의 인증 증명을 수락하며 소프트웨어 애플리케이션에 대한 인증 공급자 역할을 합니다.

권한 부여

리소스에 권한을 부여하는 프로세스입니다. 사용자 풀 액세스 토큰에는 애플리케이션이 사용자 및 시스템이 리소스에 액세스하도록 허용하는 데 사용할 수 있는 정보가 포함되어 있습니다.

권한 부여 서버

JSON 웹 토큰을 생성하는 OAuth 또는 OpenID Connect(OIDC) 시스템입니다. Amazon Cognito 사용자 풀 관리형 인증 서버는 사용자 풀의 두 인증 및 권한 부여 방법의 권한 부여 서버 구성 요소입니다. 사용자 풀은 SDK 인증에서 API 문제-응답 흐름도 지원합니다.

기밀 앱, 서버 측 앱

사용자가 애플리케이션 서버의 코드와 보안 암호에 대한 액세스를 사용하여 원격으로 연결하는 애플리케이션입니다. 이는 일반적으로 웹 애플리케이션입니다.

ID 제공업체(IdP)

사용자 ID를 저장하고 확인하는 서비스입니다. Amazon Cognito는 외부 제공업체로부터 인증을 요청할 수 있으며 앱에 대한 IdP가 될 수 있습니다.

JSON 웹 토큰(JWT)

인증된 사용자에 대한 클레임이 포함된 JSON 형식 문서입니다. ID 토큰은 사용자를 인증하고, 액세스 토큰은 사용자에게 권한을 부여하며, 새로 고침 토큰은 자격 증명을 업데이트합니다. Amazon Cognito는 외부 공급자로부터 토큰을 수신하고 앱 또는에 토큰을 발급합니다 AWS STS.

기계 간(M2M) 인증

웹 서버 애플리케이션 계층과 같은 사용자 상호 작용이 불가능한 머신 엔터티에 대한 API 엔드포인트에 대한 요청을 승인하는 프로세스입니다. 사용자 풀은 액세스 토큰의 OAuth 2.0 범위를 사용하여 클라이언트 자격 증명 권한 부여에서 M2M 권한 부여를 제공합니다.

다중 인증(MFA)

사용자가 사용자 이름과 암호를 제공한 후 추가 인증을 제공해야 하는 요구 사항입니다. Amazon Cognito 사용자 풀에는 로컬 사용자를 위한 MFA 기능이 있습니다.

OAuth 2.0(소셜) 제공업체

JWT 액세스 및 새로 고침 토큰을 제공하는 사용자 풀 또는 ID 풀에 대한 IdP입니다. Amazon Cognito 사용자 풀은 사용자가 인증한 후 소셜 제공업체와의 상호 작용을 자동화합니다.

OpenID Connect(OIDC) 제공업체

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

패스키, WebAuthn

사용자 디바이스의 암호화 키 또는 패스키가 인증 증명을 제공하는 인증의 한 형태입니다. 사용자는 하드웨어 또는 소프트웨어 인증자에 생체 인식 또는 PIN 코드 메커니즘이 있는지 확인합니다. 패스키는 피싱에 강하며 특정 웹 사이트/앱에 바인딩되어 암호 없는 안전한 환경을 제공합니다. Amazon Cognito 사용자 풀은 패스키로 로그인을 지원합니다.

암호 없음

사용자가 암호를 입력할 필요가 없는 인증의 한 형태입니다. 암호 없는 로그인 방법에는 이메일 주소와 전화번호 및 패스키로 전송된 일회용 암호(OTP)가 포함됩니다. Amazon Cognito 사용자 풀은 OTP 및 패스키를 사용한 로그인을 지원합니다.

퍼블릭 앱

코드가 로컬에 저장되고 보안 암호에 대한 액세스 권한이 없는 디바이스에 자체 포함된 애플리케이션입니다. 일반적으로 모바일 앱입니다.

리소스 서버

액세스 제어 기능이 있는 API입니다. Amazon Cognito 사용자 풀은 리소스 서버를 사용하여 API와 상호 작용하기 위한 구성을 정의하는 구성 요소를 설명합니다.

역할 기반 액세스 제어(RBAC)

사용자의 기능 지정을 기반으로 액세스 권한을 부여하는 모델입니다. Amazon Cognito ID 풀은 IAM 역할 간의 차별화를 통해 RBAC를 구현합니다.

서비스 제공업체(SP), 신뢰 당사자(RP)

IdP를 사용하여 사용자가 신뢰할 수 있다고 주장하는 애플리케이션입니다. Amazon Cognito는 외부 IdP에 대한 SP 역할을 하며 앱 기반 SP에 대한 IdP 역할을 합니다.

SAML 공급자

사용자가 Amazon Cognito에 전달하는 디지털 서명 어설션 문서를 생성하는 사용자 풀 또는 ID 풀에 대한 IdP입니다.

범용 고유 식별자(UUID)

개체에 적용되는 128비트 레이블입니다. Amazon Cognito UUIDs는 사용자 풀 또는 자격 증명 풀마다 고유하지만 RFC UUID를 포함한 특정 UUID 형식을 준수하지 않습니다. Amazon Cognito 식별자의 형식을 엄격하게 검증해서는 안 됩니다.

사용자 디렉터리

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

사용자 풀

이 가이드의 다음 목록에 용어가 표시되면 사용자 풀의 특정 기능 또는 구성을 나타냅니다.

적응형 인증

잠재적 악성 활동을 감지하고 사용자 프로필에 추가 보안을 적용하는 고급 보안 기능입니다.

앱 클라이언트

사용자 풀에 대한 설정을 하나의 앱에 대한 IdP로 정의하는 구성 요소입니다.

콜백 URL, 리디렉션 URI, 반환 URL

앱 클라이언트의 설정과 사용자 풀의 권한 부여 서버에 대한 요청의 파라미터입니다. 콜백 URL은 앱에서 인증된 사용자의 초기 대상입니다.

선택 기반 인증

사용자 풀을 사용한 API 인증의 한 형태로, 각 사용자는 로그인을 선택할 수 있습니다. MFA를 사용하거나 사용하지 않는 사용자 이름 및 암호, 패스키 로그인 또는 이메일 또는 SMS 메시지 일회용 암호를 사용한 암호 없는 로그인을 선택할 수 있습니다. 애플리케이션은 인증 옵션 목록을 요청하거나 기본 옵션을 선언하여 사용자의 선택 프로세스를 구성할 수 있습니다.

클라이언트 기반 인증과 비교합니다.

클라이언트 기반 인증

사용자 풀 API 및 AWS SDKs. 선언적 인증에서는 애플리케이션이 사용자가 수행해야 하는 로그인 유형을 독립적으로 결정하고 해당 유형을 미리 요청합니다.

선택 기반 인증과 비교합니다.

손상된 보안 인증

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

확인

새로운 사용자가 로그인할 수 있도록 하는 전제 조건이 충족되었는지 확인하는 프로세스입니다. 확인은 일반적으로 이메일 주소 또는 전화번호 확인을 통해 이루어집니다.

사용자 지정 인증

Lambda 트리거를 사용한 인증 프로세스의 확장으로 추가 사용자 문제 및 응답을 정의합니다.

디바이스 인증

신뢰할 수 있는 장치의 ID를 사용하여 로그인하는 방식으로 MFA를 대체하는 인증 프로세스입니다.

도메인, 사용자 풀 도메인

AWS에서 관리형 로그인 페이지를 호스팅하는 웹 도메인입니다. 소유한 도메인에서 DNS를 설정하거나 AWS 소유한 도메인에서 식별 하위 도메인 접두사를 사용할 수 있습니다.

Essentials 플랜

사용자 풀의 최신 개발이 포함된 기능 플랜입니다. Essentials 플랜에는 Plus 플랜의 자동 학습 보안 기능이 포함되지 않습니다.

외부 제공업체, 타사 제공업체

사용자 풀과 신뢰 관계가 있는 IdP입니다. 사용자 풀은 외부 공급자와 애플리케이션 간의 중간 엔터티 역할을 하여 SAML 2.0, OIDC 및 소셜 공급자를 통한 인증 프로세스를 관리합니다. 사용자 풀은 외부 공급자 인증 결과를 단일 IdP로 통합하여 애플리케이션이 단일 OIDC 신뢰 당사자 라이브러리로 많은 사용자를 처리할 수 있도록 합니다.

기능 플랜

사용자 풀에 대해 선택할 수 있는 기능 그룹입니다. 기능 플랜은 AWS 청구서에서 비용이 다릅니다. 새 사용자 풀은 기본적으로 Essentials 플랜으로 설정됩니다.

현재 플랜

- Lite 플랜
- Essentials 플랜
- Plus 플랜

페더레이션 사용자, 외부 사용자

외부 제공업체에서 인증한 사용자 풀의 사용자입니다.

호스트형 UI(클래식), 호스트형 UI 페이지

사용자 풀 도메인에서 인증 프런트 엔드, 신뢰 당사자 및 ID 공급자 서비스의 초기 버전입니다. 호스트형 UI는 기본 기능 세트를 포함하며 디자인이 간소화되었습니다. 로고 이미지 파일과 미리 결정된 CSS 스타일 세트가 있는 파일을 업로드하여 호스트형 UI 브랜딩을 적용할 수 있습니다. 관리형 로그인과 비교합니다.

Lambda 트리거

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

로컬 사용자

외부 제공업체에 대한 인증으로 생성되지 않은 사용자 풀 사용자 디렉터리의 사용자 프로필입니다.

연결된 사용자

ID가 로컬 사용자와 병합된 외부 제공업체의 사용자입니다.

Lite 플랜

사용자 풀로 원래 시작한 기능이 포함된 기능 플랜입니다. Lite 플랜에는 Essentials 플랜의 새 기능 또는 Plus 플랜의 자동 학습 보안 기능이 포함되지 않습니다.

관리형 권한 부여 서버, 호스트형 UI 권한 부여 서버, 권한 부여 서버

사용자 풀 도메인의 IdP 및 앱과 상호 작용하기 위한 서비스를 호스팅하는 관리형 로그인의 구성 요소입니다. 호스트형 UI는 제공하는 사용자 대화형 기능의 관리형 로그인과 다르지만 권한 부여 서버 기능은 동일합니다.

관리형 로그인, 관리형 로그인 페이지

사용자 인증을 위해 서비스를 호스팅하는 사용자 풀 도메인의 웹 페이지 세트입니다. 이러한 서비스에는 IdP로 작동하기 위한 함수, 타사 IdP의 신뢰 당사자, 사용자 대화형 인증 UI의 서버가 포함됩니다. 사용자 풀에 대한 도메인을 설정하면 Amazon Cognito는 모든 관리형 로그인 페이지를 온라인으로 가져옵니다.

애플리케이션은 사용자의 브라우저를 호출하고 가입, 로그인, 암호 관리 및 기타 인증 작업을 위해 관리형 로그인 UI로 전달하는 OIDC 라이브러리를 가져옵니다. 인증 후 OIDC 라이브러리는 인증 요청의 결과를 처리할 수 있습니다.

관리형 로그인 인증

사용자 대화형 브라우저 페이지 또는 HTTPS API 요청을 사용하여 사용자 풀 도메인의 서비스로 로그인합니다. 애플리케이션은 OpenID Connect(OIDC) 라이브러리를 사용하여 관리형 로그인 인증을 처리합니다. 이 프로세스에는 외부 공급자와의 로그인, 대화형 관리형 로그인 페이지로 로컬 사용자 로그인, M2M 권한 부여가 포함됩니다. 클래식 호스트형 UI를 사용한 인증도 이 용어에 속합니다.

AWS SDK 인증과 비교합니다.

Plus 플랜

사용자 풀의 최신 개발 및 고급 보안 기능이 포함된 기능 플랜입니다.

SDK 인증, AWS SDK 인증

AWS SDK를 사용하여 애플리케이션에 추가할 수 있는 인증 및 권한 부여 API 작업 세트입니다. 이 인증 모델에는 사용자 지정 로그인 메커니즘이 필요합니다. API는 로컬 사용자 및 연결된 사용자에 로그인할 수 있습니다.

관리형 로그인 인증과 비교합니다.

위협 방지, 고급 보안 기능

사용자 풀에서 위협 보호는 인증 및 권한 부여 메커니즘에 대한 위협을 완화하도록 설계된 기술을 말합니다. 적응형 인증, 손상된 자격 증명 탐지 및 IP 주소 차단 목록은 위협 방지 범주에 속합니다.

토큰 사용자 지정

런타임 시 사용자의 ID 또는 액세스 토큰을 수정하는 사전 토큰 생성 Lambda 트리거의 결과입니다.

사용자 풀, Amazon Cognito ID 제공업체, **`cognito-idp`**, Amazon Cognito 사용자 풀

OIDC IdP로 작업하는 애플리케이션에 대한 인증 및 권한 부여 서비스가 포함된 AWS 리소스입니다. IdPs

Verification(확인)

사용자가 이메일 주소 또는 전화번호를 소유하고 있는지 확인하는 프로세스입니다. 사용자 풀은 새 이메일 주소 또는 전화번호를 입력한 사용자에게 코드를 보냅니다. Amazon Cognito에 코드를 제출하면 메시지 대상의 소유권을 확인하고 사용자 풀에서 추가 메시지를 받을 수 있습니다. 확인 섹션도 참조하세요.

사용자 프로필, 사용자 계정

사용자 디렉터리의 사용자 항목입니다. 타사 IdP의 사용자를 포함한 모든 사용자는 사용자 풀에 프로필이 있습니다.

ID 풀

이 가이드의 다음 목록에 용어가 표시되면 ID 풀의 특정 기능 또는 구성을 나타냅니다.

액세스 제어용 속성

ID 풀에서 속성 기반 액세스 제어 구현입니다. ID 풀은 사용자 속성을 사용자 자격 증명에 태그로 적용합니다.

기본(클래식) 인증

사용자 자격 증명에 대한 요청을 사용자 지정할 수 있는 인증 프로세스입니다.

개발자 인증 자격 증명

개발자 자격 증명을 사용하여 ID 풀 사용자 자격 증명을 승인하는 인증 프로세스입니다.

개발자 자격 증명

ID 풀 관리자의 IAM API 키입니다.

향상된 인증

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

ID

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

ID 풀, Amazon Cognito 페더레이션 ID, Amazon Cognito ID, **`cognito-identity`**

임시 AWS 자격 증명을 사용하는 애플리케이션에 대한 인증 및 권한 부여 서비스가 포함된 AWS 리소스입니다.

인증되지 않은 자격 증명

ID 풀 IdP를 사용하여 로그인하지 않은 사용자입니다. 사용자가 인증하기 전에 단일 IAM 역할에 대해 제한된 사용자 자격 증명을 생성하도록 허용할 수 있습니다.

사용자 보안 인증

사용자가 자격 증명 풀 인증 후 받는 임시 AWS API 키입니다.

시작하기 AWS

Amazon Cognito 작업을 시작하기 전에 몇 가지 필수 AWS 리소스를 설정합니다. 이미에 로그인할 수 있는 경우이 섹션을 건너뛸 AWS 계정수 있습니다. 가입 및 AWS 자격 증명으로 로그인에 대한 정보를 찾고 있다면 계속 읽어보세요. 충분한 AWS Identity and Access Management (IAM) 권한을 가진 자격 증명이 있으면 사용자 풀 및 자격 증명 풀을 시작할 수 있습니다.

에 가입 AWS 계정

를 시작하려면이 AWS필요합니다 AWS 계정. 두 가지 방법으로 가입할 수 있습니다 AWS. 계정 구성 및 콘텐츠 저장 위치를 완벽하게 제어하거나 HIPAA 또는 FedRAMP와 같은 규제 대상 워크로드가 있는 경우 가입 AWS (고급)을 사용합니다. 신규 사용자이고 이미 소유한 로그인을 사용하여에 로그인 AWS 하고에서 사전 구성된 기본값으로 AWS 환경을 생성하려면 가입을 AWS AWS 리전사용합니다.

Amazon Cognito 시작하기

이 장에서는 첫 번째 사용자 풀을 생성하고 첫 번째 사용자에 로그인하는 사용자 풀 인증을 시작합니다. Amazon Cognito는 사용자의 자격 증명을 임시 AWS 자격 증명으로 교환하는 자격 증명 풀도 제공합니다. 필요한 경우 섹션을 참조하세요 AWS 자격 증명의 자격 증명 교환. 어떤 구성 요소가 필요한지 잘 모르는 경우 섹션을 참조하세요the section called “Amazon Cognito의 어떤 부분이 필요합니까?”.

> **Note**
>
> 이 장의 연습 및 빠른 시작을 완료하려면 AWS 계정이 필요합니다. 없으면 the section called “시작하기 AWS” 섹션을 참조하세요.

인증 및 액세스 제어가 필요한 애플리케이션이 있습니다. Single Sign-On(SSO)을 위한 OpenID Connect(OIDC) 프레임워크 내에서 작업할 수 있습니다. Amazon Cognito에는 AWS SDK를 사용하여 애플리케이션의 인증 로직을 처리하고 클라이언트에서 브라우저를 호출하여 관리형 권한 부여 서버에 액세스하기 위한 도구가 있습니다.

Amazon Cognito 콘솔은 선호하는 애플리케이션 프레임워크의 보기에서 사용자 풀을 생성하는 과정을 안내합니다. 여기에서 외부 소셜 또는 SAML 2.0 ID 제공업체(idP)를 통한 페더레이션 로그인과 같은 기능을 계속 추가할 수 있습니다. Amazon Cognito 콘솔의 애플리케이션 모델은 프로젝트에 OIDC 라이브러리를 추가하고 브라우저를 호출하는 데 중점을 둡니다.

기능 세트를 확장하고 Amazon Cognito의 더 많은 구성 요소를 통합하는 작업을 수행하면서 사용자 풀로 수행할 수 있는 모든 작업에 대한 전체 설명은 Amazon Cognito 사용자 풀 장을 참조하세요.

이 장과 Amazon Cognito 콘솔의 예제는 Amazon Cognito 사용자 풀과 애플리케이션 리소스의 기본적인 통합을 보여줍니다. 나중에 사용 가능한 옵션을 더 많이 사용하도록 사용자 풀을 조정할 수 있습니다. 그런 다음 애플리케이션을 업데이트하여 새로운 기능을 채택하고 IdP와 상호 작용할 수 있습니다.

관리형 로그인 페이지를 사용하지 않으려면 AWS SDK 또는를 사용하여 사용자 지정 인증 인터페이스를 사용하여 애플리케이션을 생성할 수 있습니다 AWS Amplify. 이러한 방식으로 빌드하는 애플리케이션은 사용자 풀 API와 상호 작용하며 로컬 사용자 인증에만 적합합니다. 기타 애플리케이션 옵션에서 이 인증 모델에 대해 계속 알아봅니다.

첫 번째 로그인: end-to-end 연습

이 페이지는 연결되는 세부 절차를 대체하지 않는 맵입니다. 단일 단계로 넘어가기 전에 전체 여정을 볼 수 있도록 설정 단계가 연결되는 방법을 보여줍니다. 번호가 매겨진 각 단계는 신뢰할 수 있는 절차에 연결됩니다.

> **Note**
>
> 이 연습에서는 사용자를 로그인하는 애플리케이션의 가장 일반적인 시작점인 관리형 로그인이 있는 사용자 풀을 사용합니다. 관리형 로그인은 시각적 브랜딩 편집기가 있는 현재 Amazon Cognito 호스팅 로그인 UI이며, 이전의 클래식 호스팅 UI는 더 간단한 대안입니다. 사용자 풀 도메인에서 브랜딩 버전으로 둘 중 하나를 선택하면 도메인을 사용하는 모든 앱 클라이언트에 해당 선택이 적용됩니다. 관리형 로그인에는 Essentials 또는 Plus 기능 플랜이 필요합니다. Lite 플랜에서는 클래식 호스팅 UI가 유일한 호스팅 옵션입니다. 차이점은 the section called "관리형 로그인" 및 단원을 참조하십시오the section called "사용자 풀 기능 플랜". 목표가 기존 자격 증명 소스의 임시 AWS 자격 증명인 경우 로 시작합니다the section called "자격 증명 풀 시작하기".

빈 계정에서 검증된 토큰으로

1. 사용자 풀을 생성합니다. 사용자 풀은 사용자 디렉터리와 토큰을 발급하는 자격 증명 공급자입니다. the section called "첫 번째 앱 및 사용자 풀" 섹션을 참조하세요.
2. 도메인을 추가합니다. 관리형 로그인 페이지는 Amazon Cognito 접두사 도메인 또는 사용자 지정 도메인과 같은 사용자 풀 도메인에서 제공됩니다. 도메인이 없으면 사용자를 보낼 호스팅 로그인 페이지가 없습니다. the section called "도메인 구성" 섹션을 참조하세요.
3. 앱 클라이언트를 생성합니다. 앱 클라이언트는 사용자 풀에 대한 애플리케이션을 나타냅니다. 허용된 콜백 URLs 및 로그인 방법과 같이 로그인을 제어하는 설정이 있습니다. 브랜딩 버전(관리형 로그인 또는 클래식 호스팅 UI)은 앱 클라이언트가 아닌 도메인에 설정됩니다. the section called "앱 클라이언트" 섹션을 참조하세요.
4. 사용자를 관리형 로그인으로 전송하여 로그인합니다. 도메인과 앱 클라이언트가 있는 경우 애플리케이션은 사용자를 관리형 로그인 페이지로 리디렉션하여 가입하고 로그인합니다. Amazon Cognito는 권한 부여 코드를 사용하여 사용자를 콜백 URL로 반환합니다. the section called "관리형 로그인" 섹션을 참조하세요.
5. 코드를 토큰으로 교환합니다. 애플리케이션은 토큰 엔드포인트의 권한 부여 코드를 ID 토큰, 액세스 토큰 및 새로 고침 토큰으로 교환합니다. the section called "Token 엔드포인트" 섹션을 참조하세요.

6. 토큰을 검증합니다. 애플리케이션(또는 API)은 토큰을 신뢰하기 전에 토큰의 서명과 클레임을 확인합니다. the section called “JSON 웹 토큰 확인” 섹션을 참조하세요.

이때 로그인한 사용자와 검증된 토큰이 있습니다. 가장 일반적인 다음 단계인 자체 API 또는 백엔드에 대해 해당 토큰을 승인하려면 섹션을 참조하세요the section called “리소스 액세스”. 대신 앱에서 직접 Amazon S3와 같은 AWS 서비스를 호출하기 위해 사용자에게 임시 AWS 자격 증명을 부여해야 하는 경우 로 계속 진행합니다the section called “Amazon Cognito 자격 증명 풀”.

Amazon Cognito 콘솔에서 새 애플리케이션 생성

사용자 풀은 소프트웨어 애플리케이션에 인증 옵션을 추가합니다. 가장 쉬운 시작 환경을 위해 Amazon Cognito 콘솔로 이동하여 지침을 따르세요. 여기에서 생성 프로세스는 사용자 풀 리소스 설정뿐만 아니라 애플리케이션의 초기 부분을 설정하는 과정을 안내합니다.

시작할 준비가 되면 Amazon Cognito 콘솔로 이동하여 버튼을 선택하여 새 사용자 풀을 생성합니다. 설정 프로세스는 구성 및 프로그래밍 언어 옵션을 안내합니다.

인증 개념을 위한 추가 리소스

- Amazon Cognito 사용자 풀로 인증
- API, OIDC 및 관리형 로그인 페이지 인증 이해
- Amazon Cognito에서 인증이 작동하는 방식
- 웹 및 모바일 앱과 Amazon Cognito 인증 및 권한 부여 통합

애플리케이션에 대한 Amazon Cognito 리소스를 생성하려면

1. Amazon Cognito 콘솔로 이동합니다. Amazon Cognito 리소스를 생성하고 관리할 수 있도록 IAM 보안 주체에 권한을 할당하려면 AWS Amazon Cognito에 대한 관리형 정책 섹션을 참조하세요. `AmazonCognitoPowerUser` 정책은 사용자 풀을 생성하기에 충분합니다.
2. 사용자 풀 메뉴에서 사용자 풀 생성을 선택하거나 5분 이내에 무료로 시작을 선택합니다.
3. 애플리케이션 정의에서 인증 및 권한 부여 서비스를 생성하려는 애플리케이션 시나리오에 가장 적합한 애플리케이션 유형을 선택합니다.
4. 애플리케이션 이름 지정에 설명이 포함된 이름을 입력하거나 기본 이름으로 진행합니다.
5. 사용자 풀을 생성한 후에는 변경할 수 없는 설정을 지원하는 옵션 구성에서 몇 가지 기본 선택을 해야 합니다.

a. 로그인 식별자 옵션에서 사용자가 로그인할 때 사용자를 식별하는 방법을 알려줍니다. 사용자 생성 사용자 이름, 이메일 주소 또는 전화번호를 선호할 수 있습니다. 여러 옵션의 조합을 허용할 수도 있습니다. Amazon Cognito는 관리형 로그인 로그인 양식의 사용자 이름 필드에서 여기에서 구성하는 옵션을 수락합니다.

b. 가입에 필요한 속성에서 사용자가 새 계정에 등록할 때 수집하려는 사용자 정보를 알려주세요. 관리형 로그인 페이지에서 Amazon Cognito는 모든 필수 속성에 대한 프롬프트를 표시합니다.

   로그인 식별자 옵션은 필요한 속성에 영향을 미칩니다. 사용자 이름에는 이메일 또는 SMS 메시지에서 암호 재설정 코드를 수신할 수 있도록 각 사용자에 대한 이메일 또는 전화 속성이 필요합니다. 이메일에는 이메일 속성이 필요하고 전화 번호에는 전화 번호 속성이 필요합니다.

6. 반환 URL 추가에서 사용자가 인증을 완료한 후에 대한 애플리케이션의 리디렉션 경로를 입력합니다. 이 위치는 OpenID Connect(OIDC) 라이브러리를 사용하여 사용자 인증 결과를 처리하는 애플리케이션의 경로여야 합니다. 테스트 애플리케이션의 반환 URL의 예는 `https://localhost:3000/callback`입니다. Amazon Cognito 콘솔의 예제 NodeJS 애플리케이션에서 이 경로는 openid-client를 사용하여 액세스 토큰을 수집하고 사용자 정보에 사용합니다. 리소스를 생성한 후 개발 플랫폼의 예를 찾아볼 수 있습니다.

7. 애플리케이션 생성을 선택합니다. Amazon Cognito는 애플리케이션 유형에 대한 기본 설정을 사용하여 사용자 풀 및 앱 클라이언트를 생성합니다. 초기 리소스를 생성한 후 외부 자격 증명 공급자 및 다중 인증(MFA)과 같은 추가 옵션을 구성할 수 있습니다.

8. 애플리케이션 설정 페이지에서 애플리케이션에 대한 코드 예제를 즉시 가져올 수 있습니다. 새 사용자 풀을 탐색하려면 아래로 스크롤하여 개요로 이동을 선택합니다.

9. 동일한 사용자 풀에 애플리케이션을 더 추가하려면 앱 클라이언트 메뉴로 이동하여 새 앱 클라이언트를 추가합니다. 이렇게 하면 애플리케이션 중심 생성 프로세스가 반복되지만 기존 사용자 풀에 새 앱 클라이언트만 추가됩니다.

이 프로세스로 사용자 풀과 하나 이상의 앱 클라이언트를 생성한 후 관리형 로그인으로 인증 작업 테스트를 시작할 수 있습니다. 이러한 빠른 시작 옵션은 공개 자체 가입이 가능합니다. 콘솔 프로세스를 사용하여 테스트 환경을 생성한 다음 완료된 설계를 프로덕션으로 이동하는 것이 좋습니다. 시간을 내어 Amazon Cognito의 기능을 숙지하세요. 그런 다음 프로덕션 워크로드로 이동하려면 사용자 지정 구성을 만들고 AWS CloudFormation 및와 같은 자동화 도구를 사용하여 배포합니다 AWS Cloud Development Kit (AWS CDK).

Amazon Cognito는 이 프로세스에서 되돌릴 수 없는 몇 가지 기본 구성을 만듭니다. 변경할 수 없는 사용자 풀 설정과 콘솔에서 선택할 수 있는 옵션에 대한 자세한 내용은 사용자 풀 및 앱 클라이언트 구성 업데이트 섹션을 참조하세요.

설정	Effect	변경 방법	추가 정보
클라이언트 보안 암호 (client secret)	인증 요청에 클라이언트 보안 암호 해시가 필요합니다.	기존 웹 애플리케이션 또는 머신 간 애플리케이션 프로파일을 사용하여 새 앱 클라이언트를 생성합니다.	앱 클라이언트를 사용한 애플리케이션별 설정
기본 설정 사용자 이름	사용자 풀은 `preferred_username` 속성을 별칭으로 수락하지 않습니다.	AWS SDK를 사용하여 프로그래밍 방식으로 사용자 풀을 생성합니다.	로그인 속성 사용자 지정
대소문자 구분	사용자 풀 사용자 이름은 대/소문자를 구분하지 않습니다. 예를 들어 `JohnD`는 `johnd`와 동일한 사용자로 간주됩니다.	AWS SDK를 사용하여 프로그래밍 방식으로 사용자 풀을 생성합니다.	사용자 풀 대/소문자 구분

기타 애플리케이션 옵션

Amazon Cognito 인증과 통합하려는 기존 애플리케이션 UI가 있을 수 있습니다. Amazon Cognito 사용자 풀보다 디렉터리 설정이 덜 작동하는 기존 인증 페이지가 있을 수도 있습니다. 다양한 프로그래밍 언어에 대해 Amazon Cognito 통합 AWS SDKs를 사용하여이 유형의 애플리케이션에 인증 구성 요소를 추가하거나 교체할 수 있습니다. 다음 예를 참조하세요.

Amazon Cognito 콘솔에서 이 용도로 사용자 풀을 생성하는 경우 대화형 로그인 페이지와 OpenID Connect(OIDC) 서비스를 호스팅하는 사용자 풀 도메인이 필요하지 않을 수 있습니다. 콘솔에서 사용자 풀을 생성하는 프로세스는 자동으로 도메인을 생성합니다. 사용자 풀의 도메인 탭에서 이 도메인을 삭제할 수 있습니다. 다른 옵션으로는 API 요청 AWS SDKs 및 AWS Amplify CLI의 자동 설정 옵션을

사용하여 애플리케이션을 위한 Amazon Cognito 리소스의 프로그래밍 방식 생성이 있습니다. 자세한 내용은 웹 및 모바일 앱과 Amazon Cognito 인증 및 권한 부여 통합 섹션을 참조하세요.

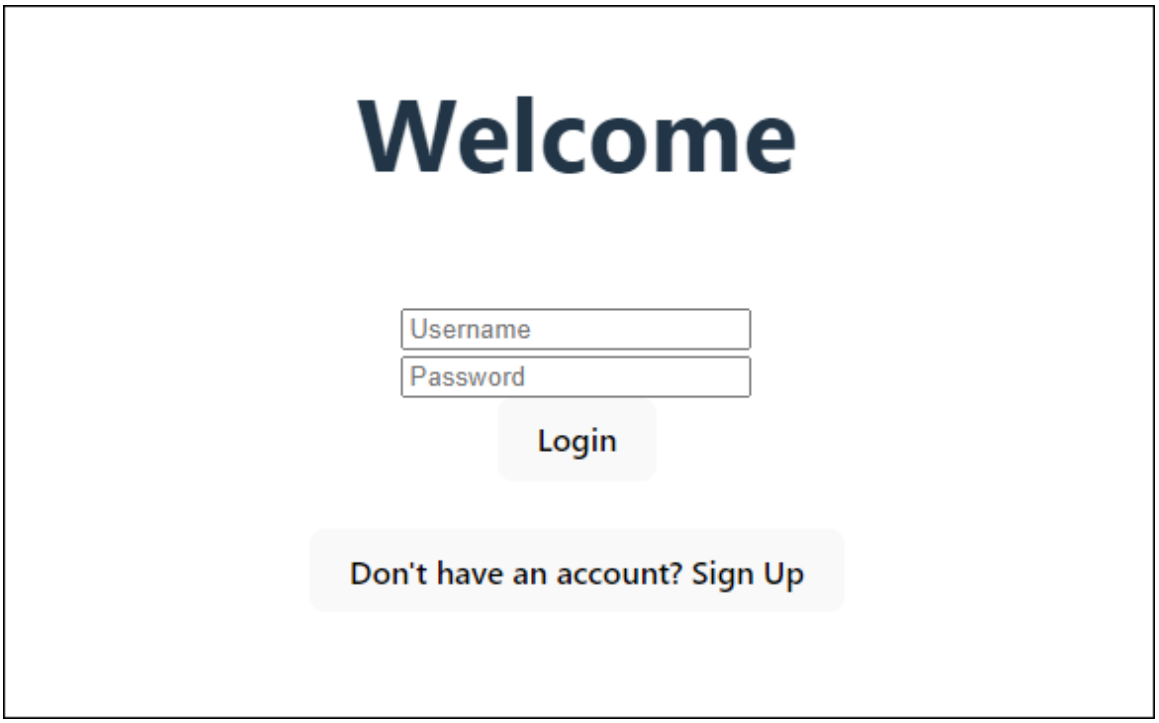
주제

- React 단일 페이지 애플리케이션 예시 설정
- Flutter를 사용하여 예시 Android 앱 설정

React 단일 페이지 애플리케이션 예시 설정

이 자습서에서는 사용자 가입, 확인 및 로그인을 테스트할 수 있는 React 단일 페이지 애플리케이션을 생성합니다. React는 사용자 인터페이스(UI)에 중점을 둔 웹 및 모바일 앱용 JavaScript 기반 라이브러리입니다. 이 예시 애플리케이션은 Amazon Cognito 사용자 풀의 몇 가지 기본 함수를 보여줍니다. React를 사용한 웹 앱 개발 경험이 있는 경우 GitHub에서 예시 앱을 다운로드합니다.

다음 스크린샷은 생성할 애플리케이션의 초기 인증 페이지입니다.



이 애플리케이션을 설정하려면 사용자 풀이 다음 요구 사항을 충족해야 합니다.

- 사용자가 이메일 주소로 로그인할 수 있습니다. Cognito 사용자 풀 로그인 옵션: 이메일.
- 사용자 이름은 대/소문자를 구분하지 않습니다. 사용자 이름 요구 사항: 사용자 이름 대소문자 구분을 선택하지 않아야 합니다.
- 다중 인증(MFA)은 필요하지 않습니다. MFA 적용: 선택적 MFA.

- 사용자 풀은 이메일 메시지를 사용하여 사용자 프로필 확인을 위한 속성을 확인합니다. 확인할 속성: 이메일 메시지 전송, 이메일 주소 확인.
- 이메일은 유일한 필수 속성입니다. 필수 속성: 이메일.
- 사용자는 사용자 풀에 직접 가입할 수 있습니다. 자체 등록: 자체 등록 활성화가 선택되었습니다.
- 초기 앱 클라이언트는 사용자 이름과 암호로 로그인할 수 있는 퍼블릭 클라이언트입니다. 앱 유형: 퍼블릭 클라이언트, 인증 흐름: `ALLOW_USER_PASSWORD_AUTH`.

애플리케이션 만들기

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

1. Node.js가 설치 및 업데이트됩니다.
2. 노드 패키지 관리자(npm)가 설치되고 최소 버전 10.2.3으로 업데이트됩니다.
3. 웹 브라우저의 TCP 포트 5173에서 환경에 액세스할 수 있습니다.

예시 React 웹 애플리케이션을 만드는 방법

1. 개발자 환경에 로그인하고 애플리케이션의 상위 디렉터리로 이동합니다.

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

2. 새 React 서비스를 만듭니다.

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

3. GitHub의 AWS 코드 예제 리포지토리에서 `cognito-developer-guide-react-example` 프로젝트 폴더를 복제합니다.

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

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

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

4. 프로젝트의 `src` 디렉터리로 이동합니다.

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

5. `config.json`을 편집하고 다음 값을 바꿉니다.

   a. 를 AWS 리전 코드`YOUR_AWS_REGION`로 바꿉니다. 예: `us-east-1`

   b. `YOUR_COGNITO_USER_POOL_ID`를 테스트를 위해 지정한 사용자 풀의 ID로 바꿉니다. 예: `us-east-1_EXAMPLE` 사용자 풀은 이전 단계에서 AWS 리전 입력한에 있어야 합니다.

   c. `YOUR_COGNITO_APP_CLIENT_ID`를 테스트를 위해 지정한 앱 클라이언트의 ID로 바꿉니다. 예: `1example23456789` 앱 클라이언트는 이전 단계의 사용자 풀에 있어야 합니다.

6. `localhost` 이외의 IP에서 예제 애플리케이션에 액세스하려면 `package.json`을 편집하고 `"dev": "vite",` 줄을 `"dev": "vite --host 0.0.0.0",`으로 변경합니다.

7. 애플리케이션을 설치합니다.

```
npm install
```

8. 애플리케이션을 시작합니다.

```
npm run dev
```

9. `http://localhost:5173` 또는 `http://[IP address]:5173`의 웹 브라우저에서 애플리케이션에 액세스합니다.

10. 유효한 이메일 주소로 새 사용자를 등록합니다.

11. 이메일 메시지에서 확인 코드를 검색합니다. 애플리케이션에 확인 코드를 입력합니다.

12. 사용자 이름과 암호로 로그인합니다.

Amazon Lightsail을 사용하여 React 개발자 환경 생성

이 애플리케이션을 시작하는 빠른 방법은 Amazon Lightsail을 사용하여 가상 클라우드 서버를 생성하는 것입니다.

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

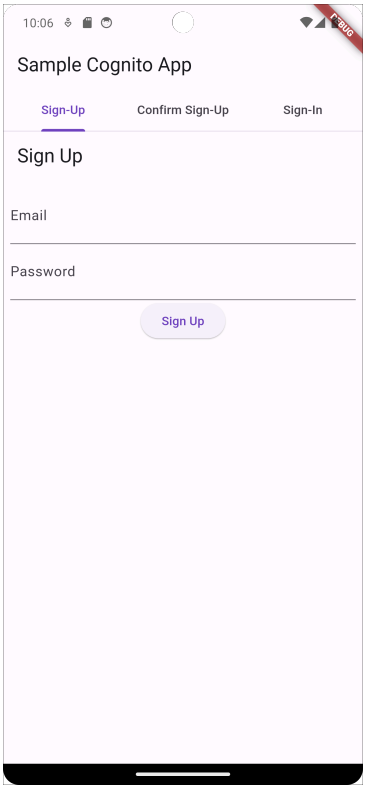
이 예시 애플리케이션의 Lightsail 인스턴스를 생성하는 방법

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

Flutter를 사용하여 예시 Android 앱 설정

이 자습서에서는 Android Studio에서 디바이스를 에뮬레이션하고 사용자 가입, 확인 및 로그인을 테스트할 수 있는 모바일 애플리케이션을 생성합니다. 이 예시 애플리케이션은 Flutter에서 Android용 기본 Amazon Cognito 사용자 풀 모바일 클라이언트를 생성합니다. Flutter를 사용한 모바일 앱 개발 경험이 있는 경우 GitHub에서 예시 앱을 다운로드하세요.

다음 스크린샷은 가상 Android 디바이스에서 실행되는 앱을 보여줍니다.



이 애플리케이션을 설정하려면 사용자 풀이 다음 요구 사항을 충족해야 합니다.

- 사용자가 이메일 주소로 로그인할 수 있습니다. Cognito 사용자 풀 로그인 옵션: 이메일.
- 사용자 이름은 대/소문자를 구분하지 않습니다. 사용자 이름 요구 사항: 사용자 이름 대소문자 구분을 선택하지 않아야 합니다.
- 다중 인증(MFA)은 필요하지 않습니다. MFA 적용: 선택적 MFA.
- 사용자 풀은 이메일 메시지를 사용하여 사용자 프로필 확인을 위한 속성을 확인합니다. 확인할 속성: 이메일 메시지 전송, 이메일 주소 확인.
- 이메일은 유일한 필수 속성입니다. 필수 속성: 이메일.
- 사용자는 사용자 풀에 직접 가입할 수 있습니다. 자체 등록: 자체 등록 활성화가 선택되었습니다.
- 초기 앱 클라이언트는 사용자 이름과 암호로 로그인할 수 있는 퍼블릭 클라이언트입니다. 앱 유형: 퍼블릭 클라이언트, 인증 흐름: `ALLOW_USER_PASSWORD_AUTH`.

애플리케이션 만들기

예시 Android 앱을 만드는 방법

1. Android Studio 및 명령줄 도구를 설치합니다.

2. Android Studio에서 Flutter 플러그인을 설치합니다.
3. 이 예제 앱의 `cognito_flutter_mobile_app` 디렉터리 콘텐츠에서 새 Android Studio 프로젝트를 생성합니다.

   - `assets/config.json`을 편집하고 `<<YOUR USER POOL ID>>` 및 `<< YOUR CLIENT ID>>`를 사용자 풀 및 앱 클라이언트의 ID로 바꿉니다.

4. Flutter를 설치합니다.

   a. PATH 변수에 Flutter를 추가합니다.

   b. 다음 명령을 사용하여 라이선스를 수락합니다.

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

   c. Flutter 환경을 확인하고 누락된 구성 요소를 설치합니다.

   ```
   flutter doctor
   ```

   - 누락된 구성 요소가 있는 경우 `flutter doctor -v`를 실행하여 문제를 해결하는 방법을 알아봅니다.

   d. 새 Flutter 프로젝트의 디렉터리로 변경하고 종속성을 설치합니다.

   - `flutter pub add amazon_cognito_identity_dart_2`를 실행합니다.

   e. `flutter pub add flutter_secure_storage`를 실행합니다.

5. 가상 Android 디바이스를 생성합니다.

   1. Android Studio GUI에서 디바이스 관리자로 새 디바이스를 생성합니다.
   2. CLI에서 `flutter emulators --create --name android-device`를 실행합니다.

6. 가상 Android 디바이스를 시작합니다.

   1. Android Studio GUI에서 가상 디바이스 옆의 시작

      아이콘을 선택합니다.

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

7. 가상 디바이스에서 앱을 시작합니다.

1. Android Studio GUI에서 배포

   아이콘을 선택합니다.

2. CLI에서 `flutter run`를 실행합니다.

8. Android Studio에서 실행 중인 가상 디바이스로 이동합니다.
9. 유효한 이메일 주소로 새 사용자를 등록합니다.
10. 이메일 메시지에서 확인 코드를 검색합니다. 애플리케이션에 확인 코드를 입력합니다.
11. 사용자 이름과 암호로 로그인합니다.

사용자 풀에 더 많은 기능 및 보안 옵션 추가

자습서를 따라 예시 애플리케이션을 완료한 후 사용자 풀 구현 범위를 넓힐 수 있습니다. 또는 테스트 애플리케이션을 생성하지 않은 경우 기본 설정에 따라 새 사용자 풀을 생성합니다. 다른 애플리케이션의 사용자 풀 기능을 사용자 지정하거나 외부 ID 제공업체를 추가할 수 있습니다. Amazon Cognito 사용자 풀을 프로덕션 애플리케이션에 배치하기 위한 전환을 계획할 때 추가 예시 및 자습서를 평가할 수 있습니다.

다음 우선순위가 사용자 풀에서 애플리케이션 보안 옵션을 검사하고 적용하는 경우 Amazon Cognito 사용자 풀을 위한 보안 모범 사례 섹션을 참조하세요.

Amazon Cognito에는 상위 계층에 옵트인할 때 기능 및 보안 옵션을 추가하는 기능 플랜이 있습니다. Lite 플랜으로 시작하고, Essentials 플랜을 사용하여 고급 인증 및 권한 부여 옵션을 추가하고, Plus 플랜을 사용하여 자동화된 이유 보안 가드레일을 추가할 수 있습니다. 자세한 내용은 사용자 풀 기능 플랜 섹션을 참조하세요.

다음은 몇 가지 추가 Amazon Cognito 사용자 풀 기능입니다.

- 관리형 로그인 페이지에 브랜딩 적용
- 사용자 풀에 MFA 추가
- 위협 방지 기능이 있는 고급 보안
- Lambda 트리거를 사용하여 사용자 풀 워크플로 사용자 정의
- 사용자 풀 분석에 Amazon Pinpoint 사용

Amazon Cognito 인증 및 권한 부여 모델에 대한 개요는 Amazon Cognito에서 인증이 작동하는 방식 섹션을 참조하세요.

사용자 풀 인증에 성공한 AWS 서비스 후 다른에 액세스하려면 섹션을 참조하세요로그인 후 자격 증명 풀을 AWS 서비스 사용하여 액세스.

AWS Management Console 및 사용자 풀 SDKs를 사용하는 것 외에도를 사용하여 사용자 풀을 관리할 수도 있습니다AWS Command Line Interface.

주제

- 사용자 풀에 소셜 로그인 추가
- SAML 2.0 ID 제공업체 추가

사용자 풀에 소셜 로그인 추가

사용자에게 기존 퍼블릭 또는 소셜을 통해 애플리케이션에 로그인할 수 있는 기능을 제공하면 ID 제공업체가 인증 환경을 개선할 수 있습니다. Amazon Cognito 사용자 풀은 Facebook, Google, Amazon 및 Apple과 같은 인기 있는 소셜 ID 제공업체(IdPs)와 통합되어 사용자에게 이미 익숙하고 편리한 로그인 옵션을 제공합니다.

소셜 로그인을 설정할 때 사용자에게 애플리케이션 전용 계정을 생성하는 대안을 제공합니다. 이렇게 하면 전환율이 향상되고 가입 프로세스가 더 원활하게 진행될 수 있습니다. 사용자의 관점에서 보면 다른 사용자 이름과 암호를 기억하지 않고도 기존 소셜 자격 증명을 적용하여 빠르게 인증할 수 있습니다.

사용자 풀에서 소셜 IdP를 구성하려면 몇 가지 주요 단계가 필요합니다. 클라이언트 ID와 보안 암호를 얻으려면 소셜 공급자에 애플리케이션을 등록해야 합니다. 그런 다음 소셜 IdP 구성을 사용자 풀에 추가하여 요청하려는 범위와 IdP 속성에서 매핑하려는 사용자 풀 속성을 지정할 수 있습니다. 런타임 시 Amazon Cognito는 공급자와의 토큰 교환을 처리하고, 사용자 속성을 매핑하고, 공유 사용자 풀 형식으로 애플리케이션에 토큰을 발급합니다.

소셜 IdP에 등록

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

Facebook으로 앱을 등록하려면

1. Facebook에서 개발자 계정을 생성합니다.

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

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

    ```
    https://your_user_pool_domain
    ```
12. 변경 사항 저장을 선택합니다.
13. 탐색 모음에서 제품을 선택한 다음 Facebook 로그인에서 구성을 선택합니다.
14. Facebook 로그인 구성 메뉴에서 설정을 선택합니다.

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

    ```
    https://your_user_pool_domain/oauth2/idpresponse
    ```
15. 변경 사항 저장을 선택합니다.

Amazon으로 앱을 등록하려면

1. Amazon에서 개발자 계정을 생성합니다.
2. Amazon 자격 증명으로 로그인합니다.
3. Amazon 보안 프로파일을 생성하여 Amazon 클라이언트 ID와 클라이언트 보안 암호를 받아야 합니다.

   페이지 상단의 탐색 모음에서 Apps and Services(앱 및 서비스)를 선택한 다음 Login with Amazon을 선택합니다.
4. 보안 프로필 생성(Create a Security Profile)을 선택합니다.

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

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

10. [허용된 반환 URL(Allowed Return URLs)]에 `/oauth2/idpresponse` 엔드포인트가 있는 사용자 풀 도메인을 입력합니다.

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

11. 저장(Save)을 선택합니다.

Google로 앱을 등록하려면

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

1. Google에서 개발자 계정을 생성합니다.
2. Google Cloud Platform 콘솔에 로그인합니다.
3. 상단 탐색 모음에서 프로젝트 선택(Select a project)을 선택합니다. Google 플랫폼에 프로젝트가 이미 있는 경우 이 메뉴에는 대신 기본 프로젝트가 표시됩니다.
4. 새 프로젝트(NEW PROJECT)를 선택합니다.
5. 프로젝트의 이름을 입력한 다음 생성(CREATE)을 선택합니다.
6. 왼쪽 탐색 모음에서 API 및 서비스와 Oauth 동의 화면을 차례로 선택합니다.
7. 앱 정보, 앱 도메인, 권한이 부여된 도메인 및 개발자 연락처 정보를 입력합니다. 권한이 부여된 도메인은 `amazoncognito.com` 및 사용자 지정 도메인의 루트를 포함해야 합니다. 예: `example.com` 저장 후 계속(SAVE AND CONTINUE)을 선택합니다.
8. 1. 범위(Scopes)에서 범위 추가 또는 제거(Add or remove scopes)를 선택한 다음 최소한 다음 OAuth 범위를 선택합니다.

1. `.../auth/userinfo.email`
2. `.../auth/userinfo.profile`
3. openid

9. 사용자 테스트(Test users)에서 사용자 추가(Add users)를 선택합니다. 이메일 주소와 기타 인증된 테스트 사용자를 입력한 다음 저장 후 계속을 선택합니다.
10. 왼쪽 탐색 모음을 다시 확장하고 API 및 서비스와 자격 증명을 차례로 선택합니다.
11. 자격 증명 생성과 OAuth 클라이언트 ID를 차례로 선택합니다.
12. 애플리케이션 유형(Application type)을 선택하고 클라이언트 이름을 지정합니다.
13. 권한 있는 JavaScript 원본(Authorized JavaScript origins)에서 URI 추가(ADD URI)를 선택합니다. 사용자 풀 도메인을 입력합니다.

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

14. 권한 있는 리디렉션 URI(Authorized redirect URIs)에서 URI 추가(ADD URI)를 선택합니다. 사용자 풀 도메인의 `/oauth2/idpresponse` 엔드포인트에 대한 경로를 입력합니다.

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

15. 생성(CREATE)을 선택합니다.
16. 내 클라이언트 ID와 내 클라이언트 보안 암호에서 Google이 표시하는 값을 안전하게 저장합니다. Google IdP를 추가할 때 이러한 값을 Amazon Cognito에 입력합니다.

Apple으로 앱을 등록하려면

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

1. Apple에서 개발자 계정을 생성합니다.
2. Apple 자격 증명으로 로그인합니다.
3. 왼쪽 탐색 모음에서 인증서, 식별자 및 프로필(Certificates, Identifiers & Profiles)을 선택합니다.
4. 왼쪽 탐색 모음에서 식별자(Identifiers)를 선택합니다.
5. 식별자(Identifiers) 페이지에서 + 아이콘을 선택합니다.
6. 새 식별자 등록(Register a New Identifier) 페이지에서 앱 ID(App IDs)를 선택한 다음 계속(Continue)을 선택합니다.

7. 유형 선택 페이지에서 앱을 선택하고 계속을 선택합니다.
8. 앱 ID 등록(Register an App ID) 페이지에서 다음을 수행합니다.

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

    1. 설명(Description)에 설명을 입력합니다.
    2. 식별자(Identifier) 아래에 식별자를 입력합니다. 사용자 풀에서 소셜 ID 제공업체 사용에서 ID 제공업체로 Apple을 선택한 후 이 값이 필요하므로 이 서비스 ID를 기록해 두세요.
    3. 계속, 등록을 차례로 선택합니다.
13. 식별자 페이지에서 방금 생성한 서비스 ID를 선택합니다.

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

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

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

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

6. 다음을 선택한 다음 완료를 선택합니다. 도메인을 확인할 필요는 없습니다.
7. 계속(Continue)과 저장(Save)을 차례로 선택합니다.

14. 왼쪽 탐색 모음에서 키(Keys)를 선택합니다.
15. 키(Keys) 페이지에 + 아이콘을 선택합니다.
16. 새 키 등록(Register a New Key) 페이지에서 다음을 수행합니다.
    1. [키 이름(Key Name)] 아래에 키 이름을 입력합니다.
    2. Apple로 로그인을 선택한 다음 구성을 선택합니다.
    3. 키 구성 페이지에서 앞서 생성한 앱 ID를 기본 앱 ID로 선택합니다. 저장을 선택합니다.
    4. 계속(Continue), 등록(Register)을 차례로 선택합니다.
17. 키 다운로드 페이지에서 다운로드를 선택하여 프라이빗 키를 다운로드하고 표시된 키 ID를 적어 둔 다음 완료를 선택합니다. 이 프라이빗 키와 이 페이지에 표시되는 키 ID(Key ID) 값은 사용자 풀에서 소셜 ID 제공업체 사용에서 자격 증명 공급자로 Apple을 선택한 후에 필요합니다.

사용자 풀에 소셜 IdP 추가

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

를 사용하여 사용자 풀 소셜 자격 증명 공급자를 구성하려면 AWS Management Console

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

- Facebook - 이전 섹션에서 생성된 앱 클라이언트 ID와 앱 클라이언트 보안 암호를 입력하고 API 버전(예: 버전 2.12)을 선택합니다. 각 Facebook API에는 수명 주기와 사용 중단 날짜가 있으므로 가능한 한 최신 버전을 선택하는 것이 좋습니다. Facebook 범위와 속성은 API 버전마다 다를 수 있습니다. Facebook으로 소셜 자격 증명 로그인을 테스트하여 페더레이션이 의도한 대로 작동하는지 확인하는 것이 좋습니다.
- Sign in with Apple - 이전 섹션에서 생성된 서비스 ID, 팀 ID, 키 ID, 프라이빗 키를 입력합니다.

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

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

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

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

- 최종 사용자가 Apple의 로그인 페이지를 떠난 후 오류가 발생합니다(Amazon Cognito 내의 내부 오류 또는 개발자가 작성한 오류일 수 있음).
- 서비스 ID 식별자는 사용자 풀 및/또는 기타 인증 서비스에서 사용됩니다.
- 사용자가 로그인한 후 개발자가 추가 범위를 추가합니다. 사용자는 인증할 때와 토큰을 새로 고칠 때만 새 정보를 검색합니다.
- 개발자가 사용자를 삭제한 후 해당 사용자가 Apple ID 프로필에서 앱을 제거하지 않고 다시 로그인합니다.

8. 자격 증명 공급자의 속성을 사용자 풀에 매핑합니다. 자세한 내용은 매핑에 대해 알아야 할 사항 섹션을 참조하세요.

9. 생성(Create)을 선택합니다.
10. 앱 클라이언트 메뉴에서 목록에 있는 앱 클라이언트 중 하나를 선택하고 호스팅 UI 설정을 편집합니다. [자격 증명 공급자(Identity providers)]에서 앱 클라이언트에 새 소셜 자격 증명 공급자를 추가합니다.
11. 변경 사항 저장을 선택합니다.

소셜 IdP 구성 테스트

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

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

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

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

SAML 2.0 ID 제공업체 추가

앱 사용자는 SAML 2.0 ID 제공업체(idP)로 로그인할 수 있습니다. 고객이 조직의 내부 고객이거나 연결된 회사인 경우 소셜 IdP보다 SAML 2.0 IdP를 선택할 수 있습니다. 소셜 IdP가 모든 사용자가 계정에 등록할 수 있도록 허용하는 경우 SAML IdP는 조직에서 제어하는 사용자 디렉터리와 페어링할 가능성이 더 높습니다. 사용자가 직접 또는 타사를 통해 로그인하는지 여부와 무관하게 모든 사용자는 사용자 풀에 프로필을 보유합니다. SAML 자격 증명 공급자를 통한 로그인 추가를 원치 않는 경우 이 단계를 건너뛰세요.

자세한 내용은 사용자 풀에서 SAML ID 제공업체 사용 섹션을 참조하세요.

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

또한 SAML ID 제공업체에 어설션 소비자 서비스(ACS) 엔드포인트를 제공해야 합니다. SAML ID 제공업체의 SAML 2.0 POST 바인딩에 대해 사용자 풀 도메인에서 다음 엔드포인트 구성합니다. 사용자 풀 도메인에 대한 자세한 정보는 사용자 풀 도메인 구성 섹션을 참조하세요.

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

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

일부 SAML ID 제공업체의 경우 대상 URI 또는 SP 엔터티 ID라고도 하는 SP(서비스 제공업체) `urn`을 다음 형식으로 제공해야 합니다.

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

Amazon Cognito 콘솔의 사용자 풀 개요 대시보드에서 사용자 풀 ID를 찾을 수 있습니다.

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

Amazon Cognito 사용자 풀은 사후 바인딩 엔드포인트와의 SAML 2.0 페더레이션을 지원합니다. 이렇게 하면 사용자 풀이 사용자 에이전트를 통해 ID 제공업체로부터 직접 SAML 응답을 받으므로 앱에서 SAML 어설션 응답을 검색하거나 구문 분석할 필요가 없습니다.

사용자 풀에 SAML 2.0 자격 증명 공급자를 구성하려면

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 자격 AWS 증명을 입력합니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. 소셜 및 외부 제공업체 메뉴를 선택합니다. [페더레이션 로그인(Federated sign-in)]을 찾아서 [자격 증명 공급자 추가(Add an identity provider)]를 선택합니다.
5. [SAML] 소셜 자격 증명 공급자를 선택합니다.

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

   > **Note**
   >
   > 이 옵션을 선택했고 SAML ID 제공업체가 서명된 로그아웃 요청을 필요로 하는 경우 SAML IdP를 사용하여 Amazon Cognito에서 제공한 서명 인증서도 구성해야 합니다. SAML IdP는 서명된 로그아웃 요청을 처리하고 Amazon Cognito 세션에서 사용자를 로그아웃합니다.

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

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

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

   예를 들어 [사용자 풀 속성(User pool attribute)] `email`을 선택한 경우 자격 증명 공급자의 SAML 어설션에 표시된 대로 SAML 속성 이름을 입력합니다. 자격 증명 공급자가 참조용으로 샘플 SAML 어설션을 제공할 수도 있습니다. `email`과 같은 간단한 이름을 사용하는 자격 증명 공급자도 있고, 다음 예제와 같이 URL 포맷의 속성 이름을 사용하는 자격 증명 공급자도 있습니다.

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

10. 생성을 선택합니다.

Amazon Cognito용 설정 옵션 안내

체계적인 안내형 환경에서 Amazon Cognito의 기능을 평가해 볼 수 있습니다. 다음은 사용자 풀 및 ID 풀에 대한 맞춤형 경험을 제공하는 몇 가지 외부 리소스입니다.

워크숍 완료

AWS 워크숍 스튜디오는 대부분의 Amazon Cognito 기능 설정을 안내하는 워크숍을 주최합니다. Amazon Cognito 이러한 기능에는 사용자 풀 API, 사용자 풀 호스팅 UI, ID 풀 및 보안 구성이 포함됩니다.

예제에서 애플리케이션 코드 추가

이 가이드의 코드 예제 장에는 사용자 풀 및 ID 풀과 함께 사용할 수 있는 애플리케이션 코드가 있습니다. 코드 예제 장의 사용자 풀 섹션에는 개별 작업을 다루는 짧은 조각과 다양한 프로그래밍 언어로 된 엔드투엔드 예제 애플리케이션을 위한 긴 예제가 있습니다.

를 사용하여 풀스택 애플리케이션 생성 AWS Amplify

AWS Amplify는 애플리케이션 및 사용자 인터페이스를 개발하고 호스팅하려는 개발자를 AWS 서비스 위한 입니다. Amazon Cognito는 Amplify의 인증 구성 요소입니다. 애플리케이션에 인증을 추가할 때 Amplify는 Amazon Cognito 사용자 풀 및 ID 풀 리소스의 배포를 자동화할 수 있습니다. 웹 및 모바일 앱과 Amazon Cognito 인증 및 권한 부여 통합 섹션도 참조하세요.

GitHub의 Amazon Cognito 애플리케이션 리소스 더 보기

- Amazon Cognito용 .NET을 사용한 인증 흐름 예제
- Amazon Cognito 암호 없는 인증
- Amazon Verified Permissions가 포함된 PetStore 예제
- ABAC + 자격 증명 풀을 사용하여 AWS 리소스에 액세스하는 샘플 React 앱
- AWS CDK를 사용한 Amazon Cognito 및 API Gateway 기반 시스템 간 권한 부여
- Amazon Cognito, API Gateway 및 IAM을 사용하여 세분화된 권한 부여 구축
- CloudFront authorization@edge

워크숍 더 보기

- Amazon Cognito 및 WebAuthn을 사용한 암호 없는 인증 구현
- Amazon Cognito 워크숍
- Amazon Cognito 문제 해결 워크숍
- Amazon Cognito 및 Verified Permissions를 사용한 인증 및 권한 부여
- Amazon Cognito JWT 심층 분석

블로그 게시물

- Amazon CloudFront 프록시를 사용하여 Amazon Cognito에 대한 퍼블릭 클라이언트 보호
- Azure AD를 사용하여 페더레이션 인증을 위해 Amazon Cognito를 설정하는 방법
- 웹 앱 인증 간소화: Amazon Cognito 사용자 풀을 사용한 AD FS 페더레이션 가이드

Amazon Cognito 설계 계획

이 장의 선택 사항은 나중에 변경하는 것보다 미리 구성하는 것이 더 쉽습니다. 이 장에서는 다음 주제를 다룹니다.

- 인증 접근 방식 선택 - 사용자가 로그인하는 방식입니다.
- 일반적인 시나리오 - 애플리케이션에 맞게 패턴을 설계합니다.
- 다중 테넌시 - 둘 이상의 고객 조직에 서비스를 제공하는 경우 각 테넌트의 사용자와 데이터를 격리하는 전략은에서 다룹니다B2B 멀티 테넌트 애플리케이션 구축.

인증 접근 방식 선택

Amazon Cognito 사용자 풀로 사용자를 인증할 때 로그인 환경이 표시되는 방법을 선택합니다. 다음 세 가지 접근 방식은 Amazon Cognito가 제공하는 인터페이스 양과 빌드하는 인터페이스 양이 다르며 상호 배타적이지 않습니다. 결합할 수 있습니다. 모두 동일한 사용자 풀에 대해 인증되고 동일한 토큰을 생성합니다. 차이점은 로그인 페이지가 실행되는 위치와 제어 권한이 얼마나 있는지입니다.

관리형 로그인(Amazon Cognito에서 페이지 호스팅)

Amazon Cognito는 사용자 풀 도메인의 가입, 로그인, 다중 인증 및 암호 재설정 페이지를 제공합니다. 브랜딩 편집기를 사용하여 모양을 사용자 지정합니다. UI를 빌드하지 않고 완전한 호스팅 로그인 환경을 원하는 경우와 표준 기반 OAuth 2.0 및 OpenID Connect 흐름을 처리하려는 경우이 옵션을 선택합니다. 관리형 로그인은 이러한 Amazon Cognito 호스팅 페이지의 최신 버전입니다. 이전 클래식 호스팅 UI는 앱 클라이언트별로 선택할 수 있는 더 간단한 1세대 대안입니다. the section called “관리형 로그인” 섹션을 참조하세요.

사용자 풀 API(인터페이스 빌드)

애플리케이션은 자격 증명을 수집하고 AWS SDK를 통해 Amazon Cognito 사용자 풀 API를 직접 호출합니다. 전체 인터페이스를 빌드하고 제어합니다. 완전 사용자 지정 로그인 환경이 필요하거나 백엔드 서비스에서 사용자에 로그인할 때이 옵션을 선택합니다. the section called “인증” 섹션을 참조하세요.

타사 자격 증명 공급자(연합)

사용자는 외부 공급자를 통해 로그인합니다. Google 또는 Apple과 같은 소셜 공급자이거나 엔터프라이즈 SAML 또는 OIDC 공급자일 수 있습니다. Amazon Cognito는 자격 증명을 사용자 풀에 페더레이션합니다. 페더레이션 로그인은 관리형 로그인 또는 사용자 지정 인터페이스에서 리

디렉션하는 권한 부여 및 로그인 엔드포인트와 같은 사용자 풀 도메인의 호스팅 로그인 엔드포인트를 통해 실행됩니다. 기본 사용자 풀 API 작업은 페더레이션 공급자를 통해 사용자를 로그인`InitiateAuth`하지 않으므로 페더레이션은 항상 직접 SDK 인증 흐름이 아닌 호스팅 엔드포인트를 사용합니다. 사용자가 이미 다른 곳에 자격 증명이 있는 경우이 옵션을 선택합니다. the section called “타사 IdP 로그인” 섹션을 참조하세요.

다음 표에는 각 접근 방식에 대해 도달해야 하는 시기가 요약되어 있습니다.

다음을 수행하려는 경우…	로 시작
빌드 및 유지 관리할 UI 없이 빠르게 로그인 발송	관리형 로그인
모든 로그인 화면의 모양과 동작을 완벽하게 제어합니다.	사용자 풀 API
백엔드 또는 machine-to-machine 호출자의 요청 인증	사용자 풀 API(및 M2M의 경우 클라이언트 자격 증명)
사용자가 Google, Apple 또는 엔터프라이즈 디렉터리로 로그인하도록 허용	관리형 로그인 또는 호스팅된 권한 부여 및 로그인 엔드포인트를 통한 타사 자격 증명 공급자(페더레이션)

> **Note**
>
> 어떤 접근 방식을 선택하든 사용자 풀의 기능 계획에 따라 사용 가능한 기능이 결정됩니다. Lite 플랜은 핵심 가입 및 로그인을 다루고, Essentials는 암호 없는 암호 로그인 및 관리형 로그인과 같은 옵션을 추가하며, Plus는 고급 위협 방지 기능을 추가합니다. 사용자 풀을 생성할 때 계획을 선택하면 나중에 변경할 수 있으므로 시작하기 전에 결정할 필요가 없습니다. 예상한 기능을 사용할 수 없는 경우 먼저 계획을 확인하세요. 전체 비교는 단원을 참조하십시오the section called “사용자 풀 기능 플랜”.

> **Note**
>
> 이들은이 사용자인 모든 답변에 접근합니까? (인증). Amazon S3와 같은 서비스에 도달하기 위해 사용자에게 임시 AWS 자격 증명을 부여해야 하는 경우, 이는 자격 증명 풀에서 응답하는

별도의 질문인 권한 부여입니다. the section called “Amazon Cognito의 어떤 부분이 필요합니까?” 섹션을 참조하세요.

일반적인 Amazon Cognito 시나리오

이 주제에서는 Amazon Cognito 사용에 대한 7가지 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 사용자에게 다른에 대한 액세스 권한을 부여하는 임시 AWS 자격 증명을 제공합니다 AWS 서비스.

사용자 풀은 Amazon Cognito의 사용자 디렉터리입니다. 앱 사용자는 사용자 풀을 통해 직접 로그인하거나 타사 ID 제공업체(idP)를 통해 연동할 수 있습니다. 사용자 풀에서는 Facebook, Google 및 Apple, 그리고 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 액세스할 수 있습니다.

자격 증명 풀을 사용하면 사용자가 Amazon S3 및 DynamoDB와 같은 서비스에 액세스 AWS 하기 위한 임시 AWS 자격 증명을 얻을 수 있습니다. 자격 증명 풀은 익명 게스트 사용자는 물론, 서드 파티 IdP를 통한 페더레이션도 지원합니다.

주제

- 사용자 풀로 인증
- 사용자 풀 토큰을 사용하여 백엔드 리소스에 액세스
- 사용자 풀로 API Gateway 및 Lambda를 사용하여 리소스 액세스
- 사용자 풀 및 자격 증명 풀을 사용하여 AWS 서비스에 액세스
- 타사로 인증하고 자격 증명 풀을 사용하여 AWS 서비스에 액세스
- Amazon Cognito를 사용하여 AWS AppSync 리소스에 액세스
- M2M(Machinemachine-to-machine) 애플리케이션 권한 부여
- 단계적 인증

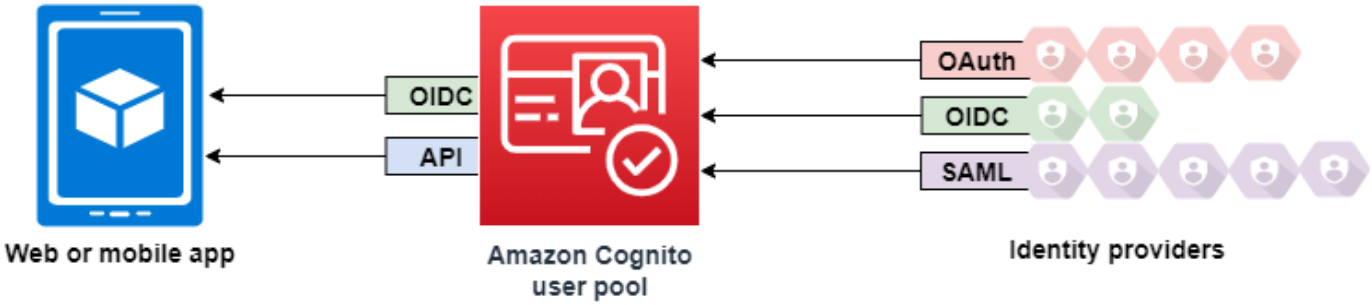
사용자 풀로 인증

사용자가 사용자 풀을 사용하여 인증하도록 설정할 수 있습니다. 앱 사용자는 사용자 풀을 통해 직접 로그인하거나 타사 ID 제공업체(idP)를 통해 연동할 수 있습니다. 사용자 풀에서는 Facebook, Google

및 Apple, 그리고 Amazon을 통한 소셜 로그인에서 반환된 토큰과 OpenID Connect(OIDC) 및 SAML IdP에서 반환된 토큰의 처리 작업을 관리합니다.

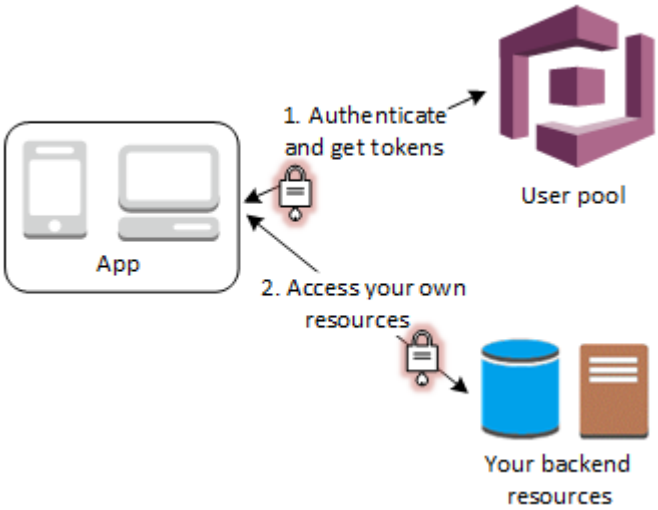
인증에 성공하면 웹 또는 모바일 앱이 Amazon Cognito로부터 사용자 풀 토큰을 수신합니다. 이러한 토큰을 사용하여 앱이 다른 AWS 서비스에 액세스할 수 있는 AWS 자격 증명을 검색하거나, 이를 사용하여 서버 측 리소스 또는 Amazon API Gateway에 대한 액세스를 제어할 수 있습니다.

자세한 내용은 인증 세션의 예 및 사용자 풀 JSON 웹 토큰(JWT) 이해 섹션을 참조하세요.



사용자 풀 토큰을 사용하여 백엔드 리소스에 액세스

사용자 풀 로그인에 성공하면 웹 또는 모바일 앱이 Amazon Cognito로부터 사용자 풀 토큰을 수신합니다. 이러한 토큰을 사용하여 서버 측 리소스에 대한 액세스를 제어할 수 있습니다. 또한 사용자 풀 그룹을 생성하여 권한을 관리하고 다른 유형의 사용자를 표시할 수도 있습니다. 그룹을 사용하여 리소스에 액세스할 권한을 제어하는 방법에 대한 자세한 내용은 사용자 풀에 그룹 추가 섹션을 참조하세요.



사용자 풀에 대한 도메인이 구성되면 Amazon Cognito는 호스트된 웹 UI를 프로비저닝합니다. 이를 사용하여 앱에 가입 및 로그인 페이지를 추가할 수 있습니다. 이 OAuth 2.0 기반을 사용하면 사용자가 보

호된 리소스에 액세스할 수 있는 자체 리소스 서버를 만들 수 있습니다. 자세한 내용은 범위, M2M 및 리소스 서버 섹션을 참조하세요.

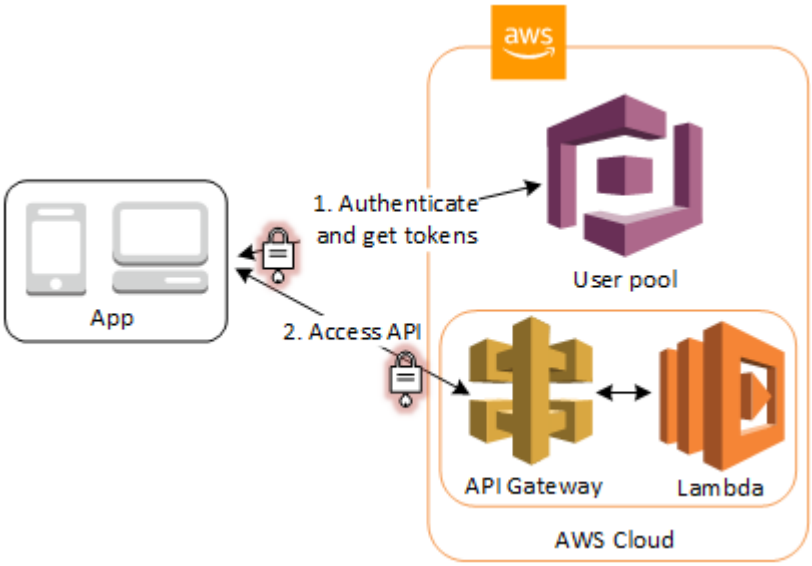
사용자 풀 인증에 대한 자세한 내용은 인증 세션의 예 및 사용자 풀 JSON 웹 토큰(JWT) 이해 섹션을 참조하세요.

사용자 풀로 API Gateway 및 Lambda를 사용하여 리소스 액세스

사용자가 API Gateway를 통해 API에 액세스하도록 할 수 있습니다. API Gateway는 성공적인 사용자 풀 인증에서 토큰을 검증하고, 토큰을 사용하여 사용자에게 Lambda 함수 또는 자체 API를 비롯한 리소스에 대한 액세스 권한을 부여합니다.

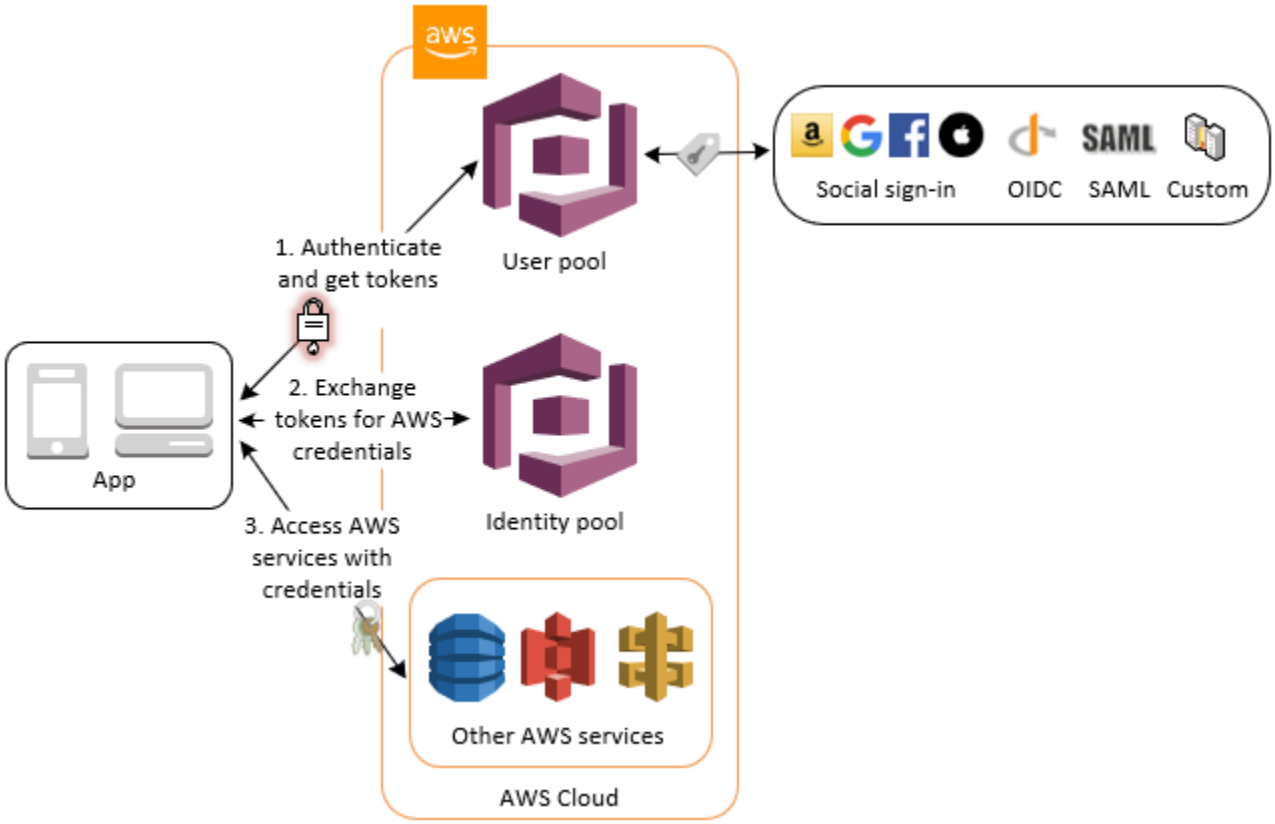
사용자 풀에 그룹을 사용하여 IAM 역할에 그룹 멤버십을 매핑함으로써 API Gateway를 통해 권한을 제어할 수 있습니다. 사용자가 속한 그룹은 앱 사용자가 로그인할 때 사용자 풀에서 제공한 ID 토큰에 포함됩니다. 사용자 풀 그룹에 대한 자세한 내용은 섹션을 참조하세요사용자 풀에 그룹 추가.

Amazon Cognito 권한 부여자 Lambda 함수에서 확인을 위해 API Gateway에 대한 요청과 함께 사용자 풀 토큰을 제출할 수 있습니다. API Gateway 대한 자세한 내용은 Amazon Cognito 사용자 풀에 API Gateway 사용을 참조하세요.



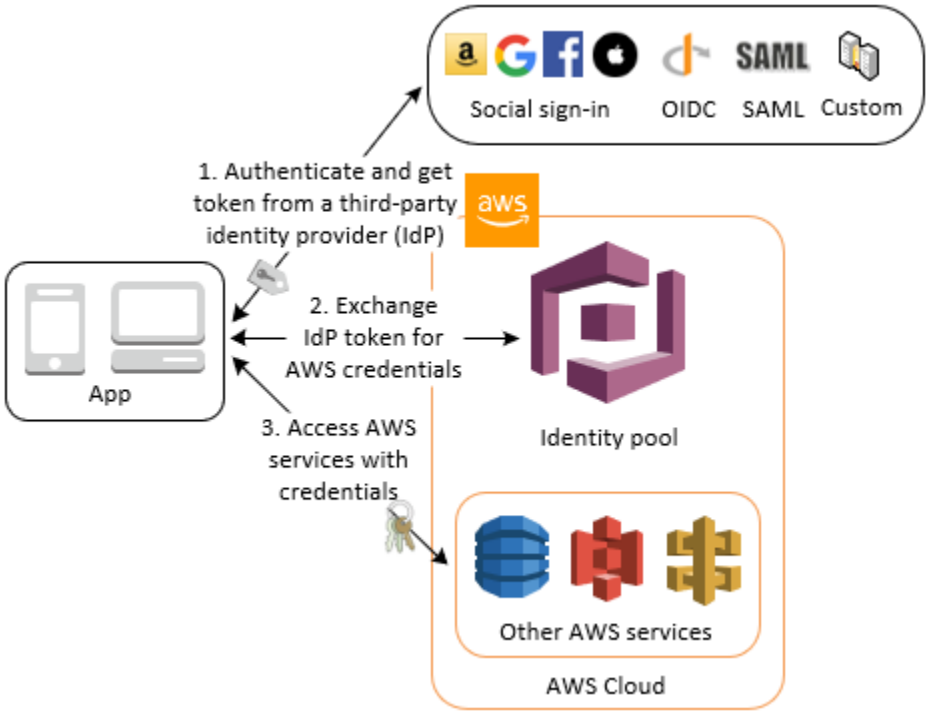
사용자 풀 및 자격 증명 풀을 사용하여 AWS 서비스에 액세스

사용자 풀 인증에 성공하면 앱은 Amazon Cognito로부터 사용자 풀 토큰을 수신합니다. 자격 증명 풀을 사용하여 다른 AWS 서비스에 대한 임시 액세스로 교환할 수 있습니다. 자세한 내용은 로그인 후 자격 증명 풀을 AWS 서비스 사용하여 액세스 및 Amazon Cognito ID 풀 시작 섹션을 참조하세요.



타사로 인증하고 자격 증명 풀을 사용하여 AWS 서비스에 액세스

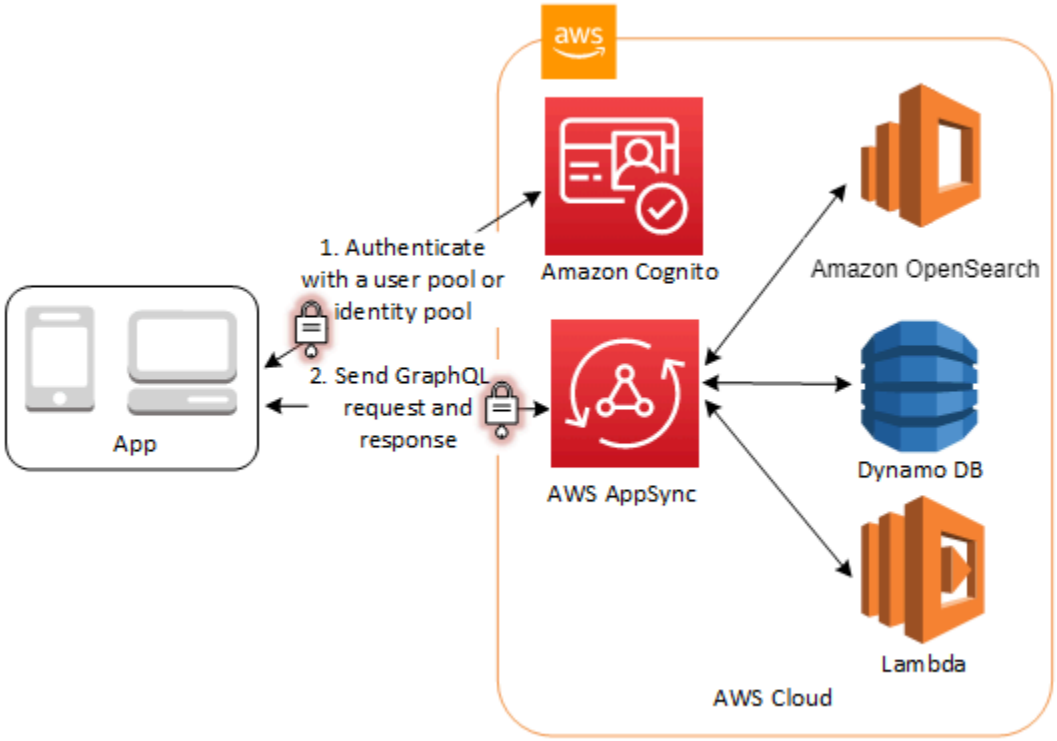
ID 풀을 통해 AWS 서비스에 대한 사용자 액세스를 활성화할 수 있습니다. 자격 증명 풀에는 서드 파티 자격 증명 공급자에 의해 인증된 사용자의 IdP 토큰이 필요합니다(익명 게스트인 경우 필요하지 않음). 그 대신 자격 증명 풀은 다른 AWS 서비스에 액세스하는 데 사용할 수 있는 임시 AWS 자격 증명을 부여합니다. 자세한 내용은 Amazon Cognito ID 풀 시작 섹션을 참조하세요.



Amazon Cognito를 사용하여 AWS AppSync 리소스에 액세스

성공적인 Amazon Cognito 사용자 풀 인증의 토큰을 사용하여 사용자에게 AWS AppSync 리소스에 대한 액세스 권한을 부여할 수 있습니다. 자세한 내용은 AWS AppSync 개발자 안내서의 AMAZON_COGNITO_USER_POOLS 인증을 참조하세요.

자격 증명 풀에서 받은 IAM 자격 증명을 사용하여 AWS AppSync GraphQL API에 대한 요청에 서명할 수도 있습니다. AWS_IAM 권한 부여를 참조하십시오.



M2M(Machinemachine-to-machine) 애플리케이션 권한 부여

M2M(Machinemachine-to-machine) 권한 부여를 사용하면 애플리케이션, 서비스 또는 디바이스가 로그인한 사용자 없이 자체적으로 인증하고 APIs에 대한 권한 있는 액세스를 얻을 수 있습니다. Amazon Cognito는 사용자 대신 애플리케이션에 권한을 부여하고 ID 토큰과 새로 고침 토큰 없이 액세스 토큰만 발급합니다. 이는 사물 인터넷(IoT) 디바이스 플릿에서 데이터를 자율적으로 수집하는 애플리케이션과 같은 워크로드용 모델로, 각 디바이스는 API에 연결하고 루프에 사람이 없는 데이터를 푸시합니다.

M2M 권한 부여를 활성화하려면 클라이언트 보안 암호가 있는 앱 클라이언트를 빌드하고 권한 `client_credentials` 부여를 위해 활성화합니다. 시스템 자격 증명은 권한 부여 서버에서 토큰을 요청하고 리소스 서버에 토큰을 제공합니다. 자세한 내용은 기계 간(M2M) 인증 및 Token 엔드포인트 섹션을 참조하세요.

단계적 인증

사용자가 사용자 이름과 암호로 로그인할 수 있는 뱅킹 애플리케이션을 생각해 보세요. 유선 전송을 수행하려면 애플리케이션에 멀티 팩터 인증(MFA)이 필요합니다. 사용자가 전송을 시도하면 애플리케이션은 사용자의 액세스 및 ID 토큰에서 `acr` 클레임을 검사하고 인증 수준이 충분하지 않은지 확인한 다

음 사용자를 Amazon Cognito로 리디렉션하여 MFA를 완료합니다. 그런 다음 Amazon Cognito는 인증 수준이 높은 새 토큰을 발급하고 애플리케이션은 전송을 허용합니다.

단계적 인증을 위해 사용자를 Amazon Cognito로 리디렉션하기 전에 애플리케이션이 이미 보유하고 있는 토큰을 평가합니다. `auth_time` 클레임을 작업에서 허용하는 최대 수명과 비교하고 토큰의 `acr` 수준을 작업에 필요한 수준과 비교합니다. 필수 인증 수준 또는 최신성이 아직 충족되지 않은 경우에만 인증 흐름을 완료하도록 사용자를 Amazon Cognito로 리디렉션합니다.

ACR 수준, AMR 값 및 기타 단계적 인증 개념에 대한 자세한 내용은 섹션을 참조하세요the section called “ACR 및 AMR 클레임을 사용한 인증 수준”. 단계적 인증 요청 방법은 USER_AUTH 흐름의 경우 , 관리형 로그인의 USER_AUTH 흐름을 사용한 단계적 인증 경우 권한 부여 엔드포인트를 참조Login 엔드포인트하세요.

로그인 표면 설정

사용자가 인증하기 전에 사용자가 로그인하는 표면을 설정합니다. 관리형 로그인의 경우, 이는 페이지를 제공하는 사용자 풀 도메인과 페이지를 가리키는 앱 클라이언트 및 적용하는 브랜딩을 의미합니다. 이 장에서는 다음 주제를 다룹니다.

- 관리형 로그인 - 호스팅된 로그인 페이지, 이를 제공하는 사용자 풀 도메인 및 브랜딩 편집기입니다.
- 기존 사용자 풀을 관리형 로그인으로 이동할 때 변경되는 클래식 호스팅 UI에서 마이그레이션.

관리형 로그인, 사용자 풀 API 및 페더레이션을 처음부터 결정하려면 섹션을 참조하세요the section called “접근 방식 선택”. 인증 흐름 자체는 섹션을 참조하세요the section called “인증”.

사용자 풀 관리형 로그인

웹 도메인을 선택하여 사용자 풀에 대한 서비스를 호스팅할 수 있습니다. Amazon Cognito 사용자 풀은 도메인을 추가할 때 관리형 로그인이라고 하는 다음 함수를 가져옵니다.

- OAuth 2.0 및 OpenID Connect(OIDC)와 함께 작동하는 애플리케이션에 대한 ID 제공업체(IdP) 역할을 하는 권한 부여 서버. 권한 부여 서버는 요청을 라우팅하고, JSON 웹 토큰(JWT)을 발급하고 관리하며, 사용자 속성 정보를 전달합니다.
- 로그인, 로그아웃, 암호 관리와 같은 인증 작업을 위해 바로 사용 가능한 사용자 인터페이스(UI). 관리형 로그인 페이지는 인증 서비스를 위한 웹 프런트 엔드 역할을 합니다.
- SAML 2.0 IdP, OIDC IdP, Facebook, Login with Amazon, Sign in with Apple 및 Google에 대한 서비스 공급자(SP) 또는 신뢰 당사자(RP)입니다.

관리형 로그인에는 클래식 호스팅 UI라는 대안이 있습니다. 사용자 풀 도메인에 대해 둘 중 하나를 선택합니다. 관리형 로그인과 클래식 호스팅 UI는 함께 사용되지 않습니다. 클래식 호스팅 UI는 관리형 로그인 서비스의 1세대 버전입니다. 호스팅 UI IdP 및 RP 서비스는 일반적으로 관리형 로그인과 동일한 특성을 갖지만 로그인 페이지는 더 간단한 설계와 더 적은 기능을 갖습니다. 예를 들어 패스키 로그인은 클래식 호스팅 UI에서 사용할 수 없습니다. Lite 기능 플랜에서 클래식 호스팅 UI는 사용자 풀 도메인 서비스에 대한 유일한 옵션입니다.

관리형 로그인 페이지는 사용자 풀의 기본 가입, 로그인, 다중 인증 및 암호 재설정 활동을 위한 웹 인터페이스 모음입니다. 또한 사용자에게 로그인 옵션을 제공하려는 경우 사용자를 하나 이상의 서드 파티 ID 제공업체(IdP)에 연결할 수 있습니다. 사용자를 인증하고 권한을 부여하고자 할 때 앱이 사용자 브라우저에서 관리형 로그인 페이지를 간접적으로 호출할 수 있습니다.

관리형 로그인 사용자 환경을 사용자 지정 로고, 배경 및 스타일로 브랜드에 맞게 만들 수 있습니다. 관리형 로그인 UI에 적용할 수 있는 브랜딩에는 관리형 로그인을 위한 브랜딩 편집기와 호스팅 UI에 대한 호스팅 UI(클래식) 브랜딩이라는 두 가지 옵션이 있습니다.

브랜딩 편집기

Amazon Cognito 콘솔의 최신 인증 옵션과 시각적 편집기를 통해 업데이트된 사용자 환경을 제공합니다.

호스팅 UI(클래식) 브랜딩

Amazon Cognito 사용자 풀을 이전에 도입한 사용자에게는 친숙한 사용자 환경입니다. 호스팅 UI의 브랜딩은 파일 기반 시스템입니다. 호스팅 UI 페이지에 브랜딩을 적용하려면 로고 이미지 파일과 미리 결정된 여러 CSS 스타일 옵션의 값을 설정하는 파일을 업로드합니다.

사용자 풀에 대한 일부 기능 계획에서는 브랜딩 편집기를 사용할 수 없습니다. 자세한 내용은 사용자 풀 기능 플랜 섹션을 참조하세요.

관리형 로그인 및 호스팅 UI 서비스에 대한 요청 구성에 대한 자세한 내용은 사용자 풀 엔드포인트 및 관리형 로그인 참조 섹션을 참조하세요.

관리형 로그인은 권한 부여 및 로그인 엔드포인트의 `acr_values` 및 `max_age` 파라미터로 단계적 인증을 지원합니다. 자세한 내용은 권한 부여 엔드포인트 및 로그인 엔드포인트를 참조하세요.

> **Note**
>
> Amazon Cognito 관리형 로그인은 사용자 지정 인증 문제 Lambda 트리거를 통한 사용자 지정 인증을 지원하지 않습니다.

주제

- 관리형 로그인 현지화
- 약관 문서
- Amazon Cognito 콘솔을 사용하여 관리형 로그인 설정
- 를 사용하여 관리형 로그인 설정 AWS Amplify
- 로그인 페이지 보기
- 인증 페이지 사용자 지정
- 관리형 로그인 및 호스팅 UI에 대해 알아야 할 사항

- 사용자 풀 도메인 구성
- 관리형 로그인 페이지에 브랜딩 적용

관리형 로그인 현지화

관리형 로그인은 기본적으로 사용자 대화형 페이지의 영어로 설정됩니다. 선택한 언어에 맞게 현지화된 관리형 로그인 페이지를 표시할 수 있습니다. 사용 가능한 언어는 AWS Management Console에서 사용 가능한 언어입니다. 사용자에게 배포하는 링크에서 다음 예시와 같이 `lang` 쿼리 파라미터를 추가합니다.

```
https://<your domain>/oauth2/authorize?lang=es&response_type=code&client_id=<your app
 client id>&redirect_uri=<your relying-party url>
```

Amazon Cognito는 `lang` 파라미터를 사용한 초기 요청 후 언어 기본 설정으로 사용자의 브라우저에서 쿠키를 설정합니다. 쿠키가 설정되면 파라미터를 표시하거나 요청에 포함할 필요 없이 언어 선택이 유지됩니다. 예를 들어 사용자가 `lang=de` 파라미터로 로그인 요청을 하면 관리형 로그인 페이지는 쿠키를 지우거나 `lang=en`과 같은 새 현지화 파라미터로 새 요청을 할 때까지 독일어로 렌더링됩니다.

현지화는 관리형 로그인에만 사용할 수 있습니다. 관리형 로그인 브랜딩을 사용하려면 Essentials 또는 Plus 기능 플랜을 사용하고 도메인을 할당해야 합니다.

사용자가 관리형 로그인에서 선택한 항목은 사용자 지정 이메일 또는 SMS 발신자 트리거에서 사용할 수 없습니다. 이러한 트리거를 구현할 때는 다른 메커니즘을 사용하여 사용자가 선호하는 언어를 결정해야 합니다. 로그인 흐름에서 `locale` 속성은 위치에 따라 사용자가 선호하는 언어를 나타낼 수 있습니다. 가입 흐름에서 사용자 풀의 리전 또는 앱 클라이언트 ID는 언어 기본 설정을 나타낼 수 있습니다.

다음 언어를 사용할 수 있습니다.

관리형 로그인 언어

언어	코드
독일어	de
영어	en
스페인어	es
프랑스어	fr

언어	코드
인도네시아어	id
네덜란드어	nl
이탈리아어	작동
일본어	ja
한국어	ko
포르투갈어(브라질)	pt-BR
중국어 간체	zh-CN
중국어 번체	zh-TW

약관 문서

사용자가 가입할 때 이용 약관 및 개인 정보 보호 정책 문서에 대한 링크를 표시하도록 관리형 로그인 페이지를 구성할 수 있습니다. 앱 클라이언트에서 두 용어 문서를 모두 설정하면 가입 중에 사용자에게 가입하면 이용 약관 및 개인 정보 보호 정책에 동의하는 것입니다라는 텍스트가 표시됩니다. 사용 약관 및 개인 정보 보호 정책 문구는 문서에 하이퍼링크로 연결된 관리형 로그인 페이지에 표시됩니다.

약관 문서는 관리형 로그인 현지화와 일치하는 언어별 URL을 지원합니다. 사용자가 `lang` 쿼리 파라미터가 있는 언어를 선택하면 Amazon Cognito는 해당 언어의 용어 문서에 대한 링크를 표시합니다. 특정 언어에 대한 URL을 구성하지 않은 경우 Amazon Cognito는 앱 클라이언트에 대해 구성한 기본 URL을 사용합니다.

앱 클라이언트에 대한 약관 문서를 구성하려면 사용자 풀의 관리형 로그인 메뉴로 이동합니다. 약관 문서에서 약관 문서 생성을 선택합니다.

Amazon Cognito console

약관 문서를 생성하는 방법

1. 사용자 풀로 이동하여 관리형 로그인 메뉴를 선택합니다. 약관 문서를 찾습니다.
2. 약관 문서 생성을 선택합니다.
3. 약관 문서를 할당할 앱 클라이언트를 선택합니다.

4. 약관 이름을 입력합니다. 그러면 콘솔에서 문서가 식별됩니다.
5. 링크에서 언어를 선택하고 해당 언어로 용어 문서를 호스팅하는 URL을 입력합니다.
6. 추가 언어에 대한 URL을 추가하려면 다른 언어 추가를 선택합니다.
7. 생성을 선택합니다.

Amazon Cognito user pools API

다음은 CreateTerms 요청 본문의 예입니다. 이렇게 하면 관리형 로그인이 해당 언어로 현지화될 때 앱 클라이언트의 가입 페이지가 개인 정보 보호 정책의 프랑스어 및 포르투갈어(브라질) 버전에 대한 링크를 `1example23456789` 렌더링합니다. 관리형 로그인이 가입 페이지에서 링크를 렌더링하기 `terms-of-use` 전에에 대한 URLs을 설정하려면 별도의 요청이 필요합니다.

```
{
   "ClientId": "1example23456789",
   "Enforcement": "NONE",
   "Links": {
      "cognito:default" : "https://example.com/privacy/",
      "cognito:french" : "https://example.com/fr/privacy/",
      "cognito:portuguese-brazil" : "https://example.com/pt/privacy/"
   },
   "TermsName": "privacy-policy",
   "TermsSource": "LINK",
   "UserPoolId": "us-east-1_EXAMPLE"
}
```

> **Note**
>
> Amazon Cognito가 관리형 로그인 페이지에 용어 문서를 표시하려면 먼저 앱 클라이언트에 대한 사용 약관과 개인 정보 보호 정책 문서를 모두 생성해야 합니다.

Amazon Cognito 콘솔을 사용하여 관리형 로그인 설정

관리형 로그인 및 호스팅 UI의 첫 번째 요구 사항은 사용자 풀 도메인입니다. 사용자 풀 콘솔에서 사용자 풀의 도메인 탭으로 이동하여 Cognito 도메인 또는 사용자 지정 도메인을 추가합니다. 새 사용자 풀을 생성하는 과정에서 도메인을 선택할 수도 있습니다. 자세한 내용은 사용자 풀 도메인 구성 섹션을 참조하세요. 사용자 풀에서 도메인이 활성 상태이면 모든 앱 클라이언트가 해당 도메인의 퍼블릭 인증 페이지를 제공합니다.

사용자 풀 도메인을 생성하거나 수정할 때 도메인의 브랜딩 버전을 설정합니다. 이 브랜딩 버전은 관리형 로그인 또는 호스팅 UI(클래식) 중에서 선택할 수 있습니다. 선택한 브랜딩 버전은 도메인에서 로그인 서비스를 사용하는 모든 앱 클라이언트에 적용됩니다.

다음 단계는 사용자 풀의 앱 클라이언트 탭에서 앱 클라이언트를 생성하는 것입니다. 앱 클라이언트를 생성하는 과정에서 Amazon Cognito는 애플리케이션에 대한 정보를 요청하고 반환 URL을 선택하라는 메시지를 표시합니다. 반환 URL은 신뢰 당사자(RP) URL, 리디렉션 URI 및 콜백 URL이라고도 합니다. `https://www.example.com` 또는 `myapp://example`과 같이 애플리케이션이 실행되는 URL입니다.

도메인에 설정한 브랜딩 버전에 따라 로그인 페이지의 스타일이 결정됩니다. 브랜딩 편집기에서 관리형 로그인을 선택하여 페이지를 설계하거나 호스팅 UI(클래식)를 선택하여 파일 기반 호스팅 UI 브랜딩을 적용합니다. 브랜딩 편집기와 이를 지원하는 기능 계획에 대한 자세한 내용은 섹션을 참조하세요사용자 풀 기능 플랜.

도메인과 앱 클라이언트를 구성하고 사용자 풀에서 브랜딩 버전을 설정하면 관리형 로그인 페이지를 인터넷에서 사용할 수 있게 됩니다.

Amazon Cognito 콘솔에서 관리형 로그인을 설정하려면

1. 사용자 풀 도메인을 추가합니다. Amazon Cognito 콘솔에서 사용자 풀을 선택하고 도메인 탭을 연 다음 Cognito 도메인(접두사 도메인) 또는 사용자 지정 도메인을 추가합니다. 자세한 내용은 사용자 풀 도메인 구성 섹션을 참조하세요.
2. 도메인의 브랜딩 버전을 설정합니다. 도메인을 생성하거나 수정할 때 관리형 로그인 또는 호스팅 UI(클래식)를 선택합니다. 이 옵션은 로그인 페이지의 스타일을 설정하고 도메인을 사용하는 모든 앱 클라이언트에 적용됩니다.
3. 앱 클라이언트를 생성합니다. 앱 클라이언트 탭에서 앱 클라이언트를 생성하고 반환 URL(콜백 URL 또는 리디렉션 URI라고도 함)을 애플리케이션이 실행되는 URL로 설정합니다.
4. (관리형 로그인만 해당) 스타일을 사용자 지정합니다. 앱 클라이언트의 브랜딩 편집기를 열어 로그인 페이지를 설계합니다. 브랜딩 편집기는 기능 계획의 하위 집합에서 사용할 수 있습니다. 자세한 내용은 섹션을 참조하세요사용자 풀 기능 플랜.
5. 로그인 페이지를 봅니다. 앱 클라이언트의 로그인 페이지 탭에서 로그인 페이지 보기를 선택하여 관리형 로그인 페이지를 엽니다.

를 사용하여 관리형 로그인 설정 AWS Amplify

AWS Amplify 를 사용하여 웹 또는 모바일 앱에 인증을 추가하는 경우 Amplify 명령줄 인터페이스(CLI)에서 관리형 로그인 페이지를 설정하고 Amplify 프레임워크에서 라이브러리를 설정할 수 있습니다. 앱

에 인증을 추가하려면 프로젝트에 `Auth` 범주를 추가합니다. 그런 다음 애플리케이션에서 Amplify 클라이언트 라이브러리를 사용하여 사용자 풀 사용자를 인증합니다.

인증을 위해 관리형 로그인 페이지를 호출하거나 IdP로 리디렉션하는 권한 부여 엔드포인트를 통해 사용자를 페더레이션할 수 있습니다. 사용자가 제공업체를 통해 성공적으로 인증하면 Amplify는 사용자 풀에 새 사용자를 만들고 사용자 토큰을 앱에 전달합니다.

다음 예제에서는를 AWS Amplify 사용하여 앱에서 소셜 공급자와의 관리형 로그인을 설정하는 방법을 보여줍니다.

- React
- Swift
- 플러터
- Android

로그인 페이지 보기

Amazon Cognito 콘솔의 앱 클라이언트 메뉴 아래에 있는 앱 클라이언트의 로그인 페이지 탭에서 로그인 페이지 보기 버튼을 선택합니다. 이 버튼을 클릭하면 다음 기본 파라미터를 통해 사용자 풀 도메인의 로그인 페이지로 이동합니다.

- 앱 클라이언트 ID
- 권한 부여 코드 부여 요청
- 현재 앱 클라이언트에 대해 활성화한 모든 범위에 대한 요청
- 현재 앱 클라이언트에 대한 목록의 첫 번째 콜백 URL

로그인 페이지 보기 버튼은 관리형 로그인 페이지의 기본 기능을 테스트하려는 경우에 유용합니다. 로그인 페이지는 사용자 풀 도메인에 할당한 브랜딩 버전과 일치합니다. 추가 및 수정된 파라미터를 사용하여 로그인 URL을 사용자 지정할 수 있습니다. 대부분의 경우 로그인 페이지 보기 링크에서 자동으로 생성된 파라미터가 앱의 요구 사항과 완전히 일치하지 않습니다. 이러한 경우 앱에서 사용자가 로그인할 때 호출하는 URL을 사용자 지정해야 합니다. 파라미터 및 파라미터 값에 대한 자세한 내용은 사용자 풀 엔드포인트 및 관리형 로그인 참조 단원을 참조하십시오.

로그인 웹 페이지는 다음 URL 형식을 사용합니다. 이 예에서는 `response_type=code` 파라미터를 사용하여 인증 코드 권한 부여를 요청합니다.

```
https://<your domain>/oauth2/authorize?response_type=code&client_id=<your app client
 id>&redirect_uri=<your relying-party url>
```

사용자 풀의 도메인 메뉴에서 사용자 풀 도메인 문자열을 조회할 수 있습니다. 앱 클라이언트 메뉴에서 앱 클라이언트 ID, 콜백 URL, 허용 범위 및 기타 구성을 식별할 수 있습니다.

사용자 지정 파라미터를 사용하여 `/oauth2/authorize` 엔드포인트로 이동하면 Amazon Cognito가 `/oauth2/login` 엔드포인트로 리디렉션하거나 `identity_provider` 또는 `idp_identifier` 파라미터가 있는 경우 자동으로 IdP 로그인 페이지로 리디렉션합니다.

묵시적 부여를 위한 요청의 예

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

```
            https://mydomain.auth.us-east-1.amazoncognito.com/authorize?
response_type=token&client_id=1example23456789&redirect_uri=https://
mydomain.example.com
```

자격 증명 및 액세스 토큰은 리디렉션 URL에 추가된 파라미터로 표시됩니다.

다음은 암시적 권한 부여 요청의 응답 예시입니다.

```
            https://auth.example.com/
#id_token=eyJraaBcDeF1234567890&access_token=eyJraGhIjKlM1112131415&expires_in=3600&token_type=
```

인증 페이지 사용자 지정

과거에 Amazon Cognito는 인증 웹 페이지에 범용 모양을 부여하는 간단한 설계인 클래식 호스팅 UI를 사용하여 로그인 페이지만 호스팅했습니다. 로고 이미지로 Amazon Cognito 사용자 풀을 사용자 지정하고 일부 CSS 스타일 값을 지정한 파일로 일부 스타일을 조정할 수 있습니다. 나중에 Amazon Cognito는 업데이트된 호스팅 인증 서비스인 관리형 로그인을 도입했습니다. 관리형 로그인은 브랜딩 편집기로 업데이트된 디자인과 느낌입니다. 브랜딩 편집기는 노코드 시각적 편집기이며 호스팅 UI 사

용자 지정 환경보다 더 큰 옵션 제품군입니다. 관리형 로그인에는 사용자 지정 백그라운드 이미지와 다크 모드 테마도 도입되었습니다.

사용자 풀에서 관리형 로그인과 호스팅 UI 브랜딩 환경 간에 전환할 수 있습니다. 관리형 로그인 페이지 사용자 지정에 대한 자세한 내용은 관리형 로그인 페이지에 브랜딩 적용 섹션을 참조하세요.

관리형 로그인 및 호스팅 UI에 대해 알아야 할 사항

1시간 관리형 로그인 및 호스팅 UI 세션 쿠키

사용자가 로그인 페이지 또는 타사 제공업체로 로그인하면 Amazon Cognito는 브라우저에 쿠키를 설정합니다. 이 쿠키를 사용하면 사용자는 1시간 동안 동일한 인증 방법으로 다시 로그인할 수 있습니다. 브라우저 쿠키로 로그인하면 앱 클라이언트 구성에 지정된 기간 동안 지속되는 새 토큰을 가져옵니다. 사용자 속성 또는 인증 요소를 변경해도 브라우저 쿠키로 다시 로그인하는 기능에는 영향을 미치지 않습니다.

세션 쿠키를 사용한 인증은 쿠키 기간을 추가 시간으로 재설정하지 않습니다. 사용자가 가장 최근에 성공한 대화형 인증 후 1시간 이상 지난 후 로그인 페이지에 액세스하려고 하면 다시 로그인해야 합니다.

사용자 계정 확인 및 사용자 속성 확인

사용자 풀 로컬 사용자의 경우, 관리형 로그인과 호스팅 UI는 사용자 풀을 Cognito에서 검증 및 확인을 위해 자동으로 메시지를 보내도록 허용으로 구성하면 가장 효과적으로 작동합니다. 이 설정을 활성화하면 Amazon Cognito는 가입한 사용자에게 확인 코드가 포함된 메시지를 보냅니다. 대신 사용자를 사용자 풀 관리자로 확인하면 가입 후 로그인 페이지에 오류 메시지가 표시됩니다. 이 상태에서 Amazon Cognito는 새 사용자를 생성했지만 검증 메시지를 보낼 수 없습니다. 여전히 사용자를 관리자로 확인할 수 있지만 오류가 발생할 경우 사용자가 지원 데스크에 문의할 수 있습니다. 관리 확인에 대한 자세한 내용은 사용자가 앱에 가입할 수 있도록 허용하지만 사용자 풀 관리자로 확인 섹션을 참조하세요.

관리형 로그인 작업 범위

관리형 로그인 및 클래식 호스팅 UI는 가입, 로그인 및 암호 관리를 지원합니다. 여기에는 다중 인증(MFA)으로 로그인 완료 및 webAuthn 인증자 등록이 포함됩니다. 관리형 로그인은 속성 변경 및 MFA 기본 설정과 같은 사용자 셀프 서비스 프로파일 관리를 지원하지 않습니다. 자체 애플리케이션 코드에서 프로필 관리를 구현해야 합니다. 관리형 로그인은 AdminUpdateUserAttributes API 작업을 통해 관리자로 이메일 주소와 전화번호를 업데이트할 때 속성 변경을 확인할 수 있는 용량도 제공하지 않습니다.

구성 변경 사항 보기

페이지의 스타일을 변경했는데 바로 적용되지 않으면 몇 분간 기다렸다가 페이지를 새로 고칩니다.

사용자 풀 토큰 디코딩

Amazon Cognito 사용자 풀 토큰은 RS256 알고리즘을 사용하여 서명됩니다. 를 사용하여 사용자 풀 토큰을 디코딩하고 확인할 수 있습니다 AWS Lambda. GitHub에서 Amazon Cognito JWT 토큰 디코딩 및 확인을 참조하세요.

TLS 버전

관리형 로그인 및 호스팅 UI 페이지에는 전송 중 암호화가 필요합니다. Amazon Cognito에서 제공하는 사용자 풀 도메인을 사용하려면 사용자 브라우저에서 최소 TLS 버전 1.2가 필요합니다. 사용자 지정 도메인은 TLS 버전 1.2와의 브라우저 연결을 지원합니다. 호스팅 UI(클래식)에는 사용자 지정 도메인에 TLS 1.2가 필요하지 않지만 최신 관리형 로그인에는 사용자 지정 도메인과 접두사 도메인 모두에 TLS 버전 1.2가 필요합니다. Amazon Cognito가 도메인 서비스 구성을 관리하므로 사용자 풀 도메인의 TLS 요구 사항을 수정할 수 없습니다.

CORS 정책

관리형 로그인이나 호스팅 UI는 모두 사용자 지정 CORS(Cross-Origin Resource Sharing) 원본 정책을 지원하지 않습니다. CORS 정책은 사용자가 요청에 인증 파라미터를 전달하지 못하도록 합니다. 대신, 애플리케이션 프런트엔드에 CORS 정책을 구현하세요. Amazon Cognito는 다음 엔드포인트에 대한 요청에 `Access-Control-Allow-Origin: *` 응답 헤더를 반환합니다.

1. Token 엔드포인트
2. 취소 엔드포인트
3. userInfo 엔드포인트

쿠키

관리형 로그인 및 호스팅 UI는 사용자 브라우저에서 쿠키를 설정합니다. 쿠키는 사이트에서 타사 쿠키를 설정하지 않는 일부 브라우저의 요구 사항을 준수합니다. 이는 사용자 풀 엔드포인트에만 적용되며 다음을 포함합니다.

- 각 요청에 대한 `XSRF-TOKEN` 쿠키.
- 사용자가 리디렉션될 때 세션 일관성을 위한 `csrf-state` 쿠키.
- 세션 일관성을 위한 `csrf-state-legacy` 쿠키로, 브라우저가 `SameSite` 속성을 지원하지 않는 경우 Amazon Cognito에서 폴백으로 읽습니다.
- 성공적인 로그인 시도를 1시간 동안 보관하는 `cognito` 세션 쿠키.

- 관리형 로그인에서 사용자가 선택한 언어 현지화를 유지하는 `lang` 쿠키입니다.
- 사용자가 관리형 로그인 페이지를 탐색할 때 필요한 데이터를 유지하는 `page-data` 쿠키입니다.

iOS에서는 모든 쿠키를 차단할 수 있습니다. 이 설정은 관리형 로그인 또는 호스팅 UI와 호환되지 않습니다. 이 설정을 활성화할 수 있는 사용자와 함께 작업하려면 AWS SDK를 사용하여 네이티브 iOS 앱에 사용자 풀 인증을 빌드합니다. 이 시나리오에서는 쿠키 기반이 아닌 고유한 세션 스토리지를 구축할 수 있습니다.

관리형 로그인 버전 변경의 영향

도메인을 추가하고 관리형 로그인 버전을 설정할 때 다음과 같은 영향을 고려합니다.

- 관리형 로그인 또는 호스팅 UI(클래식) 브랜딩을 사용하여 접두사 도메인을 추가하면 로그인 페이지를 사용할 수 있기까지 최대 60초가 걸릴 수 있습니다.
- 관리형 로그인 또는 호스팅 UI(클래식) 브랜딩을 사용하여 사용자 지정 도메인을 추가하면 로그인 페이지를 사용할 수 있기까지 최대 5분이 걸릴 수 있습니다.
- 도메인의 브랜딩 버전을 변경하면 새 브랜딩 버전에서 로그인 페이지를 사용할 수 있기까지 최대 4분이 걸릴 수 있습니다.
- 관리형 로그인과 호스팅 UI(클래식) 브랜딩 간에 전환하면 Amazon Cognito는 사용자 세션을 유지하지 않습니다. 새 인터페이스로 다시 로그인해야 합니다.

기본 스타일

에서 앱 클라이언트를 생성하면 AWS Management Console Amazon Cognito는 앱 클라이언트에 브랜딩 스타일을 자동으로 할당합니다. CreateUserPoolClient 작업을 사용하여 프로그래밍 방식으로 앱 클라이언트를 생성하면 브랜딩 스타일이 생성되지 않습니다. CreateManagedLoginBranding 요청으로 생성할 때까지 AWS SDK로 생성된 앱 클라이언트에는 관리형 로그인을 사용할 수 없습니다.

관리형 로그인 인증 프롬프트 제한 시간

Amazon Cognito는 5분 이내에 완료되지 않는 인증 요청을 취소하고 사용자를 관리형 로그인으로 리디렉션합니다. 페이지에 `Something went wrong` 오류 메시지가 표시됩니다.

사용자 풀 도메인 구성

도메인 구성은 사용자 풀 설정의 선택적 부분입니다. 사용자 풀 도메인은 사용자 인증, 타사 공급자와의 페더레이션 및 OpenID Connect(OIDC) 흐름을 위한 기능을 호스팅합니다. 여기에는 가입, 로그인 및 암호 복구와 같은 키 작업을 위한 사전 구축된 인터페이스인 관리형 로그인이 있습니다. 또한 기계

간(M2M) 권한 부여 및 기타 OIDC 및 OAuth 2.0 인증 및 권한 부여 흐름을 위해 권한 부여, userInfo 및 토큰과 같은 표준 OpenID Connect(OIDC) 엔드포인트를 호스팅합니다.

M2M 권한 부여는 도메인 없이도 사용할 수 있습니다. 토큰 엔드포인트 대신 Amazon Cognito API를 사용하는 GetClientToken API 작업을 사용하여 클라이언트 자격 증명 액세스 토큰을 요청할 수 있습니다.

사용자는 사용자 풀과 연결된 도메인에서 관리형 로그인 페이지로 인증합니다. 이 도메인을 구성하는 방법에는 두 가지가 있습니다. 기본 Amazon Cognito 접두사 도메인을 사용하거나 소유한 사용자 지정 도메인을 구성할 수 있습니다.

사용자 지정 도메인 옵션에는 유연성, 보안 및 제어를 위한 더 많은 옵션이 있습니다. 예를 들어, 친숙한 조직 소유 도메인은 사용자의 신뢰를 높이고 로그인 프로세스를 더욱 직관적으로 만들 수 있습니다. 하지만 사용자 지정 도메인 접근 방식에는 SSL 인증서 및 DNS 구성 관리와 같은 몇 가지 추가 오버헤드가 필요합니다.

OIDC 검색 엔드포인트, 엔드포인트 URL용 `/.well-known/openid-configuration` 및 토큰 서명 키용 `/.well-known/jwks.json`는 도메인에서 호스팅되지 않습니다. 자세한 내용은 ID 제공업체 및 신뢰 당사자 엔드포인트 섹션을 참조하세요.

사용자 풀의 도메인을 구성하고 관리하는 방법을 이해하는 것은 애플리케이션에 인증을 통합하는 데 중요한 단계입니다. 사용자 풀 API 및 AWS SDK를 사용한 로그인은 도메인 구성의 대안이 될 수 있습니다. API 기반 모델은 API 응답에서 토큰을 직접 전달하지만 사용자 풀의 확장된 기능을 OIDC IdP로 사용하는 구현의 경우 도메인을 구성해야 합니다. 사용자 풀에서 사용할 수 있는 인증 모델에 대한 자세한 내용은 API, OIDC 및 관리형 로그인 페이지 인증 이해 섹션을 참조하세요.

주제

- 사용자 풀 도메인에 대해 알아야 할 사항
- 관리형 로그인에 Amazon Cognito 접두사 도메인 사용
- 관리형 로그인에 자체 도메인 사용

사용자 풀 도메인에 대해 알아야 할 사항

사용자 풀 도메인은 애플리케이션의 OIDC 신뢰 당사자와 UI 요소에 대한 서비스 지점입니다. 사용자 풀에 대한 도메인 구현을 계획할 때는 다음 세부 정보를 고려하세요.

예약 용어

Amazon Cognito 접두사 도메인 이름에는 `aws`, `amazon` 또는 `cognito` 텍스트를 사용할 수 없습니다.

검색 엔드포인트가 다른 도메인에 있음

사용자 풀 검색 엔드포인트 `.well-known/openid-configuration` 및 `.well-known/jwks.json`은 사용자 풀 사용자 지정 또는 접두사 도메인에 없습니다. 이러한 엔드포인트의 경로는 다음과 같습니다.

- `https://cognito-idp.Region.amazonaws.com/your user pool ID/.well-known/openid-configuration`
- `https://cognito-idp.Region.amazonaws.com/your user pool ID/.well-known/jwks.json`

도메인 변경 유효 시간

Amazon Cognito가 접두사 도메인의 브랜딩 버전을 시작하거나 업데이트하는 데 최대 1분이 걸릴 수 있습니다. 사용자 지정 도메인에 대한 변경 사항이 전파되는 데 최대 5분이 걸릴 수 있습니다. 새 사용자 지정 도메인이 전파되는 데 최대 1시간이 걸릴 수 있습니다.

사용자 지정 도메인과 접두사 도메인을 동시에 사용

사용자 지정 도메인과가 소유한 접두사 도메인을 모두 사용하여 사용자 풀을 설정할 수 있습니다 AWS. 사용자 풀 검색 엔드포인트는 다른 도메인에서 호스팅되므로 사용자 지정 도메인에만 제공됩니다. 예를 들어는 `"authorization_endpoint"`의 단일 값을 `openid-configuration` 제공합니다`"https://auth.example.com/oauth2/authorize"`.

사용자 풀에 사용자 지정 도메인과 접두사 도메인이 모두 있는 경우 OIDC 공급자의 전체 기능과 함께 사용자 지정 도메인을 사용할 수 있습니다. 이 구성을 사용하는 사용자 풀의 접두사 도메인에는 검색 또는 token-signing-key 엔드포인트가 없으므로 그에 따라 사용해야 합니다.

패스키에 대한 신뢰 당사자 ID로 선호되는 사용자 지정 도메인

패스키를 사용하여 사용자 풀 인증을 설정할 때는 신뢰 당사자(RP) ID를 설정해야 합니다. 사용자 지정 도메인과 접두사 도메인이 있는 경우 RP ID만 사용자 지정 도메인으로 설정할 수 있습니다. Amazon Cognito 콘솔에서 접두사 도메인을 RP ID로 설정하려면 사용자 지정 도메인을 삭제하거나 접두사 도메인의 정규화된 도메인 이름(FQDN)을 타사 도메인으로 입력합니다.

도메인 계층 구조의 다른 수준에서 사용자 지정 도메인을 사용하지 마십시오.

별도의 사용자 풀을 구성하여 동일한 최상위 도메인(TLD)에 사용자 지정 도메인(예: auth.example.com 및 auth2.example.com)을 가질 수 있습니다. 관리형 로그인 세션 쿠키는 사용자 지정 도메인 및 모든 하위 도메인(예: *.auth.example.com)에 유효합니다. 이로 인해 애플리케이션의 어

떤 사용자도 상위 도메인 및 하위 도메인의 관리형 로그인에 액세스해서는 안 됩니다. 사용자 지정 도메인이 동일한 TLD를 사용하는 경우 동일한 하위 도메인 수준을 유지합니다.

사용자 지정 도메인 auth.example.com이 있는 사용자 풀이 있다고 가정해 보겠습니다. 그런 다음 다른 사용자 풀을 생성하고 사용자 지정 도메인 uk.auth.example.com.을 할당합니다. 사용자는 auth.example.com으로 로그인하고 브라우저가 와일드카드 경로 *.auth.example.com에 있는 모든 웹 사이트에 제공하는 쿠키를 받습니다. 그런 다음 uk.auth.example.com.에 로그인합니다. 이는 잘못된 쿠키를 사용자 풀 도메인에 전달하고 로그인 프롬프트 대신 오류를 수신합니다. 반면, *.auth.example.com에 대한 쿠키가 있는 사용자는 auth2.example.com에서 로그인 세션을 시작하는 데 문제가 없습니다.

브랜딩 버전

도메인을 생성할 때 브랜딩 버전을 설정합니다. 옵션은 최신 관리형 로그인 환경과 클래식 호스팅 UI 환경입니다. 이 선택 사항은 도메인에서 서비스를 호스팅하는 모든 앱 클라이언트에 적용됩니다.

관리형 로그인에 Amazon Cognito 접두사 도메인 사용

관리형 로그인의 기본 환경은가 AWS 소유한 도메인에서 호스팅됩니다. 이 방식은 접두사 이름을 선택하면 활성화되기 때문에 진입 장벽이 낮지만 사용자 지정 도메인의 신뢰감을 주는 기능은 없습니다. Amazon Cognito 도메인 옵션과 사용자 지정 도메인 옵션 간에는 비용 차이가 없습니다. 유일한 차이점은 사용자를 지시하는 웹 주소의 도메인입니다. 타사 IdP 리디렉션 및 클라이언트 자격 증명 흐름의 경우 접두사 도메인은 거의 눈에 보이지 않습니다. 사용자 지정 도메인은 사용자가 관리형 로그인으로 로그인하고 애플리케이션 도메인과 일치하지 않는 인증 도메인과 상호 작용하는 경우에 더 적합합니다.

접두사 도메인에는 선택한 접두사가 있으며 리전 `amazoncognito.com` 인증 도메인의 하위 도메인입니다. 다음은 예제입니다.

```
https://cognitoexample.auth.ap-south-1.amazoncognito.com
```

모든 접두사 도메인은 *prefix*.auth.*AWS ## code*.amazoncognito.com 형식을 따릅니다. 사용자 지정 도메인 사용자 풀은 사용자가 소유한 모든 도메인에서 관리되는 로그인이나 호스팅 UI 페이지를 호스팅할 수 있습니다.

> **Note**
>
> Amazon Cognito 애플리케이션의 보안을 강화하기 위해 사용자 풀 엔드포인트의 상위 도메인은 PSL(Public Suffix List)에 등록됩니다. PSL은 사용자의 웹 브라우저가 사용자 풀 엔드포인트와 해당 엔드포인트가 설정하는 쿠키를 일관되게 이해하는 데 도움이 됩니다.

사용자 풀 상위 도메인은 다음과 같은 형식을 갖습니다.

```
auth.Region.amazoncognito.com
auth-fips.Region.amazoncognito.com
```

`auth-fips` 도메인은 FIPS(Federal Information Processing Standard) 140-3 준수 엔드포인트입니다. FIPS 검증 암호화 모듈을 사용하여 전송 중 암호화를 제공합니다. 애플리케이션이 FIPS 규정 준수 요구 사항을 충족해야 하는 경우 사용자를이 엔드포인트로 안내합니다. Amazon Cognito 관리형 로그인을 위한 FIPS 엔드포인트는 미국의 일부 AWS 리전에서 사용할 수 있으며 일부 리전에서는 제공되지 않습니다. FIPS 및 Amazon Cognito가 FIPS 엔드포인트를 제공하는 리전에 대한 자세한 내용은 FIPS 140-3을 참조하세요.

를 사용하여 앱 클라이언트와 사용자 풀 도메인을 추가하려면 섹션을 AWS Management Console참조하세요앱 클라이언트 생성.

주제

- 사전 조건
- Amazon Cognito 도메인 접두사 구성
- 로그인 페이지 확인

사전 조건

시작하려면 다음이 필요합니다.

- 앱 클라이언트가 포함된 사용자 풀. 자세한 내용은 Amazon Cognito 콘솔에서 새 애플리케이션 생성 섹션을 참조하세요.

Amazon Cognito 도메인 접두사 구성

AWS Management Console 또는 API를 사용하여 사용자 풀 도메인을 AWS CLI 구성할 수 있습니다.

Amazon Cognito console

도메인 구성

1. 브랜딩 아래의 도메인 메뉴로 이동합니다.

2. 도메인 옆에서 작업을 선택하고 Cognito 도메인 생성을 선택합니다. 이미 사용자 풀 접두사 도메인을 구성한 경우 새 사용자 지정 도메인을 만들기 전에 Cognito 도메인 삭제를 선택합니다.
3. Amazon Cognito 도메인에서 사용할 사용 가능한 도메인 접두사를 입력합니다. 사용자 지정 도메인 설정에 대한 자세한 내용은 관리형 로그인에 자체 도메인 사용 섹션을 참조하세요.
4. 브랜딩 버전을 선택합니다. 브랜딩 버전은 해당 도메인의 모든 사용자 대화형 페이지에 적용됩니다. 사용자 풀은 모든 앱 클라이언트에 대해 관리형 로그인 또는 호스팅 UI 브랜딩을 호스팅할 수 있습니다.

> **Note**
>
> 사용자 지정 도메인과 접두사 도메인을 가질 수 있지만 Amazon Cognito는 사용자 지정 도메인의 `/.well-known/openid-configuration` 엔드포인트만 제공합니다.

5. 생성을 선택합니다.

CLI/API

다음 명령을 사용하여 도메인 접두사를 생성하고 사용자 풀에 할당합니다.

사용자 풀 도메인을 구성하려면

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

  예: `aws cognito-idp create-user-pool-domain --user-pool-id <user_pool_id> --domain <domain_name> --managed-login-version 2`
- 사용자 풀 API 작업: CreateUserPoolDomain

도메인 정보를 가져오려면

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

  예: `aws cognito-idp describe-user-pool-domain --domain <domain_name>`
- 사용자 풀 API 작업: DescribeUserPoolDomain

도메인을 삭제하려면

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

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

- 사용자 풀 API 작업: DeleteUserPoolDomain

로그인 페이지 확인

- Amazon Cognito 호스트된 도메인에서 로그인 페이지를 사용할 수 있는지 확인합니다.

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

도메인은 Amazon Cognito 콘솔의 [도메인 이름(Domain name)] 페이지에서 표시됩니다. 앱 클라이언트 설정(App client settings) 페이지에 앱 클라이언트 ID와 콜백 URL이 표시됩니다.

관리형 로그인에 자체 도메인 사용

앱 클라이언트를 설정한 후 관리형 로그인의 도메인 서비스에 대한 사용자 지정 도메인으로 사용자 풀을 구성할 수 있습니다. 사용자 지정 도메인을 사용하면 사용자는 기본 `amazoncognito.com` 접두사 도메인 대신 자체 웹 주소를 사용하여 애플리케이션에 로그인할 수 있습니다. 사용자 지정 도메인은 특히 루트 도메인이 애플리케이션을 호스팅하는 도메인과 일치하는 경우 익숙한 도메인 이름을 사용하여 애플리케이션에 대한 사용자 신뢰를 향상시킵니다. 사용자 지정 도메인을 사용하면 조직의 보안 요구 사항을 더욱 효과적으로 준수할 수 있습니다.

사용자 지정 도메인에는 사용자 풀, 앱 클라이언트, 소유한 웹 도메인을 비롯한 몇 가지 전제 조건이 있습니다. 또한 사용자 지정 도메인에는 미국 동부(버지니아 북부)의 AWS Certificate Manager (ACM)로 관리되는 사용자 지정 도메인에 대한 SSL 인증서가 필요합니다. Amazon Cognito는 ACM 인증서로 전송 중에 보안되는 Amazon CloudFront 배포를 생성합니다. 도메인을 소유하므로 트래픽을 사용자 지정 도메인의 CloudFront 배포로 보내는 DNS 레코드를 생성해야 합니다.

이러한 요소가 준비되면 Amazon Cognito 콘솔 또는 API를 통해 사용자 풀에 사용자 지정 도메인을 추가할 수 있습니다. 여기에는 도메인 이름과 SSL 인증서를 지정한 다음 제공된 별칭 대상으로 DNS 구성을 업데이트하는 작업이 포함됩니다. 이러한 변경 사항을 적용한 후 사용자 지정 도메인에서 로그인 페이지에 액세스할 수 있는지 확인할 수 있습니다.

사용자 지정 도메인을 생성하는 가장 낮은 방법은 Amazon Route 53의 퍼블릭 호스팅 영역을 사용하는 것입니다. Amazon Cognito 콘솔은 몇 단계로 올바른 DNS 레코드를 생성할 수 있습니다. 시작하기 전에 소유한 도메인 또는 하위 도메인에 대한 Route 53 호스팅 영역을 생성하는 것이 좋습니다.

주제

- 사용자 풀에 사용자 정의 도메인 추가
- 사전 조건
- 1단계: 사용자 정의 도메인 이름 입력
- 2단계: 별칭 대상 및 하위 도메인 추가
- 3단계: 로그인 페이지 확인
- 사용자 정의 도메인의 SSL 인증서 변경
- 사용자 지정 도메인의 최소 TLS 버전 설정

사용자 풀에 사용자 정의 도메인 추가

사용자 풀에 사용자 지정 도메인을 추가하려면 Amazon Cognito 콘솔에서 도메인 이름을 지정하고 AWS Certificate Manager(ACM)을 사용하여 관리하는 인증서를 제공합니다. 도메인을 추가하고 나면 Amazon Cognito는 사용자가 DNS 구성에 추가할 별칭 대상을 제공합니다.

사전 조건

시작하려면 다음이 필요합니다.

- 앱 클라이언트가 포함된 사용자 풀. 자세한 내용은 Amazon Cognito 콘솔에서 새 애플리케이션 생성 섹션을 참조하세요.
- 자신이 소유한 웹 도메인. 상위 도메인에 유효한 DNS A 레코드가 있어야 합니다. 이 레코드에 어떤 값이든 할당할 수 있습니다. 상위 도메인은 도메인의 루트 도메인이거나 도메인 계층 구조에서 한 단계 높은 자식 도메인일 수 있습니다. 예를 들어 사용자 지정 도메인이 auth.xyz.example.com인 경우 Amazon Cognito는 xyz.example.com을 IP 주소로 확인할 수 있어야 합니다. 고객 인프라에 대한 우발적인 영향을 방지하기 위해 Amazon Cognito는 사용자 지정 도메인에 대한 최상위 도메인(TLD) 사용을 지원하지 않습니다. 자세한 내용은 도메인 이름을 참조하세요.
- 사용자 지정 도메인의 하위 도메인을 생성하는 기능 하위 도메인 이름에 대한 인증을 권장합니다. 예: *auth.example.com*.

  > **Note**
  >
  > 와일드카드 인증서가 없는 경우 사용자 지정 도메인의 하위 도메인에 대한 새 인증서를 받아야 할 수 있습니다.

- 미국 동부(버지니아 북부)의 ACM에서 관리하는 퍼블릭 SSL/TLS 인증서입니다. 인증서는 글로벌 서비스인 CloudFront의 배포와 연결되어 있으므로 us-east-1에 있어야 합니다.

- 서버 이름 표시(SNI)를 지원하는 브라우저 클라이언트입니다. Amazon Cognito가 사용자 지정 도메인에 할당하는 CloudFront 배포에는 SNI가 필요합니다. 이 설정은 변경할 수 없습니다. CloudFront 배포의 SNI에 대한 자세한 내용은 Amazon CloudFront 개발자 안내서의 SNI를 사용하여 HTTPS 요청 제공을 참조하세요.
- 사용자 풀 권한 부여 서버가 사용자 세션에 쿠키를 추가할 수 있도록 허용하는 애플리케이션입니다. Amazon Cognito는 관리형 로그인 페이지에 필요한 여러 쿠키를 설정합니다. `cognito`, `cognito-fl` 및 `XSRF-TOKEN`가 포함됩니다. 각 개별 쿠키가 브라우저 크기 제한을 준수하지만 사용자 풀 구성을 변경하면 관리형 로그인 쿠키의 크기가 증가할 수 있습니다. 사용자 지정 도메인 앞에 있는 Application Load Balancer(ALB)와 같은 중간 서비스는 최대 헤더 크기 또는 총 쿠키 크기를 적용할 수 있습니다. 애플리케이션이 자체 쿠키도 설정하면 사용자의 세션이 이러한 제한을 초과할 수 있습니다. 크기 제한 충돌을 방지하려면 애플리케이션이 사용자 풀 도메인 서비스를 호스팅하는 하위 도메인에 쿠키를 설정하지 않는 것이 좋습니다.
- Amazon CloudFront 배포를 업데이트할 수 있는 권한입니다. 다음 IAM 정책 문을 AWS 계정의 IAM 사용자, 그룹 또는 역할에 연결하면 이 작업을 할 수 있습니다.

  JSON

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

  CloudFront의 작업 권한 부여에 대한 자세한 내용은 Using Identity-Based Policies (IAM Policies) for CloudFront(CloudFront에 자격 증명 기반 정책(IAM 정책) 사용)를 참조하세요.

  Amazon Cognito는 처음에 IAM 권한을 사용하여 CloudFront 배포를 구성하지만 배포는 AWS에서 관리합니다. Amazon Cognito가 사용자 풀과 연결한 CloudFront 배포의 구성을 변경할 수 없습니다. Amazon Cognito API를 사용하여 사용자 지정 도메인을 생성하거나 업데이트할 때 배포 보안 정책

의 최소 TLS 버전을 선택할 수 있습니다. 자세한 내용은 사용자 지정 도메인의 최소 TLS 버전 설정 섹션을 참조하세요.

1단계: 사용자 정의 도메인 이름 입력

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

Amazon Cognito console

Amazon Cognito 콘솔에서 사용자 풀에 도메인을 추가하려면

1. 브랜딩 아래의 도메인 메뉴로 이동합니다.
2. 도메인 옆의 작업을 선택한 다음, 사용자 지정 도메인 생성 또는 Amazon Cognito 도메인 생성을 선택합니다. 이미 사용자 풀 사용자 지정 도메인을 구성한 경우 새 사용자 지정 도메인을 만들기 전에 사용자 지정 도메인 삭제를 선택합니다.
3. 도메인 옆에서 작업을 선택하고 사용자 지정 도메인 생성을 선택합니다. 이미 사용자 지정 도메인을 구성한 경우 새 사용자 지정 도메인을 만들기 전에 사용자 지정 도메인 삭제를 선택하여 기존 도메인을 삭제합니다.
4. [사용자 정의 도메인(Custom domain)]에 Amazon Cognito에서 사용할 도메인의 URL을 입력합니다. 도메인 이름에는 소문자, 숫자, 하이픈(-)만 사용할 수 있습니다. 첫 번째나 마지막 자리에 하이픈을 사용해서는 안 됩니다. 마침표로 하위 도메인 이름을 구분하세요.
5. [ACM 인증서(ACM certificate)]에서 도메인에 사용할 SSL 인증서를 선택합니다. 사용자 풀과 관계없이 미국 동부(버지니아 북부) AWS 리전 의 ACM 인증서만 Amazon Cognito 사용자 지정 도메인과 함께 사용할 수 있습니다.

   사용 가능한 인증서가 없는 경우 ACM을 사용하여 미국 동부(버지니아 북부)에서 인증서를 프로비저닝할 수 있습니다. 자세한 내용은 AWS Certificate Manager 사용 설명서에서 시작하기를 참조하세요.
6. 브랜딩 버전을 선택합니다. 브랜딩 버전은 해당 도메인의 모든 사용자 대화형 페이지에 적용됩니다. 사용자 풀은 모든 앱 클라이언트에 대해 관리형 로그인 또는 호스팅 UI 브랜딩을 호스팅할 수 있습니다.

> **Note**
>
> 사용자 지정 도메인과 접두사 도메인을 가질 수 있지만 Amazon Cognito는 사용자 지정 도메인의 `/.well-known/openid-configuration` 엔드포인트만 제공합니다.

7. 생성을 선택합니다.
8. Amazon Cognito가 도메인 메뉴로 돌아갑니다. [도메인의 DNS에 별칭 레코드 생성(Create an alias record in your domain's DNS)]이라는 제목의 메시지가 표시됩니다. 콘솔에 표시된 [도메인(Domain)]과 [별칭 대상(Alias target)]을 적어 둡니다. 이 정보는 다음 단계에서 트래픽을 사용자 정의 도메인으로 보내는 데 사용됩니다.

API

다음 CreateUserPoolDomain 요청 본문은 사용자 지정 도메인을 생성합니다.

```
{
   "Domain": "auth.example.com",
   "UserPoolId": "us-east-1_EXAMPLE",
   "ManagedLoginVersion": 2,
   "CustomDomainConfig": {
    "CertificateArn": "arn:aws:acm:us-east-1:111122223333:certificate/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "SecurityPolicy": "TLS_V1_3_2025"
   }
}
```

선택적 `SecurityPolicy` 필드는 사용자 지정 도메인의 최소 TLS 버전을 설정합니다. 자세한 내용은 사용자 지정 도메인의 최소 TLS 버전 설정 섹션을 참조하세요.

2단계: 별칭 대상 및 하위 도메인 추가

이 단계에서는 도메인 이름 서버(DNS) 서비스 공급자를 통해, 이전 단계의 별칭 대상을 다시 가리키는 별칭을 설정합니다. DNS 주소 확인에 Amazon Route 53을 사용하는 경우 Route 53을 사용하여 별칭 대상 및 하위 도메인을 추가하려면 섹션을 선택하세요.

현재 DNS 구성에 별칭 대상 및 하위 도메인을 추가하려면

- DNS 주소 확인에 Route 53을 사용하지 않는 경우 DNS 서비스 공급자의 구성 도구를 사용하여 이전 단계의 별칭 대상을 도메인의 DNS 레코드에 추가해야 합니다. 또한 DNS 공급자가 사용자 지정 도메인에 대한 하위 도메인을 설정해야 합니다.

Route 53을 사용하여 별칭 대상 및 하위 도메인을 추가하려면

1. Route 53 콘솔에 로그인합니다. 메시지가 표시되면 자격 AWS 증명을 입력합니다.

2. Route 53에 퍼블릭 호스팅 영역이 없는 경우 루트가 사용자 지정 도메인의 상위인 영역을 생성합니다. 자세한 내용은 Amazon Route 53 개발자 안내서의 퍼블릭 호스팅 영역 생성을 참조하세요.

   a. Create Hosted Zone(호스팅 영역 생성)을 선택합니다.

   b. [도메인 이름(Domain Name)] 목록에서 사용자 정의 도메인(예: *myapp.auth.example.com*)의 상위 도메인(예: *auth.example.com*)을 입력합니다.

   c. 호스팅 영역에 대한 설명을 입력합니다.

   d. 퍼블릭 클라이언트가 사용자 정의 도메인을 확인할 수 있도록 호스팅 영역 [유형(Type)]으로 [퍼블릭 호스팅 영역(Public hosted zone)]을 선택합니다. [프라이빗 호스팅 영역(Private hosted zone)]은 선택할 수 없습니다.

   e. 원하는 대로 [태그(Tags)]를 적용합니다.

   f. 호스팅 영역 생성(Create hosted zone)을 선택합니다.

   > **Note**
   >
   > 또한 하위 도메인 호스팅 영역으로 쿼리를 전달하는 상위 호스팅 영역에 위임을 설정하여 사용자 지정 도메인에 대한 새 호스팅 영역을 생성할 수 있습니다. 그렇지 않으면 A 레코드를 생성합니다. 이 방법은 호스팅 영역에서 더 많은 유연성과 보안을 제공합니다. 자세한 내용은 Amazon Route 53을 통해 호스팅되는 도메인의 하위 도메인 생성을 참조하세요.

3. 호스팅 영역(Hosted Zones) 페이지에서 호스팅 영역의 이름을 선택합니다.

4. 사용자 지정 도메인의 상위 도메인이 아직 없는 경우 DNS 레코드를 추가합니다. 다음 속성을 사용하여 상위 도메인에 대한 DNS 레코드를 생성합니다.

   - 레코드 이름: 비워 둡니다.
   - 레코드 유형: A.
   - 별칭 : 활성화하지 않습니다.
   - 값 : 선택한 대상을 입력합니다. 이 레코드는 무언가로 확인되어야 하지만 레코드의 값은 Amazon Cognito에 중요하지 않습니다.
   - TTL: 원하는 TTL로 설정하거나 기본값으로 그대로 둡니다.
   - 라우팅 정책: 단순 라우팅을 선택합니다.

5. 레코드 생성을 선택합니다. 다음은 *example.com* 도메인에 대한 예제 레코드입니다.

   ```
   example.com. 60 IN A 198.51.100.1
   ```

> **Note**
>
> Amazon Cognito는 프로덕션 도메인이 실수로 하이재킹되지 않도록 보호하기 위해 사용자 정의 도메인의 상위 도메인에 대한 DNS 레코드가 있는지 확인합니다. 상위 도메인에 대한 DNS 레코드가 없는 경우 사용자 지정 도메인을 설정하려고 하면 Amazon Cognito가 오류를 반환합니다. 권한 시작(SOA) 레코드는 상위 도메인 확인을 위한 DNS 레코드로 충분하지 않습니다.

6. 다음 속성을 사용하여 사용자 지정 도메인에 대한 다른 DNS 레코드를 추가합니다.

   - 레코드 이름: 사용자 지정 도메인 접두사(예: `auth.example.com`에 대한 레코드를 만들기 위한 `auth`).
   - 레코드 유형: `A`.
   - 별칭: 활성화합니다.
   - 트래픽 라우팅 대상: Cloudfront 배포에 대한 별칭을 선택합니다. 이전에 기록한 별칭 대상(예: `123example.cloudfront.net`)을 입력합니다.
   - 라우팅 정책: 단순 라우팅을 선택합니다.

7. 레코드 생성을 선택합니다.

> **Note**
>
> 새 레코드가 모든 Route 53 DNS 서버에 전파되는 데 약 60초가 걸릴 수 있습니다. Route 53 GetChange API 메서드를 사용하여 변경 사항이 전파되었는지 확인할 수 있습니다.

3단계: 로그인 페이지 확인

- 사용자 지정 도메인에서 로그인 페이지를 사용할 수 있는지 확인합니다.

  브라우저에 이 주소를 입력하여 사용자 지정 도메인과 하위 도메인으로 로그인합니다. 다음은 사용자 지정 도메인이 *example.com*이고 하위 도메인이 *auth*인 URL의 예입니다.

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

사용자 정의 도메인의 SSL 인증서 변경

필요한 경우 Amazon Cognito를 사용하여 사용자 지정 도메인에 적용한 인증서를 변경할 수 있습니다.

일반적으로 ACM을 사용한 일상적인 인증서 갱신 후에는 이 작업이 필요하지 않습니다. ACM에서 기존 인증서를 갱신할 때 인증서의 ARN은 동일하게 유지되며 사용자 지정 도메인은 새 인증서를 자동으로 사용합니다.

그러나 기존 인증서를 새 인증서로 교체하는 경우에는 ACM이 새 인증서에 새 ARN을 할당합니다. 사용자 지정 도메인에 새 인증서를 적용하려면 Amazon Cognito에 이 ARN을 제공해야 합니다.

새 인증서를 제공한 후, Amazon Cognito가 사용자 지정 도메인에 인증서를 배포하는 데 최대 1시간이 소요됩니다.

> **시작하기 전 준비 사항**
>
> Amazon Cognito에서 인증서를 변경하려면 먼저 ACM에 인증서를 추가해야 합니다. 자세한 내용은 AWS Certificate Manager 사용 설명서에서 시작하기를 참조하세요.
> 인증서를 ACM에 추가할 때 미국 동부(버지니아 북부)를 AWS 리전으로 선택해야 합니다.

Amazon Cognito 콘솔 또는 API를 사용하여 인증서를 변경할 수 있습니다.

AWS Management Console

Amazon Cognito 콘솔에서 인증서를 갱신하려면

1. 에 로그인 AWS Management Console 하고에서 Amazon Cognito 콘솔을 엽니다https://console.aws.amazon.com/cognito/home.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 인증서를 업데이트할 사용자 풀을 선택합니다.
4. 도메인 메뉴를 선택합니다.
5. [작업(Actions)], [ACM 인증서 편집(Edit ACM certificate)]을 차례로 선택합니다.
6. 사용자 정의 도메인에 연결할 새 인증서를 선택합니다.
7. 변경 사항 저장(Save changes)을 선택합니다.

API

인증서를 갱신하려면(Amazon Cognito API)

- UpdateUserPoolDomain 작업을 사용합니다.

사용자 지정 도메인의 최소 TLS 버전 설정

Amazon Cognito 콘솔 또는 API를 사용하여 사용자 지정 도메인을 생성하거나 업데이트할 때 Amazon CloudFront 배포의 최소 TLS 버전을 선택할 수 있습니다. 배포는 클라이언트와 통신할 때이 버전을 사용합니다. API에서 CreateUserPoolDomain 또는 UpdateUserPoolDomain 요청의 `CustomDomainConfig` 객체 `SecurityPolicy` 필드에 최소 TLS 버전을 설정합니다. 보안 정책은 CloudFront가 클라이언트와 통신할 때 지원하는 최소 TLS 버전 및 암호 제품군을 정의합니다. 구체적인 지침은 Amazon CloudFront 개발자 안내서의 최종 사용자와 CloudFront 간에 지원되는 프로토콜 및 암호를 참조하세요. Amazon CloudFront 다음 값 중 하나를 선택할 수 있습니다.

`TLS_V1_3_2025` (가장 엄격)

TLS 1.3이 필요한 post-quantum-ready 정책입니다. 가장 강력한 보안 태세를 제공하며 모든 클라이언트와 브라우저가 최신 버전으로 업데이트되는 워크로드에 적합합니다. 자세한 내용은 TLSv1.3_2025에 지원되는 프로토콜 및 암호를 참조하세요.

`TLS_V1_2_2021`(권장)

TLS 1.3을 선호하지만 이전 클라이언트를 수용하기 위해 TLS 1.2로의 대체를 허용하는 post-quantum-ready 정책입니다. 일반적인 상용 소비자 애플리케이션에 권장되는 최소 수준입니다. 자세한 내용은 TLSv1.2_2021에 지원되는 프로토콜 및 암호를 참조하세요.

`TLS_V1` (매우 권장되지 않음)

TLS 1.0으로 대체할 수 있습니다. 10년 이상 된 레거시 클라이언트에 대한 지원을 포함하여 가장 광범위한 호환성을 제공합니다. 이러한 호환성은 상용 용도로 더 이상 안전하지 않은 것으로 간주되는 TLS 버전 및 암호화 알고리즘을 허용하는 데 사용됩니다. 자세한 내용은 TLSv1에 지원되는 프로토콜 및 암호를 참조하세요.

> **Note**
>
> 관리형 로그인(브랜딩 버전 2)을 사용하는 사용자 지정 도메인에는 이상의 최소 TLS 버전이 필요합니다`TLS_V1_2_2021`. 관리형 로그인 도메인`TLS_V1`에 대해를 설정하는 요청은 `InvalidInputException` 오류를 반환합니다.

Amazon Cognito 콘솔에서 또는 Amazon Cognito API, AWS Command Line Interface (AWS CLI) 및 AWS SDKs.

다음 UpdateUserPoolDomain 요청 본문은 기존 사용자 지정 도메인의 최소 TLS 버전을 업데이트합니다.

```
{
   "Domain": "auth.example.com",
   "UserPoolId": "us-east-1_EXAMPLE",
   "ManagedLoginVersion": 2,
   "CustomDomainConfig": {
    "CertificateArn": "arn:aws:acm:us-east-1:111122223333:certificate/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "SecurityPolicy": "TLS_V1_3_2025"
   }
}
```

관리형 로그인 페이지에 브랜딩 적용

인증 서비스와 애플리케이션 간에 일관된 사용자 경험을 제공할 수 있습니다. AWS SDK에서 사용자 지정 양식 및 백엔드 API 작업을 사용하거나 관리형 로그인을 사용하여이 목표를 달성할 수 있습니다. 관리형 로그인 및 클래식 호스팅 UI는 사용자 풀로 인증을 제공하는 애플리케이션의 구성 요소에 대한 웹 프런트 엔드입니다. 관리형 인증 서비스를 애플리케이션 UX와 동기화하려면 브랜딩 편집기와 호스팅 UI 브랜딩이라는 두 가지 사용자 지정 옵션이 있습니다. Amazon Cognito 콘솔 및 사용자 풀 API 작업에서 원하는 환경을 선택할 수 있습니다.

브랜딩 편집기

브랜딩 편집기는 최신 사용자 풀 UI 환경인 관리형 로그인을 위한 최신 사용자 지정 옵션입니다. 브랜딩 편집기는 관리형 로그인 자산 및 스타일을 위한 노코드 시각적 편집기이며, 많은 수의 구성 옵션을 프로그래밍 방식으로 구성하기 위한 API 작업 세트입니다. 도메인 및 관리형 로그인으로 구성하는 사용자 풀은 로그인 페이지의 브랜딩 디자이너 버전을 자동으로 렌더링합니다.

호스팅 UI(클래식) 브랜딩

호스팅 UI(클래식) 브랜딩 환경에는 고정된 스타일 옵션 세트로 계단식 스타일시트(CSS) 파일을 수정하고 사용자 지정 로고 이미지를 추가하는 두 가지 옵션이 있습니다. Amazon Cognito 콘솔 또는 SetUICustomization API 작업을 사용하여 이러한 옵션을 설정할 수 있습니다. 서비스가 시작된 시점에 Amazon Cognito에는 이 옵션만 있었습니다. 도메인 및 호스팅 UI 브랜딩 버전으로 구성한 사용자 풀은 로그인 페이지의 클래식 버전을 자동으로 렌더링합니다. 기능 플랜은 호스팅 UI만 지원할 수도 있습니다.

> **Note**
>
> 브랜딩 편집기와 클래식 브랜딩 환경은 호스팅 인증 서비스의 시각적 속성을 수정합니다. 현재는 여러 언어 중 하나에 현지화를 적용하는 경우를 제외하고 관리형 로그인 페이지에 표시되는 텍스트를 수정할 수 없습니다. 현지화에 대한 자세한 내용은 관리형 로그인 현지화 섹션을 참조하세요.

브랜딩 환경 선택 및 스타일 지정

Amazon Cognito 콘솔에서 새 사용자 풀은 기본적으로 관리형 로그인 브랜딩 환경으로 설정됩니다. 관리형 로그인을 사용할 수 있기 전에 설정한 사용자 풀에는 호스팅 UI(클래식) 브랜딩이 있습니다. 관리형 로그인과 호스팅 UI 브랜딩 간에 전환할 수 있습니다. 브랜딩 버전을 변경하면 Amazon Cognito는 변경 사항을 사용자 풀 도메인의 사용자 대화형 페이지에 즉시 적용합니다. 관리형 로그인 및 호스팅 UI를 사용하면 사용자 풀이 각 앱 클라이언트에 대한 스타일을 가질 수 있습니다.

각 앱 클라이언트는 고유한 브랜딩 스타일을 가질 수 있지만 사용자 풀 도메인은 관리형 로그인 또는 호스팅 UI를 제공합니다. 스타일은 앱 클라이언트에 적용되는 사용자 지정 설정 세트입니다. 사용자 풀당 하나의 사용자 지정 도메인과 하나의 접두사 도메인을 설정할 수 있습니다. 사용자 지정 도메인과 접두사 도메인에 다른 브랜딩 버전을 할당할 수 있습니다. 그러나 사용자 지정 도메인도 있는 경우 접두사 도메인이 완전히 작동하지 않습니다. `.well-known` OIDC 검색 엔드포인트에는 사용자 지정 도메인 경로만 표시됩니다. 이 구성의 사용자 풀에서 엔드포인트 검색(`openid-configuration`)이 필요하지 않은 작업에만 접두사 도메인을 사용할 수 있습니다. 이러한 사용자 풀 속성 때문에 사용자 풀당 하나의 브랜딩 버전을 효과적으로 선택할 수 있습니다.

도메인이 관리형 로그인 브랜딩 버전으로 설정된 사용자 풀의 앱 클라이언트에 스타일을 할당할 수 있습니다. 스타일은 이미지 파일, 표시 옵션, CSS 값으로 구성된 시각적 설정 세트입니다. 앱 클라이언트에 스타일을 할당하면 Amazon Cognito는 업데이트를 사용자 대화형 로그인 페이지로 즉시 푸시합니다. Amazon Cognito는 선택한 브랜딩 버전과 사용자가 적용한 사용자 지정으로 사용자 대화형 페이지를 렌더링합니다.

스타일 업데이트 및 삭제

스타일을 생성할 때 앱 클라이언트에 연결합니다. 앱 클라이언트에 대한 스타일 할당을 변경하려면 먼저 원래 스타일을 삭제해야 합니다. 현재는 스타일 간에 설정을 복사할 수 없습니다. 이 작업은 프로그래밍 방식으로 수행해야 합니다. 스타일과 앱 클라이언트 간에 설정을 복제하려면 DescribeManagedLoginBranding API 작업을 사용하여 스타일에 대한 설정을 가져오고 CreateManagedLoginBranding 또는 UpdateManagedLoginBranding을 사용하여 적용합니다. 앱 클라이언트의 할당된 스타일은 변경할 수 없습니다. 원본을 삭제하고 새 스타일만 설정할 수 있습니다. API

및 SDK 작업을 사용하여 스타일을 관리하는 방법에 대한 자세한 내용은 관리형 로그인 브랜딩을 위한 API 및 SDK 작업 섹션을 참조하세요.

> **Note**
>
> 브랜딩 스타일을 생성하거나 업데이트하는 프로그래밍 방식의 요청은 요청 크기가 2MB 이하여야 합니다. 요청이 이 제한보다 크면 최대 `UpdateManagedLoginBranding` 요청 크기를 초과하지 않는 파라미터 그룹에 대한 여러 요청으로 요청을 나눕니다. 이러한 요청으로 인해 지정되지 않은 파라미터가 기본값으로 설정되지 않으므로 기존 설정에 영향을 주지 않고 부분 요청을 보낼 수 있습니다.

Amazon Cognito 콘솔의 관리형 로그인 메뉴에서 스타일을 삭제합니다. 스타일에서 삭제할 스타일을 선택하고 스타일 삭제를 선택합니다.

개괄적으로 보면 도메인에 브랜딩을 할당하는 프로세스는 다음 단계로 구성됩니다.

1. 도메인을 생성하고 브랜딩 버전을 설정합니다.
2. 브랜딩 스타일을 생성하여 앱 클라이언트에 할당합니다.

앱 클라이언트에 스타일을 할당하려면

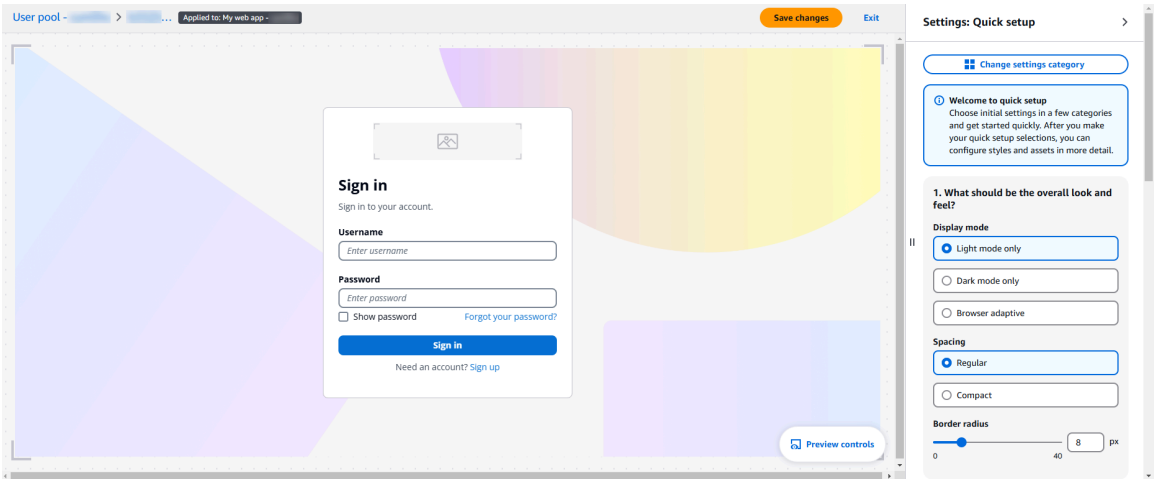
1. 사용자 풀의 도메인 메뉴에서 도메인을 생성하고 브랜딩 버전을 관리형 로그인으로 설정합니다.
2. 관리형 로그인 메뉴로 이동합니다. 스타일에서 스타일 생성을 선택합니다.
3. 스타일을 할당할 앱 클라이언트를 선택하거나 새 앱 클라이언트를 생성합니다.
4. 브랜딩 설정 구성을 시작하려면 브랜딩 편집기 시작을 선택합니다.

주제

- 브랜딩 에디터와 관리형 로그인 사용자 지정

브랜딩 에디터와 관리형 로그인 사용자 지정

브랜딩 편집기는 관리형 로그인 웹 페이지를 위한 시각적 디자인 및 편집 도구입니다. Amazon Cognito 콘솔에 내장되어 있습니다. 브랜딩 편집기에서 로그인 페이지의 미리 보기로 시작하여 빠른 설정 옵션 또는 고급 옵션이 포함된 세부 보기로 진행할 수 있습니다. 스타일 파라미터를 수정 및 미리 보거나 사용자 지정 배경 이미지 및 로고를 추가할 수 있습니다. 라이트 모드와 다크 모드를 구성할 수 있습니다.



시작하려면 사용자 풀 또는 앱 클라이언트에 적용할 수 있는 스타일을 생성합니다.

브랜딩 편집기를 시작하려면

1. 도메인 탭에서 도메인을 생성하거나 기존 도메인을 업데이트합니다. 브랜딩 버전에서 관리형 로그인을 사용하도록 도메인을 설정합니다.
2. 기존 앱 클라이언트 스타일이 있는 경우 삭제합니다.
   a. 앱 클라이언트 메뉴에서 앱 클라이언트를 선택합니다.
   b. 관리형 로그인 스타일에서 앱 클라이언트에 할당된 스타일을 선택합니다.
   c. 스타일 삭제를 선택합니다. 선택 항목을 확인합니다.
3. 사용자 풀의 관리형 로그인 메뉴로 이동합니다. 아직 선택하지 않은 경우 프롬프트에 따라 관리형 로그인이 포함된 기능 플랜을 선택합니다. 변경하지 않고 브랜딩 편집기를 체크아웃하려면 이 기능 미리 보기를 선택할 수도 있습니다.
4. 스타일에서 스타일 생성을 선택합니다.
5. 스타일을 할당할 앱 클라이언트를 선택하고 생성을 선택합니다. 새 앱 클라이언트를 생성할 수도 있습니다.
6. Amazon Cognito 콘솔이 브랜딩 편집기를 시작합니다.
7. 편집을 시작할 탭을 선택하거나 편집기 시작을 선택하고 빠른 설정을 입력합니다. 다음 탭을 사용할 수 있습니다.

   미리 보기

   관리형 로그인 페이지에서 현재 선택 항목이 어떻게 보이는지 확인합니다.

기본

전체 테마를 설정하고, 외부 ID 제공업체에 대한 링크를 구성하고, 양식 필드를 스타일화합니다.

구성 요소

헤더, 바닥글 및 개별 UI 요소에 대한 스타일을 구성합니다.

8. 초기 설정을 선택하려면 빠른 설정을 입력합니다. 설정 범주 변경을 선택하고 빠른 설정을 선택합니다. 계속을 선택하면 구성할 수 있는 기본 옵션 세트와 함께 브랜딩 편집기가 시작됩니다.

텍스트 및 현지화

브랜딩 편집기에서는 텍스트를 수정하거나 현지화할 수 없습니다. 대신 사용자에게 배포하는 URL에 `lang` 쿼리 파라미터를 추가합니다. 이 파라미터를 사용하면 관리형 로그인 페이지가 사용 가능한 여러 언어 중 하나로 현지화됩니다. 자세한 내용은 관리형 로그인 현지화 섹션을 참조하세요.

빠른 설정

브랜딩 편집기 시작 버튼은 다양한 기본 사용자 지정 옵션 중에서 선택할 수 있는 관리형 로그인 구성의 시각적 편집기를 로드합니다. 선택하면 Amazon Cognito가 미리 보기 창에서 관리형 로그인 변경 사항을 렌더링합니다. 세부 설정 메뉴로 돌아가려면 설정 범주 변경 버튼을 선택합니다.

전반적인 디자인과 느낌은 어떤가요?

관리형 로그인에 대한 기본 테마 설정을 구성합니다.

표시 모드

관리형 로그인을 위한 라이트 모드, 다크 모드 또는 적응형 환경을 선택합니다. Amazon Cognito가 관리형 로그인을 렌더링할 때 적응형 설정은 사용자의 브라우저 기본 설정에 따라 달라집니다. 브라우저 적응 모드를 선택하면 밝은 모드와 어두운 모드에 대해 다양한 색상과 로고 이미지를 선택할 수 있습니다.

Spacing

페이지의 요소 간에 기본 간격을 설정합니다.

테두리 반경

요소의 바깥쪽 테두리의 반올림 깊이를 설정합니다.

페이지 배경은 어떻게 보여야 하나요?

배경 유형

이미지 표시 확인란은 배경 이미지를 설정할지 아니면 단색 배경 색상을 설정할지를 나타냅니다.

1. 이미지를 사용하려면 이미지 표시를 선택하고 라이트 모드와 다크 모드의 배경 이미지를 선택합니다. 이미지에서 다루지 않는 배경 영역에 대해 다크 모드 및 라이트 모드 페이지 배경 색을 설정할 수도 있습니다.
2. 배경에 색상만 사용하려면 이미지 표시를 선택 취소하고 라이트 모드 및 다크 모드 페이지 배경 색상을 선택합니다.

양식은 어떻게 보여야 하나요?

관리형 로그인의 양식 요소에 대한 설정을 구성합니다. 양식 항목의 예로는 로그인 및 코드 프롬프트가 있습니다.

가로 맞춤

양식 필드의 수평 정렬을 설정합니다.

양식 로고

로고 이미지의 위치를 설정합니다.

로고 이미지

라이트 모드와 다크 모드의 양식 요소에 포함할 로고 이미지 파일을 선택합니다. 이미지를 업로드하려면 로고 이미지 드롭다운을 선택하고 새 자산 추가를 선택한 다음 로고 파일을 추가합니다.

기본 브랜딩 색상

라이트 모드와 다크 모드의 테마 색상을 설정합니다. 이 색상은 기본으로 분류된 모든 요소에 배경 색상으로 적용됩니다.

헤더는 어떻게 표시되어야 하나요?

관리형 로그인 페이지에 헤더를 포함할지 여부를 선택합니다. 헤더에는 로고 이미지가 포함될 수 있습니다.

헤더 로고

헤더에서 로고 이미지의 위치를 설정합니다.

로고 이미지

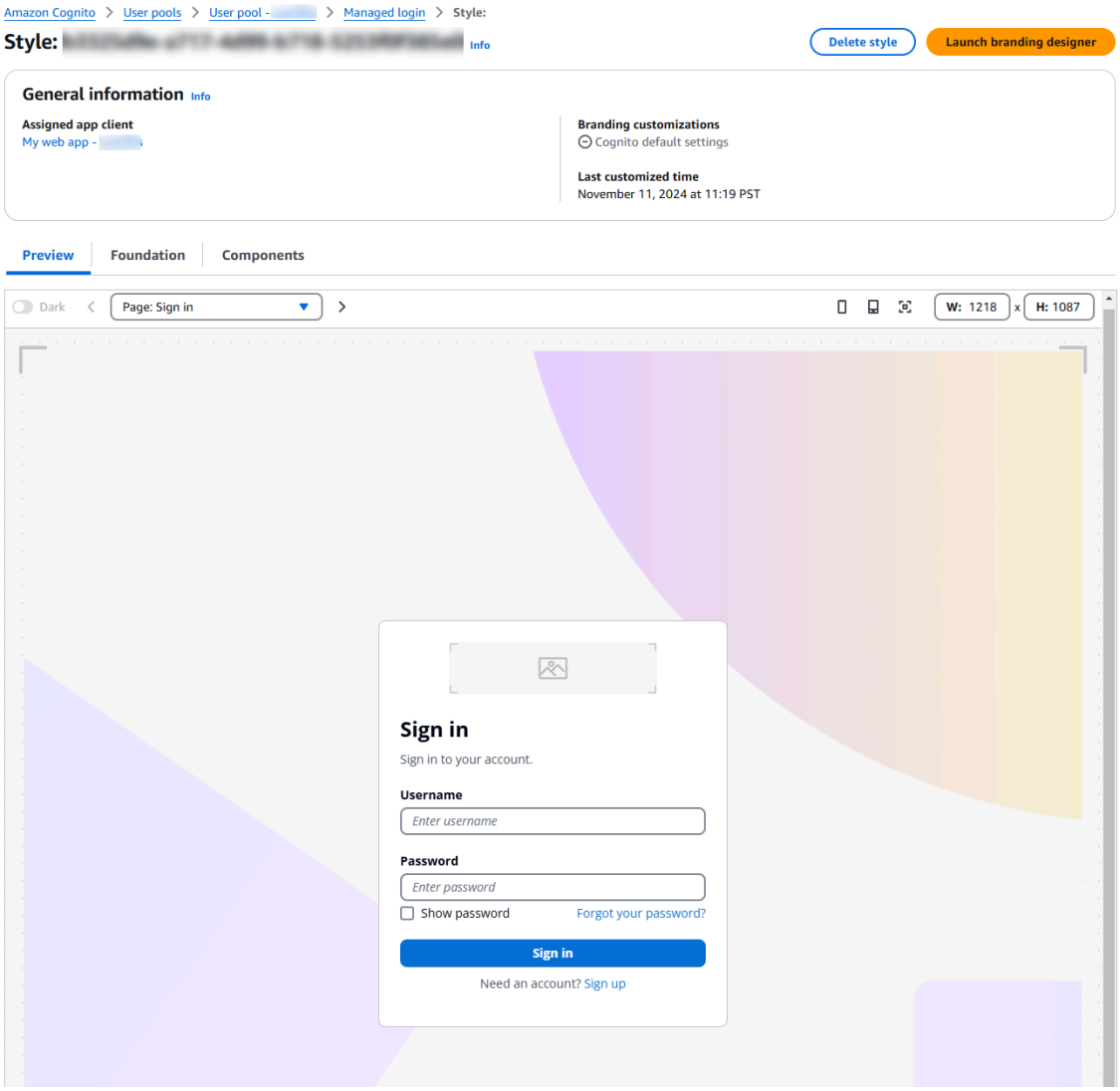
헤더에 포함할 로고 위치와 로고 이미지 파일을 선택합니다. 이미지를 업로드하려면 로고 이미지 드롭다운을 선택하고 새 자산 추가를 선택한 다음 로고 파일을 추가합니다.

헤더 배경색

헤더 배경의 라이트 모드 색상과 다크 모드 색상을 설정합니다.

세부 설정

세부 설정 보기에서 파운데이션 및 구성 요소의 개별 성 요소를 수정할 수 있습니다. 미리 보기 탭에는 사용자 지정과 함께 현재 컨텍스트의 관리형 로그인 미리 보기가 표시됩니다.



구성 요소의 시각적 편집기를 입력하려면 구성 요소의 타일에서 편집 아이콘을 선택합니다. 테마 스튜디오 편집기에서 설정 범주 변경 버튼을 사용하여 구성 요소 간에 전환할 수 있습니다.

기본

앱 스타일

관리형 로그인 구성의 기본 사항을 구성합니다. 이 범주에는 전체 테마, 텍스트 간격, 페이지 헤더 및 바닥글에 대한 설정이 있습니다.

표시 모드

관리형 로그인 페이지를 위한 라이트 모드, 다크 모드 또는 적응형 환경을 선택합니다. 브라우저 적응 모드를 선택하면 밝은 모드와 어두운 모드에 대해 다양한 색상과 로고 이미지를 선택할 수 있습니다.

Spacing

페이지의 요소 간에 기본 간격을 설정합니다.

인증 동작

사용자를 외부 ID 제공업체(idP)에 연결하는 버튼의 스타일을 구성합니다. 이 섹션에는 관리형 로그인 프롬프트 사용자에게 이메일 주소를 제공하고 이를 SAML 자격 증명 공급자 식별자와 일치시키는 도메인 검색 입력 옵션이 포함되어 있습니다.

양식 동작

입력 양식의 스타일 구성: 입력 위치, 색상 및 요소 정렬.

구성 요소

버튼

Amazon Cognito가 관리형 로그인 페이지에서 렌더링하는 버튼의 스타일입니다.

분할 도구

입력 양식과 외부 공급자 로그인 선택기와 같은 관리형 로그인 요소 간의 분할선 및 경계에 대한 스타일입니다.

Dropdown

드롭다운 메뉴의 스타일입니다.

파비콘

Amazon Cognito가 탭 및 북마크 아이콘에 제공하는 이미지 스타일입니다.

포커스 링

현재 선택된 입력을 나타내는 하이라이트의 스타일입니다.

양식 컨테이너

양식을 바인딩하는 요소의 스타일입니다.

글로벌 바닥글

Amazon Cognito가 관리형 로그인 페이지 하단에 표시하는 바닥글의 스타일입니다.

글로벌 헤더

Amazon Cognito가 관리형 로그인 페이지 상단에 표시하는 헤더의 스타일입니다.

표시

오류 및 성공 메시지의 스타일입니다.

옵션 제어

확인란, 다중 선택 및 기타 입력 프롬프트의 스타일입니다.

페이지 배경

관리형 로그인의 전체 배경에 대한 스타일입니다.

입력

양식 필드 입력 프롬프트의 스타일입니다.

링크

관리형 로그인 페이지의 하이퍼링크 스타일입니다.

페이지 텍스트

페이지 내 텍스트의 스타일입니다.

필드 텍스트

양식 입력 주변의 텍스트 스타일입니다.

관리형 로그인 브랜딩을 위한 API 및 SDK 작업

API 작업인 CreateManagedLoginBranding 및 UpdateManagedLoginBranding을 사용하여 관리형 로그인 스타일에 브랜딩을 적용할 수도 있습니다. 이러한 작업은 다른 앱 클라이언트 또는 사용자 풀에 대해 동일하거나 약간 수정된 브랜딩 스타일 버전을 생성하는 데 적합합니다. API 작업 DescribeManagedLoginBranding을 사용하여 기존 스타일의 관리형 로그인 브랜딩을 쿼리한 다음 필요에 따라 출력을 수정하고 다른 리소스에 적용합니다.

`UpdateManagedLoginBranding` 작업은 스타일이 적용되는 앱 클라이언트를 변경하지 않습니다. 앱 클라이언트에 할당된 기존 스타일만 업데이트합니다. 앱 클라이언트의 스타일을 완전히 바꾸려면 DeleteManagedLoginBranding으로 기존 스타일을 삭제하고 `CreateManagedLoginBranding`으로 새 스타일을 할당합니다. Amazon Cognito 콘솔에서도 마찬가지입니다. 기존 스타일을 삭제하고 새 스타일을 생성해야 합니다.

API 또는 SDK 요청에서 관리형 로그인 브랜딩을 설정하려면 설정을 `Document` 데이터 유형으로 변환되는 JSON 파일에 포함해야 합니다. 다음은 추가할 수 있는 이미지와 브랜딩 스타일을 구성하기 위한 프로그래밍 방식 요청을 생성하는 방법에 대한 지침입니다.

이미지 자산

CreateManagedLoginBranding 및 UpdateManagedLoginBranding에는 `Assets` 파라미터가 포함됩니다. 이 파라미터는 base64 인코딩 바이너리 형식의 이미지 파일 배열입니다.

> **Note**
>
> 브랜딩 스타일을 생성하거나 업데이트하는 프로그래밍 방식의 요청은 요청 크기가 2MB 이하여야 합니다. 요청의 자산으로 인해 이 제한을 초과할 수 있습니다. 이 경우 최대 요청 크기를 초과하지 않는 파라미터 그룹에 대해 요청을 여러 개의 `UpdateManagedLoginBranding` 요청으로 나눕니다. 이러한 요청으로 인해 지정되지 않은 파라미터가 기본값으로 설정되지 않으므로 기존 설정에 영향을 주지 않고 부분 요청을 보낼 수 있습니다.

일부 자산에는 제출할 수 있는 파일 유형에 대한 제한이 있습니다.

자산	허용되는 파일 확장명
FAVICON_ICO	ico
FAVICON_SVG	svg

자산	허용되는 파일 확장명
EMAIL_GRAPHIC	png, svg, jpeg
SMS_GRAPHIC	png, svg, jpeg
AUTH_APP_GRAPHIC	png, svg, jpeg
PASSWORD_GRAPHIC	png, svg, jpeg
PASSKEY_GRAPHIC	png, svg, jpeg
PAGE_HEADER_LOGO	png, svg, jpeg
PAGE_HEADER_BACKGROUND	png, svg, jpeg
PAGE_FOOTER_LOGO	png, svg, jpeg
PAGE_FOOTER_BACKGROUND	png, svg, jpeg
PAGE_BACKGROUND	png, svg, jpeg
FORM_BACKGROUND	png, svg, jpeg
FORM_LOGO	png, svg, jpeg
IDP_BUTTON_ICON	ico, svg

SVG 유형의 파일은 다음 속성과 요소를 지원합니다.

Attributes

```
accent-height, accumulate, additivive, alignment-baseline, ascent, attributename,
 attributetype, azimuth, basefrequency, baseline-shift, begin, bias, by, class,
 clip, clip-path, clip-rule, color, color-interpolation, color-interpolation-
filters, color-profile, color-rendering, cx, cy, d, dx, dy, diffuseconstant,
 direction, display, divisor, dur, edgemode, elevation, end, fill, fill-opacity,
 fill-rule, filter, filterunits, flood-color, flood-opacity, font-family, font-
size, font-size-adjust, font-stretch, font-style, font-variant, font-weight, fx,
 fy, g1, g2, glyph-name, glyphref, gradientunits, gradienttransform, height, href,
 id, image-rendering, in, in2, k, k1, k2, k3, k4, kerning, keypoints, keysplines,
 keytimes, lang, lengthadjust, letter-spacing, kernelmatrix, kernelunitlength,
```

```
 lighting-color, local, marker-end, marker-mid, marker-start, markerheight,
 markerunits, markerwidth, maskcontentunits, maskunits, max, mask, media,
 method, mode, min, name, numoctaves, offset, operator, opacity, order, orient,
 orientation, origin, overflow, paint-order, path, pathlength, patterncontentunits,
 patterntransform, patternunits, points, preservealpha, preserveaspectratio, r,
 rx, ry, radius, refx, refy, repeatcount, repeatdur, restart, result, rotate,
 scale, seed, shape-rendering, specularconstant, specularexponent, spreadmethod,
 stddeviation, stitchtiles, stop-color, stop-opacity, stroke-dasharray, stroke-
dashoffset, stroke-linecap, stroke-linejoin, stroke-miterlimit, stroke-opacity,
 stroke, stroke-width, style, surfacescale, tabindex, targetx, targety, transform,
 text-anchor, text-decoration, text-rendering, textlength, type, u1, u2, unicode,
 values, viewbox, visibility, vert-adv-y, vert-origin-x, vert-origin-y, width, word-
spacing, wrap, writing-mode, xchannelselector, ychannelselector, x, x1, x2, xmlns,
 y, y1, y2, z, zoomandpan
```

Elements

```
svg, a, altglyph, altglyphdef, altglyphitem, animatecolor, animatemotion,
 animatetransform, audio, canvas, circle, clippath, defs, desc, ellipse, filter,
 font, g, glyph, glyphref, hkern, image, line, lineargradient, marker, mask,
 metadata, mpath, path, pattern, polygon, polyline, radialgradient, rect, stop,
 style, switch, symbol, text, textpath, title, tref, tspan, video, view, vkern,
 feBlend, feColorMatrix, feComponentTransfer, feComposite, feConvolveMatrix,
 feDiffuseLighting, feDisplacementMap, feDistantLight, feFlood, feFuncA, feFuncB,
 feFuncG, feFuncR, feGaussianBlur, feMerge, feMergeNode, feMorphology, feOffset,
 fePointLight, feSpecularLighting, feSpotLight, feTile, feTurbulence
```

관리형 로그인 브랜딩 작업을 위한 도구

Amazon Cognito는 관리형 로그인 브랜딩 설정 객체에 대해 JSON-Schema 형식으로 파일을 관리합니다. 다음은 브랜딩 스타일을 프로그래밍 방식으로 업데이트하는 방법입니다.

사용자 풀 API에서 브랜딩을 업데이트하려면

1. Amazon Cognito 콘솔에서 사용자 풀의 관리형 로그인 메뉴에서 기본 관리형 로그인 브랜딩 스타일을 생성합니다. 앱 클라이언트에 할당합니다.
2. 예를 들어 `1example23456789`와 같이 스타일을 만든 앱 클라이언트의 ID를 기록합니다.
3. `ReturnMergedResources`가 `true`로 설정된 DescribeManagedLoginBrandingByClient API 요청을 사용하여 브랜딩 스타일의 설정을 검색합니다. 다음은 요청 본문의 예입니다.

```
{
```

```
    "ClientId": "1example23456789",
    "ReturnMergedResources": true,
    "UserPoolId": "us-east-1_EXAMPLE"
}
```

4. 사용자 지정을 사용하여 `DescribeManagedLoginBrandingByClient`의 출력을 수정합니다.

   a. 응답 본문은 생성 및 업데이트 작업을 위한 구문의 일부가 아닌 `ManagedLoginBranding` 요소로 래핑됩니다. JSON 객체의 이 최상위 수준을 제거합니다.

   b. 이미지를 바꾸려면 `Bytes` 값을 각 이미지 파일의 Base64-encoded 바이너리 데이터로 바꿉니다.

   c. 설정을 업데이트하려면 `Settings` 객체의 출력을 수정하고 다음 요청에 포함시킵니다. Amazon Cognito는 API 응답에서 수신하는 스키마에 없는 `Settings` 객체의 값을 무시합니다.

5. CreateManagedLoginBranding 또는 UpdateManagedLoginBranding 요청에서 업데이트된 응답 본문을 사용합니다. 이 요청의 크기가 2MB를 초과하는 경우 여러 요청으로 구분합니다. 이러한 작업은 달리 지정하지 않는 한 원래 설정이 변경되지 않은 상태로 유지되는 `PATCH` 모델에서 작동합니다.

클래식 호스팅 UI에서 관리형 로그인으로 마이그레이션

사용자 풀이 현재 클래식 호스팅 UI를 사용하는 경우 관리형 로그인으로 마이그레이션할 수 있습니다. 관리형 로그인과 클래식 호스팅 UI는 두 가지 브랜딩 버전입니다. 사용자 풀 도메인에서 브랜딩 버전을 설정하면 해당 도메인을 사용하는 모든 앱 클라이언트에 선택 사항이 적용되므로 마이그레이션은 도메인의 모든 앱 클라이언트를 동시에 전환합니다. 관리형 로그인은 시각적 브랜딩 편집기, 패스키 로그인 및 추가 인증 기능을 추가합니다. 클래식 호스팅 UI는 파일 기반 브랜딩을 사용하며 모든 기능 계획에서 사용할 수 있습니다. feature-by-feature 비교는 섹션을 참조하세요the section called “관리형 로그인”.

> **Important**
>
> 관리형 로그인 및 브랜딩 편집기에는 Essentials 또는 Plus 기능 플랜이 필요합니다. 클래식 호스팅 UI는 Lite를 포함한 모든 플랜에서 사용할 수 있습니다. 마이그레이션하기 전에 기능 계획을 확인합니다. the section called “관리형 로그인” 섹션을 참조하세요.

상위 수준에서 기존 사용자 풀을 클래식 호스팅 UI에서 관리형 로그인으로 이동하려면 다음 단계가 필요합니다. 전환하기 전에 현재 구성과 비교하여 각 단계를 검토합니다.

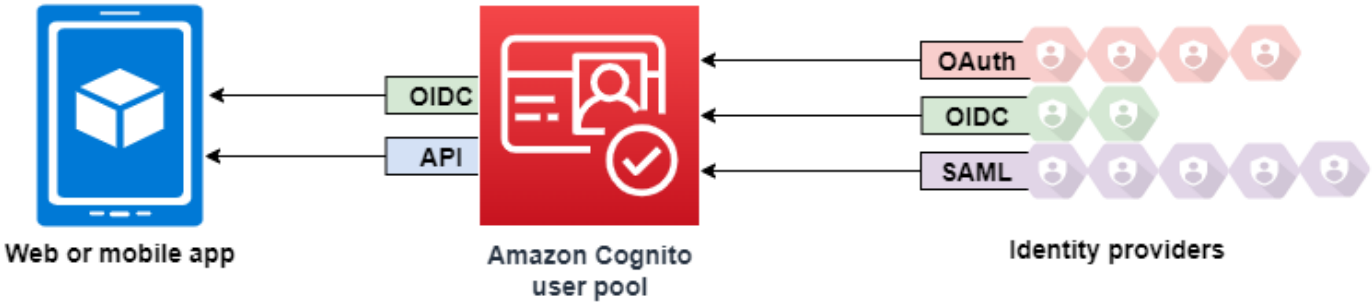
1. 사용자 풀이 관리형 로그인을 지원하는 기능 계획에 있는지 확인합니다. the section called “관리형 로그인” 섹션을 참조하세요.
2. 사용자 풀 도메인의 브랜딩 버전을 관리형 로그인으로 설정합니다. 브랜딩 버전은 도메인 설정이므로 해당 도메인을 사용하는 모든 앱 클라이언트에 적용됩니다.
3. 시각적 브랜딩 편집기에서 브랜딩을 설정합니다. 관리형 로그인은 클래식 호스팅 UI의 파일 기반 CSS 및 로고 사용자 지정과 다른 브랜딩 모델을 사용하므로 전환의 일부로 브랜딩 편집기와 비교하여 기존 브랜딩을 검토합니다. the section called “브랜딩 및 사용자 지정” 섹션을 참조하세요.
4. 프로덕션 트래픽을 전달하기 전에 관리형 로그인 페이지에서 로그인, 가입, 다중 인증(MFA) 및 암호 재설정 흐름을 테스트합니다.

> **Note**
>
> 이 페이지에서는 개념적 수준에서 마이그레이션을 요약합니다. 자세한 설정 및 브랜딩 절차는 각 단계의 신뢰할 수 있는 소스입니다.

Amazon Cognito 사용자 풀을 사용하여 사용자 인증

Amazon Cognito 사용자 풀은 웹 및 모바일 앱 인증과 권한 부여를 위한 사용자 디렉터리입니다. 앱의 관점에서 보면 Amazon Cognito 사용자 풀은 OpenID Connect(OIDC) ID 제공업체(idP)입니다. 사용자 풀은 보안, 아이덴티티 페더레이션, 앱 통합, 사용자 경험 사용자 지정을 위한 기능 계층을 추가합니다.

이 장에서는 외부 자격 증명 공급자를 통한 로그인을 포함하여 가입, 로그인 및 사용자 풀이 지원하는 인증 흐름 등 사용자가 인증하는 방법을 다룹니다. 관련 작업은 디렉터리의 사용자 관리, AWS Lambda 트리거로 동작 사용자 지정, 위협 방지로 풀 보호와 같은 자체 장에서 다룹니다.



주제

- 특성
- 사용자 풀 기능 플랜
- Amazon Cognito 사용자 풀로 인증
- 로그인 성공 후 리소스 액세스

특성

Amazon Cognito 사용자 풀에는 다음과 같은 기능이 있습니다.

가입

Amazon Cognito 사용자 풀에는 사용자 풀에 사용자 프로필을 추가하는 사용자 기반 방법, 관리자 기반 방법, 프로그래밍 방법이 있습니다. Amazon Cognito 사용자 풀은 다음과 같은 가입 모델을 지원합니다. 앱에서 이러한 모델을 결합하여 사용할 수 있습니다.

> **Important**
>
> 사용자 풀에서 사용자 가입을 활성화하면 인터넷에 있는 누구나 계정에 가입하고 앱에 로그인할 수 있습니다. 퍼블릭 가입이 가능하도록 앱을 공개하려는 경우가 아니면 사용자 풀에서 자체 등록을 활성화하지 마세요. 이 설정을 변경하려면 사용자 풀 콘솔의 인증 아래에 있는 가입 메뉴에서 셀프 서비스 가입을 업데이트하거나 CreateUserPool 또는 UpdateUserPool API 요청에서 AllowAdminCreateUserOnly 값을 업데이트합니다.
> 사용자 풀에서 설정할 수 있는 보안 기능에 대한 자세한 내용은 위협 방지를 통한 사용자 풀 보호 섹션을 참조하세요.

1. 사용자는 앱에서 정보를 입력하고 사용자 풀에 고유한 사용자 프로필을 생성할 수 있습니다. API 가입 작업을 호출하여 사용자 풀에 사용자를 등록할 수 있습니다. 이러한 가입 작업을 누구나 열거나 클라이언트 보안 암호 또는 AWS 자격 증명으로 권한을 부여할 수 있습니다.
2. Amazon Cognito에 정보를 전달할 권한을 부여할 수 있는 서드 파티 IdP로 사용자를 리디렉션할 수 있습니다. Amazon Cognito는 OIDC ID 토큰, OAuth 2.0 `userInfo` 데이터, SAML 2.0 어설션을 사용자 풀의 사용자 프로필로 처리합니다. 속성 매핑 규칙에 따라 Amazon Cognito가 수신하기를 원하는 속성을 제어할 수 있습니다.
3. 다음과 같은 방법으로 자체 데이터 소스 및 스키마를 기반으로 퍼블릭 또는 페더레이션 가입을 건너뛰고 사용자를 직접 생성할 수 있습니다.
   - Amazon Cognito 콘솔에서 직접 또는 API를 사용하여 사용자를 추가합니다.
   - CSV 파일에서 사용자를 대량으로 가져옵니다.
   - 기존 디렉터리에서 각 새 사용자를 조회하고 해당 데이터에서 프로필을 채우는 just-in-time AWS Lambda 함수를 실행합니다.

사용자가 가입한 후에는 Amazon Cognito가 액세스 및 ID 토큰에 나열하는 그룹에 사용자를 추가할 수 있습니다. ID 토큰을 자격 증명 풀에 전달할 때 사용자 풀 그룹을 IAM 역할에 연결할 수도 있습니다.

관련 주제

- 사용자 풀의 사용자 관리
- API, OIDC 및 관리형 로그인 페이지 인증 이해
- AWS SDKs를 사용하는 Amazon Cognito 자격 증명 공급자의 코드 예제

로그인

Amazon Cognito는 독립 실행형 사용자 디렉터리이자 앱에 대한 ID 제공업체(idP)가 될 수 있습니다. 사용자는 Amazon Cognito에서 호스팅하는 관리형 로그인 페이지 또는 Amazon Cognito 사용자 풀 API를 통해 사용자 지정 사용자 인증 서비스로 로그인할 수 있습니다. 맞춤형 프런트엔드 뒤에 있는 애플리케이션 계층은 몇 가지 방법 중 하나로 백엔드에서 요청을 승인하여 합법적인 요청을 확인할 수 있습니다.

사용자는 사용자 이름과 암호, 패스키, 이메일 및 SMS 메시지 일회용 암호를 사용하여 설정하고 로그인할 수 있습니다. 외부 사용자 디렉터리를 사용한 통합 로그인, 로그인 후 다중 인증(MFA), 신뢰 기억 디바이스, 설계한 사용자 지정 인증 흐름을 제공할 수 있습니다.

필요할 경우 Amazon Cognito에 기본 제공된 사용자 디렉터리와 결합된 외부 디렉터리를 사용하여 사용자를 로그인하려면 다음 통합을 추가할 수 있습니다.

1. OAuth 2.0 소셜 로그인으로 로그인하고 고객 사용자 데이터를 가져옵니다. Amazon Cognito는 OAuth 2.0을 통해 Google, Facebook, Amazon, Apple 로그인을 지원합니다.
2. SAML 및 OIDC 로그인으로 로그인하고 직장 및 학교 사용자 데이터를 가져옵니다. SAML 또는 OpenID Connect(OIDC) ID 제공업체(idP)의 클레임을 수락하도록 Amazon Cognito를 구성할 수도 있습니다.
3. 외부 사용자 프로필을 기본 사용자 프로필에 연결합니다. 연결된 사용자는 서드 파티 사용자 자격 증명으로 로그인하고 기본 제공 디렉터리에서 사용자에게 할당한 액세스 권한을 받을 수 있습니다.

관련 주제

- 타사 ID 제공업체를 통한 사용자 풀 로그인
- 페더레이션 사용자를 기존 사용자 프로필에 연결

기계 간 인증

일부 세션은 사람과 기계의 상호 작용이 아닙니다. 자동화된 프로세스를 통해 API에 대한 요청을 승인할 수 있는 서비스 계정이 필요할 수 있습니다. OAuth 2.0 범위에서 machine-to-machine 권한 부여를 위한 액세스 토큰을 생성하려면 토큰 엔드포인트에서 클라이언트 자격 증명 부여를 생성하는 앱 클라이언트를 추가하거나 사용자 풀 도메인이 필요하지 않은 `GetClientToken` API 작업을 호출할 수 있습니다.

관련 주제

- 범위, M2M 및 리소스 서버

관리형 로그인

사용자 인터페이스를 빌드하지 않으려는 경우 사용자에게 사용자 지정 관리형 로그인 페이지를 제공할 수 있습니다. 관리형 로그인은 가입, 로그인, 다중 인증(MFA), 암호 재설정을 위한 일련의 웹 페이지입니다. 기존 도메인에 관리형 로그인을 추가하거나 AWS 하위 도메인에서 접두사 도메인을 사용할 수 있습니다.

관리형 로그인에는 더 간단한 설계와 더 적은 기능을 갖춘 1세대 버전인 클래식 호스팅 UI라는 대안이 있습니다. 사용자 풀 도메인에 대해 둘 중 하나를 선택하면 함께 사용되지 않습니다. 다음은 두 옵션을 비교하는 표입니다.

기능	클래식 호스팅 UI	관리형 로그인
브랜딩 버전	호스팅 UI(클래식)	관리형 로그인
사용자 지정	파일 기반 브랜딩(로고 이미지 및 CSS 값 파일)	Amazon Cognito 콘솔의 시각적 브랜딩 편집기
기능 플랜	Lite를 포함한 모든 플랜	Essentials 및 Plus
패스키 로그인	사용할 수 없음	사용 가능

> **Note**
>
> 클래식 호스팅 UI는 Lite를 포함한 모든 기능 플랜에서 사용할 수 있습니다. 관리형 로그인 및 브랜딩 편집기에는 Essentials 또는 Plus 기능 플랜이 필요합니다. 자세한 내용은 사용자 풀 기능 플랜 섹션을 참조하세요.

관리형 로그인을 설정하려면 다음 사전 조건을 순서대로 완료하세요.

1. 사용자 풀을 생성합니다. Amazon Cognito 콘솔에서 새 애플리케이션 생성 섹션을 참조하세요.
2. 사용자 풀 도메인(접두사 도메인 또는 사용자 지정 도메인)을 추가합니다. 사용자 풀 도메인 구성 섹션을 참조하세요.

3. 앱 클라이언트를 생성하고 브랜딩 버전 및 반환 URL을 설정합니다. 사용자 풀 관리형 로그인 섹션을 참조하세요.

관련 주제

- 사용자 풀 관리형 로그인
- 사용자 풀 도메인 구성

위협 방지

다중 인증(MFA), 적응형 인증 및 AWS WAF 웹 ACLs을 사용하여 암호 추측, 계정 탈취 및 악성 트래픽으로부터 사용자를 보호합니다. 이러한 보호는에서 다릅니다위협 방지를 통한 사용자 풀 보호.

사용자 지정 사용자 경험

대부분의 가입, 로그인 또는 프로필 업데이트 단계에서 Amazon Cognito가 Lambda 트리거 및 사용자 지정 인증 흐름을 사용하여 요청을 처리하는 방법을 사용자 지정하고 관리형 로그인에 익숙한 모습과 느낌을 제공할 수 있습니다. Lambda 트리거를 사용하여 사용자 풀 워크플로 사용자 정의 섹션을 참조하세요.

모니터링 및 분석

Amazon Cognito 사용자는 API 요청을에 로깅하고 AWS CloudTrail, 성능 지표를 Amazon CloudWatch Logs에 게시하고, 메시지 전송 및 요청 볼륨을 모니터링합니다. Amazon Cognito 로그인 AWS CloudTrail 섹션을 참조하세요.

Amazon Cognito 자격 증명 풀 통합

Amazon Cognito의 나머지 절반은 자격 증명 풀입니다. 자격 증명 풀은 사용자의 Amazon DynamoDB 또는 Amazon S3 AWS 서비스에 대한 API 요청을 승인하고 모니터링하는 자격 증명을 제공합니다. 사용자 풀에서 사용자를 분류하는 방식에 따라 데이터를 보호하는 자격 증명 기반 액세스 정책을 구축할 수 있습니다. 자격 증명 풀은 또한 사용자 풀 인증과 별개로 다양한 ID 제공업체(idP)의 토큰 및 SAML 2.0 어설션을 수락할 수 있습니다.

관련 주제

- 로그인 후 자격 증명 풀을 AWS 서비스 사용하여 액세스

- Amazon Cognito 자격 증명 풀

사용자 풀 기능 플랜

비용을 이해하는 것은 Amazon Cognito 사용자 풀 인증을 구현하기 위한 준비의 중요한 단계입니다. Amazon Cognito에는 사용자 풀에 대한 기능 플랜이 있습니다. 각 플랜에는 활성 사용자당 기능 세트와 월별 비용이 있습니다. 각 기능 플랜은 이전 기능보다 더 많은 기능에 대한 액세스를 잠금 해제합니다.

사용자 풀에는 켜거나 끌 수 있는 다양한 기능이 있습니다. 예를 들어, 다중 인증(MFA)를 켜고 타사 ID 제공업체(IdP)를 통한 로그인을 끌 수 있습니다. 일부 변경 사항에서는 기능 플랜을 전환해야 합니다. 사용자 풀의 다음 특성에 따라에서 매월 사용량에 대해 AWS 청구하는 비용이 결정됩니다.

- 선택한 기능
- 애플리케이션이 사용자 풀 API에 대해 수행하는 초당 요청 수
- 한 달 동안 인증, 업데이트 또는 쿼리 활동을 수행한 사용자 수(월간 활성 사용자 또는 MAU라고도 함)
- 타사 SAML 2.0 또는 OpenID Connect(OIDC) IdP의 월간 활성 사용자 수
- 머신 간 권한 부여를 위해 클라이언트 자격 증명을 부여하는 앱 클라이언트 및 사용자 풀의 수

사용자 풀 요금에 대한 최신 정보는 Amazon Cognito 요금을 참조하세요.

기능 플랜 선택은 하나의 사용자 풀에 적용됩니다. 동일한 AWS 계정 의 사용자 풀마다 계획 선택이 다를 수 있습니다. 사용자 풀 내의 앱 클라이언트에는 별도의 기능 플랜을 적용할 수 없습니다. 새 사용자 풀의 기본 계획 선택은 Essentials입니다.

애플리케이션의 요구 사항에 맞게 언제든지 기능 플랜 간에 전환할 수 있습니다. 플랜 간에 일부 변경 사항을 적용하려면 활성 기능을 꺼야 합니다. 자세한 내용은 기능 플랜을 변경하기 위한 기능 끄기 섹션을 참조하세요.

사용자 풀 기능 플랜

Lite

Lite는 월별 활성 사용자 수가 적은 사용자 풀을 위 저렴한 기능 플랜입니다. 이 계획은 기본 인증 기능이 있는 사용자 디렉터리에 충분합니다. 여기에는 로그인 기능과 관리형 로그인의 사용자 지정이 덜 간편한 이전 버전인 클래식 호스팅 UI가 포함됩니다. 액세스 토큰 사용자 지정 및 패스키 인증과 같은 많은 최신 기능은 Lite 플랜에 포함되지 않습니다.

Essentials

Essentials에는 모든 최신 사용자 풀 인증 기능이 있습니다. 이 계획은 로그인 페이지가 관리형 로그인이든 사용자 지정 로그인이든 관계없이 애플리케이션에 새 옵션을 추가합니다. Essentials에는 선택 기반 로그인 및 이메일 MFA와 같은 고급 인증 기능이 있습니다.

Plus

Plus에는 Essentials 플랜의 모든 내용이 포함되어 있으며 사용자를 보호하는 고급 보안 기능이 추가되었습니다. 사용자 로그인, 가입 및 암호 관리 요청을 모니터링하여 손상 지표를 확인합니다. 예를 들어 사용자 풀은 사용자가 예상치 못한 위치에서 로그인하는지 아니면 퍼블릭 위반의 일부인 암호를 사용하고 있는지 감지할 수 있습니다.

Plus 플랜이 있는 사용자 풀은 사용자 활동 세부 정보 및 위험 평가 로그를 생성합니다. 외부 서비스로 로그를 내보낼 때 이러한 로그에 자체 사용 및 보안 분석을 적용할 수 있습니다.

> **Note**
>
> 이전에는 일부 사용자 풀 기능이 고급 보안 기능 요금 구조에 포함되었습니다. 이 구조에 포함된 기능은 이제 Essentials 또는 Plus 플랜에 속합니다.

주제

- 기능 플랜 선택
- 플랜별 기능
- Essentials 플랜 기능
- Plus 플랜 기능
- 기능 플랜을 변경하기 위한 기능 끄기

기능 플랜 선택

AWS Management Console

기능 플랜을 선택하려면

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 자격 AWS 증명을 입력합니다.
2. [사용자 풀(User Pools)]을 선택합니다.

3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. 설정 메뉴를 선택하고 기능 플랜 탭을 검토합니다.
5. Lite, Essentials 및 Plus 플랜에서 사용할 수 있는 기능을 검토합니다.
6. 계획을 변경하려면 Essentials로 전환 또는 Plus로 전환을 선택합니다. Lite 플랜으로 전환하려면 기타 플랜을 선택한 다음 Lite와 비교를 선택합니다.
7. 다음 화면에서 선택을 검토하고 확인을 선택합니다.

CLI/API/SDK

CreateUserPool 및 UpdateUserPool 작업은 `UserPoolTier` 파라미터에서 기능 플랜을 설정합니다. `UserPoolTier`에 값을 지정하지 않으면 사용자 풀의 기본값은 `Essentials`입니다. `AdvancedSecurityMode`를 `AUDIT` 또는 `ENFORCED`로 설정한 경우 사용자 풀 티어는 `PLUS`여야 하며 지정하지 않을 경우 기본값은 `PLUS`여야 합니다.

구문은 CreateUserPool의 예제를 참조하세요. 다양한 프로그래밍 언어에 대한 SDK의이 함수에 대한 링크는 CreateUserPool의 도 참조하세요. AWS SDKs

```
"UserPoolTier": "PLUS"
```

에서 AWS CLI이 옵션은 `--user-pool-tier` 인수입니다.

```
--user-pool-tier PLUS
```

자세한 내용은 AWS CLI 명령 참조의 create-user-pool 및 update-user-pool을 참조하세요.

플랜별 기능

사용자 풀의 기능 및 플랜

기능	설명	기능 플랜
안전하지 않은 암호로부터 보호	런타임 시 손상 지표에 대한 일반 텍스트 암호 확인	Plus
악의적인 로그인 시도로부터 보호	런타임 시 손상 지표에 대한 세션 속성 확인	Plus

기능	설명	기능 플랜
사용자 활동 로깅 및 분석	사용자 인증 세션 속성 및 위험 점수에 대한 로그 생성	Plus
사용자 활동 로그 내보내기	사용자 세션 및 위험 로그를 외부로 푸시 AWS 서비스	Plus
시각적 편집기를 사용하여 관리형 로그인 페이지 사용자 지정	Amazon Cognito 콘솔에서 시각적 편집기를 사용하여 관리형 로그인 페이지에 브랜딩 및 스타일 적용	Essentials + Plus
이메일 일회용 코드를 통한 MFA	사용자 이름 인증 후 로컬 사용자에게 추가 이메일 메시지 로그인 요소를 제공하도록 요청하거나 요구	Essentials + Plus
런타임 시 액세스 토큰 범위 및 클레임 사용자 지정	Lambda 트리거를 사용하여 사용자 풀 액세스 토큰의 권한 부여 기능 확장	Essentials + Plus
일회용 코드를 통한 암호 없는 로그인	사용자가 첫 번째 인증 요소로 이메일 또는 SMS로 일회용 암호를 수신하도록 허용	Essentials + Plus
하드웨어 또는 소프트웨어 FIDO2 인증자를 사용한 패스키 로그인	사용자가 FIDO2 인증자에 저장된 암호화 키를 첫 번째 인증 요소로 사용하도록 허용	Essentials + Plus
가입 및 로그인	인증 작업을 수행하고 새 사용자가 애플리케이션의 계정에 등록하도록 합니다.	Lite + Essentials + Plus
사용자 그룹	사용자의 논리적 그룹화를 생성하고 ID 풀 작업에 대한 기본 IAM 역할을 할당합니다.	Lite + Essentials + Plus

기능	설명	기능 플랜
소셜, SAML 및 OIDC 공급자로 로그인	사용자에게 직접 로그인하거나 선호하는 공급자를 통해 로그인할 수 있는 옵션을 제공합니다.	Lite + Essentials + Plus
OAuth 2.0/OIDC 권한 부여 서버	OIDC 발급자 역할을 합니다.	Lite + Essentials + Plus
로그인 페이지	인증을 위한 호스팅 웹 페이지 모음입니다. 관리형 로그인은 Essentials 및 Plus 계층에서 사용할 수 있습니다. 클래식 호스팅 UI는 모든 기능 계층에서 사용할 수 있습니다.	Lite + Essentials + Plus
암호, 사용자 지정, 새로 고침 토큰 및 SRP 인증	사용자에게 애플리케이션의 사용자 이름과 암호를 묻는 메시지를 표시합니다.	Lite + Essentials + Plus
클라이언트 자격 증명을 사용하는 M2M(머신 간)	사람이 아닌 엔터티의 권한 부여를 위한 액세스 토큰을 발급합니다.	Lite + Essentials + Plus
리소스 서버를 사용한 API 권한 부여	외부 시스템에 대한 액세스를 승인하는 사용자 지정 범위를 사용하여 액세스 토큰을 발급합니다.	Lite + Essentials + Plus
사용자 가져오기	CSV 파일에서 가져오기 작업을 설정하고 사용자가 로그인하는 즉시 마이그레이션을 수행합니다.	Lite + Essentials + Plus

기능	설명	기능 플랜
인증 앱 및 SMS 일회용 코드가 포함된 MFA	사용자 이름 인증 후 로컬 사용자에게 추가 SMS 메시지 또는 인증 앱 로그인 요소를 제공하도록 요청하거나 요구합니다.	Lite + Essentials + Plus
런타임 시 ID 토큰 범위 및 클레임 사용자 지정	Lambda 트리거를 사용하여 사용자 풀 ID 토큰의 인증 기능 확장	Lite + Essentials + Plus
Lambda 트리거를 사용한 사용자 지정 런타임 작업	외부 작업을 수행하고 인증에 영향을 미치는 Lambda 함수를 사용하여 런타임 시 로그인 프로세스 사용자 지정	Lite + Essentials + Plus
CSS를 사용하여 관리형 로그인 페이지 사용자 지정	관리형 로그인 페이지에서 CSS 템플릿을 다운로드하고 일부 스타일을 변경합니다.	Lite + Essentials + Plus

Essentials 플랜 기능

Essentials 기능 플랜에는 Amazon Cognito 사용자 풀의 가장 좋은 최신 기능이 대부분 포함되어 있습니다. Lite에서 Essentials 플랜으로 전환하면 관리형 로그인 페이지, 이메일 메시지 일회용 암호를 사용한 다중 인증, 향상된 암호 정책 및 사용자 지정 액세스 토큰에 대한 새로운 기능을 얻을 수 있습니다. 새 사용자 풀 기능을 최신 상태로 유지하려면 사용자 풀의 Essentials 플랜을 선택합니다.

다음 섹션에서는 Essentials 플랜을 사용하여 애플리케이션에 추가할 수 있는 기능에 대한 간략한 개요를 제공합니다. 자세한 내용은 다음 페이지를 참조하세요.

추가 리소스

- 액세스 토큰 사용자 지정: 사전 토큰 생성 Lambda 트리거
- 이메일 MFA: SMS 및 이메일 메시지 MFA
- 암호 기록: 암호, 계정 복구 및 암호 정책
- 향상된 UI: 관리형 로그인 페이지에 브랜딩 적용

주제

- 액세스 토큰 사용자 지정
- 이메일 MFA
- 암호 재사용 방지
- 관리형 로그인 호스팅 로그인 및 권한 부여 서버
- 선택 기반 인증

액세스 토큰 사용자 지정

사용자 풀 액세스 토큰은 애플리케이션에 API 액세스, userInfo 엔드포인트에서 사용자 속성 검색, 외부 시스템에 대한 그룹 멤버십 설정 등의 권한을 부여합니다. 고급 시나리오에서는 사용자 풀 디렉터리의 기본 액세스 토큰 데이터에 애플리케이션이 런타임에 결정하는 추가 임시 파라미터를 추가할 수 있습니다. 예를 들어 Amazon Verified Permissions를 사용하여 사용자의 API 권한을 확인하고 그에 따라 액세스 토큰의 범위를 조정할 수 있습니다.

Essentials 플랜은 기존 함수에 사전 토큰 생성 트리거를 추가합니다. 하위 계층 플랜을 사용하면 추가 클레임, 역할 및 그룹 멤버십으로 ID 토큰을 사용자 지정할 수 있습니다. Essentials는 액세스 토큰 클레임, 역할, 그룹 멤버십 및 범위를 사용자 지정하는 트리거 입력 이벤트의 새 버전을 추가합니다. 이벤트 버전 3을 사용하여 M2M(머신 간) 클라이언트 자격 증명을 부여하면 액세스 토큰을 사용자 지정할 수 있습니다.

액세스 토큰을 사용자 지정하는 방법

1. Essentials 또는 Plus 기능 플랜을 선택합니다.
2. 사용자의 트리거에 대해 Lambda 함수를 생성합니다. 예제 함수를 사용하려면 Node.js에 맞게 구성합니다.
3. Lambda 함수를 예제 코드로 채우거나 직접 구성하세요. 함수는 Amazon Cognito에서 요청 개체를 처리하고 포함하려는 변경 사항을 반환해야 합니다.
4. 새 함수를 버전 2 또는 3 사전 토큰 생성 트리거로 할당합니다. 버전 2 이벤트는 사용자 ID에 대한 액세스 토큰을 사용자 지정합니다. 버전 3은 사용자 및 시스템 ID에 대한 액세스 토큰을 사용자 지정합니다.

자세히 알아보기

- 액세스 토큰 사용자 지정
- Amazon Cognito 사용자 풀에서 액세스 토큰을 사용자 지정하는 방법

이메일 MFA

Amazon Cognito 사용자 풀은 이메일을 다중 인증(MFA)의 두 번째 요소로 사용하도록 구성할 수 있습니다. 이메일 MFA를 사용하면 Amazon Cognito는 인증 프로세스를 완료하기 위해 입력해야 하는 확인 코드가 포함된 이메일을 사용자에게 보낼 수 있습니다. 이렇게 하면 사용자 로그인 흐름에 중요한 추가 보안 계층이 추가됩니다. 이메일 기반 MFA를 활성화하려면 기본 이메일 구성 대신 Amazon SES 이메일 전송 구성을 사용하도록 사용자 풀을 구성해야 합니다.

사용자가 이메일 메시지로 MFA를 선택하면 Amazon Cognito는 로그인을 시도할 때마다 사용자의 등록된 이메일 주소로 일회용 확인 코드를 보냅니다. 그런 다음 사용자는 이 코드를 사용자 풀에 다시 제공하여 인증 흐름을 완료하고 액세스 권한을 얻어야 합니다. 이렇게 하면 사용자의 사용자 이름과 암호가 손상되더라도 애플리케이션 리소스에 액세스하기 전에 이메일 코드라는 추가 요소를 제공해야 합니다.

자세한 내용은 SMS 및 이메일 메시지 MFA 섹션을 참조하세요. 다음은 이메일 MFA에 대한 사용자 풀 및 사용자를 설정하는 방법에 대한 개요입니다.

Amazon Cognito 콘솔에서 이메일 MFA를 설정하려면

1. Essentials 또는 Plus 기능 플랜을 선택합니다.
2. 사용자 풀의 로그인 메뉴에서 다중 인증을 편집합니다.
3. 설정하려는 MFA 적용 수준을 선택합니다. MFA 요구를 사용하면 API의 사용자는 MFA를 설정, 확인 및 로그인하는 챌린지를 자동으로 수신합니다. MFA가 필요한 사용자 풀에서 관리형 로그인은 MFA 요소를 선택하고 설정하라는 메시지를 표시합니다. 선택적 MFA를 사용하면 애플리케이션에서 사용자에게 MFA를 설정하고 이메일 MFA에 대한 사용자의 기본 설정을 설정할 수 있는 옵션을 제공해야 합니다.
4. MFA 메서드에서 이메일 메시지를 옵션 중 하나로 선택합니다.

자세히 알아보기

- SMS 및 이메일 메시지 MFA

암호 재사용 방지

기본적으로 Amazon Cognito 사용자 풀 암호 정책은 암호 길이와 문자 유형 요구 사항 및 임시 암호 만료를 설정합니다. Essentials 플랜에는 암호 기록을 적용하는 기능이 추가되었습니다. 사용자가 암호를 재설정하려고 하면 사용자 풀에서 이전 암호로 설정하지 못할 수 있습니다. 암호 정책 구성에 대한 자

세한 내용은 사용자 풀 암호 요구 사항 추가 섹션을 참조하세요. 다음은 암호 기록 정책을 사용하여 사용자 풀을 설정하는 방법에 대한 개요입니다.

Amazon Cognito 콘솔에서 암호 기록을 설정하려면

1. Essentials 또는 Plus 기능 플랜을 선택합니다.
2. 사용자 풀의 인증 방법 메뉴에서 암호 정책을 찾아 편집을 선택합니다.
3. 사용 가능한 다른 옵션을 구성하고 이전 암호 사용 방지를 위한 값을 설정합니다.

자세히 알아보기

- 암호, 계정 복구 및 암호 정책

관리형 로그인 호스팅 로그인 및 권한 부여 서버

Amazon Cognito 사용자 풀에는 OpenID Connect(OIDC) IdP, 서비스 공급자 또는 타사 IdP, 가입 및 로그인을 위한 퍼블릭 사용자 대화형 페이지를 지원하는 선택적 웹 페이지가 있습니다. 이러한 페이지를 총칭하여 관리형 로그인이라고 합니다. 사용자 풀의 도메인을 선택하면 Amazon Cognito가 이러한 페이지를 자동으로 활성화합니다. Lite 플랜에 호스팅 UI가 있는 경우 Essentials 플랜은 이 고급 버전의 가입 및 로그인 페이지를 엽니다.

관리형 로그인 페이지에는 브랜딩 및 스타일을 사용자 지정하기 위한 더 많은 기능과 옵션이 포함된 깔끔한 최신 인터페이스가 있습니다. Essentials 플랜은 관리형 로그인에 대한 액세스를 잠금 해제하는 가장 낮은 플랜 수준입니다.

Amazon Cognito 콘솔에서 관리형 로그인을 설정하려면

1. 설정 메뉴에서 Essentials 또는 Plus 기능 플랜을 선택합니다.
2. 도메인 메뉴에서 사용자 풀에 도메인을 할당하고 관리형 로그인의 브랜딩 버전을 선택합니다.
3. 관리형 로그인 메뉴의 스타일 탭에서 스타일 생성을 선택하고 앱 클라이언트에 스타일을 할당하거나 새 앱 클라이언트를 생성합니다.

자세히 알아보기

- 사용자 풀 관리형 로그인

선택 기반 인증

Essentials 계층은 향상된 UI 및 SDK 기반 API 작업에서 인증 작업을 위한 새로운 인증 흐름을 도입합니다. 이 흐름은 선택 기반 인증입니다. 선택 기반 인증은 사용자의 인증이 로그인 방법의 애플리케이션 측 선언으로 시작하지 않고 가능한 로그인 방법의 쿼리와 선택으로 시작하는 방법입니다. 사용자 풀을 구성하여 선택 기반 인증을 지원하고 사용자 이름-암호 인증, 암호 없음, 패스키 인증을 잠금 해제할 수 있습니다. API에서는 USER_AUTH 흐름입니다.

Amazon Cognito 콘솔에서 선택 기반 인증을 설정하려면

1. Essentials 또는 Plus 기능 플랜을 선택합니다.
2. 사용자 풀의 로그인 메뉴에서 선택 기반 로그인 옵션을 편집합니다. 선택 기반 인증에서 활성화하려는 인증 방법을 선택하고 구성합니다.
3. 사용자 풀의 인증 방법 메뉴에서 로그인 작업의 구성을 편집합니다.

자세히 알아보기

- Amazon Cognito 사용자 풀로 인증

Plus 플랜 기능

Plus 기능 플랜에는 Amazon Cognito 사용자 풀에 대한 고급 보안 기능이 있습니다. 이러한 기능은 런타임 시 사용자 컨텍스트를 로깅하고 분석하여 디바이스, 위치, 요청 데이터 및 암호의 잠재적 보안 문제를 확인합니다. 그런 다음 사용자 계정에 보안 보호를 차단하거나 추가하는 자동 응답을 통해 잠재적 위험을 완화합니다. 추가 분석을 위해 보안 로그를 Amazon S3, Amazon Data Firehose 또는 Amazon CloudWatch Logs로 내보낼 수도 있습니다.

Essentials 플랜에서 Plus 플랜으로 전환하면 Essentials의 모든 기능과 다음 추가 기능을 얻을 수 있습니다. 여기에는 고급 보안 기능이라고도 하는 위협 방지 보안 옵션 세트가 포함됩니다. 인증 프런트 엔드의 위협에 자동으로 적응하도록 사용자 풀을 구성하려면 사용자 풀에 대한 Plus 플랜을 선택합니다.

다음 섹션에서는 Plus 플랜을 사용하여 애플리케이션에 추가할 수 있는 기능에 대한 간략한 개요를 제공합니다. 자세한 내용은 다음 페이지를 참조하세요.

추가 리소스

- 적응형 인증: 적응형 인증 작업
- 손상된 보안 인증: 손상된 자격 증명 감지 작업

- 로그 내보내기: Amazon Cognito 사용자 풀에서 로그 내보내기

주제

- 위협 방지: 적응형 인증
- 위협 방지: 손상된 자격 증명 탐지
- 위협 방지: 사용자 활동 로깅

위협 방지: 적응형 인증

Plus 플랜에는 적응형 인증 기능이 포함되어 있습니다. 이 기능을 활성화하면 사용자 풀이 모든 사용자 인증 세션에 대한 위험 평가를 수행합니다. 결과 위험 등급에서 사용자가 결정한 임계값을 초과하는 위험 수준으로 로그인하는 사용자를 위해 인증을 차단하거나 MFA를 푸시할 수 있습니다. 적응형 인증을 사용하면 사용자 풀과 애플리케이션이 공격을 받고 있다고 의심되는 계정에 대해 MFA를 자동으로 차단하거나 설정합니다. 사용자 풀의 위험 등급에 대한 피드백을 제공하여 향후 등급을 조정할 수도 있습니다.

Amazon Cognito 콘솔에서 적응형 인증을 설정하려면

1. Plus 기능 플랜을 선택합니다.
2. 사용자 풀의 위협 방지 메뉴의 위협 방지에서 표준 및 사용자 지정 인증을 편집합니다.
3. 표준 또는 사용자 지정 인증에 대한 적용 모드를 전체 함수로 설정합니다.
4. 적응형 인증에서 다양한 위험 수준에 대한 자동 위험 응답을 구성합니다.

자세히 알아보기

- 적응형 인증 작업
- 애플리케이션에서 위협 방지를 위한 데이터 수집

위협 방지: 손상된 자격 증명 탐지

Plus 플랜에는 손상된 자격 증명 감지 기능이 포함되어 있습니다. 이 기능은 안전하지 않은 암호 사용과 이로 인해 발생하는 의도치 않은 애플리케이션 액세스 위협을 방지합니다. 사용자가 사용자 이름과 암호로 로그인하도록 허용하면 다른 곳에서 사용한 암호를 재사용할 수 있습니다. 해당 암호가 유출되었을 수도 있고, 흔히 추측되는 암호일 수도 있습니다. 보안 인증 정보가 손상되면 사용자 풀은 사용자가 제출한 암호를 읽고 암호 데이터베이스와 비교합니다. 작업으로 인해 암호가 손상되었을 가능성이

있는 것으로 확인되면 로그인을 차단하도록 사용자 풀을 구성한 다음 애플리케이션의 사용자에 대한 암호 재설정을 시작할 수 있습니다.

손상된 자격 증명 탐지는 새 사용자가 가입할 때, 기존 사용자가 로그인할 때, 사용자가 암호를 재설정하려고 할 때 안전하지 않은 암호에 대응할 수 있습니다. 이 기능을 사용하면 사용자 풀이 사용자가 암호를 입력할 때마다 안전하지 않은 암호로 로그인하는 것을 방지하거나 경고할 수 있습니다.

Amazon Cognito 콘솔에서 손상된 자격 증명 감지를 설정하려면

1. Plus 기능 플랜을 선택합니다.
2. 사용자 풀의 위협 방지 메뉴의 위협 방지에서 표준 및 사용자 지정 인증을 편집합니다.
3. 표준 또는 사용자 지정 인증에 대한 적용 모드를 전체 함수로 설정합니다.
4. 손상된 자격 증명에서 확인하려는 인증 작업 유형과 사용자 풀에서 원하는 자동 응답을 구성합니다.

자세히 알아보기

- 손상된 자격 증명 감지 작업

위협 방지: 사용자 활동 로깅

Plus 플랜에는 사용자 인증 시도에 대한 보안 분석 및 세부 정보를 제공하는 로깅 기능이 추가되었습니다. 애플리케이션에 연결된 디바이스에 대한 위험 평가, 사용자 IP 주소, 사용자 에이전트 및 기타 정보를 볼 수 있습니다. 내장된 위협 방지 기능을 사용하여 이 정보에 따라 조치를 취할 수도 있고, 자체 시스템의 로그를 분석하여 적절한 조치를 취할 수도 있습니다. 위협 방지에서 Amazon S3, CloudWatch Logs 또는 Amazon DynamoDB로 로그를 내보낼 수 있습니다.

Amazon Cognito 콘솔에서 사용자 활동 로깅을 설정하려면

1. Plus 기능 플랜을 선택합니다.
2. 사용자 풀의 위협 방지 메뉴의 위협 방지에서 표준 및 사용자 지정 인증을 편집합니다.
3. 표준 또는 사용자 지정 인증에 대한 적용 모드를 감사 전용으로 설정합니다. 로그의 최소 설정입니다. 전체 함수 모드에서 활성화하고 다른 위협 방지 기능을 구성할 수도 있습니다.
4. AWS 서비스 타사 분석을 위해 로그를 다른 로 내보내려면 사용자 풀의 로그 스트리밍 메뉴로 이동하여 내보내기 대상을 설정합니다.

자세히 알아보기

- 사용자 인증 이벤트 내보내기
- Amazon Cognito 사용자 풀에서 로그 내보내기

기능 플랜을 변경하기 위한 기능 끄기

기능 플랜은 사용자 풀에 구성 옵션을 추가합니다. 관련 기능 플랜이 활성화된 경우에만 이러한 기능을 구성하고 사용할 수 있습니다. 예를 들어 Plus 및 Essentials 플랜에서는 액세스 토큰 사용자 지정을 구성할 수 있지만 Lite 플랜에서는 구성할 수 없습니다. 이러한 기능을 비활성화하려면 각 활성 구성 요소를 비활성화해야 합니다. Amazon Cognito 콘솔의 설정 메뉴에 있는 전환 옵션은 기능 플랜을 변경하기 전에 비활성화해야 하는 기능을 알려줍니다. 이 장을 통해 사용자 풀 구성의 비활성화 변경 사항과 이러한 기능을 개별적으로 끄는 방법을 알아볼 수 있습니다.

액세스 토큰 사용자 지정

액세스 토큰 사용자 지정이 포함되지 않은 계획으로 전환하려면 사용자 풀에서 사전 토큰 생성 Lambda 트리거를 제거해야 합니다. 액세스 토큰 사용자 지정 없이 새 사전 토큰 생성 트리거를 추가하려면 트리거에 새 기능을 할당하고 `V1_0` 이벤트에 대해 구성합니다. 이러한 버전 1 트리거 이벤트는 ID 토큰에 대한 변경만 처리할 수 있습니다.

액세스 토큰 사용자 지정을 수동으로 비활성화하려면 사전 토큰 생성 트리거를 제거하고 새 버전 1 트리거를 추가합니다.

위협 방지

위협 방지 기능이 없는 플랜으로 전환하려면 사용자 풀의 위협 방지 메뉴에서 모든 기능을 비활성화합니다.

로그 내보내기

로그 내보내기 기능이 없는 플랜으로 전환하려면 사용자 풀의 로그 스트리밍 메뉴에서 기능을 비활성화합니다. 사용자 풀은 로컬 또는 내보낸 사용자 활동 로그를 더 이상 생성하지 않습니다. `EventSource` 값이 `UserActivity`인 구성을 제거하는 SetLogDeliveryConfiguration API 요청을 보낼 수도 있습니다.

이메일 MFA

이메일 MFA 기능이 없는 플랜으로 전환하려면 사용자 풀의 로그인 메뉴로 이동합니다. 다중 인증을 편집하고 사용 가능한 MFA 메서드 중 하나로 이메일 메시지를 선택 취소합니다.

Amazon Cognito 사용자 풀로 인증

Amazon Cognito에서는 다양한 방법으로 사용자를 인증할 수 있습니다. 사용자는 암호와 WebAuthn 패스키로 로그인할 수 있습니다. Amazon Cognito는 이메일 또는 SMS 메시지로 일회용 암호를 보낼 수 있습니다. 자체 문제 및 응답 시퀀스를 오케스트레이션하는 Lambda 함수를 구현할 수 있습니다. 다음은 인증 흐름입니다. 인증 흐름에서 사용자는 보안 암호를 제공하고 Amazon Cognito는 보안 암호를 확인한 다음 애플리케이션이 OIDC 라이브러리로 처리할 수 있도록 JSON 웹 토큰(JWT)을 발급합니다. 이 장에서는 다양한 애플리케이션 환경에서 다양한 인증 흐름에 맞게 사용자 풀 및 앱 클라이언트를 구성하는 방법을 설명합니다. 관리형 로그인의 호스팅 로그인 페이지를 사용하고 AWS SDK에서 자체 로직 및 프런트 엔드를 빌드하는 옵션을 다룹니다.

도메인 유무에 관계없이 모든 사용자 풀은 사용자 풀 API에서 사용자를 인증할 수 있습니다. 사용자 풀에 도메인을 추가하면 사용자 풀 엔드포인트를 사용할 수 있습니다. 사용자 풀 API는 다양한 권한 부여 모델 및 API 요청의 요청 흐름을 지원합니다.

Amazon Cognito는 사용자의 ID를 확인하기 위해 이메일, SMS 메시지, 일회용 암호 및 패스키와 같은 암호 외에도 여러 유형의 문제를 통합하는 인증 흐름을 지원합니다.

세부 정보를 구성하기 전에 애플리케이션에 맞는 인증 접근 방식을 선택합니다. 다음 결정 도우미는 두 개의 진입점과 사용자 지정 빌드 애플리케이션 내에서 두 개의 초기 로그인 흐름을 요약합니다. 각 옵션은 세부 섹션으로 연결됩니다.

Amazon Cognito에서 로그인 페이지를 호스팅하거나 자체 프런트 엔드를 빌드하도록 하시겠습니까?

- 가장 낮은 경로를 원하는 경우 관리형 로그인을 사용합니다. Amazon Cognito는 로그인, 로그아웃 및 암호 재설정 페이지를 호스팅하고 애플리케이션은 OpenID Connect(OIDC) 신뢰 당사자 라이브러리를 사용하여 결과를 처리합니다. 관리형 로그인은 사용자 풀 및 앱 클라이언트 구성에서 허용하는 인증 방법을 자동으로 제공합니다.

- 자체 UI를 빌드하고 로그인 로직을 제어하려면 AWS SDK를 사용합니다. 애플리케이션에서 사용자 풀 API를 직접 호출하고 챌린지-응답 로직을 구현합니다.

사용자 지정 빌드 애플리케이션에서 사용자로부터 로그인 방법을 수집하나요, 아니면 미리 선언하나요?

- 사용자에게 사용 가능한 로그인 방법 목록을 제공하고 선택하도록 하려면 선택 기반 인증(USER_AUTH흐름)을 선택합니다. 이 흐름은 암호 없는 패스키 로그인을 제공하는 유일한 방법입니다.

애플리케이션이 사용자가 로그인하고와 같이 미리 흐름을 선언하는 방법을 이미 알고 있는 경우 클라이언트 기반 인증을 선택합니다USER_SRP_AUTH. 클라이언트 기반 인증은 사용자 지정 인증 및 새로 고침 토큰 흐름을 사용하는 유일한 방법입니다.

선택 기반 인증을 위한 사전 조건

선택 기반 인증은 관리형 로그인과 AWS SDKs 모두에 적용됩니다. 이를 사용하려면 앱 클라이언트의 허용된 인증 흐름ALLOW_USER_AUTH에를 추가합니다. 암호 없는 및 패스키 메서드에는 관리형 로그인 도메인과 Lite 티어(Essentials 또는 Plus) 이상의 기능 계획이 추가로 필요합니다.

주제

- 앱 클라이언트를 사용한 애플리케이션별 설정
- 인증 흐름 구현
- 사용자 풀을 사용한 인증에 대해 알아야 할 사항
- 인증 세션의 예
- 관리형 로그인을 위한 인증 방법 구성
- AWS SDKs에서 인증 방법 관리
- 인증 흐름
- API 및 SDK 인증을 위한 권한 부여 모델
- 타사 ID 제공업체를 통한 사용자 풀 로그인

앱 클라이언트를 사용한 애플리케이션별 설정

사용자 풀 앱 클라이언트는 Amazon Cognito로 인증되는 모바일 또는 웹 애플리케이션과 상호 작용하는 사용자 풀 내의 구성입니다. 앱 클라이언트는 인증된 API 작업과 인증되지 않은 API 작업을 호출하고 사용자 속성 전체 또는 일부를 읽거나 수정할 수 있습니다. 등록, 로그인 및 암호 찾기 작업을 처리하려면 앱 클라이언트에 대해 앱이 자체 식별을 수행해야 합니다. 이러한 API 요청에는 앱 클라이언트 ID를 사용한 자체 식별 및 선택적 클라이언트 보안 암호를 사용한 권한 부여가 포함되어야 합니다. 개발자는 권한이 있는 클라이언트 앱만 이러한 미인증 작업을 호출할 수 있도록 앱 클라이언트 ID 또는 보안 암호를 보호할 책임이 있습니다. 또한 자격 AWS 증명으로 인증된 API 요청에 서명하도록 앱을 구성하는 경우 사용자 검사로부터 자격 증명을 보호해야 합니다.

사용자 풀에 여러 앱 클라이언트를 생성할 수 있습니다. 앱 클라이언트는 앱의 코드 플랫폼 또는 사용자 풀의 별도 테넌트에 연결할 수 있습니다. 예를 들어 서버 측 애플리케이션에 대한 앱과 다른 Android 앱을 생성할 수 있습니다. 각 앱에는 고유한 앱 클라이언트 ID가 있습니다.

앱 클라이언트 수준에서 다음 사용자 풀 기능에 대한 설정을 적용할 수 있습니다.

1. 분석
2. 관리형 로그인 IdP, 권한 부여 유형, 콜백 URL 및 사용자 지정
3. 리소스 서버 및 사용자 지정 범위
4. 위협 방지
5. 속성 읽기 및 쓰기 권한
6. 토큰 만료 및 취소
7. 인증 흐름

앱 클라이언트 유형

Amazon Cognito에서 앱 클라이언트를 생성할 때 표준 OAuth 클라이언트 유형 퍼블릭 클라이언트(public client) 및 기밀 클라이언트(confidential client)에 따라 옵션을 미리 채울 수 있습니다. 기밀 클라이언트(confidential client)를 클라이언트 보안 암호(client secret)와 함께 구성합니다. 클라이언트 유형에 대한 자세한 내용은 IETF RFC 6749 #2.1을 참조하세요.

퍼블릭 클라이언트(Public client)

퍼블릭 클라이언트는 브라우저 또는 모바일 디바이스에서 실행됩니다. 퍼블릭 클라이언트에는 신뢰할 수 있는 서버 측 리소스가 없으므로 클라이언트 보안 암호가 없습니다.

기밀 클라이언트(Confidential client)

기밀 클라이언트에는 인증되지 않은 API 작업에 대해 클라이언트 보안 암호(client secret)로 신뢰할 수 있는 서버 측 리소스가 있습니다. 앱은 백엔드 서버에서 데몬 또는 쉘 스크립트로 실행될 수 있습니다.

클라이언트 보안 암호(client secret)

클라이언트 보안 암호는 앱이 앱 클라이언트에 대한 모든 API 요청에서 사용해야 하는 고정 문자열입니다. 앱 클라이언트에는 `client_credentials`를 부여할 클라이언트 보안 암호가 있어야 합니다. 자세한 내용은 IETF RFC 6749 #2.3.1을 참조하세요.

각 앱 클라이언트는 한 번에 최대 2개의 보안 암호를 가질 수 있으므로 가동 중지 시간 없이 보안 암호를 교체할 수 있습니다. 앱 클라이언트를 생성할 때 Amazon Cognito가 보안 암호 값을 생성하도록 하거나 사용자 지정 보안 암호 값을 제공할 수 있습니다. 앱을 생성한 후에는 보안 암호를 변경할 수 없습니다. AddUserPoolClientSecret API 작업을 사용하여 두 번째 보안 암호를 추

가하여 보안 암호를 교체할 수 있습니다. 보안 암호를 추가할 때 Amazon Cognito가 보안 암호 값을 생성하도록 하거나 사용자 지정 보안 암호 값을 제공할 수 있습니다. 보안 암호를 삭제하려면 DeleteUserPoolClientSecret API 작업을 사용합니다. 앱 클라이언트와 연결된 유일한 보안 암호는 삭제할 수 없습니다. 앱을 삭제하여 해당 앱 클라이언트 ID를 사용하는 앱의 액세스를 차단할 수도 있습니다.

> **Note**
>
> Amazon Cognito 콘솔은 애플리케이션 유형에 대해 기존 웹 애플리케이션 및 머신 간 애플리케이션 옵션을 선택할 때 클라이언트 보안 암호가 있는 앱 클라이언트를 생성합니다. 다음 옵션 중 하나를 선택하여 클라이언트 암호를 생성하거나 CreateUserPoolClient를 사용하여 프로그래밍 방식으로 클라이언트를 생성하고 `GenerateSecret`를 `true`로 설정합니다.

> **Important**
>
> 앱 클라이언트를 생성한 후에는 앱 클라이언트에 클라이언트 보안 암호가 있는지 여부를 변경할 수 없습니다. UpdateUserPoolClient 작업은 클라이언트 암호를 추가하거나 제거할 수 없습니다. 이 구성을 변경하려면 새 앱 클라이언트를 생성합니다. 사용자 풀을 다시 생성할 필요가 없으며 다른 앱 클라이언트도 영향을 받지 않습니다.

기밀 클라이언트와 클라이언트 보안 암호를 공개 앱과 함께 사용할 수 있습니다. Amazon CloudFront 프록시를 사용하여 전송 중 `SECRET_HASH`를 추가합니다. 자세한 내용은 AWS 블로그에서 Protect public clients for Amazon Cognito by using an Amazon CloudFront proxy(Amazon CloudFront 프록시를 사용하여 Amazon Cognito에 대한 퍼블릭 클라이언트 보호)를 참조하세요.

JSON 웹 토큰

Amazon Cognito 앱 클라이언트는 다음 유형의 JSON 웹 토큰(JWT)을 발급할 수 있습니다.

자격 증명(ID) 토큰

사용자가 사용자 풀에서 인증되었음을 검증할 수 있는 명령문입니다. OpenID Connect(OIDC)는 OAuth 2.0에서 정의한 액세스 및 새로 고침 토큰 표준에 ID 토큰 사양을 추가했습니다. ID 토큰에는 앱이 사용자 프로필을 생성하고 리소스를 프로비저닝하는 데 사용할 수 있는 사용자 속성과 같은 자격 증명 정보가 들어 있습니다. 자세한 내용은 자격 증명(ID) 토큰 이해 섹션을 참조하세요.

액세스 토큰

사용자 액세스 권한에 대한 검증 가능한 문입니다. 액세스 토큰에는 OIDC 및 OAuth 2.0의 기능인 범위가 포함됩니다. 앱은 백엔드 리소스에 대한 범위를 제공하고 사용자 풀이 사용자 또는 시스템에 API의 데이터 또는 자체 사용자 데이터에 액세스할 수 있는 권한을 부여했음을 입증할 수 있습니다. 사용자 지정 범위가 있는 액세스 토큰(주로 M2M client-credentials 권한 부여)은 리소스 서버에 대한 액세스를 승인합니다. 자세한 내용은 액세스 토큰 이해 섹션을 참조하세요.

새로 고침 토큰

사용자 토큰이 만료될 때 앱이 사용자 풀에 제공할 수 있는 암호화된 초기 인증 문입니다. refresh-token 요청은 만료되지 않은 새 액세스 및 ID 토큰을 반환합니다. 자세한 내용은 새로 고침 토큰 섹션을 참조하세요.

Amazon Cognito 콘솔에 있는 사용자 풀의 앱 클라이언트 메뉴에서 각 앱 클라이언트에 대해 이러한 토큰의 만료를 설정할 수 있습니다.

앱 클라이언트 용어

다음 용어는 Amazon Cognito 콘솔에서 사용할 수 있는 앱 클라이언트 속성입니다.

허용된 콜백 URLs

콜백 URL은 로그인 성공 후 사용자가 리디렉션되는 위치를 나타냅니다. 하나 이상의 콜백 URL을 선택합니다. 콜백 URL은 다음을 수행해야 합니다.

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

OAuth 2.0 - 리디렉션 엔드포인트를 참조하세요.

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

`myapp://example` 같은 앱 콜백 URL도 지원됩니다.

허용된 로그아웃 URL

로그아웃 URL은 로그아웃 후 사용자가 리디렉션되는 위치를 나타냅니다.

속성 읽기 및 쓰기 권한

사용자 풀에는 각각 고유한 앱 클라이언트와 IdP를 가진 많은 고객이 있을 수 있습니다. 앱과 관련된 사용자 속성에 대해서만 읽기 및 쓰기 액세스 권한을 갖도록 앱 클라이언트를 구성할 수 있습니다. M2M(Machine-to-Machine) 인증과 같은 경우에는 사용자 속성에 대한 액세스 권한을 부여할 수 없습니다.

속성 읽기 및 쓰기 권한 구성 시 고려 사항

- 앱 클라이언트를 생성하고 속성 읽기 및 쓰기 권한을 사용자 지정하지 않으면 Amazon Cognito는 모든 사용자 풀 속성에 읽기 및 쓰기 권한을 부여합니다.
- 변경할 수 없는 사용자 지정 속성에 쓰기 액세스 권한을 부여할 수 있습니다. 사용자의 앱 클라이언트는 사용자를 생성 또는 가입시킬 때만 변경 불가능한 사용자 지정 속성에 값을 쓸 수 있습니다. 이후에는 사용자의 변경 불가능한 사용자 지정 속성에 값을 쓸 수 없습니다.
- 앱 클라이언트는 사용자 풀의 필수 속성에 대한 쓰기 권한을 가져야 합니다. Amazon Cognito 콘솔은 필수 속성을 쓰기 가능으로 자동 설정합니다.
- 앱 클라이언트가 `email_verified` 또는 `phone_number_verified`에 대한 쓰기 액세스 권한을 갖도록 허용할 수 없습니다. 사용자 풀 관리자는 이러한 값을 수정할 수 있습니다. 사용자는 속성 확인을 통해서만 이러한 속성의 값을 변경할 수 있습니다.

인증 흐름

앱 클라이언트에서 로그인을 허용하는 방법입니다. 앱은 사용자 이름 및 암호, 이메일 및 SMS 메시지 OTP, 패스키 인증자, Lambda 트리거를 통한 사용자 지정 인증, 토큰 새로 고침을 지원할 수 있습니다. 사용자 지정 빌드 애플리케이션에서 사용자 이름 및 암호 인증에 SRP 인증을 사용하는 것이 가장 좋습니다.

사용자 지정 범위

사용자 지정 범위는 리소스 서버에서 자체 리소스 서버에 대해 정의할 수 있습니다. 형식은 *`resource-server-identifier`*/*`scope`*입니다. 범위, M2M 및 리소스 서버 섹션을 참조하세요.

기본 리디렉션 URI

타사 IdP를 사용하는 사용자에 대한 인증 요청의 `redirect_uri` 파라미터를 대체합니다. CreateUserPoolClient 또는 UpdateUserPoolClient API 요청의 `DefaultRedirectURI` 파라미터를 사용하여 이 앱 클라이언트 설정을 구성합니다. 이 URL은 앱 클라이언트에 대한 `CallbackURLs`의 멤버여야 합니다. Amazon Cognito는 다음과 같은 경우 인증된 세션을 이 URL로 리디렉션합니다.

1. 앱 클라이언트에 하나의 ID 제공업체가 할당되고 여러 콜백 URL이 정의되어 있습니다. 사용자 풀은 `redirect_uri` 파라미터를 포함하지 않는 경우 권한 부여 서버에 대한 인증 요청을 기본 리디렉션 URI로 리디렉션합니다.
2. 앱 클라이언트에 하나의 ID 제공업체가 할당되고 하나의 콜백 URL이 정의되어 있습니다. 이 시나리오에서는 기본 콜백 URL을 정의할 필요가 없습니다. `redirect_uri` 파라미터를 포함하지 않는 요청은 사용 가능한 하나의 콜백 URL로 리디렉션됩니다.

자격 증명 공급자

사용자 풀 외부 ID 제공업체(IdP) 중 일부 또는 전체를 선택하여 사용자를 인증할 수 있습니다. 앱 클라이언트는 사용자 풀의 로컬 사용자만 인증할 수도 있습니다. 앱 클라이언트에 IdP 추가 시 IdP에 대한 권한 부여 링크를 생성하여 관리형 로그인 페이지에 표시할 수 있습니다. 다중 IdP를 할당할 수도 있지만 최소한 한 개는 할당해야 합니다. 외부 IdP 사용에 대한 자세한 내용은 타사 ID 제공업체를 통한 사용자 풀 로그인 섹션을 참조하세요.

OpenID Connect 범위

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

- `openid` 범위는 사용자가 ID 토큰과 사용자의 고유 ID를 검색하고 싶어 한다고 선언합니다. 또한 요청의 추가 범위에 따라 전체 또는 일부 사용자 속성을 요청합니다. Amazon Cognito는 사용자가 `openid` 범위를 요청할 때만 ID 토큰을 반환합니다. `openid` 범위는 만료 및 키 ID와 같은 구조적 ID 토큰 클레임을 승인하고 userInfo 엔드포인트의 응답으로 받는 사용자 속성을 결정합니다.
  - 요청한 유일한 범위가 `openid`가 되는 경우 Amazon Cognito는 현재 앱 클라이언트가 읽을 수 있는 모든 사용자 속성으로 ID 토큰을 채웁니다. 이 범위의 액세스 토큰에 대한 `userInfo` 응답만으로도 모든 사용자 속성이 반환됩니다.
  - `phone`, `email`, `profile` 또는 ID 토큰과 같은 다른 범위로 `openid`를 요청하고 사용자의 고유 ID와 추가 범위에 의해 정의된 속성을 `userInfo`가 반환하는 경우입니다.
- `phone` 범위는 `phone_number` 및 `phone_number_verified` 클레임에 대한 액세스 권한을 부여합니다. 이 범위는 `openid` 범위를 통해서만 요청할 수 있습니다.
- `email` 범위는 `email` 및 `email_verified` 클레임에 대한 액세스 권한을 부여합니다. 이 범위는 `openid` 범위를 통해서만 요청할 수 있습니다.
- `aws.cognito.signin.user.admin` 범위는 액세스 토큰이 필요한 UpdateUserAttributes 및 VerifyUserAttribute와 같은 Amazon Cognito 사용자 풀 API 작업에 대한 액세스 권한을 부여합니다.

- `profile` 범위는 클라이언트가 읽을 수 있는 모든 사용자 속성에 대한 액세스 권한을 부여합니다. 이 범위는 `openid` 범위를 통해서만 요청할 수 있습니다.

범위에 대한 자세한 내용은 표준 OIDC 범위 목록을 참조하세요.

OAuth 권한 부여 유형

OAuth 권한 부여는 사용자 풀 토큰을 검색하는 인증 방법입니다. Amazon Cognito는 다음 유형의 권한 부여를 지원합니다. 이러한 OAuth 권한 부여를 앱에 통합하려면 도메인을 사용자 풀에 추가해야 합니다.

인증 코드 권한 부여

인증 코드 권한 부여는 앱이 Token 엔드포인트를 사용하여 사용자 풀 토큰과 교환할 수 있는 코드를 생성합니다 인증 코드를 교환하면 앱이 ID, 액세스 및 새로 고침 토큰을 받습니다. 이 OAuth 흐름은 암시적 권한 부여와 마찬가지로 사용자의 브라우저에서 발생합니다. 인증 코드 권한 부여는 Amazon Cognito에서 제공하는 가장 안전한 권한 부여입니다. 토큰이 사용자 세션에 표시되지 않기 때문입니다. 대신 앱은 요청을 생성하여 토큰을 반환하고 이를 보호된 스토리지에 캐시합니다. 자세한 내용은 IETF 6749 #1.3.1의 인증 코드를 참조하세요.

> **Note**
>
> 퍼블릭 클라이언트 앱의 보안 모범 사례는 인증 코드 권한 부여 OAuth 흐름만 활성화하고 코드 교환용 증명 키(PKCE)를 구현하여 토큰 교환을 제한하는 것입니다. PKCE를 사용하는 경우 클라이언트가 토큰 엔드포인트에 원래 인증 요청에서 제시한 것과 동일한 보안 암호를 제공한 경우에만 인증 코드를 교환할 수 있습니다. PKCE에 대한 자세한 내용은 IETF RFC 7636을 참조하세요.

암시적 허용

암시적 권한 부여는 권한 부여 엔드포인트에서 직접 사용자의 브라우저 세션에 액세스 및 ID 토큰을 전달하지만 새로 고침 토큰은 전달하지 않습니다. 암시적 권한 부여를 사용하면 토큰 엔드포인트를 별도로 요청할 필요가 없지만, 암시적 권한 부여는 PKCE와 호환되지 않으며 새로 고침 토큰을 반환하지 않습니다. 이 권한 부여는 인증 코드 권한 부여를 완료할 수 없는 테스트 시나리오와 앱 아키텍처에도 적용할 수 있습니다. 자세한 내용은 IETF RFC 6749 #1.3.2의 암시적 권한 부여를 참조하세요. 인증 코드 권한 부여 및 암시적 권한 부여를 모두 활성화하고 각 권한 부여를 필요한 경우 사용할 수 있습니다.

클라이언트 보안 인증 권한 부여

클라이언트 보안 인증 권한 부여는 M2M(Machine-to-Machine) 통신에 사용됩니다. 인증 코드 및 암시적 권한 부여는 인증된 인간 사용자에게 토큰을 발급합니다. 클라이언트 보안 인증은 비대화형 시스템에서 API에 범위 기반 권한을 부여합니다. 앱은 토큰 엔드포인트에서 직접 클라이언트 보안 인증 정보를 요청하고 액세스 토큰을 받을 수 있습니다. 자세한 내용은 IETF RFC 6749 #1.3.4의 클라이언트 보안 인증을 참조하세요. 클라이언트 보안 암호가 있고 인증 코드 또는 암시적 권한 부여를 지원하지 않는 앱 클라이언트에서만 클라이언트 자격 증명 권한 부여를 활성화할 수 있습니다. 사용자 풀 도메인이 필요하지 않은 대안으로 클라이언트 보안 암호와 `ALLOW_CLIENT_TOKEN_AUTH` 인증 흐름이 있는 앱 클라이언트는 GetClientToken API 작업을 통해 M2M 액세스 토큰을 얻을 수 있습니다.

> **Note**
>
> 사용자 자격으로 클라이언트 보안 인증 정보 흐름을 호출하지 않으므로, 이 권한 부여는 액세스 토큰에 사용자 지정 범위만 추가할 수 있습니다. 사용자 지정 범위는 자체 리소스 서버에 대해 정의할 수 있습니다. `openid` 및 `profile` 같은 기본 범위는 비 인간 사용자에게 적용되지 않습니다.
>
> ID 토큰은 사용자 속성의 검증이므로 M2M 통신과 관련이 없으며 클라이언트 자격 증명 부여는 사용자 속성을 발급하지 않습니다. 범위, M2M 및 리소스 서버 섹션을 참조하세요.

클라이언트 자격 증명 부여는 AWS 청구서에 비용을 추가합니다. 자세한 내용은 Amazon Cognito 요금을 참조하세요.

앱 클라이언트 생성

AWS Management Console

앱 클라이언트를 생성하려면(콘솔)

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 자격 AWS 증명을 입력합니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다. 두 옵션 모두 애플리케이션별 설정으로 앱 클라이언트를 구성하라는 메시지를 표시합니다.
4. 애플리케이션 아키텍처를 반영하는 애플리케이션 유형을 선택합니다.

> **Important**
>
> 애플리케이션 유형 선택은 Amazon Cognito가 클라이언트 암호를 생성하는지 여부를 포함하여 다른 앱 클라이언트 설정의 기본값도 설정합니다. 기존 웹 애플리케이션 및 Machine-to-machine 애플리케이션 유형은 클라이언트 암호를 생성합니다. 자세한 내용은 앱 클라이언트 유형 섹션을 참조하세요.

5. 표시 식별자를 사용하여 애플리케이션의 이름을 지정합니다.
6. 반환 URL을 입력합니다.
7. 앱 클라이언트 생성을 선택합니다. 앱 클라이언트를 생성한 후 고급 옵션을 변경할 수 있습니다.
8. Amazon Cognito에서 앱 클라이언트 세부 정보로 돌아갑니다. 애플리케이션의 예제 코드에 액세스하려면 빠른 설정 가이드 탭에서 플랫폼을 선택합니다.

AWS CLI

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

> **Note**
>
> CLI가 원격 파라미터 파일로 처리하지 않도록 콜백 및 로그아웃 URL에 JSON 형식을 사용합니다.
>
> ```
> --callback-urls "["https://example.com"]"
> --logout-urls "["https://example.com"]"
> ```

자세한 내용은 AWS CLI 명령 참조를 참조하세요. create-user-pool-client

Amazon Cognito user pools API

CreateUserPoolClient API 요청을 생성합니다. 기본값으로 설정하지 않으려는 모든 파라미터의 값을 지정해야 합니다.

사용자 풀 앱 클라이언트(AWS CLI 및 AWS API) 업데이트

에 다음 명령을 AWS CLI입력합니다.

```
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 성공하면가 확인을 AWS CLI 반환합니다.

```
{
    "UserPoolClient": {
        "ClientId": "MyClientID",
        "SupportedIdentityProviders": [
            "LoginWithAmazon",
            "MySAMLIdP"
        ],
        "CallbackURLs": [
            "https://example.com"
        ],
        "AllowedOAuthScopes": [
            "openid"
        ],
        "ClientName": "Example",
        "AllowedOAuthFlows": [
            "implicit",
            "code"
        ],
        "RefreshTokenValidity": 30,
        "AuthSessionValidity": 3,
        "CreationDate": 1524628110.29,
        "AllowedOAuthFlowsUserPoolClient": true,
        "UserPoolId": "MyUserPoolID",
        "LastModifiedDate": 1530055177.553
    }
}
```

자세한 내용은 update-user-pool-client AWS CLI 명령 참조를 참조하세요.

AWS API: UpdateUserPoolClient

사용자 풀 앱 클라이언트(AWS CLI 및 AWS API)에 대한 정보 가져오기

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

자세한 내용은 AWS CLI 명령 참조를 참조하세요. describe-user-pool-client.

AWS API: DescribeUserPoolClient

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

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

자세한 내용은 list-user-pool-clients AWS CLI 명령 참조를 참조하세요. list-user-pool-clients

AWS API: ListUserPoolClients

사용자 풀 앱 클라이언트(AWS CLI 및 AWS API) 삭제

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

자세한 내용은 AWS CLI 명령 참조를 참조하세요. delete-user-pool-client

AWS API: DeleteUserPoolClient

인증 흐름 구현

인증용 AWS SDK를 사용하여 관리형 로그인을 구현하든 사용자 지정 구축 애플리케이션 프런트엔드를 구현하든 관계없이 구현하려는 인증 유형에 맞게 앱 클라이언트를 구성해야 합니다. 다음 정보는 앱 클라이언트 및 애플리케이션의 인증 흐름 설정을 설명합니다.

App client supported flows

> Amazon Cognito 콘솔 또는 AWS SDK의 API를 사용하여 앱 클라이언트에 지원되는 흐름을 구성할 수 있습니다. 이러한 흐름을 지원하도록 앱 클라이언트를 구성한 후 애플리케이션에 배포할 수 있습니다.
>
> 다음 절차에서는 Amazon Cognito 콘솔을 사용하여 앱 클라이언트에 사용할 수 있는 인증 흐름을 구성합니다.

인증 흐름에 맞게 앱 클라이언트를 구성하는 방법(콘솔)

1. 에 로그인 AWS 하고 Amazon Cognito 사용자 풀 콘솔로 이동합니다. 사용자 풀을 선택하거나 새로운 사용자 풀을 생성합니다.
2. 사용자 풀 구성에서 앱 클라이언트 메뉴를 선택합니다. 앱 클라이언트를 선택하거나 새 앱 클라이언트를 생성합니다.
3. 앱 클라이언트 정보에서 편집을 선택합니다.
4. 앱 클라이언트 흐름에서 지원하려는 인증 흐름을 선택합니다.

인증 흐름에 대한 앱 클라이언트를 구성하는 방법(API/SDK)

Amazon Cognito API를 사용하여 앱 클라이언트에 사용할 수 있는 인증 흐름을 구성하려면 CreateUserPoolClient 또는 UpdateUserPoolClient 요청에서 `ExplicitAuthFlows` 값을 설정합니다. 다음은 클라이언트에 보안 원격 암호(SRP) 및 선택 기반 인증을 프로비저닝하는 예입니다.

```
"ExplicitAuthFlows": [
   "ALLOW_USER_AUTH",
   "ALLOW_USER_SRP_AUTH"
]
```

앱 클라이언트 지원 흐름을 구성할 때 다음 옵션과 API 값을 지정할 수 있습니다.

앱 클라이언트 흐름 지원

인증 흐름	호환성	콘솔	API
선택 기반 인증	서버 측, 클라이언트 측	로그인 시 인증 유형 선택	`ALLOW_USER_AUTH`
영구 암호로 로그인	클라이언트 측	사용자 이름과 암호로 로그인	`ALLOW_USER_PASSWORD_AUTH`
영구 암호 및 보안 페이로드로 로그인	서버 측, 클라이언트 측	보안 원격 암호(SRP)로 로그인	`ALLOW_USER_SRP_AUTH`
토큰 새로 고침	서버 측, 클라이언트 측	기존의 인증된 세션에서 새 사용자 토큰 가져오기	`ALLOW_REFRESH_TOKEN_AUTH`

인증 흐름	호환성	콘솔	API
서버 측 인증	서버 측	서버 측 관리 자격 증명으로 로그인	ALLOW_ADMIN_USER_PASSWORD_AUTH
사용자 지정 인증	서버 측과 클라이언트 측 맞춤형 애플리케이션. 관리형 로그인과 호환되지 않습니다.	Lambda 트리거의 사용자 지정 인증 흐름으로 로그인	ALLOW_CUSTOM_AUTH

Implement flows in your application

관리형 로그인을 사용하면 구성된 인증 옵션을 로그인 페이지에서 자동으로 사용할 수 있습니다. 사용자 지정 빌드 애플리케이션에서 초기 흐름 선언으로 인증을 시작합니다.

- 사용자의 흐름 옵션 목록에서 선택하려면 USER_AUTH 흐름으로 선택 기반 인증을 선언합니다. 이 흐름에는 패스키 및 암호 없는 인증과 같이 클라이언트 기반 인증 흐름에서 사용할 수 없는 사용 가능한 인증 방법이 있습니다.
- 인증 흐름을 미리 선택하려면 앱 클라이언트에서 사용할 수 있는 다른 흐름으로 클라이언트 기반 인증을 선언합니다.

사용자를 로그인할 때 InitiateAuth 또는 AdminInitiateAuth 요청 본문에 AuthFlow 파라미터가 포함되어야 합니다.

선택 기반 인증:

```
"AuthFlow": "USER_AUTH"
```

SRP를 사용한 클라이언트 기반 인증:

```
"AuthFlow": "USER_SRP_AUTH"
```

사용자 풀을 사용한 인증에 대해 알아야 할 사항

Amazon Cognito 사용자 풀을 사용한 인증 모델 설계에서 다음 정보를 고려하세요.

관리형 로그인 및 호스팅 UI의 인증 흐름

관리형 로그인에는 클래식 호스팅 UI보다 더 많은 인증 옵션이 있습니다. 예를 들어 사용자는 관리형 로그인에서만 암호 없는 패스키 인증을 수행할 수 있습니다.

AWS SDK 인증에서만 사용 가능한 사용자 지정 인증 흐름

관리형 로그인 또는 클래식 호스팅 UI로는 사용자 지정 인증 흐름 또는 Lambda 트리거를 사용한 사용자 지정 인증을 수행할 수 없습니다. AWS SDK를 사용한 인증에서 사용자 지정 인증을 사용할 수 있습니다.

외부 ID 제공업체(IdP) 로그인을 위한 관리형 로그인

AWS SDKs를 사용한 인증에서는 타사 IdPs 통해 사용자를 로그인할 수 없습니다. 관리형 로그인 또는 클래식 호스팅 UI를 구현하고 IdP로 리디렉션한 다음 애플리케이션에서 OIDC 라이브러리를 사용하여 결과 인증 객체를 처리해야 합니다. 관리형 로그인에 대한 자세한 정보는 사용자 풀 관리형 로그인 섹션을 참조하세요.

다른 사용자 기능에 대한 암호 없는 인증 효과

사용자 풀 및 앱 클라이언트에서 일회용 암호 또는 패스키를 사용하여 암호 없는 로그인을 활성화하면 사용자 생성 및 마이그레이션에 영향을 미칩니다. 암호 없는 로그인이 활성화된 경우:

1. 관리자는 암호 없이 사용자를 생성할 수 있습니다. 기본 초대 메시지 템플릿이 {###} 암호 자리 표시자를 더 이상 포함하지 않도록 변경됩니다. 자세한 내용은 관리자로 사용자 계정 생성 섹션을 참조하세요.
2. SDK 기반 SignUp 작업의 경우 사용자가 가입할 때 암호를 제공할 필요가 없습니다. 관리형 로그인 및 호스팅 UI는 암호 없는 인증이 허용되는 경우에도 가입 페이지에 암호가 필요합니다. 자세한 내용은 사용자 계정 가입 및 확인 섹션을 참조하세요.
3. CSV 파일에서 가져온 사용자는 속성에 사용 가능한 암호 없는 로그인 옵션에 대한 이메일 주소 또는 전화번호가 포함된 경우 암호 재설정 없이 암호 없는 옵션으로 즉시 로그인할 수 있습니다. 자세한 내용은 CSV 파일에서 사용자 풀로 사용자 가져오기 섹션을 참조하세요.
4. 암호 없는 인증은 사용자 마이그레이션 Lambda 트리거를 호출하지 않습니다.
5. 일회용 암호(OTP) 첫 번째 팩터로 로그인하는 사용자는 세션에 멀티 팩터 인증(MFA) 팩터를 추가할 수 없습니다. 사용자 확인이 포함된 패스키는 로 구성된 경우 MFA 요구 사항을 충족할 수 있습니다`MULTI_FACTOR_WITH_USER_VERIFICATION`.

패스키 신뢰 당사자 URL은 공개 접미사 목록에 있을 수 없습니다.

`www.example.com`과 같이 소유한 도메인 이름을 패스키 구성에서 신뢰 당사자(RP) ID로 사용할 수 있습니다. 이 구성은 소유한 도메인에서 실행되는 사용자 지정 빌드 애플리케이션을 지원하기

위한 것입니다. 퍼블릭 접미사 목록 또는 PSL에는 보호된 상위 수준 도메인이 포함되어 있습니다. Amazon Cognito는 RP URL을 PSL의 도메인으로 설정하려고 할 때 오류를 반환합니다.

인증 세션 흐름 기간

사용자 풀의 기능에 따라 앱이 Amazon Cognito에서 토큰을 검색하기 `RespondToAuthChallenge` 전에 `InitiateAuth` 및에 대한 여러 문제에 대응할 수 있습니다. Amazon Cognito는 각 요청에 대한 응답에 세션 문자열을 포함합니다. API 요청을 인증 흐름에 결합하려면 이전 요청에 대한 응답의 세션 문자열을 각 후속 요청에 포함합니다. 기본적으로 사용자는 세션 문자열이 만료되기 전에 3분 동안 각 챌린지를 완료해야 합니다. 이 기간을 조정하려면 앱 클라이언트 Authentication flow session duration(인증 흐름 세션 기간)을 변경합니다. 다음 절차에서는 앱 클라이언트 구성에서 이 설정을 변경하는 방법을 설명합니다.

> **Note**
>
> 인증 흐름 세션 기간 설정은 Amazon Cognito 사용자 풀 API를 사용한 인증에 적용됩니다. 관리형 로그인은 최종 사용자가 임시 암호로 처음 로그인하는 동안 다중 인증의 경우 세션 기간을 3분, 강제 암호 변경의 경우 8분으로 설정합니다.

Amazon Cognito console

앱 클라이언트 인증 흐름 세션 기간을 구성하는 방법(AWS Management Console)

1. 사용자 풀의 App integration(앱 통합) 탭에서 App clients and analytics(앱 클라이언트 및 분석) 컨테이너의 앱 클라이언트 이름을 선택합니다.
2. 앱 클라이언트 정보 컨테이너에서 편집을 선택합니다.
3. 인증 흐름 세션 기간 값을 SMS 및 이메일 MFA 코드에 대해 원하는 유효 기간(분)으로 변경합니다. 이렇게 하면 사용자가 앱 클라이언트에서 인증 챌린지를 완료해야 하는 시간도 변경됩니다.
4. 변경 사항 저장을 선택합니다.

User pools API

앱 클라이언트 인증 흐름 세션 기간을 구성하는 방법(Amazon Cognito API)

1. `DescribeUserPoolClient` 요청의 기존 사용자 풀 설정으로 `UpdateUserPoolClient` 요청을 준비합니다. `UpdateUserPoolClient` 요청에는 모든 기존 앱 클라이언트 속성이 포함되어야 합니다.
2. `AuthSessionValidity` 값을 SMS MFA 코드에 대해 원하는 유효 기간(분)으로 변경합니다. 이렇게 하면 사용자가 앱 클라이언트에서 인증 챌린지를 완료해야 하는 시간도 변경됩니다.

앱 클라이언트에 대한 자세한 내용은 앱 클라이언트를 사용한 애플리케이션별 설정 섹션을 참조하세요.

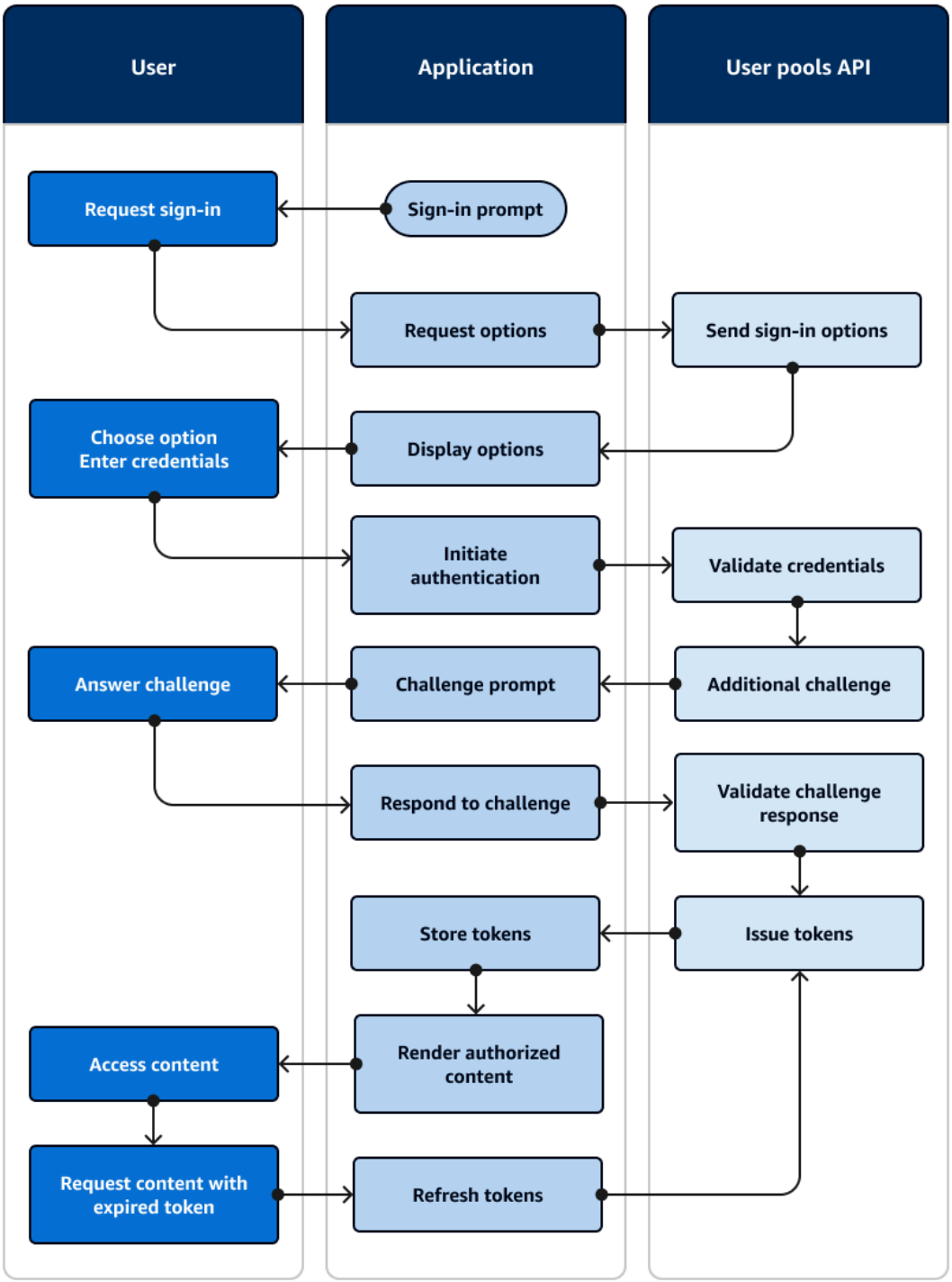
로그인 시도 실패에 대한 잠금 동작

인증되지 않은 API 작업 또는 IAM 인증 API 작업으로 요청되었는지 여부에 관계없이 사용자 암호로 로그인 시도가 5회 실패하면 Amazon Cognito는 1초 동안 사용자를 잠급니다. 이후 한 번 더 실패할 때마다 잠금 시간이 두 배로 늘어나 최대 15분 정도까지 늘어납니다.

잠금 기간 중에 시도하면 `Password attempts exceeded` 예외가 발생하며, 이후 잠금 기간에 영향을 주지는 않습니다. 누적 로그인 시도 실패 n(`Password attempts exceeded`예외 제외)회에 대해, Amazon Cognito는 2^ (n-5)초 동안 사용자를 차단합니다. 잠금을 n=0 초기 상태로 재설정하려면 사용자는 잠금 기간이 만료된 후 성공적으로 로그인하거나 잠금이 설정된 후 연속으로 15분 동안 로그인 시도를 하지 않아야 합니다. 이러한 동작은 변경될 수 있습니다. 암호 기반 인증도 수행하지 않으면 이 동작은 사용자 지정 챌린지에는 적용되지 않습니다.

인증 세션의 예

다음 다이어그램과 단계별 가이드는 사용자가 애플리케이션에 로그인하는 일반적인 시나리오를 보여줍니다. 예시 애플리케이션은 사용자에게 몇 가지 로그인 옵션을 제공합니다. 자격 증명을 입력하여 하나를 선택하고, 추가 인증 요소를 제공하고, 로그인합니다.



사용자가 사용자 이름과 암호로 로그인하거나, 이메일 메시지에서 일회용 코드를 요청하거나, 지문 옵션을 선택할 수 있는 로그인 페이지가 있는 애플리케이션을 가정해 보겠습니다.

1. 로그인 프롬프트: 애플리케이션에 로그인 버튼이 있는 홈 화면이 표시됩니다.

2. 로그인 요청: 사용자가 로그인을 선택합니다. 쿠키 또는 캐시에서 애플리케이션은 사용자 이름을 검색하거나 입력하라는 메시지를 표시합니다.
3. 요청 옵션: 애플리케이션에서 `USER_AUTH` 흐름이 포함된 `InitiateAuth` API 요청으로 사용자의 로그인 옵션을 요청하여 사용자에게 사용 가능한 로그인 방법을 요청합니다.
4. 로그인 옵션 전송: Amazon Cognito는 `PASSWORD`, `EMAIL_OTP` 및 `WEB_AUTHN`으로 응답합니다. 응답에는 다음 응답에서 재생할 수 있는 세션 식별자가 포함되어 있습니다.
5. 표시 옵션: 애플리케이션에 사용자가 사용자 이름과 암호를 입력하거나, 일회용 코드를 가져오거나, 지문을 스캔할 수 있는 UI 요소가 표시됩니다.
6. 옵션/자격 증명 입력 선택: 사용자가 사용자 이름과 암호를 입력합니다.
7. 인증 시작: 애플리케이션에서 사용자 이름-암호 로그인을 확인하고 사용자 이름과 암호를 제공하는 `RespondToAuthChallenge` API 요청과 함께 사용자의 로그인 정보를 제공합니다.
8. 자격 증명 검증: Amazon Cognito가 사용자의 자격 증명을 확인합니다.
9. 추가 문제: 사용자에게 인증 앱으로 구성된 다중 인증이 있습니다. Amazon Cognito는 `SOFTWARE_TOKEN_MFA` 문제를 반환합니다.
10. 문제 프롬프트: 애플리케이션에 사용자의 인증 앱에서 시간 기반 일회용 암호(TOTP)를 요청하는 양식이 표시됩니다.
11. 문제 답변: 사용자가 TOTP를 제출합니다.
12. 문제에 응답: 다른 `RespondToAuthChallenge` 요청에서 애플리케이션은 사용자의 TOTP를 제공합니다.
13. 문제 응답 검증: Amazon Cognito는 사용자의 코드를 확인하고 현재 사용자에게 추가 문제를 발행하지 않도록 사용자 풀이 구성되어 있는지 확인합니다.
14. 문제 토큰: Amazon Cognito는 ID, 액세스 및 새로 고침 JSON 웹 토큰(JWT)을 반환합니다. 사용자의 초기 인증이 완료되었습니다.
15. 토큰 저장: 애플리케이션은 사용자 데이터를 참조하고, 리소스에 대한 액세스를 승인하고, 토큰이 만료되면 토큰을 업데이트할 수 있도록 사용자의 토큰을 캐싱합니다.
16. 승인된 콘텐츠 렌더링: 애플리케이션은 ID 및 역할을 기반으로 리소스에 대한 사용자의 액세스를 결정하고 애플리케이션 콘텐츠를 제공합니다.
17. 액세스 콘텐츠: 사용자가 로그인하고 애플리케이션 사용을 시작합니다.
18. 토큰이 만료된 콘텐츠 요청: 나중에 사용자가 권한이 필요한 리소스를 요청합니다. 사용자의 캐시된 토큰이 만료되었습니다.
19. 새로 고침 토큰: 애플리케이션에서 사용자의 저장된 새로 고침 토큰을 사용하여 `InitiateAuth` 요청을 만듭니다.

20.문제 토큰: Amazon Cognito는 새로운 ID와 액세스 JWT를 반환합니다. 자격 증명에 대한 추가 프롬프트 없이 사용자의 세션이 안전하게 새로 고쳐집니다.

AWS Lambda 트리거를 사용하여 사용자가 인증하는 방식을 사용자 지정할 수 있습니다. 이러한 트리거는 인증 흐름의 일부로서 고유 챌린지를 발행 및 확인할 수 있습니다.

안전한 백엔드 서버에 관리자 인증 흐름을 사용할 수도 있습니다. 사용자 마이그레이션 인증 흐름을 사용하면 사용자가 암호를 재설정할 필요 없이 사용자 마이그레이션을 수행할 수 있습니다.

관리형 로그인을 위한 인증 방법 구성

사용자가 로그인, 로그아웃 또는 암호를 재설정하도록 하려는 경우 사용자 풀 인증을 위한 웹 프런트 엔드인 관리형 로그인 페이지를 호출할 수 있습니다. 이 모델에서 애플리케이션은 OIDC 라이브러리를 가져와 사용자 풀 관리형 로그인 페이지로 브라우저 기반 인증 시도를 처리합니다. 사용자가 사용할 수 있는 인증 형식은 사용자 풀 및 앱 클라이언트의 구성에 따라 다릅니다. 앱 클라이언트에서 `ALLOW_USER_AUTH` 흐름을 구현하면 Amazon Cognito는 사용자에게 사용 가능한 옵션에서 로그인 방법을 선택하라는 메시지를 표시합니다. `ALLOW_USER_PASSWORD_AUTH`를 구현하고 SAML 제공업체를 할당하면 로그인 페이지에 사용자 이름과 암호를 입력하거나 IdP에 연결할 수 있는 옵션이 표시됩니다.

Amazon Cognito 사용자 풀 콘솔을 사용하면 애플리케이션에 대한 관리형 로그인 인증을 설정할 수 있습니다. 새 사용자 풀을 생성할 때 개발 중인 플랫폼을 지정하면 콘솔에서 로그인 및 로그아웃 흐름을 구현하기 위한 스타터 코드를 사용하여 OIDC 및 OAuth 라이브러리를 구현하는 예제를 제공합니다. 많은 OIDC 신뢰 당사자 구현을 통해 관리형 로그인을 구축할 수 있습니다. 가능하면 인증된 OIDC 신뢰 당사자 라이브러리로 작업하는 것이 좋습니다. 자세한 내용은 Amazon Cognito 콘솔에서 새 애플리케이션 생성 섹션을 참조하세요.

일반적으로 OIDC 신뢰 당사자 라이브러리는 사용자 풀의 `.well-known/openid-configuration` 엔드포인트를 주기적으로 확인하여 토큰 엔드포인트 및 권한 부여 엔드포인트와 같은 발급자 URLs을 결정합니다. 가장 좋은 방법은 옵션이 있는 경우이 자동 검색 동작을 구현하는 것입니다. 발급자 엔드포인트를 수동으로 구성하면 오류가 발생할 수 있습니다. 예를 들어 사용자 풀 도메인을 변경할 수 있습니다. 의 경로`openid-configuration`는 사용자 풀 도메인에 연결되지 않으므로 서비스 엔드포인트를 자동으로 검색하는 애플리케이션이 도메인을 자동으로 선택합니다.

관리형 로그인에 대한 사용자 풀 설정

애플리케이션의 여러 공급자로 로그인을 허용하거나 Amazon Cognito를 독립 사용자 디렉터리로 사용할 수 있습니다. 사용자 속성을 수집하거나, MFA를 설정 및 프롬프트하거나, 이메일 주소를 사용자 이

름으로 요구할 수도 있습니다. 관리형 로그인 및 호스팅 UI의 필드는 직접 편집할 수 없습니다. 대신 사용자 풀의 구성은 관리형 로그인 인증 흐름의 처리를 자동으로 설정합니다.

다음 사용자 풀 구성 항목은 Amazon Cognito가 관리형 로그인 및 호스팅 UI에서 사용자에게 제공하는 인증 방법을 결정합니다.

User pool options (Sign-in menu)

다음 옵션은 Amazon Cognito 콘솔에서 사용자 풀의 로그인 메뉴에 있습니다.

Cognito 사용자 풀 로그인 옵션

사용자 이름에 대한 옵션이 있습니다. 관리형 로그인 및 호스팅 UI 페이지는 선택한 형식의 사용자 이름만 허용합니다. 예를 들어 이메일을 유일한 로그인 옵션으로 사용하여 사용자 풀을 설정하면 관리형 로그인 페이지는 이메일 형식의 사용자 이름만 허용합니다.

필수 속성

사용자 풀에서 필요에 따라 속성을 설정하면 관리형 로그인은 사용자가 가입할 때 해당 속성에 대한 값을 입력하라는 메시지를 표시합니다.

선택 기반 로그인 옵션

선택 기반 인증의 인증 방법에 대한 설정이 있습니다. 여기서 패스키 및 암호 없는과 같은 인증 방법을 켜거나 끌 수 있습니다. 이러한 방법은 Lite 계층을 초과하는 관리형 로그인 도메인 및 기능 플랜이 있는 사용자 풀에서만 사용할 수 있습니다.

다중 인증

관리형 로그인 및 호스팅 UI는 MFA에 대한 등록 및 인증 작업을 처리합니다. 사용자 풀에 MFA가 필요한 경우 로그인 페이지에 추가 요소를 설정하라는 메시지가 자동으로 표시됩니다. 또한 MFA 구성을 가진 사용자에게 MFA 코드로 인증을 완료하라는 메시지를 표시합니다. 사용자 풀에서 MFA가 꺼져 있거나 선택 사항인 경우 로그인 페이지에 MFA를 설정하라는 메시지가 표시되지 않습니다.

사용자 계정 복구

사용자 풀의 셀프 서비스 계정 복구 설정은 로그인 페이지에 사용자가 암호를 재설정할 수 있는 링크가 표시되는지 여부를 결정합니다.

User pool options (Domain menu)

다음 옵션은 Amazon Cognito 콘솔에서 사용자 풀의 도메인 메뉴에 있습니다.

도메인

사용자 풀 도메인을 선택하면 인증을 위해 브라우저를 호출할 때 사용자가 여는 링크의 경로가 설정됩니다.

브랜딩 버전

브랜딩 버전을 선택하면 사용자 풀 도메인에 관리형 로그인 또는 호스팅 UI가 표시되는지 여부가 결정됩니다.

User pool options (Social and external providers menu)

다음 옵션은 Amazon Cognito 콘솔에서 사용자 풀의 소셜 및 외부 공급자 메뉴에 있습니다.

공급자

사용자 풀에 추가하는 ID 제공업체(idP)는 사용자 풀의 각 앱 클라이언트에 대해 활성 또는 비활성 상태로 둘 수 있습니다.

App client options

다음 옵션은 Amazon Cognito 콘솔에서 사용자 풀의 앱 클라이언트 메뉴에 있습니다. 이러한 옵션을 검토하려면 목록에서 앱 클라이언트를 선택합니다.

빠른 설정 가이드

빠른 설정 가이드에는 다양한 개발자 환경에 대한 코드 예제가 있습니다. 여기에는 관리형 로그인 인증을 애플리케이션과 통합하는 데 필요한 라이브러리가 포함되어 있습니다.

앱 클라이언트 정보

이 구성을 편집하여 현재 앱 클라이언트로 표시되는 애플리케이션에 할당된 IdP를 설정합니다. 관리형 로그인 페이지에서 Amazon Cognito는 사용자에 대한 선택 사항을 표시합니다. 이러한 선택은 할당된 메서드 및 IdP에서 결정됩니다. 예를 들어 `MySAML`이라는 SAML 2.0 IdP와 로컬 사용자 풀 로그인을 할당하면 관리형 로그인 페이지에 인증 방법 프롬프트와 `MySAML`에 대한 버튼이 표시됩니다.

인증 설정

이 구성을 편집하여 애플리케이션에 대한 인증 방법을 설정합니다. 관리형 로그인 페이지에서 Amazon Cognito는 사용자에 대한 선택 사항을 표시합니다. 이러한 선택은 사용자 풀을 IdP로 사용

할 수 있는지 여부와 할당하는 방법에 따라 결정됩니다. 예를 들어 선택 기반 `ALLOW_USER_AUTH` 인증을 할당하면 관리형 로그인 페이지에 이메일 주소 입력 및 패스키로 로그인과 같은 사용 가능한 선택 항목이 표시됩니다. 관리되는 로그인 페이지는 할당된 IdP에 대한 버튼도 렌더링합니다.

로그인 페이지

이 탭에서 사용할 수 있는 옵션을 사용하여 관리형 로그인 또는 호스팅 UI 사용자 대화형 페이지의 시각적 효과를 설정합니다. 자세한 내용은 관리형 로그인 페이지에 브랜딩 적용 섹션을 참조하세요.

AWS SDKs에서 인증 방법 관리

Amazon Cognito 사용자 풀의 사용자는 다양한 초기 로그인 옵션 또는 요인으로 로그인할 수 있습니다. 일부 요소의 경우 사용자는 다중 인증(MFA)으로 후속 조치를 취할 수 있습니다. 이러한 첫 번째 요인에는 사용자 이름과 암호, 일회용 암호, 패스키 및 사용자 지정 인증이 포함됩니다. 자세한 내용은 인증 흐름 섹션을 참조하세요. 애플리케이션에 기본 제공 UI 구성 요소가 있고 AWS SDK 모듈을 가져오는 경우 인증을 위한 애플리케이션 로직을 구축해야 합니다. 두 가지 기본 방법 중 하나를 선택해야 하며, 해당 방법에서 구현하려는 인증 메커니즘을 선택해야 합니다.

애플리케이션 또는 클라이언트가 인증 유형을 미리 선언하는 클라이언트 기반 인증을 구현할 수 있습니다. 다른 옵션은 앱이 사용자 이름을 수집하고 사용자에게 사용 가능한 인증 유형을 요청하는 선택 기반 인증입니다. 요구 사항에 따라 동일한 애플리케이션에서 이러한 모델을 함께 구현하거나 앱 클라이언트 간에 분할할 수 있습니다. 각 메서드에는 클라이언트 기반 사용자 지정 인증 및 선택 기반 암호 없는 인증과 같이 고유한 기능이 있습니다.

사용자 풀 API의 AWS SDK 구현으로 인증을 수행하는 사용자 지정 빌드 애플리케이션에서는 사용자 풀 구성, 앱 클라이언트 구성 및 클라이언트 측 기본 설정에 맞게 API 요청을 구성해야 합니다. `USER_AUTH`의 `AuthFlow`로 시작하는 `InitiateAuth` 세션은 선택 기반 인증을 시작합니다. Amazon Cognito는 선호하는 인증 방법 또는 선택 목록의 문제로 API에 응답합니다. `CUSTOM_AUTH`의 `AuthFlow`로 시작하는 세션은 Lambda 트리거를 사용한 사용자 지정 인증으로 바로 전환됩니다.

일부 인증 방법은 두 흐름 유형 중 하나로 고정되며, 일부 방법은 둘 다에서 사용할 수 있습니다.

주제

- 선택 기반 인증
- 클라이언트 기반 인증
- USER_AUTH 흐름을 사용한 단계적 인증

선택 기반 인증

애플리케이션은 선택 기반 인증에서 다음 인증 방법을 요청할 수 있습니다. InitiateAuth 또는 AdminInitiateAuth의 PREFERRED_CHALLENGE 파라미터 또는 RespondToAuthChallenge 또는 AdminRespondToAuthChallenge의 ChallengeName 파라미터에서 이러한 옵션을 선언합니다.

> **Important**
>
> 선택 기반 로그인에는 두 가지 사전 조건이 있습니다. 앱 클라이언트는 ALLOW_USER_AUTH 인증 흐름을 활성화해야 하며 암호 없는 및 패스키 메서드(EMAIL_OTP, SMS_OTP, WEB_AUTHN)에는 Essentials 또는 Plus 기능 플랜의 사용자 풀이 필요합니다. 이는 Lite 티어에서 사용할 수 없습니다. PASSWORD (SRP 포함)는 모든 기능 플랜에서 사용할 수 있습니다. 사용자에게 선택 기반 메서드가 표시되지 않는 경우 추가 문제 해결을 수행하기 전에 앱 클라이언트 흐름과 기능 계획을 모두 확인합니다.

1. EMAIL_OTP 및 SMS_OTP

   일회용 암호를 통한 암호 없는 로그인

2. WEB_AUTHN

   WebAuthn 패스키를 사용한 암호 없는 로그인

3. PASSWORD

   영구 암호로 로그인

   영구 암호 및 보안 페이로드로 로그인

   로그인 후 MFA

API 컨텍스트에서 이러한 옵션을 검토하려면 RespondToAuthChallenge의 ChallengeName 섹션을 참조하세요.

선택 기반 로그인은 초기 요청에 대한 응답으로 문제를 실행합니다. 이 문제는 요청된 옵션을 사용할 수 있는지 확인하거나 사용 가능한 선택 사항 목록을 제공합니다. 애플리케이션은 이러한 선택 사항을 사용자에게 표시한 다음 선호하는 로그인 방법에 대한 자격 증명을 입력하고 문제 응답에서 인증을 진행할 수 있습니다.

인증 흐름에는 다음과 같은 선택 기반 옵션이 있습니다. 이 유형의 모든 요청은 앱이 먼저 사용자 이름을 수집하거나 캐시에서 검색해야 합니다.

1. AuthParameters의 USERNAME만 있는 요청 옵션. Amazon Cognito는 SELECT_CHALLENGE 문제를 반환합니다. 여기에서 애플리케이션은 사용자에게 문제를 선택하고 이 응답을 사용자 풀에 반환하라는 메시지를 표시할 수 있습니다.
2. PREFERRED_CHALLENGE의 AuthParameters와 원하는 문제의 파라미터를 사용하여 원하는 문제를 요청합니다. 예를 들어 PASSWORD_SRP의 PREFERRED_CHALLENGE를 요청하는 경우 SRP_A도 포함해야 합니다. 사용자, 사용자 풀, 앱 클라이언트가 모두 선호하는 문제에 대해 구성된 경우 Amazon Cognito는 해당 문제의 다음 단계로 응답합니다. 예를 들어 PASSWORD_SRP 흐름에서는 PASSWORD_VERIFIER, EMAIL_OTP 및 SMS_OTP 흐름에서는 CodeDeliveryDetails로 응답합니다. 선호하는 문제를 사용할 수 없는 경우 Amazon Cognito는 SELECT_CHALLENGE 및 사용 가능한 문제 목록으로 응답합니다.
3. 먼저 사용자를 로그인한 다음 선택 기반 인증 옵션을 요청합니다. 로그인한 사용자의 액세스 토큰이 포함된 GetUserAuthFactors 요청은 사용 가능한 선택 기반 인증 요소와 MFA 설정을 반환합니다. 이 옵션을 사용하면 사용자는 먼저 사용자 이름과 암호로 로그인한 다음 다른 형태의 인증을 활성화할 수 있습니다. 이 작업을 사용하여 선호하는 문제로 로그인한 사용자의 추가 옵션을 확인할 수도 있습니다.

선택 기반 인증을 위해 앱 클라이언트를 구성하려면 허용된 인증 흐름에 ALLOW_USER_AUTH를 추가합니다. 또한 사용자 풀 구성에서 허용할 선택 기반 요소를 선택해야 합니다. 다음 프로세스는 선택 기반 인증 요소를 선택하는 방법을 보여줍니다.

Amazon Cognito console

사용자 풀에서 선택 기반 인증 옵션을 구성하려면

1. 에 로그인 AWS 하고 Amazon Cognito 사용자 풀 콘솔로 이동합니다. 사용자 풀을 선택하거나 새로운 사용자 풀을 생성합니다.
2. 사용자 풀 구성에서 로그인 메뉴를 선택합니다. 선택 기반 로그인 옵션을 찾아 편집을 선택합니다.
3. 암호 옵션은 항상 사용할 수 있습니다. 여기에는 PASSWORD 및 PASSWORD_SRP 흐름이 포함됩니다. 사용자의 옵션에 추가할 추가 옵션을 선택합니다. WEB_AUTHN용 패스키, EMAIL_OTP용 이메일 메시지 일회용 암호, SMS_OTP용 SMS 메시지 일회용 암호를 추가할 수 있습니다.
4. 변경 사항 저장을 선택합니다.

API/SDK

다음 부분 CreateUserPool 또는 UpdateUserPool 요청 본문은 선택 기반 인증에 사용할 수 있는 모든 옵션을 구성합니다.

```
"Policies": {
    "SignInPolicy": {
        "AllowedFirstAuthFactors": [
            "PASSWORD",
            "WEB_AUTHN",
            "EMAIL_OTP",
            "SMS_OTP"
        ]
    }
},
```

클라이언트 기반 인증

클라이언트 기반 인증은 다음 인증 흐름을 지원합니다. InitiateAuth 또는 AdminInitiateAuth의 `AuthFlow` 파라미터에서 이러한 옵션을 선언합니다.

1. `USER_PASSWORD_AUTH` 및 `ADMIN_USER_PASSWORD_AUTH`

   영구 암호로 로그인

   로그인 후 MFA

   이 인증 흐름은 선택 기반 인증의 `PASSWORD`와 동일합니다.
2. `USER_SRP_AUTH`

   영구 암호 및 보안 페이로드로 로그인

   로그인 후 MFA

   이 인증 흐름은 선택 기반 인증의 `PASSWORD_SRP`와 동일합니다.
3. `REFRESH_TOKEN_AUTH`

   새로 고침 토큰

   이 인증 흐름은 클라이언트 기반 인증에서만 사용할 수 있습니다.
4. `CUSTOM_AUTH`

사용자 지정 인증

이 인증 흐름은 클라이언트 기반 인증에서만 사용할 수 있습니다.

클라이언트 기반 인증을 사용하면 Amazon Cognito는 사용자가 인증 흐름을 시작하기 전에 인증할 방법을 결정했다고 가정합니다. 사용자가 제공하려는 로그인 팩터를 결정하는 로직은 기본 설정 또는 사용자 지정 프롬프트로 결정한 다음 사용자 풀에 대한 첫 번째 요청에서 선언해야 합니다. `InitiateAuth` 요청은 `USER_SRP_AUTH`와 같이 나열된 옵션 중 하나에 직접 해당하는 로그인 `AuthFlow`를 선언합니다. 이 선언을 사용하면 요청에 `USERNAME`, `SECRET_HASH` 및 `SRP_A`와 같이 인증을 시작하는 파라미터도 포함됩니다. Amazon Cognito는 SRP의 경우 `PASSWORD_VERIFIER`, TOTP MFA를 사용한 암호 로그인의 경우 `SOFTWARE_TOKEN_MFA`와 같은 추가적인 문제로 이 요청에 응답할 수 있습니다.

클라이언트 기반 인증을 위해 앱 클라이언트를 구성하려면 허용된 인증 흐름에 `ALLOW_USER_AUTH` 이외의 모든 인증 흐름을 추가합니다. 예: `ALLOW_USER_PASSWORD_AUTH`, `ALLOW_CUSTOM_AUTH`, `ALLOW_REFRESH_TOKEN_AUTH`. 클라이언트 기반 인증 흐름을 허용하려면 추가 사용자 풀 구성이 필요하지 않습니다.

USER_AUTH 흐름을 사용한 단계적 인증

InitiateAuth 및 RespondToAuthChallenge 작업을 사용하여 기존 `USER_AUTH` 흐름에서 단계적 인증을 요청합니다. 새 API 작업은 없습니다. 해당 서버 측 작업인 AdminInitiateAuth 및 AdminRespondToAuthChallenge는 동일한 단계적 파라미터 및 동작을 지원하며 선택적으로 액세스 토큰을 가져올 수도 `AdminInitiateAuth` 있습니다. 이 흐름은 사용자에게 기존 ID 토큰이 있는 경우와 사용자가 처음으로 대상 수준에 대해 인증하는 경우 모두 작동합니다. 흐름에 대한 자세한 내용은 섹션을 참조하세요인증 세션의 예.

ACR 수준 및 AMR 값에 대한 자세한 내용은 섹션을 참조하세요the section called “ACR 및 AMR 클레임을 사용한 인증 수준”.

> **Note**
>
> 단계적 인증에는 Essentials 또는 Plus 기능 계획이 필요합니다.

`InitiateAuth` (또는 `AdminInitiateAuth`) 요청의 `AuthParameters` 맵에 기존 `MAX_AGE`, 및와 함께 새 `TARGET_ACR_VALUES` 및 `USERNAME`라는 단계적 파라미터를 전달합니

다ACCESS_TOKENPREFERRED_CHALLENGE. 각 파라미터의 정의 및 제약 조건은 Amazon Cognito 사용자 풀 API 참조의 InitiateAuth 및 AdminInitiateAuth를 참조하세요. Amazon Cognito

다음 표에서는 Amazon Cognito가 이러한 파라미터의 조합에 응답하는 방법을 설명합니다.

USER_AUTH 단계적 인증 동작

파라미터	동작
TARGET_ACR_VALUES 및 ACCESS_TOKEN , MAX_AGE 만족	Amazon Cognito는 토큰의 현재 요인에서 우선순위가 가장 높은 만족 수준으로 사용자를 단계적으로 높입니다.
TARGET_ACR_VALUES 및 ACCESS_TOKEN , 만족MAX_AGE하지 않음	Amazon Cognito는 처음부터 대상 수준으로 사용자를 인증합니다.
TARGET_ACR_VALUES 없음 ACCESS_TOKEN	Amazon Cognito는 처음부터 대상 수준으로 사용자를 인증합니다.
두 파라미터 모두	USER_AUTH 흐름은 단계적 인증 없이 작동하는 것처럼 동작합니다.

단계적 인증 흐름을 설계할 때는 다음 동작을 염두에 두세요.

- 단계적 인증은 all-or-nothing. 목표 수준에 도달하기 위해 암호 및 레벨 4의 TOTP와 같은 여러 챌린지가 필요하고 사용자가 챌린지에 실패하면 단계적 확대 시도가 완전히 실패합니다. Amazon Cognito는 중간 토큰을 발급하지 않습니다.
- 제공된 토큰이 이미 요청된 수준을 충족하거나 초과하는 경우 Amazon Cognito는 새 토큰을 발급하는 대신 오류를 반환합니다. 이렇게 하면 애플리케이션이 단계적 인증을 토큰 새로 고침 메커니즘으로 사용하는 것을 방지할 수 있습니다.
- 사용자는 요청된 수준을 충족하는 요소가 이미 있어야 합니다. 요청된 대상 수준을 충족하도록 구성된 요소가 사용자에게 없는 경우 Amazon Cognito는 오류를 반환합니다. 단계적 인증은 사용자에게 새 요소를 설정하라는 메시지를 표시하지 않습니다.
- MFA는 바닥입니다. 사용자 또는 사용자 풀의 MFA 설정에 MFA가 필요한 경우 대상 수준이 낮고 MFA가 필요하지 않은 경우에도 초기 로그인은 여전히 해당 MFA 챌린지를 적용하고 프롬프트합니다. 요청된 대상 ACR 수준에 MFA가 필요하지 않은 경우에도 위협 방지(고급 보안 기능) 프롬프트가 표시되는 MFA는 초기 로그인 시에도 적용됩니다.는 필요한 MFA 프롬프트를 억제할 TARGET_ACR_VALUES 수 없습니다. 대상은 Amazon Cognito에서 프롬프트를 표시하는 몇 가지 활

성화된 MFA 요인 중 어느 것에 영향을 미칠 수 있습니다. MFA를 사용해야 하는 사용자에 대해 레벨 1과 같은 하위 레벨을 대상으로 하는 단계적 요청이 있는 경우 Amazon Cognito는 필요한 MFA를 자동으로 적용합니다. 사용자를 단계적으로 늘리고 오류를 반환하지 않습니다.

를 사용하여 단계적 인증을 요청하려면 Essentials 또는 Plus 기능 계획이 TARGET_ACR_VALUES 필요합니다. 자세한 내용은 기능 계획 요구 사항 섹션을 참조하세요.

인증 흐름

Amazon Cognito 사용자 풀을 사용한 인증 프로세스는 사용자가 초기 선택을 하고, 자격 증명을 제출하고, 추가 문제에 대응하는 흐름으로 가장 잘 설명할 수 있습니다. 애플리케이션에서 관리형 로그인 인증을 구현하면 Amazon Cognito가 이러한 프롬프트 및 문제의 흐름을 관리합니다. 애플리케이션 백엔드에서 AWS SDK로 흐름을 구현할 때는 요청 로직을 구성하고, 사용자에게 입력을 요청하고, 문제에 대응해야 합니다.

애플리케이션 관리자는 사용자 특성, 보안 요구 사항 및 권한 부여 모델을 통해 사용자가 로그인하도록 허용하는 방법을 결정할 수 있습니다. 스스로 다음과 같은 질문을 해보세요.

- 사용자가 다른 ID 제공업체(idP)의 자격 증명으로 로그인하도록 허용하려고 하는가?
- 사용자 이름과 암호만으로 ID를 증명할 수 있는가?
- 사용자 이름 암호 인증에 대한 인증 요청을 가로챌 수 있는가? 애플리케이션이 암호를 전송하거나 해시 및 솔트를 사용하여 인증을 협상하도록 해야 하는가?
- 사용자가 암호 입력을 건너뛰고 로그인하는 일회용 암호를 받도록 허용하려고 하는가?
- 사용자가 지문, 얼굴 또는 하드웨어 보안 키로 로그인하도록 허용하려고 하는가?
- 다중 인증(MFA)이 필요한 경우는 언제인가?
- 자격 증명을 다시 프롬프트하지 않고 사용자 세션을 유지할 것인가?
- Amazon Cognito의 기본 기능 이상으로 권한 부여 모델을 확장하려고 하는가?

이러한 질문에 대한 답변을 받으면 관련 기능을 활성화하고 애플리케이션이 수행하는 인증 요청에 구현하는 방법을 배울 수 있습니다.

사용자에 대한 로그인 흐름을 설정한 후 GetUserAuthFactors API 작업에 대한 요청을 통해 MFA 및 선택 기반 인증 요소의 현재 상태를 확인할 수 있습니다. 이 작업을 수행하려면 로그인한 사용자의 액세스 토큰으로 인증해야 합니다. 사용자 인증 요소 및 MFA 설정을 반환합니다.

주제

- 타사 IdP로 로그인
- 영구 암호로 로그인
- 영구 암호 및 보안 페이로드로 로그인
- 일회용 암호를 통한 암호 없는 로그인
- WebAuthn 패스키를 사용한 암호 없는 로그인
- 로그인 후 MFA
- 새로 고침 토큰
- 사용자 지정 인증
- 사용자 마이그레이션 인증 흐름

타사 IdP로 로그인

Amazon Cognito 사용자 풀은 Sign in with Apple, Login with Amazon, OpenID Connect(OIDC) 서비스와 같은 IdP 간 인증 세션의 중간 브로커 역할을 합니다. 이 프로세스를 페더레이션 로그인 또는 페더레이션 인증이라고도 합니다. 페더레이션 인증은 앱 클라이언트에 빌드할 수 있는 인증 흐름을 사용하지 않습니다. 대신 구성된 사용자 풀 IdP 앱 클라이언트에 할당합니다. 페더레이션 로그인은 사용자가 관리형 로그인에서 IdP를 선택하거나 애플리케이션이 IdP 로그인 페이지로 리디렉션된 세션을 호출할 때 발생합니다.

페더레이션 로그인을 사용하면 기본 및 MFA 인증 요소를 사용자의 IdP에 위임합니다. Amazon Cognito는 로컬 사용자에 연결하지 않는 한이 섹션의 다른 고급 흐름을 페더레이션 사용자에게 추가하지 않습니다. 연결되지 않은 페더레이션 사용자는 사용자 이름이 있지만 브라우저 기반 흐름과 관계없이 일반적으로 로그인에 사용되지 않는 매핑된 속성 데이터의 저장소입니다.

구현 리소스

- 타사 ID 제공업체를 통한 사용자 풀 로그인

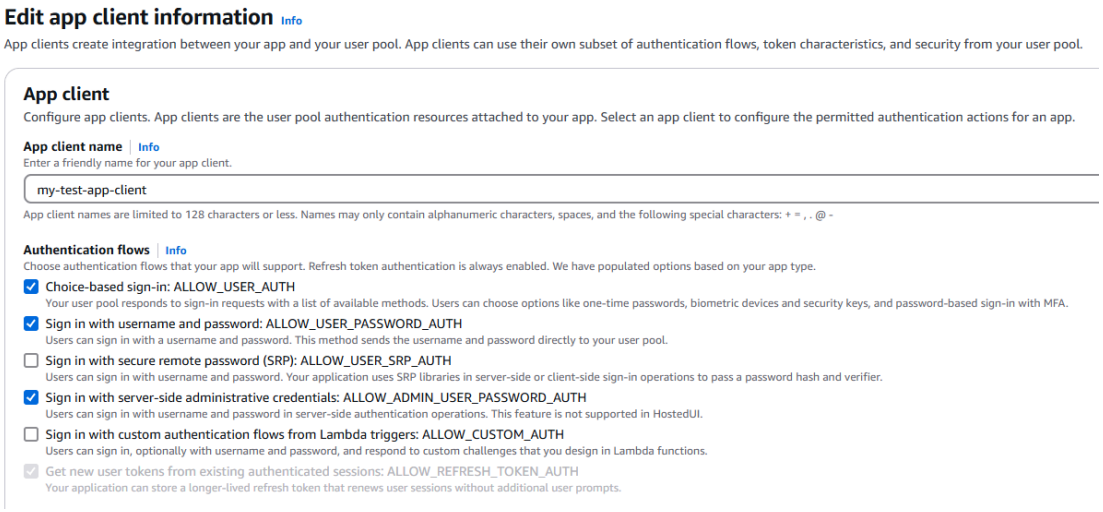
영구 암호로 로그인

Amazon Cognito 사용자 풀에서는 모든 사용자에게 사용자 이름이 있습니다. 전화번호, 이메일 주소 또는 선택한 식별자 또는 관리자가 제공한 식별자일 수 있습니다. 이 유형의 사용자는 사용자 이름과 암호로 로그인하고 선택적으로 MFA를 제공할 수 있습니다. 사용자 풀은 퍼블릭 또는 IAM 인증 API 작업 및 SDK 메서드를 사용하여 사용자 이름 암호 로그인을 수행할 수 있습니다. 애플리케이션은 인증을 위해 암호를 사용자 풀로 직접 전송할 수 있습니다. 사용자 풀은 인증 성공의 결과인 추가 문제 또는 JSON 웹 토큰(JWT)으로 응답합니다.

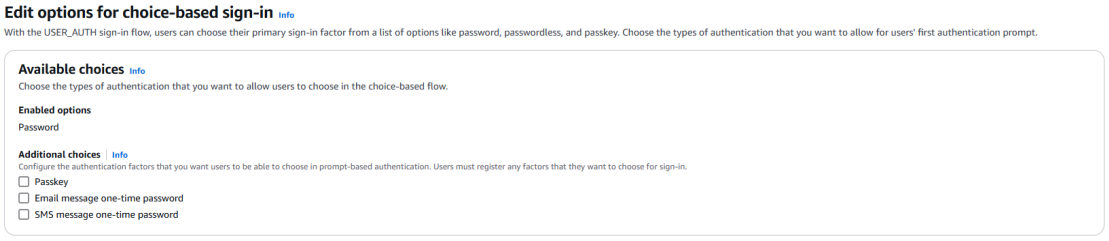
Activate password sign-in

사용자 이름과 암호를 사용하여 클라이언트 기반 인증을 활성화하려면 앱 클라이언트가 이를 허용하도록 구성합니다. Amazon Cognito 콘솔에서 사용자 풀 구성의 애플리케이션 아래에 있는 앱 클라이언트 메뉴로 이동합니다. 클라이언트 측 모바일 또는 네이티브 앱에 대한 일반 암호 로그인을 허용하려면 앱 클라이언트를 편집하고 인증 흐름에서 사용자 이름과 암호로 로그인: ALLOW_USER_PASSWORD_AUTH를 선택합니다. 서버 측 앱에 대한 일반 암호 로그인을 허용하려면 앱 클라이언트를 편집하고 서버 측 관리 자격 증명으로 로그인: ALLOW_ADMIN_USER_PASSWORD_AUTH를 선택합니다.

사용자 이름과 암호로 선택 기반 인증을 활성화하려면 앱 클라이언트가 이를 허용하도록 구성합니다. 앱 클라이언트를 편집하고 선택 기반 로그인: ALLOW_USER_AUTH를 선택합니다.



선택 기반 인증 흐름에서 암호 인증을 사용할 수 있는지 확인하려면 로그인 메뉴로 이동하여 선택 기반 로그인 옵션 아래의 섹션을 검토하세요. 사용 가능한 선택 항목에 암호가 표시되는 경우 일반 암호 인증으로 로그인할 수 있습니다. 암호 옵션에는 일반 및 SRP 사용자 이름-암호 인증 변형이 포함됩니다.



CreateUserPoolClient 또는 UpdateUserPoolClient 요청에서 원하는 username-and-password 인증 옵션으로 `ExplicitAuthFlows`를 구성합니다.

```
"ExplicitAuthFlows": [
```

```
   "ALLOW_USER_PASSWORD_AUTH",
   "ALLOW_ADMIN_USER_PASSWORD_AUTH",
   "ALLOW_USER_AUTH"
]
```

CreateUserPool 또는 UpdateUserPool 요청에서 지원하려는 선택 기반 인증 흐름으로 `Policies`를 구성합니다. `AllowedFirstAuthFactors`의 `PASSWORD` 값에는 일반 암호 및 SRP 인증 흐름 옵션이 모두 포함됩니다.

```
"Policies": {
   "SignInPolicy": {
      "AllowedFirstAuthFactors": [
         "PASSWORD",
         "EMAIL_OTP",
         "WEB_AUTHN"
      ]
   }
}
```

Choice-based sign-in with a password

사용자 이름 암호 인증으로 애플리케이션에 로그인하려면 다음과 같이 AdminInitiateAuth 또는 InitiateAuth 요청의 본문을 구성합니다. 현재 사용자가 사용자 이름 암호 인증을 받을 수 있는 경우 이 로그인 요청이 성공하거나 다음 문제를 계속 진행합니다. 그렇지 않으면 사용 가능한 기본 인증 문제 목록으로 응답합니다. 이 파라미터 세트는 로그인에 필요한 최소 파라미터입니다. 추가 파라미터를 사용할 수 있습니다.

```
{
   "AuthFlow": "USER_AUTH",
   "AuthParameters": {
      "USERNAME" : "testuser",
      "PREFERRED_CHALLENGE" : "PASSWORD",
      "PASSWORD" : "[User's password]"
   },
   "ClientId": "1example23456789"
}
```

`PREFERRED_CHALLENGE` 값을 생략하고 사용자의 적격 로그인 요인 목록이 포함된 응답을 받을 수도 있습니다.

```
{
```

```
   "AuthFlow": "USER_AUTH",
   "AuthParameters": {
      "USERNAME" : "testuser"
   },
   "ClientId": "1example23456789"
}
```

선호하는 문제를 제출하지 않았거나 제출된 사용자가 선호하는 문제에 적합하지 않은 경우 Amazon Cognito는 `AvailableChallenges`에서 옵션 목록을 반환합니다. `AvailableChallenges`에 `PASSWORD`의 `ChallengeName`이 포함된 경우 다음 형식으로 RespondToAuthChallenge 또는 AdminRespondToAuthChallenge 문제 응답을 사용하여 인증을 계속할 수 있습니다. 문제 응답을 초기 로그인 요청에 대한 API 응답과 연결하는 `Session` 파라미터를 전달해야 합니다. 이 파라미터 세트는 로그인에 필요한 최소 파라미터입니다. 추가 파라미터를 사용할 수 있습니다.

```
{
   "ChallengeName": "PASSWORD",
   "ChallengeResponses": {
      "USERNAME" : "testuser",
      "PASSWORD" : "[User's Password]"
   },
   "ClientId": "1example23456789",
   "Session": "[Session ID from the previous response"
}
```

Amazon Cognito는 토큰 또는 다중 인증(MFA)과 같은 추가 필수 문제를 사용하여 적격하고 성공적인 선호하는 문제 요청 및 `PASSWORD` 문제 응답에 응답합니다.

Client-based sign-in with a password

사용자 이름 암호 인증으로 클라이언트 측 앱에 사용자를 로그인하려면 다음과 같이 InitiateAuth 요청의 본문을 구성합니다. 이 파라미터 세트는 로그인에 필요한 최소 파라미터입니다. 추가 파라미터를 사용할 수 있습니다.

```
{
   "AuthFlow": "USER_PASSWORD_AUTH",
   "AuthParameters": {
      "USERNAME" : "testuser",
      "PASSWORD" : "[User's password]"
   },
   "ClientId": "1example23456789"
```

```
}
```

사용자 이름 암호 인증으로 서버 측 앱에 사용자를 로그인하려면 다음과 같이 AdminInitiateAuth 요청의 본문을 구성합니다. 애플리케이션이 자격 AWS 증명으로이 요청에 서명해야 합니다. 이 파라미터 세트는 로그인에 필요한 최소 파라미터입니다. 추가 파라미터를 사용할 수 있습니다.

```
{
   "AuthFlow": "ADMIN_USER_PASSWORD_AUTH",
   "AuthParameters": {
      "USERNAME" : "testuser",
      "PASSWORD" : "[User's password]"
   },
   "ClientId": "1example23456789"
}
```

Amazon Cognito는 토큰 또는 다중 인증(MFA)과 같은 추가 필수 문제를 사용하여 성공적인 요청에 응답합니다.

영구 암호 및 보안 페이로드로 로그인

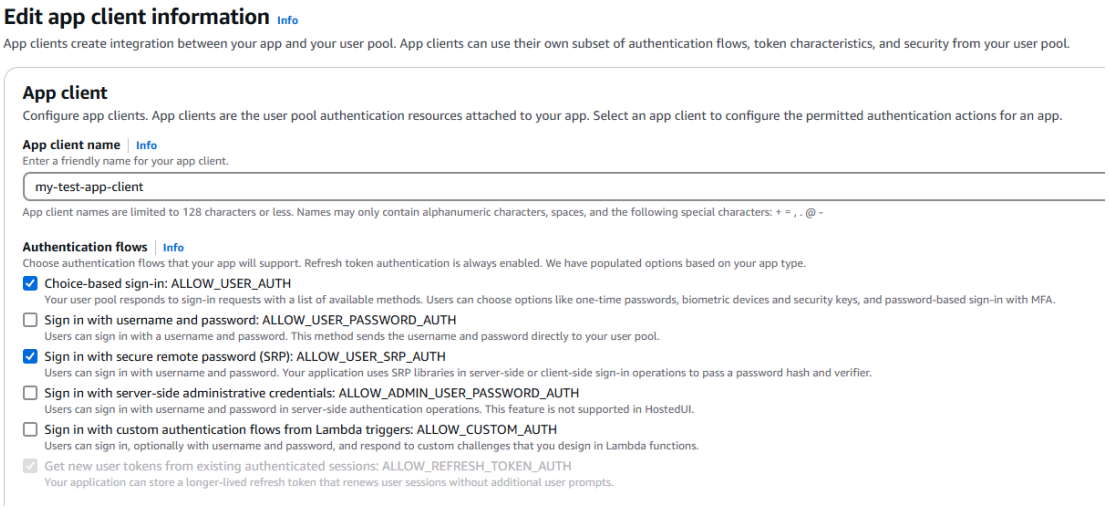
사용자 풀에서 사용자 이름-암호 로그인 방법의 또 다른 형태는 보안 원격 암호(SRP) 프로토콜을 사용하는 것입니다. 이 옵션은 사용자 풀이 확인할 수 있는 암호 해시 및 솔트와 같은 암호에 대한 지식 증명을 전송합니다. Amazon Cognito에 대한 요청에 읽을 수 있는 보안 암호 정보가 없는 경우 애플리케이션은 사용자가 입력한 암호를 처리하는 유일한 엔터티입니다. SRP 인증에는 SDK에서 가져올 수 있는 기존 구성 요소에서 가장 잘 수행되는 수학적 계산이 포함됩니다. 하위 수준 AWS SDKs는 `InitiateAuth` 및 `RespondToAuthChallenge` API 작업을 노출하지만 SRP 값은 계산하지 않습니다. Amplify 라이브러리와 같이 SRP를 구현하는 상위 수준 라이브러리는 이러한 계산을 수행합니다. SDK가 SRP 구현을 제공하지 않는 경우 이러한 작업을 호출하기 전에 SRP 값을 직접 계산해야 합니다. SRP는 일반적으로 모바일 앱과 같은 클라이언트 측 애플리케이션에서 구현됩니다. 프로토콜에 대한 자세한 내용은 Stanford SRP 홈페이지를 참조하세요. Wikipedia에는 리소스와 예시도 있습니다. 인증 흐름에 대한 SRP 계산을 수행하는 데 다양한 퍼블릭 라이브러리를 사용할 수 있습니다.

Amazon Cognito 인증의 시작-문제-응답 시퀀스는 SRP를 사용하여 사용자와 암호를 검증합니다. SRP 인증을 지원하도록 사용자 풀 및 앱 클라이언트를 구성한 다음 애플리케이션에서 로그인 요청 및 문제 응답 로직을 구현해야 합니다. SRP 라이브러리는 사용자의 암호를 소유하고 있음을 사용자 풀에 보여주는 난수와 계산된 값을 생성할 수 있습니다. 애플리케이션은 이러한 계산된 값을 Amazon Cognito 사용자 풀 API 작업 및 인증을 위한 SDK 메서드의 JSON 형식 `AuthParameters` 및 `ChallengeParameters` 필드에 채웁니다.

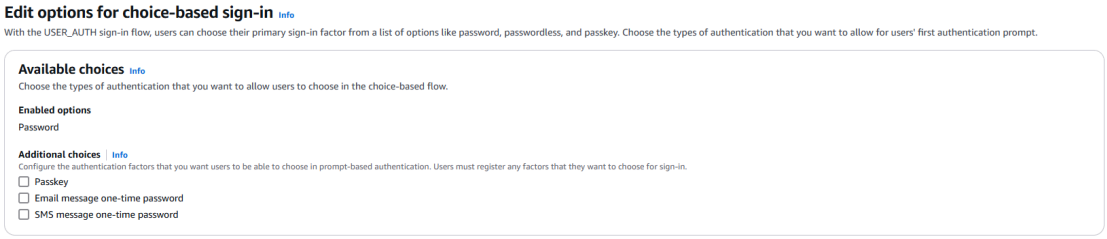
Activate SRP sign-in

사용자 이름과 SRP를 사용하여 클라이언트 기반 인증을 활성화하려면 앱 클라이언트가 이를 허용하도록 구성합니다. Amazon Cognito 콘솔에서 사용자 풀 구성의 애플리케이션 아래에 있는 앱 클라이언트 메뉴로 이동합니다. 클라이언트 측 모바일 또는 네이티브 앱에 대한 SRP 로그인을 허용하려면 앱 클라이언트를 편집하고 인증 흐름에서 보안 원격 암호(SRP)로 로그인: ALLOW_USER_SRP_AUTH를 선택합니다.

사용자 이름 및 SRP를 사용하여 선택 기반 인증을 활성화하려면 앱 클라이언트를 편집하고 선택 기반 로그인: ALLOW_USER_AUTH를 선택합니다.



선택 기반 인증 흐름에서 SRP 인증을 사용할 수 있는지 확인하려면 로그인 메뉴로 이동하여 선택 기반 로그인 옵션 아래의 섹션을 검토하세요. 사용 가능한 선택 항목에 암호가 표시되는 경우 SRP 인증으로 로그인할 수 있습니다. 암호 옵션에는 일반 텍스트 및 SRP 사용자 이름-암호 인증 변형이 포함됩니다.



CreateUserPoolClient 또는 UpdateUserPoolClient 요청에서 원하는 username-and-password 인증 옵션으로 `ExplicitAuthFlows`를 구성합니다.

```
"ExplicitAuthFlows": [
   "ALLOW_USER_SRP_AUTH",
   "ALLOW_USER_AUTH"
```

```
]
```

CreateUserPool 또는 UpdateUserPool 요청에서 지원하려는 선택 기반 인증 흐름으로 Policies를 구성합니다. AllowedFirstAuthFactors의 PASSWORD 값에는 일반 텍스트 암호 및 SRP 인증 흐름 옵션이 모두 포함됩니다.

```
"Policies": {
   "SignInPolicy": {
      "AllowedFirstAuthFactors": [
         "PASSWORD",
         "EMAIL_OTP",
         "WEB_AUTHN"
      ]
   }
}
```

Choice-based sign-in with SRP

SRP를 사용하여 사용자 이름 암호 인증으로 애플리케이션에 로그인하려면 다음과 같이 AdminInitiateAuth 또는 InitiateAuth 요청의 본문을 구성합니다. 현재 사용자가 사용자 이름 암호 인증을 받을 수 있는 경우 이 로그인 요청이 성공하거나 다음 문제를 계속 진행합니다. 그렇지 않으면 사용 가능한 기본 인증 문제 목록으로 응답합니다. 이 파라미터 세트는 로그인에 필요한 최소 파라미터입니다. 추가 파라미터를 사용할 수 있습니다.

```
{
   "AuthFlow": "USER_AUTH",
   "AuthParameters": {
      "USERNAME" : "testuser",
      "PREFERRED_CHALLENGE" : "PASSWORD_SRP",
      "SRP_A" : "[g^a % N]"
   },
   "ClientId": "1example23456789"
}
```

PREFERRED_CHALLENGE 값을 생략하고 사용자의 적격 로그인 요인 목록이 포함된 응답을 받을 수도 있습니다.

```
{
   "AuthFlow": "USER_AUTH",
   "AuthParameters": {
```

```
      "USERNAME" : "testuser"
   },
   "ClientId": "1example23456789"
}
```

선호하는 문제를 제출하지 않았거나 제출된 사용자가 선호하는 문제에 적합하지 않은 경우 Amazon Cognito는 `AvailableChallenges`에서 옵션 목록을 반환합니다. `AvailableChallenges`에 `PASSWORD_SRP`의 `ChallengeName`이 포함된 경우 다음 형식으로 RespondToAuthChallenge 또는 AdminRespondToAuthChallenge 문제 응답을 사용하여 인증을 계속할 수 있습니다. 문제 응답을 초기 로그인 요청에 대한 API 응답과 연결하는 `Session` 파라미터를 전달해야 합니다. 이 파라미터 세트는 로그인에 필요한 최소 파라미터입니다. 추가 파라미터를 사용할 수 있습니다.

```
{
   "ChallengeName": "PASSWORD_SRP",
   "ChallengeResponses": {
      "USERNAME" : "testuser",
      "SRP_A" : "[g^a % N]"
   },
   "ClientId": "1example23456789",
   "Session": "[Session ID from the previous response"
}
```

Amazon Cognito는 적합한 우선 문제 요청 및 `PASSWORD_SRP` 문제 응답에 `PASSWORD_VERIFIER` 문제로 응답합니다. 클라이언트는 SRP 계산을 완료하고 RespondToAuthChallenge 또는 AdminRespondToAuthChallenge 요청의 문제에 응답해야 합니다.

```
{
   "ChallengeName": "PASSWORD_VERIFIER",
   "ChallengeResponses": {
      "PASSWORD_CLAIM_SIGNATURE" : "string",
      "PASSWORD_CLAIM_SECRET_BLOCK" : "string",
      "TIMESTAMP" : "string"
   },
   "ClientId": "1example23456789",
   "Session": "[Session ID from the previous response]"
}
```

성공적인 `PASSWORD_VERIFIER` 문제 응답에서 Amazon Cognito는 토큰 또는 다중 인증(MFA)과 같은 다른 필수 문제를 발급합니다.

Client-based sign-in with SRP

SRP 인증은 서버 측 인증보다 클라이언트 측 인증에 더 일반적입니다. 그러나 InitiateAuth 및 AdminInitiateAuth에서 SRP 인증을 사용할 수 있습니다. 애플리케이션에 사용자를 로그인하려면 다음과 같이 `InitiateAuth` 또는 `AdminInitiateAuth` 요청의 본문을 구성합니다. 이 파라미터 세트는 로그인에 필요한 최소 파라미터입니다. 추가 파라미터를 사용할 수 있습니다.

클라이언트는 비밀 난수 a의 출력으로 증가된 생성기 모듈로 N g에서 `SRP_A`를 생성합니다.

```
{
   "AuthFlow": "USER_SRP_AUTH",
   "AuthParameters": {
      "USERNAME" : "testuser",
      "SRP_A" : "[g^a % N]"
   },
   "ClientId": "1example23456789"
}
```

Amazon Cognito가 `PASSWORD_VERIFIER` 문제로 응답합니다. 클라이언트는 SRP 계산을 완료하고 RespondToAuthChallenge 또는 AdminRespondToAuthChallenge 요청의 문제에 응답해야 합니다.

```
{
   "ChallengeName": "PASSWORD_VERIFIER",
   "ChallengeResponses": {
      "PASSWORD_CLAIM_SIGNATURE" : "string",
      "PASSWORD_CLAIM_SECRET_BLOCK" : "string",
      "TIMESTAMP" : "string"
   },
   "ClientId": "1example23456789",
   "Session": "[Session ID from the previous response]"
}
```

성공적인 `PASSWORD_VERIFIER` 문제 응답에서 Amazon Cognito는 토큰 또는 다중 인증(MFA)과 같은 다른 필수 문제를 발급합니다.

일회용 암호를 통한 암호 없는 로그인

암호를 분실하거나 도난당할 수 있습니다. 사용자가 확인된 이메일 주소, 전화번호 또는 인증 앱에 액세스할 수 있는지만 확인할 수 있습니다. 이에 대한 해결책은 암호 없는 로그인입니다. 애플리케이션은

사용자에게 사용자 이름, 이메일 주소 또는 전화번호를 입력하라는 메시지를 표시할 수 있습니다. 그런 다음 Amazon Cognito는 확인해야 하는 일회용 암호(OTP)를 생성합니다. 성공한 코드는 인증을 완료합니다.

일회용 암호(OTP) 인증 흐름은 사용자 풀의 필수 멀티 팩터 인증(MFA)과 호환되지 않습니다. `FactorConfiguration`를 로 설정하면 사용자 확인을 통한 패스키 인증이 MFA 요구 사항을 충족할 수 있습니다`MULTI_FACTOR_WITH_USER_VERIFICATION`. 사용자 풀에서 MFA가 선택 사항인 경우 MFA를 활성화한 사용자는 OTP 첫 번째 요소로 로그인할 수 없습니다. MFA 선택 사용자 풀에서 MFA 기본 설정이 없는 사용자는 암호 없는 요인으로 로그인할 수 있습니다. 자세한 내용은 사용자 풀 MFA에 대해 알아야 할 사항 섹션을 참조하세요.

사용자가 암호 없는 인증의 일부로 SMS 또는 이메일 메시지로 받은 코드를 올바르게 입력하면 사용자 인증 외에도 사용자 풀은 사용자의 검증되지 않은 이메일 주소 또는 전화번호 속성을 검증됨으로 표시합니다. 이메일 주소 또는 전화번호를 자동으로 확인하도록 사용자 풀을 구성했는지 여부에 관계없이 사용자 상태도 `UNCONFIRMED`에서 `CONFIRMED`로 변경되었습니다.

암호 없는 로그인을 사용하는 새로운 옵션

사용자 풀에서 암호 없는 인증을 활성화하면 일부 사용자 흐름의 작동 방식이 변경됩니다.

1. 사용자는 암호 없이 가입하고 로그인할 때 암호 없는 요소를 선택할 수 있습니다. 관리자로서 암호 없이 사용자를 생성할 수도 있습니다.
2. CSV 파일로 가져오는 사용자는 암호 없는 팩터로 즉시 로그인할 수 있습니다. 로그인하기 전에 암호를 설정할 필요는 없습니다.
3. 암호가 없는 사용자는 `PreviousPassword` 파라미터 없이 ChangePassword API 요청을 제출할 수 있습니다.

OTP 사용 자동 로그인

이메일 또는 SMS 메시지 OTP를 사용하여 사용자 계정을 가입하고 확인하는 사용자는 확인 메시지와 일치하는 암호 없는 팩터로 자동으로 로그인할 수 있습니다. 관리형 로그인 UI에서 계정을 확인하고 확인 코드 전송 방법을 사용하여 OTP에 로그인할 수 있는 사용자는 확인 코드를 제공한 후 자동으로 첫 번째 로그인을 진행합니다. AWS SDK를 사용하여 사용자 지정 빌드 애플리케이션에서 다음 파라미터를 InitiateAuth 또는 AdminInitiateAuth 작업에 전달합니다.

- ConfirmSignUp API 응답의 `Session` 파라미터를 `Session` 요청 파라미터로 사용합니다.
- `USER_AUTH`의 AuthFlow입니다.

EMAIL_OTP 또는 SMS_OTP의 PREFERRED_CHALLENGE를 전달할 수 있지만 필수는 아닙니다. Session 파라미터는 인증 증명을 제공하며 Amazon Cognito는 유효한 세션 코드를 전달할 때 AuthParameters를 무시합니다.

로그인 작업은 인증 성공, AuthenticationResult를 나타내는 응답을 반환하며, 다음 조건이 충족되면 추가 문제가 발생하지 않습니다.

- Session 코드가 유효하며 만료되지 않았습니다.
- 사용자는 OTP 인증 방법을 사용할 수 있습니다.

Activate passwordless sign-in

콘솔

암호 없는 로그인을 활성화하려면 하나 이상의 암호 없는 유형으로 기본 로그인을 허용하도록 사용자 풀을 구성한 다음 USER_AUTH 흐름을 허용하도록 앱 클라이언트를 구성합니다. Amazon Cognito 콘솔에서 사용자 풀 구성의 인증 아래에 있는 로그인 메뉴로 이동합니다. 선택 기반 로그인 옵션을 편집하고 이메일 메시지 일회용 암호 또는 SMS 메시지 일회용 암호를 선택합니다. 두 옵션을 모두 활성화할 수 있습니다. 변경 내용을 저장합니다.

앱 클라이언트 메뉴로 이동하여 앱 클라이언트를 선택하거나 새 클라이언트를 생성합니다. 편집을 선택하고 로그인 시 인증 유형 선택: ALLOW_USER_AUTH를 선택합니다.

API/SDK

사용자 풀 API에서 CreateUserPool 또는 UpdateUserPool 요청의 적절한 암호 없는 옵션으로 SignInPolicy를 구성합니다.

```
"SignInPolicy": {
    "AllowedFirstAuthFactors": [
        "EMAIL_OTP",
        "SMS_OTP"
    ]
}
```

CreateUserPoolClient 또는 UpdateUserPoolClient 요청에서 필요한 옵션을 사용하여 앱 클라이언트 ExplicitAuthFlows를 구성합니다.

```
"ExplicitAuthFlows": [
   "ALLOW_USER_AUTH"
```

```
]
```

Sign in with passwordless

암호 없는 로그인에는 InitiateAuth 및 AdminInitiateAuth에서 지정할 수 있는 클라이언트 기반 AuthFlow가 없습니다. OTP 인증은 기본 로그인 옵션을 요청하거나 사용자의 AvailableChallenges에서 암호 없는 옵션을 선택할 수 있는 USER_AUTH의 선택 기반 AuthFlow에서만 사용할 수 있습니다. 애플리케이션에 사용자를 로그인하려면 다음과 같이 InitiateAuth 또는 AdminInitiateAuth 요청의 본문을 구성합니다. 이 파라미터 세트는 로그인에 필요한 최소 파라미터입니다. 추가 파라미터를 사용할 수 있습니다.

이 예제에서는 사용자가 어떤 방식으로 로그인하고 싶은지 알 수 없습니다. PREFERRED_CHALLENGE 파라미터를 추가하고 사용자가 선호하는 문제를 사용할 수 있는 경우 Amazon Cognito는 해당 문제에 응답합니다.

```
{
   "AuthFlow": "USER_AUTH",
   "AuthParameters": {
      "USERNAME" : "testuser"
   },
   "ClientId": "1example23456789"
}
```

대신이 예제에서 "PREFERRED_CHALLENGE": "EMAIL_OTP" 또는 "PREFERRED_CHALLENGE": "SMS_OTP"를 AuthParameters에 추가할 수 있습니다. 사용자가 원하는 방법을 사용할 수 있는 경우 사용자 풀은 사용자의 이메일 주소 또는 전화번호로 코드를 즉시 전송하고 "ChallengeName": "EMAIL_OTP" 또는 "ChallengeName": "SMS_OTP"를 반환합니다.

기본 문제를 지정하지 않으면 Amazon Cognito가 AvailableChallenges 파라미터로 응답합니다.

```
{
   "AvailableChallenges": [
      "EMAIL_OTP",
      "SMS_OTP",
      "PASSWORD"
    ],
   "Session": "[Session ID]"
}
```

이 사용자는 이메일 메시지 OTP, SMS 메시지 OTP 및 사용자 이름 암호를 사용하여 암호 없이 로그인할 수 있습니다. 애플리케이션은 사용자에게 선택하라는 메시지를 표시하거나 내부 로직을 기반으로 선택할 수 있습니다. 그런 다음 문제를 선택하는 RespondToAuthChallenge 또는 AdminRespondToAuthChallenge 요청을 진행합니다. 사용자가 이메일 메시지 OTP를 사용하여 암호 없는 인증을 완료하려고 한다고 가정해 보겠습니다.

```
{
   "ChallengeName": "SELECT_CHALLENGE",
   "ChallengeResponses": {
      "USERNAME" : "testuser",
      "ANSWER" : "EMAIL_OTP"
   },
   "ClientId": "1example23456789",
   "Session": "[Session ID from the previous response]"
}
```

Amazon Cognito는 EMAIL_OTP 문제로 응답하고 사용자의 확인된 이메일 주소로 코드를 보냅니다. 그러면 애플리케이션이 이 문제에 다시 응답해야 합니다.

EMAIL_OTP를 PREFERRED_CHALLENGE로 요청한 경우 이는 다음 문제 응답이기도 합니다.

```
{
   "ChallengeName": "EMAIL_OTP",
   "ChallengeResponses": {
      "USERNAME" : "testuser",
      "EMAIL_OTP_CODE" : "123456"
   },
   "ClientId": "1example23456789",
   "Session": "[Session ID from the previous response]"
}
```

WebAuthn 패스키를 사용한 암호 없는 로그인

패스키는 안전하며 사용자에게 상대적으로 낮은 수준의 작업 부담을 줍니다. 패스키 로그인은 사용자가 인증할 수 있는 외부 장치인 인증자를 사용합니다. 일반 암호는 사용자를 피싱, 암호 추측 및 자격 증명 도용과 같은 취약성에 노출시킵니다. 패스키를 사용하면 애플리케이션이 정보 시스템에 연결되거나 내장된 휴대폰 및 기타 디바이스의 고급 보안 조치를 활용할 수 있습니다. 일반적인 패스키 로그인 워크플로는 iOS 키체인 또는 Google Chrome 암호 관리자와 같이 암호 또는 자격 증명 관리자를 호출하는 디바이스 호출로 시작됩니다. 디바이스 내 자격 증명 관리자는 패스키를 선택하고 기존 자격 증명 또는 디바이스 잠금 해제 메커니즘으로 권한을 부여하라는 메시지를 표시합니다. 최신 휴대폰에는 얼

굴 스캐너, 지문 스캐너, 잠금 해제 패턴 및 기타 메커니즘이 있으며, 일부는 사용자가 알고 있는 것과 강력한 인증 원칙을 동시에 충족합니다. 생체 인식을 이용한 패스키 인증의 경우, 패스키는 사용자의 신체적 특성을 나타냅니다.

암호를 지문, 얼굴 또는 보안 키 인증으로 바꿀 수 있습니다. 패스키 또는 WebAuthn 인증입니다. 애플리케이션 개발자는 사용자가 암호로 처음 로그인한 후 생체 인식 디바이스를 등록하도록 허용하는 것이 일반적입니다. Amazon Cognito 사용자 풀을 사용하면 애플리케이션에서 사용자를 위해 이 로그인 옵션을 구성할 수 있습니다. 패스키 인증은 사용자 풀이 로 `FactorConfiguration` 설정된 경우 다중 인증(MFA) 요구 사항을 충족할 수 있습니다`MULTI_FACTOR_WITH_USER_VERIFICATION`. 이 구성에서 사용자 확인이 포함된 패스키 인증은 다중 인증으로 계산됩니다.

일회용 암호(OTP) 인증 흐름은 사용자 풀의 필수 멀티 팩터 인증(MFA)과 호환되지 않습니다. `FactorConfiguration`를 로 설정하면 사용자 확인을 통한 패스키 인증이 MFA 요구 사항을 충족할 수 있습니다`MULTI_FACTOR_WITH_USER_VERIFICATION`. 사용자 풀에서 MFA가 선택 사항인 경우 MFA를 활성화한 사용자는 OTP 첫 번째 요소로 로그인할 수 없습니다. MFA 선택 사용자 풀에서 MFA 기본 설정이 없는 사용자는 암호 없는 요인으로 로그인할 수 있습니다. 자세한 내용은 사용자 풀 MFA에 대해 알아야 할 사항 섹션을 참조하세요.

패스키란 무엇인가요?

패스키를 사용하면 복잡한 암호를 기억하거나 OTP를 입력할 필요가 없어 사용자 경험이 단순화됩니다. 패스키는 W3C(World Wide Web Consortium) 및 Fast Identity Online(FIDO) Alliance에서 초안을 작성한 WebAuthn 및 CTAP2 표준을 기반으로 합니다. 브라우저 및 플랫폼은 이러한 표준을 구현하고, 웹 또는 모바일 애플리케이션을 위한 APIs를 제공하여 패스키 등록 또는 인증 프로세스를 시작하고, 사용자가 패스키 인증자를 선택하고 상호 작용할 수 있도록 UI를 제공합니다.

사용자가 웹 사이트 또는 앱에 인증자를 등록하면 인증자는 퍼블릭-프라이빗 키 페어를 생성합니다. WebAuthn 브라우저 및 플랫폼은 웹 사이트 또는 앱의 애플리케이션 백엔드에 퍼블릭 키를 제출합니다. 인증자는 사용자 및 애플리케이션에 대한 프라이빗 키, 키 ID 및 메타데이터를 유지합니다. 사용자가 등록된 인증자로 등록된 애플리케이션에서 인증하려는 경우 애플리케이션은 임의의 문제를 생성합니다. 이 문제에 대한 응답은 해당 애플리케이션 및 사용자에 대한 인증자의 프라이빗 키와 관련 메타데이터로 생성된 문제의 디지털 서명입니다. 브라우저 또는 애플리케이션 플랫폼은 디지털 서명을 수신하여 애플리케이션 백엔드로 전달합니다. 그런 다음 애플리케이션은 저장된 퍼블릭 키로 서명을 검증합니다.

> **Note**
>
> 애플리케이션은 사용자가 인증자에게 제공하는 인증 보안 암호를 수신하지 않으며 프라이빗 키에 대한 정보도 수신하지 않습니다.

다음은 현재 출시된 인증자의 몇 가지 예와 기능입니다. 인증자는 이러한 범주 중 일부 또는 전부를 충족할 수 있습니다.

- 일부 인증자는 액세스 권한을 부여하기 전에 PIN, 얼굴 또는 지문을 사용한 생체 인식 입력 또는 암호와 같은 요소를 사용하여 사용자 확인을 수행하여 합법적인 사용자만 작업을 승인할 수 있도록 합니다. 다른 인증자는 사용자 확인 기능이 없으며, 일부 인증자는 애플리케이션에 필요하지 않은 경우 사용자 확인을 건너뛸 수 있습니다.
- 예를 들어 YubiKey 하드웨어 토큰과 같은 일부 인증자는 휴대가 가능합니다. USB, Bluetooth 또는 NFC 연결을 통해 디바이스와 통신합니다. 일부 인증자는 로컬이며 PC의 Windows Hello 또는 iPhone의 Face ID와 같은 플랫폼에 바인딩됩니다. 디바이스에 바인딩된 인증자는 모바일 디바이스처럼 충분히 작다면 사용자가 휴대할 수 있습니다. 때때로 사용자는 무선 통신을 통해 하드웨어 인증자를 다양한 플랫폼과 연결할 수 있습니다. 예를 들어 데스크톱 브라우저의 사용자는 QR 코드를 스캔할 때 스마트폰을 패스키 인증자로 사용할 수 있습니다.
- 일부 플랫폼 바인딩 패스키는 클라우드와 동기화되므로 여러 위치에서 사용할 수 있습니다. 예를 들어 iPhones Face ID 패스키는 패스키 메타데이터를 iCloud Keychain의 사용자 Apple 계정과 동기화합니다. 이러한 패스키는 사용자가 각 디바이스를 독립적으로 등록하도록 요구하는 대신 Apple 디바이스에서 원활한 인증을 부여합니다. 1Password, Dashlane, Bitwarden과 같은 소프트웨어 기반 인증 앱은 사용자가 앱을 설치한 모든 플랫폼에서 패스키를 동기화합니다.

WebAuthn 용어에서 웹 사이트와 앱은 당사자를 신뢰합니다. 각 패스키는 패스키 인증을 수락하는 웹 사이트 또는 앱을 나타내는 통합 식별자인 특정 신뢰 당사자 ID와 연결됩니다. 개발자는 적절한 인증 범위를 가지려면 신뢰 당사자 ID를 신중하게 선택해야 합니다. 일반적인 신뢰 당사자 ID는 웹 서버의 루트 도메인 이름입니다. 이 신뢰 당사자 ID 사양이 있는 패스키는 해당 도메인 및 하위 도메인에 대해 인증할 수 있습니다. 사용자가 액세스하려는 웹 사이트의 URL이 신뢰 당사자 ID와 일치하지 않는 경우 브라우저 및 플랫폼은 패스키 인증을 거부합니다. 마찬가지로 모바일 앱의 경우 애플리케이션이 신뢰 당사자 ID로 표시된 경로에서 사용할 수 있도록 하는 `.well-known` 연결 파일에 앱 경로가 있는 경우에만 패스키를 사용할 수 있습니다.

패스키를 검색할 수 있습니다. 사용자가 사용자 이름을 입력할 필요 없이 브라우저 또는 플랫폼에서 자동으로 인식하고 사용할 수 있습니다. 사용자가 패스키 인증을 지원하는 웹 사이트 또는 앱을 방문하면 브라우저 또는 플랫폼이 이미 알고 있는 패스키 목록에서 선택하거나 QR 코드를 스캔할 수 있습니다.

Amazon Cognito는 패스키 인증을 어떻게 구현하나요?

패스키는 Lite를 제외한 모든 기능 플랜에서 사용할 수 있는 옵트인 기능입니다. 선택 기반 인증 흐름에서만 사용할 수 있습니다. 관리형 로그인을 통해 Amazon Cognito는 패스키 인증의 로직을 처리합니다.

AWS SDK의 Amazon Cognito 사용자 풀 API를 사용하여 애플리케이션 백엔드에서 패스키 인증을 수행할 수도 있습니다.

Amazon Cognito는 두 개의 비대칭 암호화 알고리즘 ES256(-7) 및 RS256(-257) 중 하나를 사용하여 생성된 패스키를 인식합니다. 대부분의 인증자는 두 알고리즘을 모두 지원합니다. 기본적으로 사용자는 하드웨어 토큰, 모바일 스마트폰, 소프트웨어 인증 앱 등 모든 유형의 인증자를 설정할 수 있습니다. Amazon Cognito는 현재 증명 시행을 지원하지 않습니다.

사용자 풀에서 사용자 확인을 기본 설정 또는 필수로 구성할 수 있습니다. 이 설정은 값을 제공하지 않는 API 요청에서 기본 설정으로 기본 설정되며 Amazon Cognito 콘솔에서 기본 설정이 기본적으로 선택됩니다. 사용자 확인을 기본 설정으로 설정하면 사용자는 사용자 확인 기능이 없는 인증자를 설정할 수 있으며 사용자 확인 없이 등록 및 인증 작업이 성공할 수 있습니다. 패스키 등록 및 인증에서 사용자 확인을 요구하려면 이 설정을 필수로 변경합니다.

패스키 구성에서 설정한 신뢰 당사자(RP) ID는 중요한 결정입니다. 달리 지정하지 않고 도메인 브랜딩 버전이 관리형 로그인인 경우 사용자 풀은 기본적으로 사용자 지정 도메인의 이름을 RP ID로 예상합니다. 사용자 지정 도메인이 없고 달리 지정하지 않으면 사용자 풀은 기본적으로 접두사 도메인의 RP ID로 설정됩니다. 퍼블릭 접미사 목록(PSL)에 없는 도메인 이름으로 RP ID를 구성할 수도 있습니다. RP ID 항목은 관리형 로그인 및 SDK 인증의 패스키 등록 및 인증에 적용됩니다. 패스키는 Amazon Cognito가 RP ID가 도메인인 `.well-known` 연결 파일을 찾을 수 있는 모바일 애플리케이션에서만 작동합니다. 웹 사이트 또는 앱을 공개적으로 사용할 수 있기 전에 신뢰 당사자 ID의 값을 결정하고 설정하는 것이 가장 좋습니다. RP ID를 변경하는 경우 사용자는 새 RP ID로 다시 등록해야 합니다.

각 사용자는 최대 20개의 패스키를 등록할 수 있습니다. 사용자 풀에 한 번 이상 로그인한 후에만 패스키를 등록할 수 있습니다. 관리형 로그인은 패스키 등록에서 상당한 노력을 제거합니다. 사용자 풀 및 앱 클라이언트에 대해 패스키 인증을 활성화하면 관리형 로그인 도메인이 있는 사용자 풀은 최종 사용자가 새 사용자 계정에 가입한 후 패스키를 등록하도록 알립니다. 또한 언제든지 사용자의 브라우저를 호출하여 패스키 등록을 위한 관리형 로그인 페이지로 보낼 수 있습니다. Amazon Cognito가 패스키 인증을 시작하려면 사용자가 사용자 이름을 제공해야 합니다. 관리형 로그인은 이를 자동으로 처리합니다. 로그인 페이지에는 사용자 이름을 묻는 메시지가 표시되고, 사용자가 하나 이상의 패스키를 등록했는지 확인한 다음 패스키 로그인을 묻는 메시지가 표시됩니다. 마찬가지로 SDK 기반 애플리케이션은 사용자 이름을 입력하라는 메시지를 표시하고 인증 요청에 제공해야 합니다.

패스키를 사용하여 사용자 풀 인증을 설정하고 사용자 지정 도메인과 접두사 도메인이 있는 경우 RP ID는 기본적으로 사용자 지정 도메인의 정규화된 도메인 이름(FQDN)으로 설정됩니다. Amazon Cognito 콘솔에서 접두사 도메인을 RP ID로 설정하려면 사용자 지정 도메인을 삭제하거나 접두사 도메인의 FQDN을 타사 도메인으로 입력합니다.

Activate passkey sign-in

콘솔

패스키로 로그인을 활성화하려면 하나 이상의 암호 없는 유형으로 기본 로그인을 허용하도록 사용자 풀을 구성한 다음 USER_AUTH 흐름을 허용하도록 앱 클라이언트를 구성합니다. Amazon Cognito 콘솔에서 사용자 풀 구성의 인증 아래에 있는 로그인 메뉴로 이동합니다. 선택 기반 로그인에 대한 옵션을 편집하고 사용 가능한 선택 항목 목록에 패스키를 추가합니다.

인증 방법 메뉴로 이동하여 패스키를 편집합니다.

- 사용자 확인은 사용자 풀에 현재 사용자에게 패스키 권한이 있는지 추가로 확인하는 패스키 디바이스가 필요한지 여부를 설정하는 것입니다. 사용자가 사용자 검증을 통해 디바이스를 구성하도록 권장하지만 필수는 아닌 경우, 선호 사항을 선택합니다. 사용자 확인 기능이 있는 디바이스만 지원하려면 필수를 선택합니다. 자세한 내용은 w3.org 사용자 확인을 참조하세요.
- 신뢰 당사자 ID의 도메인은 애플리케이션이 사용자의 패스키 등록 요청에 전달하는 식별자입니다. 사용자 패스키 발급자와의 신뢰 관계 대상을 설정합니다. 신뢰 당사자 ID는 다음 중 하나일 수 있습니다.

  Cognito 도메인

  사용자 풀의 Amazon Cognito 접두사 도메인.

  사용자 지정 도메인

  사용자 풀의 사용자 지정 도메인.

  서드파티 도메인

  사용자 풀 관리형 로그인 페이지를 사용하지 않는 애플리케이션의 도메인입니다. 이 설정은 일반적으로 도메인이 없고 백엔드에서 AWS SDK 및 사용자 풀 API로 인증을 수행하는 사용자 풀과 연결됩니다.

앱 클라이언트 메뉴로 이동하여 앱 클라이언트를 선택하거나 새 클라이언트를 생성합니다. 편집을 선택하고 인증 흐름에서 로그인 시 인증 유형 선택: ALLOW_USER_AUTH를 선택합니다.

API/SDK

사용자 풀 API에서 CreateUserPool 또는 UpdateUserPool 요청의 적절한 패스키 옵션으로 SignInPolicy를 구성합니다. 패스키 인증 WEB_AUTHN 옵션에는 하나 이상의 다른 옵션이 포함되어야 합니다. 패스키 등록에는 기존 인증 세션이 필요합니다.

```
"SignInPolicy": {
    "AllowedFirstAuthFactors": [
        "PASSWORD",
        "WEB_AUTHN"
    ]
}
```

SetUserPoolMfaConfig 요청의 WebAuthnConfiguration 파라미터에서 사용자 확인 기본 설정과 RP ID를 구성합니다. 패스키 인증 결과의 대상인 RelyingPartyId는 사용자 풀 접두사 또는 사용자 지정 도메인이거나 사용자가 선택한 도메인일 수 있습니다.

```
"WebAuthnConfiguration": {
   "RelyingPartyId": "example.auth.us-east-1.amazoncognito.com",
   "UserVerification": "preferred",
   "FactorConfiguration": "SINGLE_FACTOR"
}
```

CreateUserPoolClient 또는 UpdateUserPoolClient 요청에서 필요한 옵션을 사용하여 앱 클라이언트 ExplicitAuthFlows를 구성합니다.

```
"ExplicitAuthFlows": [
   "ALLOW_USER_AUTH"
]
```

Register a passkey (managed login)

관리형 로그인은 패스키의 사용자 등록을 처리합니다. 사용자 풀에서 패스키 인증이 활성화되면 Amazon Cognito는 사용자에게 새 사용자 계정에 등록할 때 패스키를 설정하라는 메시지를 표시합니다.

Amazon Cognito는 사용자가 이미 가입하여 패스키를 설정하지 않았거나 관리자로 계정을 생성한 경우 사용자에게 패스키를 설정하라는 메시지를 표시하지 않습니다. 이 상태의 사용자는 패스키를 등록하기 전에 암호 또는 암호 없는 OTP와 같은 다른 요소로 로그인해야 합니다.

패스키를 등록하려면

1. 사용자를 로그인 페이지로 안내합니다.

```
https://auth.example.com/oauth2/authorize/?
client_id=1example23456789&response_type=code&scope=email+openid
+phone&redirect_uri=https%3A%2F%2Fwww.example.com
```

2. 사용자의 인증 결과를 처리합니다. 이 예제에서 Amazon Cognito는 애플리케이션이 토큰으로 교환하는 권한 부여 코드를 사용하여 www.example.com으로 리디렉션합니다.
3. 사용자를 등록-패스키 페이지로 안내합니다. 사용자는 로그인 세션을 유지하는 브라우저 쿠키를 갖게 됩니다. 패스키 URL은 client_id 및 redirect_uri 파라미터를 사용합니다. Amazon Cognito는 인증된 사용자만 이 페이지에 액세스할 수 있도록 허용합니다. 암호, 이메일 OTP 또는 SMS OTP로 사용자를 로그인한 후, 다음 패턴과 일치하는 URL을 호출합니다.

   이 요청에 response_type 및 scope와 같은 다른 권한 부여 엔드포인트 파라미터를 추가할 수도 있습니다.

```
https://auth.example.com/passkeys/add?
client_id=1example23456789&redirect_uri=https%3A%2F%2Fwww.example.com
```

Register a passkey (SDK)

PublicKeyCreationOptions 객체의 메타데이터에 패스키 자격 증명을 등록합니다. 로그인한 사용자의 자격 증명으로 이 객체를 생성하고 API 요청에서 패스키 발급자에게 제공할 수 있습니다. 발급자는 패스키 등록을 확인하는 RegistrationResponseJSON 객체를 반환합니다.

패스키 등록 프로세스를 시작하려면 기존 로그인 옵션으로 사용자를 로그인합니다. 현재 사용자의 액세스 토큰을 사용하여 토큰 인증 StartWebAuthnRegistration API 요청을 승인합니다. 다음은 GetWebAuthnRegistrationOptions API 요청 예시의 본문입니다.

```
{
   "AccessToken": "eyJra456defEXAMPLE"
}
```

사용자 풀의 응답에는 PublicKeyCreationOptions 객체가 포함됩니다. API 요청에 이 객체를 사용자의 발급자에게 제공합니다. 퍼블릭 키 및 신뢰 당사자 ID와 같은 정보를 제공합니다. 발급자가 RegistrationResponseJSON 객체로 응답합니다.

사용자의 액세스 토큰으로 다시 승인된 CompleteWebAuthnRegistration API 요청에 등록 응답을 제공합니다. 사용자 풀이 본문이 비어 있는 HTTP 200 응답으로 응답하면 사용자의 패스키가 등록됩니다.

Sign in with a passkey

암호 없는 로그인에는 InitiateAuth 및 AdminInitiateAuth에서 지정할 수 있는 AuthFlow가 없습니다. 대신 USER_AUTH의 AuthFlow를 선언하고 로그인 옵션을 요청하거나 사용자 풀의 응답에서 암호 없는 옵션을 선택해야 합니다. 애플리케이션에 사용자를 로그인하려면 다음과 같이 InitiateAuth 또는 AdminInitiateAuth 요청의 본문을 구성합니다. 이 파라미터 세트는 로그인에 필요한 최소 파라미터입니다. 추가 파라미터를 사용할 수 있습니다.

이 예제에서는 사용자가 패스키로 로그인하려는 것을 알고 있으며 PREFERRED_CHALLENGE 파라미터를 추가합니다.

```
{
   "AuthFlow": "USER_AUTH",
   "AuthParameters": {
      "USERNAME" : "testuser",
      "PREFERRED_CHALLENGE" : "WEB_AUTHN"
   },
   "ClientId": "1example23456789"
}
```

Amazon Cognito가 WEB_AUTHN 문제로 응답합니다. 애플리케이션이 이 문제에 응답해야 합니다. 사용자의 패스키 공급자로 로그인 요청을 시작합니다. AuthenticationResponseJSON 객체를 반환합니다.

```
{
   "ChallengeName": "WEB_AUTHN",
   "ChallengeResponses": {
      "USERNAME" : "testuser",
      "CREDENTIAL" : "{AuthenticationResponseJSON}"
   },
   "ClientId": "1example23456789",
   "Session": "[Session ID from the previous response]"
}
```

로그인 후 MFA

사용자 이름-암호 흐름으로 로그인을 완료하는 사용자에게 이메일 메시지, SMS 메시지 또는 코드 생성 애플리케이션의 일회용 암호로 추가 확인을 요청하도록 설정할 수 있습니다. MFA는 일회용 암호를 사용한 암호 없는 로그인과 구별됩니다. 그러나 사용자 확인이 포함된 패스키는 사용자 풀

MULTI_FACTOR_WITH_USER_VERIFICATION에서를 FactorConfiguration로 구성할 때 첫 번째 요소로 MFA 요구 사항을 충족할 수 있습니다WebAuthnConfiguration. 패스키는 암호 로그인의 두 번째 요소로 사용할 수 없습니다. 암호 기반 흐름의 경우 사용자 풀의 MFA는 사용자가 먼저 암호를 알고 있음을 보여준 다음 등록된 보조 팩터 디바이스에 액세스할 수 있음을 보여 주는 챌린지-응답 모델입니다.

구현 리소스

- 사용자 풀에 MFA 추가

새로 고침 토큰

자격 증명을 다시 입력하지 않고 사용자를 로그인 상태로 유지하려는 경우 새로 고침 토큰은 애플리케이션이 사용자의 세션을 유지해야 하는 도구입니다. 애플리케이션은 사용자 풀에 새로 고침 토큰을 제공하고 새 ID 및 액세스 토큰으로 교환할 수 있습니다. 토큰 새로 고침을 사용하면 로그인한 사용자가 여전히 활성 상태인지 확인하고, 업데이트된 속성 정보를 가져오고, 사용자 개입 없이 액세스 제어 권한을 업데이트할 수 있습니다.

구현 리소스

- 새로 고침 토큰

사용자 지정 인증

여기에 나열되지 않은 사용자에 대한 인증 방법을 구성할 수 있습니다. Lambda 트리거를 사용한 사용자 지정 인증으로 이 작업을 수행할 수 있습니다. 일련의 Lambda 함수에서 Amazon Cognito는 문제를 실행하고, 사용자가 답변해야 하는 질문을 하고, 답변의 정확성을 확인한 다음 다른 문제를 실행해야 하는지 여부를 결정합니다. 질문과 답변에는 보안 질문, CAPTCHA 서비스에 대한 요청, 외부 MFA 서비스 API에 대한 요청 또는 이 모든 것이 순서대로 포함될 수 있습니다.

구현 리소스

- 사용자 정의 인증 챌린지 Lambda 트리거

사용자 지정 인증 흐름

Amazon Cognito 사용자 풀에서는 사용자 지정 인증 흐름을 사용할 수 있게 할 수 있으므로 AWS Lambda 트리거를 사용하여 문제/응답 기반 인증 모델을 쉽게 생성할 수 있습니다.

사용자 지정 인증 흐름은 여러 요구 사항에 맞추어 사용자 지정 챌린지 및 응답 사이클을 허용하도록 고안되었습니다. 이 흐름은 사용할 인증 유형을 나타내고 초기 인증 파라미터를 제공하는 `InitiateAuth` API 작업을 호출하면서 시작됩니다. Amazon Cognito는 다음 정보 유형 중 하나를 사용하여 `InitiateAuth` 호출에 응답합니다.

- 세션 및 파라미터와 함께 사용자에 대한 챌린지
- 사용자가 인증에 실패할 경우 오류
- `InitiateAuth` 호출에서 사용자 로그인에 필요한 파라미터가 제공된 경우 ID, 액세스 및 새로 고침 토큰. (일반적으로 사용자나 앱은 먼저 챌린지에 답변해야 하지만 사용자 지정 코드에서 이를 결정해야 합니다.)

Amazon Cognito가 문제와 함께 `InitiateAuth` 호출에 응답하면 앱은 더 많은 입력을 수집하며 `RespondToAuthChallenge` 작업을 호출합니다. 이 호출은 문제 응답을 제공하고 세션에 다시 전달합니다. Amazon Cognito는 `InitiateAuth` 호출과 유사한 방식으로 `RespondToAuthChallenge` 호출에 응답합니다. 사용자가 로그인한 경우 Amazon Cognito에서 토큰을 제공하거나 사용자가 로그인하지 않은 경우 Amazon Cognito에서 다른 문제 또는 오류를 제공합니다. Amazon Cognito에서 다른 문제를 반환한 경우 사용자가 성공적으로 로그인하거나 오류가 반환될 때까지 시퀀스가 반복되고 앱에서 `RespondToAuthChallenge`를 호출합니다. `InitiateAuth` 및 `RespondToAuthChallenge` API 작업에 대한 자세한 내용은 API 설명서를 참조하세요.

사용자 지정 인증 흐름 및 챌린지

앱은 `CUSTOM_AUTH`를 `Authflow`로 지정해 `InitiateAuth`를 호출하여 사용자 지정 인증 흐름을 시작할 수 있습니다. 사용자 지정 인증 흐름을 사용하면 세 개의 Lambda 트리거에서 문제 및 응답의 확인을 제어합니다.

- `DefineAuthChallenge` Lambda 트리거는 이전 문제와 응답의 세션 배열을 입력으로 사용합니다. 그리고 나서 다음 문제 이름 및 사용자가 인증되었는지 그리고 토큰이 부여될 수 있는지를 나타내는 부울을 생성합니다. 이 Lambda 트리거는 문제를 통해 사용자의 경로를 제어하는 상태 시스템입니다.
- `CreateAuthChallenge` Lambda 트리거는 문제 이름을 입력으로 사용하고 문제 및 파라미터를 생성하여 응답을 평가합니다. `DefineAuthChallenge`에서 `CUSTOM_CHALLENGE`를 다음 문제로 반환하면 인증 흐름에서 `CreateAuthChallenge`를 호출합니다. 이 `CreateAuthChallenge` Lambda 트리거는 다음 문제 유형을 문제 메타데이터 파라미터로 전달합니다.
- `VerifyAuthChallengeResponse` Lambda 함수가 응답을 평가하고 부울을 반환하여 응답이 유효한지 여부를 나타냅니다.

사용자 지정 인증 흐름은 SRP 암호 확인 및 SMS를 통한 MFA와 같은 기본 제공 챌린지를 조합하여 사용할 수 있습니다. 또한 CAPTCHA 또는 보안 암호 질문과 같은 사용자 지정 챌린지를 사용할 수 있습니다.

사용자 지정 인증 흐름에서 SRP 암호 확인 사용

사용자 지정 인증 흐름에 SRP를 포함하려면 SRP를 시작해야 합니다.

- 앱은 사용자 지정 흐름에서 SRP 암호 확인을 시작하기 위해 `CUSTOM_AUTH`를 `Authflow`로 지정하여 `InitiateAuth`를 호출합니다. `AuthParameters` 맵에서는 앱으로부터의 요청에 `SRP_A:`(SRP A 값) 및 `CHALLENGE_NAME: SRP_A`가 포함됩니다.
- 이 `CUSTOM_AUTH` 흐름이 `challengeName: SRP_A` 및 `challengeResult: true`의 초기 세션에서 `DefineAuthChallenge` Lambda 트리거를 호출합니다. Lambda 함수는 `challengeName: PASSWORD_VERIFIER`, `issueTokens: false` 및 `failAuthentication: false`로 응답해야 합니다.
- 다음 앱은 `challengeName: PASSWORD_VERIFIER` 및 `challengeResponses` 맵의 SRP에 필요한 다른 파라미터와 함께 `RespondToAuthChallenge`를 호출해야 합니다.
- Amazon Cognito에서 암호를 확인하면 `RespondToAuthChallenge`에서 `challengeName: PASSWORD_VERIFIER` 및 `challengeResult: true`의 두 번째 세션과 함께 `DefineAuthChallenge` Lambda 트리거를 호출합니다. 이때 `DefineAuthChallenge` Lambda 트리거는 `challengeName: CUSTOM_CHALLENGE`로 응답하여 사용자 지정 문제를 시작할 수 있습니다.
- 사용자에 대해 MFA가 활성화된 경우 Amazon Cognito가 암호를 확인한 후 사용자에게 MFA를 사용하거나 설정하거나 로그인하라는 메시지가 표시됩니다.

> **Note**
>
> Amazon Cognito 호스팅 로그인 웹 페이지는 사용자 정의 인증 챌린지 Lambda 트리거를 활성화할 수 없습니다.

Lambda 트리거에 대한 자세한 내용은 Lambda 트리거를 사용하여 사용자 풀 워크플로 사용자 정의 섹션을 참조하세요.

사용자 마이그레이션 인증 흐름

사용자 마이그레이션 Lambda 트리거는 기존 사용자 관리 시스템에서 사용자 풀로의 마이그레이션을 도와줍니다. `USER_PASSWORD_AUTH` 인증 흐름을 선택하면 사용자 마이그레이션 과정에서 사용자가 암호를 재설정할 필요가 없습니다. 이 흐름은 인증 시 암호화된 SSL 연결을 통해 사용자의 암호를 서비스로 보냅니다.

모든 사용자를 마이그레이션하면 이 흐름을 보다 안전한 SRP 흐름으로 전환합니다. SRP 흐름은 네트워크를 통해 암호를 보내지 않습니다.

Lambda 트리거에 대해 자세히 알아보려면 Lambda 트리거를 사용하여 사용자 풀 워크플로 사용자 정의 섹션을 참조하세요.

Lambda 트리거를 사용한 사용자 마이그레이션에 대한 자세한 내용은 사용자 마이그레이션 Lambda 트리거를 사용하여 사용자 가져오기 섹션을 참조하세요.

API 및 SDK 인증을 위한 권한 부여 모델

사용자 풀 인증을 사용하여 애플리케이션 개발을 시작할 때는 애플리케이션 유형에 맞는 API 권한 부여 모델을 결정해야 합니다. 권한 부여 모델은 Amazon Cognito 사용자 풀 API 및 SDK 통합에서 인증 구성 요소를 사용하여 요청할 수 있는 권한을 제공하는 시스템입니다. Amazon Cognito에는 IAM 권한 부여, 퍼블릭 및 토큰 권한 부여의 세 가지 권한 부여 모델이 있습니다.

IAM 권한 부여 요청의 경우, 요청 `Authorization` 헤더에 있는 일련의 AWS IAM 자격 증명에 의한 서명을 통해 권한 부여가 이루어집니다. 서버 측 애플리케이션의 경우 이 방법은 IAM 권한 부여를 통해 인증 작업을 보호합니다. 퍼블릭(인증되지 않은) 인증 요청의 경우 권한 부여가 필요하지 않습니다. 이는 사용자에게 배포된 클라이언트 측 애플리케이션에 적합합니다. 일반적으로 퍼블릭 작업과 함께 구현되는 토큰 인증 작업의 경우 권한 부여는 요청의 `Authorization` 헤더에 포함된 세션 토큰 또는 액세스 토큰에서 비롯됩니다. Amazon Cognito 인증을 사용하려면 일반적으로 두 개 이상의 API 작업을 순서대로 구현해야 하며, 사용하는 API 작업은 애플리케이션의 특성에 따라 달라집니다. 애플리케이션이 사용자에게 배포되는 퍼블릭 클라이언트는 로그인 요청에 권한이 필요하지 않은 퍼블릭 작업을 사용합니다. 토큰 인증 작업은 퍼블릭 애플리케이션에서 사용자 세션을 계속합니다. 애플리케이션 로직이 원격 시스템에서 호스팅되는 서버 측 클라이언트는 로그인 요청에 대한 IAM 권한 부여를 통해 인증 작업을 보호합니다. 이어지는 API 작업 페어와 해당 SDK 메서드는 사용 가능한 권한 부여 모델에 매핑됩니다.

각 퍼블릭 인증 작업에는 서버 측에서 동등한 작업이 있는데, 예를 들어 UpdateUserAttributes 및 AdminUpdateUserAttributes가 있습니다. 클라이언트 측 작업은 사용자가 시작하고 확인이 필요하지만 서버 측 작업은 변경 사항이 사용자 풀 관리자가 커밋했고 변경 사항이 즉시 적용되었다고 가정합니다. 이 예제에서 Amazon Cognito는 사용자에게 확인 코드가 포함된 메시지를 전송하고 사용자의 액세스

토큰은 코드를 제출하는 VerifyUserAttribute 요청을 승인합니다. 서버 측 애플리케이션은 로그인에 사용할 때 이메일 주소 및 전화번호 값을 변경하는 데 특별한 고려 사항이 적용되지만 속성의 값을 즉시 설정할 수 있습니다.

API 인증을 비교하고 API 작업 및 권한 부여 모델의 전체 목록을 보려면 API, OIDC 및 관리형 로그인 페이지 인증 이해 섹션을 참조하세요.

Client-side (public) authentication

다음은 클라이언트 측 애플리케이션의 일반적인 요청 시퀀스입니다.

1. 퍼블릭 InitiateAuth 작업은 사용자 이름 및 암호와 같은 기본 자격 증명을 제출합니다.
2. 토큰 인증 RespondToAuthChallenge 작업은 `InitiateAuth` 응답의 세션 토큰과 MFA와 같은 문제에 대한 답변을 제출합니다. 세션 토큰 권한 부여는 아직 완료되지 않은 인증 주기의 일부인 요청을 나타냅니다.
3. 토큰 인증 ConfirmDevice 작업은 액세스 토큰을 제출하고 기억된 디바이스를 사용자 프로필에 추가하는 쓰기 작업을 수행합니다. 액세스 토큰 권한 부여는 인증을 완료한 후 사용자 셀프 서비스 작업에 대한 요청을 나타냅니다.

자세한 내용은 클라이언트 측 인증 옵션 및 API, OIDC 및 관리형 로그인 페이지 인증 이해 섹션을 참조하세요.

Server-side authentication

다음은 서버 측 작업의 일반적인 요청 시퀀스입니다. 각 요청에는 애플리케이션 서버에 발급된 IAM 시스템 자격 증명으로 서명된 AWS 서명 버전 4 권한 부여 헤더가 있습니다.

1. AdminInitiateAuth 작업은 사용자 이름 및 암호와 같은 기본 자격 증명을 제출합니다.
2. AdminRespondToAuthChallenge 작업은 MFA와 같은 문제에 대한 답변을 제출합니다.
3. AdminUpdateDeviceStatus 작업은 `AdminInitiateAuth` 응답의 디바이스 키를 기억된 대로 설정합니다.

자세한 내용은 서버 측 인증 옵션 및 API, OIDC 및 관리형 로그인 페이지 인증 이해 섹션을 참조하세요.

인증이 실패하거나 Amazon Cognito가 사용자에게 토큰이 발행할 때까지 사용자는 연속 문제에 대답하여 인증합니다. 모든 사용자 지정 인증 흐름을 지원하기 위해 서로 다른 문제가 포함된 프로세스에서 Amazon Cognito를 통해 이러한 단계를 반복할 수 있습니다.

주제

- 서버 측 인증 옵션
- 클라이언트 측 인증 옵션
- API, OIDC 및 관리형 로그인 페이지 인증 이해
- 권한 부여 모델별로 그룹화된 API 작업 목록

서버 측 인증 옵션

웹 애플리케이션 및 기타 서버 측 애플리케이션은 클라이언트가 브라우저 또는 SSH 세션과 같은 원격 디스플레이 애플리케이션에서 로드하는 원격 서버에 인증을 구현합니다. 서버 측 애플리케이션은 일반적으로 다음과 같은 특성을 갖습니다.

- Java, Ruby 또는 Node.js와 같은 언어로 서버에 설치된 애플리케이션에 내장되어 있습니다.
- 기밀 클라이언트라고 하는 클라이언트 보안 암호가 있을 수 있는 사용자 풀 앱 클라이언트에 연결합니다.
- 자격 AWS 증명에 액세스할 수 있습니다.
- 인증을 위해 관리형 로그인을 호출하거나 AWS SDK를 사용하여 사용자 풀 API에서 IAM 인증 작업을 사용합니다.
- 내부 고객에게 서비스를 제공하고 공개 고객에게 서비스를 제공할 수 있습니다.

사용자 풀 API를 사용하는 서버 측 작업은 암호, 일회용 암호 또는 패스키를 기본 로그인 요소로 사용할 수 있습니다. 서버 측 앱의 경우 사용자 풀 인증은 클라이언트 측 앱에 대한 인증과 유사합니다. 단, 다음은 예외입니다.

- 서버 측 앱은 AdminInitiateAuth API 요청을 합니다. 이 작업에는 `cognito-idp:AdminInitiateAuth` 및를 포함하는 권한이 있는 AWS 자격 증명이 필요합니다`cognito-idp:AdminRespondToAuthChallenge`. 작업은 필요한 문제 또는 인증 결과를 반환합니다.
- 애플리케이션이 문제를 수신하면 AdminRespondToAuthChallenge API 요청을 수행합니다. 또한 `AdminRespondToAuthChallenge` API 작업에는 AWS 자격 증명이 필요합니다.

AWS 자격 증명을 사용하여 Amazon Cognito API 요청에 서명하는 방법에 대한 자세한 내용은 AWS 일반 참조의 서명 버전 4 서명 프로세스를 참조하세요.

`AdminInitiateAuth` 응답 `ChallengeParameters`에 `USER_ID_FOR_SRP` 속성이 있으면 별칭(이메일 주소 또는 전화번호)이 아닌 사용자의 실제 사용자 이름이 이 속성에 포함됩니다.

AdminRespondToAuthChallenge를 호출할 때 ChallengeResponses에서 USERNAME 파라미터에 이 사용자 이름을 전달해야 합니다.

> **Note**
>
> 백엔드 관리자 구현은 관리자 인증 흐름을 사용하므로이 흐름은 기억된 디바이스를 지원하지 않습니다. 디바이스 추적이 설정되면 관리 인증이 성공하지만 액세스 토큰을 새로 고치는 호출은 실패합니다.

클라이언트 측 인증 옵션

모바일 앱 및 기타 클라이언트 측 애플리케이션 유형은 사용자의 디바이스에 설치되며 로컬에서 인증 및 사용자 인터페이스 로직을 수행합니다. 일반적으로 다음과 같은 특성이 있습니다.

- React 네이티브, Flutter, Swift와 같은 언어로 빌드되어 사용자 디바이스에 배포됩니다.
- 퍼블릭 클라이언트라는 클라이언트 보안 암호가 없는 사용자 풀 앱 클라이언트에 연결합니다.
- IAM 인증 API 요청을 승인하는 AWS 자격 증명에 액세스할 수 없습니다.
- 인증을 위해 관리형 로그인을 호출하거나 AWS SDK를 사용하여 사용자 풀 API에서 퍼블릭 및 토큰 인증 작업을 사용합니다.
- 퍼블릭 고객에게 서비스를 제공하고 누구나 가입하고 로그인할 수 있도록 허용합니다.

사용자 풀 API를 사용하는 클라이언트 측 작업은 암호, 일회용 암호 또는 패스키를 기본 로그인 요소로 사용할 수 있습니다. 다음 프로세스는 AWS Amplify 또는 AWS SDK로 생성한 사용자 클라이언트 측 앱을 대상으로 수행됩니다.

1. 사용자가 사용자 이름 및 암호를 앱에 입력합니다.
2. 앱이 사용자의 사용자 이름 및 Secure Remote Password(SRP) 세부 정보를 사용하여 InitiateAuth 작업을 호출합니다.

   이 API 작업은 인증 파라미터를 반환합니다.

   > **Note**
   >
   > 이 앱은 AWS SDK에 기본 제공된 Amazon Cognito SRP 기능을 사용하여 SRP 세부 정보를 생성합니다.

3. 앱이 RespondToAuthChallenge 작업을 호출합니다. 호출에 성공하면 Amazon Cognito에서 사용자 토큰을 반환하고 인증 흐름이 완료됩니다.

   Amazon Cognito에 다른 문제가 필요한 경우 RespondToAuthChallenge에 대한 호출이 토큰을 반환하지 않습니다. 대신 이러한 호출은 세션을 반환합니다.

4. RespondToAuthChallenge가 세션을 반환하는 경우 이번에는 앱이 세션 및 챌린지 응답(예: MFA 코드)과 함께 RespondToAuthChallenge를 다시 호출합니다.

API, OIDC 및 관리형 로그인 페이지 인증 이해

Amazon Cognito 사용자 풀은 여러 인증 기술의 조합입니다. 이들은 외부 ID 제공업체(idP)에 대한 의존 당사자입니다. 이들은 OpenID Connect(OIDC) SDK를 사용하여 인증을 구현하는 애플리케이션에 대한 IdP입니다. 이들은 OIDC 인증과 유사한 JSON 웹 토큰(JWT) 발급자로서 인증을 제공하지만, AWS SDK의 일부인 API 메서드를 사용합니다. 또한 애플리케이션에 대한 보안 진입점일 수도 있습니다.

사용자 풀에 가입하고, 로그인하고, 사용자 풀에서 사용자를 관리할 때는 두 가지 옵션을 사용할 수 있습니다.

1. 관리형 로그인 페이지와 클래식 호스팅 UI에는 관리형 로그인 사용자 대화형 엔드포인트와 IdP 및 신뢰 당사자 역할을 처리하는 페더레이션 엔드포인트가 포함됩니다. 이 엔드포인트는 사용자 풀에 사용할 도메인을 선택할 때 Amazon Cognito가 활성화하는 퍼블릭 웹 페이지 패키지를 구성합니다. 가입, 로그인, 암호 관리 및 다중 인증(MFA) 페이지를 비롯한 Amazon Cognito 사용자 풀의 인증 및 권한 부여 기능을 빠르게 시작하려면, 관리형 로그인의 내장 사용자 인터페이스를 사용하세요.

   다른 사용자 풀 엔드포인트는 서드 파티 ID 제공업체(idP)를 사용한 인증을 용이하게 합니다. 이들이 수행하는 서비스에는 다음이 포함됩니다.

   a. saml2/idpresponse 및 oauth2/idpresponse 등 IdP의 인증된 클레임을 위한 서비스 공급자 콜백 엔드포인트. Amazon Cognito가 앱과 IdP 사이의 중간 서비스 공급자(SP)인 경우 콜백 엔드포인트는 서비스를 나타냅니다.

   b. oauth2/userInfo 및 /.well-known/jwks.json 등 환경에 대한 정보를 제공하는 엔드포인트. 앱은 OIDC 또는 OAuth 2.0 개발자 라이브러리를 사용하여 토큰을 확인하거나 사용자 프로필 데이터를 검색할 때 이러한 엔드포인트를 사용합니다.

2. Amazon Cognito 사용자 풀 API는 사용자 지정 프런트 엔드에서 로그인 정보를 수집한 후 웹 또는 모바일 앱에서 사용자를 인증하는 데 사용할 수 있는 도구 세트입니다. 사용자 풀 API 인증은 다음 JSON 웹 토큰을 생성합니다.

   a. 사용자의 검증 가능한 속성 클레임이 포함된 자격 증명 토큰.

b. AWS 서비스 엔드포인트에 대한 토큰 인증된 API 요청을 생성할 권한을 사용자에게 부여하는 액세스 토큰.

> **Note**
>
> 기본적으로 사용자 풀 API 인증의 액세스 토큰에는 `aws.cognito.signin.user.admin` 범위만 포함되어 있습니다. 추가 범위가 포함된 액세스 토큰을 생성하려면(예: 서드 파티 API에 대한 요청 승인을 위해) 사용자 풀 엔드포인트를 통한 인증 중에 범위를 요청하거나 사전 토큰 생성 Lambda 트리거에서 사용자 지정 범위를 추가합니다. 액세스 토큰 사용자 지정은 AWS 청구서에 비용을 추가합니다.

c. 새 ID 및 액세스 토큰에 대한 요청을 승인하고 사용자 ID 및 액세스 제어 속성을 새로 고치는 새로 고침 토큰입니다.

일반적으로 사용자 풀 엔드포인트를 통해 로그인하는 페더레이션 사용자를 프로필이 사용자 풀에 로컬인 사용자와 연결할 수 있습니다. 로컬 사용자는 외부 IdP를 통한 페더레이션 없이 사용자 풀 디렉터리에만 존재합니다. AdminLinkProviderForUser API 요청에서 로컬 사용자에 페더레이션 자격 증명을 연결하면 해당 사용자가 사용자 풀 API로 로그인할 수 있습니다. 자세한 내용은 페더레이션 사용자를 기존 사용자 프로필에 연결 섹션을 참조하세요.

Amazon Cognito 사용자 풀 API는 두 가지 용도로 사용합니다.

1. 이 API는 Amazon Cognito 사용자 풀 리소스를 생성하고 구성합니다. 예를 들어 사용자 풀을 생성하고, AWS Lambda 트리거를 추가하고, 관리형 로그인 페이지를 호스팅하는 사용자 풀 도메인을 구성할 수 있습니다.
2. 로컬 및 연결된 사용자의 가입, 로그인 및 기타 사용자 작업을 수행합니다.

Amazon Cognito 사용자 풀 API를 사용한 예제 시나리오

1. 앱에서 생성한 “Create an account(계정 만들기)” 버튼을 사용자가 선택합니다. 사용자가 이메일 주소와 암호를 입력합니다.
2. 앱에서 SignUp API 요청을 보내고 사용자 풀에 새 사용자를 생성합니다.
3. 앱에서 사용자에게 이메일 확인 코드를 입력하라는 메시지를 표시합니다. 사용자가 이메일 메시지로 받은 코드를 입력합니다.
4. 앱은 ConfirmSignUp API 요청과 사용자의 확인 코드를 전송합니다.
5. 앱이 사용자 이름과 암호를 입력하라는 메시지를 표시하고, 사용자가 자신의 정보를 입력합니다.

6. 앱은 InitiateAuth API 요청을 보내고 ID 토큰, 액세스 토큰 및 새로 고침 토큰을 저장합니다. 앱은 OIDC 라이브러리를 호출하여 사용자의 토큰을 관리하고 해당 사용자의 영구 세션을 유지합니다.

Amazon Cognito 사용자 풀 API에서는 IdP를 통해 페더레이션하는 사용자를 로그인할 수 없습니다. 사용자 풀 엔드포인트를 통해 이러한 사용자를 인증해야 합니다. 관리형 로그인을 포함하는 사용자 풀 엔드포인트에 대한 자세한 내용은 사용자 풀 엔드포인트 및 관리형 로그인 참조 섹션을 참조하세요.

페더레이션 사용자는 관리형 로그인에서 시작하여 IdP를 선택하거나, 관리형 로그인을 건너뛰고 사용자를 IdP로 직접 보내 로그인하게 할 수 있습니다. 권한 부여 엔드포인트에 대한 API 요청에 IdP 파라미터가 포함되면, Amazon Cognito는 사용자를 자동으로 IdP 로그인 페이지로 리디렉션합니다.

관리형 로그인 페이지가 있는 예제 시나리오

1. 앱에서 생성한 “Create an account(계정 만들기)” 버튼을 사용자가 선택합니다.
2. 관리형 로그인은 개발자 보안 인증 정보를 등록한 소셜 ID 공급자 목록을 사용자에게 제시합니다. 사용자가 Apple을 선택합니다.
3. 앱에서 공급자 이름 `SignInWithApple`로 권한 부여 엔드포인트에 대한 요청을 시작합니다.
4. 사용자의 브라우저에서 Apple 인증 페이지가 열립니다. 사용자가 로그인하여 Amazon Cognito가 자신의 프로필 정보를 읽도록 승인합니다.
5. Amazon Cognito가 Apple 액세스 토큰을 확인하고 사용자의 Apple 프로필을 쿼리합니다.
6. 사용자가 앱에 Amazon Cognito 인증 코드를 제시합니다.
7. 애플리케이션의 OIDC 라이브러리는 Token 엔드포인트와 승인 코드를 교환하고 사용자 풀에서 발급한 ID 토큰, 액세스 토큰, 새로 고침 토큰을 저장합니다. 앱은 OIDC 라이브러리를 사용하여 사용자의 토큰을 관리하고 해당 사용자의 영구 세션을 유지합니다.

사용자 풀 API와 관리형 로그인 페이지는 이 안내서 전체에서 설명하는 다양한 시나리오를 지원합니다. 다음 섹션에서는 사용자 풀 API를 가입, 로그인, 리소스 관리 요구 사항을 지원하는 클래스로 구분하는 방법을 살펴봅니다.

권한 부여 모델별로 그룹화된 API 작업 목록

리소스 관리 인터페이스이자 사용자 대면 인증 및 권한 부여 인터페이스인 Amazon Cognito 사용자 풀 API는 운영에 동반되는 권한 부여 모델을 결합합니다. API 작업에 따라 IAM 자격 증명, 액세스 토큰, 세션 토큰, 클라이언트 암호 또는 이들의 조합을 통해 인증을 제공해야 할 수 있습니다. 대부분의 사용자 인증 및 권한 부여 작업의 경우, 요청의 인증 버전과 미인증 버전을 선택할 수 있습니다. 모바일 앱처럼

사용자에게 배포하는 앱의 경우 미인증 작업이 모범 사례입니다. 코드에 보안 암호를 포함하지 않아도 됩니다.

IAM 권한 부여 관리 작업 및 IAM 권한 부여 사용자 작업에 대한 IAM 정책에서만 권한을 할당할 수 있습니다.

IAM 권한 부여 관리 작업

IAM 권한 부여 관리 작업은 AWS Management Console에서처럼 사용자 풀 및 앱 클라이언트 구성을 수정하고 확인합니다.

예를 들어 UpdateUserPool API 요청에서 사용자 풀을 수정하려면 자격 AWS 증명과 IAM 권한을 제시하여 리소스를 업데이트해야 합니다.

AWS Command Line Interface (AWS CLI) 또는 AWS SDK에서 이러한 요청을 승인하려면 요청에 IAM 자격 증명을 추가하는 환경 변수 또는 클라이언트 구성으로 환경을 구성합니다. 자세한 내용은에서 자격 AWS 증명을 AWS 사용하여 액세스를 참조하세요AWS 일반 참조. Amazon Cognito 사용자 풀 API의 서비스 엔드포인트로 바로 요청을 보낼 수도 있습니다. 요청의 헤더에 포함된 AWS 자격 증명을 사용하여 이러한 요청을 승인하거나 서명해야 합니다. 자세한 내용은 AWS API 요청 서명을 참조하세요.

IAM 권한 부여 관리 작업

AddCustomAttributes

CreateGroup

CreateIdentityProvider

CreateResourceServer

CreateUserImportJob

CreateUserPool

CreateUserPoolClient

CreateUserPoolDomain

DeleteGroup

DeleteIdentityProvider

IAM 권한 부여 관리 작업

DeleteResourceServer

DeleteUserPool

DeleteUserPoolClient

DeleteUserPoolDomain

DescribeIdentityProvider

DescribeResourceServer

DescribeRiskConfiguration

DescribeUserImportJob

DescribeUserPool

DescribeUserPoolClient

DescribeUserPoolDomain

GetCSVHeader

GetGroup

GetIdentityProviderByIdentifier

GetSigningCertificate

GetUICustomization

GetUserPoolMfaConfig

ListGroups

ListIdentityProviders

ListResourceServers

ListTagsForResource

IAM 권한 부여 관리 작업

ListUserImportJobs

ListUserPoolClients

ListUserPools

ListUsers

ListUsersInGroup

SetRiskConfiguration

SetUICustomization

SetUserPoolMfaConfig

StartUserImportJob

StopUserImportJob

TagResource

UntagResource

UpdateGroup

UpdateIdentityProvider

UpdateResourceServer

UpdateUserPool

UpdateUserPoolClient

UpdateUserPoolDomain

IAM 권한 부여 사용자 작업

IAM 권한 부여 사용자 작업은 사용자의 가입, 로그인, 보안 인증 정보 관리, 수정 및 보기를 수행합니다.

예를 들어 웹 프런트 엔드를 지원하는 서버 측 애플리케이션 계층을 보유할 수 있습니다. 서버 측 앱은 Amazon Cognito 리소스에 대한 액세스 권한이 있는 신뢰하는 OAuth 기밀 클라이언트입니다. 앱에서 사용자를 등록하려면 서버에서 AdminCreateUser API 요청에 AWS 보안 인증 정보를 포함하면 됩니다. OAuth 클라이언트 유형에 대한 자세한 내용은 OAuth 2.0 인증 프레임워크의 클라이언트 유형을 참조하세요.

AWS CLI 또는 AWS SDK에서 이러한 요청을 승인하려면 요청에 IAM 자격 증명을 추가하는 환경 변수 또는 클라이언트 구성으로 서버 측 앱 환경을 구성합니다. 자세한 내용은에서 자격 AWS 증명을 AWS 사용하여 액세스를 참조하세요AWS 일반 참조. Amazon Cognito 사용자 풀 API의 서비스 엔드포인트로 바로 요청을 보낼 수도 있습니다. 요청의 헤더에 포함된 AWS 자격 증명을 사용하여 이러한 요청을 승인하거나 서명해야 합니다. 자세한 내용은 AWS API 요청 서명을 참조하세요.

앱 클라이언트에 클라이언트 보안 암호가 있는 경우 `AuthParameters`에 IAM 자격 증명과, 작업에 따라 `SecretHash` 파라미터 또는 `SECRET_HASH` 값을 입력해야 합니다. 자세한 내용은 보안 암호 해시 값 컴퓨팅 섹션을 참조하세요.

IAM 권한 부여 사용자 작업
AdminAddUserToGroup
AdminConfirmSignUp
AdminCreateUser
AdminDeleteSoftwareToken
AdminDeleteUser
AdminDeleteUserAttributes
AdminDisableProviderForUser
AdminDisableUser
AdminEnableUser
AdminForgetDevice
AdminGetDevice
AdminGetUser

IAM 권한 부여 사용자 작업

AdminGetUserAuthFactors

AdminInitiateAuth

AdminLinkProviderForUser

AdminListDevices

AdminListGroupsForUser

AdminListUserAuthEvents

AdminRemoveUserFromGroup

AdminResetUserPassword

AdminRespondToAuthChallenge

AdminSetUserMFAPreference

AdminSetUserPassword

AdminSetUserSettings

AdminUpdateAuthEventFeedback

AdminUpdateDeviceStatus

AdminUpdateUserAttributes

AdminUserGlobalSignOut

미인증 사용자 작업

미인증 사용자 작업은 사용자의 가입, 로그인 및 암호 재설정 시작 작업을 수행합니다. 인터넷에 있는 모든 사용자가 앱에 가입하고 로그인할 수 있게 하려면 미인증, 즉 공개 API 작업을 사용해야 합니다.

예를 들어 앱에 사용자를 등록하려면 보안 암호에 대한 어떠한 권한 액세스도 제공하지 않는 OAuth 공개 클라이언트를 배포하면 됩니다. 미인증 API 작업 SignUp을 이용해 이 사용자를 등록할 수 있습니다.

AWS SDK를 사용하여 개발한 퍼블릭 클라이언트에서 이러한 요청을 보내려면 자격 증명을 구성할 필요가 없습니다. 추가 인증 없이 Amazon Cognito 사용자 풀 API의 서비스 엔드포인트로 바로 요청을 보낼 수도 있습니다.

앱 클라이언트에 클라이언트 보안 암호가 있는 경우 작업에 따라 `AuthParameters`에 `SecretHash` 파라미터 또는 `SECRET_HASH` 값을 입력해야 합니다. 자세한 내용은 보안 암호 해시 값 컴퓨팅 섹션을 참조하세요.

미인증 사용자 작업

SignUp

ConfirmSignUp

ResendConfirmationCode

ForgotPassword

ConfirmForgotPassword

InitiateAuth

토큰 권한 부여 사용자 작업

토큰으로 권한이 부여된 사용자 작업은 사용자가 로그인하거나 로그인 프로세스를 시작한 후 사용자 로그아웃, 보안 인증 정보 관리, 수정 및 확인 작업을 수행합니다. 앱에서 보안 암호를 배포하지 않고 사용자 자신의 자격 증명으로 요청을 승인하고 싶다면, 토큰으로 권한이 부여된 API 작업을 사용해야 합니다. 사용자가 로그인을 완료한 경우 액세스 토큰을 사용하여 토큰으로 권한이 부여된 API 요청을 승인해야 합니다. 사용자가 로그인 프로세스를 진행 중이라면, Amazon Cognito가 이전 요청에 대한 응답으로 반환한 세션 토큰을 사용하여 토큰으로 권한이 부여된 API 요청을 승인해야 합니다.

예를 들어 공개 클라이언트에서는 사용자 자신의 프로필에 대한 쓰기 권한만 제한하는 방식으로 사용자 프로필을 업데이트해야 할 수 있습니다. 이 업데이트를 수행하기 위해 클라이언트는 UpdateUserAttributes API 요청에 사용자의 액세스 토큰을 포함할 수 있습니다.

AWS SDK를 사용하여 개발한 퍼블릭 클라이언트에서 이러한 요청을 보내려면 자격 증명을 구성할 필요가 없습니다. 요청에 `AccessToken` 또는 `Session` 파라미터를 포함합니다. Amazon Cognito 사용자 풀 API의 서비스 엔드포인트로 바로 요청을 보낼 수도 있습니다. 서비스 엔드포인트에 대한 요청을 승인하려면 요청의 POST 본문에 액세스 또는 세션 토큰을 포함합니다.

토큰으로 권한이 부여된 작업을 위한 API 요청에 서명하려면 액세스 토큰을 `Bearer <Base64-encoded access token>` 형식의 `Authorization` 헤더로 요청에 포함하세요.

토큰 권한 부여 사용자 작업	AccessToken	세션
RespondToAuthChallenge		✓
ChangePassword	✓	
GetUser	✓	
StartWebAuthnRegistration	✓	
CompleteWebAuthnRegistration	✓	
DeleteWebAuthnCredential	✓	
ListWebAuthnCredentials	✓	
UpdateUserAttributes	✓	
DeleteUserAttributes	✓	
DeleteUser	✓	
ConfirmDevice	✓	
ForgetDevice	✓	
GetDevice	✓	
ListDevices	✓	
UpdateDeviceStatus	✓	

토큰 권한 부여 사용자 작업	AccessToken	세션
GetUserAttributeVerificationCode	✓	
VerifyUserAttribute	✓	
SetUserSettings	✓	
SetUserMFAPreference	✓	
GlobalSignOut	✓	
UpdateAuthEventFeedback		✓
AssociateSoftwareToken	✓	✓
VerifySoftwareToken	✓	✓
RevokeToken[1]		
GetTokensFromRefreshToken[1]		

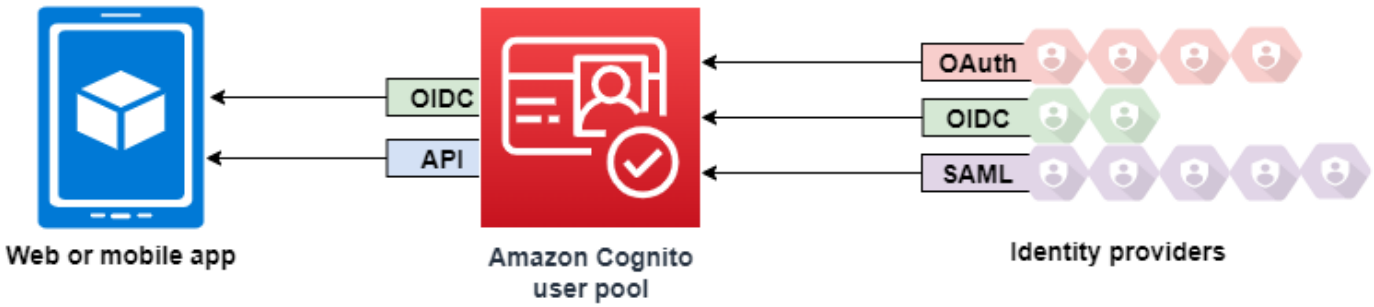
[1] `RevokeToken` 및 `GetTokensFromRefreshToken`은 새로 고침 토큰을 권한 부여 파라미터로 사용합니다. 새로 고침 토큰은 인증 토큰 및 대상 리소스 역할을 합니다.

타사 ID 제공업체를 통한 사용자 풀 로그인

앱 사용자는 사용자 풀을 통해 직접 로그인하거나 타사 ID 제공업체(idP)를 통해 연동할 수 있습니다. 사용자 풀에서는 Facebook, Google 및 Apple, 그리고 Amazon을 통한 소셜 로그인에서 반환된 토큰과 OpenID Connect(OIDC) 및 SAML IdP에서 반환된 토큰의 처리 작업을 관리합니다. Amazon Cognito는 기본 제공되는 호스팅 웹 UI를 사용하여 모든 IdP에서 인증된 사용자에 대한 토큰 처리 및 관리 기능을 제공합니다. 이렇게 하면 백엔드 시스템이 하나의 사용자 풀 토큰 세트에서 표준화할 수 있습니다.

Amazon Cognito 사용자 풀에서 페더레이션 로그인이 작동하는 방식

서드 파티(페더레이션)를 통한 로그인을 Amazon Cognito 사용자 풀에서 사용할 수 있습니다. 이 기능은 Amazon Cognito 자격 증명 풀(페더레이션 자격 증명)을 통한 페더레이션과 무관합니다.



Amazon Cognito는 사용자 디렉터리이자 OAuth 2.0 아이덴티티 제공업체(IdP)입니다. Amazon Cognito 디렉터리에 로컬 사용자를 로그인할 때 사용자 풀은 앱의 IdP입니다. 로컬 사용자는 외부 IdP를 통한 페더레이션 없이 사용자 풀 디렉터리에만 존재합니다.

Amazon Cognito를 소셜, SAML 또는 OpenID Connect(OIDC) IdP에 연결하면 사용자 풀은 여러 서비스 제공업체와 앱 간의 다리 역할을 합니다. IdP에 있어 Amazon Cognito는 서비스 제공업체(SP)입니다. IdP는 OIDC ID 토큰 또는 SAML 어설션을 Amazon Cognito에 전달합니다. Amazon Cognito는 토큰 또는 어설션에서 사용자에 대한 클레임을 읽고 이러한 클레임을 사용자 풀 디렉터리의 새 사용자 프로필에 매핑합니다.

그런 다음 Amazon Cognito는 페더레이션 사용자에 대한 사용자 프로필을 자체 디렉터리에 생성합니다. Amazon Cognito는 IdP, 그리고 OIDC와 소셜 아이덴티티 제공업체의 경우 IdP가 운영하는 퍼블릭 `userinfo` 엔드포인트의 클레임을 기반으로 사용자에게 속성을 추가합니다. 매핑된 IdP 속성이 변경되면 사용자 풀에서 사용자 속성이 변경됩니다. IdP와 별개로 속성을 더 추가할 수도 있습니다.

Amazon Cognito가 페더레이션 사용자 프로필을 생성한 후에는 해당 함수가 변경되고 앱의 IdP로 표시되어 SP가 됩니다. Amazon Cognito는 OIDC와 OAuth 2.0 IdP의 조합입니다. 액세스 토큰, ID 토큰 및 새로 고침 토큰을 생성합니다. 토큰에 대한 자세한 내용은 사용자 풀 JSON 웹 토큰(JWT) 이해 섹션을 참조하세요.

Amazon Cognito와 통합되는 앱을 설계하여 페더레이션 사용자든 로컬 사용자든 상관없이 사용자를 인증하고 권한을 부여해야 합니다.

Amazon Cognito에서 서비스 공급자로서 앱의 책임

토큰의 정보 확인 및 처리

대부분의 시나리오에서 Amazon Cognito는 인증된 사용자를 인증 코드와 함께 추가한 앱 URL로 리디렉션합니다. 앱에서는 액세스, ID 및 새로 고침 토큰용 코드를 교환합니다. 그런 다음 토큰의 유효성을 확인하고 토큰의 클레임을 기반으로 사용자에게 정보를 제공해야 합니다.

Amazon Cognito API 요청으로 인증 이벤트에 응답

앱은 Amazon Cognito 사용자 풀 API 및 인증 API 엔드포인트와 통합되어야 합니다. 인증 API는 사용자를 로그인 및 로그아웃하고 토큰을 관리합니다. 사용자 풀 API에는 사용자 풀, 사용자 및 인증 환경의 보안을 관리하는 다양한 작업이 있습니다. 앱은 Amazon Cognito로부터 응답을 받을 때 다음에 수행할 작업을 알아야 합니다.

Amazon Cognito 사용자 풀 타사 로그인에 대해 알아야 할 사항

- 사용자가 페더레이션 공급자로 로그인하도록 하려면 도메인을 선택해야 합니다. 이렇게 하면 관리형 로그인을 위한 페이지가 설정됩니다. 자세한 내용은 관리형 로그인에 자체 도메인 사용 섹션을 참조하세요.
- API 작업 InitiateAuth 및 AdminInitiateAuth를 통해 페더레이션 사용자로 로그인할 수 없습니다. Amazon Cognito의 기본 제공 인증 흐름을 사용하는 경우 페더레이션 사용자는 Login 엔드포인트 또는 로만 로그인할 수 있습니다권한 부여 엔드포인트. `CUSTOM_AUTH`를 사용하는 경우 트리거가 결과를 제어합니다. 서비스에서 사용자에게 토큰을 발급하는지 여부(연동 또는 비연동)에 대한 책임은 사용자에게 있습니다.
- 권한 부여 엔드포인트는 리디렉션 엔드포인트입니다. 요청에 `idp_identifier` 또는 `identity_provider` 파라미터를 입력하면 관리형 로그인을 우회하여 자동으로 IdP로 리디렉션됩니다. 그렇지 않으면 관리형 로그인 Login 엔드포인트로 리디렉션됩니다.
- 관리형 로그인이 세션을 페더레이션 IdP로 리디렉션하면 Amazon Cognito는 요청에 `user-agent` 헤더인 `Amazon/Cognito`를 포함합니다.
- Amazon Cognito는 고정 식별자와 IdP의 이름 조합에서 페더레이션 사용자 프로필의 `username` 속성을 추출합니다. 사용자 지정 요구 사항과 일치하는 사용자 이름을 생성하려면 `preferred_username` 속성에 대한 매핑을 생성합니다. 자세한 내용은 매핑에 대해 알아야 할 사항 섹션을 참조하세요.

  예시: `MyIDP_bob@example.com`

- Amazon Cognito는 사용자 풀에 추가하는 각 OIDC, SAMI 및 소셜 IdP에 대한 사용자 그룹을 생성합니다. 그룹 이름은 `us-east-1_EXAMPLE_MYSSO` 또는 `us-east-1_EXAMPLE_Google`과 같은 `[user pool ID]_[IdP name]` 형식입니다. 자동으로 생성된 각 고유 IdP 사용자 프로필이이 그룹에 자동으로 추가됩니다. 연결된 사용자는 이 그룹에 자동으로 추가되지 않지만 별도의 프로세스에서 그룹에 프로필을 추가할 수 있습니다.
- Amazon Cognito는 페더레이션 사용자의 ID 정보를 속성과 `identities`라는 ID 토큰의 클레임에 기록합니다. 이 클레임에는 사용자의 제공업체와 제공업체의 고유 ID가 포함됩니다. 사용자 프로필에서 `identities` 속성을 직접 변경할 수 없습니다. 페더레이션 사용자를 연결하는 방법에 대한 자세한 내용은 페더레이션 사용자를 기존 사용자 프로필에 연결 색션을 참조하세요.
- UpdateIdentityProvider API 요청에서 IdP를 업데이트하면, 변경 사항이 관리형 로그인에 표시되기까지 최대 1분이 걸립니다.
- Amazon Cognito는 Amazon Cognito와 사용자 IdP 간의 HTTP 리디렉션을 20개까지 지원합니다.
- 사용자가 관리형 로그인으로 로그인하면 브라우저는 로그인에 사용한 클라이언트 및 제공업체를 기록하는 암호화된 로그인 세션 쿠키를 저장합니다. 동일한 파라미터로 다시 로그인을 시도하면 관리형 로그인은 만료되지 않은 기존 세션을 재사용하며, 사용자는 보안 인증을 다시 제공하지 않고 인증합니다. 사용자는 다른 IdP로 다시 로그인할 경우(로컬 사용자 풀 로그인으로 전환하거나 로컬 사용자 풀 로그인에서 전환하는 경우 포함) 보안 인증을 제공하고 새 로그인 세션을 생성해야 합니다.

  사용자 풀 IdP는 어떤 앱 클라이언트에도 할당할 수 있으며, 사용자는 사용자의 앱 클라이언트에 할당된 IdP로만 로그인할 수 있습니다.

주제

- 사용자 풀의 자격 증명 공급자 구성
- 사용자 풀에서 소셜 ID 제공업체 사용
- 사용자 풀에서 SAML ID 제공업체 사용
- 사용자 풀에서 OIDC ID 제공업체 사용
- 프로필 및 토큰에 IdP 속성 매핑
- 페더레이션 사용자를 기존 사용자 프로필에 연결

사용자 풀의 자격 증명 공급자 구성

사용자 풀을 사용하면 다양한 외부 ID 제공업체(idP)를 통해 로그인을 구현할 수 있습니다. 가이드의 이 섹션에는 Amazon Cognito 콘솔에서 사용자 풀을 사용하여 이러한 ID 제공업체를 설정하는 지침이 나

와 있습니다. 또는 사용자 풀 API 및 AWS SDK를 사용하여 프로그래밍 방식으로 사용자 풀 자격 증명 공급자를 추가할 수 있습니다. 자세한 내용은 CreateIdentityProvider를 참조하세요.

지원되는 ID 제공업체 옵션에는 Facebook, Google, Amazon과 같은 소셜 제공업체와 OpenID Connect(OIDC) 및 SAML 2.0 제공업체가 포함됩니다. 시작하기 전에 IdP에 대한 관리 자격 증명을 설정합니다. 각 제공업체 유형에 대해 애플리케이션을 등록하고 필요한 자격 증명을 받은 다음 사용자 풀에서 제공업체 세부 정보를 구성해야 합니다. 그런 다음 사용자는 연결된 ID 제공업체의 기존 계정으로 애플리케이션에 가입하고 로그인할 수 있습니다.

인증 아래의 소셜 및 외부 제공업체 메뉴는 사용자 풀 IdP를 추가하고 업데이트합니다. 자세한 내용은 타사 ID 제공업체를 통한 사용자 풀 로그인 섹션을 참조하세요.

OIDC 또는 OAuth IdP의 경우 필드를 사용하여 IdP의 ACR 값을 Amazon Cognito 인증 수준에 매핑`AcrMapping`하여 페더레이션 사용자를 위한 단계적 인증을 지원할 수 있습니다. 자세한 내용은 페더레이션 사용자를 위한 단계적 인증 섹션을 참조하세요.

주제

- 소셜 IdP를 사용하여 사용자 로그인 설정
- OIDC IdP를 사용하여 사용자 로그인 설정
- SAML IdP를 사용하여 사용자 로그인 설정

소셜 IdP를 사용하여 사용자 로그인 설정

페더레이션을 사용하여 Amazon Cognito 사용자 풀을 Facebook, Google, Login with Amazon 등의 소셜 자격 증명 공급자와 통합할 수 있습니다.

소셜 자격 증명 공급자를 추가하려면 먼저 자격 증명 공급자에서 개발자 계정을 생성합니다. 개발자 계정이 생성되면 자격 증명 공급자에 앱을 등록합니다. 자격 증명 공급자가 앱에 대한 ID와 보안 암호를 생성하면 Amazon Cognito 사용자 풀에 이들 값을 구성합니다.

- Google 자격 증명 플랫폼
- 개발자용 Facebook
- Login with Amazon
- Apple로 로그인

사용자 로그인을 소셜 IdP와 통합하려면

1. Amazon Cognito 콘솔에 로그인합니다. 메시지가 표시되면 자격 AWS 증명을 입력합니다.

2. 탐색 창에서 [사용자 풀(User Pools)]을 선택한 다음 편집할 사용자 풀을 선택합니다.
3. 소셜 및 외부 제공업체 메뉴를 선택합니다.
4. 자격 증명 공급자 추가(Add an identity provider)를 선택하거나, 구성한 Facebook, Google, Amazon 또는 Apple 자격 증명 공급자를 선택하고 자격 증명 공급자 정보(Identity provider information)를 찾은 다음 편집(Edit)을 선택합니다. 소셜 자격 증명 공급자를 추가하는 방법에 대한 자세한 내용은 사용자 풀에서 소셜 ID 제공업체 사용 섹션을 참조하세요.
5. 선택한 IdP에 따라 다음 단계 중 하나를 완료하여 소셜 자격 증명 공급자의 정보를 입력합니다.

   Facebook, Google 및 Login with Amazon

   클라이언트 앱을 생성할 때 수신한 앱 ID 및 앱 보안 암호를 입력합니다.

   Apple로 로그인

   Apple에 제공한 서비스 ID와 앱 클라이언트를 생성할 때 수신한 팀 ID, 키 ID, 프라이빗 키를 입력합니다.
6. 권한 있는 범위(Authorized scopes)에서 사용자 풀 속성에 매핑할 소셜 자격 증명 공급자 범위의 이름을 입력합니다. 범위는 앱에서 액세스하려는 사용자 속성(예: 이름, 이메일)을 정의합니다. 범위를 입력하는 경우 선택한 IdP에 따라 다음 지침을 따르세요.

   - Facebook - 범위를 쉼표로 구분합니다. 예:

     ```
     public_profile, email
     ```

   - Google, Login with Amazon, Sign In with Apple - 범위를 공백으로 구분합니다. 예:
     - Google: `profile email openid`
     - Login with Amazon: `profile postal_code`
     - Apple로 로그인: `name email`

     > **Note**
     >
     > Sign in with Apple(콘솔)에서 확인란을 사용하여 범위를 선택합니다.
7. 변경 사항 저장을 선택합니다.
8. 앱 클라이언트 메뉴의 목록에서 앱 클라이언트를 선택한 다음 편집을 선택합니다. [자격 증명 공급자(Identity providers)]에서 앱 클라이언트에 새 소셜 자격 증명 공급자를 추가합니다.
9. 변경 사항 저장(Save changes)을 선택합니다.

소셜 IdP에 대한 자세한 내용은 사용자 풀에서 소셜 ID 제공업체 사용를 참조하세요.

OIDC IdP를 사용하여 사용자 로그인 설정

사용자 로그인을 Salesforce 또는 Ping Identity와 같은 OpenID Connect(OIDC) 자격 증명 공급자(IdP)와 통합할 수 있습니다.

사용자 풀에 OIDC 자격 증명 공급자 추가

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 자격 AWS 증명을 입력합니다.
2. 탐색 메뉴에서 [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. 소셜 및 외부 제공업체 메뉴를 선택하고 ID 제공업체 추가를 선택합니다.
5. OpenID Connect 자격 증명 공급자를 선택합니다.
6. 공급자 이름(Provider name)에 고유한 이름을 입력합니다.
7. 공급자로부터 수신한 클라이언트 ID를 클라이언트 ID(Client ID)에 입력합니다.
8. 공급자로부터 수신한 클라이언트 보안 암호를 클라이언트 보안 암호(Client secret)에 입력합니다.
9. 이 공급자의 권한 있는 범위(Authorized scopes)를 입력합니다. 범위는 애플리케이션이 공급자에게 요청하는 사용자 속성 그룹(예: `name` 및 `email`)을 정의합니다. OAuth 2.0 사양에 따라 범위를 공백으로 구분해야 합니다.

   사용자는 애플리케이션에 이러한 속성을 제공하는 것에 동의해야 합니다.
10. 속성 요청 메서드(Attribute request method)를 선택하여 Amazon Cognito가 공급자가 운영하는 userInfo 엔드포인트에서 사용자 세부 정보를 가져오는 데 사용하는 HTTP 메서드(GET 또는 POST)를 Amazon Cognito에 제공합니다.
11. 설정 메서드(Setup method)를 선택하여 발급자 URL을 통해 자동 채우기(Auto fill through issuer URL) 또는 수동 입력(Manual input)으로 OpenID Connect 엔드포인트를 검색합니다. 공급자에게 Amazon Cognito가 `authorization`, `token`, `userInfo` 및 `jwks_uri` 엔드포인트의 URL을 검색할 수 있는 퍼블릭 `.well-known/openid-configuration` 엔드포인트가 있는 경우 발급자 URL을 통해 자동 채우기를 사용합니다.
12. 발급자 URL이나 IdP의 `authorization`, `token`, `userInfo`, `jwks_uri` 엔드포인트 URL을 입력합니다.

> **Note**
>
> 검색 URL, 자동으로 채워진 URL 및 수동으로 입력한 URL에는 포트 번호 443과 80만 사용할 수 있습니다. OIDC 공급자가 비표준 TCP 포트를 사용하는 경우 사용자 로그인이 실패합니다.
>
> 발급자 URL은 `https://`로 시작해야 하며 / 문자로 시작하지 않아야 합니다. 예를 들어, Salesforce는 다음 URL을 사용합니다.
>
> `https://login.salesforce.com`
>
> 발급자 URL과 연결된 `openid-configuration` 문서는 `authorization_endpoint`, `token_endpoint`, `userinfo_endpoint` 및 `jwks_uri` 값에 대해 HTTPS URL을 제공해야 합니다. 마찬가지로 수동 입력(Manual input)을 선택하면 HTTPS URL만 입력할 수 있습니다.

13. OIDC 클레임 sub는 기본값으로 사용자 풀 속성 사용자 이름(Username)에 매핑됩니다. 이 외의 OIDC 클레임도 사용자 풀 속성으로 매핑할 수 있습니다. OIDC 클레임을 입력하고 드롭다운 목록에서 해당 사용자 풀 속성을 선택합니다. 예를 들어, 클레임 email은 주로 사용자 풀 속성 Email로 매핑됩니다.
14. 자격 증명 공급자의 추가 속성을 사용자 풀에 매핑합니다. 자세한 내용은 사용자 풀에 대한 자격 증명 공급자 속성 매핑 지정을 참조하세요.
15. 생성을 선택합니다.
16. 앱 클라이언트 메뉴의 목록에서 앱 클라이언트를 선택합니다. 앱 클라이언트에 새 SAML ID 제공업체를 추가하려면 로그인 페이지 탭으로 이동하여 관리형 로그인 페이지 구성에서 편집을 선택합니다.
17. 변경 사항 저장을 선택합니다.

OIDC IdP에 대한 자세한 내용은 사용자 풀에서 OIDC ID 제공업체 사용를 참조하세요.

SAML IdP를 사용하여 사용자 로그인 설정

Amazon Cognito 사용자 풀에 페더레이션을 사용하여 SAML IdP(자격 증명 공급자)와 통합할 수 있습니다. 파일을 업로드하거나 메타데이터 문서 엔드포인트 URL을 입력하여 메타데이터 문서를 제공해야 합니다. 타사 SAML IdP의 메타데이터 문서를 얻는 방법에 대한 자세한 내용은 타사 SAML ID 제공업체 구성을 참조하세요.

사용자 풀에 SAML 2.0 자격 증명 공급자를 구성하려면

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 자격 AWS 증명을 입력합니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. 소셜 및 외부 제공업체 메뉴를 선택하고 ID 제공업체 추가를 선택합니다.
5. [SAML] 자격 증명 공급자를 선택합니다.
6. 식별자(Identifiers)를 쉼표로 구분하여 입력합니다. 식별자는 Amazon Cognito가 사용자 로그인 이메일 주소를 확인한 다음 해당 도메인에 해당하는 공급자로 사용자를 안내하도록 합니다.
7. 사용자가 로그아웃할 때 Amazon Cognito에서 서명된 로그아웃 요청을 공급자에게 보내도록 하려는 경우 로그아웃 흐름 추가(Add sign-out flow)를 선택합니다. 관리형 로그인을 구성할 때 Amazon Cognito가 생성하는 `https://mydomain.auth.us-east-1.amazoncognito.com/saml2/logout` 엔드포인트에 로그아웃 응답을 보내도록 SAML 2.0 ID 제공업체를 구성합니다. `saml2/logout` 엔드포인트는 POST 바인딩을 사용합니다.

   > **Note**
   >
   > 이 옵션을 선택했고 SAML 자격 증명 공급자가 서명된 로그아웃 요청을 필요로 하는 경우 SAML IdP를 사용하여 Amazon Cognito에서 제공한 서명 인증서도 구성해야 합니다. SAML IdP는 서명된 로그아웃 요청을 처리하고 Amazon Cognito 세션에서 사용자를 로그아웃합니다.

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

   > **Note**
   >
   > 공급자에게 퍼블릭 엔드포인트가 있는 경우 파일을 업로드하지 않고 메타데이터 문서 URL을 입력하는 것이 좋습니다. URL을 사용하는 경우 Amazon Cognito는 메타데이터를 자동으로 새로 고칩니다. 일반적으로 메타데이터 새로 고침은 6시간마다 또는 메타데이터가 만료되기 전 중 더 빠른 시간에 발생합니다.

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

   예를 들어 [사용자 풀 속성(User pool attribute)] `email`을 선택한 경우 자격 증명 공급자의 SAML 어설션에 표시된 대로 SAML 속성 이름을 입력합니다. 자격 증명 공급자가 참조용으로 샘플 SAML 어설션을 제공할 수도 있습니다. `email`과 같은 간단한 이름을 사용하는 자격 증명 공급자도 있고, 다음과 같이 URL 포맷의 속성 이름을 사용하는 자격 증명 공급자도 있습니다.

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

10. 생성을 선택합니다.

> **Note**
>
> HTTPS 메타데이터 엔드포인트 URL을 사용하여 SAML IdP를 생성하는 동안 `InvalidParameterException`이 나타나는 경우 메타데이터 엔드포인트에 SSL이 올바르게 설정되어 있고 유효한 SSL 인증서가 연결되어 있는지 확인합니다. 이러한 예외의 한 예로는 "<메타데이터 엔드포인트>에서 메타데이터를 수신하는 동안 오류가 발생했습니다.(Error retrieving metadata from *<metadata endpoint>*)"가 있습니다.

서명 인증서를 추가하도록 SAML IdP를 설정하려면

- IdP가 서명된 로그아웃 요청을 확인하는 데 사용하는 퍼블릭 키가 포함된 인증서를 가져오려면 다음을 수행합니다.
  1. 사용자 풀의 소셜 및 외부 제공업체 메뉴로 이동합니다.
  2. SAML 제공업체를 선택합니다.
  3. 서명 인증서 보기를 선택합니다.

SAML IdP에 관한 자세한 내용은 사용자 풀에서 SAML ID 제공업체 사용 섹션을 참조하세요.

사용자 풀에서 소셜 ID 제공업체 사용

웹 및 모바일 앱 사용자는 Google, Facebook, Amazon, Apple과 같은 소셜 자격 증명 공급자(IdP)를 통해 로그인할 수 있습니다. 관리형 로그인을 활성화하면 Amazon Cognito가 각 소셜 IdP에 대한 로그인 버튼을 관리형 로그인 페이지에 추가합니다. Amazon Cognito는 공급자와의 OAuth 2.0 및 OpenID

Connect 교환을 처리하고 표준 사용자 풀 토큰 세트 하나를 앱에 발급하므로 백엔드 시스템은 사용자가 로그인한 방식에 관계없이 사용자 풀 토큰을 표준화할 수 있습니다. Amazon Cognito가 이러한 교환을 위해 생성하는 엔드포인트에 대한 자세한 내용은 Amazon Cognito 사용자 풀 인증 API 참조를 참조하세요.

각 공급자에는 자체 개발자 콘솔, 입력할 자체 값 및 자체 함정이 있습니다. 단계별 지침을 보려면 step-by-step

- Google을 소셜 자격 증명 공급자로 추가
- 소셜 자격 증명 공급자로 Sign in with Apple 추가
- Facebook을 소셜 자격 증명 공급자로 추가
- 소셜 자격 증명 공급자로 Login with Amazon 추가

> **Note**
>
> 서드 파티(페더레이션)를 통한 로그인을 Amazon Cognito 사용자 풀에서 사용할 수 있습니다. 이 기능은 Amazon Cognito 자격 증명 풀(페더레이션 자격 증명)을 통한 페더레이션과 무관합니다.

주제

- Google을 소셜 자격 증명 공급자로 추가
- 소셜 자격 증명 공급자로 Sign in with Apple 추가
- Facebook을 소셜 자격 증명 공급자로 추가
- 소셜 자격 증명 공급자로 Login with Amazon 추가

Google을 소셜 자격 증명 공급자로 추가

> **Note**
>
> 공급자 콘솔에 대해 마지막으로 확인된 날짜: 2026년 10월 2일. Google 콘솔 단계는 Google Auth 플랫폼(브랜딩, 대상, 데이터 액세스 및 클라이언트 섹션)을 따릅니다. Google 콘솔이 이러한 단계와 다른 경우 Google Auth 플랫폼 설명서를 참조하세요.

사용자가 Google로 로그인하도록 하려면 Google Auth 플랫폼에 OAuth 클라이언트를 등록한 다음 사용자 풀에 Google을 ID 제공업체(IdP)로 추가합니다. 먼저 관리형 로그인을 활성화해야 합니다.

Google Auth 플랫폼에 OAuth 클라이언트 등록

Google OAuth 클라이언트를 등록하려면

1. Google Cloud 콘솔에 로그인하고 프로젝트를 생성하거나 선택합니다.
2. 메뉴를 연 다음 Google Auth 플랫폼을 엽니다. 처음 사용하는 경우 개요 페이지에서 시작하기를 선택하고 앱 이름, 사용자 지원 이메일, 대상 유형(외부) 및 연락처 정보를 제공합니다.
3. 브랜딩을 선택합니다. 권한 있는 도메인에서 전체 URLs.
   - `amazoncognito.com`기본 Amazon Cognito 도메인의 경우 입니다.
   - 사용자 지정 도메인의 루트입니다. 예를 들어 `example.com`를 사용하는 경우 입니다.
4. 데이터 액세스를 선택한 다음 범위 추가 또는 제거를 선택합니다. `openid`, `.../auth/userinfo.email` 및 `.../auth/userinfo.profile`를 추가합니다.
5. 클라이언트를 선택한 다음 클라이언트 생성을 선택합니다. 애플리케이션 유형에서 웹 애플리케이션을 선택합니다.
6. 승인된 JavaScript 오리진에서 경로 없이 사용자 풀 도메인을 입력합니다.
   - 기본 Amazon Cognito 도메인: `https://<your-prefix>.auth.<region>.amazoncognito.com`
   - 사용자 지정 도메인`https://auth.example.com`
7. 권한 있는 리디렉션 URIs에서 사용자 풀 도메인의 `/oauth2/idpresponse` 엔드포인트를 입력합니다.
   - 기본 Amazon Cognito 도메인: `https://<your-prefix>.auth.<region>.amazoncognito.com/oauth2/idpresponse`
   - 사용자 지정 도메인`https://auth.example.com/oauth2/idpresponse`
8. 생성을 선택합니다. Google이 표시하는 클라이언트 ID와 클라이언트 보안 암호를 안전하게 저장합니다. 사용자 풀에 Google을 추가할 때 이러한 값을 입력합니다.

> **Note**
>
> 2025년 6월 이후에 생성된 클라이언트의 경우 Google은 클라이언트 보안 암호를 한 번만 표시합니다. 즉시 저장합니다.

사용자 풀에 Google 추가

에서 Google을 IdP로 추가하려면 AWS Management Console

1. Amazon Cognito 콘솔에서 사용자 풀, 소셜 및 외부 공급자, 자격 증명 공급자 추가를 차례로 선택합니다.
2. Google을 선택합니다. Google Auth 플랫폼에서 클라이언트 ID와 클라이언트 암호를 입력합니다.
3. 승인된 범위에 공백으로 `profile email openid`구분하여를 입력합니다.
4. 원하는 Google 속성을 사용자 풀 속성에 매핑합니다. 최소한 `email`에 매핑`email`하고에 매핑`email_verified`합니다`email_verified`. 를 매핑하지 않으면 `email_verified` Amazon Cognito는 이메일을 확인되지 않은 것으로 취급합니다.
5. 자격 증명 공급자 추가를 선택하고 앱 클라이언트에서 Google을 활성화합니다.

Google 로그인 테스트

관리형 로그인 로그인 페이지를 열고 Google 계속을 선택하거나 권한 부여 엔드포인트를 직접 엽니다.

```
https://<your-prefix>.auth.<region>.amazoncognito.com/oauth2/authorize?
response_type=code&client_id=<app-client-id>&redirect_uri=<your-app-
callback>&identity_provider=Google
```

체크포인트

브라우저는 Google 계정 선택기를 완료하고 권한 부여 코드와 함께 `/oauth2/idpresponse` 앱으로 돌아갑니다. 사용자 관리에서 Google 공급자, 채워진 `email`,가 로 `email_verified` 설정된 새 페더레이션 사용자가 나타납니다`true`. `redirect_uri_mismatch` 오류가 표시되지 않습니다.

Google 관련 함정

확인되지 않은 동의 화면 사용자 한도

> 테스트 게시 상태가 인 앱은 나열하는 테스트 사용자로 제한됩니다. 확인되지 않은 앱 화면을 보여주는 앱은 프로젝트 수명 동안 100명의 새 사용자로 제한되며, 제한은 재설정할 수 없습니다. 그러나 앱이 기본 로그인 범위(`openid`, `email`, `profile`)만 요청하는 경우 테스트 사용자를 나열할 필요가 없으며 경고가 표시되지 않으며 권한 부여가 만료되지 않습니다. 이러한 범위를 유지하여 한도를 피하거나 확인 센터에서 앱을 확인합니다.

리디렉션-URI 전파

승인된 오리진 및 리디렉션 URIs에 대한 변경 사항이 적용되려면 5분에서 몇 시간 정도 걸릴 수 있습니다. 불일치는를 반환하고 등록되지 않은 오리진은 `redirect_uri_mismatch`를 반환합니다`origin_mismatch`.

email_verified 및 계정 연결

Google은 Gmail 및 Google Workspace 계정에 `email_verified=true` 대해 어설션합니다. Amazon Cognito는 Google 사용자를 동일한 이메일을 가진 기존 사용자와 자동으로 병합하지 않습니다. 연결하려면 연결(예: 사전 가입 Lambda 트리거 및 `AdminLinkProviderForUser`)을 구현하고 신뢰할 수 있는 `email_verified` 값에만 연결합니다.

소셜 자격 증명 공급자로 Sign in with Apple 추가

> **Note**
>
> 공급자 콘솔에 대해 마지막으로 확인된 날짜: 2026년 10월 2일. Apple 단계는 Apple 개발자 계정의 인증서, 식별자 및 프로필을 사용합니다. 계정 보유자 또는 관리자 역할이 있어야 합니다.

Apple로 로그인하려면 앱 ID, 서비스 ID(Amazon Cognito 클라이언트 ID), 프라이빗 키, 팀 ID 등 네 가지 아티팩트가 필요합니다. 그런 다음 사용자 풀에 Apple을 ID 제공업체(IdP)로 추가합니다. 먼저 관리형 로그인을 활성화해야 합니다.

Apple 식별자 및 키 생성

Apple 아티팩트를 생성하려면

1. 인증서, 식별자 및 프로필에서 앱 ID(식별자, +, 앱 IDs, 앱)를 생성하고 Apple로 로그인 기능을 활성화합니다.
2. 서비스 ID(식별자, +, 서비스 IDs 생성합니다. 와 같은 설명과 역도메인 식별자를 입력합니다`com.example.auth.cognito`. 이 식별자는 Amazon Cognito Services ID(클라이언트 ID)입니다.
3. 서비스 ID를 선택하고 Sign in with Apple을 활성화한 다음 구성을 선택합니다. 기본 앱 ID를 선택한 다음 웹 URLs 입력합니다.

- 도메인 및 하위 도메인(호스트만 해당, 스키마 없음): 기본 Amazon Cognito 도메인의 `<your-prefix>.auth.<region>.amazoncognito.com` 경우 또는 사용자 지정 도메인의 `auth.example.com` 경우.
- 사용자 지정 도메인URLs(가 있는 전체 URL): 또는를 반환합니다. `https://` `https://<your-prefix>.auth.<region>.amazoncognito.com/oauth2/idpresponse` `https://auth.example.com/oauth2/idpresponse`

4. 프라이빗 키(키, +)를 생성하고 Sign in with Apple을 활성화한 다음 `.p8` 파일을 다운로드합니다. 한 번만 다운로드할 수 있습니다. 키 ID를 기록합니다.
5. 멤버십 페이지에서 10자 팀 ID를 기록합니다. 사용자 풀에 Apple을 추가할 때 서비스 ID, 팀 ID, 키 ID 및 프라이빗 키를 입력합니다.

사용자 풀에 Apple 추가

에서 Apple을 IdP로 추가하려면 AWS Management Console

1. Amazon Cognito 콘솔에서 사용자 풀, 소셜 및 외부 공급자, 자격 증명 공급자 추가, Apple로 로그인을 차례로 선택합니다.
2. 서비스 ID, 팀 ID, 키 ID 및 프라이빗 키를 입력합니다. Amazon Cognito는 프라이빗 키를 사용하여 클라이언트 보안 암호를 생성합니다.
3. 승인된 범위에를 입력합니다`email name`. 사용자의 이름이 필요하지 않은 `email` 경우에만를 사용합니다.
4. `email`에 매핑`email`하고 사용자 이름 속성`sub`에 매핑합니다. Apple은 첫 번째 로그인 시에만 사용자 이름을 반환하므로 매핑하는 경우 캡처합니다.
5. 자격 증명 공급자 추가를 선택하고 앱 클라이언트에서 Apple을 활성화합니다.

Apple로 로그인 테스트

관리형 로그인 로그인 페이지를 열고 Apple 계속을 선택합니다. 내 이메일 공유와 내 이메일 숨기기를 모두 테스트합니다.

체크포인트

브라우저는 권한 부여 코드와 함께 `/oauth2/idpresponse`를 통해 앱으로 돌아갑니다. 사용자가 SignInWithApple 공급자 및와 함께 사용자 풀에 나타납니다`sub`. 내 이메일 공유 실행에는 실제 이메일이 표시되고 내 이메일 숨기기 실행에는 `@privaterelay.appleid.com` 주소가 표시됩니다.

`invalid_client` 오류는 거의 항상 클라이언트 보안 JWT가 만료되었거나 서비스 ID 또는 반환 URL이 `/oauth2/idpresponse` 정확히 일치하지 않음을 의미합니다.

Apple 관련 함정

클라이언트 보안 암호는 만료되는 JWT입니다.

Apple의 클라이언트 보안 암호는 정적 문자열이 아닙니다. 최대 수명이 ES256-signed JWT이며, 만료되면 Apple 로그인이 경고 `invalid_client` 없이 중단됩니다. 여기에 설명된 구성에서는 Amazon Cognito에 프라이빗 키(서비스 ID, 팀 ID 및 키 ID 포함)를 부여하고 Amazon Cognito는 해당 키에서 클라이언트 보안 암호 JWT를 생성하므로 사전 빌드된 보안 암호를 붙여넣지 않습니다. Apple에 등록한 프라이빗 키를 IdP 구성에서 유효하고 최신 상태로 유지합니다. Apple 키를 취소하거나 교체하는 경우 IdP(예:)를 새 키`UpdateIdentityProvider`로 업데이트합니다.

프라이빗 릴레이 이메일 및 계정 연결

사용자가 내 이메일 숨기기를 선택하면 Apple은 실제 주소 대신 `@privaterelay.appleid.com` 주소를 반환합니다. 이 주소는 동일한 사람이 다른 공급자와 사용하는 이메일과 다르므로 Apple이 관여할 때 자격 증명 공급자 간에 계정을 이메일로 연결하지 마십시오. `sub` 대신 공급자가 연결합니다. 릴레이 주소로 메일을 보내려면 이메일 통신을 위해 Apple에 로그인에서 전송 도메인을 등록합니다.

email_verified type 및 관리형 계정

Apple의 `email_verified` 클레임은 문자열 `"true"` 또는 부울로 도착할 수 있습니다. `true`둘 다 처리합니다. 소비자 Apple 계정의 경우 항상 마찬가지입니다. Apple at Work & School(관리형) 계정의 경우 false일 수 있습니다. 병합 로직 앞에 `sub`기반 연결을 선호하고 문자열 값을 정규화합니다.

Facebook을 소셜 자격 증명 공급자로 추가

> **Note**
>
> 공급자 콘솔에 대해 마지막으로 확인된 날짜: 2026년 10월 2일. Facebook 단계에서는 Meta for Developers 콘솔을 사용합니다.이 콘솔은 사용 사례에 대한 앱 설정을 구성합니다. OAuth 리디렉션 설정은 사용 사례, 사용자 지정, 설정 아래에 있습니다.

사용자가 Facebook으로 로그인하도록 하려면 Facebook 로그인 사용 사례로 메타 앱을 생성한 다음 사용자 풀에 Facebook을 ID 제공업체(IdP)로 추가합니다. 먼저 관리형 로그인을 활성화해야 합니다.

Facebook 로그인용 메타 앱 생성

Meta 앱을 생성하고 구성하려면

1. 개발자용 Meta에 로그인하고 내 앱을 선택한 다음 앱 생성을 선택합니다.
2. 사용 사례의 경우 Facebook Login을 사용하여 사용자로부터 데이터 인증 및 요청을 선택합니다. 앱 생성을 완료합니다. 새 앱은 개발 모드에서 시작됩니다.
3. 앱 설정에서 기본에서 앱 ID와 앱 암호를 기록합니다. 앱 도메인을 사용자 풀 도메인 호스트(`<your-prefix>.auth.<region>.amazoncognito.com` 또는 `auth.example.com`)로 설정하고 관리형 로그인 로그인 URL을 사이트 URL로 사용하여 웹 사이트 플랫폼을 추가합니다.
4. 사용 사례로 이동하여 Facebook 로그인 사용 사례를 선택한 다음 사용자 지정을 선택합니다. `public_profile` 권한(필수)을 확인하고를 추가합니다`email`.
5. 사용 사례 설정에서 유효한 OAuth 리디렉션 URIs 사용자 풀 도메인의 `/oauth2/idpresponse` 엔드포인트로 설정합니다.

   - 기본 Amazon Cognito 도메인: `https://<your-prefix>.auth.<region>.amazoncognito.com/oauth2/idpresponse`
   - 사용자 지정 도메인`https://auth.example.com/oauth2/idpresponse`

사용자 풀에 Facebook 추가

에서 Facebook을 IdP로 추가하려면 AWS Management Console

1. Amazon Cognito 콘솔에서 사용자 풀, 소셜 및 외부 공급자, 자격 증명 공급자 추가, Facebook을 차례로 선택합니다.
2. 앱 ID와 앱 암호를 입력하고 Facebook API 버전을 선택합니다. 각 버전에는 중단 날짜가 있으며 범위는 버전마다 다를 수 있으므로 사용 가능한 최신 버전을 선택합니다.
3. 승인된 범위에를 쉼표로 `public_profile,email`구분하여 입력합니다.
4. `email`에 매핑`email`하고 `name`에 매핑합니다`name`.
5. 자격 증명 공급자 추가를 선택하고 앱 클라이언트에서 Facebook을 활성화합니다.

Facebook 로그인 테스트

앱이 개발 모드에 있는 동안 자체 Facebook 계정을 앱 역할(관리자, 개발자 또는 테스터)로 추가합니다. 그런 다음 관리형 로그인 로그인 페이지를 열고 Facebook 계속을 선택합니다.

체크포인트

Facebook은 권한 부여 코드와 함께 `/oauth2/idpresponse`를 통해 앱으로 돌아가고 및 `email` `name`가 채워진 사용자 풀에 새 페더레이션 사용자가 나타납니다. 그런 다음 앱을 라이브 모드로 전환하고 역할이 아닌 계정이 로그인할 수 있는지 확인합니다. '앱이 활성화되지 않음' 또는 '개발 모드'가 표시되는 비역할 계정은 앱이 여전히 개발 모드에 있거나 고급 액세스가 부족함을 의미합니다.

Facebook 관련 함정

개발 모드 및 앱 검토

새 앱은 개발 모드이며 앱 역할 사용자(관리자, 개발자, 테스터)만 로그인할 수 있습니다. 일반 퍼블릭이 로그인하도록 하려면 앱을 라이브 모드로 전환하고 비즈니스 확인을 완료한 다음에 대한 고급 액세스 권한을 얻습니다`email`(및에 대한 액세스 권한 증가`public_profile`). 처음 테스터는 역할이 아닌 계정으로 테스트하기 때문에 종종 "앱이 활성화되지 않음"에 도달합니다.

email_verified 클레임 없음

Facebook은 `email_verified` 클레임을 반환하지 않으므로 매핑할 사항이 없으며 Amazon Cognito는 Facebook 이메일을 확인되지 않은 것으로 취급합니다. Facebook에서 확인된 이메일을 가정하지 말고 의도적이고 신뢰할 수 있는 로직을 통해서만 이메일로 계정을 연결하세요.

범위를 작게 유지

로그인 완료는 약 4개의 권한을 급감합니다. `public_profile` 및에 대한 요청을 유지합니다`email`. Amazon Cognito 페더레이션은 iOS의 Facebook Limited Login과도 호환되지 않습니다.

소셜 자격 증명 공급자로 Login with Amazon 추가

> **Note**
>
> 공급자 콘솔에 대해 마지막으로 확인된 날짜: 2026년 10월 2일. Login with Amazon 단계는 Amazon Developer 콘솔(콘 AWS 솔 아님)을 사용합니다.

사용자가 Amazon 계정으로 로그인하도록 하려면 Login with Amazon 보안 프로필을 등록한 다음 사용자 풀에 Login with Amazon을 ID 제공업체(IdP)로 추가합니다. 먼저 관리형 로그인을 활성화해야 합니다.

Login with Amazon 보안 프로필 등록

보안 프로필을 생성하고 구성하려면

1. Amazon Developer 콘솔에 로그인하거나 앱 및 서비스로 이동한 다음 Login with Amazon으로 이동합니다.
2. 새 보안 프로필 생성을 선택합니다. 이름(동의 화면에 표시됨), 설명 및 동의 개인 정보 보호 고지 URL을 입력합니다. 저장을 선택합니다.
3. 보안 프로필의 클라이언트 ID와 클라이언트 보안 암호를 공개합니다. 사용자 풀에 Login with Amazon을 추가할 때 이를 입력합니다.
4. 프로필의 웹 설정을 열고 편집을 선택합니다. 아래 값을 입력한 다음 저장을 입력합니다.

   - 허용된 오리진(HTTPS, 경로 없음): `https://<your-prefix>.auth.<region>.amazoncognito.com` 또는 `https://auth.example.com`.
   - 허용된 반환 URLs(`/oauth2/idpresponse`엔드포인트): `https://<your-prefix>.auth.<region>.amazoncognito.com/oauth2/idpresponse` 또는 `https://auth.example.com/oauth2/idpresponse`. Amazon Cognito는 리디렉션 흐름을 사용하므로 반환 URL은 의존하는 값입니다.

사용자 풀에 Login with Amazon 추가

에서 Login with Amazon을 IdP로 추가하려면 AWS Management Console

1. Amazon Cognito 콘솔에서 사용자 풀, 소셜 및 외부 공급자, 자격 증명 공급자 추가, Amazon으로 로그인을 차례로 선택합니다.
2. 보안 프로필에서 클라이언트 ID와 클라이언트 보안 암호를 입력합니다.
3. 승인된 범위에를 입력합니다`profile`. 그러면 사용자의 이름, 이메일 및 사용자 ID가 반환됩니다. (우편 번호가 필요한 `postal_code` 경우에만를 추가합니다.)
4. `email`에 매핑`email`하고, `name`에 매핑하고`name`, 사용자 이름 속성`user_id`에 매핑합니다.
5. 자격 증명 공급자 추가를 선택하고 앱 클라이언트에서 Login with Amazon을 활성화합니다.

Amazon 로그인으로 로그인 테스트

관리형 로그인 로그인 페이지를 열고 Amazon 계속을 선택한 다음 인증하고 동의 화면을 수락합니다.

체크포인트

브라우저는 권한 부여 코드와 함께 /oauth2/idpresponse를 통해 앱으로 돌아가며, 사용자는 사용자 풀에 사용자 이름 앞에 접두사가 붙은 페더레이션 사용자로 표시됩니다LoginWithAmazon_. ID 토큰에는 매핑된 email 및가 포함됩니다name. 트리거로 설정false하지 않는 한는를 email_verified 읽습니다(위험 참조).

Amazon으로 로그인 함정

email_verified 클레임 없음

profile 범위는 email, name및 만 반환user_id합니다. 매핑할 email_verified 클레임은 없습니다. 매핑된 이메일은 기본적으로 확인되지 않으므로 Login with Amazon users는 로 email_verified 설정된 상태로 도착합니다false.

계정 연결에 미치는 영향

이메일이 확인되지 않았으므로 Amazon Cognito는 Login with Amazon 사용자를 기존 이메일 확인 사용자에게 자동으로 병합하지 않습니다. 확인된 이메일을 연결하거나 표시하려면 신뢰 모델이 Amazon이 작성한 이메일에 대한 신뢰를 정당화하는 경우에만 사전 가입 Lambda 트리거(및 AdminLinkProviderForUser)를 사용합니다.

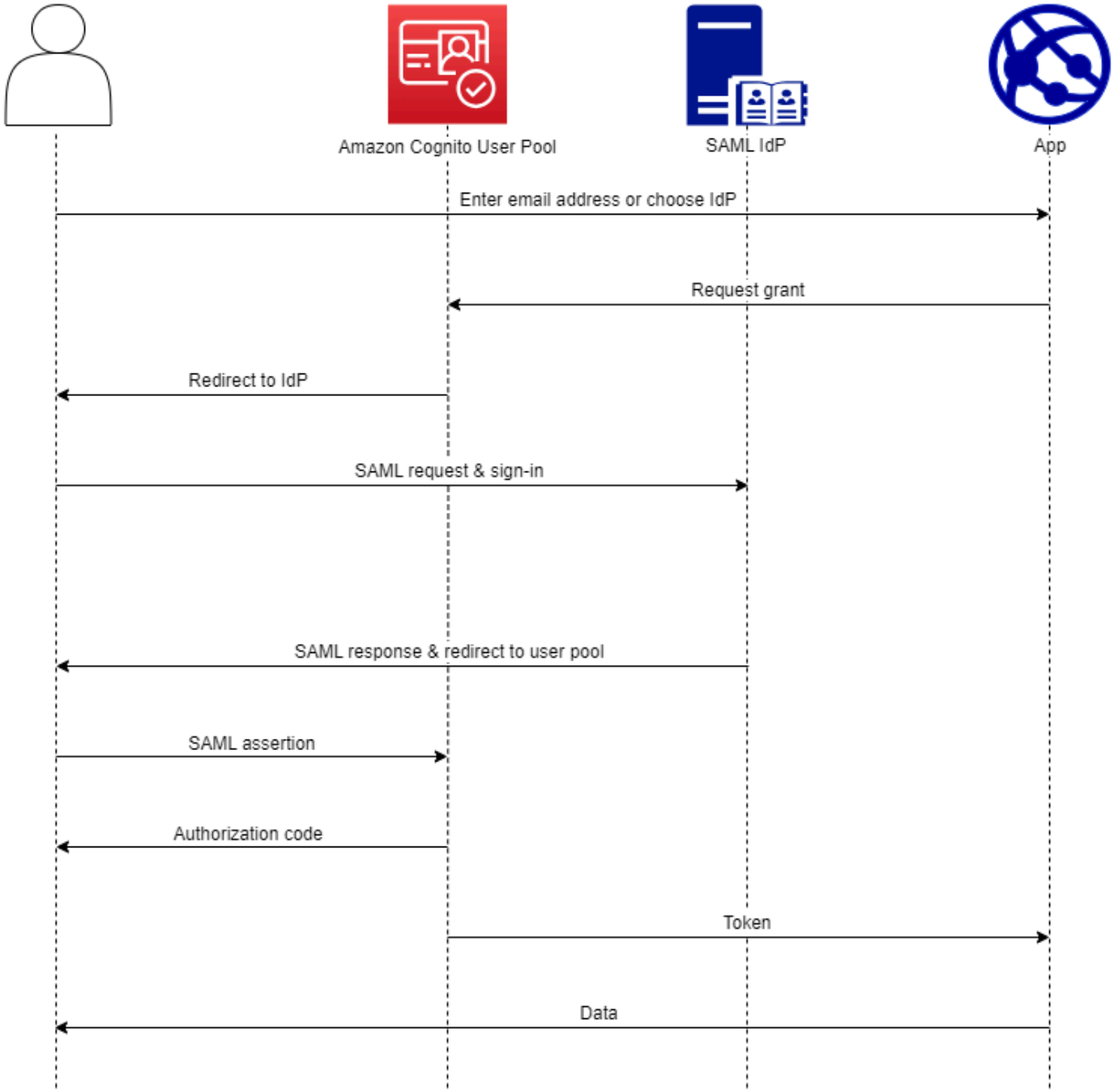
매핑된 속성은 쓰기 가능해야 합니다.

매핑하는 모든 사용자 지정 속성은 앱 클라이언트에서 변경 및 쓰기가 가능해야 합니다. 그렇지 않으면 로그인이 실패하거나 자동으로 속성을 건너뜁니다. 표준 email 및 name은 괜찮습니다.

사용자 풀에서 SAML ID 제공업체 사용

Microsoft Active Directory Federation Services(ADFS) 또는 Shibboleth 같은 SAML ID 제공업체(idP)를 통해 웹 및 모바일 앱 사용자가 로그인하게 할 수 있습니다. SAML 2.0 표준을 지원하는 SAML IdP를 선택해야 합니다.

관리형 로그인을 통해 Amazon Cognito는 로컬 및 타사 IdP 사용자를 인증하고 JSON 웹 토큰(JWT)을 발급합니다. Amazon Cognito에서 발급하는 토큰을 사용하면 여러 ID 소스를 모든 앱에 걸쳐 범용 OpenID Connect(OIDC) 표준으로 통합할 수 있습니다. Amazon Cognito는 타사 제공업체의 SAML 어설션을 해당 SSO 표준으로 처리할 수 있습니다. 에서 AWS Management Console,를 통해 AWS CLI또는 Amazon Cognito 사용자 풀 API를 사용하여 SAML IdP를 생성하고 관리할 수 있습니다. 에서 첫 번째 SAML IdP를 생성하려면 섹션을 AWS Management Console참조하세요사용자 풀에서 SAML ID 제공업체 추가 및 관리.



Note

타사 IdP를 통해 로그인하는 페더레이션은 Amazon Cognito 사용자 풀의 기능입니다. Amazon Cognito 페더레이션 ID라고도 하는 Amazon Cognito ID 풀은 각 ID 풀에서 별도로 설정해야 하

는 페더레이션의 구현입니다. 사용자 풀은 ID 풀에 대한 타사 IdP일 수 있습니다. 자세한 내용은 Amazon Cognito 자격 증명 풀 섹션을 참조하세요.

IdP 구성에 대한 빠른 참조

요청을 수락하고 사용자 풀로 응답을 보내도록 SAML IdP를 구성해야 합니다. SAML IdP 설명서에는 사용자 풀을 SAML 2.0 IdP의 신뢰 당사자 또는 애플리케이션으로 추가하는 방법에 대한 정보가 포함되어 있습니다. 다음 설명서에서는 SP 엔터티 ID 및 어설션 소비자 서비스(ACS) URL에 제공해야 하는 값을 제공합니다.

사용자 풀 SAML 값 빠른 참조

SP 엔터티 ID

```
urn:amazon:cognito:sp:us-east-1_EXAMPLE
```

ACS URL

```
https://Your user pool domain/saml2/idpresponse
```

ID 제공업체를 지원하도록 사용자 풀을 구성해야 합니다. 외부 SAML IdP를 추가하는 상위 단계는 다음과 같습니다.

1. IdP 에서 SAML 메타데이터를 다운로드하거나 메타데이터 엔드포인트에 대한 URL을 검색합니다. 타사 SAML ID 제공업체 구성 섹션을 참조하세요.
2. 사용자 풀에 새 IdP를 추가합니다. SAML 메타데이터를 업로드하거나 메타데이터 URL을 제공합니다. 사용자 풀에서 SAML ID 제공업체 추가 및 관리 섹션을 참조하세요.
3. 앱 클라이언트에 IdP를 할당합니다. 앱 클라이언트를 사용한 애플리케이션별 설정 섹션을 참조하세요.

주제

- Amazon Cognito 사용자 풀 SAML IdP에 대해 알아야 할 사항
- SAML 사용자 이름의 대/소문자 구분
- 타사 SAML ID 제공업체 구성
- 사용자 풀에서 SAML ID 제공업체 추가 및 관리

- Amazon Cognito 사용자 풀에서 SAML 세션 시작
- 단일 로그아웃으로 SAML 사용자 로그아웃
- SAML 서명 및 암호화
- SAML ID 제공업체 이름 및 식별자

Amazon Cognito 사용자 풀 SAML IdP에 대해 알아야 할 사항

SAML 2.0 IdP 구현에는 몇 가지 요구 사항 및 제한 사항이 있습니다. IdP를 구현할 때 이 섹션을 참조하세요. 또한 사용자 풀을 사용한 SAML 페더레이션 중에 오류를 해결하는 데 유용한 정보를 찾을 수 있습니다.

Amazon Cognito가 SAML 어설션을 처리합니다.

Amazon Cognito 사용자 풀은 사후 바인딩 엔드포인트와의 SAML 2.0 페더레이션을 지원합니다. 이렇게 하면 사용자 풀이 사용자 에이전트를 통해 IdP로부터 직접 SAML 응답을 받으므로 앱에서 SAML 어설션 응답을 검색하거나 구문 분석할 필요가 없습니다. 사용자 풀은 애플리케이션을 대신하여 서비스 공급자(SP) 역할을 합니다. Amazon Cognito는 SAML V2.0 기술 개요의 섹션 5.1.2 및 5.1.4에 설명된 대로 SP 시작 및 IdP 시작 Single Sign-On(SSO)을 지원합니다.

유효한 IdP 서명 인증서 제공

사용자 풀에서 SAML IdP를 구성할 경우 SAML 제공업체 메타데이터의 서명 인증서가 만료되지 않아야 합니다.

사용자 풀은 여러 서명 인증서를 지원합니다.

SAML IdP가 SAML 메타데이터에 서명 인증서를 두 개 이상 포함하는 경우, 로그인 시 사용자 풀은 SAML 어설션이 SAML 메타데이터에 있는 인증서와 일치할 경우 유효하다고 판단합니다. 각 서명 인증서의 길이는 4,096자 이하여야 합니다.

릴레이 상태 파라미터 유지

Amazon Cognito와 SAML IdP는 `relayState` 파라미터를 사용하여 세션 정보를 유지 관리합니다.

1. Amazon Cognito는 80바이트보다 큰 `relayState` 값을 지원합니다. SAML 사양에는 `relayState` 값의 "길이가 80바이트를 초과해서는 안 된다"고 명시되어 있지만, 현재 업계 관행은 이 동작에서 벗어나는 경우가 많습니다. 결과적으로 80바이트보다 큰 `relayState` 값을 거부하면 많은 표준 SAML 공급자 통합이 중단됩니다.
2. `relayState` 토큰은 Amazon Cognito Cognito에서 유지 관리하는 상태 정보에 대한 불투명한 참조입니다. Amazon Cognito는 `relayState` 파라미터의 내용을 보장하지 않습니다. 앱이 결과에 따라 달라지도록 내용을 구문 분석하지 마세요. 자세한 내용은 SAML 2.0 사양을 참조하세요.

ACS 엔드포인트 식별

SAML 자격 증명 공급자에 어설션 소비자 엔드포인트를 설정해야 합니다. IdP는 SAML 어설션을 통해 사용자를 이 엔드포인트로 리디렉션합니다. SAML ID 제공업체의 SAML 2.0 POST 바인딩에 대해 사용자 풀 도메인에서 다음 엔드포인트 구성합니다.

```
https://Your user pool domain/saml2/idpresponse
With an Amazon Cognito domain:
https://mydomain.auth.us-east-1.amazoncognito.com/saml2/idpresponse
With a custom domain:
https://auth.example.com/saml2/idpresponse
```

사용자 풀 도메인에 대한 자세한 정보는 사용자 풀 도메인 구성 단원을 참조하세요.

재생된 어설션 없음

Amazon Cognito `saml2/idpresponse` 엔드포인트에 대한 SAML 어설션을 반복하거나 재생할 수 없습니다. 재생된 SAML 어설션에는 이전 IdP 응답의 ID와 중복되는 어설션 ID가 있습니다.

사용자 풀 ID는 SP 엔터티 ID입니다.

대상 그룹 URI 또는 SP 엔터티 ID라고도 하는 서비스 제공업체(SP) `urn`의 사용자 풀 ID를 IdP에 제공해야 합니다. 사용자 풀에 대한 대상 URI는 다음 형식을 갖습니다.

```
urn:amazon:cognito:sp:us-east-1_EXAMPLE
```

Amazon Cognito 콘솔의 사용자 풀 개요에서 사용자 풀 ID를 찾을 수 있습니다.

모든 필수 속성 매핑

SAML IdP에서 사용자 풀에 필요한 대로 설정한 속성에 대한 속성 값도 제공하도록 구성해야 합니다. 예를 들어 `email`은 사용자 풀의 일반적인 필수 속성입니다. 사용자가 로그인하려면 먼저 SAML IdP 어설션에 사용자 풀 속성 `email`에 매핑되는 클레임이 포함되어야 합니다. 속성 매핑에 대한 자세한 내용은 프로필 및 토큰에 IdP 속성 매핑을 참조하십시오.

어설션 형식에는 특정 요구 사항이 있습니다.

SAML IdP는 SAML 어설션에 다음 클레임을 포함해야 합니다.

- `NameID` 클레임. Amazon Cognito는 `NameID`를 통해 SAML 어설션을 대상 사용자와 연결합니다. `NameID`가 변경되면 Amazon Cognito는 어설션을 새 사용자를 위한 것으로 간주합니다. IdP 구성에서 `NameID`로 설정한 속성에는 영구 값이 있어야 합니다. 사용자 풀의 일관된 사용자 프로필에 SAML 사용자를 할당하려면 변경되지 않는 값이 있는 속성에서 `NameID` 클레임을 할당합니다.

```
<saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:persistent">
  carlos
</saml2:NameID>
```

`urn:oasis:names:tc:SAML:1.1:nameid-format:persistent`의 IdP NameID 클레임에서 Format은 IdP가 변경되지 않는 값을 전달하고 있음을 나타냅니다. Amazon Cognito는 이 형식 선언을 필요로 하지 않으며, IdP가 NameID 클레임의 형식을 지정하지 않는 경우 `urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified` 형식을 할당합니다. 이 동작은 SAML 2.0 사양의 섹션 2.2.2 복합 유형 NameIDType을 준수합니다.

- 사용자 풀 SP 엔터티 ID를 응답 대상으로 설정하는 Audience 값이 포함된 AudienceRestriction 클레임.

```
<saml:AudienceRestriction>
  <saml:Audience> urn:amazon:cognito:sp:us-east-1_EXAMPLE
</saml:AudienceRestriction>
```

- SP가 시작한 Single Sign-On의 경우 원본 SAML 요청 ID의 InResponseTo 값이 있는 Response 요소입니다.

```
<saml2p:Response Destination="https://mydomain.auth.us-east-1.amazoncognito.com/
saml2/idpresponse" ID="id123" InResponseTo="_dd0a3436-bc64-4679-
a0c2-cb4454f04184" IssueInstant="Date-time stamp" Version="2.0"
 xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol" xmlns:xs="http://
www.w3.org/2001/XMLSchema">
```

> **Note**
>
> IdP가 시작한 SAML 어설션에는 InResponseTo 값이 포함되어서는 안 됩니다.

- 사용자 풀 saml2/idpresponse 엔드포인트의 Recipient 값과 SP가 시작한 SAML의 경우 원본 SAML 요청 ID와 일치하는 InResponseTo 값이 있는 SubjectConfirmationData 요소입니다.

```
<saml2:SubjectConfirmationData InResponseTo="_dd0a3436-bc64-4679-a0c2-
cb4454f04184" NotOnOrAfter="Date-time stamp" Recipient="https://mydomain.auth.us-
east-1.amazoncognito.com/saml2/idpresponse"/>
```

SP가 시작한 로그인 요청

권한 부여 엔드포인트에서 사용자를 IdP 로그인 페이지로 리디렉션하면, Amazon Cognito는 SAML 요청을 `HTTP GET` 요청의 URL 파라미터에 포함합니다. SAML 요청에는 ACS 엔드포인트를 포함하여 사용자 풀에 대한 정보가 포함되어 있습니다. 이러한 요청에 암호화 서명을 선택적으로 적용할 수 있습니다.

요청에 서명하고 응답을 암호화합니다.

SAML 제공업체가 있는 모든 사용자 풀은 Amazon Cognito가 SAML 요청에 할당하는 디지털 서명에 대한 비대칭 키 페어 및 서명 인증서를 생성합니다. 암호화된 SAML 응답을 지원하도록 구성하는 모든 외부 SAML IdP는 Amazon Cognito가 해당 제공업체에 대한 새 키 페어 및 암호화 인증서를 생성하도록 합니다. 퍼블릭 키가 포함된 인증서를 보고 다운로드하려면 Amazon Cognito 콘솔의 소셜 및 외부 제공업체 메뉴에서 IdP를 선택하세요.

사용자 풀의 SAML 요청과 신뢰를 설정하려면 IdP에 사용자 풀 SAML 2.0 서명 인증서 사본을 제공합니다. 서명된 요청을 수락하도록 IdP를 구성하지 않으면 IdP가 사용자 풀이 서명한 SAML 요청을 무시할 수 있습니다.

1. Amazon Cognito는 사용자가 IdP 에 전달하는 SAML 요청에 디지털 서명을 적용합니다. 사용자 풀은 모든 단일 로그아웃(SLO) 요청에 서명하며, 모든 SAML 외부 IdP 에 대한 Single Sign-On(SSO) 요청에 서명하도록 사용자 풀을 구성할 수 있습니다. 인증서 사본을 제공하면 IdP가 사용자 SAML 요청의 무결성을 확인할 수 있습니다.
2. SAML IdP는 암호화 인증서로 SAML 응답을 암호화할 수 있습니다. SAML 암호화로 IdP를 구성할 경우 IdP는 암호화된 응답만 전송해야 합니다.

영숫자가 아닌 문자 인코딩

Amazon Cognito는 IdP가 속성 값으로 전달하는 4바이트 UTF-8 문자(예: # 또는 #)를 허용하지 않습니다. 문자를 Base64로 인코딩하여 텍스트로 전달한 다음 앱에서 디코딩할 수 있습니다.

다음 예제에서는 속성 클레임이 허용되지 않습니다.

```
<saml2:Attribute Name="Name" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:unspecified">
  <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">#</saml2:AttributeValue>
</saml2:Attribute>
```

이전 예제와 달리 다음 속성 클레임이 허용됩니다.

```
<saml2:Attribute Name="Name" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:unspecified">
  <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">8J+YkA==</saml2:AttributeValue>
</saml2:Attribute>
```

메타데이터 엔드포인트에는 유효한 전송 계층 보안이 있어야 합니다.

HTTPS 메타데이터 엔드포인트 URL을 사용하여 SAML IdP를 생성하는 동안 `InvalidParameterException`(예: "*<metadata endpoint>*에서 메타데이터를 가져오는 동안 오류가 발생했습니다(Error retrieving metadata from <metadata endpoint>)")이 발생할 경우 메타데이터 엔드포인트에 SSL이 올바르게 설정되어 있고 유효한 SSL 인증서가 연결되어 있는지 확인합니다. 인증서 유효성 검사에 대한 자세한 내용은 SSL/TLS 인증서란? 섹션을 참조하십시오.

메타데이터 엔드포인트는 HTTP 또는 HTTPS용 표준 TCP 포트에 있어야 합니다.

Amazon Cognito는 HTTP용 표준 TCP 포트 80 및 HTTPS용 443의 SAML 제공업체에 대한 메타데이터 URL만 허용합니다. 보안 모범 사례로 `https://` 접두사가 있는 TLS 암호화 URL에서 SAML 메타데이터를 호스팅합니다. 메타데이터 URL을 *`http://www.example.com/saml2/metadata.xml`* 또는 *`https://www.example.com/saml2/metadata.xml`* 형식으로 입력합니다. Amazon Cognito 콘솔은 `https://` 접두사가 있는 메타데이터 URL만 허용합니다. CreateIdentityProvider 및 UpdateIdentityProvider를 사용하여 IdP 메타데이터를 구성할 수도 있습니다.

IdP가 시작된 SAML이 있는 앱 클라이언트는 SAML로만 로그인할 수 있습니다.

앱 클라이언트에서 IdP 시작 서명을 지원하는 SAML 2.0 IdP에 대한 지원을 활성화할 경우 해당 앱 클라이언트에 다른 SAML 2.0 IdP만 추가할 수 있습니다. 이러한 방식으로 구성된 앱 클라이언트에 사용자 풀의 사용자 디렉터리 및 모든 비SAML 외부 ID 제공업체를 추가할 수 없습니다.

로그아웃 응답은 POST 바인딩을 사용해야 합니다.

`/saml2/logout` 엔드포인트는 `LogoutResponse`를 `HTTP POST` 요청으로 수락합니다. 사용자 풀은 `HTTP GET` 바인딩이 있는 로그아웃 응답을 수락하지 않습니다.

메타데이터 서명 인증서 교체

Amazon Cognito는 URL로 메타데이터를 제공할 때 최대 6시간 동안 SAML 메타데이터를 캐싱합니다. 메타데이터 서명 인증서 교체를 수행할 때 최소 6시간 동안 원본 인증서와 새 인증서를 모두 게시하도록 메타데이터 소스를 구성합니다. Amazon Cognito는 메타데이터 URL에서 캐시를 새로 고치면 각 인증서를 유효한 것으로 취급하고 SAML IdP가 새 인증서로 SAML 어설션 서명을 시작할 수 있습니다. 이 기간이 경과한 후에는 게시된 메타데이터에서 원본 인증서를 제거할 수 있습니다.

SAML 사용자 이름의 대/소문자 구분

페더레이션 사용자가 로그인을 시도할 때 SAML ID 제공업체(IdP)는 사용자의 SAML 어설션에서 Amazon Cognito에 고유한 `NameId`를 전달합니다. Amazon Cognito는 SAML 페더레이션 사용자를 해당 `NameId` 클레임으로 식별합니다. 사용자 풀의 대/소문자 구분 설정에 관계없이 Amazon Cognito는 고유하고 대소문자를 구분하는 `NameId` 클레임을 전달할 때 SAML IdP에서 반환된 페더레이션 사용자를 인식합니다. `email`과 같은 속성을 `NameId`에 매핑하고 사용자가 자신의 이메일 주소를 변경하는 경우 해당 사용자는 앱에 로그인할 수 없습니다.

변경되지 않는 값을 가진 IdP 속성의 SAML 어설션에서 `NameId`를 매핑합니다.

예를 들어 Carlos라는 사용자가 `NameId` 값인 `Carlos@example.com`을 전달한 Active Directory Federation Services(ADFS) SAML 어설션에서 가져온 대/소문자를 구분하지 않는 사용자 풀에서 사용자 프로필을 갖고 있는 경우 Carlos가 다음에 로그인을 시도할 때 ADFS IdP에서 `NameId` 값인 `carlos@example.com`을 전달합니다. `NameId`는 대/소문자를 정확하게 구분해야 하기 때문에 로그인이 실패합니다.

사용자가 `NameID` 변경 후 로그인할 수 없는 경우 사용자 풀에서 사용자 프로필을 삭제합니다. Amazon Cognito는 다음에 로그인할 때 새 사용자 프로필을 생성합니다.

주제

- 타사 SAML ID 제공업체 구성
- 사용자 풀에서 SAML ID 제공업체 추가 및 관리
- Amazon Cognito 사용자 풀에서 SAML 세션 시작
- 단일 로그아웃으로 SAML 사용자 로그아웃
- SAML 서명 및 암호화
- SAML ID 제공업체 이름 및 식별자

타사 SAML ID 제공업체 구성

사용자 풀에 SAML ID 제공업체(idP)를 추가하려면 IdP 의 관리 인터페이스에서 일부 구성을 업데이트해야 합니다. 이 섹션에서는 IdP에 제공해야 하는 값의 형식을 지정하는 방법을 설명합니다. 또한 사용자 풀에 대한 IdP 및 SAML 클레임을 식별하는 정적 또는 활성 URL 메타데이터 문서를 검색하는 방법에 대해서도 알아볼 수 있습니다.

타사 SAML 2.0 ID 제공업체(IdP) 솔루션에서 Amazon Cognito 사용자 풀에 대한 페더레이션 작업을 수행하도록 구성하려면 다음 어설션 소비자 서비스(ACS) URL `https://mydomain.auth.us-`

east-1.amazoncognito.com/saml2/idpresponse로 리디렉션하도록 SAML IdP를 구성해야 합니다. 사용자 풀에 Amazon Cognito 도메인이 있는 경우 Amazon Cognito 콘솔에 있는 사용자 풀의 도메인 메뉴에서 사용자 풀 도메인 경로를 찾을 수 있습니다.

일부 SAML IdP의 경우 대상 URI 또는 SP 엔터티 ID라고도 하는 urn를 urn:amazon:cognito:sp:*us-east-1_EXAMPLE* 형식으로 제공해야 합니다. Amazon Cognito 콘솔의 사용자 풀 개요에서 사용자 풀 ID를 찾을 수 있습니다.

또한 사용자 풀에서 필수 속성으로 지정한 모든 속성에 대한 값을 제공하도록 SAML IdP를 구성해야 합니다. 일반적으로 email은 사용자 풀에 필요한 속성이며, 이 경우 SAML IdP는 SAML 어설션에서 특정 형식의 email 클레임을 제공해야 하며, 해당 클레임을 해당 제공업체의 속성에 매핑해야 합니다.

타사 SAML 2.0 IdP 솔루션에 대한 다음 구성 정보는 Amazon Cognito 사용자 풀을 사용한 페더레이션 설정을 시작하기에 좋은 곳입니다. 최신 정보는 제공업체의 설명서를 직접 참조하세요.

SAML 요청에 서명하려면 사용자 풀 서명 인증서로 서명된 요청을 신뢰하도록 IdP를 구성해야 합니다. 암호화된 SAML 응답을 수락하려면 모든 SAML 응답을 사용자 풀에 암호화하도록 IdP를 구성해야 합니다. 공급자에게 이러한 기능 구성에 대한 설명서가 있습니다. Microsoft의 예제는 Microsoft Entra SAML 토큰 암호화 구성을 참조하세요.

> **Note**
>
> Amazon Cognito에는 ID 제공업체 메타데이터 문서만 필요합니다. 제공업체는 IAM 또는 AWS IAM Identity Center와의 SAML 2.0 페더레이션을 위한 사용자 지정 구성 정보를 제공할 수도 있습니다. Amazon Cognito 통합을 설정하는 방법을 알아보려면 메타데이터 문서를 검색하고 사용자 풀의 나머지 구성을 관리하기 위한 일반적인 지침을 확인하세요.

Solution	추가 정보
Microsoft Entra ID	페더레이션 메타데이터
Okta	SAML 앱 통합을 위해 IdP 메타데이터 및 SAML 서명 인증서를 다운로드하는 방법
Auth0	Auth0을 SAML ID 제공업체로 구성
Ping ID(PingFederate)	PingFederate에서 SAML 메타데이터 내보내기

Solution	추가 정보
JumpCloud	SAML 구성 정보
SecureAuth	SAML 애플리케이션 통합

사용자 풀에서 SAML ID 제공업체 추가 및 관리

Amazon Cognito에서 작동하도록 ID 제공업체를 구성한 후 사용자 풀 및 앱 클라이언트에 추가할 수 있습니다. 다음 절차에서는 Amazon Cognito 사용자 풀에서 SAML 제공업체를 생성, 수정 및 삭제하는 방법을 보여줍니다.

AWS Management Console

AWS Management Console 를 사용하여 SAML 자격 증명 공급자(IdPs.

SAML IdP를 생성하기 전에 타사 IdP로부터 얻은 SAML 메타데이터 문서가 있어야 합니다. 필요한 SAML 메타데이터 문서를 가져오거나 생성하는 방법에 대한 지침은 타사 SAML ID 제공업체 구성 섹션을 참조하세요.

사용자 풀에 SAML 2.0 IdP를 구성하려면

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 AWS 자격 증명을 입력합니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. 소셜 및 외부 제공업체 메뉴를 선택한 다음 ID 제공업체 추가를 선택합니다.
5. SAML IdP를 선택합니다.
6. 제공업체 이름을 입력합니다. `identity_provider` 요청 파라미터에서 이 친숙한 이름을 권한 부여 엔드포인트에 전달할 수 있습니다.
7. 식별자(Identifiers)를 쉼표로 구분하여 입력합니다. 식별자는 사용자가 로그인할 때 입력한 이메일 주소를 확인한 다음, 사용자 도메인에 해당하는 공급자로 사용자를 보내도록 Amazon Cognito에 지시합니다.
8. 사용자가 로그아웃할 때 Amazon Cognito에서 서명된 로그아웃 요청을 공급자에게 보내도록 하려는 경우 로그아웃 흐름 추가(Add sign-out flow)를 선택합니다. 관리형 로그인을 구성할 때 생성되는 `https://mydomain.auth.us-east-1.amazoncognito.com/saml2/logout` 엔드포인트에 로그아웃 응답을 보내도록 SAML 2.0 IdP를 구성해야 합니다. `saml2/logout` 엔드포인트는 POST 바인딩을 사용합니다.

> **Note**
>
> 이 옵션을 선택하고 SAML IdP에 서명된 로그아웃 요청이 필요한 경우 사용자 풀의 서명 인증서도 SAML IdP에 제공해야 합니다.
> SAML IdP는 서명된 로그아웃 요청을 처리하고 Amazon Cognito 세션에서 사용자를 로그아웃합니다.

9. IdP 시작 SAML 로그인 구성을 선택합니다. 보안 모범 사례로 SP 시작 SAML 어설션만 수락을 선택합니다. 원치 않는 SAML 로그인 세션을 안전하게 수락하도록 환경을 준비한 경우 SP 시작 및 IdP 시작 SAML 어설션 수락을 선택합니다. 자세한 내용은 Amazon Cognito 사용자 풀에서 SAML 세션 시작 섹션을 참조하세요.
10. 메타데이터 문서 소스(Metadata document source)를 선택합니다. IdP가 퍼블릭 URL에서 SAML 메타데이터를 제공하는 경우 메타데이터 문서 URL(Metadata document URL)을 선택하고 해당 퍼블릭 URL을 입력할 수 있습니다. 그렇지 않은 경우 메타데이터 문서 업로드(Upload metadata document)를 선택한 다음, 이전에 공급자로부터 다운로드한 메타데이터 파일을 선택합니다.

> **Note**
>
> 제공업체에게 퍼블릭 엔드포인트가 있는 경우 파일을 업로드하는 대신 메타데이터 문서 URL을 입력하는 것이 좋습니다. Amazon Cognito는 메타데이터 URL에서 메타데이터를 자동으로 새로 고칩니다. 일반적으로 메타데이터 새로 고침은 6시간마다 또는 메타데이터가 만료되기 전 중 더 빠른 시간에 발생합니다.

11. SAML 제공업체와 앱 간에 속성 매핑을 선택하여 SAML 제공업체 속성을 사용자 풀의 사용자 프로파일에 매핑합니다. 속성 맵에 사용자 풀 필수 속성을 포함합니다.

    예를 들어, 사용자 풀 속성(User pool attribute) `email`을 선택한 경우 IdP의 SAML 어설션에 표시된 대로 SAML 속성 이름을 입력합니다. IdP가 샘플 SAML 어설션을 제공하는 경우 이 샘플 어설션을 사용하여 이름을 찾을 수 있습니다. 일부 IdP는 `email`과 같은 간단한 이름을 사용하는 반면 다른 IdP는 다음과 같은 이름을 사용합니다.

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

12. 생성을 선택합니다.

API/CLI

다음 명령을 사용하여 SAML 자격 증명 공급자(IdP)를 생성 및 관리합니다.

IdP를 생성하고 메타데이터 문서를 업로드하려면

- AWS CLI: `aws cognito-idp create-identity-provider`

  메타데이터 파일이 포함된 예제: `aws cognito-idp create-identity-provider --user-pool-id us-east-1_EXAMPLE --provider-name=SAML_provider_1 --provider-type SAML --provider-details file:///details.json --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

  여기서 `details.json`에 다음 사항이 포함됩니다.

```
"ProviderDetails": {
      "MetadataFile": "<SAML metadata XML>",
      "IDPSignout" : "true",
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

> **Note**
>
> *<SAML ##### XML>*에 " 문자의 인스턴스가 포함된 경우 \를 이스케이프 문자(\")로 추가해야 합니다.

  메타데이터 URL이 포함된 예제: `aws cognito-idp create-identity-provider --user-pool-id us-east-1_EXAMPLE --provider-name=SAML_provider_1 --provider-type SAML --provider-details MetadataURL=https://myidp.example.com/sso/saml/metadata --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

- AWS API: CreateIdentityProvider

IdP에 대한 새 메타데이터 문서를 업로드하려면

- AWS CLI: `aws cognito-idp update-identity-provider`

  메타데이터 파일이 포함된 예제: `aws cognito-idp update-identity-provider --user-pool-id us-east-1_EXAMPLE --provider-name=SAML_provider_1 --provider-details file:///details.json --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

  여기서 `details.json`에 다음 사항이 포함됩니다.

```
"ProviderDetails": {
      "MetadataFile": "<SAML metadata XML>",
      "IDPSignout" : "true",
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

> **Note**
>
> *<SAML ##### XML>*에 " 문자의 인스턴스가 포함된 경우 \를 이스케이프 문자(\")로 추가해야 합니다.

  메타데이터 URL이 포함된 예제: `aws cognito-idp update-identity-provider --user-pool-id us-east-1_EXAMPLE --provider-name=SAML_provider_1 --provider-details MetadataURL=https://myidp.example.com/sso/saml/metadata --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

- AWS API: UpdateIdentityProvider

특정 IdP에 대한 정보를 가져오려면

- AWS CLI: `aws cognito-idp describe-identity-provider`

```
aws cognito-idp describe-identity-provider --user-pool-id us-east-1_EXAMPLE --provider-name=SAML_provider_1
```

- AWS API: DescribeIdentityProvider

모든 IdP에 대한 정보를 나열하려면

- AWS CLI: `aws cognito-idp list-identity-providers`

  예시: `aws cognito-idp list-identity-providers --user-pool-id us-east-1_EXAMPLE --max-results 3`

- AWS API: ListIdentityProviders

IdP를 삭제하려면

- AWS CLI: `aws cognito-idp delete-identity-provider`

  ```
  aws cognito-idp delete-identity-provider --user-pool-id us-east-1_EXAMPLE --provider-name=SAML_provider_1
  ```

- AWS API: DeleteIdentityProvider

사용자 풀을 신뢰 당사자로 추가하도록 SAML IdP를 설정하려면

- 사용자 풀 서비스 공급자 URN은 `urn:amazon:cognito:sp:us-east-1_EXAMPLE`입니다. Amazon Cognito에는 SAML 응답에서 이 URN과 일치하는 대상 제한 값이 필요합니다. IdP에서 SP로 보내는 응답 메시지에 대해 다음 POST 바인딩 엔드포인트를 사용하도록 IdP를 구성합니다.

  ```
  https://mydomain.auth.us-east-1.amazoncognito.com/saml2/idpresponse
  ```

- SAML IdP가 SAML 어설션의 사용자 풀에 대한 필수 속성과 `NameID`를 채워야 합니다. `NameID`는 사용자 풀에서 SAML 페더레이션 사용자를 고유하게 식별하는 데 사용됩니다. IdP는 각 사용자의 SAML 이름 ID를 대/소문자를 구분하는 일관된 형식으로 전달해야 합니다. 사용자 이름 ID의 값이 변경되면 새 사용자 프로필이 생성됩니다.

SAML 2.0 IDP에 서명 인증서를 제공하려면

- IdP가 SAML 로그아웃 요청을 검증하는 데 사용할 수 있는 퍼블릭 키의 사본을 Amazon Cognito에서 다운로드하려면 사용자 풀의 소셜 및 외부 제공업체 메뉴를 선택하고 IdP를 선택한 다음 서명 인증서 보기에서 .crt로 다운로드를 선택합니다.

Amazon Cognito 콘솔을 사용하여 사용자 풀에 설정한 SAML 공급자를 삭제할 수 있습니다.

SAML 공급자 삭제

1. Amazon Cognito 콘솔에 로그인합니다.
2. 탐색 창에서 [사용자 풀(User Pools)]을 선택한 다음 편집할 사용자 풀을 선택합니다.
3. 소셜 및 외부 제공업체 메뉴를 선택합니다.
4. 삭제할 SAML IdP 옆에 있는 라디오 버튼을 선택합니다.
5. 자격 증명 공급자 삭제(Delete identity provider) 메시지가 표시되면 SAML 공급자의 이름을 입력하여 삭제를 확인하고 삭제(Delete)를 선택합니다.

Amazon Cognito 사용자 풀에서 SAML 세션 시작

Amazon Cognito는 서비스 제공업체 시작(SP 시작), Single Sign-On(SSO) 및 IdP 시작 SSO를 지원합니다. 가장 좋은 방법은 사용자 풀에서 SP 시작 SSO를 구현하는 것입니다. SAML V2.0 Technical Overview(SAML V2.0 기술 개요)의 섹션 5.1.2에서는 SP 시작 SSO에 대해 설명합니다. Amazon Cognito는 앱의 자격 증명 공급자(IdP)입니다. 이 앱은 인증된 사용자에 대한 토큰을 검색하는 서비스 공급자(SP)입니다. 그러나 서드 파티 IdP를 사용하여 사용자를 인증할 때 Amazon Cognito가 SP입니다. SAML 2.0 사용자가 SP 시작 흐름으로 인증하는 경우 항상 먼저 Amazon Cognito에 요청하고 인증을 위해 IdP로 리디렉션해야 합니다.

일부 엔터프라이즈 사용 사례의 경우 내부 애플리케이션에 대한 액세스는 엔터프라이즈 IdP가 호스팅하는 대시보드의 북마크에서 시작됩니다. 사용자가 북마크를 선택하면 IdP가 SAML 응답을 생성하고 SP로 전송하여 해당 애플리케이션을 사용하여 사용자를 인증합니다.

IdP 시작 SSO를 지원하도록 사용자 풀에서 SAML IdP를 구성할 수 있습니다. IdP 시작 인증을 지원하는 경우 Amazon Cognito는 SAML 요청으로 인증을 시작하지 않기 때문에 Amazon Cognito는 수신한 SAML 응답을 요청했는지 확인할 수 없습니다. SP 시작 SSO에서 Amazon Cognito는 원본 요청에 대한 SAML 응답을 검증하는 상태 파라미터를 설정합니다. SP 시작 로그인을 사용하면 사이트 간 요청 위조(CSRF)로부터 보호할 수도 있습니다.

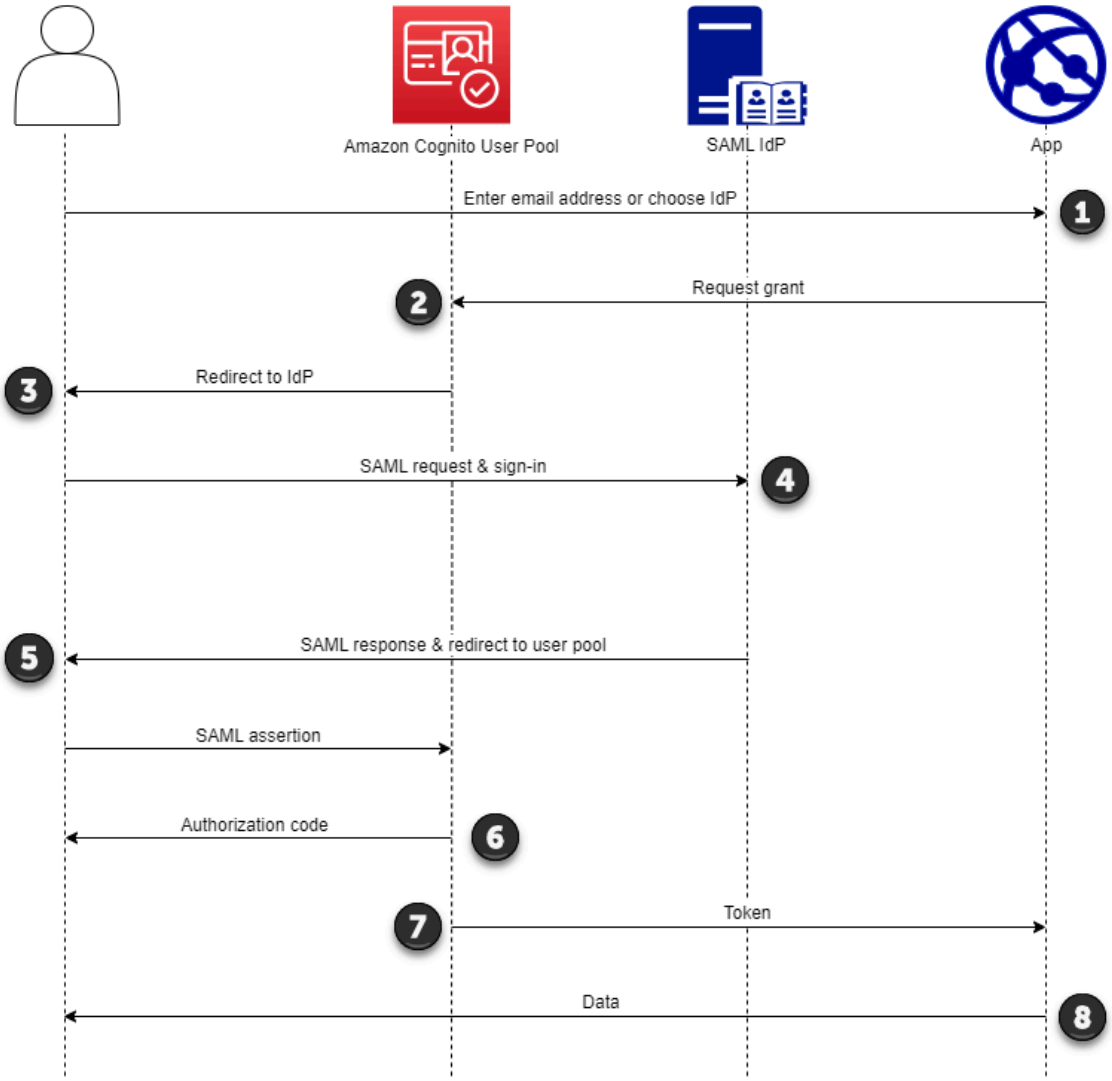
주제

- SP 시작 SAML 로그인 구현
- IdP 시작 SAML 로그인 구현

SP 시작 SAML 로그인 구현

가장 좋은 방법은 사용자 풀에 대한 서비스 제공업체 시작(SP 시작) 로그인을 구현하는 것입니다. Amazon Cognito는 사용자의 세션을 시작하고 이를 IdP 로 리디렉션합니다. 이 메서드를 사용하면 로그인 요청을 제시하는 사용자를 가장 잘 제어할 수 있습니다. 특정 조건에서 IdP가 시작한 로그인을 허용할 수도 있습니다.

다음 프로세스는 사용자가 SAML 제공업체를 통해 사용자 풀에 대한 SP 시작 로그인을 완료하는 방법을 보여줍니다.



1. 사용자가 로그인 페이지에 이메일 주소를 입력합니다. 사용자가 IdP로 리디렉션되는 방식을 확인하려면 맞춤형 애플리케이션에서 사용자의 이메일 주소를 수집하거나 웹 보기에서 관리형 로그인을 호출하면 됩니다.

   IdP 목록을 표시하거나 이메일 주소를 입력하라는 메시지를 표시하고 이를 SAML IdP의 식별자와 일치시키도록 관리형 로그인 페이지를 구성할 수 있습니다. 이메일 주소를 입력하라는 메시지를 표시하려면 관리형 로그인 브랜딩 스타일을 편집하고 파운데이션에서 인증 동작을 찾은 다음 제공업체 표시에서 표시 스타일을 도메인 검색 입력으로 설정합니다.
2. 앱은 사용자 풀 리디렉션 엔드포인트를 호출하고 앱에 해당하는 클라이언트 ID와 사용자에 해당하는 IdP ID로 세션을 요청합니다.
3. Amazon Cognito는 `AuthnRequest` 요소에서 선택적으로 서명된 SAML 요청과 함께 사용자를 IdP로 리디렉션합니다.
4. IdP는 대화형으로 또는 브라우저 쿠키에서 기억된 세션으로 사용자를 인증합니다.
5. IdP는 POST 페이로드에서 선택적으로 암호화된 SAML 어설션을 사용하여 사용자를 사용자 풀 SAML 응답 엔드포인트로 리디렉션합니다.

   > **Note**
   >
   > Amazon Cognito는 5분 이내에 응답을 받지 못하는 세션을 취소하고 사용자를 관리형 로그인으로 리디렉션합니다. 사용자에게 이 결과가 발생하면 `Something went wrong` 오류 메시지가 표시됩니다.
6. Amazon Cognito는 SAML 어설션을 확인하고 응답의 클레임에서 사용자 속성을 매핑한 후 내부적으로 사용자 풀에서 사용자 프로필을 생성하거나 업데이트합니다. 일반적으로 사용자 풀은 사용자의 브라우저 세션에 권한 부여 코드를 반환합니다.
7. 사용자가 앱에 인증 코드를 제시하면 앱은 코드를 JSON 웹 토큰(JWT)으로 교환합니다.
8. 앱은 사용자의 ID 토큰을 인증으로 수락 및 처리하고, 액세스 토큰을 사용하여 리소스에 대한 승인된 요청을 생성하고, 새로 고침 토큰을 저장합니다.

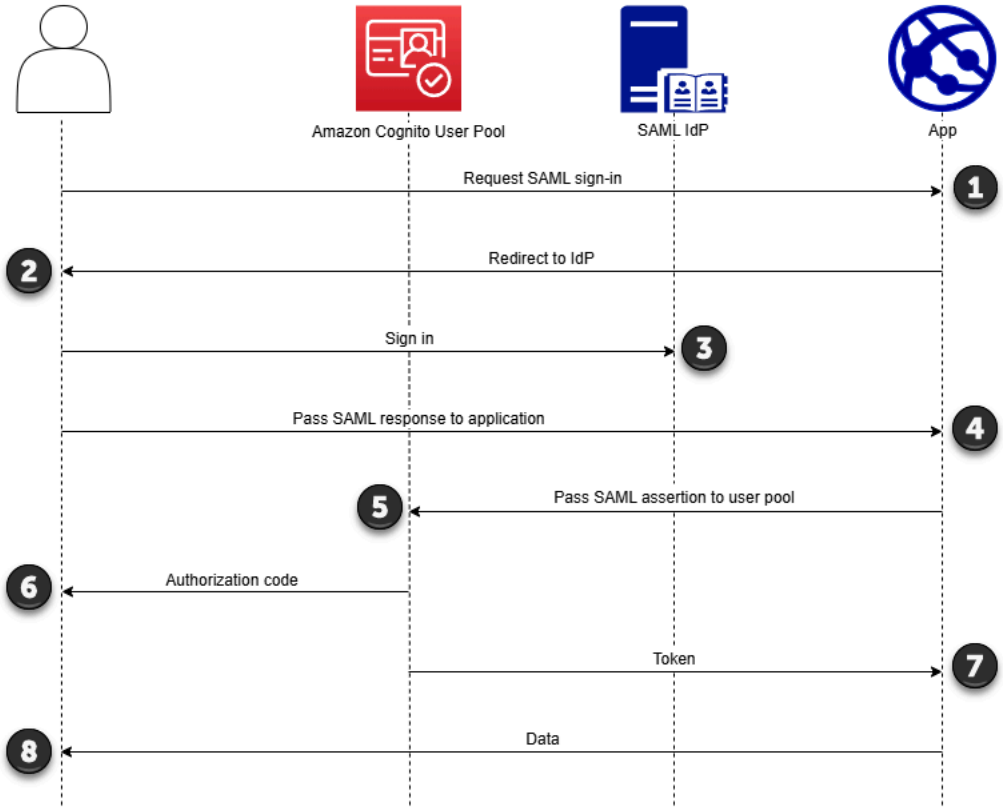
사용자가 인증하고 권한 부여 코드를 부여 받으면 사용자 풀은 ID, 액세스 및 새로 고침 토큰을 반환합니다. ID 토큰은 OIDC 기반 ID 관리를 위한 인증 개체입니다. 액세스 토큰은 OAuth 2.0 범위가 있는 권한 부여 개체입니다. 새로 고침 토큰은 사용자의 현재 토큰이 만료되었을 때 새 ID 및 액세스 토큰을 생성하는 개체입니다. 사용자 풀 앱 클라이언트에서 사용자 토큰의 기간을 구성할 수 있습니다.

새로 고침 토큰의 기간을 선택할 수도 있습니다. 사용자의 새로 고침 토큰이 만료되면 다시 로그인해야 합니다. SAML IdP를 통해 인증된 경우 사용자의 세션 기간은 IdP를 사용한 세션 만료가 아니라 토큰

만료로 설정됩니다. 앱은 각 사용자의 새로 고침 토큰을 저장하고 만료 시 세션을 갱신해야 합니다. 관리형 로그인은 1시간 동안 유효한 브라우저 쿠키에 사용자 세션을 유지합니다.

IdP 시작 SAML 로그인 구현

IdP 시작 SAML 2.0 로그인을 위해 ID 제공업체를 구성할 경우 권한 부여 엔드포인트에서 세션을 시작할 필요 없이 사용자 풀 도메인의 `saml2/idpresponse` 엔드포인트에 SAML 어설션을 제공할 수 있습니다. 이 구성의 사용자 풀은 요청된 앱 클라이언트가 지원하는 사용자 풀 외부 ID 제공업체로부터 IdP 시작 SAML 어설션을 수락합니다.



1. 사용자가 애플리케이션으로 SAML 로그인을 요청합니다.
2. 애플리케이션은 브라우저를 호출하거나 사용자를 SAML 제공업체의 로그인 페이지로 리디렉션합니다.
3. IdP는 대화형으로 또는 브라우저 쿠키에서 기억된 세션으로 사용자를 인증합니다.
4. IdP는 POST 본문에서 SAML 어설션 또는 응답을 사용하여 사용자를 애플리케이션으로 리디렉션합니다.
5. 애플리케이션은 요청의 POST 본문에 SAML 어설션을 사용자 풀 `saml2/idpresponse` 엔드포인트에 추가합니다.

6. Amazon Cognito는 사용자에게 승인 코드를 발급합니다.
7. 사용자가 앱에 인증 코드를 제시하면 앱은 코드를 JSON 웹 토큰(JWT)으로 교환합니다.
8. 애플리케이션은 사용자의 ID 토큰을 인증으로 수락 및 처리하고, 액세스 토큰을 사용하여 리소스에 대한 승인된 요청을 생성하고, 새로 고침 토큰을 저장합니다.

다음 단계에서는 IdP 시작 SAML 2.0 제공업체를 구성하고 로그인하는 전체 프로세스를 설명합니다.

1. 사용자 풀 및 앱 클라이언트를 생성하거나 지정합니다.
2. 사용자 풀에 SAML 2.0 IdP를 생성합니다.
3. IdP 시작을 지원하도록 IdP를 구성합니다. IdP 시작 SAML은 다른 SSO 제공업체에 적용되지 않는 보안 고려 사항을 도입합니다. 따라서 사용자 풀 자체를 포함하여 IdP 시작 로그인이 포함된 SAML 제공업체를 사용하는 앱 클라이언트에는 비SAML IdP 추가할 수 없습니다.
4. IdP 시작 SAML 제공업체를 사용자 풀의 앱 클라이언트와 연결합니다.
5. 사용자를 SAML IdP의 로그인 페이지로 안내하고 SAML 어설션을 검색합니다.
6. SAML 어설션을 사용하여 사용자를 사용자 풀 `saml2/idpresponse` 엔드포인트로 안내합니다.
7. JSON 웹 토큰(JWT)을 받습니다.

사용자 풀에서 원치 않는 SAML 어설션을 수락하려면 앱 보안에 미치는 영향을 고려해야 합니다. IdP에서 시작한 요청을 수락할 때 요청 스푸핑 및 CSRF 시도가 발생할 수 있습니다. 사용자 풀은 IdP에서 시작한 로그인 세션을 확인할 수 없지만 Amazon Cognito는 요청 파라미터와 SAML 어설션을 검증합니다.

또한 SAML 어설션에는 `InResponseTo` 클레임이 포함되어서는 안 되며 이전 6분 이내에 발급되었어야 합니다.

IdP가 시작한 SAML을 사용하여 요청을 사용자의 `/saml2/idpresponse`에 제출해야 합니다. SP 시작 및 관리형 로그인 권한 부여 요청의 경우 요청된 앱 클라이언트, 범위, 리디렉션 URI 및 기타 세부 정보를 식별하는 파라미터를 `HTTP GET` 요청의 쿼리 문자열 파라미터로 제공해야 합니다. 그러나 IdP 시작 SAML 어설션의 경우 요청의 세부 정보는 `HTTP POST` 요청 본문의 `RelayState` 파라미터로 형식이 지정되어야 합니다. 요청 본문에 SAML 어설션도 `SAMLResponse` 파라미터로 포함되어야 합니다.

다음은 IdP가 시작한 SAML 제공업체에 대한 요청 및 응답의 예시입니다.

```
POST /saml2/idpresponse HTTP/1.1
User-Agent: USER_AGENT
```

```
Accept: */*
Host: example.auth.us-east-1.amazoncognito.com
Content-Type: application/x-www-form-urlencoded

SAMLResponse=[Base64-encoded SAML assertion]&RelayState=identity_provider
%3DMySAMLIdP%26client_id%3D1example23456789%26redirect_uri%3Dhttps%3A%2F
%2Fwww.example.com%26response_type%3Dcode%26scope%3Demail%2Bopenid%2Bphone

HTTP/1.1 302 Found
Date: Wed, 06 Dec 2023 00:15:29 GMT
Content-Length: 0
x-amz-cognito-request-id: 8aba6eb5-fb54-4bc6-9368-c3878434f0fb
Location: https://www.example.com?code=[Authorization code]
```

AWS Management Console

IdP 시작 SAML에 대한 IdP를 구성하는 방법

1. 사용자 풀 , 앱 클라이언트 및 SAML ID 제공업체를 생성합니다.
2. 연결된 경우 앱 클라이언트에서 모든 소셜 및 OIDC ID 제공업체의 연결을 해제합니다.
3. 사용자 풀의 소셜 및 외부 제공업체 메뉴로 이동합니다.
4. SAML 제공업체를 편집하거나 추가합니다.
5. IdP 시작 SAML 로그인에서 SP 시작 및 IdP 시작 SAML 어설션 수락을 선택합니다.
6. 변경 사항 저장을 선택합니다.

API/CLI

IdP 시작 SAML에 대한 IdP를 구성하는 방법

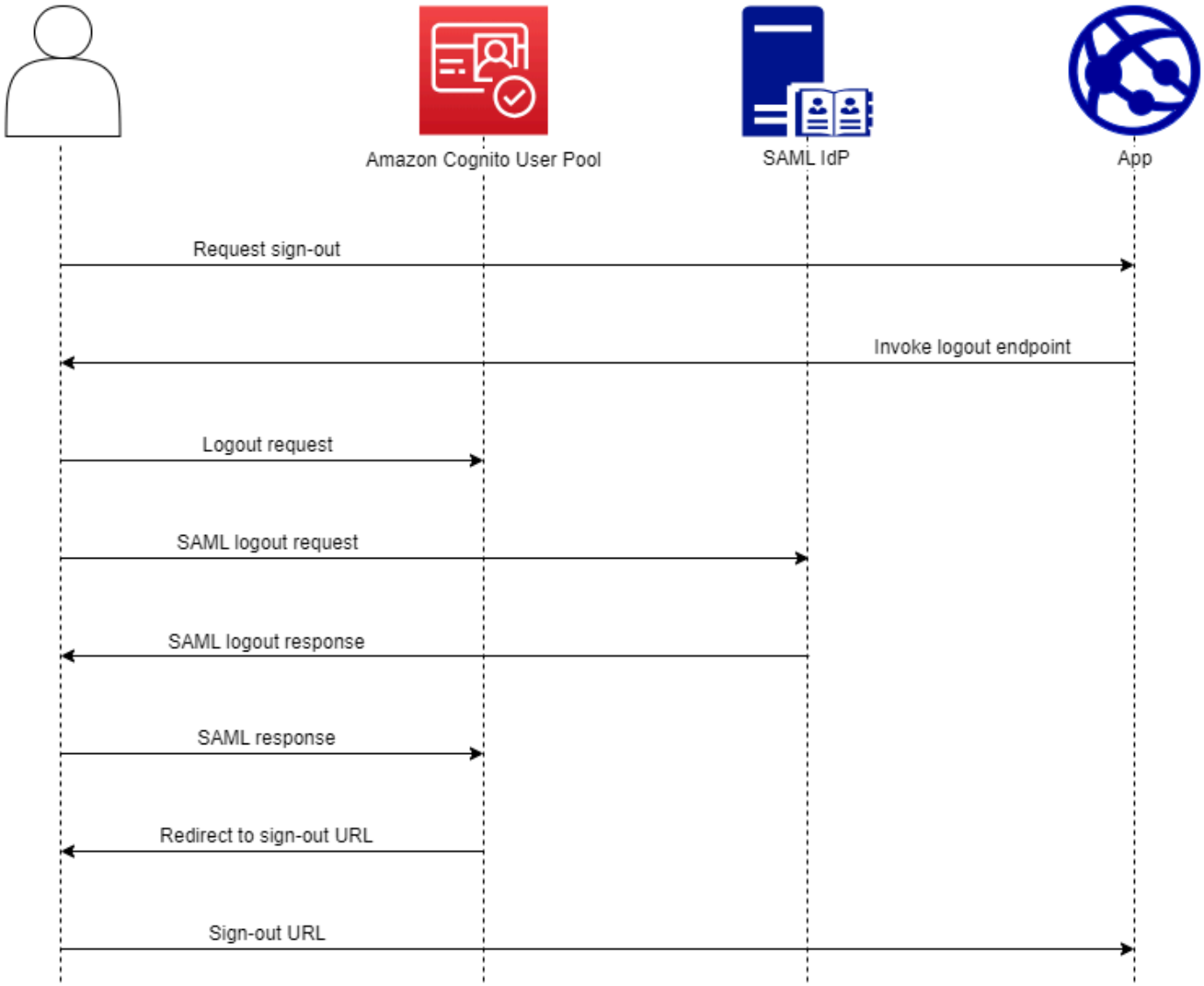
CreateIdentityProvider 또는 UpdateIdentityProvider API 요청의 `IDPInit` 파라미터로 IdP 시작 SAML을 구성합니다. 다음은 IdP 시작 SAML을 지원하는 IdP의 `ProviderDetails` 예시입니다.

```
"ProviderDetails": {
      "MetadataURL" : "https://myidp.example.com/saml/metadata",
      "IDPSignout" : "true",
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

단일 로그아웃으로 SAML 사용자 로그아웃

Amazon Cognito는 SAML 2.0 단일 로그아웃(SLO)을 지원합니다. SLO를 사용하면 애플리케이션이 사용자 풀에서 로그아웃할 때 SAML ID 제공업체(IdPs)에서 사용자를 로그아웃할 수 있습니다. 이렇게 하면 사용자가 애플리케이션에 다시 로그인하려는 경우 SAML IdP 로 인증해야 합니다. 그렇지 않으면 자격 증명을 제공할 필요 없이 애플리케이션으로 전달하는 IdP 또는 사용자 풀 브라우저 쿠키가 있을 수 있습니다.

로그아웃 흐름을 지원하도록 SAML IdP를 구성하면 Amazon Cognito는 서명된 SAML 로그아웃 요청을 사용하여 사용자를 IdP로 리디렉션합니다. Amazon Cognito는 IdP 메타데이터의 `SingleLogoutService` URL에서 리디렉션 위치를 결정합니다. Amazon Cognito는 사용자 풀 서명 인증서로 로그아웃 요청에 서명합니다.



SAML 세션이 있는 사용자를 사용자 풀 `/logout` 엔드포인트로 연결하면 Amazon Cognito는 다음 요청을 사용하여 SAML 사용자를 IdP 메타데이터에 지정된 SLO 엔드포인트로 리디렉션합니다.

```
https://[SingleLogoutService endpoint]?
SAMLRequest=[encoded SAML request]&
RelayState=[RelayState]&
SigAlg=http://www.w3.org/2001/04/xmldsig-more#rsa-sha256&
Signature=[User pool RSA signature]
```

그런 다음 사용자는 IdP에서 `LogoutResponse`를 사용하여 `saml2/logout` 엔드포인트로 돌아갑니다. IdP는 `HTTP POST` 요청으로 `LogoutResponse`를 전송해야 합니다. 그런 다음 Amazon Cognito는 초기 로그아웃 요청에서 리디렉션 대상으로 리디렉션합니다.

SAML 제공업체는 두 개 이상의 `AuthnStatement`가 포함된 `LogoutResponse`를 보낼 수 있습니다. 이 유형의 응답에서 첫 번째 `AuthnStatement`의 `sessionIndex`는 원래 사용자를 인증한 SAML 응답의 `sessionIndex`와 일치해야 합니다. `sessionIndex`가 다른 `AuthnStatement`에 있는 경우 Amazon Cognito는 세션을 인식하지 못하며 사용자는 로그아웃되지 않습니다.

AWS Management Console

SAML 로그아웃을 구성하는 방법

1. 사용자 풀 , 앱 클라이언트 및 SAML IdP를 생성합니다.
2. SAML ID 제공업체를 생성하거나 편집할 때 ID 제공업체 정보에서 로그아웃 흐름 추가라는 제목의 확인란을 선택합니다.
3. 사용자 풀의 소셜 및 외부 제공업체 메뉴에서 IdP를 선택하고 서명 인증서를 찾습니다.
4. .crt로 다운로드를 선택합니다.
5. SAML 단일 로그아웃 및 요청 서명을 지원하도록 SAML 제공업체를 구성하고 사용자 풀 서명 인증서를 업로드합니다. IdP는 사용자 풀 도메인의 `/saml2/logout`으로 리디렉션해야 합니다.

API/CLI

SAML 로그아웃을 구성하는 방법

CreateIdentityProvider 또는 UpdateIdentityProvider API 요청의 `IDPSignout` 파라미터를 사용하여 단일 로그아웃을 구성합니다. 다음은 SAML 단일 로그아웃을 지원하는 IdP의 `ProviderDetails` 예시입니다.

```
"ProviderDetails": {
      "MetadataURL" : "https://myidp.example.com/saml/metadata",
      "IDPSignout" : "true",,
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

SAML 서명 및 암호화

SAML 2.0 로그인은 애플리케이션 사용자가 인증 흐름에서 요청과 응답을 전달하는 주체가 되어 구축됩니다. 사용자가 전송 중에 이러한 SAML 문서를 읽거나 수정하지 않도록 해야 할 수도 있습니다. 이렇게 하려면 사용자 풀의 SAML ID 제공업체(IdPs)에 SAML 서명 및 암호화를 추가하십시오. SAML 서명으로 사용자 풀은 SAML 로그인 및 로그아웃 요청에 서명을 추가합니다. 사용자 풀 퍼블릭 키를 사용하면 IdP가 수정되지 않은 SAML 요청을 수신하고 있는지 확인할 수 있습니다. 그런 다음 IdP가 응답하고 SAML 어설션을 사용자의 브라우저 세션에 전달하면 IdP는 사용자가 자신의 속성과 권한을 검사할 수 없도록 해당 응답을 암호화할 수 있습니다.

SAML 서명 및 암호화를 사용하면 사용자 풀 SAML 작업 중 모든 암호화 작업은 Amazon Cognito가 생성하는 사용자 풀 제공 키를 사용하여 서명 및 사이퍼텍스트를 생성해야 합니다. 현재 외부 키를 사용하여 요청에 서명하거나 암호화된 어설션을 수락하도록 사용자 풀을 구성할 수 없습니다.

> **Note**
>
> 사용자 풀 인증서는 10년 동안 유효합니다. Amazon Cognito는 1년에 한 번 사용자 풀에 대한 새 서명 및 암호화 인증서를 생성합니다. Amazon Cognito는 서명 인증서를 요청할 때 가장 최근의 인증서를 반환하고 가장 최근의 서명 인증서로 요청에 서명합니다. IdP는 만료되지 않은 모든 사용자 풀 암호화 인증서로 SAML 어설션을 암호화할 수 있습니다. 이전 인증서는 전체 기간 동안 계속 유효하며 퍼블릭 키는 인증서 간에 변경되지 않습니다. 가장 좋은 방법은 매년 제공업체 구성에서 인증서를 업데이트하는 것입니다.

주제

- IdP에서 암호화된 SAML 응답 수락
- SAML 요청 서명

IdP에서 암호화된 SAML 응답 수락

Amazon Cognito와 IdP는 사용자가 로그인하고 로그아웃할 때 SAML 응답에서 기밀성을 설정할 수 있습니다. Amazon Cognito는 사용자 풀에서 구성한 각 외부 SAML 제공업체에 퍼블릭 프라이빗 RSA 키 페어와 인증서를 할당합니다. 사용자 풀 SAML 제공업체에 대한 응답 암호화를 활성화할 경우 암호화된 SAML 응답을 지원하는 IdP에 인증서를 업로드해야 합니다. IdP가 제공된 키로 모든 SAML 어설션 암호화를 시작하기 전에는 SAML IdP에 대한 사용자 풀 연결이 작동하지 않습니다.

다음은 암호화된 SAML 로그인의 흐름에 대한 개요입니다.

1. 사용자가 로그인을 시작하고 SAML IdP를 선택합니다.
2. 사용자 풀 권한 부여 엔드포인트은 SAML 로그인 요청을 통해 사용자를 SAML IdP로 리디렉션합니다. 사용자 풀은 선택적으로 이 요청에 IdP의 무결성 검증을 가능하게 하는 서명을 추가할 수 있습니다. SAML 요청에 서명하려면 사용자 풀이 서명 인증서의 퍼블릭 키로 서명한 요청을 수락하도록 IdP를 구성해야 합니다.
3. SAML IdP는 사용자에 서명하고 SAML 응답을 생성합니다. IdP는 퍼블릭 키를 사용하여 응답을 암호화하고 사용자를 사용자 풀 `/saml2/idpresponse` 엔드포인트로 리디렉션합니다. IdP는 SAML 2.0 사양에 정의된 대로 응답을 암호화해야 합니다. 자세한 내용은 OASIS SAML(Security Assertion Markup Language) V2.0에 대한 어설션 및 프로토콜의 `Element <EncryptedAssertion>` 섹션을 참조하세요.
4. 사용자 풀은 프라이빗 키를 사용하여 SAML 응답의 사이퍼텍스트를 해독하고 사용자에 로그인합니다.

> **Important**
>
> 사용자 풀에서 SAML IdP에 대한 응답 암호화를 활성화할 때 IdP는 제공업체 전용 퍼블릭 키로 모든 응답을 암호화해야 합니다. Amazon Cognito는 암호화를 지원하도록 구성한 SAML 외부 IdP에서 암호화되지 않은 SAML 응답을 수락하지 않습니다.

사용자 풀에 있는 모든 외부 SAML IdP는 응답 암호화를 지원할 수 있으며, 각 IdP는 자체 키 페어를 받습니다.

AWS Management Console

SAML 응답 암호화를 구성하는 방법

1. 사용자 풀 , 앱 클라이언트 및 SAML IdP를 생성합니다.

2. SAML ID 제공업체를 생성하거나 편집할 때 서명 요청 및 응답 암호화에서 이 제공업체로부터 암호화된 SAML 어설션 필요라는 제목의 확인란을 선택합니다.
3. 사용자 풀의 소셜 및 외부 제공업체 메뉴에서 SAML IdP를 선택하고 암호화 인증서 보기를 선택합니다.
4. .crt로 다운로드를 선택하고 다운로드한 파일을 SAML IdP에 제공합니다. 인증서의 키를 사용하여 SAML 응답을 암호화하도록 SAML IdP를 구성합니다.

API/CLI

SAML 응답 암호화를 구성하는 방법

CreateIdentityProvider 또는 UpdateIdentityProvider API 요청의 `EncryptedResponses` 파라미터를 사용하여 응답 암호화를 구성합니다. 다음은 요청 서명을 지원하는 IdP의 `ProviderDetails` 예시입니다.

```
"ProviderDetails": {
      "MetadataURL" : "https://myidp.example.com/saml/metadata",
      "IDPSignout" : "true",
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

사용자 풀에서 암호화 인증서를 가져오려면 DescribeIdentityProvider API 요청을 수행하고 응답 파라미터 `ProviderDetails`에서 `ActiveEncryptionCertificate` 값을 검색합니다. 이 인증서를 저장하고 사용자 풀의 로그인 요청에 대한 암호화 인증서로 IdP에 제공합니다.

SAML 요청 서명

IdP에 대한 SAML 2.0 요청의 무결성을 증명할 수 있는 기능은 Amazon Cognito SP 시작 SAML 로그인이 가진 보안 이점입니다. 도메인이 있는 각 사용자 풀은 사용자 풀 X.509 서명 인증서를 받습니다. 이 인증서의 퍼블릭 키를 사용하면 사용자 풀은 사용자가 SAML IdP 를 선택할 때 사용자 풀이 생성하는 로그아웃 요청에 암호화 서명을 적용합니다. 선택적으로 SAML 로그인 요청에 서명하도록 앱 클라이언트를 구성할 수 있습니다. SAML 요청에 서명하면 IdP는 요청의 XML 메타데이터에 있는 서명이 사용자가 제공한 사용자 풀 인증서의 퍼블릭 키와 일치하는지 확인할 수 있습니다.

AWS Management Console

SAML 요청 서명을 구성하는 방법

1. 사용자 풀 , 앱 클라이언트 및 SAML IdP를 생성합니다.
2. SAML ID 제공업체를 생성하거나 편집할 때 요청 서명 및 응답 암호화에서 이 제공업체에 대한 SAML 요청 서명이라는 제목의 확인란을 선택합니다.
3. 사용자 풀의 소셜 및 외부 제공업체 메뉴에서 서명 인증서 보기를 선택합니다.
4. .crt로 다운로드를 선택하고 다운로드한 파일을 SAML IdP에 제공합니다. 수신 SAML 요청의 서명을 확인하도록 SAML IdP를 구성합니다.

API/CLI

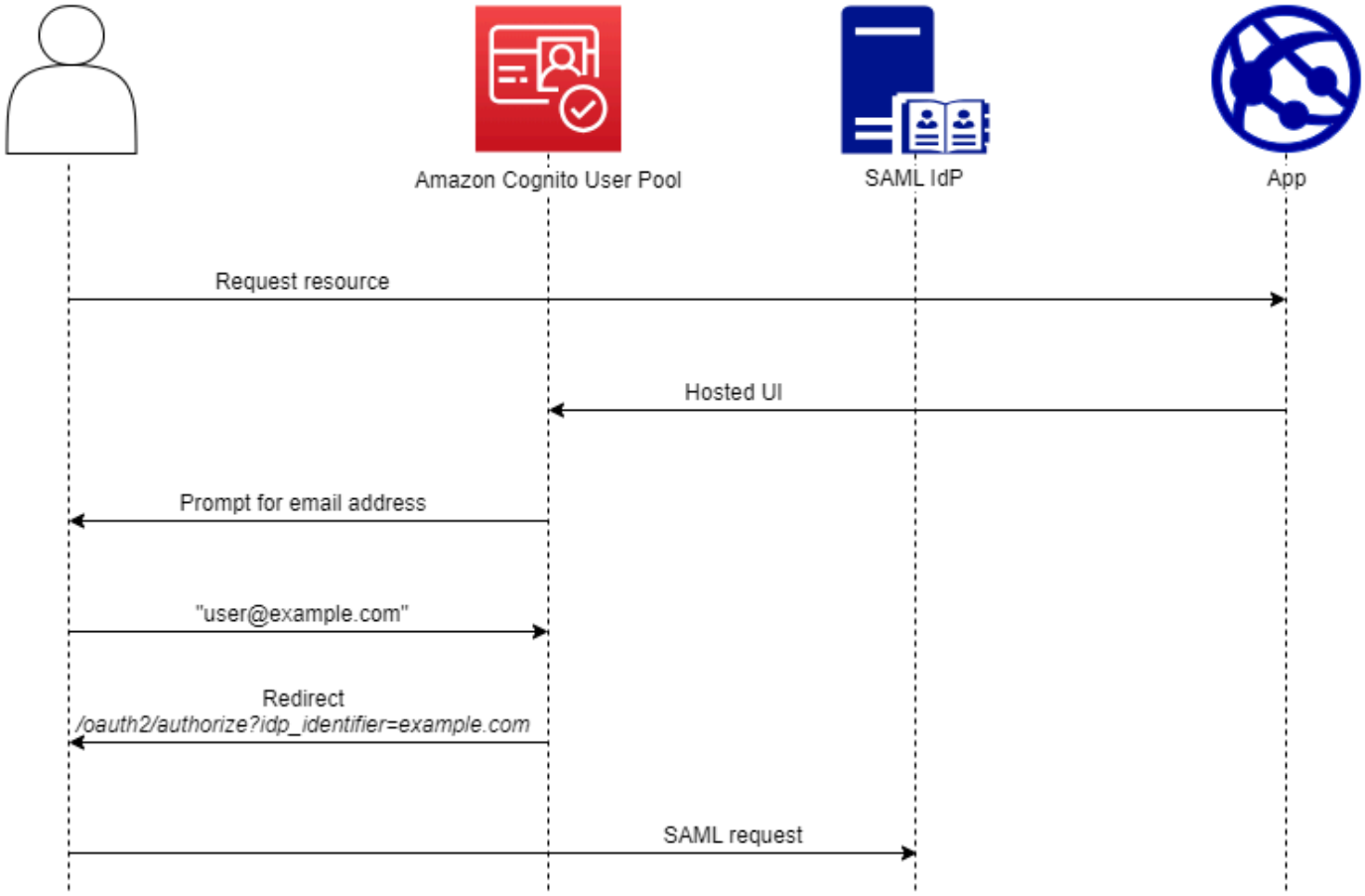
SAML 요청 서명을 구성하는 방법

CreateIdentityProvider 또는 UpdateIdentityProvider API 요청의 `RequestSigningAlgorithm` 파라미터를 사용하여 요청 서명을 구성합니다. 다음은 요청 서명을 지원하는 IdP의 `ProviderDetails` 예시입니다.

```
"ProviderDetails": {
      "MetadataURL" : "https://myidp.example.com/saml/metadata",
      "IDPSignout" : "true",
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

SAML ID 제공업체 이름 및 식별자

SAML ID 제공업체(IdPs)의 이름을 지정하고 IdP 식별자를 할당하면 SP가 시작한 로그인 및 로그아웃 요청의 흐름을 해당 제공업체에 자동화할 수 있습니다. 제공업체 이름의 문자열 제약 조건에 대한 자세한 내용은 CreateIdentityProvider의 `ProviderName` 속성을 참조하세요.



또한 SAML 제공업체에 대해 최대 50개의 식별자를 선택할 수도 있습니다. 식별자는 사용자 풀의 IdP에 대한 표시 이름이며 사용자 풀 내에서 고유해야 합니다. SAML 식별자가 사용자의 이메일 도메인과 일치하는 경우 관리형 로그인은 각 사용자의 이메일 주소를 요청하고, 이메일 주소의 도메인을 평가한 다음, 해당 도메인에 해당하는 IdP로 리디렉션합니다. 같은 회사가 여러 도메인을 소유할 수 있기 때문에 단일 IdP는 여러 식별자를 가질 수 있습니다.

이메일 도메인 식별자를 사용하든 사용하지 않든 멀티 테넌트 앱에서 식별자를 사용하여 사용자를 올바른 IdP로 리디렉션할 수 있습니다. 관리형 로그인을 완전히 우회하려는 경우 권한 부여 엔드포인트를 통해 IdP로 직접 리디렉션되도록 사용자에게 제공하는 링크를 사용자 지정할 수 있습니다. 식별자를 사용하여 사용자를 로그인하고 IdP로 리디렉션하려면 초기 권한 부여 요청의 요청 파라미터에 `idp_identifier=`*`myidp.example.com`* 형식의 식별자를 포함합니다.

사용자를 IdP로 전달하는 또 다른 방법은 다음 URL 형식의 IdP 이름으로 `identity_provider` 파라미터를 채우는 것입니다.

```
https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
```

```
response_type=code&
identity_provider=MySAMLIdP&
client_id=1example23456789&
redirect_uri=https://www.example.com
```

사용자가 SAML IdP로 로그인하면 IdP는 HTTP POST 본문의 SAML 응답과 함께 /saml2/idpresponse 엔드포인트로 리디렉션합니다. Amazon Cognito는 SAML 어설션을 처리하고 응답의 클레임이 기대치를 충족하는 경우 앱 클라이언트 콜백 URL로 리디렉션합니다. 사용자가 이러한 방식으로 인증을 완료한 후에는 IdP 및 앱에 대한 웹페이지와만 상호 작용하게 됩니다.

도메인 형식의 IdP 식별자를 사용하면 관리형 로그인은 로그인 시 이메일 주소를 요청한 다음 이메일 도메인이 IdP 식별자와 일치하면 사용자를 IdP의 로그인 페이지로 리디렉션합니다. 예를 들어 서로 다른 두 회사의 직원이 로그인해야 하는 앱을 빌드한다고 가정해 보겠습니다. 첫 번째 회사인 AnyCompany A는 exampleA.com 및 exampleA.co.uk를 소유합니다. 두 번째 회사인 AnyCompany B는 exampleB.com을 소유합니다. 이 예에서는 회사마다 하나씩, 총 두 개의 IdP를 설정했습니다.

- IdP A에는 exampleA.com 및 exampleA.co.uk 식별자를 정의합니다.
- IdP B에는 exampleB.com 식별자를 정의합니다.

앱에서 앱 클라이언트에 관리형 로그인을 호출하여 각 사용자에게 이메일 주소를 입력하라는 메시지를 표시합니다. Amazon Cognito는 이메일 주소에서 도메인을 추출하고, 도메인 식별자를 사용하여 도메인을 IdP와 상호 연결한 다음 idp_identifier 요청 파라미터가 포함된 권한 부여 엔드포인트에 대한 요청과 함께 사용자를 올바른 IdP로 리디렉션합니다. 예를 들어 사용자가 bob@exampleA.co.uk를 입력하면 상호 작용하는 다음 페이지는 https://auth.exampleA.co.uk/sso/saml의 IdP 로그인 페이지입니다.

동일한 로직을 독립적으로 구현할 수도 있습니다. 앱에서 사용자 입력을 수집하고 사용자 자신의 로직에 따라 올바른 IdP와 상호 연관시키는 사용자 지정 양식을 구축할 수 있습니다. 각 앱 테넌트에 대해 사용자 지정 포털을 생성할 수 있으며, 각 링크는 요청 파라미터에 테넌트의 식별자를 사용하여 권한 부여 엔드포인트에 연결됩니다.

이메일 주소를 수집하고 관리형 로그인에서 도메인을 구문 분석하려면 앱 클라이언트에 할당한 각 SAML IdP에 하나 이상의 식별자를 할당합니다. 기본적으로 관리형 로그인 화면에는 앱 클라이언트에 할당된 각 IdP에 대한 버튼이 표시됩니다. 하지만 식별자를 성공적으로 할당한 경우에는 클래식 호스팅 UI 로그인 페이지가 다음 이미지와 같이 나타납니다.
로컬 사용자 로그인 및 페더레이션 사용자에게 이메일 주소를 입력하라는 메시지를 표시하는 Amazon Cognito 관리형 로그인 페이지입니다.

> **Note**
>
> 클래식 호스팅 UI에서 IdP에 식별자를 할당하면 앱 클라이언트의 로그인 페이지에 이메일 주소가 자동으로 표시됩니다. 관리형 로그인 환경에서는 브랜딩 편집기에서 이 동작을 활성화해야 합니다. 인증 동작 설정 범주의 제공업체 표시 제목에서 도메인 검색 입력을 선택합니다.

관리형 로그인의 도메인 구문 분석에는 도메인을 IdP 식별자로 사용해야 합니다. 앱 클라이언트의 각 SAML IdP에 유형과 관계없이 식별자를 할당하면 해당 앱의 관리형 로그인에 더 이상 IdP 선택 버튼이 표시되지 않습니다. 이메일 구문 분석 또는 사용자 지정 로직을 사용하여 리디렉션을 생성하려는 경우 SAML에 대한 IdP 식별자를 추가합니다. 자동 리디렉션을 생성하고 관리형 로그인에 IdP 목록을 표시하려면 식별자를 할당하지 말고 권한 부여 요청에 `identity_provider` 요청 파라미터를 사용합니다.

- 앱 클라이언트에 SAML IdP 하나만 할당하는 경우 관리형 로그인 페이지에 해당 IdP로 로그인하는 버튼이 표시됩니다.
- 앱 클라이언트에 대해 활성화한 모든 SAML IdP에 식별자를 할당하는 경우 이메일 주소를 입력하라는 사용자 입력 프롬프트가 관리형 로그인 페이지에 표시됩니다.
- 여러 IdP가 있고 모든 IdP에 식별자를 할당하지 않은 경우 관리형 로그인 페이지에는 할당된 각 IdP로 로그인할 수 있는 버튼이 표시됩니다.
- IdP에 식별자를 할당했고 선택한 IdP 버튼을 관리형 로그인에 표시하고 싶다면, 식별자가 없는 새 IdP를 앱 클라이언트에 추가하거나 새 앱 클라이언트를 생성합니다. 기존 IdP를 삭제하고 식별자 없이 다시 추가할 수도 있습니다. 새 IdP를 생성하면 SAML 사용자가 새 사용자 프로필을 생성합니다. 이러한 활성 사용자 중복은 IdP 구성을 변경하는 달의 청구에 영향을 미칠 수 있습니다.

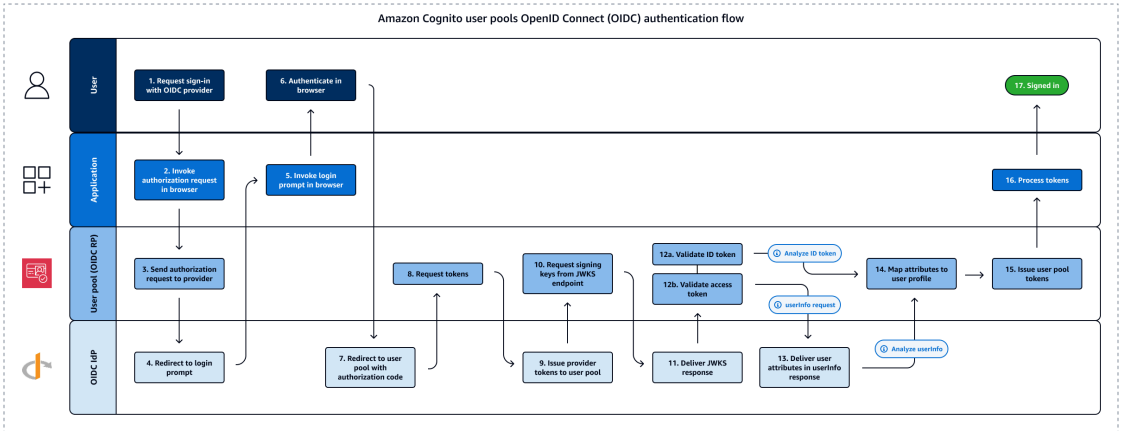
IdP 설정에 대한 자세한 내용은 사용자 풀의 자격 증명 공급자 구성 섹션을 참조하세요.

사용자 풀에서 OIDC ID 제공업체 사용

사용자는 OIDC(OpenID Connect) ID 제공업체(idP)의 기존 계정을 사용하여 애플리케이션에 로그인할 수 있습니다. OIDC 제공업체를 사용하면 독립적인 Single Sign-On 시스템의 사용자가 기존 자격 증명을 제공하는 동안 애플리케이션은 사용자 풀의 공유 형식으로 OIDC 토큰을 받을 수 있습니다. OIDC IdP를 구성하려면 사용자 풀을 RP로 처리하도록 IdP를 설정하고 사용자 풀을 IdP로 처리하도록 애플리케이션을 구성합니다. Amazon Cognito는 여러 OIDC IdPs와 애플리케이션 간의 중간 단계 역할을 합니다. 사용자 풀은 제공업체가 사용자 풀에 직접 전달하는 ID 및 액세스 토큰의 클레임에 속성 매핑 규칙을 적용합니다. 그런 다음 Amazon Cognito는 매핑된 사용자 속성과 Lambda 트리거를 사용하여 인증 흐름을 추가로 조정한 내용을 기반으로 새 토큰을 발급합니다.

OIDC IdP로 로그인하는 사용자는 사용자 풀 애플리케이션에 액세스하기 위해 새 자격 증명 또는 정보를 제공할 필요가 없습니다. 애플리케이션은 로그인을 위해 사용자 풀을 백그라운드에서 애플리케이션의 토큰 형식을 표준화하는 도구로 사용하여 IdP로 자동으로 리디렉션할 수 있습니다. IdP 리디렉션에 대한 자세한 내용은 권한 부여 엔드포인트 섹션을 참조하세요.

다른 타사 ID 제공업체와 마찬가지로 OIDC 제공업체에 애플리케이션을 등록하고 사용자 풀에 연결하려는 IdP 애플리케이션에 대한 정보를 얻어야 합니다. 사용자 풀 OIDC IdP에는 클라이언트 ID, 클라이언트 보안 암호, 요청하려는 범위 및 제공업체 서비스 엔드포인트에 대한 정보가 필요합니다. 사용자 풀은 검색 엔드포인트에서 제공업체 OIDC 엔드포인트를 검색하거나 수동으로 입력할 수 있습니다. 또한 제공업체 ID 토큰을 검사하고 IdP와 사용자 풀의 속성 간에 속성 매핑을 생성해야 합니다.



이 인증 흐름에 대한 자세한 내용은 OIDC 사용자 풀 IdP 인증 흐름 섹션을 참조하세요.

> **Note**
>
> 서드 파티(페더레이션)를 통한 로그인을 Amazon Cognito 사용자 풀에서 사용할 수 있습니다. 이 기능은 Amazon Cognito ID 풀을 사용하는 OIDC 페더레이션과 무관합니다.

AWS Management Console,를 통해 AWS CLI또는 사용자 풀 API 메서드 CreateIdentityProvider를 사용하여 사용자 풀에 OIDC IdP를 추가할 수 있습니다.

주제

- 사전 조건
- OIDC IdP에 애플리케이션 등록
- 사용자 풀에 OIDC IdP 추가
- OIDC IdP 구성 테스트

- OIDC 사용자 풀 IdP 인증 흐름
- 페더레이션 사용자를 위한 단계적 인증

사전 조건

시작하려면 다음이 필요합니다.

- 앱 클라이언트와 사용자 풀 도메인이 있는 사용자 풀. 자세한 내용은 사용자 풀 생성을 참조하세요.
- 다음과 같은 구성의 OIDC IdP:
  - `client_secret_post` 클라이언트 인증을 지원합니다. Amazon Cognito는 IdP에 대한 OIDC 검색 엔드포인트에서 `token_endpoint_auth_methods_supported` 신청을 확인하지 않습니다. Amazon Cognito는 `client_secret_basic` 클라이언트 인증을 지원하지 않습니다. 클라이언트 인증에 대한 자세한 내용은 OpenID Connect 설명서에서 클라이언트 인증을 참조하세요.
  - `openid_configuration`, `userInfo` 및 `jwks_uri`와 같은 OIDC 엔드포인트에만 HTTPS를 사용합니다.
  - OIDC 엔드포인트에만 TCP 포트 80 및 443을 사용합니다.
  - HMAC-SHA 또는 RSA 알고리즘을 사용하여 ID 토큰에만 서명합니다.
  - 키 ID `kid` 클레임을 `jwks_uri`에 게시하며 토큰에 `kid` 클레임을 포함합니다.
  - 유효한 루트 CA 신뢰 체인이 있는 만료되지 않은 퍼블릭 키를 제공합니다.

OIDC IdP에 애플리케이션 등록

OIDC IdP를 사용자 풀 구성에 추가하고 앱 클라이언트에 할당하기 전에 IdP에서 OIDC 클라이언트 애플리케이션을 설정합니다. 사용자 풀은 IdP로 인증을 관리하는 신뢰 당사자 애플리케이션입니다.

OIDC IdP로 등록하려면

1. OIDC IdP에서 개발자 계정을 생성합니다.

   OIDC IdP 링크

OIDC IdP	설치 방법	OIDC 검색 URL
Salesforce	OpenID Connect ID 제공업체로서의 Salesforce	`https://MyDomainName.my.salesforce.com/.well-known/openid-configuration`

OIDC IdP	설치 방법	OIDC 검색 URL
OneLogin	OIDC 지원 앱 연결	`https://your-domain.onelogin.com/oidc/2/.well-known/openid-configuration`
JumpCloud	OIDC를 사용한 SSO	`https://oauth.id.jumpcloud.com/.well-known/openid-configuration`
Okta	Okta 자격 증명 공급자 설치	`https://Your Okta subdomain .okta.com/.well-known/openid-configuration`
Microsoft Entra ID	Microsoft ID 플랫폼의 OpenID Connect	`https://login.microsoftonline.com/ {tenant}/v2.0` `tenant`의 값에는 테넌트 ID, `common`, `organizations` 또는 `consumers` 가 포함될 수 있습니다.

2. `/oauth2/idpresponse` 엔드포인트가 있는 사용자 풀 도메인 URL을 OIDC IdP에 등록합니다. 그래야만 사용자를 인증할 때 나중에 Amazon Cognito에서 OIDC IdP를 수락합니다.

```
https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/idpresponse
```

3. 사용자 디렉터리가 사용자 풀과 공유할 범위를 선택합니다. OIDC IdP가 사용자 정보를 제공하려면 범위 openid가 필요합니다. `email` 범위는 `email` 및 `email_verified` 클레임에 대한 액세스를 부여하는 데 필요합니다. OIDC 사양의 추가 범위는 모든 사용자 속성의 경우 `profile`이며 `phone_number` 및 `phone_number_verified`의 경우 `phone`입니다.
4. OIDC IdP에서 클라이언트 ID와 클라이언트 보안 암호를 제공합니다. 이러한 값을 기록해 두고 나중에 사용자 풀에 추가하는 OIDC IdP의 구성에 추가합니다.

예: 사용자 풀에서 Salesforce를 OIDC IdP로 사용

Salesforce 같은 OIDC 호환 IdP와 사용자 풀 사이에 신뢰를 설정해야 할 때 OIDC IdP를 사용합니다.

1. Salesforce 개발자 웹사이트에서 계정을 생성합니다.
2. 이전 단계에서 설정한 개발자 계정을 통해 로그인합니다.
3. Salesforce 페이지에서 다음 중 하나를 수행합니다.
   - Lightning Experience를 사용 중인 경우에는 설정 기어 아이콘을 선택하고 Setup Home(설정 홈)을 선택합니다.
   - Salesforce Classic을 사용 중인 경우에는 사용자 인터페이스 헤더에 Setup(설정)이 표시됩니다. 이를 선택합니다.
   - Salesforce Classic을 사용하고 있는데 헤더에 설정이 표시되지 않는 경우 상단 탐색 모음에서 이름을 선택하고 드롭다운 목록에서 설정을 선택합니다.
4. 왼쪽 탐색 모음에서 Company Settings(회사 설정)를 선택합니다.
5. 탐색 모음에서 [도메인(Domain)]을 선택하고 도메인을 입력한 다음, [생성(Create)]을 선택합니다.
6. 왼쪽 탐색 모음에서 [플랫폼 도구(Platform Tools)]에서 [앱(Apps)]을 선택합니다.
7. App Manager(앱 관리자)를 선택합니다.
8. a. [새 연결된 앱(New connected app)]을 선택합니다.

   b. 필수 필드를 작성합니다.

      시작 URL(Start URL) 아래에서 Salesforce IdP로 로그인하는 사용자 풀 도메인에 대한 `/authorize` 엔드포인트에 URL을 입력합니다. 사용자가 연결된 앱에 액세스하면 Salesforce가 사용자를 이 URL로 안내하여 로그인을 완료합니다. 그런 다음 Salesforce는 사용자를 앱 클라이언트와 연결된 콜백 URL로 리디렉션합니다.

      ```
      https://mydomain.auth.us-east-1.amazoncognito.com/authorize?
      response_type=code&client_id=<your_client_id>&redirect_uri=https://
      www.example.com&identity_provider=CorpSalesforce
      ```

   c. OAuth 설정(OAuth settings)을 사용하고 콜백 URL(Callback URL)에 사용자 풀 도메인에 대한 `/oauth2/idpresponse` 엔드포인트의 URL을 입력합니다. 이는 Salesforce에서 Amazon Cognito가 OAuth 토큰과 교환하는 권한 부여 코드를 발행하는 URL입니다.

      ```
      https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/idpresponse
      ```

9. 사용자의 범위를 선택합니다. openid 범위를 포함해야 합니다. email 및 email_verified claims에 대한 액세스 권한을 부여하려면 email 범위를 추가합니다. 범위를 공백으로 구분합니다.
10. 생성을 선택합니다.

    Salesforce에서는 클라이언트 ID를 사용자 키(Consumer Key)로 부르고, 클라이언트 보안 암호를 사용자 보안 암호(Consumer Secret)로 부릅니다. 클라이언트 ID와 클라이언트 보안 암호를 메모합니다. 다음 섹션에서 이 둘을 사용합니다.

사용자 풀에 OIDC IdP 추가

IdP를 설정한 후 OIDC IdP로 인증 요청을 처리하도록 사용자 풀을 구성할 수 있습니다.

Amazon Cognito console

콘솔에 OIDC IdP 추가

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 AWS 자격 증명을 입력합니다.
2. 탐색 메뉴에서 [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. 소셜 및 외부 제공업체 메뉴를 선택한 다음 ID 제공업체 추가를 선택합니다.
5. OpenID Connect IdP를 선택합니다.
6. 고유한 제공업체 이름을 입력합니다.
7. IdP 클라이언트 ID를 입력합니다. OIDC IdP에서 빌드하는 애플리케이션 클라이언트의 ID입니다. 제공하는 클라이언트 ID는 `https://[your user pool domain]/oauth2/idpresponse`의 콜백 URL로 구성한 OIDC 제공업체여야 합니다.
8. IdP 클라이언트 보안 암호를 입력합니다. 이전 단계의 동일한 애플리케이션 클라이언트에 대한 클라이언트 보안 암호여야 합니다.
9. 이 공급자의 권한 있는 범위(Authorized scopes)를 입력합니다. 범위는 애플리케이션이 공급자에게 요청하는 사용자 속성 그룹(예: `name` 및 `email`)을 정의합니다. OAuth 2.0 사양에 따라 범위를 공백으로 구분해야 합니다.

    IdP는 사용자가 로그인할 때 애플리케이션에 이러한 속성을 제공하는 데 동의하라는 메시지를 표시할 수 있습니다.
10. 속성 요청 방법을 선택합니다. IdP는 `userInfo` 엔드포인트에 대한 요청이 `GET` 또는 `POST` 형식으로 지정되도록 요구할 수 있습니다. 예를 들어 Amazon Cognito `userInfo` 엔드포인트에는 `HTTP GET` 요청이 필요합니다.

11. 사용자 풀이 IdP의 주요 OIDC 페더레이션 엔드포인트 경로를 결정하도록 하려면 설정 방법을 선택합니다. 일반적으로 IdP는 발급자 기반 URL에서 `/well-known/openid-configuration` 엔드포인트를 호스팅합니다. 제공업체의 경우 발급자 URL 자동 채우기 옵션은 해당 기본 URL을 묻는 메시지를 표시하고, 이 URL에서 `/well-known/openid-configuration` 경로에 액세스하려고 시도하고, 여기에 나열된 엔드포인트를 읽습니다. 일반적이지 않은 엔드포인트 경로가 있거나 대체 프록시를 통해 하나 이상의 엔드포인트에 요청을 전달하려고 할 수 있습니다. 이 경우 수동 입력을 선택하고 `authorization`, `token`, `userInfo`, `jwks_uri` 엔드포인트의 경로를 지정합니다.

    > **Note**
    >
    > URL은 `https://`로 시작해야 하며, 슬래시 `/`로 끝나면 안 됩니다. 포트 번호 443 및 80만 이 URL에 사용할 수 있습니다. 예를 들어, Salesforce는 다음 URL을 사용합니다. `https://login.salesforce.com`
    > 자동 채우기를 선택하는 경우 검색 문서에서 `authorization_endpoint`, `token_endpoint`, `userinfo_endpoint`, `jwks_uri` 값에 HTTPS를 사용해야 합니다. 그렇지 않으면 로그인이 실패합니다.

12. OpenID Connect 제공업체와 사용자 풀 간의 속성 매핑에서 속성 매핑 규칙을 구성합니다. 사용자 풀 속성은 Amazon Cognito 사용자 프로필의 대상 속성이고 OpenID Connect 속성은 Amazon Cognito가 ID 토큰 클레임 또는 `userInfo` 응답에서 찾을 소스 속성입니다. Amazon Cognito는 대상 사용자 프로필에서 OIDC 클레임 하위를 `username`에 자동으로 매핑합니다.

    자세한 내용은 프로필 및 토큰에 IdP 속성 매핑 섹션을 참조하세요.

13. ID 제공업체 추가를 선택합니다.
14. 앱 클라이언트 메뉴의 목록에서 앱 클라이언트를 선택합니다. 로그인 페이지 탭으로 이동하여 관리형 로그인 페이지 구성에서 편집을 선택합니다. ID 제공업체를 찾고 새 OIDC IdP를 추가합니다.
15. 변경 사항 저장을 선택합니다.

API/CLI

CreateIdentityProvider에서 예제 2의 OIDC 구성을 참조하세요. 이 구문을 수정하여 `CreateIdentityProvider`, `UpdateIdentityProvider` 또는 create-identity-provider에 대한 `--cli-input-json` 입력 파일의 요청 본문으로 사용할 수 있습니다.

OIDC IdP 구성 테스트

애플리케이션에서 사용자가 OIDC 제공업체로 로그인할 수 있도록 사용자의 클라이언트에서 브라우저를 호출해야 합니다. 이전 단원의 설정 절차를 완료한 후 제공업체로 로그인을 테스트합니다. 다음 예제 URL은 접두사 도메인이 있는 사용자 풀의 로그인 페이지를 로드합니다.

```
https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&client_id=1example23456789&redirect_uri=https://www.example.com
```

이 링크는 앱 클라이언트 메뉴로 이동하여 앱 클라이언트를 선택하고 로그인 페이지 탭으로 이동한 다음 로그인 페이지 보기를 선택할 때 Amazon Cognito에서 안내하는 페이지입니다. 사용자 풀 도메인에 대한 자세한 정보는 사용자 풀 도메인 구성 섹션을 참조하세요. 클라이언트 IDs 및 콜백 URLs을 포함한 앱 클라이언트에 대한 자세한 내용은 앱 클라이언트를 사용한 애플리케이션별 설정 섹션을 참조하세요.

다음 예제 링크는 `identity_provider` 쿼리 파라미터를 사용하여권한 부여 엔드포인트 에서 `MyOIDCIdP` 제공업체로 자동 리디렉션을 설정합니다. 이 URL은 관리형 로그인을 통한 대화형 사용자 풀 로그인을 우회하고 IdP 로그인 페이지로 바로 이동합니다.

```
https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
identity_provider=MyOIDCIdP&response_type=code&client_id=1example23456789&redirect_uri=https://
www.example.com
```

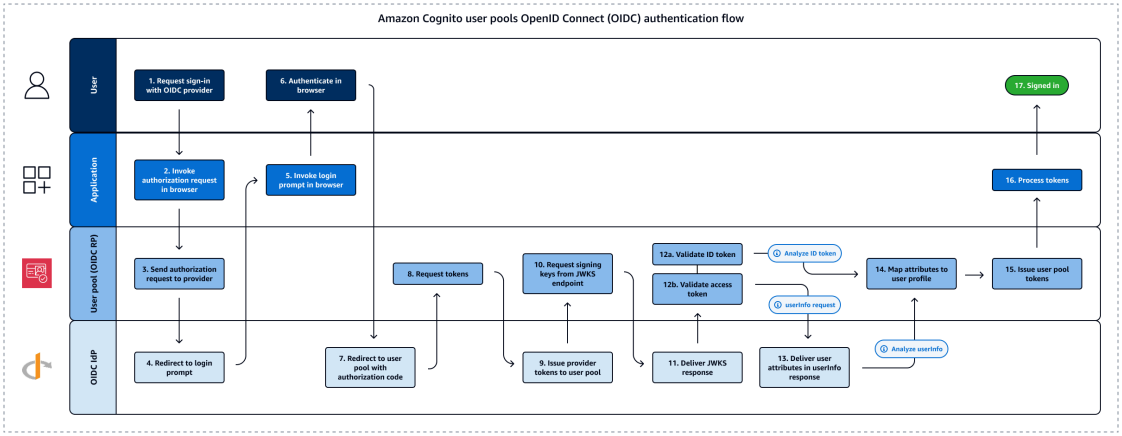
OIDC 사용자 풀 IdP 인증 흐름

OpenID Connect(OIDC) 로그인을 사용하면 사용자 풀이 ID 제공업체(IdP)를 통한 권한 부여 코드 로그인 흐름을 자동화합니다. 사용자가 IdP로 로그인을 완료하면 Amazon Cognito는 외부 제공업체의 `oauth2/idpresponse` 엔드포인트에서 코드를 수집합니다. 생성된 액세스 토큰을 사용하여 사용자 풀은 IdP `userInfo` 엔드포인트를 쿼리하여 사용자 속성을 검색합니다. 그런 다음 사용자 풀은 수신된 속성을 설정한 속성 매핑 규칙과 비교하고 그에 따라 사용자의 프로필과 ID 토큰을 채웁니다.

OIDC 제공업체 구성에서 요청하는 OAuth 2.0 범위는 IdP가 Amazon Cognito에 제공하는 사용자 속성을 정의합니다. 가장 좋은 방법은 사용자 풀에 매핑하려는 속성에 해당하는 범위만 요청하는 것입니다. 예를 들어 사용자 풀이 `openid profile`을 요청하면 가능한 모든 속성을 얻을 수 있지만 `openid email phone_number`를 요청하면 사용자의 이메일 주소와 전화번호만 가져옵니다. OIDC IdP에서 요청하는 범위가 앱 클라이언트 및 사용자 풀 인증 요청에서 승인하고 요청하는 범위와 달라지도록 구성할 수 있습니다.

사용자가 OIDC IdP를 사용하여 애플리케이션에 로그인하면 사용자 풀은 다음과 같은 인증 흐름을 수행합니다.

1. 사용자가 관리형 로그인 페이지에 액세스하고 OIDC IdP로 로그인하도록 선택합니다.
2. 애플리케이션은 사용자의 브라우저를 사용자 풀의 권한 부여 엔드포인트로 전달합니다.
3. 사용자 풀은 요청을 OIDC IdP의 권한 부여 엔드포인트로 리디렉션합니다.
4. IdP에 로그인 프롬프트가 표시됩니다.
5. 애플리케이션에서 사용자의 세션은 OIDC IdP에 대한 로그인 프롬프트를 표시합니다.
6. 사용자가 IdP에 대한 자격 증명을 입력하거나 이미 인증된 세션에 대한 쿠키를 제공합니다.
7. 사용자 인증 후 OIDC IdP에서 인증 코드를 사용하여 Amazon Cognito로 리디렉션합니다.
8. 사용자 풀은 승인 코드를 ID와 액세스 토큰으로 교환합니다. Amazon Cognito는 `openid` 범위로 IdP를 구성할 때 액세스 토큰을 수신합니다. ID 토큰 및 `userInfo` 응답의 클레임은 `profile` 및 `email`과 같은 IdP 구성의 추가 범위에 따라 결정됩니다.
9. IdP는 요청된 토큰을 발급합니다.
10. 사용자 풀은 IdP 구성의 발급자 URLs에서 IdP `jwks_uri` 엔드포인트의 경로를 결정하고 JSON 웹 키 세트(JWKS) 엔드포인트에서 토큰 서명 키를 요청합니다.
11. IdP는 JWKS 엔드포인트에서 서명 키를 반환합니다.
12. 사용자 풀은 토큰의 서명 및 만료 데이터에서 IdP 토큰을 검증합니다.
13. 사용자 풀은 액세스 토큰을 사용하여 IdP `userInfo` 엔드포인트에 대한 요청을 승인합니다. IdP는 액세스 토큰 범위에 따라 사용자 데이터로 응답합니다.
14. 사용자 풀은 IdP의 ID 토큰 및 `userInfo` 응답을 사용자 풀의 속성 매핑 규칙과 비교합니다. 매핑된 IdP 속성을 사용자 풀 프로필 속성에 기록합니다.
15. Amazon Cognito는 자격 증명, 액세스 권한 및 새로 고침 토큰이 포함될 수 있는 애플리케이션 보유자 토큰을 발행합니다.
16. 애플리케이션은 사용자 풀 토큰을 처리하고 사용자를 로그인합니다.



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

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 문서를 참조하세요.

사용자가 인증 코드 부여를 통해 인증할 때 사용자 풀이 ID, 액세스 및 새로 고침 토큰을 반환합니다. ID 토큰은 자격 증명 관리용 표준 OIDC 토큰이며, 액세스 토큰은 표준 OAuth 2.0 토큰입니다. 사용자 풀 앱 클라이언트가 지원할 수 있는 권한 부여 유형에 대한 자세한 내용은 권한 부여 엔드포인트 섹션을 참조하세요.

사용자 풀이 OIDC 제공업체의 클레임을 처리하는 방법

사용자가 서드 파티 OIDC 제공업체로 로그인을 완료하면 관리형 로그인이 IdP에서 권한 부여 코드를 검색합니다. 사용자 풀은 액세스 및 ID 토큰의 권한 부여 코드를 IdP의 `token` 엔드포인트와 교환합니다. 사용자 풀은 이러한 토큰을 사용자 또는 앱에 전달하지 않지만, 이를 사용하여 자체 토큰의 클레임에 표시되는 데이터로 사용자 프로필을 구축합니다.

Amazon Cognito는 액세스 토큰을 독립적으로 검증하지 않습니다. 대신 제공업체 `userInfo` 엔드포인트에서 사용자 속성 정보를 요청하고 토큰이 유효하지 않을 경우 요청이 거부될 것으로 예상합니다.

Amazon Cognito는 다음 확인을 통해 제공업체 ID 토큰을 검증합니다.

1. 제공업체가 RSA, HMAC, Elliptic Curve 세트의 알고리즘을 사용하여 토큰에 서명했는지 확인합니다.
2. 제공업체가 비대칭 서명 알고리즘으로 토큰에 서명한 경우 토큰 `kid` 클레임의 서명 키 ID가 제공업체 `jwks_uri` 엔드포인트에 나열되어 있는지 확인합니다. Amazon Cognito는 처리하는 각 IdP ID 토큰에 대해 IdP 구성의 JWKS 엔드포인트에서 서명 키를 새로 고칩니다.
3. ID 토큰 서명을 제공업체 메타데이터를 기반으로 예상하는 서명과 비교합니다.
4. `iss` 클레임을 IdP용으로 구성된 OIDC 발급자와 비교합니다.
5. `aud` 클레임이 IdP에 구성된 클라이언트 ID와 일치하는지 또는 `aud` 클레임에 여러 값이 있는 경우 구성된 클라이언트 ID가 포함되어 있는지 비교합니다.
6. `exp` 클레임의 타임스탬프가 현재 시간 이전이 아닌지 확인합니다.

사용자 풀은 ID 토큰을 검증한 다음 제공업체 액세스 토큰으로 제공업체 `userInfo` 엔드포인트에 요청을 시도합니다. 액세스 토큰의 범위에 따라 읽기 권한이 부여된 모든 사용자 프로필 정보를 검색합니다. 그런 다음 사용자 풀은 사용자 풀에서 필요에 따라 설정한 사용자 속성을 검색합니다. 필수 속성에 대해서는 제공업체 구성에서 속성 매핑을 생성해야 합니다. 사용자 풀은 제공업체 ID 토큰과 `userInfo` 응답을 확인합니다. 사용자 풀은 매핑 규칙과 일치하는 모든 클레임을 사용자 풀 사용자 프로필의 사용자 속성에 기록합니다. 사용자 풀은 매핑 규칙과 일치하지만 필수는 아니며 제공업체의 클레임에도 없는 속성은 무시합니다.

페더레이션 사용자를 위한 단계적 인증

OpenID Connect(OIDC) ID 제공업체(IdP)를 통해 로그인하는 페더레이션 사용자의 경우 Amazon Cognito는 자체 값을 계산하는 대신 OIDC IdP의 ACR 및 AMR 값을 전달합니다. OIDC IdP는 이러한 페더레이션 사용자에 대한 인증을 소유합니다. 페더레이션 사용자에 대한 단계적 인증은 OIDC IdP를 통해 이루어집니다. Amazon Cognito는를 사용하여 IdP로 리디렉션합니다`acr_values`. ACR 수준 및 AMR 값에 대한 자세한 내용은 섹션을 참조하세요the section called “ACR 및 AMR 클레임을 사용한 인증 수준”.

IdP의 `AcrMapping` 필드를 사용하여 OIDC IdP의 ACR 값을 Amazon Cognito 수준에 매핑합니다. CreateIdentityProvider 또는 UpdateIdentityProvider 작업을 사용하여이 필드를 구성합니다. Amazon Cognito는 OIDC IdP의 ACR 값을 Amazon Cognito 레벨에 매핑한 다음 사용자 풀에 대해 구성한 해당 레벨 이름을 Amazon Cognito 토큰에 스탬프합니다. 다음 예제에서는 OIDC 공급자에 `AcrMapping` 대해를 구성합니다.

```
{
  "ProviderName": "MyOktaIdP",
```

```
  "ProviderType": "OIDC",
  "AcrMapping": {
    "Level1": "urn:okta:loa:1",
    "Level2": "urn:okta:loa:2",
    "Level3": "urn:okta:loa:3",
    "Level4": "urn:okta:loa:4"
  }
}
```

자격 증명 공급자에 AcrMapping 대한 설정은 모든 기능 플랜에서 사용할 수 있습니다. Essentials 또는 Plus 기능 플랜으로 제한되지 않습니다.

다음 동작은 OIDC IdP를 통해 로그인하는 페더레이션 사용자에게 적용됩니다.

- OIDC IdP가 구성된에 없는 ACR 값을 반환하는 경우 AcrMapping Amazon Cognito는 토큰의 acr 클레임을 가장 낮은 수준(기본 urn:cognito:loa:1, 또는 사용자 풀에 대해 구성한 level-1 이름)으로 설정합니다. 오류는 아닙니다.
- Amazon Cognito는 인식하지 못하는 모든 AMR 값을 삭제하고 알려진 값만 유지합니다.
- OIDC IdP가 ACR 및 AMR을 지원하지 않거나 Amazon Cognito가 인식하는 AMR 값을 반환하지 않는 경우 Amazon Cognito는 fed를 기본 자리 표시자 AMR 값으로 설정합니다.
- Amazon Cognito는 매핑 및 필터링에 따라 OIDC IdP가 어설션하는 acr 및 amr 값을 신뢰합니다. 실제로 수행한 IdP 문제를 검증하지 않습니다.

페더레이션 흐름에서 애플리케이션은 대상 수준 또는 사용자 풀의 이름으로 표현된 수준을 사용하여 /oauth2/authorize 엔드포인트로 리디렉션합니다. Amazon Cognito는 OIDC IdP가에서 예상하는 값으로 값을 변환하고AcrMapping, IdP로 리디렉션하고, IdP의 ACR 및 AMR 값을 수신하고, 다시 매핑하고, Amazon Cognito 토큰을 발급합니다. IdP 구성에 대한 자세한 내용은 자격 증명 공급자 구성을 참조하세요.

프로필 및 토큰에 IdP 속성 매핑

Amazon Cognito를 포함한 ID 제공업체(IdP) 서비스는 일반적으로 사용자에 대한 추가 정보를 기록할 수 있습니다. 어떤 회사에서 근무하는지, 어떻게 연락하는지, 기타 식별 정보를 알고 싶을 수 있습니다. 그러나 이러한 속성이 사용하는 형식은 제공업체마다 차이가 있습니다. 예를 들어 사용자 풀을 사용하여 서로 다른 세 공급업체에서 세 개의 IdP를 설정하고 각각에서 예제 SAML 어설션, ID 토큰 또는 userInfo 페이로드를 검사합니다. 하나는 사용자의 이메일 주소를 email로, 다른 하나는 emailaddress로, 세 번째는 http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress로 나타냅니다.

IdPs 사용자 풀과 통합하면 다양한 속성 이름을 일관되고 예측 가능한 공유 속성 이름으로 단일 OIDC 토큰 스키마에 매핑할 수 있다는 이점이 있습니다. 이렇게 하면 개발자가 복잡한 다양한 Single Sign-On 이벤트를 처리하기 위한 로직을 유지할 필요가 없습니다. 이 형식 통합은 속성 매핑입니다. 사용자 풀 속성 매핑은 IdP 속성 이름을 해당 사용자 풀 속성 이름에 할당합니다. 예를 들어 표준 사용자 풀 속성 `email`에 `emailaddress` 클레임 값을 쓰도록 사용자 풀을 구성할 수 있습니다.

각 사용자 풀 IdP에는 별도의 속성 매핑 스키마가 있습니다. IdP에 대한 속성 매핑을 지정하려면 Amazon Cognito 콘솔, AWS SDK 또는 사용자 풀 REST API에서 사용자 풀 ID 제공업체를 구성합니다.

매핑에 대해 알아야 할 사항

사용자 속성 매핑 설정을 시작하기 전에 다음 중요 세부 정보를 검토합니다.

- IdP 클레임을 사용자 지정 속성에 매핑하려면 먼저 해당 사용자 지정 속성이 사용자 풀 스키마에 이미 있어야 합니다. 사용자 풀을 생성하거나 나중에 추가할 때 사용자 지정 속성을 정의하면 스키마에 아직 추가하지 않은 사용자 지정 속성에 IdP 속성을 매핑할 수 없습니다. 풀에서 속성을 정의하는 방법에 대한 자세한 내용은 섹션을 참조하세요사용자 속성 작업.
- 페더레이션 사용자가 애플리케이션에 로그인할 때 사용자 풀에 필요한 각 사용자 풀 속성의 매핑이 있어야 합니다. 예를 들어, 사용자 풀에 로그인용 `email` 속성이 필요한 경우 이 속성을 IdP의 해당 속성에 매핑합니다.
- 기본적으로 매핑된 이메일 주소는 확인되지 않습니다. 일회용 코드를 사용하여 매핑된 이메일 주소는 확인할 수 없습니다. 대신 IdP의 속성을 매핑하여 확인 상태를 가져옵니다. 예를 들어 Google과 대부분의 OIDC 공급자는 `email_verified` 속성을 포함합니다.
- ID 제공업체(IdP) 토큰을 사용자 풀의 사용자 지정 속성에 매핑할 수 있습니다. 소셜 제공업체는 액세스 토큰을 제시하고, OIDC 제공업체는 액세스 및 ID 토큰을 제공합니다. 토큰을 매핑하려면 최대 2,048자의 사용자 지정 속성을 추가하고, 앱 클라이언트에 속성에 대한 쓰기 권한을 부여하고, IdP에서 사용자 지정 속성으로 `access_token` 또는 `id_token`을 매핑합니다.
- 매핑된 각 사용자 풀 속성의 경우 최대 값 길이(2,048자)는 Amazon Cognito가 IdP에서 가져오는 값에 대해 충분히 커야 합니다. 그러지 않으면 사용자가 애플리케이션에 로그인할 때 Amazon Cognito에서 오류를 보고합니다. Amazon Cognito는 토큰 길이가 2,048자를 초과하는 경우 IdP 토큰을 사용자 지정 속성에 매핑하는 것을 지원하지 않습니다.
- Amazon Cognito는 다음 표에 설명된 것과 같이 페더레이션 IdP가 전달하는 특정 클레임에서 페더레이션 사용자 프로필의 `username` 속성을 추출합니다. Amazon Cognito는 이 속성 값 앞에 IdP 이름(예: `MyOIDCIdP_[sub]`)을 추가합니다. 페더레이션 사용자가 외부 사용자 디렉터리의 속성과 정확히 일치하는 속성을 갖도록 하려면 해당 속성을 Amazon Cognito 로그인 속성(예: `preferred_username`)에 매핑합니다.

ID 공급자	**username** 소스 속성
Facebook	`id`
Google	`sub`
Login with Amazon	`user_id`
Apple로 로그인	`sub`
SAML 공급자	`NameID`
OpenID Connect(OIDC) 공급자	`sub`

- 사용자 풀이 대소문자를 구분하지 않는 경우 Amazon Cognito는 페더레이션 사용자의 자동 생성된 전체 사용자 이름을 IdP 이름 접두사를 포함한 소문자로 변환합니다. `MySAML_TestUser@example.com`은 대/소문자를 구분하는 사용자 풀의 사용자 이름 예시입니다. `mysaml_testuser@example.com`은 대/소문자를 구분하지 않는 사용자 풀의 사용자 이름과 동일합니다.

  대/소문자를 구분하지 않는 사용자 풀에서 사용자 이름을 처리하는 Lambda 트리거는 전체 사용자 이름의 변경 사항을 고려해야 합니다. 이러한 변경 사항에는 IdP에서 매핑한 사용자 이름 소스 속성의 혼합 대소문자 값이 포함됩니다. 현재 풀과 대/소문자 구분 설정이 다른 사용자 풀에 IdP를 연결하려면 새 사용자 풀을 생성합니다.
- 사용자가 애플리케이션에 로그인할 때 Amazon Cognito는 매핑된 사용자 풀 속성을 업데이트할 수 있어야 합니다. 사용자가 IdP를 통해 로그인하면 Amazon Cognito가 매핑된 속성을 IdP의 최신 정보로 업데이트합니다. Amazon Cognito는 값이 변경될 때만 매핑된 속성을 업데이트합니다. Amazon Cognito가 속성을 업데이트할 수 있는지 확인하려면 다음 요구 사항을 확인하세요.
  - IdP에서 매핑하는 모든 사용자 풀 사용자 지정 속성은 변경할 수 있어야 합니다. 변경 가능한 사용자 지정 속성은 언제든지 업데이트할 수 있습니다. 반대로 사용자 프로필을 처음 만들 때 사용자의 변경할 수 없는 사용자 지정 속성 값만 설정할 수 있습니다. Amazon Cognito 콘솔에서 변경 가능한 사용자 지정 속성을 생성하려면 가입 메뉴에서 사용자 지정 속성 추가를 선택할 때 추가한 속성의 변경 가능 확인란을 활성화합니다. 또는 CreateUserPool API 작업을 사용하여 사용자 풀을 생성하는 경우 이러한 각 속성에 대한 `Mutable` 파라미터를 `true`로 설정할 수 있습니다. IdP가 매핑된 변경 불가능한 속성에 대한 값을 전송하는 경우 Amazon Cognito는 오류를 반환하고 로그인에 실패합니다.

  - 애플리케이션의 앱 클라이언트 설정에서 매핑된 속성은 쓰기 가능해야 합니다. Amazon Cognito 콘솔의 [앱 클라이언트(App clients)] 페이지에 쓸 수 있는 속성을 설정할 수 있습니다. 또는 `CreateUserPoolClient` API 작업을 사용하여 앱 클라이언트를 생성하는 경우 이러한 속성을 `WriteAttributes` 어레이에 추가할 수 있습니다. IdP가 매핑된 쓰기 불가능한 속성에 대한 값을 전송하는 경우 Amazon Cognito는 속성 값을 설정하지 않고 인증을 진행합니다.
- IdP 속성에 여러 값이 포함된 경우 Amazon Cognito는 모든 값을 대괄호 [및]로 묶인 쉼표로 구분된 단일 문자열로 평면화합니다. Amazon Cognito URL은 `.`, `-`, `*`, `_`를 제외한 영숫자가 아닌 문자를 포함하는 값을 양식 인코딩합니다. 이러한 개별 값은 앱에 사용하기 전에 디코딩 및 구문 분석해야 합니다.
- 대상 속성은 로그인 또는 관리 작업이 변경하지 않는 한 속성 매핑 규칙이 할당하는 모든 값을 유지합니다. Amazon Cognito는 소스 속성이 더 이상 공급자 토큰 또는 SAML 어설션으로 전송되지 않을 때 사용자의 속성을 제거하지 않습니다. 다음 작업은 페더레이션 사용자의 사용자 풀 프로파일에서 속성 값을 제거합니다.

  1. IdP는 소스 속성의 빈 값을 전송하고 매핑 규칙은 빈 값을 대상 속성에 적용합니다.
  2. DeleteUserAttributes 또는 AdminDeleteUserAttributes 요청을 사용하여 매핑된 속성의 값을 지웁니다.

사용자 풀에 대한 자격 증명 공급자 속성 매핑 지정(AWS Management Console)

AWS Management Console 를 사용하여 사용자 풀의 IdP에 대한 속성 매핑을 지정할 수 있습니다.

> **Note**
>
> Amazon Cognito는 클레임이 수신 토큰에 있는 경우에만 수신 클레임을 사용자 풀 속성에 매핑합니다. 이전에 매핑된 클레임이 수신 토큰에 더 이상 존재하지 않으면 삭제되거나 변경되지 않습니다. 애플리케이션에 삭제된 클레임의 매핑이 필요한 경우 사전 인증 Lambda 트리거를 사용하여 인증 중에 사용자 지정 속성을 삭제하고 이러한 속성이 수신 토큰에서 다시 채워지도록 허용할 수 있습니다.

소셜 IdP 속성 매핑을 지정하려면

1. Amazon Cognito 콘솔에 로그인합니다. 메시지가 표시되면 자격 AWS 증명을 입력합니다.
2. 탐색 창에서 [사용자 풀(User Pools)]을 선택한 다음 편집할 사용자 풀을 선택합니다.
3. 소셜 및 외부 제공업체 메뉴를 선택합니다.

4. ID 제공업체 추가를 선택하거나 구성한 Facebook, Google, Amazon 또는 Apple IdP를 선택합니다. 속성 매핑(Attribute mapping)을 찾아서 편집(Edit)을 선택합니다.

   소셜 IdP 추가에 대한 자세한 내용은 사용자 풀에서 소셜 ID 제공업체 사용 섹션을 참조하세요.

5. 매핑해야 하는 각 속성에 대해 다음 단계를 완료합니다.

   a. [사용자 풀 속성(User pool attribute)] 열에서 속성을 선택합니다. 사용자 풀의 사용자 프로파일에 할당되는 속성입니다. 사용자 정의 속성은 표준 속성 뒤에 나열됩니다.

   b. 속성(attribute) 열에서 속성을 선택합니다. 공급자 디렉터리에서 전달되는 속성입니다. 소셜 공급자의 알려진 속성은 드롭다운 목록에 제공됩니다.

   c. IdP와 Amazon Cognito 간에 추가 속성을 매핑하려면 다른 속성 추가(Add another attribute)를 선택합니다.

6. 변경 사항 저장(Save changes)을 선택합니다.

SAML 공급자 속성 매핑을 지정하려면

1. Amazon Cognito 콘솔에 로그인합니다. 메시지가 표시되면 자격 AWS 증명을 입력합니다.
2. 탐색 창에서 [사용자 풀(User Pools)]을 선택한 다음 편집할 사용자 풀을 선택합니다.
3. 소셜 및 외부 제공업체 메뉴를 선택합니다.
4. 자격 증명 공급자 추가(Add an identity provider)를 선택하거나 구성한 SAML IdP를 선택합니다. 속성 매핑(Attribute mapping)을 찾아서 편집(Edit)을 선택합니다. SAML IdP 추가에 대한 자세한 내용은 사용자 풀에서 SAML ID 제공업체 사용 섹션을 참조하세요.
5. 매핑해야 하는 각 속성에 대해 다음 단계를 완료합니다.

   a. [사용자 풀 속성(User pool attribute)] 열에서 속성을 선택합니다. 사용자 풀의 사용자 프로파일에 할당되는 속성입니다. 사용자 정의 속성은 표준 속성 뒤에 나열됩니다.

   b. [SAML 속성(SAML attribute)] 열에서 속성을 선택합니다. 공급자 디렉터리에서 전달되는 속성입니다.

      IdP가 샘플 SAML 어설션을 참조용으로 제공할 수도 있습니다. `email`과 같은 간단한 이름을 사용하는 IdP도 있고, 다음과 같이 URL 형식의 속성 이름을 사용하는 IdP도 있습니다.

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

   c. IdP와 Amazon Cognito 간에 추가 속성을 매핑하려면 다른 속성 추가(Add another attribute)를 선택합니다.

6. 변경 사항 저장을 선택합니다.

사용자 풀(AWS CLI 및 AWS API)에 대한 자격 증명 공급자 속성 매핑 지정

CreateIdentityProvider 또는 UpdateIdentityProvider에 대한 다음 요청 본문은 SAML 제공업체 “MyIdP” 속성 `emailaddress`, `birthdate` 및 `phone`을 사용자 풀 속성 `email`, `birthdate` 및 `phone_number`에 해당 순서로 매핑합니다. 이는 SAML 2.0 공급자를 위한 전체 요청 본문입니다. 요청 본문은 IdP 유형 및 특정 세부 정보에 따라 달라집니다. 속성 매핑은 `AttributeMapping` 파라미터에 있습니다.

```
{
   "AttributeMapping": {
      "email" : "emailaddress",
      "birthdate" : "birthdate",
      "phone_number" : "phone"
   },
   "IdpIdentifiers": [
      "IdP1",
      "pdxsaml"
   ],
   "ProviderDetails": {
      "IDPInit": "true",
      "IDPSignout": "true",
      "EncryptedResponses" : "true",
      "MetadataURL": "https://auth.example.com/sso/saml/metadata",
      "RequestSigningAlgorithm": "rsa-sha256"
   },
   "ProviderName": "MyIdP",
   "ProviderType": "SAML",
   "UserPoolId": "us-west-2_EXAMPLE"
}
```

다음 명령을 사용하여 사용자 풀에 대한 IdP 속성 매핑을 지정합니다.

공급자 생성 시 속성 매핑을 지정하려면

- AWS CLI: `aws cognito-idp create-identity-provider`

  메타데이터 파일이 포함된 예제: `aws cognito-idp create-identity-provider --user-pool-id <user_pool_id> --provider-name=SAML_provider_1 --provider-`

`type SAML --provider-details file:///details.json --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

여기서 `details.json`에 다음 사항이 포함됩니다.

```
{
    "MetadataFile": "<SAML metadata XML>"
}
```

> **Note**
>
> *<SAML metadata XML>*에 따옴표(")가 있으면 이스케이프해야 합니다(\").

메타데이터 URL이 포함된 예시:

```
aws cognito-idp create-identity-provider \
--user-pool-id us-east-1_EXAMPLE \
--provider-name=SAML_provider_1 \
--provider-type SAML \
--provider-details MetadataURL=https://myidp.example.com/saml/metadata \
--attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/
emailaddress
```

- API/SDK: CreateIdentityProvider

기존 IdP에 대한 속성 매핑을 지정하려면

- AWS CLI: `aws cognito-idp update-identity-provider`

  예시: `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`
- API/SDK: UpdateIdentityProvider

특정 IdP에 대한 속성 매핑 정보를 가져오려면

- AWS CLI: `aws cognito-idp describe-identity-provider`

예시: `aws cognito-idp describe-identity-provider --user-pool-id <user_pool_id> --provider-name <provider_name>`

- API/SDK: DescribeIdentityProvider

페더레이션 사용자를 기존 사용자 프로필에 연결

동일한 사용자에게 사용자 풀에 연결한 아이덴티티 제공업체(IdP)가 여러 개인 프로필이 있는 경우가 많습니다. Amazon Cognito는 사용자의 각 발생을 디렉터리에 있는 동일한 사용자 프로필에 연결할 수 있습니다. 이렇게 하면 여러 IdP 사용자를 보유한 한 사람이 앱에서 일관된 경험을 할 수 있습니다. AdminLinkProviderForUser는 Amazon Cognito에 페더레이션 디렉터리에 있는 사용자의 고유 ID를 사용자 풀의 사용자로 인식하도록 지시합니다. 사용자 프로필과 연결된 페더레이션 ID가 0개 이상인 경우 사용자 풀의 사용자는 청구 목적으로 1명의 월별 활성 사용자(MAU)로 계산됩니다.

페더레이션 사용자가 처음으로 사용자 풀에 로그인하면 Amazon Cognito는 해당 사용자의 자격 증명에 연결된 로컬 프로필을 찾습니다. 연결된 프로필이 없는 경우 사용자 풀이 새 프로필을 생성합니다. 로컬 프로필을 생성하여 페더레이션 사용자가 처음 로그인하기 전에 언제든지 계획된 사전 준비 작업 또는 사전 가입 Lambda 트리거에서 `AdminLinkProviderForUser` API 요청을 통해 페더레이션 사용자에게 연결할 수 있습니다. 사용자가 로그인하고 Amazon Cognito가 연결된 로컬 프로필을 감지하면 사용자 풀이 사용자의 클레임을 읽고 이를 IdP의 매핑 규칙과 비교합니다. 그런 다음 사용자 풀은 로그인에서 매핑된 클레임으로 연결된 로컬 프로필을 업데이트합니다. 이렇게 하여 액세스 클레임이 포함된 로컬 프로필을 구성하고 제공업체를 통해 자격 증명 클레임을 최신 상태로 유지할 수 있습니다. Amazon Cognito가 페더레이션 사용자를 연결된 프로필과 매칭하면 페더레이션 사용자는 항상 해당 프로필에 로그인하게 됩니다. 그런 다음 더 많은 사용자 제공업체 자격 증명을 동일한 프로필에 연결하여 각 고객에게 일관된 앱 경험을 제공할 수 있습니다. 이전에 로그인한 페더레이션 사용자를 연결하려면 먼저 기존 프로필을 삭제해야 합니다. *`[Provider name]`*`_identifier` 형식으로 기존 프로필을 식별할 수 있습니다. 예: `LoginWithAmazon_amzn1.account.AFAEXAMPLE` 생성한 다음 타사 사용자 ID에 연결한 사용자는 생성된 사용자 이름과 연결된 ID의 세부 정보가 포함된 `identities` 속성을 갖습니다.

> **Important**
>
> `AdminLinkProviderForUser`에서는 외부 페더레이션 ID를 가진 사용자가 사용자 풀에 기존 사용자로 로그인할 수 있도록 허용하므로, 애플리케이션 소유자가 신뢰하는 외부 IdP 및 공급자 속성으로만 사용해야 합니다.

예를 들어, 여러 고객과 공유하는 앱을 사용하는 관리형 서비스 제공업체(MSP)라고 가정해 봅니다. 각 고객은 Active Directory Federation Services(ADFS)를 통해 앱에 로그인합니다. IT 관리자인 Carlos는 각 고객 도메인에 계정을 가지고 있습니다. IdP에 관계없이 Carlos가 로그인할 때마다 앱 관리자로 인식되기를 원합니다.

ADFS IdP는 Carlos의 SAML 어설션에 대한 `email` 클레임에 있는 Carlos의 이메일 주소 `msp_carlos@example.com`을 Amazon Cognito에 제시합니다. 사용자 이름을 사용하여 사용자 풀에 사용자를 생성합니다`Carlos`. 다음 AWS Command Line Interface (AWS CLI) 명령은 IdP ADFS1, ADFS2, ADFS3에서 Carlos의 ID를 연결합니다.

> **Note**
>
> 특정 속성 클레임을 기반으로 사용자를 연결할 수 있습니다. 이 기능은 OIDC 및 SAML IdP에 고유합니다. 다른 공급자 유형의 경우 고정된 소스 속성을 기반으로 연결해야 합니다. 자세한 내용은 AdminLinkProviderForUser를 참조하세요. 소셜 IdP를 사용자 프로필에 연결할 때 `ProviderAttributeName`을 `Cognito_Subject`로 설정해야 합니다. `ProviderAttributeValue`는 IdP와 함께 사용자의 고유 식별자여야 합니다.

```
aws cognito-idp admin-link-provider-for-user \
--user-pool-id us-east-1_EXAMPLE \
--destination-user ProviderAttributeValue=Carlos,ProviderName=Cognito \
--source-user
 ProviderName=ADFS1,ProviderAttributeName=email,ProviderAttributeValue=msp_carlos@example.com

aws cognito-idp admin-link-provider-for-user \
--user-pool-id us-east-1_EXAMPLE \
--destination-user ProviderAttributeValue=Carlos,ProviderName=Cognito \
--source-user
 ProviderName=ADFS2,ProviderAttributeName=email,ProviderAttributeValue=msp_carlos@example.com

aws cognito-idp admin-link-provider-for-user \
--user-pool-id us-east-1_EXAMPLE \
--destination-user ProviderAttributeValue=Carlos,ProviderName=Cognito \
--source-user
 ProviderName=ADFS3,ProviderAttributeName=email,ProviderAttributeValue=msp_carlos@example.com
```

이제 사용자 풀의 `Carlos` 사용자 프로필에 다음 `identities` 속성이 있습니다.

```
[{
```

```
    "userId": "msp_carlos@example.com",
    "providerName": "ADFS1",
    "providerType": "SAML",
    "issuer": "http://auth.example.com",
    "primary": false,
    "dateCreated": 111111111111111
}, {
    "userId": "msp_carlos@example.com",
    "providerName": "ADFS2",
    "providerType": "SAML",
    "issuer": "http://auth2.example.com",
    "primary": false,
    "dateCreated": 111111111111111
}, {
    "userId": "msp_carlos@example.com",
    "providerName": "ADFS3",
    "providerType": "SAML",
    "issuer": "http://auth3.example.com",
    "primary": false,
    "dateCreated": 111111111111111
}]
```

페더레이션 사용자 연결에 대해 알아야 할 사항

- 최대 5명의 페더레이션 사용자를 각 사용자 프로필에 연결할 수 있습니다.
- AdminLinkProviderForUser API 요청에서 SourceUser의 ProviderAttributeName 파라미터로 정의된 대로 최대 5개의 IdP 속성 클레임에서 사용자를 각 IdP에 연결할 수 있습니다. 예를 들어 소스 속성 email, phone, department, given_name 및 location에 사용자를 하나 이상 연결한 경우 해당 5개 속성 중 하나에 대한 추가 사용자만 연결할 수 있습니다.
- 페더레이션 사용자를 기존 페더레이션 사용자 프로필이나 로컬 사용자에 연결할 수 있습니다.
- 에서는 공급자를 사용자 프로필에 연결할 수 없습니다 AWS Management Console.
- 사용자의 ID 토큰에는 identities 클레임에 연결된 모든 공급자가 포함되어 있습니다.
- AdminSetUserPassword API 요청에서 자동으로 생성된 페더레이션 사용자 프로필의 암호를 설정할 수 있습니다. 그러면 해당 사용자의 상태가 EXTERNAL_PROVIDER에서 CONFIRMED로 변경됩니다. 이 상태의 사용자는 페더레이션 사용자로 로그인할 수 있으며, 연결된 로컬 사용자처럼 API에서 인증 흐름을 시작할 수 있습니다. 또한 ChangePassword 및 UpdateUserAttribute와 같은 토큰 인증 API 요청에서 암호와 속성을 수정할 수 있습니다. 보안을 유지하고 사용자를 외부 IdP와 동기화된 상태를 유지하기 위해 페더레이션 사용자에서 암호를 설정하지 않는 것이 좋습니다. 대신 AdminLinkProviderForUser를 사용하여 사용자를 로컬 프로필에 연결하세요.

- Amazon Cognito는 사용자가 IdP를 통해 로그인할 때 연결된 로컬 사용자 프로필에 사용자 속성을 채웁니다. Amazon Cognito는 OIDC IdP의 ID 토큰에 있는 자격 증명 클레임을 처리하고, OAuth 2.0 및 OIDC 공급자의 `userInfo` 엔드포인트도 확인합니다. Amazon Cognito는 `userInfo`의 정보보다 ID 토큰에 있는 정보를 우선시합니다.

프로필에 연결한 외부 사용자 계정을 사용자가 더 이상 사용하지 않는다는 사실을 알게 되면 해당 사용자 계정과 사용자 풀 사용자의 연결을 끊을 수 있습니다. 사용자를 연결할 때 요청에 사용자의 속성 이름, 속성 값 및 제공업체 이름을 제공했습니다. 사용자에게 더 이상 필요하지 않은 프로필을 제거하려면 동등한 파라미터를 사용하여 AdminDisableProviderforUser API 요청을 수행합니다.

AWS SDK의 예 및 추가 명령 구문은 AdminLinkProviderForUser를 참조하세요.

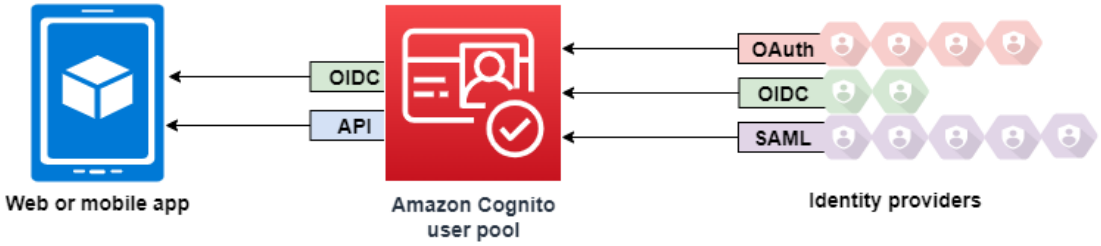
로그인 성공 후 리소스 액세스

앱 사용자는 사용자 풀을 통해 직접 로그인하거나 타사 ID 제공업체(idP)를 통해 연동할 수 있습니다. 사용자 풀에서는 Facebook, Google 및 Apple, 그리고 Amazon을 통한 소셜 로그인에서 반환된 토큰과 OpenID Connect(OIDC) 및 SAML IdP에서 반환된 토큰의 처리 작업을 관리합니다. 자세한 내용은 사용자 풀 JSON 웹 토큰(JWT) 이해 섹션을 참조하세요.

인증에 성공하면 앱은 Amazon Cognito로부터 사용자 풀 토큰을 수신합니다. 사용자 풀 토큰을 사용하여 다음을 수행할 수 있습니다.

- Amazon DynamoDB 및 Amazon S3와 AWS 서비스 같은의 애플리케이션 리소스에 대한 요청을 승인하는 AWS 자격 증명을 검색합니다.
- 취소 가능한 임시 인증 증명을 제공합니다.
- 앱의 사용자 프로필에 ID 데이터를 채웁니다.
- 사용자 풀 디렉터리에서 로그인한 사용자의 프로필에 대한 변경 사항을 승인합니다.
- 액세스 토큰을 사용하여 사용자 정보에 대한 요청을 승인합니다.
- 액세스 토큰을 사용하여 액세스 보호 외부 API 뒤에 있는 데이터에 대한 요청을 승인합니다.
- Amazon Verified Permissions를 사용하여 클라이언트 또는 서버에 저장된 애플리케이션 자산에 대한 액세스를 승인합니다.

자세한 내용은 인증 세션의 예 및 사용자 풀 JSON 웹 토큰(JWT) 이해 섹션을 참조하세요.



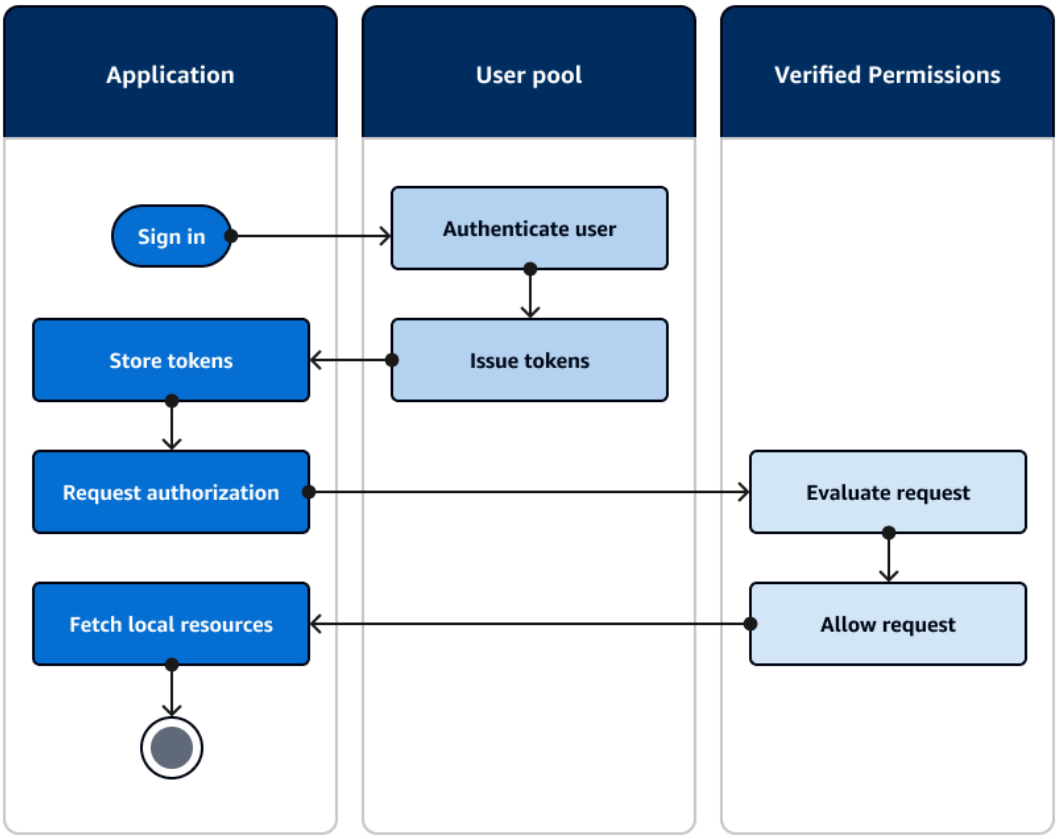
주제

- Amazon Verified Permissions를 사용하여 클라이언트 또는 서버 리소스에 대한 액세스 권한 부여
- 로그인 후 API Gateway를 사용하여 리소스 액세스
- 로그인 후 자격 증명 풀을 AWS 서비스 사용하여 액세스

Amazon Verified Permissions를 사용하여 클라이언트 또는 서버 리소스에 대한 액세스 권한 부여

앱은 로그인한 사용자의 토큰을 Amazon Verified Permissions에 전달할 수 있습니다. Verified Permissions는 사용자가 빌드한 애플리케이션을 위한 확장 가능하고 세분화된 권한 관리 및 권한 부여 서비스입니다. Amazon Cognito 사용자 풀은 Verified Permissions 정책 저장소의 ID 소스일 수 있습니다. Verified Permissions는 요청된 작업 및 리소스(예: `premium_badge.png`의 `GetPhoto`)에 대한 권한 부여 결정을 보안 주체 및 사용자 풀 토큰의 해당 속성에서 수행합니다.

다음 다이어그램은 애플리케이션이 권한 부여 요청의 Verified Permissions에 사용자 토큰을 전달하는 방법을 보여줍니다.



Amazon Verified Permissions 시작

사용자 풀을 Verified Permissions와 통합하면 모든 Amazon Cognito 앱에 대한 세분화된 권한 부여의 중앙 소스를 얻을 수 있습니다. 이렇게 하면 모든 앱 간에 코딩하고 복제해야 하는 세분화된 보안 로직이 필요하지 않습니다. Verified Permissions를 사용한 권한 부여에 대한 자세한 내용은 Amazon Verified Permissions를 통한 권한 부여 섹션을 참조하세요.

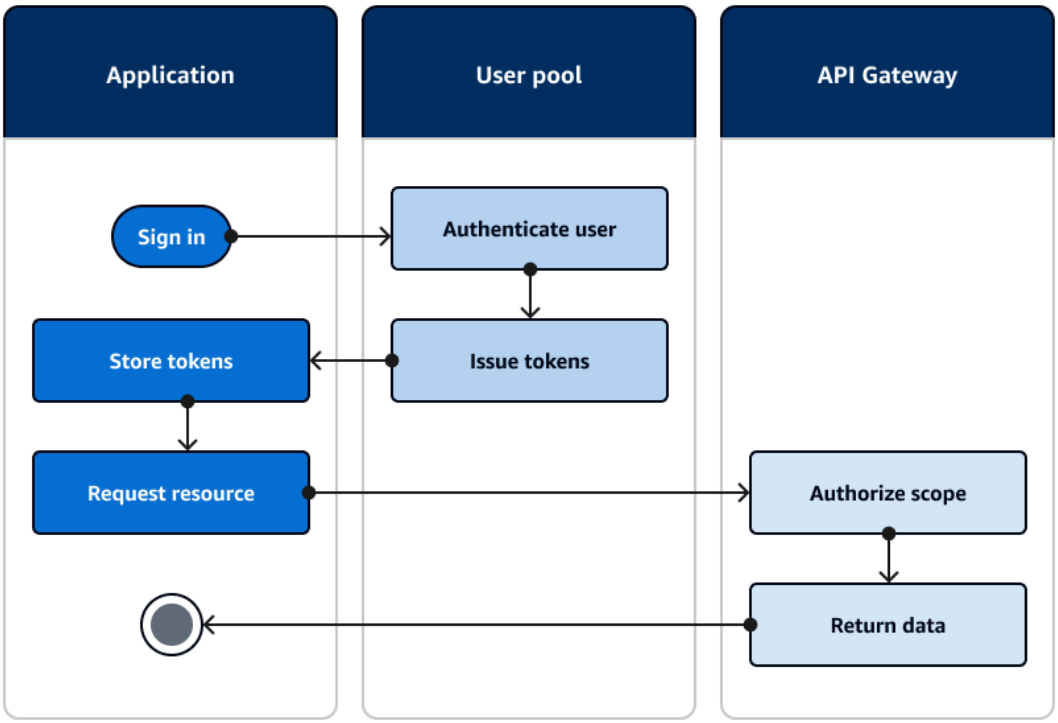
Verified Permissions 권한 부여 요청에는 AWS 자격 증명이 필요합니다. 다음 기법 중 일부를 구현하여 권한 부여 요청에 자격 증명을 안전하게 적용할 수 있습니다.

- 서버 백엔드에 암호를 저장할 수 있는 웹 애플리케이션을 운영합니다.
- 인증된 ID 풀 자격 증명을 획득합니다.
- 프록시 사용자는 액세스 토큰이 승인된 API를 통해 요청하고 요청에 AWS 자격 증명을 추가합니다.

로그인 후 API Gateway를 사용하여 리소스 액세스

Amazon Cognito 사용자 풀 토큰의 일반적인 사용은 API Gateway REST API에 대한 요청을 승인하는 것입니다. 액세스 토큰의 OAuth 2.0 범위는 `/app_assets`에 대한 `HTTP GET`과 같은 메서드 및 경로를 승인할 수 있습니다. ID 토큰은 API에 대한 일반 인증 역할을 할 수 있으며 사용자 속성을 백엔드 서비스에 전달할 수 있습니다. API Gateway에는 HTTP API용 JWT 권한 부여자 및 Lambda 권한 부여자와 같이 보다 세분화된 로직을 적용할 수 있는 추가 사용자 지정 권한 부여 옵션이 있습니다.

다음 다이어그램은 액세스 토큰의 OAuth 2.0 범위를 사용하여 REST API에 액세스하는 애플리케이션을 보여줍니다.



앱은 인증된 세션에서 토큰을 수집하여 요청의 `Authorization` 헤더에 무기명 토큰으로 추가해야 합니다. API, 경로 및 메서드에 대해 구성한 권한 부여자를 구성하여 토큰 콘텐츠를 평가합니다. API Gateway는 요청이 권한 부여자에 대해 설정한 조건과 일치하는 경우에만 데이터를 반환합니다.

API Gateway API가 애플리케이션에서 액세스를 승인할 수 있는 몇 가지 잠재적 방법은 다음과 같습니다.

- 액세스 토큰이 유효하고 만료되지 않았으며 올바른 OAuth 2.0 범위를 포함합니다. REST API에 대한 Amazon Cognito 사용자 풀 권한 부여자는 진입 장벽이 낮은 일반적인 구현입니다. 이 유형의 권한 부여자에 대한 요청의 본문, 쿼리 문자열 파라미터 및 헤더를 평가할 수도 있습니다.

- ID 토큰이 유효하며 만료되지 않았습니다. Amazon Cognito 권한 부여자에게 ID 토큰을 전달하면 애플리케이션 서버의 ID 토큰 콘텐츠에 대한 추가 검증을 수행할 수 있습니다.
- 액세스 또는 ID 토큰의 그룹, 클레임, 속성 또는 역할은 Lambda 함수에서 정의한 요구 사항을 충족합니다. Lambda 권한 부여자는 요청 헤더에서 토큰을 구문 분석하고 권한 부여 결정을 위해 평가합니다. 함수에서 사용자 지정 로직을 구성하거나 Amazon Verified Permissions에 API를 요청할 수 있습니다.

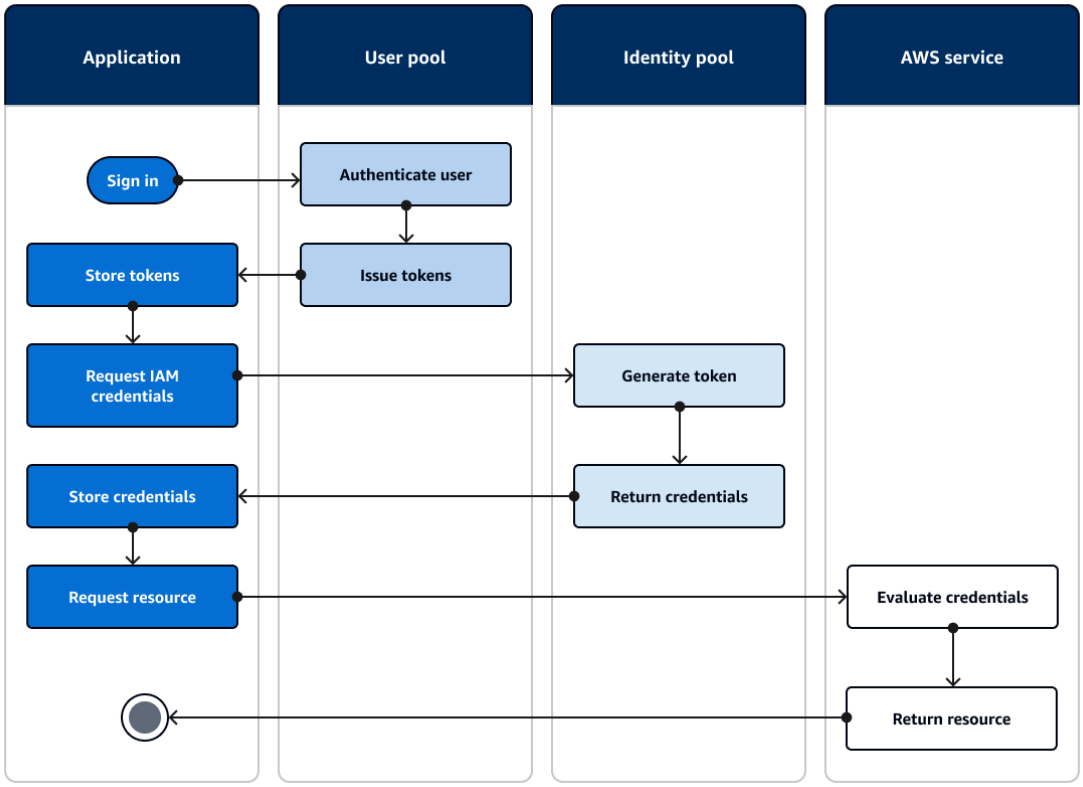
사용자 풀의 토큰을 사용하여 AWS AppSync GraphQL API에 대한 요청을 승인할 수도 있습니다.

로그인 후 자격 증명 풀을 AWS 서비스 사용하여 액세스

사용자는 사용자 풀로 로그인한 후 자격 증명 풀에서 발급된 임시 API 자격 증명 AWS 서비스 으로에 액세스할 수 있습니다.

웹 또는 모바일 앱은 사용자 풀에서 토큰을 수신합니다. 사용자 풀을 자격 증명 풀의 자격 증명 공급자로 구성하면 자격 증명 풀이 토큰을 임시 AWS 자격 증명으로 교환합니다. 이러한 자격 증명은 IAM 역할 및 사용자에게 제한된 AWS 리소스 세트에 대한 액세스 권한을 부여하는 정책으로 범위가 지정될 수 있습니다. 자세한 내용은 자격 증명 풀 인증 흐름 섹션을 참조하세요.

다음 다이어그램은 애플리케이션이 사용자 풀로 로그인하고, ID 풀 자격 증명을 검색하고, AWS 서비스에서 자산을 요청하는 방법을 보여줍니다.



ID 풀 자격 증명을 사용하여 다음을 수행할 수 있습니다.

- 사용자의 자체 자격 증명을 사용하여 Amazon Verified Permissions에 세분화된 권한 부여 요청을 합니다.
- IAM과의 연결을 승인하는 Amazon API Gateway REST API 또는 an AWS AppSync GraphQL API에 연결합니다.
- IAM과의 연결을 승인하는 Amazon DynamoDB 또는 Amazon RDS와 같은 데이터베이스 백엔드에 연결합니다.
- Amazon S3 버킷에서 애플리케이션 자산을 검색합니다.
- Amazon WorkSpaces 가상 데스크톱으로 세션을 시작합니다.

ID 풀은 사용자 풀이 있는 인증된 세션 내에서만 작동하지 않습니다. 또한 타사 ID 제공업체로부터 직접 인증을 수락하고 인증되지 않은 게스트 사용자에 대한 자격 증명을 생성할 수 있습니다.

사용자 풀 그룹과 함께 자격 증명 풀을 사용하여 AWS 리소스에 대한 액세스를 제어하는 방법에 대한 자세한 내용은 사용자 풀에 그룹 추가 및 섹션을 참조하세요역할 기반 액세스 제어 사용. 또한 자격 증명 풀 및에 대한 자세한 내용은 섹션을 AWS Identity and Access Management참조하세요자격 증명 풀 인증 흐름.

를 사용하여 사용자 풀 설정 AWS Management Console

Amazon Cognito 사용자 풀을 생성하고 사용자 풀 ID와 각 클라이언트 앱의 앱 클라이언트 ID를 기록해 둡니다. 사용자 풀 생성에 대한 자세한 내용은 Amazon Cognito 콘솔에서 새 애플리케이션 생성 섹션을 참조하세요.

를 사용하여 자격 증명 풀 설정 AWS Management Console

다음 절차에서는 AWS Management Console 를 사용하여 자격 증명 풀을 하나 이상의 사용자 풀 및 클라이언트 앱과 통합하는 방법을 설명합니다.

Amazon Cognito 사용자 풀 ID 제공업체(idP)를 추가하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. ID 제공업체 추가를 선택합니다.
4. Amazon Cognito 사용자 풀을 선택합니다.
5. 사용자 풀 ID와 앱 클라이언트 ID를 입력합니다.
6. Amazon Cognito가 이 공급자를 통해 인증된 사용자에게 보안 인증을 발급할 때 요청하는 역할을 설정하려면 역할 설정을 구성하세요.

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

        i. 규칙으로 역할 선택을 선택한 경우 사용자 인증의 소스 클레임, 클레임을 규칙과 비교하는 데 사용할 연산자,이 역할 선택과 일치시키는 값, 역할 할당이 일치할 때 할당할 역할을 입력합니다. 다른 조건에 따라 추가 규칙을 생성하려면 다른 항목 추가를 선택합니다.

        ii. 토큰에서 preferred_role 클레임을 포함한 역할 선택을 선택한 경우 Amazon Cognito는 사용자의 `cognito:preferred_role` 클레임에서 해당 역할에 대한 자격 증명을 발급합니다. 기본 역할 클레임이 없는 경우 Amazon Cognito는 역할 해결을 기반으로 보안 인증 정보를 발급합니다.

    b. 역할 해결을 선택합니다. 사용자의 클레임이 규칙과 일치하지 않는 경우 보안 인증을 거부하거나 인증된 역할의 보안 인증을 발급할 수 있습니다.
7. Amazon Cognito가 이 공급자를 통해 인증한 사용자에게 보안 인증을 발급할 때 할당하는 보안 주체 태그를 변경하려면 액세스 제어를 위한 속성을 구성합니다.

- 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.
- sub 및 aud 클레임 기반 보안 주체 태그를 적용하려면 기본 매핑 사용을 선택합니다.
- 보안 주체 태그에 대한 속성의 자체 사용자 지정 스키마를 생성하려면 사용자 지정 매핑 사용을 선택합니다. 그런 다음 태그에 표시하려는 각 클레임에서 소싱하려는 태그 키를 입력합니다.

8. 변경 사항 저장을 선택합니다.

자격 증명 풀과 사용자 풀 통합

앱 사용자가 인증되면 자격 증명 공급자의 로그인 맵에 해당 사용자의 자격 증명 토큰을 추가합니다. 공급자 이름은 Amazon Cognito 사용자 풀 ID에 따라 다릅니다. 이 변수의 구조는 다음과 같습니다.

```
cognito-idp.<region>.amazonaws.com/<YOUR_USER_POOL_ID>
```

사용자 풀 ID 에서 *<region>* 값을 도출할 수 있습니다. 예를 들어 사용자 풀 ID가 us-east-1_EXAMPLE1인 경우 *<region>*은 us-east-1입니다. 사용자 풀 ID가 us-west-2_EXAMPLE2인 경우 *<region>*은 us-west-2입니다.

JavaScript

```
import { fromCognitoIdentityPool } from "@aws-sdk/credential-providers";

var cognitoUser = userPool.getCurrentUser();

if (cognitoUser != null) {
 cognitoUser.getSession(function(err, result) {
  if (result) {
   console.log('You are now logged in.');

   // Create a credentials provider with the user's ID token in the logins map.
   const region = '<region>';
   const credentials = fromCognitoIdentityPool({
    identityPoolId: 'YOUR_IDENTITY_POOL_ID',
    logins: {
     'cognito-idp.<region>.amazonaws.com/<YOUR_USER_POOL_ID>':
 result.getIdToken().getJwtToken()
    },
    clientConfig: { region }
   });
```

```
   // Attach the provider to each service client that needs credentials.
   const client = new SomeServiceClient({ region, credentials });
  }
 });
}
```

Android

```
cognitoUser.getSessionInBackground(new AuthenticationHandler() {
 @Override
 public void onSuccess(CognitoUserSession session) {
  String idToken = session.getIdToken().getJWTToken();

  Map<String, String> logins = new HashMap<String, String>();
  logins.put("cognito-idp.<region>.amazonaws.com/<YOUR_USER_POOL_ID>",
 session.getIdToken().getJWTToken());
  credentialsProvider.setLogins(logins);
 }

});
```

iOS - objective-C

```
AWSServiceConfiguration *serviceConfiguration = [[AWSServiceConfiguration alloc]
 initWithRegion:AWSRegionUSEast1 credentialsProvider:nil];
AWSCognitoIdentityUserPoolConfiguration *userPoolConfiguration =
 [[AWSCognitoIdentityUserPoolConfiguration alloc] initWithClientId:@"YOUR_CLIENT_ID"
  clientSecret:@"YOUR_CLIENT_SECRET" poolId:@"YOUR_USER_POOL_ID"];
[AWSCognitoIdentityUserPool
 registerCognitoIdentityUserPoolWithConfiguration:serviceConfiguration
 userPoolConfiguration:userPoolConfiguration forKey:@"UserPool"];
AWSCognitoIdentityUserPool *pool = [AWSCognitoIdentityUserPool
 CognitoIdentityUserPoolForKey:@"UserPool"];
AWSCognitoCredentialsProvider *credentialsProvider = [[AWSCognitoCredentialsProvider
 alloc] initWithRegionType:AWSRegionUSEast1 identityPoolId:@"YOUR_IDENTITY_POOL_ID"
 identityProviderManager:pool];
```

iOS - swift

```
let serviceConfiguration = AWSServiceConfiguration(region: .USEast1,
 credentialsProvider: nil)
let userPoolConfiguration = AWSCognitoIdentityUserPoolConfiguration(clientId:
 "YOUR_CLIENT_ID", clientSecret: "YOUR_CLIENT_SECRET", poolId: "YOUR_USER_POOL_ID")
```

```
AWSCognitoIdentityUserPool.registerCognitoIdentityUserPoolWithConfiguration(serviceConfigura
 userPoolConfiguration: userPoolConfiguration, forKey: "UserPool")
let pool = AWSCognitoIdentityUserPool(forKey: "UserPool")
let credentialsProvider = AWSCognitoCredentialsProvider(regionType: .USEast1,
 identityPoolId: "YOUR_IDENTITY_POOL_ID", identityProviderManager:pool)
```

사용자 풀 JSON 웹 토큰(JWT) 이해

토큰은 애플리케이션이 OIDC 인증 증명으로 사용하고 리소스에 대한 액세스를 요청하는 데 사용할 수 있는 인증 아티팩트입니다. 토큰의 클레임은 사용자에 대한 정보입니다. ID 토큰에는 사용자 이름, 성, 이메일 주소와 같은 사용자의 신원에 대한 클레임이 포함되어 있습니다. 액세스 토큰에는 인증된 사용자가 서드 파티 API, Amazon Cognito 사용자 셀프 서비스 API 작업, userInfo 엔드포인트에 액세스하는 데 사용할 수 있는 `scope`과 같은 클레임이 포함되어 있습니다. 액세스 토큰과 ID 토큰에는 모두 사용자 풀의 사용자 그룹 멤버십이 포함된 `cognito:groups` 클레임이 포함되어 있습니다. 사용자 풀 그룹에 대한 자세한 내용은 사용자 풀에 그룹 추가 섹션을 참조하세요.

Amazon Cognito에는 새 토큰을 얻거나 기존 토큰을 취소하는 데 사용할 수 있는 새로 고침 토큰도 있습니다. 토큰을 새로 고쳐 새 ID와 액세스 토큰을 가져옵니다. 토큰을 취소하여 새로 고침 토큰을 통해 부여되는 사용자 액세스 권한을 취소할 수 있습니다.

Amazon Cognito는 base64url로 인코딩된 문자열으로 토큰을 발급합니다. 모든 Amazon Cognito ID 또는 액세스 토큰을 `base64url`에서 일반 텍스트 JSON으로 디코딩할 수 있습니다. Amazon Cognito 새로 고침 토큰은 암호화되므로 사용자 풀 사용자 및 관리자가 읽을 수 없으며 사용자 풀만 읽을 수 있습니다.

토큰으로 인증

사용자가 앱에 로그인하면 Amazon Cognito가 로그인 정보를 확인합니다. 성공적으로 로그인하면 Amazon Cognito가 세션을 생성하고 인증된 사용자를 위한 ID 토큰, 액세스 토큰 및 새로 고침 토큰을 반환합니다. 이 토큰을 사용하여 Amazon API Gateway와 같은 API 및 다운스트림 리소스에 대한 액세스 권한을 사용자에게 부여할 수 있습니다. 또는 임시 AWS 보안 인증 정보로 다른 AWS 서비스에 액세스할 수 있도록 이 토큰을 교환할 수 있습니다.



토큰 저장

앱이 다양한 크기의 토큰을 저장할 수 있어야 합니다. 토큰 크기는 추가 클레임, 인코딩 알고리즘의 변경 및 암호화 알고리즘의 변경을 포함하되 이에 국한되지 않는 이유로 변경될 수 있습니다. 사용자 풀에서 토큰 취소를 활성화하면 Amazon Cognito가 JSON 웹 토큰에 추가 클레임을 추가하여 크기를 늘

립니다. 새 클레임 `origin_jti` 및 `jti`가 액세스 토큰과 ID 토큰에 추가됩니다. 토큰 취소에 대한 자세한 내용은 토큰 철회를 참조하세요.

> **Important**
>
> 모범 사례는 전송 중인 토큰 및 애플리케이션 컨텍스트의 스토리지에 있는 모든 토큰을 보호하는 것입니다. 토큰에는 사용자에 대한 개인 식별 정보와 사용자 풀에 사용하는 보안 모델에 대한 정보가 포함될 수 있습니다.

토큰 사용자 지정

Amazon Cognito가 앱에 전달하도록 액세스 및 ID 토큰을 사용자 지정할 수 있습니다. 사전 토큰 생성 Lambda 트리거에서는 토큰 클레임을 추가, 수정 및 차단할 수 있습니다. 사전 토큰 생성 트리거는 Amazon Cognito가 기본 클레임 세트를 보내는 Lambda 함수입니다. 클레임에는 OAuth 2.0 범위, 사용자 풀 그룹 멤버십, 사용자 속성 등이 포함됩니다. 그러면 함수를 런타임에서 변경하고 업데이트된 토큰 클레임을 Amazon Cognito에 반환할 수 있습니다.

버전 2 이벤트를 사용한 토큰 사용자 지정에 액세스하는 경우 추가 비용이 적용됩니다. 자세한 내용은 Amazon Cognito 요금을 참조하세요.

주제

- 자격 증명(ID) 토큰 이해
- 액세스 토큰 이해
- 사용자 풀 토큰을 안전하게 구문 분석
- 새로 고침 토큰
- 토큰 취소로 사용자 세션 종료
- JSON 웹 토큰 확인
- 사용자 풀 토큰 만료 및 캐싱 관리
- 범위, M2M 및 리소스 서버

자격 증명(ID) 토큰 이해

ID 토큰은 `name`, `email`, `phone_number` 등 인증된 사용자의 자격 증명에 대한 클레임이 포함된 JSON 웹 토큰(JWT)입니다. 애플리케이션 내에서 이 자격 증명 정보를 사용할 수 있습니다. 리소스 서

버 또는 서버 애플리케이션에 사용자를 인증하는 경우에도 ID 토큰을 사용할 수 있습니다. 애플리케이션 외부에서 웹 API 작업에 ID 토큰을 사용할 수도 있습니다. 이러한 경우 ID 토큰의 서명을 확인해야 ID 토큰 내의 모든 클레임을 신뢰할 수 있습니다. JSON 웹 토큰 확인 섹션을 참조하세요.

ID 토큰 만료는 5분에서 1일 사이로 설정할 수 있습니다. 앱 클라이언트별로 이 값을 설정할 수 있습니다.

ID 토큰에는 사용자가 만족한 인증 수준과 완료한 방법을 설명하는 `acr` 및 `amr` 클레임이 포함됩니다. 자세한 내용은 ACR 및 AMR 토큰 클레임 섹션을 참조하세요.

> **Important**
>
> 사용자가 관리형 로그인으로 로그인하면 Amazon Cognito는 1시간 동안 유효한 세션 쿠키를 설정합니다. 애플리케이션에서 인증을 위해 관리형 로그인을 사용하고 액세스 및 ID 토큰에 대한 최소 기간을 1시간 미만으로 지정하는 경우, 쿠키가 만료될 때까지 사용자는 유효한 세션을 유지합니다. 사용자에게 1시간 세션 동안 만료되는 토큰이 있는 경우 사용자는 재인증 없이 토큰을 새로 고칠 수 있습니다.

ID 토큰 헤더

헤더에는 키 ID(`kid`) 및 알고리즘(`alg`) 등 두 가지 정보가 포함되어 있습니다.

```
{
"kid" : "1234example=",
"alg" : "RS256"
}
```

`kid`

키 ID입니다. 이 값은 토큰의 JSON 웹 서명(JWS)을 보호하는 데 어떤 키가 사용되었는지 나타냅니다. `jwks_uri` 엔드포인트에서 사용자 풀 서명 키 ID를 볼 수 있습니다.

`kid` 파라미터에 대한 자세한 내용은 키 식별자(kid) 헤더 파라미터를 참조하세요.

`alg`

Amazon Cognito가 액세스 토큰을 보호하는 데 사용한 암호화 알고리즘입니다. 사용자 풀은 SHA-256에서의 RSA 서명인 RS256 암호화 알고리즘을 사용합니다.

`alg` 파라미터에 대한 자세한 내용은 알고리즘(alg) 헤더 파라미터를 참조하세요.

ID 토큰 기본 페이로드

ID 토큰의 페이로드 예제입니다. 여기에는 인증된 사용자에 대한 클레임이 포함되어 있습니다. OpenID Connect(OIDC) 표준 클레임에 대한 자세한 내용은 OIDC 표준 클레임 목록을 참조하세요. 사전 토큰 생성 Lambda 트리거을 사용하여 자체 설계의 클레임을 추가할 수 있습니다.

이 페이로드는 전체가 아니라 대표적입니다. Amazon Cognito ID 토큰의 클레임 세트는 새 기능이 새 클레임을 추가함에 따라 시간이 지남에 따라 증가하므로 토큰이 전달하는 클레임은이 예제와 다를 수 있습니다. Amazon Cognito 토큰을 구문 분석하는 코드를 작성하기 이 토큰을 안전하게 구문 분석하는 방법 전에이 예제를 완전하고 수정된 목록 대신 일반적인 클레임의 스냅샷으로 취급하고 읽습니다.

```
<header>.{
    "sub": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
    "cognito:groups": [
        "test-group-a",
        "test-group-b",
        "test-group-c"
    ],
    "email_verified": true,
    "cognito:preferred_role": "arn:aws:iam::111122223333:role/my-test-role",
    "iss": "https://cognito-idp.us-west-2.amazonaws.com/us-west-2_example",
    "cognito:username": "my-test-user",
    "middle_name": "Jane",
    "nonce": "abcdefg",
    "origin_jti": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
    "cognito:roles": [
        "arn:aws:iam::111122223333:role/my-test-role"
    ],
    "aud": "xxxxxxxxxxxxexample",
    "identities": [
        {
            "userId": "amzn1.account.EXAMPLE",
            "providerName": "LoginWithAmazon",
            "providerType": "LoginWithAmazon",
            "issuer": null,
            "primary": "true",
            "dateCreated": "1642699117273"
        }
    ],
    "event_id": "64f513be-32db-42b0-b78e-b02127b4f463",
```

```
    "token_use": "id",
    "auth_time": 1676312777,
    "exp": 1676316377,
    "iat": 1676312777,
    "jti": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
    "email": "my-test-user@example.com"
}
.<token signature>
```

sub

인증된 사용자에 대한 고유 식별자(UUID) 또는 제목입니다. 사용자 풀에서 사용자 이름이 고유하지 않을 수 있습니다. sub 클레임은 특정 사용자를 식별하는 가장 좋은 방법입니다.

> **Important**
>
> Amazon Cognito는 RFC UUIDsub를 포함한 특정 UUID 형식을 준수하지 않는 Amazon Cognito별 형식으로를 생성합니다. 형식을 엄격하게 검증해서는 안 됩니다sub. 보다 일반적으로 토큰 클레임을 안전하게 구문 분석하는 방법에 대한 지침에 설명된 대로이 페이지에서 설명하는 유형으로 각 클레임을 읽고 설명서에서 설명하는 것보다 더 엄격한 형식을 적용하지 않습니다.

cognito:groups

문자열 배열입니다. 각 문자열은 사용자가 멤버인 사용자 풀 그룹의 이름입니다. 그룹은 앱에 표시하는 식별자가 될 수도 있고 자격 증명 풀에서 선호하는 IAM 역할에 대한 요청을 생성할 수도 있습니다. 이 클레임은 사용자가 단일 그룹에 속하더라도 항상 배열입니다.

cognito:preferred_role

사용자의 우선순위가 가장 높은 사용자 풀 그룹과 연결한 IAM 역할의 ARN입니다. 사용자 풀이 이 역할 클레임을 선택하는 방법에 대한 자세한 내용은 그룹에 우선 순위 값 할당 섹션을 참조하세요.

iss

단일 문자열입니다. 토큰의 발행자입니다. 이 클레임은 토큰을 생성한 사용자 풀을 식별합니다. 애플리케이션은이 값이 사용자 풀의 예상 발급자 URL과 일치하는지 확인해야 합니다. 클레임의 형식은 다음과 같습니다.

```
https://cognito-idp.<Region>.amazonaws.com/<your user pool ID>
```

사용자 풀은 원본 또는 업데이트된 발급자를 사용할 수 있습니다. 업데이트된 발급자는 여러 리전에서 동일한 JWKS 콘텐츠를 호스팅하여 복원력과 효율성을 향상시킵니다. 자세한 내용은 OIDC 발급자로서의 Amazon Cognito 사용자 풀 섹션을 참조하세요.

`cognito:username`

사용자 풀에 있는 사용자의 사용자 이름입니다.

`nonce`

nonce 클레임은 OAuth 2.0 `authorize` 엔드포인트에 대한 요청에 추가할 수 있는 동일한 이름의 파라미터에서 비롯됩니다. 파라미터를 추가하면 `nonce` 클레임은 Amazon Cognito가 발행하는 ID 토큰에 포함되어 있으며, 이를 사용하여 재생 공격으로부터 보호할 수 있습니다. 요청에 `nonce` 값을 제공하지 않을 경우, 사용자가 서드 파티 자격 증명 공급자를 통해 인증할 때 Amazon Cognito는 자동으로 논스를 생성 및 검증한 다음 ID 토큰에 대한 `nonce` 클레임으로 추가합니다. Amazon Cognito에서 `nonce` 클레임 구현은 OIDC 표준에 기반합니다.

`origin_jti`

사용자의 새로 고침 토큰과 연결된 토큰 취소 식별자입니다. Amazon Cognito는 취소 엔드포인트 또는 RevokeToken API 작업을 통해 사용자의 토큰을 취소했는지 확인할 때 `origin_jti` 클레임을 참조합니다. 토큰을 취소하면 Amazon Cognito는 동일한 `origin_jti` 값을 가진 모든 액세스 및 ID 토큰을 무효화합니다.

`cognito:roles`

문자열 배열입니다. 각 문자열은 사용자 그룹 중 하나와 연결된 IAM 역할의 ARN입니다. 모든 사용자 풀 그룹에는 하나의 IAM 역할을 연결할 수 있습니다. 이 배열은 우선 순위에 관계없이 사용자 그룹의 모든 IAM 역할을 나타내며, 단일 역할을 포함하는 경우에도 항상 배열입니다. 자세한 내용은 사용자 풀에 그룹 추가 섹션을 참조하세요.

`aud`

단일 문자열입니다. 사용자를 인증한 사용자 풀 앱 클라이언트입니다. Amazon Cognito는 액세스 토큰 `client_id` 클레임에서 동일한 값을 렌더링합니다.

`identities`

객체 어레이. 각 객체는 페더레이션 로그인 또는 페더레이션 사용자를 로컬 프로필에 연결하여 사용자에게 연결한 하나의 타사 자격 증명 공급자 프로필을 설명합니다. 각 객체에는 필드`userId`(공급자의 사용자 고유 ID), `providerName`, `providerType`, `issuer`, 및 `primary`가 포함됩니다`dateCreated`. 이러한 모든 필드는 문자열이며, 세밀하게 구문 분석할 두 개의 직렬화 세부 정보가 있습니다. `primary`는 JSON 부울이 `"false"`아닌 문자열 `"true"`

또는 이고, `dateCreated`는 숫자가 아닌 Unix 시간을 밀리초 단위로 유지하는 문자열(예: `"1642699117273"`)입니다. `issuer` 필드는 일 수 있습니다`null`. 이 클레임은 사용자에게 연결된 자격 증명이 하나 있더라도 항상 배열입니다.

`token_use`

토큰의 의도된 목적입니다. ID 토큰에서 값은 `id`입니다.

`auth_time`

수입니다. 사용자가 인증을 완료한 Unix 시간 형식의 인증 시간입니다.

`exp`

수입니다. 사용자의 토큰이 만료되는 Unix 시간 형식의 만료 시간입니다.

`iat`

수입니다. Amazon Cognito가 사용자 토큰을 발급한 Unix 시간 형식의 동시 발급입니다.

`jti`

JWT의 고유 식별자입니다.

`email`

단일 문자열입니다. `email` 속성이 존재하고 앱 클라이언트가 읽을 수 있는 경우 사용자의 이메일 주소입니다.

ID 토큰에는 OIDC 표준 클레임에 정의된 OIDC 표준 클레임이 포함될 수 있습니다. 또한 ID 토큰에는 사용자 풀에 정의된 사용자 지정 속성도 포함될 수 있습니다. Amazon Cognito는 ID 토큰에 사용자 지정 속성 값을 속성 유형에 상관없이 문자열로 씁니다.

> **Note**
>
> 사용자 풀의 사용자 지정 속성은 항상 `custom:`으로 시작합니다.

이 토큰을 안전하게 구문 분석하는 방법

Amazon Cognito ID 토큰에 설정된 클레임은 수정되지 않습니다. 토큰에 대한 클레임은 인증 흐름, 사용자의 그룹 멤버십 및 자격 증명 공급자에 따라 달라집니다. 또한 사용자 풀에서 활성화된 기능에 따라 달라집니다. OpenID Connect 사양에는 `iss`, `sub`, `audexp`, 및 만 필요하며 `iat`모든 Amazon Cognito ID 토큰에 있습니다. 또한 Amazon Cognito는 항상 값과 `token_use` 함께를 추가`id`하므로 액

세스 토큰이 아닌 ID 토큰을 받았는지 확인해야 합니다. 다른 모든 클레임은 사용자가 인증한 방식에 따라 선택 사항으로 취급합니다.

ID 토큰은 여러 Amazon Cognito 토큰 유형 중 하나이며 모든 유형에 동일한 구문 분석 규칙이 적용됩니다. 토큰을 안전하게 구문 분석하는 방법에 대한 자세한 내용은 섹션을 참조하세요사용자 풀 토큰을 안전하게 구문 분석.

시나리오별 ID 토큰 예제

는 여러 선택적 클레임을 단일 예제로 ID 토큰 기본 페이로드 결합합니다. 실제로 토큰에 대한 클레임은 사용자가 로그인한 방식에 따라 달라집니다. 다음 예제는 Amazon Cognito가 네 가지 일반적인 시나리오에 대해 내보내는 클레임 조합을 보여줍니다. 각 예제는 서명 및 직렬화된 JWT가 아닌 디코딩된 페이로드입니다. 실제 토큰은 사용자 풀의 키로 `header.payload.signature`, base64url 인코딩 및 RS256-signed됩니다. `sub`, 사용자 풀 ID 및 타임스탬프 값은 설명 자리 표시자입니다.

기준 권한 부여 코드 ID 토큰

대부분의 애플리케이션이 토큰 엔드포인트에서 권한 부여 코드를 교환한 후 사용하는 자격 증명 토큰입니다. 의 `token_use` 값은 액세스 토큰과 `id` 구분합니다. 클라이언트가 다시 보기 공격을 방지하는 데 도움이 되도록 권한 부여 요청에 `nonce` 파라미터를 전송했기 때문에 `nonce` 클레임이 존재합니다.

```
{
    "sub": "a1b2c3d4-5678-90ab-cdef-1234567890ab",
    "iss": "https://cognito-idp.us-east-1.amazonaws.com/us-east-1_ExAmPlE",
    "aud": "6o7h8i9jexampleclientid",
    "cognito:username": "my-test-user",
    "email": "my-test-user@example.com",
    "email_verified": true,
    "origin_jti": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
    "token_use": "id",
    "auth_time": 1759345200,
    "iat": 1759345200,
    "exp": 1759348800,
    "jti": "11112222-3333-4444-5555-666677778888",
    "nonce": "n-0S6_WzA2Mj"
}
```

그룹 및 역할 멤버십

Amazon Cognito는 사용자가 하나 이상의 사용자 풀 그룹에 속할 때 그룹 및 역할 권한 부여 클레임을 추가합니다. `cognito:groups` 및 `cognito:roles` 클레임은 배열입니다. 단일 요소가 포함된 경우에도 배열로 읽습니다. `cognito:preferred_role`는 단일 문자열입니다.

```
{
    "sub": "a1b2c3d4-5678-90ab-cdef-1234567890ab",
    "iss": "https://cognito-idp.us-east-1.amazonaws.com/us-east-1_ExAmPlE",
    "aud": "6o7h8i9jexampleclientid",
    "cognito:username": "my-test-user",
    "cognito:groups": ["admins", "beta-testers"],
    "cognito:roles": ["arn:aws:iam::111122223333:role/my-test-role"],
    "cognito:preferred_role": "arn:aws:iam::111122223333:role/my-test-role",
    "origin_jti": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
    "token_use": "id",
    "auth_time": 1759345200,
    "iat": 1759345200,
    "exp": 1759348800,
    "jti": "11112222-3333-4444-5555-666677778888"
}
```

페더레이션 로그인

소셜 또는 SAML/OIDC 공급자와 같은 외부 자격 증명 공급자를 통해 로그인한 사용자는 외부 공급자 링크를 설명하는 `identities` 클레임을 전달합니다.

```
{
    "sub": "a1b2c3d4-5678-90ab-cdef-1234567890ab",
    "iss": "https://cognito-idp.us-east-1.amazonaws.com/us-east-1_ExAmPlE",
    "aud": "6o7h8i9jexampleclientid",
    "cognito:username": "LoginWithAmazon_amzn1.account.EXAMPLE",
    "identities": [
        {
            "userId": "amzn1.account.EXAMPLE",
            "providerName": "LoginWithAmazon",
            "providerType": "LoginWithAmazon",
            "issuer": null,
            "primary": "true",
            "dateCreated": "1642699117273"
        }
    ],
    "origin_jti": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
    "token_use": "id",
    "auth_time": 1759345200,
    "iat": 1759345200,
    "exp": 1759348800,
    "jti": "11112222-3333-4444-5555-666677778888"
```

```
}
```

> **Note**
>
> 각 `identities` 객체 내에서 `primary` 및 `dateCreated`는 JSON 부울 및 숫자가 아닌 문자열(`"true"` 및 `"1642699117273"`)로 직렬화됩니다. `identity.primary === true` 같은 비교는 자동으로 false로 평가됩니다. 문자열 `"true"` 또는 `primary`로 `"false"`읽고 Unix 시간을 밀리초 단위로 유지하는 문자열`dateCreated`로 읽습니다.

인증 강도(ACR 및 AMR)

사용자 풀이 사용자가 도달한 인증 수준을 보고하면 Amazon Cognito는 `acr` 및 `amr` 클레임을 추가합니다. 다음 토큰은 가장 높은 수준의 인증 앱의 암호 및 TOTP로 인증한 사용자의 토큰입니다. 각 토큰에는 가능한 `acr` 수준 중 정확히 하나가 포함되어 있습니다. 고정된 레벨 세트, 각 레벨을 충족하는 팩터 조합 및 `amr` 값의 전체 목록은 섹션을 참조하세요ACR 및 AMR 클레임을 사용한 인증 수준.

```
{
    "sub": "a1b2c3d4-5678-90ab-cdef-1234567890ab",
    "iss": "https://cognito-idp.us-east-1.amazonaws.com/us-east-1_ExAmPlE",
    "aud": "6o7h8i9jexampleclientid",
    "cognito:username": "my-test-user",
    "acr": "urn:cognito:loa:4",
    "amr": ["pwd", "otp", "mfa"],
    "origin_jti": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
    "token_use": "id",
    "auth_time": 1759345200,
    "iat": 1759345200,
    "exp": 1759348800,
    "jti": "11112222-3333-4444-5555-666677778888"
}
```

인증 수준을 요청하려면 Essentials 또는 Plus 기능 플랜이 필요합니다. `acr` 및 `amr` 클레임은 Amazon Cognito에서 계산되며 사전 토큰 생성 Lambda 트리거로 수정할 수 없습니다. Amazon Cognito 수준을 NIST 또는 eIDAS와 같은 다른 표준으로 변환하는 것은 애플리케이션의 책임입니다.

ID 토큰 서명

ID 토큰의 서명은 JWT 토큰의 헤더와 페이로드에 따라 계산됩니다. 앱에서 받는 ID 토큰의 클레임을 수락하기 전에 토큰의 서명을 확인하세요. 자세한 내용은 JSON 웹 토큰 확인을 참조하세요JSON 웹 토큰 확인.

액세스 토큰 이해

사용자 풀 액세스 토큰에는 인증된 사용자에 대한 클레임, 사용자 그룹 목록 및 범위 목록이 포함되어 있습니다. 액세스 토큰의 목적은 API 작업을 승인하는 것입니다. 사용자 풀은 액세스 토큰을 받아 사용자 셀프 서비스 작업을 승인합니다. 예를 들어 액세스 토큰을 사용하여 사용자 속성을 추가, 변경 또는 삭제하기 위한 액세스 권한을 사용자에게 부여할 수 있습니다.

사용자 풀에 추가한 사용자 지정 범위에서 파생된 액세스 토큰의 OAuth 2.0 범위를 사용하면 사용자에게 API에서 정보를 검색하도록 권한을 부여할 수 있습니다. 예를 들어, Amazon API Gateway는 Amazon Cognito 액세스 토큰을 통한 권한 부여를 지원합니다. 사용자 풀의 정보로 REST API 권한 부여자를 채울 수 있습니다. 자세한 내용은 Amazon Cognito 사용자 풀을 권한 부여자로 사용하여 REST APIs에 대한 액세스 제어를 참조하세요. Amazon Cognito를 HTTP API의 JWT 권한 부여자로 사용할 수도 있습니다. 자세한 내용은 APIs 사용하여 HTTP API에 대한 액세스 제어를 참조하세요. 사용자 지정 범위가 있는 액세스 토큰을 생성하려면 사용자 풀 퍼블릭 엔드포인트를 통해 요청해야 합니다.

Essentials 또는 Plus 기능 플랜을 사용하면 런타임 시 액세스 토큰에 범위를 추가하는 사전 토큰 생성 Lambda 트리거를 구현할 수도 있습니다. 자세한 내용은 사전 토큰 생성 Lambda 트리거 섹션을 참조하세요.

`openid` 범위의 사용자 액세스 토큰은 userInfo 엔드포인트에서 사용자 속성에 대한 추가 정보를 요청할 수 있는 권한입니다. `userInfo` 엔드포인트의 정보 양은 액세스 토큰의 추가 범위에서 파생됩니다. 예를 들어, 모든 사용자 데이터의 경우 `profile`, 이메일 주소의 경우 `email`입니다.

`aws.cognito.signin.user.admin` 범위가 있는 사용자 액세스 토큰은 사용자 속성을 읽고 쓰고, 인증 요소를 나열하고, 다중 인증(MFA) 기본 설정을 구성하고, 기억된 디바이스를 관리할 수 있는 권한입니다. 액세스 토큰이 이 범위에 부여하는 속성에 대한 액세스 수준은 앱 클라이언트에 할당하는 속성 읽기/쓰기 권한과 일치합니다.

액세스 토큰은 JSON 웹 토큰(JWT)입니다. 액세스 토큰의 헤더는 ID 토큰과 구조가 동일합니다. Amazon Cognito는 ID 토큰을 서명하는 키와는 다른 키를 사용하여 액세스 토큰을 서명합니다. 액세스 키 ID(`kid`) 클레임의 값은 동일한 사용자 세션의 ID 토큰에 있는 `kid` 클레임 값과 일치하지 않습니다. 앱 코드에서 ID 토큰과 액세스 토큰을 독립적으로 확인하세요. 서명을 확인하기 전까지는 액세스 토큰

의 클레임을 신뢰하지 마세요. 자세한 내용은 JSON 웹 토큰 확인 섹션을 참조하세요. 액세스 토큰 만료 시간은 5분~1일 사이로 설정할 수 있습니다. 앱 클라이언트별로 이 값을 설정할 수 있습니다.

> **Important**
>
> 액세스 및 ID 토큰의 경우 관리형 로그인을 사용한다면 최솟값을 1시간 미만으로 지정하지 마세요. 관리형 로그인은 1시간 동안 유효한 브라우저 쿠키를 설정합니다. 액세스 토큰 기간을 1시간 미만으로 구성하면 관리형 로그인 쿠키의 유효성 및 최초 로그인 후 1시간 동안 추가 자격 증명 없이 다시 인증할 수 있는 사용자의 기능에 영향을 미치지 않습니다.

액세스 토큰은 ID 토큰과 동일한 방식으로 추가 클레임을 전달하며 동일한 구문 분석 규칙이 적용됩니다. 토큰을 안전하게 구문 분석하는 방법에 대한 자세한 내용은 섹션을 참조하세요사용자 풀 토큰을 안전하게 구문 분석.

액세스 토큰 헤더

헤더에는 키 ID(`kid`) 및 알고리즘(`alg`) 등 두 가지 정보가 포함되어 있습니다.

```
{
"kid" : "1234example="
"alg" : "RS256",
}
```

`kid`

키 ID입니다. 이 값은 토큰의 JSON 웹 서명(JWS)을 보호하는 데 어떤 키가 사용되었는지 나타냅니다. `jwks_uri` 엔드포인트에서 사용자 풀 서명 키 ID를 볼 수 있습니다.

`kid` 파라미터에 대한 자세한 내용은 키 식별자(kid) 헤더 파라미터를 참조하세요.

`alg`

Amazon Cognito가 액세스 토큰을 보호하는 데 사용한 암호화 알고리즘입니다. 사용자 풀은 SHA-256에서의 RSA 서명인 RS256 암호화 알고리즘을 사용합니다.

`alg` 파라미터에 대한 자세한 내용은 알고리즘(alg) 헤더 파라미터를 참조하세요.

액세스 토큰 기본 페이로드

액세스 토큰에서 나온 샘플 페이로드입니다. 자세한 내용은 JWT 클레임을 참조하세요. 사전 토큰 생성 Lambda 트리거을 사용하여 자체 설계의 클레임을 추가할 수 있습니다.

```
<header>.
{
   "sub":"aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
   "device_key": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
   "cognito:groups":[
      "testgroup"
   ],
   "iss":"https://cognito-idp.us-west-2.amazonaws.com/us-west-2_example",
   "version":2,
   "client_id":"xxxxxxxxxxxxexample",
   "aud": "https://api.example.com",
   "origin_jti":"aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
   "event_id":"aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
   "token_use":"access",
   "scope":"phone openid profile resourceserver.1/appclient2 email",
   "auth_time":1676313851,
   "exp":1676317451,
   "iat":1676313851,
   "jti":"aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
   "username":"my-test-user"
}
.<token signature>
```

sub

인증된 사용자에 대한 고유 식별자(UUID) 또는 제목입니다. 사용자 풀에서 사용자 이름이 고유하지 않을 수 있습니다. sub 클레임은 특정 사용자를 식별하는 가장 좋은 방법입니다.

> **Important**
>
> Amazon Cognito는 RFC UUIDsub를 포함한 특정 UUID 형식을 준수하지 않는 Amazon Cognito별 형식으로를 생성합니다. 형식을 엄격하게 검증해서는 안 됩니다sub.

cognito:groups

사용자를 멤버로 가진 사용자 풀 그룹의 이름 배열입니다.

`iss`

토큰의 발행자입니다. 이 클레임은 토큰을 생성한 사용자 풀을 식별합니다. 애플리케이션은이 값이 사용자 풀의 예상 발급자 URL과 일치하는지 확인해야 합니다. 클레임의 형식은 다음과 같습니다.

`https://cognito-idp.us-east-1.amazonaws.com/us-east-1_EXAMPLE`

사용자 풀은 원본 또는 업데이트된 발급자를 사용할 수 있습니다. 업데이트된 발급자는 여러 리전에서 동일한 JWKS 콘텐츠를 호스팅하여 복원력과 효율성을 향상시킵니다. 자세한 내용은 OIDC 발급자로서의 Amazon Cognito 사용자 풀 섹션을 참조하세요.

`client_id`

사용자를 인증한 사용자 풀 앱 클라이언트입니다. Amazon Cognito는 ID 토큰 `aud` 클레임에서 동일한 값을 렌더링합니다.

aud

액세스 토큰이 권한을 부여하려는 API의 URL입니다. 애플리케이션이 권한 부여 서버에서 리소스 바인딩을 요청한 경우에만 표시됩니다.

`origin_jti`

사용자의 새로 고침 토큰과 연결된 토큰 취소 식별자입니다. Amazon Cognito는 취소 엔드포인트 또는 RevokeToken API 작업을 통해 사용자의 토큰을 취소했는지 확인할 때 `origin_jti` 클레임을 참조합니다. 토큰을 취소하면 Amazon Cognito는 더 이상 동일한 `origin_jti` 값을 가진 액세스 토큰과 ID 토큰을 검증하지 않습니다.

`token_use`

토큰의 의도된 목적입니다. 액세스 토큰에서 값은 `access`입니다.

`scope`

로그인한 사용자에게 발급된 OAuth 2.0 범위 목록입니다. 범위는 토큰이 외부 API, 사용자 셀프 서비스 작업 및 `userInfo` 엔드포인트의 사용자 데이터에 제공하는 액세스를 정의합니다. Token 엔드포인트의 토큰에는 앱 클라이언트가 지원하는 범위가 포함될 수 있습니다. Amazon Cognito API 로그인의 토큰에는 `aws.cognito.signin.user.admin` 범위만 포함됩니다.

`auth_time`

사용자가 인증을 완료한 인증 시간(Unix 시간 형식)입니다.

`exp`

사용자의 토큰이 만료되는 만료 시간(Unix 시간 형식)입니다.

iat

Amazon Cognito가 사용자의 토큰을 발급한 발급 시간(Unix 시간 형식)입니다.

jti

JWT의 고유 식별자입니다.

username

사용자 풀의 사용자 이름입니다.

추가 리소스

- Amazon Cognito 사용자 풀에서 액세스 토큰을 사용자 지정하는 방법

Machine-to-machine 액세스 토큰 페이로드

M2M(Machinemachine-to-machine) 액세스 토큰은 사용자 대신 애플리케이션에 권한을 부여하기 때문에 페이로드는 앞서 설명한 사용자 액세스 토큰과 다릅니다. M2M 권한 부여에서 `sub` 클레임에는 사용자 식별자가 아닌 토큰을 요청한 앱 클라이언트의 ID가 포함되며 및 `sub` `client_id` 클레임의 값은 동일합니다. 토큰에는 , `token_use` (값 `access`), `iss`, , 등의 표준 클레임과 함께 리소스 서버에 대한 요청을 승인하는 사용자 지정 범위가 포함됩니다`expiatjti`.

M2M 권한 부여에는 사용자가 포함되지 않고 새로 고침 토큰이 발행되지 않으므로 M2M 액세스 토큰은 사용자 액세스 토큰에 포함된 여러 클레임을 생략합니다. 리소스 서버는 다음 클레임을 예상할 수 없습니다.

- 인증된 사용자와 해당 그룹 멤버십을 식별하는 `username` 및 `cognito:groups` 클레임입니다.
- 기억된 사용자 디바이스를 식별하는 `device_key` 클레임입니다.
- M2M 권한 부여가 발급하지 않는 취소 가능한 새로 고침 토큰에 토큰을 연결하는 `origin_jti` 클레임입니다.

액세스 토큰 서명

`.well-known/jwks.json` 엔드포인트에 광고된 키로 서명된 액세스 토큰의 서명은 토큰 헤더 및 페이로드의 무결성을 검증합니다. 액세스 토큰을 사용하여 외부 API에 대한 액세스를 승인하는 경우 항상 API 권한 부여자가 이 서명을 서명한 키와 비교하여 확인하도록 구성합니다. 자세한 내용은 JSON 웹 토큰 확인 섹션을 참조하세요.

사용자 풀 토큰을 안전하게 구문 분석

Amazon Cognito 토큰의 클레임 세트는 고정되지 않습니다. 토큰에 대한 클레임은 인증 흐름, 사용자의 그룹 멤버십, 자격 증명 공급자, 사용자 풀에서 활성화하는 기능에 따라 달라집니다. Amazon Cognito는 시간이 지남에 따라 새로운 클레임을 추가하며, 새 키를 도입하지만이 가이드가 문서화하는 기존 클레임의 이름을 바꾸거나, 유형을 변경하거나, 제거하지는 않습니다. 이러한 규칙은 ID 토큰과 액세스 토큰 모두에 적용됩니다.

자체 토큰 구문 분석기를 작성하지 마십시오. JSON 웹 토큰(JWT) 라이브러리와 토큰이 증가함에 따라 계속 작동하는 JSON 구문 분석기를 사용합니다. 거의 모든 실제 중단을 방지하는 두 가지 규칙은 다음과 같습니다. 인식할 수 없는 클레임을 허용하고 읽은 각 클레임의 유형과 카디널리티를 올바르게 가정합니다.

인식할 수 없는 클레임 무시

애플리케이션에 필요한 클레임을 읽고 나머지는 그대로 둡니다. 토큰에이 가이드에 나열되지 않은 클레임이 포함된 경우 거부하거나 실패하거나 오류를 발생시키지 마세요. 새 클레임은 추가되며 목록에 없는 클레임이 포함된 토큰은 여전히 유효합니다.

문서화된 유형의 각 클레임 읽기

JSON 구문 분석기는 JSON 자체에서 각 값의 기본 유형을 식별하므로 위험은 코드가 해당 값에 대해 가정하는 것입니다. 이 가이드에서는 각 클레임을 유형 및 카디널리티로 취급합니다. 배열로 문서화된 클레임은 현재 단일 값을 보유하고 있더라도 항상 배열입니다. 문자열로 축소하지 마세요. 예를 들어 , `cognito:roles`, `cognito:groupsidentities`는 항상 배열입니다. 다른 사양에서 정의한 클레임에도 동일한 원칙이 적용됩니다. 예를 들어 `amr`는 존재하는 경우 IETF 웹 사이트의 RFC 8176에 정의된 문자열 배열이므로 해당 유형으로 읽습니다.

클레임 형식을 과도하게 검증하지 마세요.

클레임에 필요한 사양을 검증하고 더 이상 검증하지 않습니다. 예를 들어 `sub`는 고유한 문자열이지만 특정 UUID 형식일 필요는 없습니다. 와 같은 사용자 지정 클레임 내의 일부 값은 JSON 부울 또는 숫자`identities`가 아닌 문자열로 직렬화되며 불투명한 값은 특정 길이로 보장되지 않습니다. 설명서 상태보다 더 엄격한 형식을 적용하면 유효한 토큰이 실패합니다.

클레임 수 또는 주문을 가정하지 마십시오.

토큰의 클레임은 어떤 순서로든 나타날 수 있으며, 토큰 간에 시간이 지남에 따라 개수가 변경될 수 있습니다. 각 클레임을 위치보다는 이름으로 처리하고 토큰을 고정 구조가 아닌 일반 JSON 객체로 구문 분석합니다.

토큰 크기 증가 예산

클레임 세트가 커지기 때문에 토큰의 전체 크기도 커집니다. 버퍼, 열 너비, 헤더 또는 쿠키에서 고정 또는 최대 토큰 크기를 가정하지 마십시오.

`openid-configuration` 검색 문서는 동일한 애디티브, 전방 호환 접근 방식을 따릅니다. 문서를 안전하게 구문 분석하는 방법에 대한 자세한 내용은 섹션을 참조하세요검색 문서 변경 준비.

새로 고침 토큰

새로 고침 토큰을 사용하여 새 ID와 액세스 토큰을 검색할 수 있습니다. 기본적으로 새로 고침 토큰은 애플리케이션 사용자가 사용자 풀에 로그인하고 30일이 지나면 만료됩니다. 사용자 풀에 대한 애플리케이션을 생성할 경우 애플리케이션의 새로 고침 토큰 만료를 60분에서 10년 사이의 값으로 설정할 수 있습니다.

새로 고침 토큰으로 새 액세스 및 자격 증명 토큰 받기

Amazon Cognito는 관리형 로그인 권한 부여 코드 흐름 및 API 작업 또는 SDK 메서드를 사용한 성공적인 인증에 대한 응답으로 새로 고침 토큰을 발급합니다. 새로 고침 토큰은 새 ID 및 액세스 토큰과 선택적으로 새 새로 고침 토큰을 반환합니다. 다음과 같은 방법으로 새로 고침 토큰을 사용할 수 있습니다.

GetTokensFromRefreshToken

GetTokensFromRefreshToken API 작업은 유효한 새로 고침 토큰에서 새 ID와 액세스 토큰을 발급합니다. 새로 고침 토큰 교체를 활성화한 경우에도 새 새로 고침 토큰이 생성됩니다.

InitiateAuth 및 AdminInitiateAuth

AdminInitiateAuth 또는 InitiateAuth API 작업에는 `REFRESH_TOKEN_AUTH` 인증 흐름이 포함됩니다. 이 흐름에서는 새로 고침 토큰을 전달하고 새 ID 및 액세스 토큰을 가져옵니다. 새로 고침 토큰 교체가 활성화된 앱 클라이언트에서는 `REFRESH_TOKEN_AUTH`를 사용하여 인증할 수 없습니다.

OAuth 토큰 엔드포인트

도메인이 있는 사용자 풀의 토큰 엔드포인트에는 유효한 새로 고침 토큰에서 새 ID, 액세스 및 선택적으로(새로 고침 토큰 교체 포함) 새로 고침 토큰을 발급하는 `refresh_token` 권한 부여 유형이 있습니다.

새로 고침 토큰 교체

선택적으로 앱 클라이언트에서 새로 고침 토큰 교체를 구성할 수 있습니다. 새로 고침 토큰 교체를 통해 클라이언트는 원래 새로 고침 토큰을 무효화하고 토큰 새로 고침마다 새로운 새로 고침 토큰을 발급할 수 있습니다. 이 설정이 활성화되면 모든 형태의 토큰 새로 고침에서 성공한 각 요청은 새 ID, 액세스 및 새로 고침 토큰을 반환합니다. 이 설정을 비활성화하면 토큰 새로 고침 요청은 새 액세스 및 ID 토큰만 반환하고 원래 새로 고침 토큰은 유효한 상태로 유지됩니다. 새 새로 고침 토큰은 원래 새로 고침 토큰의 남은 기간 동안 유효합니다. 새로 고침 토큰을 교체하거나 원래 새로 고침 토큰을 전달하도록 앱 클라이언트를 구성할 수 있습니다. 짧은 기간 동안 재시도를 허용하려면 원래 새로 고침 토큰의 유예 기간을 최대 60초로 구성할 수도 있습니다.

> **Note**
>
> 교체는 동일한 고정 유효 기간 내에 새로 고침 토큰을 다시 발급합니다. 기간은 연장되지 않습니다. 각 새 새로 고침 토큰은 토큰이 교체되는 횟수에 관계없이 원래 `RefreshTokenValidity` 기간에 남아 있는 시간 동안만 유효합니다. 예를 들어 `RefreshTokenValidity`를 30일로 설정하고 세션이 매일 새로 고침 토큰을 교체한다고 가정해 보겠습니다. 세션은 가장 최근 교체 후 30일이 아니라 첫 번째 로그인 후 30일이 지나도 만료됩니다. 사용자가 해당 기간 이후에도 서명하도록 하려면 다시 인증해야 합니다.

새로 고침 토큰 교체에 대해 알아야 할 사항

- 새로 고침 토큰 교체를 활성화하면 사용자 풀의 JSON 웹 토큰에 새 클레임이 추가됩니다. `origin_jti` 및 `jti` 클레임이 액세스 토큰과 ID 토큰에 추가됩니다. 이러한 클레임으로 인해 JWT의 크기가 커집니다.
- 새로 고침 토큰 교체는 인증 흐름 `REFRESH_TOKEN_AUTH`와 호환되지 않습니다. 새로 고침 토큰 교체를 구현하려면 앱 클라이언트에서 이 인증 흐름을 비활성화하고 GetTokensFromRefreshToken API 작업 또는 이에 상응하는 SDK 방법을 사용하여 토큰 새로 고침 요청을 제출하도록 애플리케이션을 설계해야 합니다.
- 새로 고침 토큰 교체가 비활성화된 상태에서 `GetTokensFromRefreshToken` 또는 `REFRESH_TOKEN_AUTH`를 사용하여 토큰 새로 고침 요청을 완료할 수 있습니다.
- 사용자 풀에서 디바이스 기억하기가 활성 상태인 경우 `GetTokensFromRefreshToken` 요청에 디바이스 키를 제공해야 합니다. 사용자에게 애플리케이션이 초기 인증 요청에서 제출하는 확인된 디바이스 키가 없는 경우 Amazon Cognito는 새 키를 발급합니다. 이 구성에서 토큰을 새로 고치려면 `AuthParameters`에서 토큰을 지정했는지 아니면 인증 응답에서 새 토큰을 수신했는지에 관계없이 디바이스 키를 제공해야 합니다.

- `GetTokensFromRefreshToken` 요청에서 사전 토큰 생성 Lambda 트리거에 `ClientMetadata`를 전달할 수 있습니다. 트리거의 입력 이벤트에 전달되는 이 데이터는 Lambda 함수의 사용자 지정 로직에 사용할 수 있는 추가 컨텍스트를 제공합니다.

보안 모범 사례로 앱 클라이언트에서 새로 고침 토큰 교체를 활성화합니다.

Enable refresh token rotation (console)

다음 절차에서는 앱 클라이언트에 대해 새로 고침 토큰 교체를 켜거나 끕니다. 이 절차에는 기존 앱 클라이언트가 필요합니다. 앱 클라이언트 생성에 대해 자세히 알아보려면 앱 클라이언트를 사용한 애플리케이션별 설정 섹션을 참조하세요.

새로 고침 토큰 교체 활성화 방법

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 자격 AWS 증명을 입력합니다.
2. 사용자 풀(User Pools)을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택합니다.
4. 앱 클라이언트 메뉴로 이동하여 기존 앱 클라이언트를 선택합니다.
5. 페이지의 앱 클라이언트 정보 섹션에서 편집을 선택합니다.
6. 고급 보안 구성에서 새로 고침 토큰 교체 활성화 옵션을 찾습니다.
7. 교체를 활성화하려면 확인란을 선택합니다. 교체를 비활성화하려면 확인란을 선택 취소합니다.
8. 새로 고침 토큰 교체 유예 기간에 교체된 새로 고침 토큰이 취소되기 전의 지연으로 설정할 초 수를 최대 60까지 입력합니다.

Enable refresh token rotation (API)

CreateUserPoolClient 또는 UpdateUserPoolClient API 요청에서 새로 고침 토큰 교체를 구성합니다. 다음 부분 요청 본문은 새로 고침 토큰 교체를 켜고 유예 기간을 10초로 설정합니다.

```
"RefreshTokenRotation" : {
   "Feature" : "ENABLED",
   "RetryGracePeriodSeconds" : 10
}
```

API 및 SDK 토큰 새로 고침

새로 고침 토큰 교체가 활성 상태인지 여부에 따라 새로 고침 토큰을 사용하여 사용자 풀 API로 새 ID 및 액세스 토큰을 가져오는 두 가지 방법이 있습니다. 새로 고침 토큰 교체가 활성화된 앱 클라이언트에서 GetTokensFromRefreshToken API 작업을 사용합니다. 새로 고침 토큰 교체가 없는 앱 클라이언트에서는 AdminInitiateAuth 또는 InitiateAuth API 작업의 REFRESH_TOKEN_AUTH 흐름을 사용합니다.

> **Note**
>
> 사용자는 관리형 로그인 또는 AWS SDKs 및 Amazon Cognito API 작업으로 빌드하는 사용자 지정 애플리케이션에서 사용자 풀로 인증할 수 있습니다. REFRESH_TOKEN_AUTH 흐름 및 GetTokensFromRefreshToken 모두 관리형 로그인 사용자의 토큰 새로 고침을 완료할 수 있습니다. 사용자 지정 애플리케이션의 토큰 새로 고침은 관리형 로그인 세션에 영향을 주지 않습니다. 이러한 세션은 브라우저 쿠키에 설정되며 1시간 동안 유효합니다. GetTokensFromRefreshToken 응답은 새 ID, 액세스 및 선택적으로 새로 고침 토큰을 발급하지만 관리형 로그인 세션 쿠키는 갱신하지 않습니다.
>
> 새로 고침 토큰 교체가 활성화된 앱 클라이언트에서는 REFRESH_TOKEN_AUTH를 사용할 수 없습니다.

GetTokensFromRefreshToken

GetTokensFromRefreshToken은 새로 고침 토큰으로 권한을 부여하는 요청에서 새 ID, 액세스 및 새로 고침 토큰을 반환합니다. 다음은 GetTokensFromRefreshToken에 대한 요청 본문의 예입니다. 이 작업에 대한 요청에서 Lambda 트리거에 클라이언트 메타데이터를 제출할 수 있습니다.

```
{
    "RefreshToken": "eyJjd123abcEXAMPLE",
    "ClientId": "1example23456789",
    "ClientSecret": "myappclientsecret123abc",
    "ClientMetadata": {
      "MyMetadataKey" : "MyMetadataValue"
   },
}
```

AdminInitiateAuth/InitiateAuth

새로 고침 토큰 교체가 비활성 상태일 때 새로 고침 토큰을 사용하려면 AdminInitiateAuth 또는 InitiateAuth API 작업을 사용합니다. AuthFlow 파라미터에 REFRESH_TOKEN_AUTH를 전달합니다. AuthFlow의 AuthParameters 속성에서 사용자의 새로 고침 토큰을 "REFRESH_TOKEN" 값

으로 전달합니다. Amazon Cognito는 API 요청이 모든 챌린지를 전달한 후 새 ID 및 액세스 토큰을 반환합니다.

다음은 `InitiateAuth` 또는 `AdminInitiateAuth` API를 사용한 토큰 새로 고침 요청 본문의 예입니다.

```
{
    "AuthFlow": "REFRESH_TOKEN_AUTH",
    "ClientId": "1example23456789",
    "UserPoolId": "us-west-2_EXAMPLE",
    "AuthParameters": {
        "REFRESH_TOKEN": "eyJjd123abcEXAMPLE",
        "SECRET_HASH": "kT5acwCVrbD6JexhW3EQwnRSe6fLuPTRkEQ50athqv8="
    }
}
```

OAuth 토큰 새로 고침

도메인을 구성한 사용자 풀의 Token 엔드포인트에 새로 고침 토큰을 제출할 수도 있습니다. 요청 본문에 `refresh_token`의 `grant_type` 값과 사용자 새로 고침 토큰의 `refresh_token` 값을 포함합니다.

토큰 엔드포인트에 대한 요청은 새로 고침 토큰 교체가 활성화된 앱 클라이언트와 토큰 교체가 비활성화된 앱 클라이언트에서 사용할 수 있습니다. 새로 고침 토큰 교체가 활성화되면 토큰 엔드포인트가 새 새로 고침 토큰을 반환합니다.

다음은 새로 고침 토큰을 사용한 요청의 예입니다.

```
POST /oauth2/token HTTP/1.1
Host: auth.example.com
Content-Type: application/x-www-form-urlencoded
Authorization: Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw
Content-Length: **

client_id=1example23456789&grant_type=refresh_token&refresh_token=eyJjd123abcEXAMPLE
```

새로 고침 토큰 철회

사용자에게 속한 새로 고침 토큰을 취소할 수 있습니다. 토큰 취소에 대한 자세한 내용은 토큰 취소로 사용자 세션 종료 섹션을 참조하세요.

> **Note**
>
> 새로 고침 토큰을 취소하면 Amazon Cognito가 해당 토큰을 사용한 새로 고침 요청에서 발급한 모든 ID 및 액세스 토큰이 취소됩니다.

현재 로그인한 모든 세션에서 사용자를 로그아웃하려면 GlobalSignOut 또는 AdminUserGlobalSignOut API 요청을 사용하여 모든 토큰을 취소합니다. 사용자가 로그아웃하면 다음과 같은 효과가 발생합니다.

- 사용자의 새로 고침 토큰을 사용하여 사용자에 대한 새 토큰을 가져올 수 없습니다.
- 사용자의 액세스 토큰을 사용하여 토큰으로 권한이 부여된 API 요청을 할 수 없습니다.
- 새 토큰을 가져오려면 사용자가 다시 인증해야 합니다. 관리형 로그인 세션 쿠키는 자동으로 만료되지 않으므로 보안 인증 정보를 입력하라는 추가 메시지 없이 사용자가 세션 쿠키를 사용하여 다시 인증할 수 있습니다. 관리형 로그인 사용자를 로그아웃한 후 Logout 엔드포인트로 리디렉션하면 Amazon Cognito가 세션 쿠키를 지웁니다.

새로 고침 토큰을 사용하면 앱에서 사용자의 세션을 오랫동안 유지할 수 있습니다. 시간이 지남에 따라 사용자는 새로 고침 토큰으로 로그인한 상태로 유지된 일부 애플리케이션의 권한을 취소할 수 있습니다. 단일 세션에서 사용자를 로그아웃하려면 새로 고침 토큰을 취소합니다. 사용자가 인증된 모든 세션에서 로그아웃하려는 경우 GlobalSignOut API 요청을 생성합니다. 앱은 사용자에게 모든 디바이스에서 로그아웃과 같은 선택 사항을 제공할 수 있습니다.는 변경되지 않은, 만료되지 않은, 취소되지 않은 사용자의 유효한 액세스 토큰을 `GlobalSignOut` 수락합니다. 이 API는 토큰으로 권한이 부여되었기 때문에 한 사용자가 이 API를 사용하여 다른 사용자의 로그아웃을 시작할 수 없습니다.

그러나 자격 AWS 증명으로 모든 디바이스에서 사용자를 로그아웃하도록 권한을 부여하는 AdminUserGlobalSignOut API 요청을 생성할 수 있습니다. 관리자 애플리케이션은 AWS 개발자 자격 증명으로이 API 작업을 호출하고 사용자 풀 ID와 사용자의 사용자 이름을 파라미터로 전달해야 합니다. `AdminUserGlobalSignOut` API를 사용하면 사용자 풀의 모든 사용자를 로그아웃할 수 있습니다.

자격 AWS 증명 또는 사용자의 액세스 토큰으로 권한을 부여할 수 있는 요청에 대한 자세한 내용은 섹션을 참조하세요권한 부여 모델별로 그룹화된 API 작업 목록.

토큰 취소로 사용자 세션 종료

다음 방법을 사용하여 새로 고침 토큰과 최종 사용자 세션을 취소할 수 있습니다. 새로 고침 토큰을 취소하면 해당 새로 고침 토큰에서 이전에 발급한 모든 액세스 토큰이 무효화됩니다. 사용자에게 발급된 다른 새로 고침 토큰은 영향을 받지 않습니다.

RevokeToken 작업

RevokeToken은 대화형 로그인의 초기 액세스 토큰을 포함하여 지정된 새로 고침 토큰에 대한 모든 액세스 토큰을 취소합니다. 이 작업은 사용자의 다른 새로 고침 토큰이나 다른 새로 고침 토큰의 ID 및 액세스 토큰 하위 항목에 영향을 주지 않습니다.

취소 엔드포인트

취소 엔드포인트는 지정된 새로 고침 토큰과 새로 고침 토큰이 생성한 모든 ID 및 액세스 토큰을 취소합니다. 또한 이 엔드포인트는 대화형 로그인에서 초기 액세스 토큰을 취소합니다. 이 엔드포인트에 대한 요청은 사용자의 다른 새로 고침 토큰이나 다른 새로 고침 토큰의 ID 및 액세스 토큰 하위 항목에 영향을 주지 않습니다.

GlobalSignOut 작업

GlobalSignOut은 사용자가 액세스 토큰으로 권한을 부여하는 셀프 서비스 작업입니다. 이 작업은 요청하는 사용자의 새로 고침, ID 및 액세스 토큰을 모두 취소합니다.

AdminUserGlobalSignOut 작업

AdminUserGlobalSignOut은 관리자가 IAM 자격 증명으로 권한을 부여하는 서버 측 작업입니다. 이 작업은 대상 사용자의 모든 새로 고침, ID 및 액세스 토큰을 취소합니다.

토큰 취소에 대해 알아야 할 사항

- 새로 고침 토큰을 취소하는 요청에는 토큰을 얻는 데 사용한 것과 동일한 클라이언트 ID가 포함되어야 합니다.
- 사용자 풀 JWT는 토큰이 생성될 때 서명 및 만료 시간이 할당되는 독립형 토큰입니다. 취소된 토큰은 토큰이 필요한 Amazon Cognito API 호출에 사용할 수 없습니다. 그러나 취소된 토큰은 토큰의 서명 및 만료를 확인하는 JWT 라이브러리를 사용하여 확인되는 경우 여전히 유효합니다.
- 새 사용자 풀 클라이언트를 생성하면 토큰 취소가 기본적으로 사용됩니다.
- 토큰 취소가 활성화된 앱 클라이언트에서만 새로 고침 토큰을 취소할 수 있습니다.

- 토큰 취소를 활성화하면 Amazon Cognito JSON 웹 토큰에 새 클레임이 추가됩니다. `origin_jti` 및 `jti` 클레임이 액세스 토큰과 ID 토큰에 추가됩니다. 이러한 클레임은 애플리케이션 클라이언트 액세스 토큰 및 ID 토큰의 크기를 늘립니다.
- 이전에 활성화된 앱 클라이언트에서 토큰 해지를 비활성화하면 취소된 토큰이 다시 활성화되지 않습니다.
- 사용자 계정을 비활성화하면(새로 고침 및 액세스 토큰 취소) 사용자 계정을 다시 활성화해도 취소된 토큰이 활성화되지 않습니다.
- AWS Management Console, AWS CLI또는 API를 사용하여 새 사용자 풀 클라이언트를 AWS 생성하면 토큰 취소가 기본적으로 활성화됩니다.

토큰 취소 사용

기존 사용자 풀 클라이언트의 토큰을 취소하려면 먼저 토큰 취소를 사용하도록 설정해야 합니다. 또는 AWS API를 사용하여 기존 사용자 풀 클라이언트에 AWS CLI 대한 토큰 취소를 활성화할 수 있습니다. 이렇게 하려면 `aws cognito-idp describe-user-pool-client` CLI 명령 또는 `DescribeUserPoolClient` API 작업을 호출하여 앱 클라이언트에서 현재 설정을 검색합니다. 그런 다음 `aws cognito-idp update-user-pool-client` CLI 명령 또는 `UpdateUserPoolClient` API 작업을 호출합니다. 앱 클라이언트의 현재 설정을 포함하고 `EnableTokenRevocation` 파라미터를 `true`로 설정합니다.

Amazon Cognito API 또는 AWS SDK를 사용하여 토큰 취소가 활성화된 앱 클라이언트를 생성하거나 수정하려면 CreateUserPoolClient 또는 UpdateUserPoolClient API 요청에 다음 파라미터를 포함합니다.

```
"EnableTokenRevocation": true
```

Amazon Cognito 콘솔에서 토큰 해지를 구성하려면 사용자 풀의 앱 클라이언트 메뉴에서 앱 클라이언트를 선택합니다. 앱 클라이언트 정보에서 편집 버튼을 선택하고 고급 구성에서 토큰 취소를 활성화하거나 비활성화합니다.

토큰 취소

RevokeToken API 요청(예: `aws cognito-idp revoke-token` CLI 명령)을 사용하여 새로 고침 토큰을 취소할 수 있습니다. 취소 엔드포인트을 사용하여 토큰을 취소할 수도 있습니다. 이 엔드포인트는 사용자 풀에 도메인을 추가한 후에 사용할 수 있습니다. Amazon Cognito 호스트된 도메인 또는 자체 사용자 지정 도메인의 취소 엔드포인트를 사용할 수 있습니다.

다음은 RevokeToken API 요청 예시의 본문입니다.

```
{
   "ClientId": "1example23456789",
   "ClientSecret": "abcdef123456789ghijklexample",
   "Token": "eyJjdHkiOiJKV1QiEXAMPLE"
}
```

다음은 사용자 지정 도메인이 있는 사용자 풀의 /oauth2/revoke 엔드포인트에 대한 cURL 요청의 예입니다.

```
curl --location 'auth.mydomain.com/oauth2/revoke' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--header 'Authorization: Basic Base64Encode(client_id:client_secret)' \
--data-urlencode 'token=abcdef123456789ghijklexample' \
--data-urlencode 'client_id=1example23456789'
```

앱 클라이언트에 클라이언트 보안 암호가 없는 한, RevokeToken 작업 및 /oauth2/revoke 엔드포인트에서 추가 권한 부여가 필요하지 않습니다.

JSON 웹 토큰 확인

JSON 웹 토큰(JWT)은 쉽게 디코딩, 읽기 및 수정할 수 있습니다. 수정된 액세스 토큰은 권한 에스컬레이션 위험을 초래합니다. 수정된 ID 토큰은 사칭 위험을 초래합니다. 애플리케이션은 사용자 풀을 토큰 발급자로 신뢰하지만 사용자가 전송 중인 토큰을 가로채면 어떻게 됩니까? 애플리케이션이 Amazon Cognito가 발급한 것과 동일한 토큰을 수신하고 있는지 확인해야 합니다.

Amazon Cognito는 OpenID Connect(OIDC) 사양의 무결성 및 기밀성 기능 중 일부를 사용하는 토큰을 발행합니다. 사용자 풀 토큰은 만료 시간, 발급자 및 디지털 서명과 같은 개체의 유효성을 나타냅니다. .으로 구분된 JWT의 세 번째 및 마지막 세그먼트인 서명은 토큰 검증의 주요 구성 요소입니다. 악의적인 사용자는 토큰을 수정할 수 있지만 애플리케이션이 퍼블릭 키를 검색하고 서명을 비교하면 일치하지 않습니다. OIDC 인증에서 JWT를 처리하는 모든 애플리케이션은 로그인할 때마다 이 확인 작업을 수행해야 합니다.

이 페이지에서는 JWT 확인을 위한 몇 가지 일반적이고 구체적인 권장 사항을 제시합니다. 애플리케이션 개발은 다양한 프로그래밍 언어와 플랫폼을 포괄합니다. Amazon Cognito는 공개 사양에 충분히 근접한 OIDC를 구현하므로 선택한 개발자 환경의 평판이 좋은 JWT 라이브러리가 인증 요구 사항을 처리할 수 있습니다.

이러한 단계들은 사용자 풀 JSON 웹 토큰(JWT)의 확인 방법을 설명합니다.

주제

- 사전 조건
- aws-jwt-verify를 사용한 토큰 검증
- 토큰 이해 및 검사

사전 조건

라이브러리, SDK 또는 소프트웨어 프레임워크에서 이 섹션의 작업을 이미 처리했을 수 있습니다. AWS SDKs 앱에서 Amazon Cognito 사용자 풀 토큰 처리 및 관리를 위한 도구를 제공합니다. 에는 Amazon Cognito 토큰을 검색하고 새로 고치는 함수가 AWS Amplify 포함되어 있습니다.

자세한 내용은 다음 페이지를 참조하십시오.

- 웹 및 모바일 앱과 Amazon Cognito 인증 및 권한 부여 통합
- AWS SDKs를 사용하는 Amazon Cognito 자격 증명 공급자의 코드 예제
- Amplify 개발 센터의 Advanced workflows(고급 워크플로)

많은 라이브러리를 사용하여 JSON 웹 토큰(JWT)을 디코딩하고 확인할 수 있습니다. 서버 측 API 처리를 위해 토큰을 수동으로 처리하고자 하는 경우 또는 다른 프로그래밍 언어를 사용 중인 경우 이러한 라이브러리가 유용할 수 있습니다. JWT 토큰 작업을 위한 OpenID Foundation 라이브러리 목록을 참조하세요.

aws-jwt-verify를 사용한 토큰 검증

Node.js 앱에서는 사용자가 앱에 전달하는 토큰의 파라미터를 검증하도록 aws-jwt-verify 라이브러리를 AWS 권장합니다. `aws-jwt-verify`를 사용하면 하나 이상의 사용자 풀에 대해 확인하려는 클레임 값으로 `CognitoJwtVerifier`를 채울 수 있습니다. 확인할 수 있는 값에는 다음이 포함됩니다.

- 액세스 또는 ID 토큰의 형식이 잘못되었거나 만료되지 않았으며 유효한 서명이 있음
- 액세스 토큰이 올바른 사용자 풀과 앱 클라이언트에서 제공됨
- 액세스 토큰 클레임에 올바른 OAuth 2.0 범위가 포함됨
- 액세스 및 ID 토큰을 서명한 키가 사용자 풀의 JWKS URI에 있는 `kid` 서명 키와 일치함

  JWKS URI에는 사용자 토큰을 서명한 프라이빗 키에 대한 공개 정보가 포함됩니다. 사용자 풀의 JWKS URI는 `https://cognito-idp.<Region>.amazonaws.com/<userPoolId>/.well-known/jwks.json`에서 찾을 수 있습니다.

Node.js 앱 또는 AWS Lambda 권한 부여자에서 사용할 수 있는 자세한 정보 및 예시 코드는 GitHub의 aws-jwt-verify 섹션을 참조하세요.

> **Note**
>
> 토큰의 `iss` (발급자) 클레임이를 구성한 사용자 풀`CognitoJwtVerifier`과 일치하지 않으면를 사용하여 확인이 실패합니다`ParameterValidationError`. 이는 일반적으로 토큰이 다른 사용자 풀 또는 AWS 리전의 확인자에 제공될 때 발생합니다. 에 `userPoolId` 전달하는이 애플리케이션이 수신하는 토큰의 발급자와 `CognitoJwtVerifier.create()` 일치하는지 확인합니다.

토큰 이해 및 검사

토큰 검사를 앱에 통합하기 전에 Amazon Cognito가 JWT를 어떻게 조합하는지 생각해 보세요. 사용자 풀에서 예시 토큰을 검색하세요. 디코딩하고 자세히 검사하여 특성을 이해하고 언제 무엇을 검증할지 결정하세요. 예를 들어 한 시나리오에서는 그룹 멤버십을 검사하고 또 다른 시나리오에서는 범위를 검사할 수 있습니다.

다음 섹션에서는 앱을 준비하면서 Amazon Cognito JWT를 수동으로 검사하는 프로세스를 설명합니다.

JWT의 구조 확인

JSON 웹 토큰(JWT)에는 사이에 `.` (점) 구분 기호가 있는 세 개의 섹션이 포함되어 있습니다.

헤더

Amazon Cognito가 토큰을 서명하는 데 사용한 키 ID(`kid`) 및 RSA 알고리즘(`alg`)입니다. Amazon Cognito는 `RS256`이라는 `alg`로 토큰을 서명합니다. `kid`는 사용자 풀이 보유한 2048비트 RSA 프라이빗 서명 키에 대한 잘린 참조입니다.

Payload

토큰 클레임입니다. ID 토큰의 클레임에는 사용자 속성 및 사용자 풀(`iss`), 앱 클라이언트(aud)에 대한 정보가 포함됩니다. 액세스 토큰의 페이로드에는 범위, 그룹 멤버십과 포함되며 사용자 풀이 `iss`로, 앱 클라이언트가 `client_id`로 포함됩니다.

서명

서명은 헤더 및 페이로드와 같이 디코딩할 수 있는 base64url이 아닙니다. JWKS URI에서 관찰할 수 있는 서명 키와 파리미터에서 파생된 RSA256 식별자입니다.

헤더와 페이로드는 base64url로 인코딩된 JSON입니다. 시작 문자 {로 디코딩되는 시작 문자 eyJ로 알 수 있습니다. 사용자가 base64url로 인코딩된 JWT를 앱에 제시했는데 형식이 [JSON Header].[JSON Payload].[Signature]가 아닌 경우, 이는 유효한 Amazon Cognito 토큰이 아니므로 폐기해도 됩니다.

다음 예제 애플리케이션은 aws-jwt-verify를 사용하여 사용자 풀 토큰을 확인합니다.

```
// cognito-verify.js
// Usage example: node cognito-verify.js eyJra789ghiEXAMPLE

const { CognitoJwtVerifier } = require('aws-jwt-verify');

// Replace with your Amazon Cognito user pool ID
const userPoolId = 'us-west-2_EXAMPLE';

async function verifyJWT(token) {
  try {
    const verifier = CognitoJwtVerifier.create({
      userPoolId,
      tokenUse: 'access', // or 'id' for ID tokens
      clientId: '1example23456789', // you must verify the token audience
    });

    const payload = await verifier.verify(token);
    console.log('Decoded JWT:', payload);
  } catch (err) {
    console.error('Error verifying JWT:', err);
  }
}

// Example usage
if (process.argv.length < 3) {
  console.error('Please provide a JWT token as an argument.');
  process.exit(1);
}

const MyToken = process.argv[2];
verifyJWT(MyToken);
```

JWT 검증

JWT 서명은 헤더와 페이로드의 해시 조합입니다. Amazon Cognito는 각 사용자 풀에서 RSA 암호화 키를 두 쌍 생성합니다. 하나의 프라이빗 키는 액세스 토큰을 서명하고 다른 하나는 ID 토큰을 서명합니다.

JWT 토큰의 서명을 확인하는 방법

1. ID 토큰을 디코딩합니다.

   또한 OpenID Foundation에는 JWT 토큰 작업을 위한 라이브러리 목록이 포함되어 있습니다.

   AWS Lambda 를 사용하여 사용자 풀 JWTs. 자세한 내용은를 사용하여 Amazon Cognito JWT 토큰 디코딩 및 확인을 참조하세요 AWS Lambda.

2. 로컬 키 ID(`kid`)를 퍼블릭 `kid`와 비교합니다.

   a. 사용자 풀에 해당되는 퍼블릭 JSON 웹 키(JWK)를 다운로드하고 저장합니다. 이 키는 JSON 웹 키 집합(JWKS)의 일부로 사용이 가능합니다. 사용자 환경에 다음 `jwks_uri` URL을 구성하여 찾을 수 있습니다.

   ```
   https://cognito-idp.<Region>.amazonaws.com/<userPoolId>/.well-known/jwks.json
   ```

   JWK 및 JWK 집합에 대한 자세한 내용은 JSON 웹 키(JWK)를 참조하십시오.

   > **Note**
   >
   > Amazon Cognito는 사용자 풀의 서명 키를 교체할 수 있습니다. 가장 좋은 방법은 `kid`를 캐시 키로 사용하여 앱에서 퍼블릭 키를 캐시하고 주기적으로 캐시를 새로 고치는 것입니다. 앱에서 받는 토큰의 `kid`를 캐시와 비교합니다.
   > 발급자는 올바르지만 `kid`가 다른 토큰을 받은 경우 Amazon Cognito가 서명 키를 교체했을 수 있습니다. 사용자 풀 `jwks_uri` 엔드포인트에서 캐시를 새로 고칩니다.

   다음은 샘플 `jwks.json` 파일입니다.

   ```
   {
    "keys": [{
     "kid": "1234example=",
     "alg": "RS256",
   ```

```
  "kty": "RSA",
  "e": "AQAB",
  "n": "1234567890",
  "use": "sig"
 }, {
  "kid": "5678example=",
  "alg": "RS256",
  "kty": "RSA",
  "e": "AQAB",
  "n": "987654321",
  "use": "sig"
 }]
}
```

키 ID(**kid**)

kid는 토큰의 JSON 웹 서명(JWS)을 보호하는 데 어떤 키가 사용되었는지 나타내는 힌트입니다.

알고리즘(**alg**)

alg 헤더 파라미터는 ID 토큰을 보호하는 데 사용되는 암호화 알고리즘을 나타냅니다. 사용자 풀은 SHA-256에서의 RSA 서명인 RS256 암호화 알고리즘을 사용합니다. RSA에 대한 자세한 내용은 RSA 암호화를 참조하세요.

키 유형(**kty**)

kty 파라미터는 이 예제의 'RSA'와 같이 키에서 사용되는 암호화 알고리즘 그룹을 식별합니다.

RSA 지수(**e**)

e 파라미터는 RSA 퍼블릭 키의 지수 값을 포함합니다. 이 값은 Base64urlUInt 인코딩 값으로 표현됩니다.

RSA 모듈러스(**n**)

n 파라미터는 RSA 퍼블릭 키의 모듈러스 값을 포함합니다. 이 값은 Base64urlUInt 인코딩 값으로 표현됩니다.

사용(**use**)

use 파라미터는 퍼블릭 키의 용도를 설명합니다. 이 예제에서 use 값 sig는 서명을 나타냅니다.

b. JWT의 kid와 일치하는 kid에서 퍼블릭 JSON 웹 키를 검색합니다.

클레임 확인

JWT 클레임을 확인하는 방법

1. 토큰이 만료되지 않았는지 확인합니다. 토큰을 디코딩하고 exp 클레임을 현재 시간과 비교합니다. 사용자 풀 JWTs는 독립형이며 자체 만료 시간을 보유하므로 Amazon Cognito에 대한 네트워크 호출 없이 오프라인으로 만료를 확인할 수 있습니다.

   > **Note**
   >
   > GetUser 또는 userInfo 엔드포인트 엔드포인트와 같은 API 작업은 만료된 액세스 토큰에 대한 오류를 반환하지만 거부된 요청은 토큰 확인 방법이 아닌 권한 부여의 부작용입니다. 이러한 네트워크 호출을 사용하여 만료를 확인하지 마십시오. 토큰을 디코딩하고 대신 exp 클레임을 비교합니다.

2. ID 토큰의 aud 클레임과 액세스 토큰의 client_id 클레임은 Amazon Cognito 사용자 풀에서 생성된 앱 클라이언트 ID와 일치해야 합니다.
3. 발급자(iss) 클레임은 사용자 풀과 일치해야 합니다. 예를 들어 us-east-1 리전에서 생성된 사용자 풀의 iss 값은 다음과 같습니다.

   `https://cognito-idp.us-east-1.amazonaws.com/<userpoolID>`.
4. token_use 클레임을 확인합니다.

   - 웹 API 작업에서 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.
5. 토큰이 취소되지 않았는지 확인합니다. 사용자가 로그아웃하거나 RevokeToken API 작업을 호출하는 경우와 같이 만료 시간 전에 유효한 서명과 만료되지 않은 exp 클레임이 있는 토큰은 여전히 취소될 수 있습니다. 오프라인 서명 및 만료 검사는 자체적으로 해지를 감지할 수 없습니다. 토큰이 만료되기 전에 애플리케이션이 해지를 준수해야 하는 경우 승인된 API 요청과 같이 Amazon Cognito에 대해 토큰 상태를 확인하거나 액세스 토큰 수명을 짧게 유지합니다. 자세한 내용은 토큰 취소로 사용자 세션 종료 섹션을 참조하세요.

이제 토큰 내에 있는 클레임을 신뢰할 수 있습니다.

> **Note**
>
> 일부는 애플리케이션 코드 대신 Amazon Cognito JWTs 확인할 AWS 서비스 수 있습니다. Amazon Cognito를 Amazon API Gateway HTTP API의 JWT 권한 부여자로 사용하는 경우 권한 부여자 구성에서 동일한 발급자 및 대상 값을 제공합니다. 발급자를 사용자 풀 `/.well-known/openid-configuration` 문서의 `issuer` 필드 값이기도 `https://cognito-idp.<Region>.amazonaws.com/<userPoolId>`한 사용자 풀 엔드포인트 로 설정합니다. 대상을 앱 클라이언트 ID로 설정합니다.
> API Gateway는 대상을 토큰의 `aud` 클레임과 일치시키거나 토큰에 `client_id` 클레임이 없는 경우 `aud` 클레임과 일치시킵니다. Amazon Cognito ID 토큰은를 전달하고`aud`, 액세스 토큰은 요청에 리소스-서버 바인딩이 포함되지 `aud` 않는 한 `client_id` 및를 전달합니다. ID 토큰 또는 리소스 바인딩이 없는 액세스 토큰의 경우 대상을 앱 클라이언트 ID로 설정합니다. 애플리케이션이 리소스 서버에 바인딩된 액세스 토큰을 요청할 때 토큰의 `aud` 클레임은 앱 클라이언트 ID 대신 리소스 식별자(리소스 서버 URL)로 설정됩니다.이 경우 API Gateway가 일치하도록 리소스 서버 URL을 대상으로 구성합니다. API Gateway 구성 단계는 Amazon Cognito 사용자 풀을 권한 부여자로 사용하여 REST APIs에 대한 액세스 제어 및 API Gateway의 JWT 권한 부여자를 사용하여 HTTP APIs에 대한 액세스 제어를 참조하세요.
> Application Load Balancer는 사용자 풀로 사용자를 인증할 수도 있습니다. 로드 밸런서는 서명된 사용자 클레임을 애플리케이션이 확인하는 JWT 형식 `x-amzn-oidc-data` HTTP 헤더의 대상에 전달합니다. 자세한 내용은 Application Load Balancer를 사용하여 사용자 인증을 참조하세요.

사용자 풀 토큰 만료 및 캐싱 관리

앱은 새 JSON 웹 토큰(JWT)을 가져올 때마다 다음 요청 중 하나를 성공적으로 완료해야 합니다.

- Token 엔드포인트에서 클라이언트 자격 증명 또는 권한 부여 코드 부여를 요청합니다.
- 사용자 풀 도메인이 필요하지 않은 GetClientToken API 작업을 사용하여 클라이언트 자격 증명 액세스 토큰을 요청합니다.
- 관리형 로그인 페이지에서 암시적 권한 부여를 요청합니다.
- InitiateAuth와 같은 Amazon Cognito API 요청에서 로컬 사용자를 인증합니다.

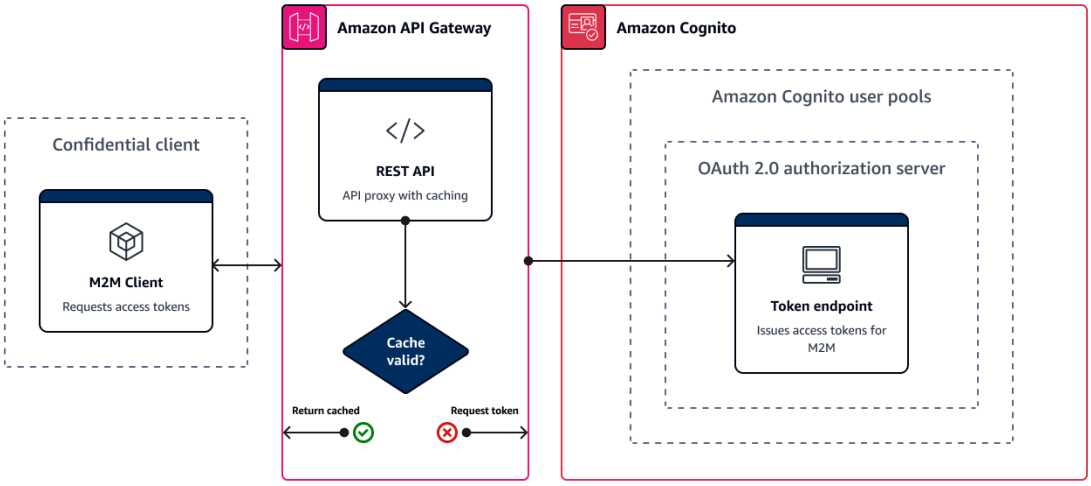
사용자 풀을 구성하여 토큰이 분, 시간 또는 일 단위로 만료되도록 설정할 수 있습니다. 앱의 성능과 가용성을 보장하려면 토큰 수명의 약 75% 동안 Amazon Cognito 토큰을 사용한 다음 새 토큰을 가져와야 합니다. 앱용으로 구축한 캐시 솔루션은 토큰을 계속 사용할 수 있도록 하고 요청 비율이 너무 높을 때 Amazon Cognito의 요청 거부를 방지합니다. 클라이언트 측 앱은 메모리 캐시에 토큰을 저장해야 합니다. 서버 측 앱은 암호화된 캐시 메커니즘을 추가하여 토큰을 저장할 수 있습니다.

사용자 풀이 많은 양의 사용자 또는 머신 간 활동을 생성하는 경우 Amazon Cognito가 수행할 수 있는 토큰 요청 수에 대해 설정하는 제한에 이를 수 있습니다. Amazon Cognito 엔드포인트에 대한 요청 수를 줄이기 위해 인증 데이터를 안전하게 저장 및 재사용하거나 지수 백오프 및 재시도를 구현할 수 있습니다.

인증 데이터는 두 가지 종류의 클래스 엔드포인트에서 제공됩니다. Amazon Cognito OAuth 2.0 엔드포인트에는 클라이언트 자격 증명 및 관리형 로그인 인증 코드 요청 서비스를 제공하는 토큰 엔드포인트가 포함됩니다. 서비스 엔드포인트는 `InitiateAuth` 및 `RespondToAuthChallenge`와 같은 사용자 풀 API 요청에 응답합니다. 각 요청 유형에는 고유한 제한이 있습니다. 제한에 대한 자세한 내용은 Amazon Cognito의 할당량 섹션을 참조하세요.

Amazon API Gateway를 통해 머신 간 액세스 토큰 캐싱

API Gateway 토큰 캐싱을 사용하면 Amazon Cognito OAuth 엔드포인트의 기본 요청 속도 할당량보다 큰 이벤트에 대한 응답으로 앱을 확장할 수 있습니다.



캐시된 토큰이 만료된 경우에만 앱이 새 액세스 토큰을 요청하도록 액세스 토큰을 캐시할 수 있습니다. 그렇지 않으면 캐싱 엔드포인트가 캐시에서 토큰을 반환합니다. 이렇게 하면 Amazon Cognito API 엔드포인트에 대한 추가 호출이 방지됩니다. Amazon API Gateway를 Token 엔드포인트에 대한 프록시로 사용하면 API가 요청 할당량에 기여하는 대부분의 요청에 응답하여 속도 제한으로 인해 실패한 요청을 방지합니다.

다음 API 게이트웨이 기반 솔루션은 짧은 대기 시간, 낮은 코드/코드 없는 토큰 캐싱 구현을 제공합니다. API Gateway API는 전송 중 암호화되며 필요에 따라 미사용 시에도 암호화됩니다. API 게이트웨이 캐시는 머신 간 및 마이크로서비스 세션을 승인하기 위해 액세스 토큰을 생성하는 자주 대용량 부여 유형인 OAuth 2.0 클라이언트 자격 증명 부여에 이상적입니다. 마이크로서비스가 수평적으로 확장되는 트래픽 급증과 같은 경우 사용자 풀 또는 앱 클라이언트의 AWS 요청 속도 제한을 초과하는 볼륨에서 동일한 클라이언트 자격 증명을 사용하는 많은 시스템으로 끝날 수 있습니다. 앱 가용성을 유지하고 지연 시간을 줄이려면 이러한 시나리오에서는 캐싱 솔루션이 가장 좋습니다.

이 솔루션에서는 API에 캐시를 정의하여 앱에서 요청하려는 OAuth 범위 및 앱 클라이언트의 각 조합에 대해 별도의 액세스 토큰을 저장합니다. 앱이 캐시 키와 일치하는 요청을 하면 API는 캐시 키와 일치하는 첫 번째 요청에 대해 Amazon Cognito가 발급한 액세스 토큰으로 응답합니다. 캐시 키 기간이 만료되면 API는 요청을 토큰 엔드포인트로 전달하고 새 액세스 토큰을 캐싱합니다.

> **Note**
>
> 캐시 키 지속 시간은 앱 클라이언트의 액세스 토큰 기간보다 짧아야 합니다.

캐시 키는 요청 본문의 `scope` 파라미터와 요청의 `Authorization` 헤더에서 요청한 OAuth 범위의 조합입니다. 이 `Authorization` 헤더에는 앱 클라이언트 ID와 클라이언트 보안 암호가 포함됩니다. 이 솔루션을 구현하기 위해 앱에서 추가 로직을 구현할 필요가 없습니다. 사용자 풀 토큰 엔드포인트의 경로를 변경하려면 구성만 업데이트해야 합니다.

ElastiCache(Redis OSS)를 사용하여 토큰 캐싱을 구현할 수도 있습니다. AWS Identity and Access Management (IAM) 정책을 사용하여 세밀하게 제어하려면 Amazon DynamoDB 캐시를 사용해 보세요.

> **Note**
>
> API Gateway의 캐싱에는 추가 비용이 부과됩니다. 자세한 내용은 요금을 참조하세요.

API Gateway를 사용하여 캐싱 프록시를 설정하려면 다음과 같이 하세요.

1. API Gateway 콘솔을 열고 REST API를 생성합니다.
2. Resources(리소스)에서 POST 메서드를 생성합니다.
    a. HTTP Integration type(통합 유형)을 선택합니다.
    b. Use HTTP proxy integration(HTTP 프록시 통합 사용)을 선택합니다.

c. `https://<your user pool domain>/oauth2/token`의 Endpoint URL(엔드포인트 URL)을 입력합니다.

3. Resources(리소스)에서 캐시 키를 구성합니다.

   a. POST 메서드의 Method request(메서드 요청)를 편집합니다.

   > **Note**
   >
   > 이 메서드 요청 검증은 클라이언트 보안 암호가 요청 헤더에 인코딩되는 토큰 `Authorization` 요청에서 `client_secret_basic` 권한 부여와 함께 사용됩니다. `client_secret_post` 권한 부여에서 JSON 요청 본문을 검증하려면 client_secret이 있어야 하는 데이터 모델을 대신 생성합니다. 이 모델에서는 요청 검사기가 본문, 쿼리 문자열 파라미터 및 헤더를 검증해야 합니다.

   b. 쿼리 문자열 파라미터와 헤더를 검증하기 위해 요청 검사기 메서드를 구성합니다. 요청 검증에 대한 자세한 내용은 Amazon API Gateway 개발자 가이드의 요청 검증을 참조하세요.

   c. `scope` 파라미터와 `Authorization` 헤더를 캐싱 키로 설정합니다.

      i. URL 쿼리 문자열 파라미터에 쿼리 문자열을 추가합니다. `scope`의 쿼리 문자열 이름을 입력하고 필수 및 캐싱을 선택합니다.

      ii. HTTP 요청 헤더에 헤더를 추가합니다. `Authorization`의 요청 헤더 이름을 입력하고 필수 및 캐싱을 선택합니다.

4. Stages(단계)에서 캐싱을 구성합니다.

   a. 수정할 스테이지를 선택하고 스테이지 세부 정보에서 편집을 선택합니다.

   b. 추가 설정의 캐시 설정에서 프로비저닝 API 캐시 옵션을 켭니다.

   c. Cache capacity(캐시 용량)를 선택합니다. 캐시 용량이 높을수록 성능이 향상되지만 추가 비용이 발생합니다.

   d. Require authorization(권한 부여 필요) 확인란의 선택을 취소합니다. 계속을 선택합니다.

   e. API Gateway는 단계 수준의 GET 메서드에만 캐시 정책을 적용합니다. 캐시 정책 재정의를 POST 메서드에 적용해야 합니다.

      구성한 단계를 확장하고 `POST` 메서드를 선택합니다. 메서드에 대한 캐시 설정을 생성하려면 재정의 생성을 선택합니다.

   f. 메서드 캐시 활성화 옵션을 활성화합니다.

   g. 3,600초의 캐시 TTL(time-to-live)을 입력합니다. 저장을 선택합니다.

5. Stages(단계)에서 Invoke URL(URL 호출)을 기록해 둡니다.
6. 사용자 풀의 `/oauth2/token` 엔드포인트 대신 API의 Invoke URL(URL 호출)에 대한 POST 토큰 요청으로 앱을 업데이트합니다.

범위, M2M 및 리소스 서버

사용자 풀에 대한 도메인을 구성한 후 Amazon Cognito는 OAuth 2.0 권한 부여 서버와 앱에서 사용자에게 표시할 수 있는 등록 및 로그인 페이지가 포함된 호스팅 웹 UI를 자동으로 프로비저닝합니다. 자세한 내용은 사용자 풀 관리형 로그인 섹션을 참조하세요. 권한 부여 서버가 액세스 토큰에 추가할 범위를 선택할 수 있습니다. 범위는 리소스 서버 및 사용자 데이터에 대한 액세스 권한을 부여합니다.

리소스 서버는 OAuth 2.0 API 서버입니다. 액세스 보호 리소스의 보안을 위해 이 서버는 보호하는 API에서 요청된 메서드와 경로를 승인하는 범위가 사용자 풀 액세스 토큰에 포함되어 있는지 확인합니다. 또한 토큰 서명, 토큰 만료 시간에 따른 유효성 및 토큰 클레임의 범위에 따라 액세스 수준을 기반으로 발급자를 확인합니다. 사용자 풀 범위는 액세스 토큰 `scope` 클레임에 있습니다. Amazon Cognito 액세스 토큰의 클레임에 대한 자세한 내용은 액세스 토큰 이해 섹션을 참조하세요.

Amazon Cognito를 사용하면 액세스 토큰의 범위가 외부 API 또는 사용자 속성에 대한 액세스를 승인할 수 있습니다. 로컬 사용자, 페더레이션 사용자 또는 시스템 ID에 액세스 토큰을 발급할 수 있습니다.

주제

- API 권한 부여
- 기계 간(M2M) 인증
- 범위에 대한 정보
- 리소스 서버에 대한 정보
- 리소스 바인딩

API 권한 부여

다음은 Amazon Cognito 토큰을 사용하여 API에 대한 요청을 승인할 수 있는 몇 가지 방법입니다.

액세스 토큰

REST API 메서드 요청 구성에 Amazon Cognito 권한 부여자를 추가할 때 권한 부여자 구성에 권한 부여 범위를 추가합니다. 이 구성을 사용하면 API가 `Authorization` 헤더의 액세스 토큰을 수락하고 수락된 범위를 검토합니다.

ID 토큰

유효한 ID 토큰을 REST API의 Amazon Cognito 권한 부여자에게 전달하면 API Gateway는 요청을 수락하고 ID 토큰 콘텐츠를 API 백엔드로 전달합니다.

Amazon Verified Permissions

Verified Permissions에는 API 연결 정책 스토어를 만들 수 있는 옵션이 있습니다. Verified Permissions는 요청 `Authorization` 헤더에서 ID 또는 액세스 토큰을 처리하는 Lambda 권한 부여자를 생성하고 할당합니다. 이 Lambda 권한 부여자는 토큰을 정책 스토어에 전달합니다. 여기서 Verified Permissions는 토큰을 정책과 비교하고 권한 부여자에게 허용 또는 거부 결정을 반환합니다.

추가 리소스

- API Gateway에서 REST API에 대한 액세스 제어 및 관리
- Amazon Verified Permissions를 통한 권한 부여

기계 간(M2M) 인증

Amazon Cognito는 시스템 ID를 사용하여 API 데이터에 액세스하는 애플리케이션을 지원합니다. 사용자 풀의 시스템 ID는 애플리케이션 서버에서 실행되고 원격 API에 연결되는 기밀 클라이언트입니다. 이러한 작업은 예약된 작업, 데이터 스트림 또는 자산 업데이트와 같은 사용자 상호 작용 없이 수행됩니다. 이러한 클라이언트가 액세스 토큰으로 요청을 승인하면 기계 간 또는 M2M, 권한 부여를 수행합니다. M2M 권한 부여에서 공유 보안 암호는 액세스 제어의 사용자 자격 증명을 대체합니다.

M2M 권한 부여를 사용하여 API에 액세스하는 애플리케이션에는 클라이언트 ID와 클라이언트 보안 암호가 있어야 합니다. 사용자 풀에서 클라이언트 보안 암호가 있는 앱 클라이언트를 빌드합니다. 그런 다음 시스템 자격 증명은 두 가지 방법 중 하나로 M2M 액세스 토큰을 얻을 수 있습니다. 즉Token 엔드포인트,에서 클라이언트 자격 증명 부여를 요청하거나 GetClientToken API 작업을 호출할 수 있습니다. 두 접근 방식 모두 리소스 서버의 사용자 지정 범위만 승인하는 액세스 토큰을 발급합니다. 설정이 다릅니다. 토큰 엔드포인트에는 사용자 풀 도메인이 필요하고 OIDC 라이브러리를 사용하는 애플리케이션에 적합한 반면, `GetClientToken`에는 도메인이 필요하지 않으며 AWS SDK AWS CLI또는 API를 통해 작동합니다. 앱 클라이언트 설정에 대한 자세한 내용은 앱 클라이언트를 사용한 애플리케이션별 설정 섹션을 참조하세요.

M2M 액세스 토큰은 시스템 자격 증명이 API에서 요청하도록 허용하려는 작업의 확인 가능한 문입니다. 액세스 토큰이 API 요청을 승인하는 방법에 대해 자세히 알아보려면 계속 읽어보세요. 예제 애플리

케이션은 AWS CDK를 사용한 Amazon Cognito 및 API Gateway 기반 기계 간 권한 부여를 참조하세요.

M2M 권한 부여에는 월별 활성 사용자(MAU)에게 청구되는 방식과 다른 청구 모델이 있습니다. 사용자 인증에 활성 사용자당 비용이 부과되는 경우 M2M 결제에는 클라이언트 자격 증명 부여 또는 `GetClientToken` API 작업 사용 여부와 상관없이 활성 M2M 앱 클라이언트와 총 토큰 요청 볼륨이 반영됩니다. 자세한 내용은 Amazon Cognito 요금을 참조하세요. M2M 권한 부여의 비용을 제어하려면 액세스 토큰의 기간과 애플리케이션이 수행하는 토큰 요청 수를 최적화하십시오. API Gateway 캐싱을 사용하여 M2M 권한 부여에서 새 토큰에 대한 요청을 줄이는 방법은 사용자 풀 토큰 만료 및 캐싱 관리 섹션을 참조하세요.

AWS 청구서에 비용을 추가하는 Amazon Cognito 작업 최적화에 대한 자세한 내용은 섹션을 참조하세요비용 관리.

M2M(머신 간) 클라이언트 자격 증명을 위한 클라이언트 메타데이터

M2M 요청에서 클라이언트 메타데이터를 전달할 수 있습니다. 클라이언트 메타데이터는 사전 토큰 생성 Lambda 트리거의 결과에 기여할 수 있는 사용자 또는 애플리케이션 환경의 추가 정보입니다. 사용자 보안 주체를 사용한 인증 작업에서 AdminRespondToAuthChallenge 및 RespondToAuthChallenge API 요청 본문의 사전 토큰 생성 트리거에 클라이언트 메타데이터를 전달할 수 있습니다. 애플리케이션은 Token 엔드포인트에 대한 직접 요청으로 M2M에 대한 액세스 토큰 생성을 위한 흐름을 수행하므로 모델이 다릅니다. 클라이언트 자격 증명에 대한 토큰 요청의 POST 본문에서 클라이언트 메타데이터 객체 URL 인코딩(`x-www-form-urlencoded`)이 포함된 `aws_client_metadata` 파라미터를 문자열에 전달합니다. 요청 예시는 기본 권한이 있는 클라이언트 자격 증명 섹션을 참조하세요. 다음은 키-값 페어 `{"environment": "dev", "language": "en-US"}`를 전달하는 예제 파라미터입니다.

```
aws_client_metadata=%7B%22environment%22%3A%20%22dev%22,%20%22language%22%3A%20%22en-US%22%7D
```

GetClientToken을 사용하여 M2M 액세스 토큰 가져오기

GetClientToken API 작업은 사용자 풀 도메인 없이 AWS SDK AWS CLI또는 API를 통해 기밀 앱 클라이언트에 대한 M2M 액세스 토큰을 발급합니다. 토큰 엔드포인트에서 부여한 클라이언트 자격 증명과 동일한 기능을 제공합니다. 둘 다 사용자가 아닌 애플리케이션에 권한을 부여합니다.

> **Note**
>
> Amazon Cognito는 AWS Identity and Access Management (IAM) 자격 증명 기반 정책을 사용하여를 승인하지 않습니다`GetClientToken`. IAM 자격 증명을 사용하여 요청을 승인할 수 없습니다. 권한 부여는 앱 클라이언트의 클라이언트 보안 암호에서 가져옵니다.

를 사용하려면 클라이언트 보안 암호와 `ALLOW_CLIENT_TOKEN_AUTH` 인증 흐름만 있는 앱 클라이언트를 `GetClientToken`구성합니다. 이 흐름은 사용자 인증 흐름과 상호 배타적입니다. 앱 클라이언트에는 최대 2개의 활성 보안 암호가 있을 수 있습니다. 리소스 서버 사용자 지정 범위를 앱 클라이언트와 연결합니다. 다음 AWS CLI 예시에서는 호환되는 앱 클라이언트를 생성합니다.

```
aws cognito-idp create-user-pool-client \
    --user-pool-id us-west-2_EXAMPLE \
    --client-name my-m2m-client \
    --generate-secret \
    --explicit-auth-flows ALLOW_CLIENT_TOKEN_AUTH \
    --allowed-o-auth-scopes "solar-system-data/asteroids.add"
```

액세스 토큰을 가져오려면 앱 클라이언트 ID, 활성 클라이언트 보안 암호 및 토큰에서 원하는 사용자 지정 범위를 `GetClientToken` 사용하여를 호출합니다. `GetClientToken`는 `UserPoolId` 파라미터를 가져오지 않습니다. Amazon Cognito는 클라이언트 ID에서 사용자 풀을 식별합니다. 를 지정하지 않으면 `Scopes` Amazon Cognito는 앱 클라이언트에 대해 구성된 범위를 승인합니다.

```
aws cognito-idp get-client-token \
    --client-id 1example23456789 \
    --secret exampleClientSecret123EXAMPLE \
    --scopes "solar-system-data/asteroids.add"
```

응답에는 `ClientAuthenticationResult` 객체의 액세스 토큰과 메타데이터가 포함됩니다.

```
{
    "ClientAuthenticationResult": {
        "AccessToken": "eyJra456defEXAMPLE",
        "ExpiresIn": 3600,
        "TokenType": "Bearer"
    }
}
```

Amazon API Gateway REST API에 대한 요청의 `Authorization` 헤더와 같이 리소스 서버에 액세스 토큰을 제공합니다. 리소스 서버는 토큰 서명 및 만료를 확인한 다음 토큰의 범위를 기반으로 요청을 승인합니다.

토큰을 사용자 사전 토큰 생성 Lambda 트리거 지정하는에 데이터를 전달하려면 `--client-metadata` 파라미터와 함께 요청에 `ClientMetadata` 맵을 포함합니다. M2M 흐름에서 클라이언트 메타데이터를 수신하려면 트리거 이벤트 버전 3 이상에 대해 사전 토큰 생성 트리거를 구성해야 합니다. Amazon Cognito는이 데이터를 저장, 검증 또는 암호화하지 않으며 Lambda 트리거에서만 사용할 수 있으므로 민감한 정보를 포함하지 마세요.

범위에 대한 정보

범위는 앱이 리소스에 요청할 수 있는 액세스의 수준입니다. Amazon Cognito 액세스 토큰에서 범위는 사용자 풀에 설정한 신뢰, 즉 알려진 디지털 서명이 있는 신뢰할 수 있는 액세스 토큰 발급자에 의해 뒷받침됩니다. 사용자 풀은 고객이 자신의 사용자 프로필 중 일부 또는 전부를 관리하거나 백엔드 API에서 데이터를 검색할 수 있음을 입증하는 범위를 사용하여 액세스 토큰을 생성할 수 있습니다. Amazon Cognito 사용자 풀은 사용자 풀에 예약된 API 범위, 사용자 지정 범위 및 OpenID Connect(OIDC) 범위를 사용하여 액세스 토큰을 발급합니다.

사용자 풀의 예약된 API 범위

`aws.cognito.signin.user.admin` 범위는 Amazon Cognito 사용자 풀 API의 현재 사용자에 대한 셀프 서비스 작업을 승인합니다. 액세스 토큰 보유자에게 GetUser 및 UpdateUserAttributes API 작업 등을 통해 보유자에 대한 모든 정보를 쿼리하고 업데이트할 수 있는 권한을 부여합니다. 이 범위는 Amazon Cognito 사용자 풀 API로 사용자를 인증할 때 액세스 토큰에서 수신하는 유일한 범위입니다. 또한 앱 클라이언트에 읽기 및 쓰기 권한을 부여한 사용자 속성을 읽고 쓰는 데 필요한 유일한 범위이기도 합니다. 요청 시 이 범위를 권한 부여 엔드포인트에 요청할 수도 있습니다. 이 범위만으로는 userInfo 엔드포인트에서 사용자 속성을 요청하기에 충분하지 않습니다. 사용자 풀 API 및 사용자에 대한 `userInfo` 요청을 모두 승인하는 액세스 토큰의 경우 `/oauth2/authorize` 요청 시 범위 `openid` 및 `aws.cognito.signin.user.admin`을 모두 요청해야 합니다.

사용자 지정 범위

사용자 지정 범위는 리소스 서버가 보호하는 외부 API에 대한 요청을 승인합니다. 다른 유형의 범위를 사용하여 사용자 지정 범위를 요청할 수 있습니다. 이 페이지 전반에서 사용자 지정 범위에 대한 자세한 정보를 찾을 수 있습니다.

OpenID Connect(OIDC) 범위

관리형 로그인을 포함하여 사용자 풀 권한 부여 서버로 사용자를 인증하는 경우 범위를 요청해야 합니다. Amazon Cognito 권한 부여 서버에서 사용자 풀 로컬 사용자와 서드 파티 페더레이션 사용자를 인증할 수 있습니다. OIDC 범위는 앱이 사용자 풀의 userInfo 엔드포인트에서 사용자 정보를 읽을 수 있는 권한을 부여합니다. `userInfo` 엔드포인트에서 사용자 속성을 쿼리하는 OAuth 모델은 사용자 속성에 대한 대량의 요청에 맞게 앱을 최적화할 수 있습니다. `userInfo` 엔드포인트는 액세스 토큰의 범위에 따라 결정되는 권한 수준의 속성을 반환합니다. 앱 클라이언트가 다음과 같은 OIDC 범위를 사용하여 액세스 토큰을 발급하도록 권한을 부여할 수 있습니다.

openid

OpenID Connect(OIDC) 쿼리의 필수 범위입니다. ID 토큰, 고유 식별자 클레임 `sub` 및 다른 범위를 요청하는 기능을 승인합니다.

> **Note**
>
> `openid` 범위를 요청하고 다른 범위를 요청하지 않은 경우 사용자 풀 ID 토큰과 `userInfo` 응답에는 앱 클라이언트가 읽을 수 있는 모든 사용자 속성에 대한 클레임이 포함됩니다. 요청할 때 `openid` 및 `profile`, `email`, phone 등 기타 OIDC 범위(예: ID 토큰 및 userInfo 응답)는 추가 범위의 제약 조건으로 제한됩니다.
> 예를 들어, 파라미터와 권한 부여 엔드포인트를 함께 요청하면 `sub`, `email`, `email_verified` 등이 포함된 ID 토큰이 `scope=openid+email`에서 반환됩니다.
> 이 요청의 액세스 토큰은 userInfo 엔드포인트와 동일한 속성을 반환합니다. 파라미터 `scope=openid`가 있는 요청은 ID 토큰 및 원본 `userInfo`에서 클라이언트가 읽을 수 있는 모든 속성을 반환합니다.

profile

앱 클라이언트가 읽을 수 있는 모든 사용자 속성을 승인합니다.

이메일

사용자 속성 `email` 및 `email_verified`를 승인합니다. Amazon Cognito는 값이 명시적으로 설정된 경우 `email_verified`를 반환합니다.

phone

사용자 속성 `phone_number` 및 `phone_number_verified`를 승인합니다.

리소스 서버에 대한 정보

리소스 서버 API는 데이터베이스의 정보에 대한 액세스 권한을 부여하거나 IT 리소스를 제어할 수 있습니다. Amazon Cognito 액세스 토큰은 OAuth 2.0을 지원하는 API에 대한 액세스 권한을 부여할 수 있습니다. Amazon API Gateway REST API에는 Amazon Cognito 액세스 토큰을 통한 권한 부여 지원 기능이 내장되어 있습니다. 앱은 API 호출 시 액세스 토큰을 리소스 서버로 전달합니다. 리소스 서버는 액세스 토큰을 검사하여 액세스 권한을 부여할지 여부를 결정합니다.

Amazon Cognito는 향후 사용자 풀 액세스 토큰의 스키마를 업데이트할 수 있습니다. 앱이 액세스 토큰을 API로 전달하기 전에 콘텐츠를 분석하는 경우 스키마 업데이트를 수락하도록 코드를 엔지니어링해야 합니다.

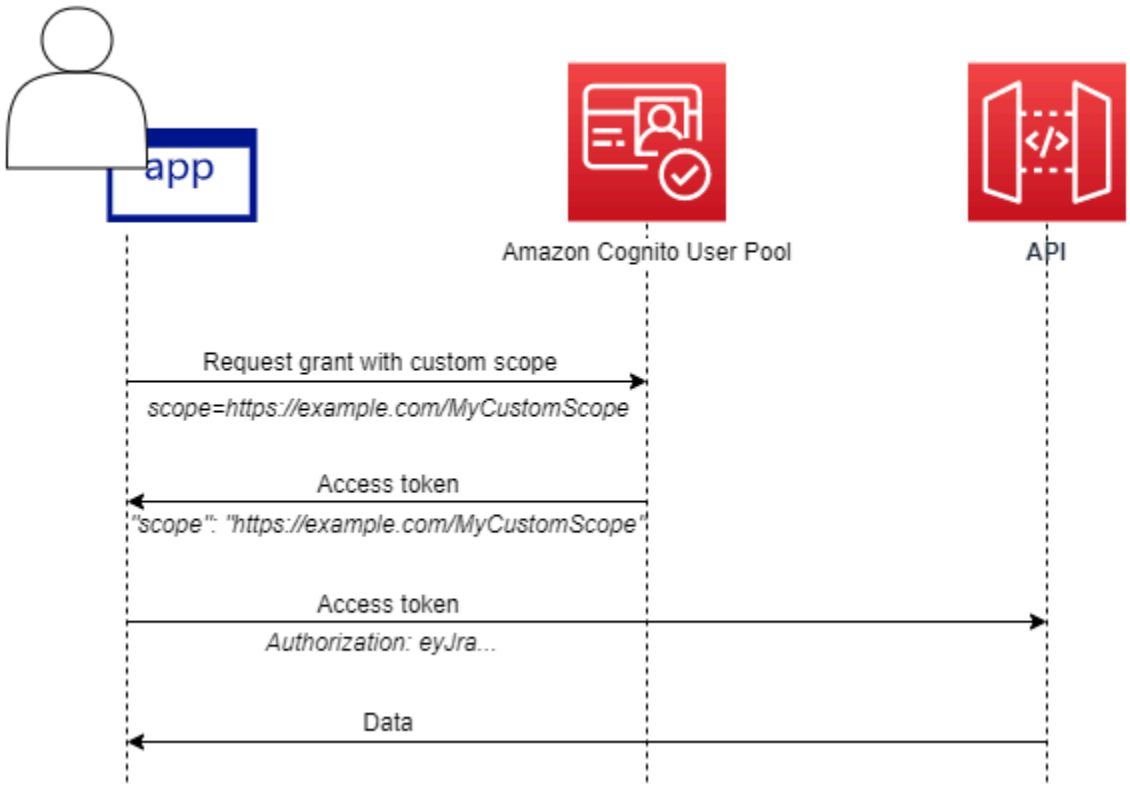
사용자 지정 범위는 사용자가 정의하며, 사용자 풀의 권한 부여 기능을 확대하여 사용자 및 해당 속성의 쿼리 및 수정과 관련이 없는 용도까지 포함합니다. 예를 들어 사진에 대한 리소스 서버를 가지고 있는 경우에는 두 개의 범위를 정의하게 되는데, `photos.read`는 사진에 대한 읽기 액세스를 위한 범위이고 `photos.write`는 쓰기/삭제 액세스를 위한 범위입니다. 권한 부여를 위해 액세스 토큰을 수락하도록 API를 구성하고, `scope` 클레임에 `photos.read`가 있는 액세스 토큰에 대한 `HTTP GET` 요청과 `photos.write`가 있는 토큰에 대한 `HTTP POST` 요청을 허용할 수 있습니다. 이들은 사용자 지정 범위입니다.

> **Note**
>
> 리소스 서버는 토큰 내에서 클레임을 처리하기 앞서 액세스 토큰 서명과 만료 날짜를 확인해야 합니다. 토큰 확인에 대한 자세한 내용은 JSON 웹 토큰 확인 섹션을 참조하세요. Amazon API Gateway에서 사용자 풀 토큰을 확인하고 사용하는 방법에 대한 자세한 내용은 API Gateway와 Amazon Cognito 사용자 풀 통합을 참조하세요. API Gateway는 액세스 토큰을 검사하여 리소스를 보호하는 데 적합한 옵션입니다. API Gateway Lambda 권한 부여자에 대한 자세한 내용은 API Gateway Lambda 권한 부여자 사용을 참조하세요.

개요

Amazon Cognito를 사용하여 OAuth 2.0 리소스 서버를 생성하고 사용자 지정 범위를 리소스 서버와 연결할 수 있습니다. 액세스 토큰의 사용자 지정 범위는 API의 특정 작업을 승인합니다. 사용자 풀의 모든 앱 클라이언트가 모든 리소스 서버에서 사용자 지정 범위를 발급하도록 권한을 부여할 수 있습니다. 사용자 지정 범위를 앱 클라이언트와 연결하고 Token 엔드포인트에서 OAuth 2.0 인증 코드 권한 부여, 암시적 권한 부여 및 클라이언트 보안 인증 권한 부여에서 해당 범위를 요청할 수 있습니다. M2M의 경우 GetClientToken API 작업을 사용하여 사용자 지정 범위를 요청할 수도 있습니다. Amazon Cognito는 액세스 토큰의 `scope` 클레임에 사용자 지정 범위를 추가합니다. 클라이언트는 리소스 서버에 대해

액세스 토큰을 사용할 수 있습니다. 리소스 서버는 토큰에 있는 범위를 기준으로 권한 부여를 결정합니다. 액세스 토큰 범위에 대한 자세한 내용은 사용자 풀을 통해 토큰 사용을 참조하세요.



사용자 지정 범위가 있는 액세스 토큰을 얻기 위해 앱은 Token 엔드포인트에 권한 부여 코드를 사용하거나 클라이언트 자격 증명 부여를 요청하도록 요청할 수 있습니다. 관리형 로그인에서 암시적 권한 부여에서 액세스 토큰의 사용자 지정 범위를 요청할 수도 있습니다. M2M 권한 부여의 경우 앱은 GetClientToken API 작업을 사용하여 사용자 지정 범위를 요청할 수도 있습니다.

> **Note**
>
> 사용자 풀을 IdP로 사용하는 인간 대화형 인증을 위해 설계되었기 때문에 InitiateAuth 및 AdminInitiateAuth 요청은 단일 값 `aws.cognito.signin.user.admin`을 사용하여 액세스 토큰에서 `scope` 클레임만 생성합니다.

리소스 서버 및 사용자 지정 범위 관리

리소스 서버를 생성할 때 반드시 리소스 서버 이름과 리소스 서버 식별자를 제공해야 합니다. 리소스 서버에서 생성한 각 범위에 대해 범위 이름과 설명을 입력해야 합니다.

- 리소스 서버 이름: `Solar system object tracker` 또는 `Photo API` 같은 기억하기 쉬운 리소스 서버 이름입니다.
- 리소스 서버 식별자: 리소스 서버의 고유 식별자입니다. 식별자는 API와 연결할 이름(예: `solar-system-data`)입니다. 예를 들어 API URI 경로에 대한 보다 직접적인 참조로 `https://solar-system-data-api.example.com` 같은 더 긴 식별자를 구성할 수 있지만 문자열이 길면 액세스 토큰의 크기가 커집니다.
- 범위 이름: `scope` 클레임에서 사용할 값입니다. 예: `sunproximity.read`
- 설명: 범위에 대한 알기 쉬운 설명입니다. 예: `Check current proximity to sun`

Amazon Cognito는 로컬 사용자 풀의 사용자이든 서드 파티 ID 제공업체와의 페더레이션 사용자이든, 모든 사용자의 액세스 토큰에 사용자 지정 범위를 포함할 수 있습니다. 관리형 로그인이 포함된 OAuth 2.0 권한 부여 서버를 사용하여 인증 흐름 중에 사용자 액세스 토큰의 범위를 선택할 수 있습니다. 사용자 인증은 권한 부여 엔드포인트에서 `scope`를 요청 파라미터 중 하나로 사용하여 시작해야 합니다. 다음은 리소스 서버의 권장 형식입니다. 식별자에는 API 친화적인 이름을 사용합니다. 그리고 사용자 지정 범위에는 해당 범위가 승인하는 작업을 사용합니다.

```
resourceServerIdentifier/scopeName
```

예를 들어, 카이퍼 벨트에서 새로운 소행성을 발견했고 이를 `solar-system-data` API를 통해 등록하려고 합니다. 소행성 데이터베이스에 대한 쓰기 작업을 승인하는 범위는 `asteroids.add`입니다. 검색을 등록할 수 있는 권한을 부여하는 액세스 토큰을 요청할 때 `scope` HTTPS 요청 파라미터의 형식을 로 지정합니다`scope=solar-system-data/asteroids.add`.

리소스 서버에서 범위를 삭제해도 모든 클라이언트와의 연결이 삭제되지는 않습니다. 대신, 범위를 삭제하면 범위가 inactive로 표시됩니다. Amazon Cognito는 액세스 토큰에 비활성 범위를 추가하지 않지만 앱에서 요청할 경우 정상적으로 진행합니다. 나중에 리소스 서버에 범위를 다시 추가하면 Amazon Cognito가 해당 범위를 액세스 토큰에 다시 기록합니다. 앱 클라이언트와 연결하지 않은 범위를 요청하면 사용자 풀 리소스 서버에서 해당 범위를 삭제했는지 여부에 관계없이 인증이 실패합니다.

, AWS Management Console API 또는 CLI를 사용하여 사용자 풀의 리소스 서버 및 범위를 정의할 수 있습니다.

사용자 풀의 리소스 서버 정의(AWS Management Console)

를 사용하여 사용자 풀 AWS Management Console 에 대한 리소스 서버를 정의할 수 있습니다.

리소스 서버를 정의하려면

1. Amazon Cognito 콘솔에 로그인합니다.
2. 탐색 창에서 [사용자 풀(User Pools)]을 선택한 다음 편집할 사용자 풀을 선택합니다.
3. 브랜딩에서 도메인 메뉴를 선택하고 리소스 서버를 찾습니다.
4. [리소스 서버 생성(Create a resource server)]을 선택합니다.
5. [리소스 서버 이름(Resource server name)]을 입력합니다. 예: `Photo Server`
6. [리소스 서버 식별자(Resource server identifier)]를 입력합니다. 예: `com.example.photos`
7. 리소스에 대한 [사용자 정의 범위(Custom scopes)]를 입력합니다(예: `read`, `write`).
8. 각 [범위 이름(Scope name)]에 대한 [설명(Description)]을 입력합니다(예: `view your photos`, `update your photos`).
9. 생성을 선택합니다.

사용자 지정 범위는 리소스 서버 아래의 도메인 메뉴에서 사용자 지정 범위 열을 통해 검토할 수 있습니다. 애플리케이션의 앱 클라이언트 메뉴에서 앱 클라이언트에 사용자 지정 범위를 사용하도록 설정할 수 있습니다. 앱 클라이언트를 선택하고 로그인 페이지를 찾은 다음 편집을 선택합니다. [사용자 정의 범위(Custom scopes)]를 추가하고 [변경 사항 저장(Save changes)]을 선택합니다.

사용자 풀(AWS CLI 및 AWS API)에 대한 리소스 서버 정의

다음 명령을 사용하여 사용자 풀에 대한 리소스 서버 설정을 지정합니다.

리소스 서버를 생성하려면

- AWS CLI: `aws cognito-idp create-resource-server`
- AWS API: CreateResourceServer

리소스 서버 설정에 대한 정보를 가져오려면

- AWS CLI: `aws cognito-idp describe-resource-server`
- AWS API: DescribeResourceServer

사용자 풀의 모든 리소스 서버에 대한 정보를 나열하려면

- AWS CLI: `aws cognito-idp list-resource-servers`
- AWS API: ListResourceServers

리소스 서버 삭제

- 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

리소스 바인딩

리소스 지표라고도 하는 리소스 바인딩을 사용하면 사용자 풀 권한 부여 서버에 API별 권한 부여를 요청할 수 있습니다. 리소스 바인딩은 클라이언트가 권한 부여 요청 중에 액세스하려는 리소스 서버를 명시적으로 지정할 수 있도록 RFC 8707에 정의된 OAuth 2.0 확장입니다. 리소스 바인딩을 사용하면 API 구성이 특별히 의도하지 않은 토큰에 대한 액세스를 거부할 수 있습니다.

> **Note**
>
> 액세스 토큰은 사용자의 리소스에만 바인딩할 수 있습니다. 클라이언트 자격 증명 M2M 권한 부여로는 리소스 바인딩을 요청할 수 없습니다.

Amazon Cognito 사용자 풀에서 리소스 바인딩을 사용하는 경우 클라이언트는 사용자 풀 권한 부여 서버에 대한 인증 요청에 `resource` 파라미터를 포함할 수 있습니다. 사용자 풀은 요청된 리소스의 값이 앱 클라이언트 콜백 URL과 동일한 스키마 규칙(`localhost`만 사용하는 `https://`, `http://` 또는 `myapp://`와 같은 사용자 지정 스키마)을 따르는 URL인지 확인합니다. Amazon Cognito는 요청된 URI를 액세스 토큰 `aud` 클레임의 대상으로 설정합니다. 요청된 리소스가 사용자 풀 리소스 서버인 경우 리소스 서버 식별자는 URL 형식이어야 합니다. 인증 요청당 하나의 리소스를 요청할 수 있습니다.

이 기능은 사용자 풀 OAuth 2.0 권한 부여 서버를 사용한 관리형 로그인 인증에만 해당됩니다. 권한 부여 엔드포인트에서 암시적 및 권한 부여 코드 권한 부여로 리소스 바인딩을 요청할 수 있습니다. 원래 요청에서 `aud` 클레임을 이어받아 Token 엔드포인트에서 토큰 새로 고침 권한을 부여합니다. 현재 SDK 인증 모델에서는 사용할 수 없습니다.

Amazon Cognito 사용자 풀을 사용하여 리소스 바인딩 구현

1. 고유 식별자를 사용하여 사용자 풀에 하나 이상의 리소스 서버를 구성합니다.

2. /oauth2/authorize에 대한 권한 부여 요청에서 권한 부여 코드 또는 암시적 권한 부여를 요청하고 resource 파라미터를 포함합니다. resource의 값은 URL 형식 리소스 서버 식별자 또는 URL이어야 합니다. 예: &resource=https://solar-system-data-api.example.com
3. 권한 부여 서버는 리소스 요청을 검증하고, 인증을 완료하고, 액세스 토큰 aud 클레임을 요청된 리소스 URL로 설정합니다.
4. 토큰이 특별히 발급되었는지 검증하기 위해 사용자의 액세스 토큰을 사용하는 리소스가 aud 클레임을 확인합니다.

사용자 관리

사용자가 로그인할 수 있게 되면 애플리케이션 수명 동안 계정을 관리할 수 있습니다. 이 섹션에서는 사용자 디렉터리의 day-to-day 관리, 즉 사용자 생성 및 가져오기, 그룹으로 구성, 사용자 풀을 자격 증명 풀에 연결하여 사용자가 자격 AWS 증명을 얻을 수 있도록 하는 방법을 다룹니다.

주제

- 사용자 풀의 사용자 관리
- 사용자 풀 및 앱 클라이언트 구성 업데이트

사용자 풀의 사용자 관리

사용자 풀을 생성한 후 사용자 계정을 생성, 확인 및 관리할 수 있습니다. Amazon Cognito 사용자 풀 그룹을 사용하면 IAM 역할을 그룹에 매핑하여 사용자와 리소스에 대한 액세스를 관리할 수 있습니다.

Amazon Cognito 사용자 풀의 사용자 관리에는 다양한 구성 옵션과 관리 작업이 포함됩니다. 사용자 풀은 수백만 명의 사용자로 확장할 수 있습니다. 이 정도 규모의 사용자 디렉터리에는 똑같이 확장 가능하고 반복 가능한 관리 도구가 필요합니다. 많은 사용자 프로필을 생성하거나, 비활성 사용자를 관리하거나, 거버넌스 및 규정 준수 보고서를 생성하거나, 사용자가 대부분의 작업을 수행하는 셀프 서비스 도구를 설정할 수 있습니다. 사용자 풀을 생성한 후 이메일 또는 전화번호 확인 요구 등 사용자가 계정을 등록하고 확인하는 방법을 제어할 수 있습니다. 관리자는 사용자 계정을 직접 생성하고 환영 메시지 및 암호 요구 사항을 사용자 지정할 수도 있습니다.

사용자 풀에는 사용자의 그룹 멤버십을 기반으로 리소스에 대한 액세스를 관리할 수 있는 사용자 그룹이 있습니다. 이러한 그룹에 IAM 역할을 할당하여 ID 풀을 통해 AWS 서비스에 대한 액세스를 관리할 수 있습니다. 사용자의 그룹 멤버십은 ID와 액세스 토큰 모두에 있습니다. 이 정보를 사용하면 애플리케이션에서 런타임 시 또는 Amazon Verified Permissions와 같은 정책 엔진을 사용하여 액세스 제어 결정을 내릴 수 있습니다.

사용자 풀에는 많은 사용자가 있는 경우가 많습니다. 사용자 계정을 검색하고 업데이트하는 경우가 많습니다. Amazon Cognito 콘솔 및 API는 사용자 이름, 이메일 및 전화번호와 같은 표준 속성을 기반으로 사용자 쿼리를 지원합니다. 관리자는 암호를 재설정하고, 계정을 비활성화하고, 사용자 이벤트 기록을 볼 수도 있습니다.

기존 사용자 데이터를 마이그레이션하기 위해 Amazon Cognito에는 CSV 파일에서 사용자를 가져오고 Lambda 트리거를 사용하여 처음 로그인할 때 사용자를 자동으로 마이그레이션할 수 있는 옵션이 있습니다. 이러한 옵션은 다른 사용자 디렉터리에서 사용자 풀로 사용자 전환을 지원합니다.

사용자 풀의 사용자 관리 기능을 사용하여 사용자 수명 주기 및 인증 환경을 세밀하게 제어할 수 있습니다. 셀프 서비스 가입, 관리자가 만든 계정, 그룹 및 마이그레이션 도구의 조합으로 Amazon Cognito 사용자 풀은 유연한 사용자 디렉터리로 만들어집니다.

주제

- 사용자 생성을 위한 정책 구성
- 사용자 계정 가입 및 확인
- 관리자로 사용자 계정 생성
- 사용자 풀에 그룹 추가
- 사용자 계정 관리 및 검색
- 암호, 계정 복구 및 암호 정책
- 사용자 풀로 사용자 가져오기
- 사용자 속성 작업

사용자 생성을 위한 정책 구성

사용자 풀에서 사용자가 가입하도록 허용하거나 사용자를 관리자로 생성할 수 있습니다. 또한 가입 후 인증 및 확인 프로세스를 사용자에게 맡기는 정도를 제어할 수 있습니다. 예를 들어, 가입을 검토하고 외부 검증 프로세스에 따라 가입을 수락하고 싶을 수 있습니다. 이 구성 또는 관리자로 사용자 생성 정책은 또한 사용자가 더 이상 사용자 계정을 확인할 수 없을 때까지의 시간을 설정합니다.

Amazon Cognito는 소프트웨어를 위한 고객 ID 및 액세스 관리(CIAM) 플랫폼으로서 일반 고객의 요구를 충족할 수 있습니다. 가입을 수락하고 관리형 로그인 유무에 관계없이 앱 클라이언트가 있는 사용자 풀은 공개적으로 검색 가능한 앱 클라이언트 ID와 가입 요청을 알고 있는 인터넷상의 모든 사용자를 위한 사용자 프로필을 생성합니다. 가입한 사용자 프로필은 액세스 및 ID 토큰을 받을 수 있으며 앱에 대해 승인된 리소스에 액세스할 수 있습니다. 사용자 풀에서 가입을 활성화하기 전에 옵션을 검토하고 구성이 보안 표준을 준수하는지 확인하십시오. 다음 절차에 설명된 대로 자체 등록 활성화 및 `AllowAdminCreateUserOnly`를 주의해서 설정하십시오.

AWS Management Console

사용자 풀의 가입 메뉴에는 사용자 풀에 있는 사용자의 가입 및 관리 생성을 위한 몇 가지 설정이 포함되어 있습니다.

가입 환경 구성

1. Cognito 지원 검증 및 확인에서 Cognito가 검증 및 확인을 위한 메시지를 자동으로 보내도록 허용 여부를 선택합니다. 이 설정을 활성화하면 Amazon Cognito는 사용자 풀에 제시해야 하는 코드가 포함된 이메일 또는 SMS 메시지를 새 사용자에게 보냅니다. 이렇게 하면 이메일 주소 또는 전화 번호의 소유권을 확인하고, 동등한 속성을 확인된 것으로 설정하고, 로그인할 사용자 계정을 확인합니다. 선택한 확인할 속성에 따라 확인 메시지의 전달 방법 및 대상이 결정됩니다.
2. 사용자를 생성할 때 속성 변경 확인은 중요하지 않지만, 이는 속성 확인과 관련이 있습니다. 로그인 속성을 변경했지만 아직 확인하지 않은 사용자가 새 속성 값이나 원래 속성으로 계속 로그인하도록 허용할 수 있습니다. 자세한 내용은 사용자가 이메일 또는 전화 번호를 변경하는 경우 확인 섹션을 참조하세요.
3. 필수 속성에는 사용자가 가입하거나 사용자를 생성하기 전에 값을 제공해야 하는 속성이 표시됩니다. 사용자 풀을 생성할 때 필수 속성만 설정할 수 있습니다.
4. 사용자 지정 속성은 사용자를 처음 만들 때만 변경할 수 없는 사용자 지정 속성의 값을 설정할 수 있기 때문에 사용자 생성 및 가입 프로세스에서 중요합니다. 사용자 지정 속성에 대한 자세한 내용은 사용자 지정 속성 섹션을 참조하십시오.
5. 사용자가 인증되지 않은 `SignUp` API로 새 계정을 생성할 수 있도록 하려면 셀프 서비스 가입에서 자체 등록 활성화를 선택하세요. 자체 등록을 비활성화하면 Amazon Cognito 콘솔에서 또는 AdminCreateUser API 요청으로 새 사용자만 관리자로 생성할 수 있습니다. 자체 등록이 비활성화된 사용자 풀에서는 SignUp API 요청이 `NotAuthorizedException`을 반환하고 관리형 로그인에 가입 링크가 표시되지 않습니다.

사용자를 관리자로 생성하려는 사용자 풀의 경우 관리자가 설정한 임시 암호 만료 기간의 인증 방법 메뉴에서 임시 암호의 사용 기간을 구성할 수 있습니다.

사용자를 관리자로서 생성할 때 고려해야 할 또 다른 중요한 요소는 초대 메시지입니다. 새 사용자를 생성하면 Amazon Cognito에서 앱 링크가 포함된 메시지를 전송하여 사용자가 처음으로 로그인할 수 있도록 합니다. 메시지 템플릿의 인증 방법 메뉴에서 이 메시지 템플릿을 사용자 지정합니다.

앱 클라이언트 보안 암호가 없으면 가입을 방지하는 클라이언트 보안 암호를 사용하여 기밀 앱 클라이언트(일반적으로 웹 애플리케이션)를 구성할 수 있습니다. 보안 모범 사례에 따라 공용 앱 클라이언트(일반적으로 모바일 앱)에서 앱 클라이언트 보안 암호를 배포하지 마세요. Amazon Cognito 콘솔의 앱 클라이언트 메뉴에서 클라이언트 보안 암호를 사용하여 앱 클라이언트를 생성할 수 있습니다.

Amazon Cognito user pools API

CreateUserPool 또는 UpdateUserPool API 요청에서 사용자 풀의 사용자 생성을 위한 파라미터를 프로그래밍 방식으로 설정할 수 있습니다.

AdminCreateUserConfig 요소는 사용자 풀의 다음 속성에 대한 값을 설정합니다.

1. 셀프 서비스 가입 활성화
2. 새 관리자 생성 사용자에게 보내는 초대 메시지

다음 예시를 전체 API 요청 본문에 추가하면 셀프 서비스 가입이 비활성화되고 기본 초대 이메일이 포함된 사용자 풀을 설정합니다.

```
"AdminCreateUserConfig": {
      "AllowAdminCreateUserOnly": true,
      "InviteMessageTemplate": {
         "EmailMessage": "Your username is {username} and temporary password is
 {####}.",
         "EmailSubject": "Welcome to ExampleApp",
         "SMSMessage": "Your username is {username} and temporary password is
 {####}."
      }
   }
```

CreateUserPool 또는 UpdateUserPool API 요청의 다음 추가 파라미터는 새 사용자 생성을 제어합니다.

AutoVerifiedAttributes

새 사용자를 등록할 때 메시지를 자동으로 전송하려는 속성, 이메일 주소 또는 전화번호입니다.

정책

사용자 풀 암호 정책입니다.

스키마

사용자 풀 사용자 지정 특성입니다. 이들은 사용자를 처음 만들 때만 변경할 수 없는 사용자 지정 속성의 값을 설정할 수 있기 때문에 사용자 생성 및 가입 프로세스에서 중요합니다.

또한 이 파라미터는 사용자 풀에 필요한 속성을 설정합니다. 다음 텍스트를 전체 API 요청 본문의 Schema 요소에 삽입한 후 필요에 따라 email 속성을 설정합니다.

```
{
            "Name": "email",
            "Required": true
}
```

사용자 계정 가입 및 확인

사용자 계정은 다음 방법 중 하나로 사용자 풀에 추가됩니다.

- 사용자가 사용자 풀의 클라이언트 앱에 가입합니다. 모바일 또는 웹 앱일 수 있습니다.
- 사용자 계정을 사용자 풀로 가져올 수 있습니다. 자세한 내용은 CSV 파일에서 사용자 풀로 사용자 가져오기 섹션을 참조하세요.
- 사용자 풀에 사용자 계정을 만들고 로그인하도록 사용자를 초대할 수 있습니다. 자세한 내용은 관리자로 사용자 계정 생성 섹션을 참조하세요.

가입한 사용자가 로그인하려면 먼저 확인을 받아야 합니다. 가져온 사용자 및 생성된 사용자는 이미 확인되었지만 처음 로그인할 때 자신의 암호를 생성해야 합니다. 다음 섹션에서는 확인 프로세스와 이메일 및 전화 확인에 대해 설명합니다.

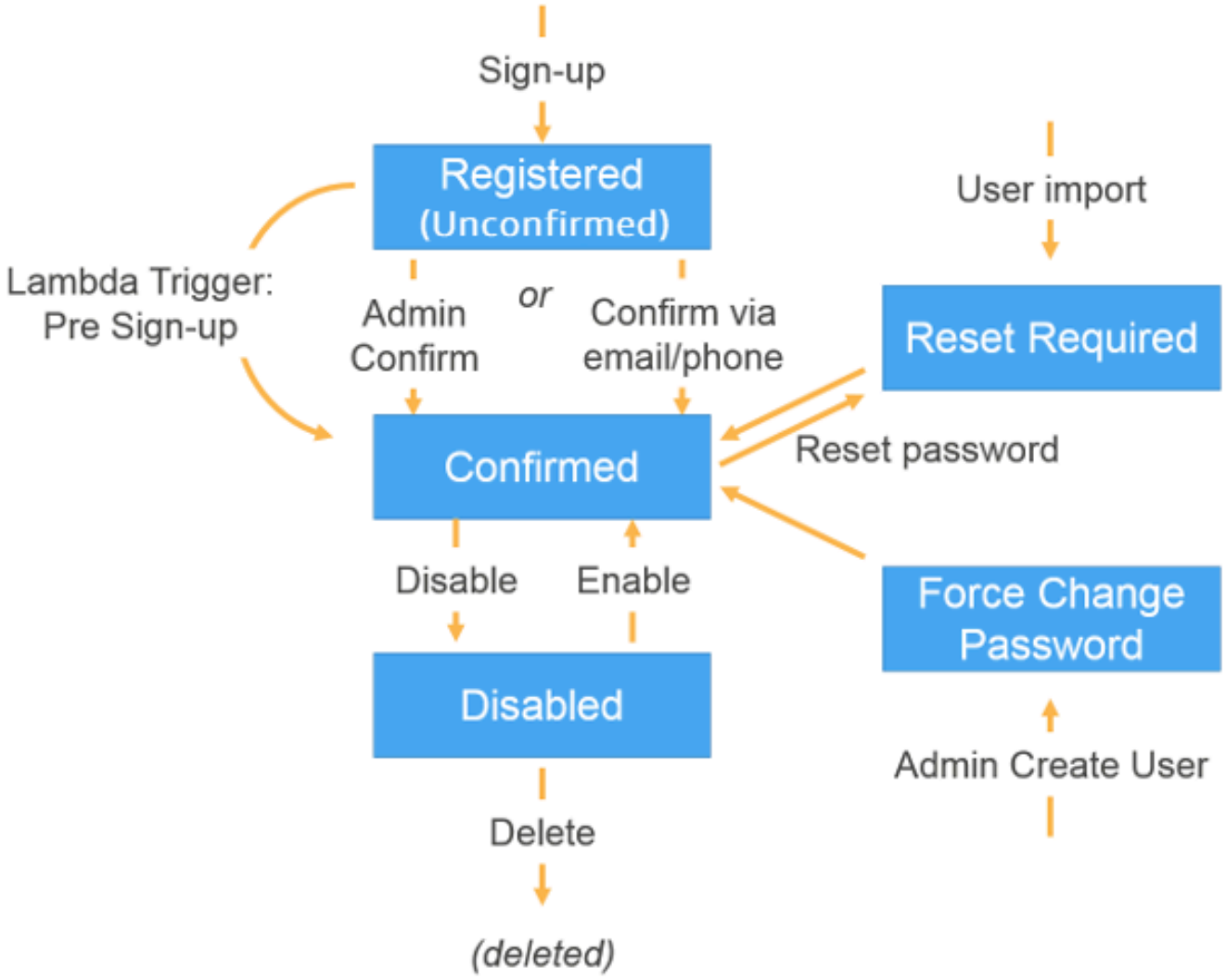
가입 시 암호

Amazon Cognito는 다음 조건을 제외하고 가입 시 모든 사용자의 암호를 요구합니다. 이러한 조건이 모두 충족되면 가입 작업에서 암호를 생략할 수 있습니다.

1. 사용자 풀 및 앱 클라이언트에서 암호 없는 로그인이 활성 상태입니다.
2. 애플리케이션은 AWS SDK의 인증 모듈로 사용자 지정 빌드됩니다. 관리형 로그인 및 호스팅 UI에는 항상 암호가 필요합니다.
3. 사용자는 사용자가 허용하는 이메일 또는 SMS 메시지 일회용 암호(OTP) 등 암호 없는 로그인 방법에 대한 속성 값을 제공합니다. 예를 들어 이메일 및 전화 OTP로 로그인을 허용하는 경우 사용자는 전화번호 또는 이메일 주소를 제공할 수 있지만 이메일로만 로그인을 허용하는 경우 이메일 주소를 제공해야 합니다.
4. 사용자 풀은 사용자가 암호 없는 로그인에 사용할 수 있는 속성을 자동으로 확인합니다.
5. 지정된 SignUp 요청의 경우 사용자는 암호 파라미터 값을 제공하지 않습니다.

사용자 계정 확인 개요

다음 다이어그램은 확인 프로세스를 보여 줍니다.



사용자 계정은 다음 상태 중 하나일 수 있습니다.

등록됨(확인되지 않음)

사용자가 성공적으로 가입했지만 사용자 계정이 확인될 때까지 로그인할 수 없습니다. 이 상태에서는 사용자가 활성화되었지만 확인되지 않았습니다.

자신을 등록하는 새 사용자는 이 상태에서 시작됩니다.

확인됨

사용자 계정이 확인되었으며 사용자가 로그인할 수 있습니다. 사용자가 코드를 입력하거나 이메일 링크를 따라 사용자 계정을 확인하면 해당 이메일 또는 전화번호가 자동으로 확인됩니다. 코드나 링크는 24시간 동안 유효합니다.

관리자 또는 사전 가입 Lambda 트리거에서 사용자 계정을 확인한 경우 계정에 연결되어 있는 확인된 이메일이나 전화번호가 없을 수 있습니다.

암호 재설정 필요

사용자 계정이 확인되었지만, 사용자가 로그인하기 전에 코드를 요청하고 암호를 재설정해야 합니다.

관리자 또는 개발자가 가져온 사용자 계정은 이 상태에서 시작합니다.

암호 강제 변경

사용자 계정이 확인되고 사용자가 임시 암호를 사용하여 로그인할 수 있지만, 처음 로그인할 때 다른 작업을 수행하기 전에 암호를 새 값으로 변경해야 합니다.

관리자 또는 개발자가 만든 사용자 계정은 이 상태에서 시작합니다.

비활성화됨

사용자 계정을 삭제하려면 먼저 해당 사용자에 대한 로그인 액세스를 비활성화해야 합니다.

추가 리소스

- Amazon Cognito를 사용하여 비활성 사용자 계정 감지 및 수정

가입 시 연락처 정보 확인

새 사용자가 앱에 가입할 때 적어도 하나의 연락 방법을 제공하도록 하고 싶을 것입니다. 예를 들어 사용자의 연락처 정보를 사용하여 다음을 수행할 수 있습니다.

- 사용자가 암호를 재설정하기로 선택한 경우 임시 암호를 보냅니다.
- 개인 정보 또는 재무 정보가 업데이트되면 사용자에게 알립니다.
- 특별 제품 및 서비스 또는 할인과 같은 프로모션 메시지를 보냅니다.
- 계정 요약 또는 결제 미리 알림을 보냅니다.

이와 같은 사용 사례의 경우 확인된 대상에 메시지를 보내는 것이 중요합니다. 그렇지 않으면 잘못 입력한 이메일 주소 또는 전화 번호로 메시지를 보내게 될 수 있습니다. 또는 최악의 경우, 사용자를 사칭하는 악의적인 행위자에게 민감한 정보를 보내게 될 수도 있습니다.

올바른 사람에게만 메시지를 보내려면 사용자가 가입할 때 다음을 반드시 제공하도록 Amazon Cognito 사용자 풀을 구성합니다.

a. 이메일 주소 또는 전화 번호.

b. Amazon Cognito가 이메일 주소나 전화 번호로 전송하는 인증 코드. 24시간이 지났고 사용자 코드 또는 링크가 더 이상 유효하지 않은 경우 ResendConfirmationCode API 작업을 호출하여 새 코드 또는 링크를 생성 및 전송합니다.

확인 코드를 제공하면 코드를 받은 메일함이나 휴대폰에 액세스할 수 있음을 사용자가 증명합니다. 사용자가 코드를 제공하고 나면 Amazon Cognito는 다음과 같은 방법으로 사용자 풀에서 사용자에 대한 정보를 업데이트합니다.

- 사용자의 상태를 `CONFIRMED`로 설정합니다.
- 이메일 주소나 전화 번호가 확인되었음을 나타내도록 사용자 속성을 업데이트합니다.

이 정보를 보려면 Amazon Cognito 콘솔을 사용하면 됩니다. 또는 `AdminGetUser` API 작업, `admin-get-user` 명령을와 함께 AWS CLI사용하거나 AWS SDKs.

사용자에게 확인된 연락 방법이 있는 경우 사용자가 암호 재설정을 요청할 때 Amazon Cognito가 자동으로 사용자에게 메시지를 보냅니다.

사용자 속성을 확인하고 확인하는 기타 작업

다음 사용자 활동은 사용자 속성을 확인합니다. 이러한 속성을 자동으로 확인하도록 설정할 필요는 없습니다. 나열된 작업은 모든 경우에 해당 속성을 확인된 것으로 표시합니다.

이메일 주소

1. 이메일 일회용 암호(OTP)를 사용하여 암호 없는 인증을 성공적으로 완료했습니다.
2. 이메일 OTP를 사용하여 다중 인증(MFA)을 성공적으로 완료했습니다.

전화번호

1. SMS OTP를 사용하여 암호 없는 인증을 성공적으로 완료했습니다.
2. SMS OTP로 MFA를 성공적으로 완료했습니다.

이메일 또는 전화 확인을 요구하도록 사용자 풀을 구성하려면

사용자의 이메일 주소와 전화번호를 확인할 때 사용자에게 연락할 수 있는지 확인합니다. 에서 다음 단계를 완료 AWS Management Console 하여 사용자가 이메일 주소 또는 전화번호를 확인하도록 요구하는 사용자 풀을 구성합니다.

> **Note**
>
> 아직 계정에 사용자 풀을 만들지 않았으면 Amazon Cognito 콘솔에서 새 애플리케이션 생성 섹션을 참조하세요.

사용자 풀을 구성하려면

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 자격 AWS 증명을 입력합니다.
2. 탐색 창에서 사용자 풀(User Pools)을 선택합니다. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
3. 가입 탭을 선택하고 속성 확인 및 사용자 계정 확인을 찾습니다. 편집을 선택합니다.
4. Cognito 지원 검증 및 확인에서 Cognito가 검증 및 확인을 위한 메시지를 자동으로 보내도록 허용 여부를 선택합니다. 이 설정을 활성화하면 Amazon Cognito는 사용자가 가입하거나 사용자 프로필을 생성할 때 선택한 사용자 연락처 속성으로 메시지를 보냅니다. 속성을 확인하고 로그인을 위한 사용자 프로필을 확인하기 위해 Amazon Cognito는 코드나 링크를 메시지로 사용자에게 보냅니다. 이후 사용자가 UI에 코드를 입력해야 앱이 `ConfirmSignUp` 또는 `AdminConfirmSignUp` API 요청에서 이를 확인할 수 있습니다.

   > **Note**
   >
   > [Cognito 지원 검증 및 확인(Cognito-assisted verification and confirmation)]을 사용하지 않도록 설정하고 인증된 API 작업이나 Lambda 트리거를 사용하여 속성을 검증하고 사용자를 확인할 수도 있습니다.
   > 이 옵션을 선택하면 사용자가 가입할 때 Amazon Cognito가 확인 코드를 보내지 않습니다. Amazon Cognito의 확인 코드를 사용하지 않고 하나 이상의 연락 방법을 확인하는 사용자 지정 인증 흐름을 사용하는 경우 이 옵션을 선택합니다. 예를 들어 특정 도메인에 속하는 이메일 주소를 자동으로 확인하는 사전 가입 Lambda 트리거를 사용할 수 있습니다.
   > 사용자의 연락처 정보를 확인하지 않으면 사용자가 앱을 사용하지 못할 수 있습니다. 다음을 위해 사용자로부터 확인된 연락처 정보를 요구하는 것입니다.

- [암호 재설정(Reset their passwords)] - 사용자가 앱에서 `ForgotPassword` API 작업을 호출하는 옵션을 선택하면 Amazon Cognito는 사용자의 이메일 주소나 전화 번호로 임시 암호를 전송합니다. Amazon Cognito는 사용자에게 확인된 연락 방법이 하나 이상 있는 경우에만 이 암호를 전송합니다.
- [이메일 주소나 전화 번호를 별칭으로 사용하여 로그인(Sign in by using an email address or phone number as an alias)] - 이러한 별칭을 허용하도록 사용자 풀을 구성하면 사용자는 별칭이 확인된 경우에만 별칭으로 로그인할 수 있습니다. 자세한 내용은 로그인 속성 사용자 지정 섹션을 참조하세요.

5. [확인할 속성(Attributes to verify)]을 선택합니다.

   [SMS 메시지 전송, 전화 번호 확인(Send SMS message, verify phone number)]

   Amazon Cognito에서 사용자가 가입할 때 SMS 메시지로 확인 코드를 전송합니다. 일반적으로 SMS 메시지를 통해 사용자와 커뮤니케이션하는 경우 이 옵션을 선택합니다. 예를 들어 배송 알림, 약속 확인 또는 알림을 보내는 경우 확인된 전화 번호를 사용할 수 있습니다. 사용자 전화 번호는 계정이 확인될 때 확인된 속성입니다. 사용자 이메일 주소를 확인하고 통신하려면 추가 조치를 취해야 합니다.

   [이메일 메시지 전송, 이메일 주소 확인(Send email message, verify email address)]

   Amazon Cognito에서 사용자가 가입할 때 이메일 메시지를 통해 확인 코드를 전송합니다. 일반적으로 이메일을 통해 사용자와 커뮤니케이션하는 경우 이 옵션을 선택합니다. 예를 들어 결제 청구서, 주문 요약 또는 특별 제품 및 서비스를 보내는 경우 확인된 이메일 주소를 사용할 수 있습니다. 사용자 이메일 주소는 계정이 확인될 때 확인된 속성입니다. 사용자 전화번호를 확인하고 통신하려면 추가 조치를 취해야 합니다.

   [전화 번호를 사용할 수 있는 경우 SMS 메시지 전송, 그렇지 않으면 이메일 메시지 전송(Send SMS message if phone number is available, otherwise send email message)]

   모든 사용자에게 동일한 확인된 연락 방법을 요구하지 않는 경우 이 옵션을 선택합니다. 이 경우 앱의 가입 페이지에서 사용자에게 선호하는 연락 방법만 확인하도록 요청할 수 있습니다. 확인 코드를 전송할 때 Amazon Cognito는 앱에서 `SignUp` 요청 시에 제공한 연락 수단으로 코드를 보냅니다. 사용자가 이메일 주소와 전화 번호를 모두 제공하고 앱이 `SignUp` 요청에서 두 가지 연락 수단을 모두 제공할 경우 Amazon Cognito는 전화 번호로만 확인 코드를 전송합니다.

   사용자가 이메일 주소와 전화 번호를 모두 확인하도록 요구하는 경우 이 옵션을 선택합니다. Amazon Cognito는 사용자가 등록할 때 하나의 연락 방법을 확인하며, 사용자가 로그인한 후

앱이 다른 연락 방법을 확인해야 합니다. 자세한 내용은 사용자가 이메일 주소와 전화 번호를 둘 다 확인하도록 요구하는 경우 섹션을 참조하세요.

6. 변경 사항 저장(Save changes)을 선택합니다.

이메일 또는 전화 확인을 사용하는 인증 흐름

사용자 풀에서 사용자가 연락처 정보를 확인해야 하는 경우 사용자가 가입할 때 앱은 다음과 같은 흐름을 지원해야 합니다.

1. 사용자가 사용자 이름, 전화 번호 및/또는 이메일 주소와 기타 속성을 입력하여 앱에 가입합니다.
2. Amazon Cognito 서비스가 앱으로부터 가입 요청을 받습니다. 요청에 가입에 필요한 모든 속성이 포함되어 있는지 확인한 후, 이 서비스는 가입 프로세스를 완료하고 사용자의 휴대폰(SMS 메시지) 또는 이메일로 확인 코드를 보냅니다. 이 코드는 24시간 동안 유효합니다.
3. 가입이 완료되었으며 사용자 계정이 확인 대기 중이라는 메시지를 서비스가 앱에 반환합니다. 해당 응답에는 확인 코드가 전송된 위치에 대한 정보가 포함됩니다. 이 시점에 사용자의 계정은 확인되지 않은 상태이며 사용자의 이메일 주소와 전화 번호가 확인되지 않았습니다.
4. 이제 앱에서 확인 코드를 입력하라는 메시지를 표시할 수 있습니다. 사용자가 코드를 즉시 입력할 필요는 없습니다. 단, 사용자는 확인 코드를 입력할 때까지 로그인할 수 없습니다.
5. 사용자가 앱에서 확인 코드를 입력합니다.
6. 앱이 `ConfirmSignUp`을 호출하여 코드를 Amazon Cognito 서비스로 전송합니다. 이 서비스는 코드를 확인하고 코드가 올바르면 사용자 계정을 확인 상태로 설정합니다. 사용자 계정을 성공적으로 확인한 후 Amazon Cognito 서비스는 확인하는 데 사용된 속성(이메일 또는 전화번호)을 자동으로 확인된 것으로 표시합니다. 이 속성의 값이 변경되지 않는 한, 사용자가 다시 확인할 필요가 없습니다.
7. 이 시점에 사용자 계정은 확인된 상태이며 사용자가 로그인할 수 있습니다.

사용자가 이메일 주소와 전화 번호를 둘 다 확인하도록 요구하는 경우

Amazon Cognito는 사용자가 가입할 때 하나의 연락 방법만 확인합니다. 이메일 주소 또는 전화번호 확인 중에서 선택해야 하는 경우 Amazon Cognito는 SMS 메시지를 통해 확인 코드를 전송하여 전화번호를 확인하도록 선택합니다. 예를 들어 사용자가 이메일 주소 또는 전화 번호를 확인할 수 있도록 사용자 풀을 구성하고 가입 시 앱이 이러한 속성을 모두 제공하는 경우 Amazon Cognito는 전화 번호만 확인합니다. 사용자가 전화번호를 확인하고 나면 Amazon Cognito가 사용자의 상태를 `CONFIRMED`로 설정하며, 사용자는 앱에 로그인할 수 있습니다.

사용자가 로그인하면 가입 중에 확인되지 않은 연락 방법을 확인하는 옵션을 앱에서 제공할 수 있습니다. 이 두 번째 수단을 확인하기 위해 앱은 `VerifyUserAttribute` API 작업을 호출합니다. 이 작업을 수행하려면 `AccessToken` 파라미터가 필요하며, Amazon Cognito는 인증된 사용자에 대한 액세스 토큰만 제공합니다. 따라서 사용자가 로그인한 후에만 두 번째 연락 방법을 확인할 수 있습니다.

사용자가 이메일 주소와 전화 번호를 모두 확인하도록 요구하는 경우 다음을 수행합니다.

1. 사용자가 이메일 주소 또는 전화 번호를 확인할 수 있도록 사용자 풀을 구성합니다.
2. 앱의 가입 흐름에서 사용자에게 이메일 주소와 전화 번호를 모두 제공하도록 요구합니다. `SignUp` API 작업을 호출하고 `UserAttributes` 파라미터에서 이메일 주소와 전화 번호를 제공합니다. 이때 Amazon Cognito는 사용자의 휴대폰으로 확인 코드를 전송합니다.
3. 앱 인터페이스에서 사용자가 확인 코드를 입력하는 확인 페이지를 표시합니다. `ConfirmSignUp` API 작업을 호출하여 사용자를 확인합니다. 이 시점에 사용자의 상태는 `CONFIRMED`이며 사용자의 전화 번호는 확인되었지만 이메일 주소는 확인되지 않았습니다.
4. 로그인 페이지를 표시하고 `InitiateAuth` API 작업을 호출하여 사용자를 인증합니다. 사용자가 인증되면 Amazon Cognito는 앱에 액세스 토큰을 반환합니다.
5. `GetUserAttributeVerificationCode` API 작업을 호출합니다. 요청에 다음 파라미터를 지정합니다.

   - `AccessToken` - 사용자가 로그인할 때 Amazon Cognito가 반환하는 액세스 토큰입니다.
   - `AttributeName` - `"email"`을 속성 값으로 지정합니다.

   Amazon Cognito가 사용자의 이메일 주소로 확인 코드를 전송합니다.
6. 사용자가 확인 코드를 입력하는 확인 페이지를 표시합니다. 사용자가 코드를 제출하면 `VerifyUserAttribute` API 작업을 호출합니다. 요청에 다음 파라미터를 지정합니다.

   - `AccessToken` - 사용자가 로그인할 때 Amazon Cognito가 반환하는 액세스 토큰입니다.
   - `AttributeName` - `"email"`을 속성 값으로 지정합니다.
   - `Code` - 사용자가 제공한 확인 코드입니다.

   이 시점에 이메일 주소는 확인된 상태입니다.

사용자가 앱에 가입할 수 있도록 허용하지만 사용자 풀 관리자로 확인

사용자 풀이 사용자 풀에 자동으로 확인 메시지를 보내는 것을 원하지 않지만 누구나 계정에 가입하도록 허용하려는 경우가 있을 수 있습니다. 이 모델에서는 예를 들어 신규 가입 요청을 사람이 직접 검토

하고 가입을 일괄적으로 검증하고 처리할 수 있는 여지가 남아 있습니다. Amazon Cognito 콘솔에서 또는 IAM 인증 API 작업 AdminConfirmSignup을 사용하여 새 사용자 계정을 확인할 수 있습니다. 사용자 풀의 확인 메시지 전송 여부에 관계없이 관리자로서 사용자 계정을 확인할 수 있습니다.

이 방법을 통해서만 사용자 셀프 서비스 가입을 확인할 수 있습니다. 관리자로 생성한 사용자를 확인하려면 `Permanent`를 `True`로 설정하여 AdminSetUserPassword API 요청을 생성합니다.

1. 사용자가 사용자 이름, 전화 번호 및/또는 이메일 주소와 기타 속성을 입력하여 앱에 가입합니다.
2. Amazon Cognito 서비스가 앱으로부터 가입 요청을 받습니다. 요청에 가입에 필요한 모든 속성이 포함되어 있는지 확인한 후, 이 서비스는 가입 프로세스를 완료하고 등록이 완료되었으며 확인 대기 중이라는 메시지를 앱에 반환합니다. 이 시점에 사용자의 계정은 확인되지 않은 상태입니다. 사용자는 계정이 확인될 때까지 로그인할 수 없습니다.
3. 사용자의 계정을 확인합니다. 계정을 확인하려면에 로그인 AWS Management Console 하거나 자격 AWS 증명으로 API 요청에 서명해야 합니다.

    a. Amazon Cognito 콘솔에서 사용자를 확인하려면 사용자 메뉴로 이동하여 확인하려는 사용자를 선택한 다음 작업 메뉴에서 확인을 선택합니다.

    b. AWS API 또는 CLI에서 사용자를 확인하려면에서 AdminConfirmSignUp API 요청 또는 admin-confirm-sign-up을 생성합니다 AWS CLI.
4. 이 시점에 사용자 계정은 확인된 상태이며 사용자가 로그인할 수 있습니다.

보안 암호 해시 값 컴퓨팅

모범 사례에 따라 기밀 앱 클라이언트에 클라이언트 보안 암호를 할당합니다. 앱 클라이언트에 클라이언트 보안 암호를 할당할 때, Amazon Cognito 사용자 풀 API 요청에는 클라이언트 보안 암호가 포함된 해시가 요청 본문에 포함되어야 합니다. 다음 목록의 API 작업에 대한 클라이언트 보안 암호를 알고 있는지 확인하려면, 클라이언트 보안 암호를 앱 클라이언트 ID 및 사용자의 사용자 이름과 연결한 다음 문자열을 base64로 인코딩합니다.

앱이 보안 암호 해시가 있는 클라이언트에 사용자를 로그인시키는 경우 모든 사용자 풀 로그인 속성의 값을 암호 해시의 사용자 이름 요소로 사용할 수 있습니다. 앱이 `REFRESH_TOKEN_AUTH`가 있는 인증 작업에서 새 토큰을 요청하는 경우 사용자 이름 요소의 값은 로그인 속성에 따라 달라집니다. 사용자 풀에 로그인 속성으로 `username`이 없는 경우 액세스 또는 ID 토큰의 사용자 `sub` 클레임에서 보안 암호 해시 사용자 이름 값을 설정하십시오. `username`가 로그인 속성인 경우에는 `username` 클레임에서 보안 암호 해시 사용자 이름 값을 설정합니다.

다음 Amazon Cognito 사용자 풀 API는 `SecretHash` 파라미터의 클라이언트 보안 암호 해시 값을 허용합니다.

- ConfirmForgotPassword
- ConfirmSignUp
- ForgotPassword
- ResendConfirmationCode
- SignUp

또한 다음 API는 인증 파라미터 또는 챌린지 응답의SECRET_HASH 파라미터에 클라이언트 보안 암호 해시 값을 허용합니다.

API 작업	SECRET_HASH의 상위 파라미터
InitiateAuth	AuthParameters
AdminInitiateAuth	AuthParameters
RespondToAuthChallenge	ChallengeResponses
AdminRespondToAuthChallenge	ChallengeResponses

보안 암호 해시 값은 사용자 풀 클라이언트 및 사용자 이름의 보안 암호 키와 메시지의 클라이언트 ID를 사용하여 계산된 Base 64 인코딩 키 추가 해시 메시지 인증 코드(HMAC)입니다. 다음 유사 코드는 이 값이 계산되는 방식을 보여줍니다. 이 유사 코드에서 +는 연결을 나타내고, HMAC_SHA256은 HmacSHA256을 사용해 HMAC 값을 생성하는 기능을 나타내며, Base64는 해시 출력의 Base-64 인코딩 버전을 생성하는 기능을 나타냅니다.

```
Base64 ( HMAC_SHA256 ( "Client Secret Key", "Username" + "Client Id" ) )
```

SecretHash 파라미터를 계산하고 사용하는 방법에 대한 자세한 개요는 Amazon Cognito 사용자 풀 API에서 “클라이언트 <client-id>의 보안 암호 해시를 확인할 수 없음” 오류를 해결하려면 어떻게 해야 하나요?를 참조하세요. AWS 지식 센터의 .

서버 측 앱 코드에서 다음 코드 예를 사용할 수 있습니다.

Shell

```
echo -n "[username][app client ID]" | openssl dgst -sha256 -hmac [app client secret]
 -binary | openssl enc -base64
```

Java

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

public static String calculateSecretHash(String userPoolClientId, String
 userPoolClientSecret, String userName) {
    final String HMAC_SHA256_ALGORITHM = "HmacSHA256";

    SecretKeySpec signingKey = new SecretKeySpec(
            userPoolClientSecret.getBytes(StandardCharsets.UTF_8),
            HMAC_SHA256_ALGORITHM);
    try {
        Mac mac = Mac.getInstance(HMAC_SHA256_ALGORITHM);
        mac.init(signingKey);
        mac.update(userName.getBytes(StandardCharsets.UTF_8));
        byte[] rawHmac =
 mac.doFinal(userPoolClientId.getBytes(StandardCharsets.UTF_8));
        return Base64.getEncoder().encodeToString(rawHmac);
    } catch (Exception e) {
        throw new RuntimeException("Error while calculating ");
    }
}
```

Python

```
import sys
import hmac, hashlib, base64
username = sys.argv[1]
app_client_id = sys.argv[2]
key = sys.argv[3]
message = bytes(sys.argv[1]+sys.argv[2],'utf-8')
key = bytes(sys.argv[3],'utf-8')
secret_hash = base64.b64encode(hmac.new(key, message,
 digestmod=hashlib.sha256).digest()).decode()
print("SECRET HASH:",secret_hash)
```

이메일 또는 전화 번호를 확인하지 않고 사용자 계정 확인

Lambda 사전 가입 트리거를 사용하면 확인 코드를 요구하거나 이메일 또는 전화번호를 확인하지 않고도 가입 시 사용자 계정을 자동으로 확인할 수 있습니다. 이 방법으로 확인된 사용자는 코드를 받지 않고도 즉시 로그인할 수 있습니다.

이 트리거를 통해 확인된 사용자의 이메일 또는 전화 번호를 표시할 수도 있습니다.

> **Note**
>
> 이 방법은 사용자가 시작할 때 편리하지만 이메일 또는 전화 번호 중 하나 이상을 자동 확인하는 것이 좋습니다. 그렇지 않으면 사용자가 암호를 잊어 버린 경우 복구할 수 없게 될 수 있습니다.

사용자가 가입 시 확인 코드를 받아 입력하도록 요구하지 않고 Lambda 사전 가입 트리거에서 이메일 및 전화번호를 자동으로 확인하지 않으면, 해당 사용자 계정에 대해 확인된 이메일 주소 또는 전화번호가 없을 위험이 있습니다. 이 경우 사용자가 나중에 이메일 주소 또는 전화 번호를 확인할 수 있습니다. 그러나 사용자가 암호를 잊어버렸는데 확인된 이메일 주소 또는 전화번호가 없는 경우 암호 찾기 흐름에서 사용자에게 확인 코드를 보내는 데 확인된 이메일 또는 전화번호가 필요하기 때문에 사용자의 계정이 잠깁니다.

사용자가 이메일 또는 전화 번호를 변경하는 경우 확인

여러 로그인 이름으로 구성한 사용자 풀에서 사용자는 로그인 시 전화번호 또는 이메일 주소를 사용자 이름으로 입력할 수 있습니다. 사용자가 앱에서 이메일 주소 또는 전화번호를 업데이트하면 Amazon Cognito는 새 속성 값의 소유권을 확인하는 코드가 포함된 메시지를 즉시 보낼 수 있습니다. 이러한 확인 코드의 자동 전송을 활성화하려면 이메일 또는 전화 확인 구성 섹션을 참조하세요.

확인 코드를 받는 사용자는 VerifyUserAttribute 요청에서 해당 코드를 Amazon Cognito에 다시 제공해야 합니다. 코드를 제공하면 해당 속성이 확인된 것으로 표시됩니다. 일반적으로 사용자가 이메일 주소 또는 전화번호를 업데이트할 때 새 값을 소유하고 있는지 확인한 후에 로그인하고 메시지를 수신할 수 있습니다. 사용자 풀에는 사용자가 이메일 주소 또는 전화번호에 대한 업데이트를 확인해야 하는지 여부를 결정하는 구성 가능한 옵션이 있습니다.

이 옵션은 사용자 풀 속성 `AttributesRequireVerificationBeforeUpdate`입니다. CreateUserPool 또는 UpdateUserPool 요청에서 구성하거나 Amazon Cognito 콘솔의 가입 메뉴에서 업데이트가 보류 중일 때 원래 속성 값 활성 유지를 설정하여 구성합니다.

사용자 풀이 이메일 주소에 대한 업데이트를 처리하는 방법 및 전화번호는 사용자 풀의 사용자 이름 구성에 연결됩니다. 사용자 풀 사용자 이름은 로그인 이름이 이메일 주소, 전화번호 또는 둘 다인 사용자 이름 속성 구성일 수 있습니다. 또한 `username` 속성이 이메일 주소, 전화번호 또는 대체 로그인 이름으로 선호하는 사용자 이름과 함께 로그인 이름인 별칭 속성 구성에 있을 수 있습니다. 자세한 내용은 로그인 속성 사용자 지정 섹션을 참조하세요.

또한 사용자 지정 메시지 Lambda 트리거를 사용하여 확인 메시지를 사용자 지정할 수 있습니다. 자세한 내용은 사용자 정의 메시지 Lambda 트리거 섹션을 참조하세요. 사용자의 이메일 주소나 전화번호가 확인되지 않은 경우, 애플리케이션은 사용자에게 해당 속성을 확인해야 한다는 사실을 알리고, 사용자가 확인 코드를 입력할 수 있는 버튼이나 링크를 제공해야 합니다.

다음 표에서는 `AttributesRequireVerificationBeforeUpdate` 및 별칭 설정이 사용자가 로그인 속성의 값을 변경할 때 결과를 결정하는 방법을 설명합니다.

사용자 이름 구성	사용자가 새 속성을 확인해야 하는 동작	사용자가 새로운 속성을 확인할 필요가 없는 경우의 동작
사용자 이름 속성	원래 속성은 확인되고 로그인에 적합하며 원래 값으로 유지됩니다. 사용자가 새 값을 확인하면 Amazon Cognito는 속성 값을 업데이트하고 확인된 것으로 표시하며 로그인에 적합합니다.	Amazon Cognito는 속성을 새 값으로 업데이트합니다. 새 값은 로그인할 수 있습니다. 사용자가 새 값을 확인하면 Amazon Cognito는 해당 값을 확인된 것으로 표시합니다.
별칭 속성	원래 속성은 확인되고 로그인에 적합하며 원래 값으로 유지됩니다. 사용자가 새 값을 확인하면 Amazon Cognito는 속성 값을 업데이트하고 확인된 것으로 표시하며 로그인에 적합합니다.	Amazon Cognito는 속성을 새 값으로 업데이트합니다. 원래 속성 값이나 새 속성 값 모두 로그인할 수 없습니다. 사용자가 새 값을 확인하면 Amazon Cognito는 속성 값을 업데이트하고 확인된 것으로 표시하며 로그인에 적합합니다.

예제 1.

사용자 1은 이메일 주소 `user1@example.com`으로 애플리케이션에 로그인하고 사용자 이름 `user1`(별칭 속성)을 갖습니다. 사용자 풀은 로그인 속성에 대한 업데이트를 확인하고 확인 메

시지를 자동으로 전송하도록 구성됩니다. 이메일 주소를 user1+foo@example.com으로 업데이트하도록 요청합니다. user1+foo@example.com에서 확인 이메일을 수신하고 이메일 주소 user1@example.com으로만 다시 로그인할 수 있습니다. 나중에 확인 코드를 입력하고 이메일 주소 user1+foo@example.com으로만 다시 로그인할 수 있습니다.

예제 2.

사용자 2는 이메일 주소 user2@example.com으로 애플리케이션에 로그인하고 사용자 이름 (별칭 속성)을 갖습니다. 사용자 풀은 로그인 속성에 대한 업데이트를 확인하지 않고 확인 메시지를 자동으로 보내도록 구성되어 있습니다. 이메일 주소를 user2+bar@example.com으로 업데이트하도록 요청합니다. user2+bar@example.com에서 확인 이메일을 수신하고 다시 로그인할 수 없습니다. 나중에 확인 코드를 입력하고 이메일 주소 user2+bar@example.com으로만 다시 로그인할 수 있습니다.

예제 3

사용자 3은 이메일 주소 user3@example.com으로 애플리케이션에 로그인하고 사용자 이름(사용자 이름 속성)이 없습니다. 사용자 풀은 로그인 속성에 대한 업데이트를 확인하지 않고 확인 메시지를 자동으로 보내도록 구성되어 있습니다. 이메일 주소를 user3+baz@example.com으로 업데이트하도록 요청합니다. user3+baz@example.com에서 확인 이메일을 수신하지만 확인 코드로 추가 조치를 취하지 않고 즉시 로그인할 수 있습니다.

관리자 또는 개발자가 생성한 사용자 계정에 대한 확인 및 검증 프로세스

관리자 또는 개발자가 만든 사용자 계정은 이미 확인 상태이므로 사용자가 확인 코드를 입력할 필요가 없습니다. Amazon Cognito 서비스가 이러한 사용자에게 보내는 초대 메시지에는 사용자 이름과 임시 암호가 포함됩니다. 사용자는 로그인하기 전에 이 암호를 변경해야 합니다. 자세한 내용은 관리자로 사용자 계정 생성의 이메일 및 SMS 메시지 사용자 정의 섹션과 Lambda 트리거를 사용하여 사용자 풀 워크플로 사용자 정의의 사용자 지정 메시지 트리거 섹션을 참조하세요.

가져온 사용자 계정에 대한 확인 및 검증 프로세스

AWS Management Console, CLI 또는 API(참조CSV 파일에서 사용자 풀로 사용자 가져오기)에서 사용자 가져오기 기능을 사용하여 생성된 사용자 계정은 이미 확인된 상태이므로 사용자는 확인 코드를 입력할 필요가 없습니다. 초대 메시지가 전송되지 않습니다. 그러나 가져온 사용자 계정은 사용자가 먼저 ForgotPassword API를 호출하여 코드를 요청한 다음, ConfirmForgotPassword API를 호출하고 전달된 코드를 사용하여 암호를 만들어야 로그인할 수 있습니다. 자세한 내용은 가져온 사용자에게 암호 재설정 요구 섹션을 참조하세요.

사용자 계정을 가져올 때 사용자의 이메일 또는 전화 번호가 확인된 상태로 표시되어야 하므로 사용자가 로그인 시에는 확인이 필요하지 않습니다.

앱을 테스트하는 동안 이메일 전송

Amazon Cognito는 클라이언트 앱에서 사용자 풀을 위해 계정을 생성하고 관리할 때 사용자에게 이메일 메시지를 보냅니다. 이메일 확인을 요구하도록 사용자 풀을 구성하는 경우 Amazon Cognito는 다음과 같은 경우 이메일을 보냅니다.

- 사용자가 가입할 경우.
- 사용자가 이메일 주소를 업데이트할 경우.
- 사용자가 `ForgotPassword` API 작업을 호출하는 작업을 수행할 경우.
- 사용자 계정을 관리자로 생성할 경우.

이메일을 시작한 작업이 무엇인지에 따라 이메일에는 확인 코드가 들어가거나 임시 암호가 들어갑니다. 사용자는 이러한 이메일을 수신하고 메시지를 이해해야 합니다. 그렇지 않으면 로그인하여 앱을 사용하지 못할 수 있습니다.

이메일이 성공적으로 전송되고 메시지가 올바른지 확인하려면 앱에서 Amazon Cognito로부터의 이메일 전송을 시작하는 작업을 테스트합니다. 예를 들어 앱에서 가입 페이지를 사용하거나 `SignUp` API 작업을 사용하면, 테스트 이메일 주소로 등록하여 이메일을 시작할 수 있습니다. 이 방법으로 테스트하려면 다음을 기억해야 합니다.

> **중요**
>
> 이메일 주소를 사용하여 Amazon Cognito에서 이메일을 시작하는 작업을 테스트하는 경우, 가짜 이메일 주소(메일박스가 없는 이메일 주소)를 사용하지 마세요. Amazon Cognito로부터 이메일을 수신할 실제 이메일 주소를 사용하여 하드 바운스를 유발하지 않도록 합니다.
> 하드 바운스 메일은 Amazon Cognito가 수신자의 메일박스로 이메일을 전송하지 못할 때 발생하며, 메일박스가 없는 경우 항상 발생합니다.
> Amazon Cognito는 지속적으로 하드 바운스가 발생하는 AWS 계정에서 보낼 수 있는 이메일 수를 제한합니다.

이메일을 시작하는 작업을 테스트할 때 다음 이메일 주소 중 하나를 사용하여 하드 바운스를 방지합니다.

- 자신이 소유하고 테스트에 사용하는 이메일 계정의 주소. 자신의 이메일 주소를 사용하면 Amazon Cognito에서 보내는 이메일을 받게 됩니다. 이 이메일을 통해 인증 코드를 사용하여 앱의 가입 환경을 테스트할 수 있습니다. 사용자 풀에 대한 이메일 메시지를 사용자 지정한 경우 사용자 지정 내용이 올바른지 확인할 수 있습니다.

- 메일박스 시뮬레이터 주소, success@simulator.amazonses.com. 시뮬레이터 주소를 사용하는 경우 Amazon Cognito가 이메일을 성공적으로 전송하지만 사용자는 이메일을 볼 수 없습니다. 이 옵션은 인증 코드를 사용할 필요가 없고 이메일 메시지를 확인할 필요가 없을 때 유용합니다.
- 임의의 레이블이 추가된 메일박스 시뮬레이터 주소, 예: success+user1@simulator.amazonses.com 또는 success+user2@simulator.amazonses.com. Amazon Cognito가 이들 주소로 이메일을 성공적으로 전송하지만 사용자는 전송된 이메일을 볼 수 없습니다. 이 옵션은 사용자 풀에 여러 테스트 사용자를 추가하여 가입 프로세스를 테스트하며, 각 테스트 사용자에게 고유한 이메일 주소가 있는 경우에 유용합니다.

이메일 또는 전화 확인 구성

인증 방법 메뉴에서 이메일 또는 전화 확인 설정을 선택할 수 있습니다. 멀티 팩터 인증(MFA)에 대한 자세한 내용은 SMS Text Message MFA(SMS 문자 메시지 MFA)를 참조하세요.

Amazon Cognito는 Amazon SNS를 사용하여 SMS 메시지를 전송합니다. AWS 서비스 이전에 Amazon Cognito 또는 다른에서 SMS 메시지를 전송하지 않은 경우 Amazon SNS가 SMS 샌드박스에 계정을 배치할 수 있습니다. 샌드박스에서 프로덕션으로 계정을 제거하기 전에 확인된 전화 번호로 테스트 메시지를 보내는 것이 좋습니다. 또한 SMS 메시지를 미국 대상 전화 번호로 보내려는 경우 Amazon Pinpoint에서 발신 또는 발신자 ID를 받아야 합니다. SMS 메시지를 사용하도록 Amazon Cognito 사용자 풀을 구성하려면 Amazon Cognito 사용자 풀의 SMS 메시지 설정 섹션을 참조하세요.

Amazon Cognito는 이메일 주소 또는 전화번호를 자동으로 확인할 수 있습니다. 이 확인 작업을 수행하기 위해 Amazon Cognito에서는 확인 코드를 보내거나 확인 링크를 보냅니다. 이메일 주소의 경우 Amazon Cognito에서 코드 또는 링크를 이메일 메시지에 넣어 보냅니다. Amazon Cognito 콘솔의 메시지 템플릿 메뉴에서 인증 메시지 템플릿을 편집할 때 인증 유형의 코드 또는 링크를 선택할 수 있습니다. 자세한 내용은 이메일 확인 메시지 사용자 정의 단원을 참조하십시오.

전화번호의 경우 Amazon Cognito에서 코드를 SMS 문자 메시지에 넣어 보냅니다.

Amazon Cognito는 사용자를 확인하고 사용자가 잊어버린 암호를 복구할 수 있도록 전화 번호 또는 이메일 주소를 확인해야 합니다. 또는 사전 가입 Lambda 트리거나 AdminConfirmSignUp API 작업을 사용하여 사용자를 자동으로 확인할 수 있습니다. 자세한 내용은 사용자 계정 가입 및 확인 단원을 참조하십시오.

확인 코드나 링크는 24시간 동안 유효합니다.

이메일 주소 또는 전화 번호를 확인하도록 선택하면 사용자가 가입할 때 Amazon Cognito에서 확인 코드 또는 링크를 자동으로 보냅니다. 사용자 풀에 사용자 지정 SMS 발신자 Lambda 트리거 또는 사용자 정의 이메일 발신자 Lambda 트리거가 구성된 경우 해당 함수가 대신 호출됩니다.

참고

- Amazon SNS는 전화 번호 확인을 위해 사용하는 SMS 문자 메시지 사용 요금을 별도로 청구합니다. 이메일 메시지는 무료로 보낼 수 있습니다. Amazon SNS 요금에 대한 자세한 내용은 전 세계 SMS 요금을 참조하세요. 현재 SMS 메시징을 사용할 수 있는 국가의 목록은 지원되는 리전 및 국가를 참조하세요.
- Amazon Cognito에서 이메일 메시지를 생성하는 작업을 앱에서 테스트하는 경우 Amazon Cognito에서 하드 바운스 없이 연결할 수 있는 실제 이메일 주소를 사용합니다. 자세한 내용은 the section called “앱을 테스트하는 동안 이메일 전송” 섹션을 참조하세요.
- 암호 찾기 흐름에서는 사용자를 확인하기 위해 사용자의 이메일 또는 전화 번호를 요구합니다.

Important

사용자가 전화번호 및 이메일 주소 둘 다를 사용하여 가입했으며 사용자 풀 설정에서 이 두 속성의 확인을 요구하는 경우 Amazon Cognito에서 확인 코드를 SMS 메시지를 통해 전화번호로 보냅니다. Amazon Cognito에선 아직 이메일 주소를 확인하지 않았기 때문에 앱에서 GetUser를 호출하여 이메일 주소가 확인을 기다리는지 여부를 확인해야 합니다. 확인이 필요한 경우 앱에서 GetUserAttributeVerificationCode를 호출하여 이메일 확인 흐름을 시작해야 합니다. 그런 다음 VerifyUserAttribute를 호출하여 확인 코드를 제출해야 합니다.

AWS 계정 및 개별 메시지에 대한 SMS 메시지 지출 할당량을 조정할 수 있습니다. 이러한 한도는 SMS 메시지 전송 비용에만 적용됩니다. 자세한 내용은 Amazon SNS FAQ의 계정 수준 및 메시지 수준의 지출 할당량이란 무엇이며 어떻게 작동합니까?를 참조하세요.

Amazon Cognito AWS 리전 는 사용자 풀을 생성한 또는 다음 표의 레거시 Amazon SNS 대체 리전에서 Amazon SNS 리소스를 사용하여 SMS 메시지를 전송합니다. Amazon SNS 아시아 태평양(서울) 리전에 있는 Amazon Cognito 사용자 풀은 예외입니다. 이러한 사용자 풀은 Amazon SNS 구성을 아시아 태평양(도쿄) 리전에서 사용합니다. 자세한 내용은 SMS 메시지 AWS 리전 용 선택 섹션을 참조하세요.

Amazon Cognito 리전	레거시 Amazon SNS 대체 리전
미국 동부(오하이오)	미국 동부(버지니아 북부)

Amazon Cognito 리전	레거시 Amazon SNS 대체 리전
아시아 태평양(뭄바이)	아시아 태평양(싱가포르)
아시아 태평양(서울)	아시아 태평양(도쿄)
캐나다(중부)	미국 동부(버지니아 북부)
유럽(프랑크푸르트)	유럽(아일랜드)
유럽(런던)	유럽(아일랜드)

예: Amazon Cognito 사용자 풀이 아시아 태평양(뭄바이)에 있고 ap-southeast-1의 지출 한도가 늘어난 경우 ap-southeast-1의 별도 증가를 요청하지 않을 수 있습니다. 대신 Amazon SNS 리소스를 아시아 태평양(싱가포르)에서 사용할 수 있습니다.

이메일 주소 및 전화번호에 대한 업데이트 확인

이메일 주소 또는 전화번호 속성은 사용자가 값을 변경한 직후에 미확인 상태로 활성화됩니다. 또한 Amazon Cognito에서 속성을 업데이트하기 전에 사용자가 새 값을 확인하도록 요구할 수도 있습니다. 사용자가 먼저 새 값을 확인하도록 요청하는 경우 원래 값을 사용하여 로그인하고 새 값을 확인할 때까지 메시지를 수신할 수 있습니다.

사용자가 이메일 주소 또는 전화번호를 사용자 풀의 로그인 별칭으로 사용할 수 있는 경우 업데이트된 속성의 로그인 이름은 업데이트된 속성의 확인이 필요한지에 따라 달라집니다. 업데이트된 속성을 확인해야 하는 경우 사용자는 새 값을 확인할 때까지 원래 속성 값으로 로그인할 수 있습니다. 사용자가 업데이트된 속성을 확인하지 않아도 되는 경우 사용자는 새 값을 확인할 때까지 새 속성 값이나 원래 속성 값으로 로그인하거나 메시지를 받을 수 없습니다.

예를 들어, 사용자 풀은 이메일 주소 별칭으로 로그인을 허용하고 사용자가 업데이트할 때 이메일 주소를 확인해야 합니다. `sue@example.com`으로 로그인하는 Sue는 이메일 주소를 `sue2@example.com`으로 바꾸기를 원했지만, 실수로 **s**`sue2@example.com`을 입력했습니다. Sue는 확인 이메일을 받지 못하니 `ssue2@example.com`을 확인할 수가 없습니다. Sue는 `sue@example.com`으로 로그인한 후 앱에서 양식을 다시 제출하여 이메일 주소를 `sue2@example.com`으로 업데이트합니다. 확인 이메일을 받고 앱에 확인 코드를 입력한 그녀는 이제 `sue2@example.com`으로 로그인하기 시작합니다.

사용자가 속성을 업데이트하고 사용자 풀이 새 속성 값을 검증하는 경우

- 새 값을 검증하기 위한 코드를 확인하기 전에 원래 속성 값으로 로그인할 수 있습니다.

- 새 값을 검증하기 위한 코드를 확인한 후에만 새 속성 값으로 로그인할 수 있습니다.
- AdminUpdateUserAttributes API 요청에서 `email_verified` 또는`phone_number_verified`를 `true`로 설정하면 Amazon Cognito가 전송한 코드를 확인하기 전에 로그인할 수 있습니다.

사용자가 속성을 업데이트하고 사용자 풀이 새 속성 값을 검증하지 않는 경우

- 원래 속성 값으로 로그인하거나 메시지를 수신할 수 없습니다.
- 새 값을 검증하기 위한 코드를 확인하기 전에는 새 속성 값으로 로그인하거나 확인 코드 이외의 메시지를 받을 수 없습니다.
- AdminUpdateUserAttributes API 요청에서 `email_verified` 또는`phone_number_verified`를 `true`로 설정하면 Amazon Cognito가 전송한 코드를 확인하기 전에 로그인할 수 있습니다.

사용자가 이메일 주소 또는 전화번호를 업데이트할 때 속성 확인을 요청하려면

1. Amazon Cognito 콘솔에 로그인합니다. 메시지가 표시되면 자격 AWS 증명을 입력합니다.
2. 탐색 창에서 [사용자 풀(User Pools)]을 선택한 다음 편집할 사용자 풀을 선택합니다.
3. 가입 메뉴의 속성 확인 및 사용자 계정 확인에서 편집을 선택합니다.
4. 업데이트가 보류 중일 때 원래 속성 값을 활성 상태로 유지(Keep original attribute value active when an update is pending)를 선택합니다.
5. 업데이트가 보류 중일 때 활성 속성 값(Active attribute values when an update is pending)에서 Amazon Cognito가 값을 업데이트하기 전에 사용자가 확인해야 할 속성을 선택합니다.
6. 변경 사항 저장을 선택합니다.

Amazon Cognito API를 사용하여 속성 업데이트를 확인하도록 하려면 UpdateUserPool 요청에 `AttributesRequireVerificationBeforeUpdate` 파라미터를 설정하면 됩니다.

Amazon Cognito에 자동으로 SMS 메시지를 전송할 수 있는 권한 부여

Amazon Cognito가 자동으로 사용자에게 SMS 메시지를 보내려면 권한이 필요합니다. 해당 권한을 부여하려면 AWS Identity and Access Management (IAM) 역할을 생성할 수 있습니다. Amazon Cognito 콘솔의 인증 방법 메뉴에서 SMS에서 편집을 선택하여 역할을 설정합니다.

MFA, 인증, 확인 및 초대 메시지 구성

Amazon Cognito를 사용하면 SMS 및 이메일 인증, 검증, 사용자 초대 메시지를 사용자 지정하여 애플리케이션의 보안과 사용자 경험을 강화할 수 있습니다. 일부 메시지에 대한 코드 기반 및 원클릭 링크 확인 중에서 선택할 수 있습니다. 이 주제에서는 Amazon Cognito 콘솔에서 인증 및 확인 통신을 개인화하는 방법을 설명합니다.

메시지 템플릿 메뉴에서 다음을 사용자 지정할 수 있습니다.

- 일회용 암호(OTP) 및 다중 인증(MFA)을 위한 이메일 및 SMS 메시지 템플릿
- SMS 및 이메일 확인 메시지
- 이메일의 확인 유형(코드 또는 링크)

> **Note**
>
> Amazon Cognito는 사용자가 확인 코드를 가입하거나 재전송할 때 확인 메시지에서 링크 기반 템플릿이 포함된 링크를 전송합니다. 속성 업데이트 및 암호 재설정 작업의 이메일은 코드 템플릿을 사용합니다.

- 사용자 초대 메시지
- 사용자 풀을 통해 전송되는 이메일의 FROM 및 REPLY-TO 이메일 주소

> **Note**
>
> SMS 및 이메일 확인 메시지 템플릿은 전화 번호 및 이메일 확인을 요구하도록 선택한 경우에만 표시됩니다. 마찬가지로, SMS MFA 메시지 템플릿은 MFA 설정이 [필수(required)] 또는 [선택 사항(optional)]인 경우에만 표시됩니다.

주제

- 메시지 템플릿
- 이메일 및 SMS MFA 메시지 사용자 지정
- 이메일 확인 메시지 사용자 정의
- 사용자 초대 메시지 사용자 정의
- 이메일 주소 사용자 정의

- Amazon Cognito가 사용자를 대신하여 (사용자 정의 FROM 이메일 주소에서) Amazon SES 이메일을 전송하도록 권한 부여

메시지 템플릿

메시지 템플릿을 사용하여 메시지에 자리 표시자를 삽입할 수 있습니다. Amazon Cognito는 자리 표시자를 해당 값으로 바꿉니다. 든 유형의 메시지 템플릿에서 범용 템플릿 자리 표시자를 참조할 수 있지만 이러한 값이 모든 메시지 유형에 있는 것은 아닙니다.

범용 템플릿 자리 표시자

설명	토큰	메시지 유형
확인 코드	`{####}`	검증, 확인 및 MFA 메시지
임시 암호	`{####}`	암호 분실 및 초대 메시지
사용자 이름	`{username}`	초대 및 고급 보안 메시지

위협 방지 기능이 있는 사용 가능한 자동 응답 중 하나는 Amazon Cognito가 잠재적으로 악의적인 활동을 탐지했음을 사용자에게 알리는 것입니다. 고급 보안 템플릿 자리 표시자를 사용하여 다음을 수행할 수 있습니다.

- IP 주소, 도시, 국가, 로그인 시간 및 디바이스 이름과 같은 이벤트에 대한 특정 세부 정보를 포함합니다. Amazon Cognito 위협 보호는 이러한 세부 정보를 분석할 수 있습니다.
- 원클릭 링크가 유효한지 확인할 수 있습니다.
- 이벤트 ID, 피드백 토큰 및 사용자 이름을 사용하여 자체 원클릭 링크를 빌드합니다.

> **Note**
>
> 원클릭 링크를 생성하고 고급 보안 이메일 템플릿에서 `{one-click-link-valid}` 및 `{one-click-link-invalid}` 자리 표시자를 사용하려면 이미 사용자 풀에 대한 도메인이 구성되어 있어야 합니다.

위협 방지는 메시지 템플릿에 삽입할 수 있는 다음과 같은 자리 표시자를 추가합니다. 이러한 자리 표시자는 Amazon Cognito가 위험 수준에 대해 세션이 평가된 사용자에게 보내는 알림인 적응형 인증 메

시지에 적용됩니다. 이러한 변수로 메시지 템플릿을 구성하려면 Amazon Cognito 콘솔에서 위협 보호의 전체 함수 구성을 업데이트하거나 SetRiskConfiguration 요청에 템플릿을 제출하세요.

고급 보안 템플릿 자리 표시자

설명	토큰
IP 주소	`{ip-address}`
City	`{city}`
국가	`{country}`
로그인 시간	`{login-time}`
디바이스 이름	`{device-name}`
원클릭 링크가 유효합니다.	`{one-click-link-valid}`
원클릭 링크가 유효하지 않습니다.	`{one-click-link-invalid}`
이벤트 ID	`{event-id}`
피드백 토큰	`{feedback-token}`

이메일 및 SMS MFA 메시지 사용자 지정

다중 인증(MFA)을 위한 SMS 및 이메일 메시지를 사용자 지정하려면 Amazon Cognito 사용자 풀 콘솔의 메시지 템플릿 메뉴에서 MFA 메시지를 편집합니다.

> **Important**
>
> 사용자 정의 메시지에는 `{####}` 자리 표시자가 포함되어야 합니다. 이 자리 표시자는 메시지를 보내기 전에 인증 코드로 대체됩니다.

Amazon Cognito는 인증 코드를 포함한 SMS 메시지의 최대 길이를 UTF-8 문자 140자로 설정합니다.

SMS 확인 메시지 사용자 정의

전화번호 확인을 위해 SMS 메시지를 사용자 지정하려면 사용자 풀의 메시지 템플릿 메뉴에서 확인 메시지 템플릿을 편집합니다.

> **Important**
>
> 사용자 정의 메시지에는 `{####}` 자리 표시자가 포함되어야 합니다. 이 자리 표시자는 메시지를 보내기 전에 확인 코드로 대체됩니다.

메시지의 최대 길이는 확인 코드를 포함하여 140자(UTF-8)입니다.

이메일 확인 메시지 사용자 정의

Amazon Cognito를 사용하여 사용자 풀에 있는 사용자의 이메일 주소를 확인하려면 사용자가 선택할 수 있는 링크가 포함된 이메일 메시지를 사용자에 보내거나 사용자가 입력할 수 있는 코드를 사용자에게 보냅니다.

이메일 주소 인증 메시지의 이메일 제목 및 메시지 콘텐츠를 사용자 지정하려면 사용자 풀의 메시지 템플릿 메뉴에서 확인 메시지 템플릿을 편집합니다. 확인 메시지 템플릿을 편집할 때 인증 유형의 코드 또는 링크를 선택할 수 있습니다.

확인 유형으로 코드를 선택한 경우 사용자 지정 메시지에 `{####}` 자리 표시자가 포함되어야 합니다. 메시지를 전송할 때 이 자리 표시자는 확인 코드로 대체됩니다.

확인 유형으로 링크를 선택한 경우 사용자 지정 메시지에 `{##Verify Your Email##}` 형식의 자리 표시자가 포함되어야 합니다. 예를 들어 `{##Click here##}` 같이 자리 표시자 문자 사이의 텍스트 문자열을 변경할 수 있습니다. 이 자리 표시자는 이메일 확인(Verify Your Email)이라는 확인 링크로 대체됩니다.

이메일 확인 메시지에 대한 링크를 클릭하면 다음 예와 같은 URL로 이동합니다.

```
https://<your user pool domain>/confirmUser/?
client_id=abcdefg12345678&user_name=emailtest&confirmation_code=123456
```

메시지의 최대 길이는 확인 코드(있는 경우)를 포함하여 20,000자(UTF-8)입니다. 이 메시지에 HTML 태그를 사용하여 내용을 포맷할 수 있습니다.

사용자 초대 메시지 사용자 정의

메시지 템플릿 메뉴의 초대 메시지 템플릿을 편집하여 Amazon Cognito가 SMS 또는 이메일 메시지를 통해 새 사용자에게 보내는 사용자 초대 메시지를 사용자 정의할 수 있습니다.

> **Important**
>
> 사용자 정의 메시지에는 `{username}` 및 `{####}` 자리 표시자가 포함되어야 합니다. Amazon Cognito는 초대 메시지를 전송할 때 이러한 자리 표시자를 사용자의 사용자 이름과 암호로 바꿉니다.

SMS 메시지의 최대 길이는 확인 코드를 포함하여 140자(UTF-8)입니다. 이메일 메시지의 최대 길이는 확인 코드를 포함하여 20,000자(UTF-8)입니다. 이메일 메시지에 HTML 태그를 사용하여 내용을 포맷할 수 있습니다.

이메일 주소 사용자 정의

기본값으로 Amazon Cognito는 사용자 풀의 사용자에게 no-reply@verificationemail.com 주소에서 이메일 메시지를 전송합니다. no-reply@verificationemail.com 대신 사용자 정의 FROM 및 REPLY-TO 이메일 주소를 지정할 수 있습니다.

FROM 및 REPLY-TO 이메일 주소를 사용자 정의하려면

1. Amazon Cognito 콘솔로 이동하여, [사용자 풀(User Pools)]을 선택합니다.
2. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
3. 인증 방법 메뉴를 선택합니다. [이메일(Email)]에서 [편집(Edit)]을 선택합니다.
4. SES 리전(SES Region)을 선택합니다.
5. 선택한 SES 리전(SES Region)의 Amazon SES로 확인한 이메일 주소 목록에서 발신 이메일 주소(FROM email address)를 선택합니다. 확인된 도메인의 이메일 주소를 사용하려면 AWS Command Line Interface 또는 AWS API에서 이메일 설정을 구성합니다. 자세한 내용은 Amazon Simple Email Service 개발자 가이드에서 Amazon SES에서 이메일 주소 및 도메인 확인을 참조하세요.
6. 선택한 [SES 리전(SES Region)]의 구성 집합 목록에서 [구성 집합(Configuration set)]을 선택합니다.
7. 이메일 메시지의 친숙한 [FROM 발신자 이름(FROM sender name)]을 `John Stiles <johnstiles@example.com>` 포맷으로 입력합니다.

8. 회신 이메일 주소를 사용자 지정하려면 회신 이메일 주소 필드에 유효한 이메일 주소를 입력합니다.

Amazon Cognito가 사용자를 대신하여 (사용자 정의 FROM 이메일 주소에서) Amazon SES 이메일을 전송하도록 권한 부여

기본 주소 대신 사용자 정의 FROM 이메일 주소에서 이메일을 전송하도록 Amazon Cognito를 구성할 수 있습니다. 사용자 정의 주소를 사용하려면 Amazon SES 확인 자격 증명에서 이메일 메시지를 전송할 수 있는 권한을 Amazon Cognito에 부여해야 합니다. 대부분의 경우에 전송 권한 부여 정책을 생성하여 권한을 부여할 수 있습니다. 자세한 내용은 Amazon Simple Email Service 개발자 가이드에서 Amazon SES에서 전송 권한 부여 사용을 참조하세요.

이메일 메시지에 Amazon SES를 사용하도록 사용자 풀을 구성하면 Amazon Cognito는 계정에 `AWSServiceRoleForAmazonCognitoIdpEmailService` 역할을 생성하여 Amazon SES에 대한 액세스 권한을 부여합니다. `AWSServiceRoleForAmazonCognitoIdpEmailService` 서비스 연결 역할을 사용하는 경우에는 전송 권한 부여 정책이 필요하지 않습니다. 사용자 풀의 기본 이메일 기능 및 FROM 주소로 확인된 Amazon SES 자격 증명을 둘 다 사용하는 경우에만 전송 권한 부여 정책을 추가해야 합니다.

Amazon Cognito가 생성하는 서비스 연결 역할에 대한 자세한 내용은 Amazon Cognito에 서비스 연결 역할 사용 섹션을 참조하세요.

다음 전송 권한 부여 정책 예제는 Amazon Cognito에 Amazon SES 확인 자격 증명을 사용할 수 있는 제한적인 권한을 부여합니다. Amazon Cognito는 `aws:SourceArn` 조건의 사용자 풀과 `aws:SourceAccount` 조건의 계정을 둘 다 대신하는 경우에만 이메일 메시지를 전송할 수 있습니다. 자세한 내용은 Amazon Simple Email Service 개발자 가이드에서 Amazon SES 전송 권한 부여 정책 예제를 참조하세요.

> **Note**
>
> 이 예제에서 "Sid" 값은 명령문을 고유하게 식별하는 임의 문자열입니다. 정책 구문에 대한 자세한 내용은 Amazon Simple Email Service 개발자 가이드에서 Amazon SES 전송 권한 부여 정책을 참조하세요.

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "stmnt1234567891234",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "email.cognito-idp.amazonaws.com"
                ]
            },
            "Action": [
                "SES:SendEmail",
                "SES:SendRawEmail"
            ],
            "Resource": "arn:aws:ses:us-
east-1:111122223333:identity/support@example.com",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:cognito-idp:us-
east-1:111122223333:userpool/us-east-1_EXAMPLE"
                }
            }
        }
    ]
}
```

Amazon Cognito 콘솔은 드롭다운 메뉴에서 Amazon SES 자격 증명을 선택할 때 유사한 정책을 추가합니다. CLI 또는 API를 사용하여 사용자 풀을 구성하는 경우 이전 예제와 같이 구성된 정책을 Amazon SES 자격 증명에 연결해야 합니다.

관리자로 사용자 계정 생성

사용자 풀은 인터넷에 있는 모든 사용자가 애플리케이션의 사용자 프로필에 등록할 수 있는 고객 ID 및 액세스 관리(CIAM) 사용자 디렉터리일 뿐만 아니라, 사용자 프로필을 등록할 수 있는 곳이기도 합니다. 셀프 서비스 가입을 비활성화할 수 있습니다. 이미 고객을 알고 있으며 미리 승인된 고객만 허용하고 싶을 수도 있습니다. 프라이빗 SAML 2.0 또는 OIDC ID 제공업체를 사용하여, 사용자를 가져오거나, 가입 시 사용자를 선별하거나, 관리 API 작업을 사용하여 사용자를 생성하여 애플리케이션에 수동 인증 가드레일을 설정할 수 있습니다. 사용자를 관리적으로 생성하는 워크플로는 프로그래밍 방식으

로 사용자를 다른 시스템에 등록한 후 프로비저닝하거나, Amazon Cognito 콘솔에서 사례별 또는 테스트 기반으로 할 수 있습니다.

사용자를 관리자로 생성할 때 Amazon Cognito는 사용자를 위한 임시 암호를 설정하고 환영 메시지 또는 초대 메시지를 보냅니다. 초대 메시지의 링크를 따라가 처음으로 로그인하고 암호를 설정하고 계정을 확인할 수 있습니다. 다음 페이지에서는 새 사용자를 생성하고 환영 메시지를 구성하는 방법을 설명합니다. 사용자 풀 API 및 AWS SDK 또는 CDK를 사용한 사용자 생성에 대한 자세한 내용은 AdminCreateUser를 참조하세요.

> 임시 암호가 있는 사용자는 FORCE_CHANGE_PASSWORD 상태에서 시작합니다.
>
> 임시 암호로 사용자를 생성하면 Amazon Cognito는 사용자를 `FORCE_CHANGE_PASSWORD` 상태로 전환합니다. 이 상태의 사용자는 `NEW_PASSWORD_REQUIRED` 챌린지에 대한 응답으로 영구 암호를 설정할 때까지 로그인할 수 없습니다.

관리자가 생성한 사용자를 처음 로그인하지 않고 `CONFIRMED` 상태로 이동하려면가 로 `Permanent` 설정된 AdminSetUserPassword API 작업을 호출합니다`True`. 이 요청은 영구 암호를 설정하고 사용자를 확인합니다.

사용자 풀을 생성한 후 AWS Command Line Interface 또는 Amazon Cognito API뿐만 아니라 AWS Management Console를 사용하여 사용자를 생성할 수 있습니다. 사용자 풀에서 새 사용자에 대한 프로파일을 생성하고, SMS 또는 이메일을 통해 사용자에게 가입 지침이 포함된 환영 메시지를 전송할 수 있습니다.

다음은 관리자가 사용자 풀에서 사용자를 관리하는 방법의 몇 가지 예입니다.

- Amazon Cognito 콘솔 또는 `AdminCreateUser` API 작업을 사용하여 새 사용자 프로필을 생성합니다.
- 사용자 풀과 앱 클라이언트에서 사용자 이름 및 암호, 암호 없음, 패스키 및 사용자 지정 인증 흐름을 사용할 수 있습니다.
- 사용자 속성 값을 설정합니다.
- 사용자 지정 속성을 생성합니다.
- `AdminCreateUser` API 요청에서 변경할 수 없는 사용자 지정 속성의 값을 설정합니다. 이 기능은 Amazon Cognito 콘솔에서 사용할 수 없습니다.
- 임시 암호를 지정하거나, 암호 없이 사용자를 생성하거나, Amazon Cognito가 암호를 자동으로 생성하도록 허용합니다.

- 새 사용자를 생성하고 계정을 자동으로 확인하거나, 이메일 주소를 확인하거나, 전화번호를 확인합니다.
- 사용자 지정 메시지, 사용자 지정 SMS 발신자 및 사용자 지정 이메일 발신자와 같은 AWS Management Console 또는 Lambda 트리거를 통해 새 사용자의 사용자 지정 SMS 및 이메일 초대 메시지를 지정합니다. ???
- 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.
- 사용자가 생성될 때 초대 메시지의 전송을 제한합니다.
- 새로운 사용자 계정에 대해 최대 90일의 만료 기간을 지정합니다.
- 사용자가 직접 가입하거나 관리자가 새 사용자만 추가하도록 합니다.

관리자는 서버 측 애플리케이션의 AWS 자격 증명으로 사용자를 로그인할 수도 있습니다. 자세한 내용은 API 및 SDK 인증을 위한 권한 부여 모델 섹션을 참조하세요.

사용자 인증 흐름 및 사용자 생성

사용자의 관리 생성에는 사용자 풀의 구성에 따라 다른 옵션이 있습니다. 인증 흐름 또는 사용자가 로그인 및 MFA에 사용할 수 있는 메서드는 사용자를 생성하는 방법과 사용자에게 보내는 메시지를 변경할 수 있습니다. 다음은 사용자 풀에서 사용할 수 있는 몇 가지 인증 흐름입니다.

- 사용자 이름 및 암호
- 패스키
- 타사 IdP로 로그인
- 이메일 및 SMS 일회용 암호(OTP)를 사용한 암호 없음
- 이메일, SMS 및 인증자 앱 OTP 사용한 다중 인증
- Lambda 트리거를 사용한 사용자 지정 인증

이러한 로그인 요소를 구성하는 방법에 대한 자세한 내용은 Amazon Cognito 사용자 풀로 인증 섹션을 참조하세요.

암호 없이 사용자 생성

사용자 풀에 암호 없는 로그인을 활성화한 경우 암호 없이 사용자를 생성할 수 있습니다. 암호 없이 사용자를 생성하려면 사용 가능한 암호 없는 로그인 요소의 속성 값을 제공해야 합니다. 예를 들어 사용

자 풀에서 이메일 OTP 암호 없는 로그인을 사용할 수 있는 경우 암호가 없고 이메일 주소 속성이 있는 사용자를 생성할 수 있습니다. 새 사용자가 사용할 수 있는 유일한 인증 흐름에 암호 또는 사용자 이름 암호와 같은 암호가 필요한 경우 각 새 사용자에 대한 임시 암호를 생성하거나 생성해야 합니다.

암호 없이 새 사용자를 생성하려면

- Amazon Cognito 콘솔에서 암호 설정 안 함을 선택합니다.
- `AdminCreateUser` API 요청의 `TemporaryPassword` 파라미터를 생략하거나 비워 둡니다.

암호가 없는 사용자는 자동으로 확인됩니다.

일반적으로 신규 사용자는 임시 암호를 받고 생성할 때 `FORCE_CHANGE_PASSWORD` 상태로 전환됩니다. 암호 없이 사용자를 생성하면 즉시 `CONFIRMED` 상태가 됩니다. `CONFIRMED` 상태의 이러한 사용자에게는 확인 코드를 다시 보낼 수 없습니다.

암호가 없는 사용자의 경우 초대 메시지가 변경됩니다.

기본적으로 Amazon Cognito는 새 사용자에게 `Your username is {userName} and your password is {####}.` 암호 없이 사용자를 생성할 때 `Your username is {userName}.` 사용자에 대한 암호 설정 여부를 반영하도록 초대 메시지를 사용자 지정하라고 표시하는 초대 메시지를 보냅니다. `{####}` 암호 없는 인증 모델에서 암호 변수를 생략합니다.

암호 없는 요소를 사용할 수 있는 경우 암호를 자동 생성할 수 없습니다.

이메일 또는 전화 OTP 암호 없는 로그인을 지원하도록 사용자 풀을 구성한 경우 암호를 자동으로 생성할 수 없습니다. 암호를 갖게 될 각 사용자에 대해 프로필을 생성할 때 임시 암호를 설정해야 합니다.

암호 없는 사용자는 필요한 모든 속성에 대한 값을 가져야 합니다.

암호 없이 사용자를 생성하면 사용자가 사용자 풀에서 필수로 표시한 모든 속성에 대한 값을 제공하는 경우에만 요청이 성공합니다. 이는 OTP 전송에 필요한 전화번호 및 이메일 속성뿐만 아니라 필요한 모든 속성에 적용됩니다.

나중에 필수 속성 값을 제공할 사용자 생성

사용자 풀에 속성이 필요하지만 애플리케이션에서 사용자 상호 작용 중에 사용자를 관리 방식으로 생성한 후 해당 속성을 수집해야 할 수 있습니다. 관리자는 임시 암호로 사용자를 생성할 때 필요한 속성의 값을 생략할 수 있습니다. 암호 없는 사용자의 필수 속성 값은 생략할 수 없습니다.

필수 속성 및 임시 암호 값이 누락된 사용자는 처음 로그인할 때 NEW_PASSWORD_REQUIRED 문제를 받습니다. 그런 다음 `requiredAttributes` 파라미터에서 누락된 필수 속성에 대한 값을

제공할 수 있습니다. 필요한 모든 속성을 변경할 수 있는 경우에만 암호와 필수 속성 없이 사용자를 생성할 수 있습니다. 사용자는 로그인하는 앱 클라이언트에서 필수 속성을 쓸 수 있는 경우에만 NEW_PASSWORD_REQUIRED 문제 및 필수 속성 값으로 로그인을 완료할 수 있습니다.

관리자가 생성한 사용자의 영구 암호를 설정하면 상태가 CONFIRMED로 변경되고 사용자 풀은 처음 로그인할 때 새 암호 또는 필수 속성을 입력하라는 메시지를 표시하지 않습니다.

에서 새 사용자 생성 AWS Management Console

Amazon Cognito 콘솔을 사용하여 사용자 암호 요구 사항을 설정하고, 사용자에게 전송되는 초대 및 확인 메시지를 구성하고, 새 사용자를 추가할 수 있습니다.

암호 정책 설정 및 자체 등록 사용

최소 암호 복잡성에 대한 설정과 사용자가 퍼블릭 API를 사용하여 사용자 풀에 가입할 수 있는지 여부를 구성할 수 있습니다.

암호 정책 구성

1. Amazon Cognito 콘솔로 이동하여, [사용자 풀(User Pools)]을 선택합니다.
2. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
3. 인증 방법 메뉴를 선택하고 암호 정책을 찾습니다. 편집을 선택합니다.
4. [암호 정책 모드(Password policy mode)]로 [사용자 정의(Custom)]를 선택합니다.
5. [암호 최소 길이(Password minimum length)]를 선택합니다. 암호 길이 요구 사항에 대한 제한은 사용자 풀 리소스 할당량을 참조하세요.
6. [암호 복잡성(Password complexity)] 요구 사항을 선택합니다.
7. 관리자가 설정한 암호가 유효한 기간을 선택합니다.
8. 변경 사항 저장(Save changes)을 선택합니다.

셀프 서비스 가입 허용

1. Amazon Cognito 콘솔로 이동하여, [사용자 풀(User Pools)]을 선택합니다.
2. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
3. 가입 메뉴를 선택하고 셀프 서비스 가입을 찾습니다. [편집(Edit)]을 선택합니다.
4. 자체 등록 사용(Enable self-registration) 여부를 선택합니다. 자체 등록은 일반적으로 클라이언트 보안 암호 또는 AWS Identity and Access Management (IAM) API 자격 증명을 배포하지 않고 사용자 풀에 새 사용자를 등록해야 하는 퍼블릭 앱 클라이언트에서 사용됩니다.

> **자체 등록 사용 중지**
>
> 자체 등록을 사용하지 않는 경우 IAM API 자격 증명을 사용하는 관리 API 작업 또는 페더레이션 공급자를 사용한 로그인을 통해 새 사용자를 생성해야 합니다.

5. 변경 사항 저장(Save changes)을 선택합니다.

이메일 및 SMS 메시지 사용자 정의

사용자 메시지 사용자 정의

사용자를 로그인하도록 초대하거나, 사용자가 사용자 계정에 가입하거나, 사용자가 로그인하고 멀티 팩터 인증(MFA) 메시지가 표시될 때 Amazon Cognito가 사용자에게 보내는 메시지를 사용자 정의할 수 있습니다.

> **Note**
>
> [초대 메시지(Invitation message)]는 사용자 풀에 사용자를 생성하고 로그인하도록 초대할 때 전송됩니다. Amazon Cognito는 사용자의 이메일 주소나 전화 번호로 초기 로그인 정보를 전송합니다.
> [확인 메시지(Verification message)]는 사용자가 사용자 풀의 사용자 계정에 가입할 때 전송됩니다. Amazon Cognito가 사용자에게 코드를 전송합니다. 사용자는 Amazon Cognito에 코드를 제공할 때 연락처 정보를 검증하고 로그인 계정을 확인합니다. 확인 코드는 24시간 동안 유효합니다.
> [MFA 메시지(MFA message)]는 사용자 풀에서 SMS MFA를 사용하도록 설정하고, SMS MFA를 구성한 사용자가 로그인하고 MFA 메시지가 표시될 때 전송됩니다.

1. Amazon Cognito 콘솔로 이동하여, [사용자 풀(User Pools)]을 선택합니다.
2. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
3. 메시지 템플릿 메뉴를 선택하고 확인 메시지, 초대 메시지 또는 MFA 메시지를 선택한 다음 편집을 선택합니다.
4. 선택한 메시지 유형에 맞게 메시지를 사용자 정의합니다.

> **Note**
>
> 메시지를 사용자 정의할 때 메시지 템플릿의 모든 변수를 포함해야 합니다. 변수(예: {####})를 포함하지 않으면 사용자에게 메시지 작업을 완료하기에 충분한 정보가 없습니다.
> 자세한 내용은 메시지 템플릿을 참조하세요.

5. a. [확인 메시지(Verification messages)]

   i. [이메일(Email)] 메시지에 대한 [확인 유형(Verification type)]을 선택합니다. [코드(Code)] 확인은 사용자가 입력해야 하는 숫자 코드를 전송합니다. [링크(Link)] 확인은 사용자가 연락처 정보를 확인하기 위해 클릭할 수 있는 링크를 전송합니다. [링크(Link)] 메시지에 대한 변수의 텍스트는 하이퍼링크 텍스트로 표시됩니다. 예를 들어 {##Click here##} 변수를 사용하는 메시지 템플릿은 이메일 메시지에 Click here로 표시됩니다.

   ii. [이메일(Email)] 메시지의 [이메일 제목(Email subject)]을 입력합니다.

   iii. [이메일(Email)] 메시지용 사용자 정의 [이메일 메시지(Email message)] 템플릿을 입력합니다. HTML을 사용하여 이 템플릿을 사용자 정의할 수 있습니다.

   iv. [SMS] 메시지용 사용자 정의 [SMS 메시지(SMS message)] 템플릿을 입력합니다.

   v. 변경 사항 저장(Save changes)을 선택합니다.

   b. [초대 메시지(Invitation messages)]

   i. [이메일(Email)] 메시지의 [이메일 제목(Email subject)]을 입력합니다.

   ii. [이메일(Email)] 메시지용 사용자 정의 [이메일 메시지(Email message)] 템플릿을 입력합니다. HTML을 사용하여 이 템플릿을 사용자 정의할 수 있습니다.

   iii. [SMS] 메시지용 사용자 정의 [SMS 메시지(SMS message)] 템플릿을 입력합니다.

   iv. 변경 사항 저장(Save changes)을 선택합니다.

   c. [MFA 메시지(MFA messages)]

   i. [SMS] 메시지용 사용자 정의 [SMS 메시지(SMS message)] 템플릿을 입력합니다.

   ii. 변경 사항 저장(Save changes)을 선택합니다.

사용자 생성

사용자 생성

Amazon Cognito 콘솔에서 사용자 풀의 새 사용자를 생성할 수 있습니다. 일반적으로 사용자는 암호를 설정한 후에 로그인할 수 있습니다. 이메일 주소를 사용하여 로그인하려면 사용자가 `email` 속성을 확인해야 합니다. 전화 번호를 사용하여 로그인하려면 사용자가 `phone_number` 속성을 확인해야 합니다. 관리자로서 계정을 확인하려면 AWS CLI 또는 API를 사용하거나 페더레이션 자격 증명 공급자를 사용하여 사용자 프로필을 생성할 수도 있습니다. 자세한 내용은 Amazon Cognito API 참조를 참조하세요.

1. Amazon Cognito 콘솔로 이동하여, [사용자 풀(User Pools)]을 선택합니다.
2. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
3. 사용자 메뉴를 선택한 다음 사용자 생성을 선택합니다.
4. 사용자 풀에 대한 암호 요구 사항, 사용 가능한 계정 복구 방법 및 별칭 속성에 대한 지침은 사용자 풀 로그인 및 보안 요구 사항(User pool sign-in and security requirements)을 검토합니다.
5. 초청 메시지(Invitation message)를 보내는 방식을 선택합니다. SMS 메시지와 이메일 메시지 중 하나 또는 둘 다를 선택합니다. 초대 메시지를 표시하지 않으려면 초대를 보내지 않음을 선택합니다.

   > **Note**
   >
   > 초청 메시지를 보내려면 먼저 사용자 풀의 인증 방법 메뉴에서 Amazon Simple Notification Service 및 Amazon Simple Email Service를 사용하여 발신자 및 AWS 리전 을 구성합니다. 수신자 메시지 및 데이터 요금이 적용됩니다. Amazon SES에서 이메일 메시지를 별도로 청구하며, Amazon SNS에서 SMS 메시지를 별도로 청구합니다.

6. 새 사용자의 사용자 이름(Username)을 선택합니다.
7. 사용자에 대해 암호 생성(Create a password)을 직접 수행할지 아니면 Amazon Cognito에서 암호 생성(Generate a password)을 수행하도록 할지를 선택합니다. 사용자 풀에서 암호 없는 로그인을 사용할 수 있는 경우 암호를 생성하는 옵션을 사용할 수 없습니다. 임시 암호는 사용자 풀 암호 정책을 준수해야 합니다.
8. 생성을 선택합니다.
9. 사용자 메뉴를 선택하고 사용자에 대한 사용자 이름 항목을 선택합니다. 사용자 속성(User attributes) 및 그룹 멤버십(Group memberships)을 추가하고 편집합니다. 사용자 이벤트 기록(User event history)을 검토합니다.

사용자 풀에 그룹 추가

Amazon Cognito 사용자 풀에서 그룹에 대한 지원을 통해 그룹을 생성하고 관리하고, 사용자를 그룹에 추가하고, 그룹에서 사용자를 제거할 수 있습니다. 그룹을 통해 사용자 모음을 생성하여 권한을 관리하거나 다른 유형의 사용자를 표시합니다. 그룹에 AWS Identity and Access Management (IAM) 역할을 할당하여 그룹 구성원에 대한 권한을 정의할 수 있습니다.

그룹을 사용하여 사용자 풀에서 사용자 모음을 생성할 수 있습니다. 이 작업은 해당 사용자에 대한 권한을 설정하여 수행되는 경우가 많습니다. 예를 들어, 웹 사이트 및 앱의 독자, 기고자 및 편집자에 대해 별도의 사용자 그룹을 생성할 수 있습니다. 그룹과 연관된 IAM 역할을 사용하면 Amazon S3에 기고자만 콘텐츠를 넣을 수 있으며, 편집자만 Amazon API Gateway의 API를 통해 콘텐츠를 게시할 수 있도록 다른 그룹에 대해 서로 다른 권한을 설정할 수도 있습니다.

Amazon Cognito는 사용자 풀에 추가하는 각 OIDC, SAMI 및 소셜 ID 제공업체(idP)에 대한 사용자 그룹을 생성합니다. 그룹 이름은 `us-east-1_EXAMPLE_MYSS0` 또는 `us-east-1_EXAMPLE_Google`과 같은 `[user pool ID]_[IdP name]` 형식입니다. 자동으로 생성된 각 고유 IdP 사용자 프로필이이 그룹에 자동으로 추가됩니다. 연결된 사용자는 이 그룹에 자동으로 추가되지 않지만 별도의 프로세스에서 그룹에 프로필을 추가할 수 있습니다.

AWS Management Console, APIs 및 CLI에서 사용자 풀의 그룹을 생성하고 관리할 수 있습니다. 개발자(자 AWS 격 증명 사용)는 사용자 풀의 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 요금을 참조하세요.

그룹에 IAM 역할 할당

그룹을 사용하면 IAM 역할을 사용하는 리소스에 대한 권한을 제어할 수 있습니다. IAM 역할에는 신뢰 정책 및 권한 정책이 포함됩니다. 역할 신뢰 정책은 역할을 사용할 수 있는 사용자를 지정합니다. 권한 정책은 그룹 구성원이 액세스할 수 있는 작업과 리소스를 지정합니다. IAM 역할을 생성하는 경우 그룹 사용자가 역할을 수임할 수 있도록 역할 트러스트 정책을 설정합니다. 역할 권한 정책에서 그룹에 부여할 권한을 지정합니다.

Amazon Cognito에서 그룹을 생성할 때 역할의 ARN을 제공하여 IAM 역할을 지정합니다. 그룹 구성원이 Amazon Cognito를 사용하여 로그인하면 자격 증명 풀에서 임시 자격 증명을 받을 수 있습니다. 해당 권한은 연결된 IAM 역할에 따라 결정됩니다.

개별 사용자가 여러 그룹에 있을 수 있습니다. 사용자가 여러 그룹에 있는 경우 개발자는 다음과 같이 IAM 역할을 자동으로 선택하기 위한 옵션을 보유합니다.

- 각 그룹에 우선 순위 값을 할당할 수 있습니다. 우선 순위가 더 높은(낮은) 그룹이 선택되고 연결된 IAM 역할이 적용됩니다.
- 자격 증명 풀을 통해 사용자에 대한 AWS 자격 증명을 요청할 경우 GetCredentialsForIdentity `CustomRoleARN` 파라미터에서 역할 ARN을 지정하여 사용 가능한 역할 중에서 앱을 선택할 수도 있습니다. 지정된 IAM 역할은 사용자에 대해 사용 가능한 역할과 일치해야 합니다.

그룹에 우선 순위 값 할당

사용자는 둘 이상의 그룹에 속할 수 있습니다. 사용자의 액세스 및 ID 토큰에 있는 `cognito:groups` 클레임에는 사용자가 속한 모든 그룹의 목록이 포함되어 있습니다. `cognito:roles` 클레임에는 그룹에 해당하는 역할 목록이 포함되어 있습니다.

사용자가 둘 이상의 그룹에 속할 수 있으므로 각 그룹에 우선 순위가 할당될 수 있습니다. 우선 순위는 사용자 풀에서 사용자가 속한 다른 그룹을 기준으로 이 그룹의 우선 순위를 지정하는, 음수가 아닌 숫자입니다. 0은 가장 높은 우선 순위 값입니다. 우선 순위 값이 낮은 그룹은 우선 순위 값이 높거나 null인 그룹보다 우선합니다. 사용자가 두 개 이상의 그룹에 속하는 경우 우선 순위 값이 가장 낮은 그룹에는 사용자 ID 토큰의 `cognito:preferred_role` 클레임에 IAM 역할이 적용됩니다.

두 개의 그룹에 동일한 우선 순위 값이 있을 수 있습니다. 이러한 경우 두 그룹 모두 우선 적용되지 않습니다. 우선 순위 값이 동일한 두 그룹에 동일한 역할 ARN이 있는 경우 해당 역할은 각 그룹의 사용자에 대한 ID 토큰의 `cognito:preferred_role` 클레임에 사용됩니다. 두 그룹의 역할 ARN이 서로 다른 경우 `cognito:preferred_role` 클레임은 사용자의 ID 토큰에서 설정되지 않습니다.

그룹을 사용하여 Amazon API Gateway로 권한 제어

Amazon API Gateway를 통해 사용자 풀에서 그룹을 사용하여 권한을 제어할 수 있습니다. 사용자가 속한 그룹은 `cognito:groups` 클레임에 있는 사용자 풀의 ID 토큰과 액세스 토큰에 모두 포함됩니다. 요청과 함께 ID 또는 액세스 토큰을 Amazon API Gateway에 제출하고 REST API에 Amazon Cognito 사용자 풀 권한 부여자를 사용할 수 있습니다. 자세한 내용은 API Gateway 개발자 가이드에서 Amazon Cognito 사용자 풀을 권한 부여자로 사용하여 REST API에 대한 액세스 제어를 참조하세요.

사용자 정의 JWT 권한 부여자를 사용하여 Amazon API Gateway HTTP API에 대한 액세스 권한을 부여할 수도 있습니다. 자세한 내용은 API Gateway 개발자 가이드에서 JWT 권한 부여자를 사용하여 HTTP API에 대한 액세스 제어를 참조하세요.

그룹에 대한 제한 사항

사용자 그룹에 다음 제한이 적용됩니다.

- 생성할 수 있는 그룹 수는 Amazon Cognito 서비스 할당량에 따라 제한됩니다.
- 그룹은 중첩될 수 없습니다.
- 그룹에서 사용자를 검색할 수 없습니다.
- 이름으로 그룹을 검색할 수 없지만 그룹을 나열할 수 있습니다.

에서 새 그룹 생성 AWS Management Console

새 그룹을 생성하려면 다음 절차를 따르세요.

새 그룹을 생성하려면

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 자격 AWS 증명을 입력합니다.
2. 사용자 풀(User Pools)을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택합니다.
4. 그룹 메뉴를 선택한 다음 그룹 생성을 선택합니다.
5. [그룹 생성(Create a group)] 페이지의 [그룹 이름(Group name)]에 새 그룹의 친숙한 이름을 입력합니다.
6. 다음 필드를 사용하여 이 그룹에 대한 추가 정보를 선택적으로 제공할 수 있습니다.

   - [설명(Description)] - 새 그룹의 용도에 대한 세부 정보를 입력합니다.
   - [우선 순위(Precedence)] - Amazon Cognito는 우선 순위가 더 낮은 소속 그룹을 기준으로 지정된 사용자의 모든 그룹 권한을 평가하고 적용합니다. 우선 순위가 낮은 그룹이 선택되고 연결된 IAM 역할이 적용됩니다. 자세한 내용은 그룹에 우선 순위 값 할당 섹션을 참조하세요.
   - [IAM 역할(IAM role)] - 리소스에 대한 권한을 제어해야 하는 경우 그룹에 IAM 역할을 할당할 수 있습니다. 사용자 풀을 자격 증명 풀과 통합한 경우 IAM 역할 설정은 자격 증명 풀이 토큰에서 역할을 선택하도록 구성된 경우 사용자의 ID 토큰에 할당될 역할을 결정합니다. 자세한 내용은 그룹에 IAM 역할 할당 섹션을 참조하세요.
   - [이 그룹에 사용자 추가(Add users to this group)] - 그룹이 생성된 후 기존 사용자를 이 그룹의 구성원으로 추가합니다.
7. [생성(Create)]을 선택하여 확인합니다.

사용자 계정 관리 및 검색

사용자 풀에는 수백만 명의 사용자가 포함될 수 있습니다. 이 크기의 데이터세트를 사용하는 것은 관리자에게 어려운 일입니다. Amazon Cognito에는 사용자 프로필을 찾고 수정할 수 있는 도구가 있습니다.

사용자를 찾는 가장 좋은 방법은 Amazon Cognito 콘솔의 사용자 메뉴와 ListUsers를 사용하는 것입니다. 사용자에 대한 정보를 검색하는 방법 중에서 AdminGetUser와 같이 비용에 영향이 없는 옵션은 다음과 같습니다.

가이드의 이 섹션에는 사용자 풀에서 사용자 프로필을 찾고 업데이트하는 방법에 대한 정보가 나와 있습니다.

사용자 속성 보기

Amazon Cognito 콘솔에서 사용자 속성을 보려면 다음 절차를 따르세요.

사용자 속성을 보려면

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 자격 AWS 증명을 입력합니다.
2. 사용자 풀(User Pools)을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택합니다.
4. 사용자 메뉴를 선택하고 목록에서 사용자를 선택합니다.
5. 사용자 세부 정보 페이지의 [사용자 속성(User attributes)]에서 사용자와 연결된 속성을 볼 수 있습니다.

사용자 암호 재설정

Amazon Cognito 콘솔에서 사용자 암호를 재설정하려면 다음 절차를 따르세요.

사용자 암호를 재설정하려면

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 자격 AWS 증명을 입력합니다.
2. 사용자 풀(User Pools)을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택합니다.
4. 사용자 메뉴를 선택하고 목록에서 사용자를 선택합니다.
5. 사용자 세부 정보 페이지에서 [작업(Actions)], [암호 재설정(Reset password)]을 차례로 선택합니다.
6. [암호 재설정(Reset password)] 대화 상자에서 정보를 검토한 다음, 준비가 되면 [재설정(Reset)]을 선택합니다.

   이 작업으로 인해 사용자에게 확인 코드가 즉시 전송되며 사용자 상태가 `RESET_REQUIRED`로 변경되어 사용자의 현재 암호를 사용할 수 없게 됩니다. [암호 재설정(Reset password)] 코드는 1시간 동안 유효합니다.

사용자 계정 활성화, 비활성화 및 삭제

사용하지 않는 사용자 프로필을 삭제하거나 일시적으로 액세스를 차단하려면 비활성화할 수 있습니다. 사용자는 자신의 계정을 삭제할 수 있지만 사용자 풀 관리자만 사용자 계정을 활성화 및 비활성화할 수 있습니다.

삭제의 영향

사용자는 삭제된 사용자 계정으로 로그인할 수 없으며 액세스 권한을 다시 얻으려면 다시 가입하거나 생성해야 합니다.

계정 비활성화의 영향

사용자 계정을 비활성화하면 Amazon Cognito는 인증된 모든 세션을 자동으로 무효화하고, 로그인을 위해 사용자 계정을 비활성화하고, 액세스 및 새로 고침 토큰을 취소합니다. Amazon Cognito는 사용자가 비활성화한 계정에 로그인하려고 할 때 `User is not enabled` 메시지와 함께 `invalid_request` 오류를 반환합니다. 이 동작은 앱 클라이언트에 대한 사용자 존재 공개 설정에 따라 변경되지 않습니다. 로컬 사용자 계정과 페더레이션 사용자 계정의 로컬 프로필을 비활성화할 수 있습니다. 사용자가 관리형 로그인 또는 클래식 호스팅 UI로 로그인하면 계정을 비활성화한 다음 인증된 세션을 유지 관리하는 브라우저 쿠키로 다시 로그인하려고 하면 Amazon Cognito가 사용자를 로그인 페이지로 리디렉션합니다.

계정 활성화의 영향

사용자는 계정을 활성화한 후 즉시 계정에 로그인할 수 있습니다. 사용자 계정은 기본적으로 활성화되어 있습니다. 사용자의 속성과 암호는 계정이 비활성화되기 전과 동일하게 유지됩니다. 사용자 계정을 비활성화했든 새로 고침 토큰을 별도로 취소했든 상관없이 애플리케이션이 취소한 토큰은 토큰을 소유한 사용자 계정을 활성화한 후에도 유효하지 않습니다.

Delete a user account (console)

사용자 계정을 삭제하려면

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 자격 AWS 증명을 입력합니다.
2. 사용자 풀(User Pools)을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택합니다.
4. 사용자 메뉴를 선택하고 목록에서 사용자의 사용자 이름 옆에 있는 라디오 버튼을 선택합니다.
5. 삭제를 선택합니다.

6. 사용자 액세스 권한 비활성화를 선택합니다.
7. 삭제를 선택합니다.

Delete a user account (API)

사용자는 셀프 서비스 액세스 토큰으로 인증된 DeleteUser API 작업을 사용하여 계정을 삭제할 수 있습니다. 다음은 `DeleteUser` 요청 본문의 예입니다.

```
{
   "AccessToken": "eyJra456defEXAMPLE"
}
```

관리자는 IAM 인증 AdminDeleteUser API 작업을 사용하여 사용자 계정을 삭제할 수 있습니다. 다음은 `AdminDeleteUser` 요청 본문의 예입니다.

```
{
   "Username": "testuser",
   "UserPoolId": "us-west-2_EXAMPLE"
}
```

Disable a user account (console)

사용자 계정을 비활성화하려면

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 자격 AWS 증명을 입력합니다.
2. 사용자 풀(User Pools)을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택합니다.
4. 사용자 메뉴를 선택하고 목록에서 사용자의 사용자 이름을 선택합니다.
5. 사용자 세부 정보 페이지에서 작업, 사용자 액세스 비활성화를 선택합니다.
6. 생성하는 대화 상자에서 비활성화를 선택합니다.

Disable a user account (API)

관리자는 IAM 인증 AdminDisableUser API 작업을 사용하여 사용자 계정을 비활성화할 수 있습니다. 다음은 `AdminDisableUser` 요청 본문의 예입니다.

```
{
```

```
    "Username": "testuser",
    "UserPoolId": "us-west-2_EXAMPLE"
}
```

Enable a user account (console)

사용자 계정을 활성화하려면

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 자격 AWS 증명을 입력합니다.
2. 사용자 풀(User Pools)을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택합니다.
4. 사용자 메뉴를 선택하고 목록에서 사용자의 사용자 이름을 선택합니다.
5. 사용자 세부 정보 페이지에서 작업, 사용자 액세스 활성화를 선택합니다.
6. 생성하는 대화 상자에서 활성화를 선택합니다.

Enable a user account (API)

관리자는 IAM 인증 AdminEnableUser API 작업을 사용하여 사용자 계정을 활성화할 수 있습니다. 다음은 `AdminEnableUser` 요청 본문의 예입니다.

```
{
    "Username": "testuser",
    "UserPoolId": "us-west-2_EXAMPLE"
}
```

사용자 속성 검색

사용자 풀을 이미 생성한 경우 AWS Management Console의 [사용자(Users)] 패널에서 검색할 수 있습니다. Filter 파라미터를 받는 Amazon Cognito ListUsers API를 사용할 수도 있습니다.

다음 표준 속성 중 하나를 검색할 수 있습니다. 사용자 지정 속성은 검색할 수 없습니다.

- username(대/소문자 구분)
- email
- phone_number
- name

- given_name
- family_name
- preferred_username
- cognito:user_status(콘솔에서 상태라고 함)(대/소문자 구분)
- status(콘솔에서 활성이라고 함)(대/소문자 구분)
- sub

> **Note**
>
> 클라이언트 측 필터를 사용하여 사용자를 나열할 수도 있습니다. 서버 측 필터는 1개 이하의 속성을 일치시킵니다. 고급 검색의 경우 AWS Command Line Interface에서 `list-users` 작업의 `--query` 파라미터에 클라이언트 측 필터를 사용합니다. 클라이언트 측 필터를 사용하는 경우 ListUsers는 0명 이상의 사용자를 페이지가 매겨진 목록으로 반환합니다. 결과가 0인 여러 페이지를 연속으로 받을 수 있습니다. Null 페이지 매김 토큰 값을 받을 때까지 반환되는 각 페이지 매김 토큰으로 쿼리를 반복한 다음, 결합된 결과를 검토합니다.
> 서버 측 및 클라이언트 측 필터링에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 출력 필터링을 참조하세요 AWS CLI.

를 사용하여 사용자 검색 AWS Management Console

사용자 풀을 이미 생성한 경우 AWS Management Console의 사용자(Users) 패널에서 검색할 수 있습니다.

AWS Management Console 검색은 항상 접두사("starts with") 검색입니다.

Amazon Cognito 콘솔에서 사용자를 검색하려면

1. Amazon Cognito 콘솔로 이동합니다. 자격 AWS 증명을 입력하라는 메시지가 표시될 수 있습니다.
2. 사용자 풀(User Pools)을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택합니다.
4. 사용자 메뉴를 선택하고 검색 필드에 사용자 이름을 입력합니다. 일부 속성 값은 대/소문자를 구분합니다(예: [사용자 이름(User name)]).

   검색 필터를 조정하여 다른 사용자 속성(예: [이메일(Email)], [전화 번호(Phone number)] 또는 [성(Last name)])으로 범위를 좁혀 사용자를 찾을 수도 있습니다.

`ListUsers` API를 사용하여 사용자 검색

앱에서 사용자를 검색하려면 Amazon Cognito ListUsers API를 사용합니다. 이 API는 다음 파라미터를 사용합니다.

- `AttributesToGet`: 문자열의 어레이입니다. 여기에서 각 문자열은 검색 결과에서 각 사용자에 대해 반환될 사용자 속성의 이름입니다. 모든 속성을 검색하려면 `AttributesToGet` 파라미터를 포함하거나 리터럴 문자열 `null`의 값으로 `AttributesToGet`을 요청하지 마세요.
- `Filter`: `"AttributeName Filter-Type "AttributeValue""` 양식의 필터 문자열입니다. 필터 문자열 내의 인용 부호는 백슬래시(\) 문자를 사용하여 이스케이프되어야 합니다. 예를 들어 `"family_name = \"Reddy\""`입니다. 필터 문자열이 비어 있는 경우 `ListUsers`는 사용자 풀에서 모든 사용자를 반환합니다.
  - `AttributeName`: 검색할 속성의 이름입니다. 한 번에 속성 하나만 검색할 수 있습니다.

    > **Note**
    >
    > 표준 속성만 검색할 수 있습니다. 사용자 지정 속성은 검색할 수 없습니다. 이는 인덱싱된 속성만 검색할 수 있으며 사용자 지정 속성은 인덱싱될 수 없기 때문입니다.

  - `Filter-Type`: 정확하게 일치해야 하는 경우 `given_name = "Jon"`과 같이 `=`을 사용하세요. 접두사("다음으로 시작")가 일치해야 하는 경우 `given_name ^= "Jon"`과 같이 `^=`를 사용하세요.
  - `AttributeValue`: 각 사용자에 대해 일치해야 하는 속성 값입니다.
- `Limit`: 반환할 최대 사용자 수입니다.
- `PaginationToken`: 이전 검색에서 더 많은 결과를 가져올 토큰입니다. Amazon Cognito는 1시간 후에 페이지 매김 토큰을 만료합니다.
- `UserPoolId`: 검색을 수행해야 할 사용자 풀에 대한 사용자 풀 ID입니다.

모든 검색은 대/소문자를 구분합니다. 검색 결과는 `AttributeName` 문자열로 명명되는 속성별로 오름차순으로 정렬됩니다.

`ListUsers` API 사용 예제

다음은 모든 사용자를 반환하고 모든 속성을 포함하는 예제입니다.

```
{
```

```
    "AttributesToGet": null,
    "Filter": "",
    "Limit": 10,
    "UserPoolId": "us-east-1_samplepool"
}
```

다음은 전화 번호가 "+1312"로 시작하는 모든 사용자를 반환하고 모든 속성을 포함하는 예제입니다.

```
{
    "AttributesToGet": null,
    "Filter": "phone_number ^= \"+1312\"",
    "Limit": 10,
    "UserPoolId": "us-east-1_samplepool"
}
```

다음은 성이 "Reddy"인 사용자 중 처음 10명의 사용자를 반환하는 예제입니다. 각 사용자에 대해 검색 결과에는 사용자의 지정된 이름, 전화 번호 및 이메일 주소가 포함됩니다. 사용자 풀에 일치하는 사용자가 10명을 초과하는 경우 응답에 페이지 매김 토큰이 포함됩니다.

```
{
    "AttributesToGet": [
        "given_name",
        "phone_number",
        "email"
    ],
    "Filter": "family_name = \"Reddy\"",
    "Limit": 10,
    "UserPoolId": "us-east-1_samplepool"
}
```

이전 예제가 페이지 매김 토큰을 반환하는 경우 다음 예제는 동일한 필터 문자열과 일치하는 다음 10명의 사용자를 반환합니다.

```
{
    "AttributesToGet": [
        "given_name",
        "phone_number",
        "email"
```

```
    ],
    "Filter": "family_name = \"Reddy\"",
    "Limit": 10,
    "PaginationToken": "pagination_token_from_previous_search",
    "UserPoolId": "us-east-1_samplepool"
}
```

암호, 계정 복구 및 암호 정책

페더레이션 사용자를 포함하여 사용자 풀에 로그인하는 모든 사용자는 사용자 프로필에 암호가 할당됩니다. 로컬 사용자와 연결된 사용자는 로그인할 때 암호를 제공해야 합니다. 페더레이션 사용자는 사용자 풀 암호를 사용하지 않지만 해당 ID 제공업체(idP)를 사용하여 로그인합니다. 사용자가 자신의 암호를 재설정하고, 관리자로서 암호를 재설정 또는 변경하고, 암호 복잡성 및 기록에 대한 정책을 설정하도록 허용할 수 있습니다.

Amazon Cognito는 사용자 암호를 일반 텍스트로 저장하지 않습니다. 대신 각 사용자의 암호 해시를 사용자별 솔트와 함께 저장합니다. 이로 인해 사용자 풀의 사용자 프로필에서 기존 암호를 검색할 수 없습니다. 가장 좋은 방법은 일반 텍스트 사용자 암호를 어디에도 저장하지 않는 것입니다. 사용자가 암호를 잊어버린 경우 암호 재설정을 수행합니다.

암호 재설정 및 복구

사용자가 암호를 잊어버렸습니다. 암호를 직접 재설정하거나 관리자가 암호를 재설정하도록 요구할 수 있습니다. Amazon Cognito 사용자 풀에는 두 모델 모두에 대한 옵션이 있습니다. 가이드의 이 부분에서는 사용자 풀 설정과 암호 재설정을 위한 API 작업을 설명합니다.

ForgotPassword API 작업 및 관리형 로그인 옵션 암호 찾기는 사용자에게 코드를 전송하여 올바른 코드를 가지고 있음을 확인하면 ConfirmForgotPassword를 사용하여 새 암호를 설정할 수 있는 기회를 제공합니다. 이는 셀프 서비스 암호 복구 모델입니다.

확인되지 않은 사용자의 복구

이메일 주소 또는 전화번호를 확인한 사용자에게 복구 메시지를 보낼 수 있습니다. 확인된 복구 이메일 또는 전화가 없는 경우 사용자 풀 관리자는 이메일 주소 또는 전화 번호를 확인됨으로 표시할 수 있습니다. Amazon Cognito 콘솔에서 사용자의 사용자 속성을 편집하고 전화번호를 확인됨으로 표시 또는 이메일 주소를 확인됨으로 표시 옆의 확인란을 선택합니다. AdminUpdateUserAttributes 요청에서 `email_verified` 또는 `phone_number_verified`를 true로 설정할 수도 있습니다. 신규 사용자의 경우 ResendConfirmationCode API 작업은 이메일 주소 또는 전화번호로 새 코드를 전송하고 셀프 서비스 확인 및 검증을 완료할 수 있습니다.

관리자로서 암호 재설정

AdminSetUserPassword 및 AdminResetUserPassword API 작업은 관리자가 암호를 재설정하는 방법입니다. `AdminSetUserPassword`는 임시 또는 영구 암호를 설정하고, `AdminResetUserPassword`는 `ForgotPassword`와 동일한 방식으로 사용자에게 암호 재설정 코드를 보냅니다.

암호 재설정 및 복구 구성

Amazon Cognito는 콘솔에서 사용자 풀을 생성할 때 선택하는 필수 속성 및 로그인 옵션에서 계정 복구 옵션을 자동으로 선택합니다. 이러한 기본 설정을 수정할 수 있습니다.

사용자가 선호하는 MFA 메서드는 암호를 복구하는 데 사용할 수 있는 메서드에 영향을 미칩니다. 선호하는 MFA가 이메일 메시지인 사용자는 이메일로 암호 재설정 코드를 받을 수 없습니다. 선호하는 MFA가 SMS 메시지인 사용자는 SMS로 암호 재설정 코드를 받을 수 없습니다.

사용자가 선호하는 암호 재설정 방법을 사용할 수 없는 경우 암호 복구 설정에서 대체 옵션을 제공해야 합니다. 예를 들어 복구 메커니즘에 이메일이 최우선 순위일 수 있고 이메일 MFA가 사용자 풀의 옵션일 수 있습니다. 이 경우 SMS 메시지 계정 복구를 두 번째 옵션으로 추가하거나 관리 API 작업을 사용하여 해당 사용자의 암호를 재설정합니다.

Amazon Cognito는 `InvalidParameterException` 오류 응답과 함께 유효한 복구 방법이 없는 사용자의 암호 재설정 요청에 응답합니다.

> **Note**
>
> 사용자는 동일한 이메일 주소 또는 전화번호로 MFA 및 암호 재설정 코드를 받을 수 없습니다. MFA용 이메일 메시지의 일회용 암호(OTP)를 사용하는 경우 계정 복구에 SMS 메시지를 사용해야 합니다. MFA에 SMS 메시지의 OTP를 사용하는 경우 계정 복구에 이메일 메시지를 사용해야 합니다. MFA가 있는 사용자 풀에서 사용자가 이메일 주소 속성은 있지만 전화번호가 없거나, 전화번호 속성은 있지만 이메일 주소가 없는 경우 셀프 서비스 암호 복구를 완료하지 못할 수 있습니다.
>
> 사용자가 이 구성으로 사용자 풀에서 암호를 재설정할 수 없는 상태를 방지하려면 `email` 및 `phone_number` 속성을 필요에 따라 설정합니다. 또는 사용자가 가입하거나 관리자가 사용자 프로필을 생성할 때 항상 이러한 속성을 수집하고 설정하는 프로세스를 설정할 수 있습니다. 사용자에게 두 속성이 모두 있는 경우 Amazon Cognito는 사용자의 MFA 요소가 아닌 대상으로 암호 재설정 코드를 자동으로 전송합니다.

다음 절차에서는 사용자 풀에서 셀프 서비스 계정 복구를 구성합니다.

Configure self-service password reset (API/SDK)

`AccountRecoverySetting` 파라미터는 사용자가 ForgotPassword API 요청에서 암호를 복구하는 데 사용할 수 있는 방법이나 관리형 로그인에서 암호 찾기를 선택할 때 사용할 수 있는 방법을 설정하는 사용자 풀 파라미터입니다. `ForgotPassword`는 확인된 이메일이나 확인된 전화번호로 복구 코드를 보냅니다. 복구 코드는 1시간 동안 유효합니다. 사용자 풀에 대해 AccountRecoverySetting을 지정할 때 Amazon Cognito는 사용자가 설정한 우선 순위에 따라 코드 전달 대상을 선택합니다.

사용자가 `AccountRecoverySetting`를 정의하고 SMS MFA를 구성한 경우 SMS를 계정 복구 메커니즘으로 사용할 수 없습니다. 이 설정의 우선 순위는 1이 가장 높은 우선 순위로 결정됩니다. Amazon Cognito는 지정된 메서드 중 하나에만 확인을 보냅니다. 다음 예시 `AccountRecoverySetting`에서는 이메일 주소를 계정 복구 코드의 기본 대상으로 설정하여 사용자에게 이메일 주소 속성이 없는 경우 SMS 메시지로 돌아갑니다.

```
"AccountRecoverySetting": {
   "RecoveryMechanisms": [
      {
         "Name": "verified_email",
         "Priority": 1
      },
      {
         "Name": "verified_phone_number",
         "Priority": 2
      }
   ]
}
```

`admin_only` 값은 셀프 서비스 계정 복구를 끕니다. 대신 사용자가 암호 재설정을 위해 관리자에게 문의하도록 요구합니다. `admin_only`는 다른 계정 복구 메커니즘과 함께 사용할 수 없습니다. 다음 e

```
"AccountRecoverySetting": {
   "RecoveryMechanisms": [
      {
         "Name": "admin_only",
         "Priority": 1
      }
   ]
}
```

`AccountRecoverySetting`을 지정하지 않으면 Amazon Cognito는 먼저 확인된 전화번호로 복구 코드를 보내고, 사용자에게 전화번호 속성이 없는 경우 확인된 이메일 주소로 복구 코드를 보냅니다.

`AccountRecoverySetting`에 대한 자세한 내용은 CreateUserPool 및 UpdateUserPool을 참조하세요.

Configure self-service password reset (console)

사용자 풀의 로그인 메뉴에서 계정 복구 및 암호 재설정 옵션을 구성합니다.

사용자 계정 복구를 설정하려면

1. Amazon Cognito 콘솔에 로그인합니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. 로그인 메뉴를 선택합니다. 사용자 계정 복구를 찾아 편집을 선택합니다.
5. 사용자가 자신의 암호를 재설정하도록 허용하려면 셀프 서비스 계정 복구 활성화를 선택합니다.
6. 사용자 풀이 사용자에게 보내는 암호 복구 코드에 대한 전송 방법을 구성합니다. 사용자 계정 복구 메시지의 전송 방법에서 사용 가능한 옵션을 선택합니다. 가장 좋은 방법은 메시지 전송을 위한 보조 방법이 있는 옵션을 선택하는 것입니다. 예를 들어 가능한 경우 이메일, 그렇지 않으면 SMS를 선택하는 것입니다. 보조 전송 방법을 사용하면 Amazon Cognito는 MFA와 다른 암호 재설정 미디어를 사용해야 하는 방식으로 사용자에게 코드를 보낼 수 있습니다.
7. 변경 사항 저장을 선택합니다.

암호 찾기 동작

암호 및 암호 확인 작업의 일부로서 지정된 시간 동안 사용자가 암호 재설정 코드를 요청 또는 입력하는 데 5~20회의 시도가 허용됩니다. 정확한 값은 요청과 관련된 위험 매개 변수에 따라 다릅니다. 이 동작은 변경될 수 있습니다.

사용자 풀 암호 요구 사항 추가

강력하고 복잡한 암호를 설정하는 것이 사용자 풀을 위해 가장 좋습니다. 특히 인터넷에 공개된 애플리케이션에서 약한 암호를 사용하면 사용자의 자격 증명이 암호를 추측하고 데이터에 액세스하려고 시도하는 시스템에 노출될 수 있습니다. 암호가 복잡할수록 추측하기가 더 어렵습니다. Amazon Cognito에는 위협 방지 및 AWS WAF 웹 ACL과 같은 보안 의식이 있는 관리자를 위한 추가 도구가 있지만 암호 정책이 사용자 디렉터리 보안의 핵심 요소입니다.

Amazon Cognito 사용자 풀의 로컬 사용자 암호는 자동으로 만료되지 않습니다. 사용자 암호 재설정의 시간, 날짜 및 메타데이터를 외부 시스템에 기록하는 것이 가장 좋습니다. 암호 수명에 대한 외부 로그를 사용하면 애플리케이션 또는 Lambda 트리거가 사용자의 암호 기간을 조회하고 지정된 기간 후에 재설정을 요구할 수 있습니다.

사용자 풀을 구성하여 보안 표준을 준수하는 최소한의 암호 복잡성을 요구할 수 있습니다. 복잡한 암호의 길이는 최소 8자입니다. 대문자, 숫자 및 특수 문자를 혼합한 것도 포함됩니다.

Essentials 또는 Plus 기능 계층을 사용하면 암호 재사용 정책을 설정할 수도 있습니다. 사용자가 현재 암호 또는 최대 23개의 이전 암호와 일치하는 새 암호로 재설정하지 못하도록 설정하여 최대 총 24개의 암호를 사용할 수 있도록 할 수 있습니다.

> **Note**
>
> Amazon Cognito는 암호 해시가 있는 사용자를 가져올 때 암호 정책을 적용하지 않습니다. 가져오기를 위해 해시를 생성하기 전에 소스 시스템의 암호 강도를 검증합니다.

사용자 풀 암호 정책 설정

1. 사용자 풀을 생성하고 보안 요구 사항 구성 단계로 이동하거나 기존 사용자 풀에 액세스하여 인증 방법 메뉴로 이동합니다.
2. 암호 정책으로 이동합니다.
3. 암호 정책 모드를 선택합니다. Cognito 기본값은 권장 최소 설정으로 사용자 풀을 구성합니다. 사용자 지정 암호 정책을 선택할 수도 있습니다.
4. 암호 최소 길이를 설정합니다. 모든 사용자는 이 값보다 크거나 같은 길이의 암호를 등록하거나 생성해야 합니다. 이 최솟값을 99까지 설정할 수 있지만 사용자는 최대 256자의 암호를 설정할 수 있습니다.
5. 암호 요구 사항에서 암호 복잡성 규칙을 구성하십시오. 각 사용자 암호에 필요한 문자 유형(숫자, 특수 문자, 대문자, 소문자)을 선택합니다.

   암호에 다음 문자 중 하나 이상을 요구할 수 있습니다. Amazon Cognito가 암호에 최소 필수 문자가 포함되어 있는지 확인한 후에는 사용자의 암호에 최대 암호 길이까지 모든 유형의 추가 문자를 포함할 수 있습니다.

   - 기본 라틴 문자의 대문자와 소문자
   - 숫자
   - 다음은 특수 문자입니다.

```
^ $ * . [ ] { } ( ) ? " ! @ # % & / \ , > < ' : ; | _ ~ ` = + -
```

- 비선행 비후행 공백 문자

6. 관리자가 설정한 임시 암호의 만료 기간을 설정합니다. 이 기간이 지나면 Amazon Cognito 콘솔에서 또는 `AdminCreateUser`를 통해 생성한 새 사용자는 로그인하거나 새 암호를 설정할 수 없습니다. 임시 암호로 로그인한 후에는 사용자 계정이 만료되지 않습니다. Amazon Cognito 사용자 풀 API에서 암호 지속 기간을 업데이트하려면 CreateUserPool 또는 UpdateUserPool API 요청에서 TemporaryPasswordValidityDays 값을 설정합니다.
7. 사용 가능한 경우 이전 암호 사용 방지 값을 설정합니다. 이 기능을 사용하려면 사용자 풀에서 Essentials 또는 Plus 기능 계층을 선택합니다. 이 파라미터의 값은 사용자가 암호를 재설정할 때 새 암호와 일치하지 않도록 하는 이전 암호의 수입니다.

만료된 사용자 계정의 액세스 권한을 재설정하려면 다음 중 하나를 수행합니다.

- `MessageAction`가 `RESEND`로 설정된 AdminCreateUser API 요청을 사용하여 새 임시 암호를 보내고 만료 기간을 재설정합니다.
- 사용자 프로필을 삭제하고 새 프로필을 생성합니다.
- AdminResetUserPassword API 요청에서 새 확인 코드를 생성합니다.

사용자 풀로 사용자 가져오기

두 가지 방법으로 기존 사용자 디렉터리 또는 사용자 데이터베이스에서 Amazon Cognito 사용자 풀로 사용자를 가져오거나 마이그레이션할 수 있습니다. Amazon Cognito를 사용하여 처음 로그인할 때 사용자 마이그레이션 Lambda 트리거를 사용하여 사용자를 마이그레이션할 수 있습니다. 이 접근 방식을 사용하면 사용자는 기존 암호를 계속 사용할 수 있으며 사용자 풀로 마이그레이션한 후 암호를 재설정할 필요가 없습니다. 또는 모든 사용자의 사용자 프로파일 속성이 포함된 CSV 파일을 업로드하여 사용자를 대량으로 마이그레이션할 수 있습니다. 다음 섹션에서는 이 두 가지 방법을 설명합니다.

추가 리소스

- 사용자를 Amazon Cognito 사용자 풀로 마이그레이션하는 방법
- AWS re:Inforce 2023 - Amazon Cognito로 마이그레이션

주제

- 사용자 마이그레이션 Lambda 트리거를 사용하여 사용자 가져오기

- CSV 파일에서 사용자 풀로 사용자 가져오기

사용자 마이그레이션 Lambda 트리거를 사용하여 사용자 가져오기

이 접근 방식을 사용하면 사용자가 앱으로 처음 로그인하거나 암호 재설정을 요청할 때 기존 사용자 디렉터리에 있는 사용자를 사용자 풀로 원활하게 마이그레이션할 수 있습니다. 사용자 마이그레이션 Lambda 트리거 함수를 사용자 풀에 추가합니다. 이 함수는 로그인하려고 하는 사용자에 대한 메타데이터를 수신하고 외부 자격 증명 소스에서 사용자 프로필 정보를 반환합니다. 요청 및 응답 파라미터를 포함한 이 Lambda 트리거에 대한 자세한 내용과 코드 예는 사용자 마이그레이션 Lambda 트리거 파라미터 섹션을 참조하세요.

사용자 마이그레이션을 시작하기 전에 AWS 계정에서 사용자 마이그레이션 Lambda 함수를 생성하고, 이 Lambda 함수를 사용자 풀의 사용자 마이그레이션 트리거로 설정합니다. Amazon Cognito 서비스 계정 보안 주체 `cognito-idp.amazonaws.com`만 사용자 자신의 사용자 풀 컨텍스트에서만 Lambda 함수를 호출하도록 허용하는 권한 부여 정책을 Lambda 함수에 추가합니다. 자세한 내용은 AWS Lambda 에 대한 리소스 기반 정책(Lambda 함수 정책) 사용을 참조하세요.

로그인 프로세스

1. 사용자가 앱을 열고 Amazon Cognito 사용자 풀 API 또는 관리형 로그인을 통해 로그인합니다. Amazon Cognito API로 쉽게 로그인하는 방법에 대한 자세한 내용은 웹 및 모바일 앱과 Amazon Cognito 인증 및 권한 부여 통합 섹션을 참조하세요.
2. 앱이 사용자 이름과 암호를 Amazon Cognito로 전송합니다. 앱에 AWS SDK로 빌드한 사용자 지정 로그인 UI가 있는 경우 앱은 `USER_PASSWORD_AUTH` 또는 `ADMIN_USER_PASSWORD_AUTH` 흐름과 함께 InitiateAuth 또는 AdminInitiateAuth를 사용해야 합니다. 앱에서 이러한 흐름 중 하나를 사용하면 SDK가 암호를 서버로 보냅니다.

   > **Note**
   >
   > 사용자 마이그레이션 트리거를 추가하기 전에 앱 클라이언트 설정에서 `USER_PASSWORD_AUTH` 또는 `ADMIN_USER_PASSWORD_AUTH` 흐름을 활성화합니다. 기본 `USER_SRP_AUTH` 흐름 대신 이러한 흐름을 사용해야 합니다. Amazon Cognito는 다른 디렉터리에서 사용자의 인증을 확인할 수 있도록 Lambda 함수에 암호를 보내야 합니다. SRP는 Lambda 함수에서 사용자의 암호를 가립니다.

3. Amazon Cognito는 제출된 사용자 이름이 사용자 풀의 사용자 이름 또는 별칭과 일치하는지 확인합니다. 사용자의 이메일 주소, 전화번호 또는 기본 사용자 이름을 사용자 풀의 별칭으로 설정할 수 있

습니다. 사용자가 없는 경우 Amazon Cognito는 사용자 이름 및 암호를 포함한 파라미터를 사용자 마이그레이션 Lambda 트리거 함수에 전송합니다.

4. 사용자 마이그레이션 Lambda 트리거 함수는 기존 사용자 디렉터리 또는 사용자 데이터베이스를 사용하여 사용자를 확인하거나 인증합니다. 이 함수는 Amazon Cognito가 사용자 풀의 사용자 프로필에 저장하는 사용자 속성을 반환합니다. 제출된 사용자 이름이 별칭 속성과 일치하는 경우에만 `username` 파라미터를 반환할 수 있습니다. 사용자가 기존 암호를 계속 사용하도록 하려는 경우 함수가 Lambda 응답에서 `finalUserStatus` 속성을 `CONFIRMED`로 설정합니다. 앱은 사용자 마이그레이션 Lambda 트리거 파라미터에 표시된 모든 `"response"` 파라미터를 반환해야 합니다.

   > **Important**
   >
   > 사용자 마이그레이션 Lambda 코드에 전체 요청 이벤트 객체를 기록하지 마세요. 이 요청 이벤트 객체에는 사용자의 암호가 포함됩니다. 로그를 폐기하지 않는 경우 암호가 CloudWatch 로그에 나타납니다.

5. Amazon Cognito가 사용자 풀에 사용자 프로파일을 생성하고 앱 클라이언트에 토큰을 반환합니다.
6. 앱이 토큰 수집을 수행하고, 사용자 인증을 수락하고, 요청된 콘텐츠로 진행합니다.

사용자를 마이그레이션한 후 `USER_SRP_AUTH`를 로그인에 사용합니다. SRP(보안 원격 암호) 프로토콜에서 네트워크를 통해 암호를 보내지 않고 마이그레이션 중에 사용하는 `USER_PASSWORD_AUTH` 흐름에서 보안 강화를 제공합니다.

마이그레이션 중에 클라이언트 디바이스 문제, 네트워크 문제 등의 오류가 발생하면 앱이 Amazon Cognito 사용자 풀 API로부터 오류 응답을 수신합니다. 이 경우 Amazon Cognito가 사용자 풀에서 사용자 계정을 생성하거나 생성하지 않을 수도 있습니다. 그러면 사용자가 다시 로그인을 시도해야 합니다. 로그인이 반복해서 실패할 경우 앱에서 암호 찾기 흐름을 사용하여 사용자의 암호 재설정을 시도합니다.

암호 찾기 흐름도 `UserMigration_ForgotPassword` 이벤트 소스를 사용하여 사용자 마이그레이션 Lambda 트리거 암호를 호출합니다. 사용자가 암호 재설정을 요청할 때 암호를 제출하지 않으므로 Amazon Cognito는 Lambda 함수로 전송하는 경우 암호를 포함하지 않습니다. 함수는 기존 사용자 디렉터리에서 사용자를 조회하고 사용자 풀의 사용자 프로필에 추가할 속성을 반환할 수만 있습니다. 함수가 간접 호출을 완료하고 Amazon Cognito에 응답을 반환하면 사용자 풀이 이메일 또는 SMS로 암호 재설정 코드를 보냅니다. 앱에서 사용자에게 확인 코드와 새 암호를 입력하라는 메시지를 표시한 다음 ConfirmForgotPassword API 요청에서 해당 정보를 Amazon Cognito로 전송합니다. 관리형 로그인에서 암호를 잊어버린 경우를 위한 기본 제공 페이지를 사용할 수도 있습니다.

추가 리소스

- 사용자를 Amazon Cognito 사용자 풀로 마이그레이션하는 방법

CSV 파일에서 사용자 풀로 사용자 가져오기

외부 ID 저장소가 있고 새 로컬 사용자를 위해 사용자 풀을 준비할 시간이 있는 경우 쉼표로 구분된 값(CSV) 파일에서 대량 사용자 가져오기는 낮은 비용과 적은 노력으로 Amazon Cognito 사용자 풀로 마이그레이션할 수 있는 옵션일 수 있습니다. CSV 파일 가져오기는 템플릿 파일을 다운로드하고 작성한 다음 가져오기 작업의 사용자 풀로 파일을 전달하는 프로세스입니다. CSV 가져오기를 사용하여 테스트 사용자를 빠르게 생성할 수 있습니다. 또한 외부 ID 저장소에 대한 읽기 API 요청으로 파일을 프로그래밍 방식으로 작성한 다음 세부 정보 및 속성을 구문 분석하여 파일에 쓰기 작업을 수행할 수도 있습니다.

기본적으로 가져오기 프로세스는 암호를 제외한 모든 사용자 속성의 값을 설정합니다. 즉, 사용자가 처음 로그인할 때 암호를 변경해야 합니다. 사용자는 이 방법을 사용하여 가져올 때 `RESET_REQUIRED` 상태가 됩니다.

또는 기존 암호 해시를 사용하여 사용자를 가져올 수 있습니다. 가져오기 작업 생성 중에 암호 해싱 알고리즘을 지정하고 CSV 파일에 암호 해시를 포함하면 Amazon Cognito는 기존 암호로 사용자를 가져옵니다. 이러한 사용자는 `CONFIRMED` 상태로 생성되며 암호를 재설정하지 않고도 즉시 로그인할 수 있습니다. 자세한 내용은 암호 해시를 사용하여 사용자 가져오기 섹션을 참조하세요.

CSV에서 사용자를 가져오는 가장 쉬운 방법은 사용자 풀에서 암호 없는 로그인을 활성화하는 것입니다. 이메일 주소 및 전화번호 속성과 올바른 사용자 풀 구성을 사용하면 가져오기 작업이 완료된 직후 이메일 또는 SMS 일회용 암호(OTP)로 로그인할 수 있습니다. 자세한 내용은 가져온 사용자에게 암호 재설정 요구 섹션을 참조하세요.

`Permanent` 파라미터를 `true`로 설정하는 AdminSetUserPassword API 요청을 이용해 사용자의 암호를 설정할 수도 있습니다. CSV 가져오기는 사용자 풀의 청구된 월별 활성 사용자(MAU)에 영향을 주지 않습니다. 그러나 암호 재설정 작업은 MAU를 생성합니다. 즉시 활성화되지 않을 수 있는 암호를 사용하는 많은 수의 사용자를 가져올 때 비용을 관리하려면 사용자가 로그인하고 `RESET_REQUIRED` 문제를 받을 때 새 암호를 입력하도록 애플리케이션을 설정하세요.

> **Note**
>
> 각 사용자의 생성 날짜는 해당 사용자를 사용자 풀로 가져온 시간입니다. 생성 날짜는 가져온 속성 중 하나가 아닙니다.

사용자 가져오기 작업을 생성하는 단계

1. AWS Identity and Access Management (IAM) 콘솔에서 Amazon CloudWatch Logs 역할을 생성합니다.
2. 사용자 가져오기.csv 파일을 만듭니다.
3. 사용자 가져오기 작업을 생성하고 실행합니다. 필요에 따라 암호 해싱 알고리즘을 지정하여 기존 암호 해시가 있는 사용자를 가져옵니다.
4. 사용자 가져오기.csv 파일을 업로드합니다.
5. 사용자 가져오기 작업을 시작하고 실행합니다.
6. CloudWatch를 사용하여 이벤트 로그를 확인합니다.
7. 암호 해시를 가져오지 않은 경우 가져온 사용자에게 암호를 재설정하도록 요구합니다.

추가 리소스

- 사용자 풀 간 사용자 계정 내보내기를 위한 Cognito 사용자 프로필 내보내기 참조 아키텍처

주제

- CloudWatch Logs IAM 역할 생성
- 사용자 가져오기 CSV 파일 생성
- Amazon Cognito 사용자 풀 가져오기 작업 생성 및 실행
- CloudWatch 콘솔에서 사용자 풀 가져오기 결과 보기
- 가져온 사용자에게 암호 재설정 요구
- 암호 해시를 사용하여 사용자 가져오기

CloudWatch Logs IAM 역할 생성

Amazon Cognito CLI 또는 API를 사용하는 경우 CloudWatch IAM 역할을 생성해야 합니다. 다음 절차에서는 Amazon Cognito가 가져오기 작업의 결과를 CloudWatch Logs Logs에 기록하는 데 사용할 수 있는 IAM 역할을 생성하는 방법을 설명합니다.

> **Note**
>
> Amazon Cognito 콘솔에서 가져오기 작업을 생성할 때 IAM 역할을 생성할 수 있습니다. Create a new IAM role(새 IAM 역할 생성)을 선택하면 Amazon Cognito는 해당 역할에 적절한 신뢰 정책과 IAM 정책을 자동으로 적용합니다.

사용자 풀 가져오기를 위한 CloudWatch Logs IAM 역할을 생성하려면(AWS CLI, API)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/ IAM 콘솔을 엽니다.
2. 에 대한 새 IAM 역할을 생성합니다 AWS 서비스. 자세한 지침은 AWS Identity and Access Management 사용 설명서의 AWS 서비스에 대한 역할 생성을 참조하세요.
   a. Trusted entity type(신뢰할 수 있는 엔티티 유형)의 Use case(사용 사례)를 선택할 때 아무 서비스를 선택합니다. Amazon Cognito는 현재 서비스 사용 사례에 나열되지 않습니다.
   b. Add permissions(권한 추가) 화면에서 Create policy(정책 생성)을 선택하고 다음 정책 설명을 삽입합니다. 예를 들어 *REGION*을 사용자 풀 AWS 리전 의 로 바꿉니다`us-east-1`. *ACCOUNT*를 AWS 계정 ID(예: `111122223333`)로 바꿉니다.

      JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:111122223333:log-group:/aws/
cognito/*"
            ]
        }
    ]
```

```
}
```

3. 역할을 만들 때 Amazon Cognito를 신뢰할 수 있는 개체로 선택하지 않았으므로, 이제 역할의 신뢰 관계를 수동으로 편집해야 합니다. IAM 콘솔의 탐색 창에서 Roles(역할)를 선택하고 생성한 새 역할을 선택합니다.
4. 신뢰 관계 탭을 선택합니다.
5. 신뢰 정책 편집을 선택합니다.
6. 다음 정책 설명을 Edit trust policy(신뢰 정책 편집)에 붙여넣어 기존 텍스트를 대체합니다.

   JSON

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

7. 정책 업데이트를 선택합니다.
8. 역할 ARN을 기록해 둡니다. 가져오기 작업을 생성할 때 ARN을 입력해야 합니다.

또는 AWS CLI를 사용하여 동일한 역할을 생성할 수 있습니다. 다음 명령은 Amazon Cognito가 수임할 수 있는 역할을 생성하고, CloudWatch Logs에 쓰기를 허용하는 정책을 연결하고, 역할 ARN을 반환합니다. *REGION*을 사용자 풀 AWS 리전 의 로 바꾸고 *ACCOUNT*를 AWS 계정 ID로 바꿉니다.

사용자 풀 가져오기를 위한 CloudWatch Logs IAM 역할을 생성하려면(AWS CLI)

1. Amazon Cognito가 역할을 수임하도록 허용하는 신뢰 정책을 사용하여 역할을 생성합니다.

```
aws iam create-role \
    --role-name CognitoUserImportRole \
    --assume-role-policy-document '{
        "Version": "2012-10-17",
```

```
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": { "Service": "cognito-idp.amazonaws.com" },
                "Action": "sts:AssumeRole"
            }
        ]
    }'
```

2. 역할이 CloudWatch Logs에 가져오기 결과를 쓸 수 있도록 허용하는 정책을 연결합니다.

```
aws iam put-role-policy \
    --role-name CognitoUserImportRole \
    --policy-name CognitoUserImportLogsPolicy \
    --policy-document '{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "logs:CreateLogGroup",
                    "logs:CreateLogStream",
                    "logs:PutLogEvents",
                    "logs:DescribeLogStreams"
                ],
                "Resource": "arn:aws:logs:REGION:ACCOUNT:log-group:/aws/cognito/*"
            }
        ]
    }'
```

3. 가져오기 작업을 생성할 때 *ROLE_ARN## ### ## ARN*을 검색합니다.

```
aws iam get-role \
    --role-name CognitoUserImportRole \
    --query 'Role.Arn' --output text
```

사용자 가져오기 CSV 파일 생성

기존 사용자를 사용자 풀로 가져오려면 먼저 가져올 사용자와 사용자의 속성이 포함된 쉼표로 구분된 값(CSV) 파일을 생성해야 합니다. 사용자 풀에서는 사용자 풀의 속성 스키마를 반영하는 헤더가 있는 사용자 가져오기 파일을 검색할 수 있습니다. 그런 다음 CSV 파일 형식 지정의 형식 지정 요구 사항과 일치하는 사용자 정보를 삽입할 수 있습니다.

CSV 파일 헤더 다운로드(콘솔)

다음 절차를 사용하여 CSV 헤더 파일을 다운로드합니다.

CSV 파일 헤더를 다운로드하려면

1. Amazon Cognito 콘솔로 이동합니다. 자격 AWS 증명을 입력하라는 메시지가 표시될 수 있습니다.
2. 사용자 풀(User Pools)을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택합니다.
4. 사용자 메뉴를 선택합니다.
5. Import users(사용자 가져오기) 섹션에서 Create an import job(가져오기 작업 생성)을 선택합니다.
6. Upload CSV(CSV 업로드)에서 template.csv 링크를 선택하고 CSV 파일을 다운로드합니다.

CSV 파일 헤더 다운로드(AWS CLI)

올바른 헤더 목록을 얻으려면 사용자 가져오기의 사용자 메뉴에서 가져오기 작업 만들기를 선택합니다. 다음 대화 상자에서 `template.csv` 링크를 선택하여 사용자 풀 속성이 포함된 템플릿 파일을 다운로드합니다.

다음 CLI 명령을 실행할 수도 있습니다. 여기서 *`USER_POOL_ID`*는 사용자를 가져올 사용자 풀의 사용자 풀 식별자입니다.

```
aws cognito-idp get-csv-header --user-pool-id "USER_POOL_ID"
```

샘플 응답:

```
{
    "CSVHeader": [
        "name",
        "given_name",
        "family_name",
        "middle_name",
        "nickname",
        "preferred_username",
        "profile",
        "picture",
        "website",
        "email",
```

```
        "email_verified",
        "gender",
        "birthdate",
        "zoneinfo",
        "locale",
        "phone_number",
        "phone_number_verified",
        "address",
        "updated_at",
        "cognito:mfa_enabled",
        "cognito:username"
    ],
    "UserPoolId": "USER_POOL_ID"
}
```

CSV 파일 형식 지정

다운로드한 사용자 가져오기 CSV 헤더 파일은 다음 문자열처럼 보입니다. 사용자 풀에 추가한 사용자 지정 속성도 포함되어 있습니다.

```
cognito:username,name,given_name,family_name,middle_name,nickname,preferred_username,profile,pi
```

CSV 파일을 편집하여 이 헤더와 사용자의 속성 값이 포함되고 다음 규칙에 따라 형식이 지정되게 해야 합니다.

> **Note**
>
> 전화 번호의 올바른 형식 등, 속성 값에 대한 자세한 내용은 사용자 속성 작업 섹션을 참조하세요.

- 파일의 첫 번째 행은 사용자 속성 이름을 포함하는 다운로드된 헤더 행입니다.
- CSV 파일의 열 순서는 중요하지 않습니다.
- 첫 번째 행 뒤의 각 행에는 사용자의 속성 값이 포함됩니다.
- 헤더의 모든 열이 있어야 하지만 모든 열에 값을 제공할 필요는 없습니다.
- 다음과 같은 속성이 필요합니다.
  - cognito:username
  - email_verified 또는 phone_number_verified

  - 각 사용자의 자동 확인 속성 중 하나 이상이 `true`이어야 합니다. 자동 확인 속성은 새 사용자가 사용자 풀에 가입할 때 Amazon Cognito에서 자동으로 코드를 보내는 이메일 주소 또는 전화 번호입니다.
  - 사용자 풀에는 하나 이상의 자동 확인 속성(email_verified 또는 phone_number_verified)이 있어야 합니다. 사용자 풀에 자동 확인된 속성이 없으면 가져오기 작업이 시작되지 않습니다.
  - 사용자 풀에 자동 확인 속성이 하나만 있는 경우 각 사용자에 대해 해당 속성을 확인해야 합니다. 예를 들어 사용자 풀에 phone_number만 자동 검증 속성으로 설정되어 있는 경우 각 사용자의 phone_number_verified 값이 `true`이어야 합니다.

    > **Note**
    >
    > 사용자가 암호를 재설정하려면 확인된 이메일 또는 전화 번호가 있어야 합니다. Amazon Cognito는 CSV 파일에 지정된 이메일 또는 전화 번호로 재설정 암호 코드가 포함된 메시지를 보냅니다. 전화 번호로 메시지를 전송하는 경우 SMS 메시지를 통해 전송됩니다. 자세한 내용은 가입 시 연락처 정보 확인 섹션을 참조하세요.

  - email(email_verified가 `true`인 경우)
  - phone_number(phone_number_verified가 `true`인 경우)
  - 사용자 풀을 생성할 때 필수 속성으로 표시한 모든 속성
- 문자열 속성 값은 인용 부호로 묶여 있지 않아야 합니다.
- 속성 값에 쉼표가 포함된 경우 쉼표 앞에 백슬래시(\)를 넣어야 합니다. 이는 CSV 파일의 필드가 쉼표로 구분되어 있기 때문입니다.
- CSV 파일 내용은 바이트 순서 표시가 없는 UTF-8 형식이어야 합니다.
- cognito:username 필드는 필수이며 사용자 풀 내에서 고유해야 합니다. 유니 코드 문자열일 수 있습니다. 단, 공백 또는 탭을 포함할 수 없습니다.
- birthdate 값이 있는 경우 이 값은 *`mm/dd/yyyy`* 포맷이어야 합니다. 즉, 예를 들어 생년월일 1985년 2월 1일은 **`02/01/1985`**로 인코딩되어야 합니다.
- cognito:mfa_enabled 필드는 사용자 풀의 MFA 요구 사항에 부합해야 합니다. 사용자 풀에서 다중 인증(MFA)을 필수로 설정한 경우, 모든 사용자에 대해 이 필드는 `true`이거나 비어 있어야 합니다. MFA를 해제하도록 설정한 경우 이 필드는 모든 사용자에 대해 `false`이거나 비어 있어야 합니다. 빈 값은 가져온 사용자의 MFA 활성화 상태를 사용자 풀에 필요한 상태로 설정합니다. `cognito:mfa_enabled` 값을 설정했는지 여부에 관계없이 유효한 MFA 요소 없이 MFA 필수 사용자 풀에서 사용자를 가져올 수 있습니다. 이 상태의 사용자는 MFA가 활성화되어 있지만 이메일 속성, 전화번호 속성 또는 TOTP를 구성하고 해당 구성이 사용자 풀에서 유효한 MFA 팩터가 될 때까지 로그인할 수 없습니다.

- 최대 행 길이는 16,000자입니다.
- 최대 CSV 파일 크기는 100MB입니다.
- 파일의 최대 행(사용자) 수는 500,000개입니다. 헤더 행은 이 최대값에 포함되지 않습니다.
- updated_at 필드 값은 초 단위의 Epoch 시간이어야 합니다(예: **1471453471**).
- 속성 값의 선행 또는 후행 공백은 모두 잘립니다.

다음 목록은 사용자 지정 속성이 없는 사용자 풀에 대한 CSV 가져오기 파일의 예입니다. 사용자 풀 스키마가 이 예시와 다를 수 있습니다. 이 경우 사용자 풀에서 다운로드한 CSV 템플릿에 테스트 값을 입력해야 합니다.

```
cognito:username,name,given_name,family_name,middle_name,nickname,preferred_username,profile,pi
John,,John,Doe,,,,,,,johndoe@example.com,TRUE,,02/01/1985,,,+12345550100,TRUE,123 Any
 Street,,FALSE
Jane,,Jane,Roe,,,,,,,janeroe@example.com,TRUE,,01/01/1985,,,+12345550199,TRUE,100 Main
 Street,,FALSE
```

Amazon Cognito 사용자 풀 가져오기 작업 생성 및 실행

이 섹션에서는 Amazon Cognito 콘솔과 AWS Command Line Interface ()를 사용하여 사용자 풀 가져오기 작업을 생성하고 실행하는 방법을 설명합니다AWS CLI.

주제

- CSV 파일에서 사용자 가져오기(콘솔)
- 사용자 가져오기(AWS CLI)

CSV 파일에서 사용자 가져오기(콘솔)

다음 절차에서는 CSV 파일에서 사용자를 가져오는 방법에 대해 설명합니다.

CSV 파일에서 사용자를 가져오려면(콘솔)

1. Amazon Cognito 콘솔로 이동합니다. 자격 AWS 증명을 입력하라는 메시지가 표시될 수 있습니다.
2. 사용자 풀(User Pools)을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택합니다.
4. 사용자 메뉴를 선택합니다.
5. Import users(사용자 가져오기) 섹션에서 Create an import job(가져오기 작업 생성)을 선택합니다.

6. Create import job(가져오기 작업 생성) 페이지에 Job name(작업 이름)을 입력합니다.
7. Create a new IAM role(새 IAM 역할 생성) 또는Use an existing IAM role(기존 IAM 역할 사용)을 선택합니다.

   a. Create a new IAM role(새 IAM 역할 생성)을 선택한 경우 새 역할의 이름을 입력합니다. Amazon Cognito는 올바른 권한과 신뢰 관계를 가진 역할을 자동으로 생성합니다. 가져오기 작업을 생성하는 IAM 보안 주체는 IAM 역할을 생성할 권한이 있어야 합니다.

   b. Use an existing IAM role(기존 IAM 역할 사용)을 선택했다면 IAM role selection(IAM 역할 선택) 목록에서 역할을 선택합니다. 이 역할에는 CloudWatch Logs IAM 역할 생성에서 설명하는 권한 및 신뢰 정책이 있어야 합니다.
8. CSV 업로드에서 파일 선택을 선택하고 준비한 CSV 파일을 연결합니다.
9. Create job(작업 생성)을 선택하여 작업을 제출하되, 시작은 나중에 합니다. 작업을 제출하고 즉시 시작하려면 Create and start job(작업 생성 및 시작)을 선택합니다.
10. 작업을 생성했지만 시작하지 않았다면 나중에 작업을 시작할 수 있습니다. 사용자 가져오기의 사용자 메뉴에서 가져오기 작업을 선택하고 시작을 선택합니다. AWS SDK에서 StartUserImportJob API 요청을 제출할 수도 있습니다.
11. 사용자 가져오기의 사용자 메뉴에서 사용자 가져오기 작업의 진행 상황을 모니터링합니다. 작업이 실패할 경우 Status(상태) 값을 선택합니다. 자세한 내용을 보려면 View the CloudWatch logs for more details(CloudWatch 로그 상세 정보 보기)를 선택하고 CloudWatch Logs 콘솔에서 문제를 검토합니다.

사용자 가져오기(AWS CLI)

다음 CLI 명령을 사용하여 사용자를 사용자 풀로 가져올 수 있습니다.

- `create-user-import-job`
- `get-csv-header`
- `describe-user-import-job`
- `list-user-import-jobs`
- `start-user-import-job`
- `stop-user-import-job`

이러한 명령의 명령줄 옵션 목록을 표시하려면 `help` 명령줄 옵션을 사용합니다. 예:

```
aws cognito-idp get-csv-header help
```

사용자 가져오기 작업 생성

CSV 파일을 생성한 후 다음 CLI 명령을 실행하여 사용자 가져오기 작업을 생성합니다. 여기서 *JOB_NAME*은 작업에 대해 선택하는 이름이고, *USER_POOL_ID*는 새 사용자가 추가되는 사용자 풀의 사용자 풀 ID이며, *ROLE_ARN*은에서 받은 역할 ARN입니다. CloudWatch Logs IAM 역할 생성

```
aws cognito-idp create-user-import-job --job-name "JOB_NAME" --user-pool-id
 "USER_POOL_ID" --cloud-watch-logs-role-arn "ROLE_ARN"
```

응답에 반환되는 *PRE_SIGNED_URL*은 15 분 동안 유효합니다. 이 기간이 지나면 만료되며 새 URL을 가져오려면 새 사용자 가져오기 작업을 생성해야 합니다.

Example응답:

```
{
    "UserImportJob": {
        "Status": "Created",
        "SkippedUsers": 0,
        "UserPoolId": "USER_POOL_ID",
        "ImportedUsers": 0,
        "JobName": "JOB_NAME",
        "JobId": "JOB_ID",
        "PreSignedUrl": "PRE_SIGNED_URL",
        "CloudWatchLogsRoleArn": "ROLE_ARN",
        "FailedUsers": 0,
        "CreationDate": 1470957431.965
    }
}
```

사용자 가져오기 작업의 상태 값

사용자 가져오기 명령에 대한 응답에 다음 `Status` 값 중 하나가 표시됩니다.

- `Created` - 작업이 생성되었지만 시작되지 않았습니다.
- `Pending` - 전환 상태입니다. 작업을 시작했지만 아직 사용자 가져오기를 시작하지 않았습니다.
- `InProgress` - 작업이 시작되었으며 사용자를 가져오는 중입니다.
- `Stopping` - 작업을 중지했지만 작업에서 아직 사용자 가져오기가 중지되지 않았습니다.
- `Stopped` - 작업을 중지했으며 작업에서 사용자 가져오기가 중지되었습니다.

- Succeeded - 작업이 성공적으로 완료되었습니다.
- Failed - 오류로 인해 작업이 중지되었습니다.
- Expired - 작업을 생성했지만 24~48시간 이내에 작업을 시작하지 않았습니다. 작업과 연결된 모든 데이터가 삭제되었으며 작업을 시작할 수 없습니다.

CSV 파일 업로드

다음 curl 명령을 사용하여 사용자 데이터가 포함된 CSV 파일을 create-user-import-job 명령의 응답에서 얻은 미리 서명된 URL에 업로드합니다.

```
curl -v -T "PATH_TO_CSV_FILE" -H "x-amz-server-side-encryption:aws:kms"
 "PRE_SIGNED_URL"
```

이 명령의 출력에서 "We are completely uploaded and fine"이라는 문구를 찾습니다. 이 구문은 파일이 성공적으로 업로드되었음을 나타냅니다. 가져오기 작업을 실행한 후에는 사용자 풀이 가져오기 파일에 정보를 보관하지 않습니다. 완료되거나 만료되면 Amazon Cognito는 업로드된 CSV 파일을 삭제합니다.

사용자 가져오기 작업 설명

사용자 가져오기 작업에 대한 설명을 보려면 다음 명령을 사용합니다. 여기서 *USER_POOL_ID*는 사용자 풀 ID이며, *JOB_ID*는 사용자 가져오기 작업을 생성할 때 반환된 작업 ID입니다.

```
aws cognito-idp describe-user-import-job --user-pool-id "USER_POOL_ID" --job-id
 "JOB_ID"
```

Example샘플 응답:

```
{
    "UserImportJob": {
        "Status": "Created",
        "SkippedUsers": 0,
        "UserPoolId": "USER_POOL_ID",
        "ImportedUsers": 0,
        "JobName": "JOB_NAME",
        "JobId": "JOB_ID",
        "PreSignedUrl": "PRE_SIGNED_URL",
        "CloudWatchLogsRoleArn":"ROLE_ARN",
        "FailedUsers": 0,
        "CreationDate": 1470957431.965
```

```
    }
}
```

앞의 샘플 출력에서 *PRE_SIGNED_URL*은 CSV 파일을 업로드한 URL입니다. *ROLE_ARN*은 역할을 생성할 때 받은 CloudWatch Logs 역할 ARN입니다.

사용자 가져오기 작업 나열

사용자 가져오기 작업을 나열하려면 다음 명령을 사용합니다.

```
aws cognito-idp list-user-import-jobs --user-pool-id "USER_POOL_ID" --max-results 2
```

Example샘플 응답:

```
{
    "UserImportJobs": [
        {
            "Status": "Created",
            "SkippedUsers": 0,
            "UserPoolId": "USER_POOL_ID",
            "ImportedUsers": 0,
            "JobName": "JOB_NAME",
            "JobId": "JOB_ID",
            "PreSignedUrl":"PRE_SIGNED_URL",
            "CloudWatchLogsRoleArn":"ROLE_ARN",
            "FailedUsers": 0,
            "CreationDate": 1470957431.965
        },
        {
            "CompletionDate": 1470954227.701,
            "StartDate": 1470954226.086,
            "Status": "Failed",
            "UserPoolId": "USER_POOL_ID",
            "ImportedUsers": 0,
            "SkippedUsers": 0,
            "JobName": "JOB_NAME",
            "CompletionMessage": "Too many users have failed or been skipped during the
 import.",
            "JobId": "JOB_ID",
            "PreSignedUrl":"PRE_SIGNED_URL",
            "CloudWatchLogsRoleArn":"ROLE_ARN",
            "FailedUsers": 5,
            "CreationDate": 1470953929.313
```

```
        }
    ],
    "PaginationToken": "PAGINATION_TOKEN"
}
```

작업은 마지막으로 생성한 것부터 처음으로 생성한 것까지 시간순으로 나열됩니다. 두 번째 작업 다음에 있는 *PAGINATION_TOKEN* 문자열은 이 list 명령에 대한 추가 결과가 있음을 나타냅니다. 추가 결과를 나열하려면 다음과 같이 `--pagination-token` 옵션을 사용합니다.

```
aws cognito-idp list-user-import-jobs --user-pool-id "USER_POOL_ID" --max-results 10 --
pagination-token "PAGINATION_TOKEN"
```

사용자 가져오기 작업 시작

사용자 가져오기 작업을 시작하려면 다음 명령을 사용합니다.

```
aws cognito-idp start-user-import-job --user-pool-id "USER_POOL_ID" --job-id "JOB_ID"
```

계정당 한 번에 하나의 가져오기 작업만 활성화할 수 있습니다.

Example샘플 응답:

```
{
    "UserImportJob": {
        "Status": "Pending",
        "StartDate": 1470957851.483,
        "UserPoolId": "USER_POOL_ID",
        "ImportedUsers": 0,
        "SkippedUsers": 0,
        "JobName": "JOB_NAME",
        "JobId": "JOB_ID",
        "PreSignedUrl":"PRE_SIGNED_URL",
        "CloudWatchLogsRoleArn": "ROLE_ARN",
        "FailedUsers": 0,
        "CreationDate": 1470957431.965
    }
}
```

사용자 가져오기 작업 중지

진행 중인 사용자 가져오기 작업을 중지하려면 다음 명령을 사용합니다. 작업을 중지한 후에는 다시 시작할 수 없습니다.

```
aws cognito-idp stop-user-import-job --user-pool-id "USER_POOL_ID" --job-id "JOB_ID"
```

Example샘플 응답:

```
{
    "UserImportJob": {
        "CompletionDate": 1470958050.571,
        "StartDate": 1470958047.797,
        "Status": "Stopped",
        "UserPoolId": "USER_POOL_ID",
        "ImportedUsers": 0,
        "SkippedUsers": 0,
        "JobName": "JOB_NAME",
        "CompletionMessage": "The Import Job was stopped by the developer.",
        "JobId": "JOB_ID",
        "PreSignedUrl":"PRE_SIGNED_URL",
        "CloudWatchLogsRoleArn": "ROLE_ARN",
        "FailedUsers": 0,
        "CreationDate": 1470957972.387
    }
}
```

CloudWatch 콘솔에서 사용자 풀 가져오기 결과 보기

Amazon CloudWatch 콘솔에서 가져오기 작업의 결과를 볼 수 있습니다.

주제

- 결과 보기
- 결과 해석

결과 보기

다음 단계에서는 사용자 풀 가져오기 결과를 보는 방법을 설명합니다.

사용자 풀 가져오기의 결과를 보려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/cloudwatch/ CloudWatch 콘솔을 엽니다.
2. [Logs]를 선택합니다.

3. 사용자 풀 가져오기 작업에 대한 로그 그룹을 선택합니다. 로그 그룹 이름은 `/aws/cognito/userpools/`*`USER_POOL_ID`*`/`*`USER_POOL_NAME`* 형식입니다.
4. 방금 실행한 사용자 가져오기 작업에 대한 로그를 선택합니다. 로그 이름은 *`JOB_ID`*`/`*`JOB_NAME`* 형식입니다. 로그의 결과에서는 행 번호로 사용자를 나타냅니다. 사용자 데이터는 로그에 기록되지 않습니다. 각 사용자에 대해 다음과 유사한 행이 표시됩니다.

   - `[SUCCEEDED] Line Number 5956 - The import succeeded.`
   - `[SKIPPED] Line Number 5956 - The user already exists.`
   - `[FAILED] Line Number 5956 - The User Record does not set any of the auto verified attributes to true. (Example: email_verified to true).`

결과 해석

성공적으로 가져온 사용자는 상태가 'PasswordReset'으로 설정되어 있습니다.

다음과 같은 경우에는 사용자가 가져와지지 않지만 가져오기 작업은 계속됩니다.

- `true`로 설정된 자동 확인 속성이 없는 경우.
- 사용자 데이터가 스키마와 일치하지 않는 경우.
- 내부 오류로 인해 사용자를 가져올 수 없는 경우.

다음과 같은 경우 가져오기 작업이 실패합니다.

- Amazon CloudWatch Logs 역할을 수임할 수 없거나, 해당 역할에 올바른 액세스 정책이 없거나, 해당 ㅇ역할이 삭제된 경우.
- 사용자 풀이 삭제된 경우.
- Amazon Cognito가 .csv 파일을 구문 분석할 수 없는 경우.

가져온 사용자에게 암호 재설정 요구

사용자 풀이 암호 기반 로그인만 제공하는 경우 사용자는 암호를 가져온 후 암호를 재설정해야 합니다. 처음 로그인할 때는 어떤 암호든 입력할 수 있습니다. Amazon Cognito는 애플리케이션의 로그인 요청에 대한 API 응답에 새 암호를 입력하라는 메시지를 표시합니다.

사용자 풀에 암호 없는 인증 요소가 있는 경우 Amazon Cognito는 가져온 사용자의 인증 요소를 기본값으로 사용합니다. 새 암호를 입력하라는 메시지가 표시되지 않으며 암호 없는 이메일 또는 SMS OTP

로 즉시 로그인할 수 있습니다. 또한 사용자에게 암호를 설정하라는 메시지를 표시하여 사용자 이름-암호 및 패스키와 같은 다른 로그인 방법을 완료할 수 있습니다. 다음 조건은 사용자 가져오기 후 암호 없는 로그인에 적용됩니다.

1. 사용 가능한 암호 없는 로그인 요소에 해당하는 속성을 가진 사용자를 가져와야 합니다. 사용자가 이메일 주소로 로그인할 수 있는 경우 `email` 속성을 가져와야 합니다. 전화번호인 경우 `phone_number` 속성을 가져와야 합니다. 둘 다인 경우 두 속성의 값을 가져옵니다.
2. 일반적으로 사용자는 암호를 재설정해야 하는 `RESET_REQUIRED` 상태로 가져옵니다. 암호 없는 팩터로 로그인할 수 있는 기능으로 가져온 경우 Amazon Cognito는 상태를 `CONFIRMED`로 설정합니다.

설정 방법 및 애플리케이션에서 인증 흐름을 구성하는 방법을 포함하여 암호 없는 인증에 대한 자세한 내용은 Amazon Cognito 사용자 풀로 인증 섹션을 참조하세요.

다음 절차에서는 CSV 파일을 가져온 후 `RESET_REQUIRED`의 로컬 사용자를 대상으로 맞춤형 로그인 메커니즘을 구축하여 사용자 경험을 개선하는 방법을 설명합니다. 사용자가 관리형 로그인으로 로그인하는 경우 암호 찾기 옵션을 선택하고 이메일 또는 문자 메시지의 코드를 제공하고 암호를 설정하도록 지시합니다.

가져온 사용자에게 암호 재설정 요구

1. 앱에서 임의의 암호를 사용하여 `InitiateAuth`를 통해 현재 사용자의 로그인을 자동으로 시도합니다.
2. `PreventUserExistenceErrors`를 사용할 경우 Amazon Cognito가 `NotAuthorizedException`을 반환합니다. 그렇지 않은 경우에는 `PasswordResetRequiredException`를 반환합니다.
3. 앱에서 `ForgotPassword` API를 요청하고 사용자의 암호를 재설정합니다.

    a. 앱에서 `ForgotPassword` API 요청에 사용자 이름을 제출합니다.

    b. Amazon Cognito는 확인된 이메일 또는 전화 번호에 코드를 보냅니다. 대상은 CSV 파일의 `email_verified` 및 `phone_number_verified`에 입력한 값에 따라 달라집니다. `ForgotPassword` 요청에 대한 응답은 코드의 대상을 나타냅니다.

> **Note**
>
> 이메일이나 전화 번호를 인증하도록 사용자 풀을 구성해야 합니다. 자세한 내용은 사용자 계정 가입 및 확인 섹션을 참조하세요.

c. 앱은 코드가 전송된 위치를 확인하라는 메시지를 사용자에게 표시하며, 코드와 새 암호를 입력하라고 지시합니다.

d. 사용자가 앱에 코드와 새 암호를 입력합니다.

e. 앱은 `ConfirmForgotPassword` API 요청에 코드와 새 암호를 제출합니다.

f. 앱이 사용자를 로그인으로 리디렉션합니다.

암호 해시를 사용하여 사용자 가져오기

기존 인증 시스템에서 Amazon Cognito로 사용자를 마이그레이션할 때 사용자 속성과 함께 암호 해시를 가져올 수 있습니다. 그러면 사용자는 암호 재설정 없이 기존 암호로 즉시 로그인할 수 있습니다. 사용자가 처음 로그인하면 Amazon Cognito는 자격 증명을 네이티브 Amazon Cognito 인증 시스템으로 투명하게 마이그레이션합니다.

암호 해시 가져오기 작동 방식

암호 해시 가져오기를 사용하면 기존 암호로 로그인하는 기능을 유지하면서 사용자를 마이그레이션할 수 있습니다. 가져오기 프로세스는 다음과 같이 작동합니다.

1. 가져오기 작업을 생성하고 소스 시스템이 암호 해시를 생성하는 데 사용한 암호 해싱 알고리즘을 지정합니다.
2. CSV 파일에 각 사용자의 암호 해시 값과 함께 `password_hash` 열을 포함합니다.
3. Amazon Cognito는 암호 해시가 있는 사용자를 가져와`CONFIRMED`서 상태로 생성합니다. 사용자는 즉시 로그인할 수 있습니다.
4. 사용자가 처음으로 로그인하면 Amazon Cognito는 가져온 해시와 비교하여 암호를 확인합니다. 확인에 성공하면 Amazon Cognito는 사용자의 자격 증명을 기본 인증 시스템으로 마이그레이션합니다.
5. 이후의 모든 로그인은 기본 Amazon Cognito 인증 시스템을 사용합니다.

> **Important**
>
> 단일 가져오기 작업의 모든 암호 해시는 동일한 알고리즘을 사용해야 합니다. 가져오기 작업을 생성할 때 작업 수준에서 알고리즘을 지정합니다.

> **Note**
>
> 사용자가 첫 번째 로그인을 완료하고 Amazon Cognito가 자격 증명을 마이그레이션할 때까지 해당 사용자에 대해 보안 원격 암호(SRP) 인증을 사용할 수 없습니다. 아직 로그인하지 않은 가져온 암호 해시가 있는 사용자에게 `USER_PASSWORD_AUTH` 또는 `ADMIN_USER_PASSWORD_AUTH` 흐름을 사용합니다.

지원되는 암호 해싱 알고리즘

Amazon Cognito는 가져오기를 위해 다음과 같은 암호 해싱 알고리즘을 지원합니다.

`BCRYPT`

bcrypt 적응형 해시 함수입니다. Amazon Cognito는 해시 문자열에서 필요한 모든 파라미터(염, 비용 계수)를 추출합니다.

형식: `$2<a/b/x/y>$[cost]$[22-char salt][31-char hash]`

예: `$2b$10$CtA.Rcu/szzn9U00wpUjOuN3vrgJRZycv4aOzcP3GzqzO8UDPEFq6`

최대 비용 계수: 12

`SCRYPT`

암호 기반 키 파생 함수입니다. Amazon Cognito는 해시 문자열에서 필요한 모든 파라미터를 추출합니다.

형식: `N$r$p$hexSalt$hexHash`

예: `65536$8$1$304dbaef7c5e828dc19c98f0600d18fe $4f69c498c12cd102d057356facf8d77e8d42407090491ea32c5b038f5a18c099`

최대 파라미터: N(CPU/메모리 비용) = 65536, r(블록 크기) = 8, p(병렬성) = 1

`ARGON2ID`

Argon2id 암호 해싱 알고리즘입니다. Amazon Cognito는 해시 문자열에서 필요한 모든 파라미터를 추출합니다.

형식: `$argon2id$v=N$m=M,t=T,p=P$salt$hash`

예: `$argon2id$v=19$m=19456,t=2,p=1$ko/G5o1ms+ML08P95sQ8DA $AkVbvWSOqz7Hs3qthhWKxicOWnGLN+MBmpwc3emi5VA`

최대 파라미터: m(KiB의 메모리) = 19456, t(반복) = 2, p(병렬성) = 1

`PBKDF2_SHA256`

SHA-256을 사용하는 암호 기반 키 파생 함수 2. Amazon Cognito는 해시 문자열에서 필요한 모든 파라미터를 추출합니다.

형식: `$pbkdf2-sha256$iterations$salt$hash`

예: `$pbkdf2-sha256$600000$1XZlmwLQ2hhM3JYuCPiArQ$Pfheg9Zi/v5lXU4yyLA0WFUYEd/rlaVbzrM9oMD6IrA`

최대 반복 횟수: 600000

> **Note**
>
> 지원되는 모든 알고리즘은 자체 설명이므로 Amazon Cognito는 해시 문자열에서 직접 필요한 모든 파라미터(예: 솔트, 비용 계수 및 반복)를 추출할 수 있습니다. 가져오기 작업을 생성할 때만 알고리즘 이름을 지정하면 됩니다.

암호 해시의 파라미터 값이 위에 나열된 최대 범위를 초과하는 경우 해당 사용자의 가져오기가 실패합니다. 가져오기 전에 소스 시스템의 암호 해싱 구성을 검토하여 호환성을 확인합니다.

CSV 파일에 암호 해시 추가

사용자 가져오기를 위해 CSV 템플릿을 다운로드하면 템플릿에 `password_hash` 열이 포함됩니다. 기존 암호로 가져오려는 사용자의 암호 해시 값으로이 열을 채웁니다.

암호 해시 가져오기에 대한 CSV 형식 지정 규칙

- 가져오기 작업이 암호 해싱 알고리즘을 지정할 때 `password_hash` 열이 필요합니다. 특정 사용자의 값을 비워 두면 Amazon Cognito는 암호 없이 해당 사용자를 가져오고 사용자를 `RESET_REQUIRED` 상태로 설정합니다.
- 가져오기 작업이 암호 해싱 알고리즘을 지정하지만 CSV 파일에 `password_hash` 열이 포함되지 않은 경우 작업이 실패합니다.
- 암호 해시 값의 형식이 잘못되었거나 지정된 알고리즘의 예상 형식과 일치하지 않는 경우 해당 사용자의 가져오기가 실패합니다. Amazon Cognito는 사용자를 생성하지 않으며 Amazon CloudWatch Logs 로그에 실패를 기록합니다.

- 암호 해시 값은 대/소문자를 구분합니다. 알고리즘이 기대하는 정확한 형식과 일치하는지 확인합니다.

Example암호 해시가 포함된 샘플 CSV

다음 예제에서는 bcrypt 암호 해시가 있는 사용자를 가져오는 CSV 파일을 보여줍니다.

```
cognito:username,email,email_verified,phone_number,phone_number_verified,password_hash
alejandro_rosalez,alejandro_rosalez@example.com,TRUE,+12345550100,TRUE,$2b$10$CtA.Rcu/
szzn9U00wpUjOuN3vrgJRZycv4aOzcP3GzqzO8UDPEFq6
mary_major,mary_major@example.com,TRUE,+12345550199,TRUE,$2b$10$CtA.Rcu/
szzn9U00wpUjOuN3vrgJRZycv4aOzcP3GzqzO8UDPEFq6
```

암호 해시를 사용하여 가져오기 작업 생성(AWS CLI)

암호 해시가 있는 사용자를 가져오려면 가져오기 작업을 생성할 때 암호 해시 알고리즘을 지정해야 합니다. 가져오기 작업의 모든 사용자는 동일한 알고리즘을 사용해야 합니다.

다음 AWS Command Line Interface 명령을 사용하여 암호 해시가 있는 가져오기 작업을 생성합니다. `--password-hashing-algorithm` 파라미터는 CSV 파일에서 암호 해시를 생성하는 데 사용되는 알고리즘을 지정합니다.

```
aws cognito-idp create-user-import-job \
    --job-name "JOB_NAME" \
    --user-pool-id "USER_POOL_ID" \
    --cloud-watch-logs-role-arn "ROLE_ARN" \
    --password-hashing-algorithm BCRYPT
```

*BCRYPT*를 지원되는 알고리즘 중 하나인 BCRYPT, SCRYPTARGON2ID, 또는 로 바꿉니다PBKDF2_SHA256.

Example샘플 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",
        "PasswordHashingAlgorithm": "BCRYPT",
        "FailedUsers": 0,
        "CreationDate": 1470957431.965
    }
}
```

응답에는 지정한 알고리즘을 확인하는 PasswordHashingAlgorithm 필드가 포함됩니다. CSV 파일을 미리 서명된 URL에 업로드하고에 설명된 대로 작업을 시작합니다CSV 파일 업로드.

암호 해시 가져오기 후 사용자 경험

암호 해시로 가져온 사용자는 다음과 같은 경험을 할 수 있습니다.

- 즉시 로그인: 사용자는 가져온 직후 기존 암호로 로그인할 수 있습니다. 사용자는 암호를 재설정하라는 메시지를 받지 않습니다.
- 사용자 상태: Amazon Cognito는 대신 CONFIRMED 상태로 이러한 사용자를 생성합니다RESET_REQUIRED.
- 투명한 마이그레이션: 사용자가 처음으로 로그인하면 Amazon Cognito는 가져온 해시와 비교하여 암호를 확인합니다. 확인에 성공하면 Amazon Cognito는 자격 증명을 기본 인증 시스템으로 마이그레이션합니다. 이후의 모든 로그인은 네이티브 시스템을 사용합니다.
- 초기 로그인 지연 시간: 가져온 암호 해시가 있는 사용자의 첫 번째 로그인은 후속 로그인보다 약간 더 오래 걸릴 수 있습니다. Amazon Cognito는 가져온 해시에 대해 암호를 확인하고 자격 증명을 기본 인증 시스템으로 마이그레이션해야 합니다.
- 암호 재설정: 사용자가 처음 로그인하기 전에 암호를 재설정하면 가져온 암호 해시는 기본 Amazon Cognito 인증 시스템을 사용하여 새 암호로 대체됩니다.

암호 해시 가져오기 오류 문제 해결

사용자에 대한 암호 해시 가져오기가 실패하면 Amazon Cognito는 Amazon CloudWatch Logs 로그에 실패를 기록합니다. 일반적인 오류 시나리오는 다음과 같습니다.

잘못된 해시

암호 해시가 지정된 알고리즘의 예상 형식과 일치하지 않습니다. 해시 형식이 올바르고에 나열된 형식 중 하나와 일치하는지 확인합니다지원되는 암호 해싱 알고리즘.

파라미터가 범위를 벗어남

암호 해시에는 허용되는 최대 값을 초과하는 파라미터 값(예: 비용 계수 또는 반복)이 포함되어 있습니다. 해시가 호환되지 않는 사용자의 경우 다음 옵션 중 하나를 선택합니다.

- password_hash 필드를 비워 두고 암호 해시 없이 해당 사용자를 가져옵니다. Amazon Cognito는 이를 RESET_REQUIRED 상태로 설정하고 처음 로그인할 때 암호를 재설정해야 합니다.
- 사용자 마이그레이션 Lambda 트리거를 사용하여 CSV 가져오기 대신 로그인 시 해당 사용자를 마이그레이션합니다.
- 소스 시스템을 폐기하기 전에 사용자가 기존 시스템에 로그인할 때 호환되는 파라미터로 암호를 다시 해시합니다.

알고리즘 불일치

암호 해시가 가져오기 작업에 지정된 알고리즘과 다른 알고리즘으로 생성되었습니다. 단일 가져오기 작업의 모든 해시는 동일한 알고리즘을 사용해야 합니다. 알고리즘이 다른 사용자가 있는 경우 각 알고리즘에 대해 별도의 가져오기 작업을 생성합니다.

password_hash 열 누락

작업을 생성할 때 암호 해싱 알고리즘을 지정했지만 CSV 파일에 password_hash 열이 포함되어 있지 않습니다. 알고리즘을 지정하지 않고 열을 CSV 파일에 추가하거나 새 작업을 생성합니다.

사용자의 암호 해시 가져오기가 실패하면 Amazon Cognito는 사용자를 생성하지 않습니다. 가져오기 작업의 다른 사용자는 계속 처리됩니다. Amazon CloudWatch Logs 로그를 검토하여 실패한 사용자와 각 실패의 이유를 식별합니다.

암호 해시 가져오기에 대한 보안 고려 사항

Amazon Cognito는 가져온 암호 해시에 추가 보안 조치를 적용합니다.

- 이중 해싱: Amazon Cognito는 원래 알고리즘의 강도에 관계없이 스토리지 전에 가져온 모든 암호 해시를 추가 암호화 보호 계층과 함께 다시 해시합니다.
- 자동 마이그레이션: 첫 번째 인증에 성공하면 Amazon Cognito는 사용자의 자격 증명을 Amazon Cognito에서 사용하는 기본 보안 원격 암호(SRP) 프로토콜로 마이그레이션합니다. 이렇게 하면 모든 사용자가 궁극적으로 기본 보안 인증 메커니즘을 사용할 수 있습니다.
- 데이터 정리: 가져오기 작업이 완료되면 Amazon Cognito는 임시 스토리지에서 암호 해시가 포함된 업로드된 CSV 파일을 제거합니다.
- 암호 정책이 적용되지 않음: Amazon Cognito는 암호 해시가 있는 사용자를 가져올 때 사용자 풀에 대해 구성한 암호 정책을 적용하지 않습니다. 암호가 구성한 암호 정책을 충족하지 않는 경우에도

Amazon Cognito는 사용자를 계속 가져옵니다. 암호 강도를 적용하고 정책을 충족하지 않는 암호를 거부하려면 가져오기를 위해 해시를 생성하기 전에 소스 시스템에서 암호 강도를 확인합니다.

> **Important**
>
> 암호 해시가 포함된 CSV 파일을 민감한 데이터로 취급합니다. 전송 중 데이터를 보호하고 가져오기가 완료된 후 안전하게 삭제합니다.

사용자 속성 작업

속성은 이름, 이메일 주소, 전화 번호 등 개별 사용자를 식별하는 데 도움이 되는 정보입니다. 새 사용자 풀에는 기본 표준 속성 집합이 있습니다. 에서 사용자 풀 정의에 사용자 지정 속성을 추가할 수도 있습니다 AWS Management Console. 이 섹션에서는 이러한 속성을 자세히 설명하며 사용자 풀을 설정하는 방법에 대한 팁을 제공합니다.

사용자에 대한 일부 정보는 속성에 저장하지 마세요. 예를 들어, 사용 통계나 게임 점수와 같이 자주 변경되는 사용자 데이터는 Amazon Cognito Sync 또는 Amazon DynamoDB와 같은 별도의 데이터 스토어에 보관합니다.

사용자-속성 문자열 값을 사용자 풀에 제출하기 전에 입력을 삭제합니다. 제안된 사용자 속성 값을 분석하는 한 가지 방법은 사전 가입과 같은 Lambda 트리거를 사용하는 것입니다.

> **Note**
>
> 일부 문서 및 표준에서는 속성을 멤버로 참조합니다.

주제

- 표준 속성
- 사용자 이름 및 기본 설정 사용자 이름
- 로그인 속성 사용자 지정
- 사용자 지정 속성
- 속성 권한 및 범위

표준 속성

Amazon Cognito는 OpenID Connect 사양에 따라 모든 사용자에게 표준 속성 집합을 할당합니다. 기본적으로 표준 및 사용자 지정 속성 값은 최대 2,048자의 문자열일 수 있지만 일부 속성 값에는 형식 제한이 있습니다.

표준 속성은 다음과 같습니다.

- `name`
- `family_name`
- `given_name`
- `middle_name`
- `nickname`
- `preferred_username`
- `profile`
- `picture`
- `website`
- `gender`
- `birthdate`
- `zoneinfo`
- `locale`
- `updated_at`
- `address`
- `email`
- `phone_number`
- `sub`

`sub`를 제외하고 표준 속성은 기본적으로 모든 사용자에게 선택 사항입니다. 필요한 속성을 만들려면 사용자 풀 생성 프로세스 중에 속성 옆에 있는 필수 확인란을 선택합니다. Amazon Cognito는 각 사용자의 `sub` 속성에 고유한 사용자 식별자 값을 할당합니다. email 및 phone_number 속성만 확인할 수 있습니다.

표준 속성에는 DescribeUserPool API 응답의 `SchemaAttributes` 파라미터에서 볼 수 있는 사전 정의된 속성이 있습니다. 이러한 특성 속성에 대한 사용자 지정 값(예: 데이터 유형, 변경 가능성 및 길이

제약 조건)을 설정할 수 있습니다. 표준 특성 속성을 수정하려면 CreateUserPool 스키마 파라미터에서 사용자 지정 값을 설정합니다. 스키마는 필요한 속성을 설정하는 위치이기도 합니다. Amazon Cognito 콘솔에서 사용자 풀을 생성할 때는 표준 특성의 속성을 수정할 수 없습니다.

> **Note**
>
> 표준 속성이 필수(Required)로 표시된 경우 사용자는 해당 속성의 값을 제공해야 등록할 수 있습니다. 사용자를 생성하지만 필수 속성에는 값을 제공하지 않으려면 관리자가 AdminCreateUser API를 사용합니다. 사용자 풀을 생성한 후에는 필수 속성과 선택적 속성 간을 전환할 수 없습니다.

표준 속성 세부 정보 및 형식 제한

birthdate

값은 YYYY-MM-DD 형식으로 유효한 10자 날짜여야 합니다.

이메일

사용자 및 관리자는 이메일 주소 값을 확인할 수 있습니다.

적절한 AWS 계정 권한이 있는 관리자는 사용자의 이메일 주소를 변경하고 확인된 것으로 표시할 수도 있습니다. AdminUpdateUserAttributes API 또는 admin-update-user-attributes AWS Command Line Interface (AWS CLI) 명령을 사용하여 이메일 주소를 확인된 것으로 표시합니다. 관리자는 이 명령을 사용하여 `email_verified` 속성을 `true`로 변경할 수 있습니다. Amazon Cognito 콘솔의 사용자 메뉴에서 사용자를 편집하여 이메일 주소를 확인됨으로 표시할 수도 있습니다.

값은 @ 기호 및 도메인이 포함된 표준 이메일 형식을 따르는 유효한 이메일 주소 문자열이어야 하며 길이는 최대 2,048자여야 합니다.

phone_number

SMS 멀티 팩터 인증(MFA)이 활성화되어 있는 경우 사용자는 전화 번호를 제공해야 합니다. 자세한 내용은 사용자 풀에 MFA 추가 섹션을 참조하세요.

사용자 및 관리자는 전화 번호 값을 확인할 수 있습니다.

적절한 AWS 계정 권한이 있는 관리자는 사용자의 전화번호를 변경하고 확인된 것으로 표시할 수도 있습니다. AdminUpdateUserAttributes API 또는 admin-update-user-attributes

AWS CLI 명령을 사용하여 확인된 전화번호로 표시합니다. 관리자는 이 명령을 사용하여 `phone_number_verified` 속성을 `true`로 변경할 수 있습니다. Amazon Cognito 콘솔의 사용자 메뉴에서 사용자를 편집하여 전화번호를 확인됨으로 표시할 수도 있습니다.

> **Important**
>
> 전화 번호는 다음과 같은 형식 규칙을 따라야 합니다. 전화 번호는 더하기(**+**) 기호로 시작하고 바로 뒤에 국가 코드가 와야 합니다. **+** 기호와 숫자만 전화 번호에 포함할 수 있습니다. 서비스에 값을 제출하기 전에 전화 번호에서 괄호, 공백 또는 대시(**-**)와 같은 다른 문자는 모두 제거합니다. 예를 들어 미국 기반 전화 번호는 **+14325551212** 형식을 따라야 합니다.

preferred_username

`preferred_username`을 필수 또는 별칭으로 선택할 수 있지만 둘 다로 선택할 수는 없습니다. `preferred_username`이 별칭인 경우 UpdateUserAttributes API 작업을 요청하고 사용자 확인 후 속성 값을 추가할 수 있습니다.

sub

`sub` 속성을 기반으로 사용자를 인덱싱하고 검색합니다. `sub` 속성은 각 사용자 풀 내의 고유한 사용자 식별자입니다. 사용자는 `phone_number` 및 `email`와 같은 속성을 변경할 수 있습니다. `sub` 속성에는 고정 값이 있습니다. 사용자 검색에 대한 자세한 내용은 사용자 계정 관리 및 검색을 참조하십시오.

> **Important**
>
> Amazon Cognito는 RFC UUID`sub`를 포함한 특정 UUID 형식을 준수하지 않는 Amazon Cognito별 형식으로를 생성합니다. 형식을 엄격하게 검증해서는 안 됩니다`sub`.

필수 속성 보기

지정된 사용자 풀의 필수 속성을 보려면 다음 절차를 따르세요.

> **Note**
>
> 사용자 풀을 생성한 후에는 필수 속성을 변경할 수 없습니다.

필수 속성을 보려면

1. 에서 Amazon Cognito로 이동합니다 AWS Management Console. 콘솔에 메시지가 표시되면 자격 AWS 증명을 입력합니다.
2. 사용자 풀(User Pools)을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택합니다.
4. 가입 메뉴를 선택합니다.
5. 필수 속성(Required attributes) 섹션에서 사용자 풀의 필수 속성을 봅니다.

사용자 이름 및 기본 설정 사용자 이름

username 값은 별도의 속성이며 name 속성과 동일하지 않습니다. 각 사용자는 username 속성이 있습니다. Amazon Cognito는 페더레이션 사용자의 사용자 이름을 자동으로 생성합니다. Amazon Cognito 디렉터리에서 로컬 사용자를 생성하려면 username 속성을 제공해야 합니다. 사용자를 생성한 후에는 username 속성의 값을 변경할 수 없습니다.

개발자는 preferred_username 속성을 사용하여 사용자에게 변경할 수 있는 사용자 이름을 제공할 수 있습니다. 자세한 내용은 로그인 속성 사용자 지정 섹션을 참조하세요.

애플리케이션에 사용자 이름이 필요하지 않은 경우, 사용자에게 사용자 이름 제공을 요청할 필요가 없습니다. 앱이 백그라운드에서 고유한 사용자 이름을 생성할 수 있습니다. 이는 사용자가 이메일 주소와 암호를 사용하여 등록 및 로그인하게 하려는 경우 유용할 수 있습니다. 자세한 내용은 로그인 속성 사용자 지정 섹션을 참조하세요.

username은 사용자 풀에서 고유해야 합니다. username은 삭제되고 더 이상 사용되지 않는 경우에만 재사용될 수 있습니다. username 속성에 대한 문자열 제약 조건에 대한 자세한 내용은 SignUp API 요청의 사용자 이름 속성 섹션을 참조하세요.

로그인 속성 사용자 지정

사용자 풀을 생성할 때 사용자 이름 속성을 설정하여 사용자가 이메일 주소 또는 전화번호를 사용자 이름으로 사용하여 가입하고 로그인하게 할 수 있습니다. 또는 별칭 속성을 설정하여 사용자에게 선택지를 제공할 수도 있습니다. 사용자는 가입할 때 여러 속성을 포함한 다음 사용자 이름, 기본 설정 사용자 이름, 이메일 주소 또는 전화번호로 로그인할 수 있습니다.

> **Important**
>
> 사용자 풀이 생성된 후에는 이 설정을 변경할 수 없습니다.

별칭 속성과 사용자 이름 속성 중에서 선택하는 방법

사용자의 요구 사항	별칭 속성	사용자 이름 속성
Users have multiple sign-in attributes	Yes[1]	No[2]
Users must verify email address or phone number before they can sign in with it	Yes	No
Sign up users with duplicate email addresses or phone numbers and prevent `UsernameExistsException` errors[3]	Yes	No
Can assign the same email address or phone number attribute value to more than one user	Yes[4]	No

[1] 사용 가능한 로그인 속성은 사용자 이름, 이메일 주소, 전화번호 및 기본 설정 사용자 이름입니다.

[2] 이메일 주소나 전화번호로 로그인할 수 있습니다.

3 사용자가 잠재적으로 중복될 수 있는 이메일 주소 또는 전화번호로 등록할 때 사용자 풀은 `UsernameExistsException` 오류를 생성하지 않지만 사용자 이름은 생성하지 않습니다. 이 동작은 사용자 이름 존재 오류 방지와는 별개로, 로그인 작업에는 적용되지만 가입 작업에는 적용되지 않습니다.

[3] 속성을 검증한 마지막 사용자만 해당 속성을 사용하여 로그인할 수 있습니다.

옵션 1: 다중 로그인 속성(별칭 속성)

속성은 사용자에게 사용자 이름이 있지만 해당 속성으로 로그인할 수도 있는 경우 별칭입니다. 사용자가 로그인 양식의 사용자 이름 필드에서 사용자 이름과 다른 속성 값 중에서 선택할 수 있도록 허용하려면 별칭을 설정합니다. `username` 속성은 사용자가 변경할 수 없는 고정 값입니다. 속성을 별칭으로 표시하면 사용자가 사용자 이름 대신 해당 속성을 사용하여 로그인할 수 있습니다. 이메일 주소, 전화

번호 및 기본 설정 사용자 이름 속성을 별칭으로 표시할 수 있습니다. 예를 들어 이메일 주소 및 전화번호를 사용자 풀 별칭으로 선택하는 경우 해당 사용자 풀의 사용자는 자신의 사용자 이름, 이메일 주소 또는 전화번호를 암호와 함께 사용하여 로그인할 수 있습니다.

별칭 특성을 선택하려면 사용자 풀을 만들 때 User name(사용자 이름)을 선택하고 하나 이상의 추가 로그인 옵션을 선택합니다.

> **Note**
>
> 사용자 풀을 대/소문자를 구분하지 않도록 구성하면 사용자가 소문자 또는 대문자를 사용하여 자신의 별칭으로 가입하거나 로그인할 수 있습니다. 자세한 내용은 Amazon Cognito 사용자 풀 API 참조에서 CreateUserPool을 참조하세요.

이메일 주소를 별칭으로 선택하는 경우 Amazon Cognito는 유효한 이메일 주소 형식과 일치하는 사용자 이름을 수락하지 않습니다. 마찬가지로 전화번호를 별칭으로 선택하는 경우 Amazon Cognito는 유효한 전화번호 형식과 일치하는 해당 사용자 풀에 대한 사용자 이름을 수락하지 않습니다.

> **Note**
>
> 별칭 값은 사용자 풀에서 고유해야 합니다. 별칭을 이메일 주소 또는 전화 번호용으로 구성하는 경우 제공하는 값은 하나의 계정에서만 확인됨 상태가 될 수 있습니다. 가입하는 동안 사용자가 이메일 주소 또는 전화 번호를 별칭 값으로 제공하고 다른 사용자가 이미 해당 별칭 값을 사용한 경우 등록이 성공합니다. 그러나 사용자가 이 이메일(또는 전화 번호)로 계정 확인을 시도하고 유효한 코드를 입력하면 Amazon Cognito에서 `AliasExistsException` 오류를 반환합니다. 이 오류는 이 이메일 주소(또는 전화 번호)의 계정이 이미 있음을 사용자에게 나타냅니다. 이때 사용자는 새 계정 생성 시도를 중단하고 대신 이전 계정의 암호 재설정을 시도할 수 있습니다. 사용자가 계속해서 새 계정을 생성하는 경우 앱은 `forceAliasCreation` 옵션을 사용하여 `ConfirmSignUp` API를 호출해야 합니다. `ConfirmSignUp`과 `forceAliasCreation`은 함께 사용하여 이전 계정의 별칭을 새로 생성된 계정으로 이동하고 속성을 이전 계정에서 확인되지 않음으로 표시합니다.
> 사용자가를 사용하여 속성을 확인할 때 계정 간에 별칭을 전송할 수도 있습니다`VerifyUserAttribute`. 사용자가 이미 다른 계정에서 확인된 별칭인 이메일 주소 또는 전화번호를 확인하는 경우 Amazon Cognito는 별칭을 자신의 계정으로 전송하고 원래 계정에서 속성을 확인되지 않은 것으로 표시합니다. 이 전송은를 발생시키지 않습니다`AliasExistsException`.

전화 번호 및 이메일 주소는 사용자가 전화 번호 및 이메일 주소를 확인한 후에만 사용자에 대한 활성 별칭이 됩니다. 이메일 주소 및 전화 번호를 별칭으로 사용하는 경우 해당 이메일 주소 및 전화 번호의 자동 확인을 선택하는 것이 좋습니다.

사용자가 가입할 때 이메일 주소 및 전화번호 속성에 `UsernameExistsException` 오류가 발생하지 않도록, 별칭 속성을 선택하세요.

사용자의 `username` 속성 값이 변경되지 않는 동안 사용자가 로그인할 때 사용하는 사용자 이름을 사용자가 변경할 수 있도록 `preferred_username` 속성을 활성화합니다. 이 사용자 경험을 설정하려는 경우 새 `username` 값을 `preferred_username`으로 제출하고 `preferred_username`을 별칭으로 선택합니다. 그러면 사용자가 자신이 입력한 새 값으로 로그인할 수 있습니다. `preferred_username`을 별칭으로 선택하는 경우 사용자는 계정을 확인할 때에만 값을 제공할 수 있습니다. 등록 중에는 값을 제공할 수 없습니다.

사용자가 사용자 이름을 사용하여 가입할 때, 다음 별칭 중 하나 이상을 사용하여 로그인할 수 있는지 여부를 선택할 수 있습니다.

- 확인된 이메일 주소
- 확인된 전화 번호
- 기본 설정 사용자 이름

사용자가 가입한 후에 이러한 별칭을 변경할 수 있습니다.

> **Important**
>
> 사용자 풀에서 별칭을 사용한 로그인을 지원하고 사용자를 인증하거나 검색하려는 경우 로그인 속성으로 사용자를 식별하지 마십시오. 고정 값 사용자 식별자 `sub`은 사용자 ID를 나타내는 유일한 일관된 지표입니다.

사용자가 별칭을 사용하여 로그인할 수 있도록 사용자 풀을 생성할 때 다음 단계를 포함합니다.

Phone number or email address (console)

사용자 풀을 생성할 때 이메일 주소와 전화번호를 별칭 속성으로 설정해야 합니다.

Amazon Cognito 콘솔에서 사용자 이름 별칭이 있는 사용자 풀을 생성하려면

1. AWS Management Console의 Amazon Cognito로 이동합니다. 콘솔에 메시지가 표시되면 자격 AWS 증명을 입력합니다.
2. 시작하기 또는 사용자 풀 생성 버튼을 사용하여 새 사용자 풀을 생성합니다.
3. 애플리케이션 정의에서 애플리케이션 설정을 선택합니다.
4. 로그인 식별자 옵션의 옵션 구성에서 사용자 이름 옆의 확인란과 이메일 및 전화번호 중 하나 이상을 선택합니다.
5. 별칭 속성을 가입에 필요한 속성으로 선택합니다. 관리형 로그인 가입 양식에서 Amazon Cognito는 새 사용자에게 필수 속성 값을 제공하라는 메시지를 표시합니다.
6. 반환 URL 추가에서 관리형 로그인 후 리디렉션할 애플리케이션 콜백 URL을 설정합니다.
7. 생성을 선택합니다.

Phone number or email address (API/SDK)

CreateUserPool API 작업을 사용하여 새 사용자 풀을 생성합니다. 표시된 대로 `AliasAttributes` 파라미터를 구성합니다. 전화번호 별칭만 원하는 경우 `email` 항목을 제거하거나 이메일 주소 별칭만 원하는 경우 `phone_number` 항목을 제거할 수 있습니다.

```
"AliasAttributes": [
   "email",
   "phone_number"
],
```

Preferred username (API/SDK)

Amazon Cognito 콘솔은 `preferred_username`을 별칭으로 사용하지 않고 사용자 풀을 생성합니다. `preferred_username` 별칭이 있는 사용자 풀을 생성하려면 AWS SDK에서 CreateUserPool API 요청을 사용하여 사용자 풀을 설정합니다. 가입 시 기본 사용자 이름 속성 생성을 지원하려면 `preferred_username`을 필수 속성으로 설정합니다. 관리형 로그인 가입 양식에서 Amazon Cognito는 새 사용자에게 필수 속성 값을 제공하라는 메시지를 표시합니다. Amazon Cognito 콘솔에서 `preferred_username`을 필수 속성으로 설정할 수 있지만 별칭으로 사용할 수는 없습니다.

별칭으로 구성

그림과 같이 `CreateUserPool` 요청의 `AliasAttributes` 파라미터에서 별칭으로 `preferred_username`을 구성합니다. 목록에서 별칭 속성으로 사용하지 않으려는 값을 제거합니다.

```
"AliasAttributes": [
   "email",
   "phone_number",
   "preferred_username"
],
```

필요에 따라 구성

관리형 로그인 가입 양식에서 Amazon Cognito는 새 사용자에게 필수 속성 값을 제공하라는 메시지를 표시합니다. CreateUserPool 요청의 SchemaAttributes 파라미터에서 필요에 따라 preferred_username을 구성합니다.

기본 사용자 이름을 필수 속성으로 설정하려면 다음과 같이 구성합니다. 다음 예시에서는 preferred_username의 기본 스키마를 수정하여 필요에 따라 설정합니다. AttributeDataType(기본값은 string) 및 StringAttributeConstraints(기본값은 1~99자)와 같은 다른 스키마 파라미터는 기본값을 가정합니다.

```
"Schema": [
   {
      "Name": "preferred_username",
      "Required": true
   }
]
```

옵션 2: 로그인 속성으로 사용하는 이메일 주소 또는 전화번호(사용자 이름 속성)

사용자가 이메일 주소 또는 전화번호를 사용자 이름으로 사용하여 가입할 때 사용자가 이메일 주소와 전화번호 중 하나만 사용하여 가입할 수 있는지 여부를 선택할 수 있습니다.

사용자 이름 특성을 선택하려면 사용자 풀을 만들 때 사용자 이름을 로그인 옵션으로 선택하지 마세요.

이메일 주소 또는 전화 번호는 고유해야 하며 다른 사용자가 이미 사용하고 있어서는 안 됩니다. 확인을 받을 필요는 없습니다. 사용자가 이메일 주소 또는 전화 번호를 사용하여 가입한 후에는 동일한 이메일 주소 또는 전화 번호로 새 계정을 생성할 수 없습니다. 사용자는 기존 계정을 재사용하고 필요한 경우 계정 암호를 재설정할 수만 있습니다. 그러나 사용자는 이메일 주소 또는 전화 번호를 새 이메일 주소 또는 전화 번호로 변경할 수 있습니다. 이메일 주소 또는 전화번호가 아직 사용 중이 아닌 경우 새 사용자 이름이 됩니다.

이메일 주소와 전화번호를 모두 사용자 이름 속성으로 선택하면 두 속성에 대한 값을 제공하더라도 사용자가 둘 중 하나로 로그인할 수 있습니다. 로그인 사용자 이름은 SignUp의 `Username` 파라미터에서 전달하는 값을 기반으로 합니다.

> **Note**
>
> 사용자가 이메일 주소를 사용자 이름으로 사용하여 가입하는 경우 사용자 이름을 다른 이메일 주소로 변경할 수 있으나 전화번호로 변경할 수는 없습니다. 사용자가 전화번호를 사용하여 가입하는 경우 사용자 이름을 다른 전화번호로 변경할 수 있지만 이메일 주소로 변경할 수는 없습니다.

이메일 주소 또는 전화 번호를 사용하는 가입 및 로그인을 설정하려면 사용자 풀 생성 프로세스 동안 다음 단계를 사용합니다.

Username attributes (console)

다음 절차에서는 이메일 주소 또는 전화번호 사용자 이름 속성을 사용하여 사용자 풀을 생성합니다. Amazon Cognito 콘솔에서 사용자 이름 속성 프로세스의 차이점은 사용자 이름을 로그인 속성으로 설정하지 않는다는 것입니다.

Amazon Cognito 콘솔에서 사용자 이름 속성을 사용하여 사용자 풀을 생성하려면

1. AWS Management Console의 Amazon Cognito로 이동합니다. 콘솔에 메시지가 표시되면 자격 AWS 증명을 입력합니다.
2. 시작하기 또는 사용자 풀 생성 버튼을 사용하여 새 사용자 풀을 생성합니다.
3. 애플리케이션 정의에서 애플리케이션 설정을 선택합니다.
4. 로그인 식별자 옵션의 옵션 구성에서 이메일, 전화번호 또는 둘 다와 같은 사용자 이름 속성을 선택합니다. 사용자 이름을 선택하지 않은 상태로 둡니다.
5. 가장 좋은 방법은 사용자 이름 속성을 가입에 필요한 속성으로 선택하는 것입니다. 관리형 로그인 가입 양식에서 Amazon Cognito는 새 사용자에게 필수 속성 값을 제공하라는 메시지를 표시합니다. 필요에 따라 사용자 이름 속성을 설정하지 않으면 Amazon Cognito는 새 사용자에게 해당 속성을 입력하라는 메시지를 표시하지 않습니다. 이 시나리오에서는 사용자가 로그인하기 전에 각 사용자의 이메일 주소 또는 전화번호를 수집하고 제출하도록 애플리케이션을 구성해야 합니다.
6. 반환 URL 추가에서 관리형 로그인 후 리디렉션할 애플리케이션 콜백 URL을 설정합니다.
7. 생성을 선택합니다.

Username attributes (API/SDK)

CreateUserPool 요청에서 표시된 대로 UsernameAttributes 파라미터를 구성합니다. 이메일 주소 사용자 이름으로만 로그인을 허용하려면 이 목록에 email만 지정합니다. 전화번호 사용자 이름으로만 로그인을 허용하려면 phone_number만 지정합니다. 이 파라미터는 로그인 옵션으로 사용자 이름을 재정의합니다.

```
"UsernameAttributes": [
   "email",
   "phone_number"
],
```

다음 예제에서는 각 사용자가 이메일 주소만 사용자 이름으로 사용하여 로그인하는 사용자 풀을 구성합니다. 또한 관리형 로그인을 통해 가입할 때 새 사용자가 이메일 주소를 제공하도록를 필수 가입 속성email으로 설정하는 것이 좋습니다.

```
"UsernameAttributes": [
   "email"
],
```

사용자 이름 속성을 구성할 때 username 파라미터에서 이메일 주소 또는 전화번호를 전달하는 SignUp API 요청을 할 수 있습니다. 다음은 사용자 이름 속성을 사용하는 SignUp API 작업의 동작입니다.

- username 문자열이 유효한 이메일 주소 형식(예: user@example.com)인 경우 사용자 풀은 자동으로 사용자 email 속성을 username 값으로 채웁니다.
- username 문자열이 유효한 전화 번호 형식(예: +12065551212)인 경우 사용자 풀은 자동으로 사용자 phone_number 속성을 username 값으로 채웁니다.
- username 문자열 형식이 이메일 주소 또는 전화 번호 형식이 아닌 경우 SignUp API에서 예외를 반환합니다.
- username 문자열에 이미 사용 중인 이메일 주소 또는 전화 번호가 포함되어 있는 경우 SignUp API에서 예외가 발생합니다.
- SignUp API는 사용자에 대한 UUID로 username 속성을 채웁니다. 이 UUID는 사용자 ID 토큰의 sub 클레임과 동일한 값을 갖습니다.

ListUsers 작업을 제외한 모든 API에서 사용자 이름 대신 이메일 주소나 전화번호를 사용할 수 있습니다. 여기에는 ConfirmSignUp 및 ResendConfirmationCode와 같은 post-sign-up 작업이 포함됩

니다. 가 username 속성을 UUID로 SignUp 설정하더라도 사용자의 이메일 주소 또는 전화 번호를 SignUp 요청의 Username 값으로 전달할 수 있습니다. ListUsers API 요청에서 email 또는 phone_number의 Filter를 지정할 수 있습니다. 를 기준으로 필터링하려면 UUID 사용자 이름을 username입력합니다. username 필터에서 이메일 주소 또는 전화 번호를 기준으로 필터링할 수 없습니다.

사용자 지정 속성

사용자 풀에 사용자 정의 속성을 최대 50개까지 추가할 수 있습니다. 사용자 지정 속성에 대해 최소 길이, 최대 길이 또는 둘 다를 지정할 수 있습니다. 단, 사용자 지정 속성의 최대 길이는 2,048자를 넘을 수 없습니다. 사용자 지정 속성의 이름은 SchemaAttributeType의 Name 파라미터에 설명된 정규식 패턴과 일치해야 합니다.

> ⚠ 사용자 지정 속성은 영구적입니다.
>
> 사용자 풀에 추가한 후에는 사용자 지정 속성을 제거하거나 변경할 수 없습니다. 속성을 삭제하거나 이름을 바꿀 수 없습니다. 또한 데이터 유형, 변경 가능성 또는 길이 제약 조건을 변경할 수 없습니다. 프로덕션 사용자 풀에 속성을 추가하기 전에 사용자 지정 속성 스키마를 계획합니다.

각 사용자 지정 속성의 특성은 다음과 같습니다.

- 문자열, 숫자, 부울 또는 DateTime 객체로 정의할 수 있습니다. Amazon Cognito는 사용자 지정 속성 값을 ID 토큰에 문자열로만 씁니다.

> **Note**
>
> Amazon Cognito 콘솔에서는 문자열 및 숫자 데이터 유형의 사용자 지정 속성만 추가할 수 있습니다. 부울 및 DateTime 속성 데이터 유형과 같은 추가 옵션은 CreateUserPool 및 UpdateUserPool API 요청의 SchemaAttributes 속성에서만 사용할 수 있습니다.

> **Note**
>
> 할당하는 데이터 유형에 관계없이 Amazon Cognito는 모든 사용자 지정 속성 값을 문자열로 저장하고 전송합니다. AdminUpdateUserAttributes, UpdateUserAttributes 또는 SignUp과 같은 API 또는 SDK 요청에서 값을 설정하는 경우 , Number Boolean또는 DateTime 속성에

대해서도 값을 문자열로 전달해야 합니다. "42" 대신 42를 전달하고 "true" 대신를 전달합니다true. 값이 저장되기 전에 기본 숫자 또는 부울이 거부됩니다. SDK에 따라이 값은 클라이언트 측 유형 검증 오류 또는 서비스로 표시됩니다InvalidParameterException. 애플리케이션은 속성을 읽을 때 문자열을 의도한 유형으로 다시 구문 분석할 책임이 있습니다.

- 사용자에게 이 속성의 값을 제공하도록 요구할 수 없습니다.
- 이 속성을 사용자 풀에 추가한 후에는 제거하거나 변경할 수 없습니다.
- 이 속성 이름의 문자 길이는 Amazon Cognito가 허용하는 한도 내에 있습니다. 자세한 내용은 Amazon Cognito의 할당량 섹션을 참조하세요.
- 이 속성은 변경 가능 또는 변경 불가능일 수 있습니다. 사용자를 생성할 때만 변경 불가능한 사용자 지정 속성에 값을 쓸 수 있습니다. 앱 클라이언트에 속성에 대한 쓰기 권한이 있는 경우 변경 가능한 속성의 값을 변경할 수 있습니다. 자세한 내용은 속성 권한 및 범위 섹션을 참조하세요.

> **Note**
>
> 코드에서 그리고 역할 기반 액세스 제어 사용에 대한 규칙 설정에서는 사용자 지정 속성을 표준 속성과 구별하기 위해 사용자 지정 속성에 custom: 접두사가 필요합니다.

사용자 풀을 생성할 때 CreateUserPool의 SchemaAttributes 속성에 개발자 속성을 추가할 수도 있습니다. 개발자 속성에는 dev: 접두사가 있습니다. 자격 AWS 증명으로만 사용자의 개발자 속성을 수정할 수 있습니다. 개발자 속성은 Amazon Cognito가 앱 클라이언트 읽기-쓰기 권한으로 대체한 레거시 특성입니다.

새 사용자 정의 속성을 생성하려면 다음 절차를 따르세요.

콘솔을 사용하여 사용자 지정 속성을 추가하려면

1. 에서 Amazon Cognito로 이동합니다 AWS Management Console. 콘솔에 메시지가 표시되면 자격 AWS 증명을 입력합니다.
2. 사용자 풀(User Pools)을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택합니다.
4. 가입 탭을 선택하고 사용자 지정 속성 섹션에서 사용자 지정 속성 추가를 선택합니다.
5. [사용자 정의 속성 추가(Add custom attributes)] 페이지에서 새 속성에 대한 다음 세부 정보를 제공합니다.

- 이름을 입력합니다.
- 문자열 또는 숫자의 형식을 선택합니다.
- 최소 문자열 길이 또는 숫자 값을 입력합니다.
- 최대 문자열 길이 또는 숫자 값을 입력합니다.
- 사용자에게 사용자 지정 속성의 초기 값을 설정한 후 해당 값을 변경할 수 있는 권한을 제공하려는 경우 변경 가능(Mutable)을 선택합니다.

6. 변경 사항 저장(Save changes)을 선택합니다.

속성 권한 및 범위

각 앱 클라이언트에 대해 각 사용자 속성의 읽기 및 쓰기 권한을 설정할 수 있습니다. 이렇게 하면 모든 앱이 사용자를 위해 저장되는 각 속성을 읽고 수정하기 위해 갖고 있어야 할 액세스 권한을 제어할 수 있습니다. 예를 들어, 사용자가 유료 고객인지 여부를 나타내는 사용자 지정 속성이 있을 수 있습니다. 앱에서 이 속성을 볼 수 있지만 직접 변경할 수는 없습니다. 대신 관리 도구 또는 백그라운드 프로세스를 사용하여 이 속성을 업데이트합니다. Amazon Cognito 콘솔, API 또는 AWS CLI에서 사용자 속성에 대한 권한을 설정할 수 있습니다. 기본적으로 새 사용자 지정 속성은 해당 속성에 대한 읽기 및 쓰기 권한이 설정된 후에만 사용할 수 있습니다. 기본적으로 새 앱 클라이언트를 만들 때 모든 표준 및 사용자 지정 특성에 대한 읽기 및 쓰기 권한을 앱에 부여합니다. 앱이 필요한 양의 정보만 사용하도록 제한하려면 앱 클라이언트 구성의 속성에 특정 권한을 할당합니다.

앱 클라이언트를 생성할 때 속성 읽기 및 쓰기 권한을 지정하는 것이 가장 좋습니다. 앱 클라이언트에 애플리케이션 작업에 필요한 최소 사용자 속성 집합에 대한 액세스 권한을 부여합니다.

> **Important**
>
> 앱 클라이언트`WriteAttributes`에를 지정하지 않으면 앱 클라이언트는 모든 표준 및 사용자 지정 속성에 대한 쓰기 액세스 권한을 갖습니다. 사용자는를 사용하여 이러한 속성의 값을 변경할 수 있습니다`UpdateUserAttributes`.
>
> 애플리케이션 또는 자격 증명 풀은 속성을 기반으로 권한 부여 결정을 내릴 수 있습니다. 이 경우 사용자 풀에 있는 모든 앱 클라이언트`WriteAttributes`의에서 속성을 제거합니다. 사용자 속성은 사용자 풀의 모든 앱 클라이언트에서 공유됩니다. 사용자가 하나의 앱 클라이언트를 통해 쓰는 값은 모든 앱 클라이언트가 발급하는 토큰에 표시됩니다.

> **Note**
>
> DescriptionUserPoolClient는 기본값이 아닌 앱 클라이언트 권한을 구성하는 경우에만 `ReadAttributes` 및 `WriteAttributes` 값을 반환합니다.

속성 권한을 업데이트하려면(AWS Management Console)

1. 에서 Amazon Cognito로 이동합니다 AWS Management Console. 콘솔에 메시지가 표시되면 자격 AWS 증명을 입력합니다.
2. 사용자 풀(User Pools)을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택합니다.
4. 앱 클라이언트 메뉴를 선택하고 목록에서 앱 클라이언트를 선택합니다.
5. 속성 권한 탭에서 편집을 선택합니다.
6. 속성 읽기 및 쓰기 권한 편집(Edit attribute read and write permissions) 페이지에서 읽기 및 쓰기 권한을 구성한 다음 변경 사항 저장(Save changes)을 선택합니다.

사용자 지정 속성을 사용하는 각 앱 클라이언트에 대해 이러한 단계를 반복합니다. 앱에서 속성을 읽기 전용으로 만들려면 사용자 풀의 모든 앱 클라이언트에서 해당 속성에 대한 쓰기 권한을 제거합니다.

각 앱 클라이언트에 대해 속성을 읽기 가능 또는 쓰기 가능으로 표시할 수 있습니다. 표준 속성과 사용자 정의 속성에 모두 적용됩니다. 앱에서 읽기 가능으로 표시한 속성 값을 검색하고 쓰기 가능으로 표시한 속성 값을 설정하거나 수정할 수 있습니다. 앱이 쓰기 권한이 없는 속성에 값을 설정하려고 하면 Amazon Cognito가 `NotAuthorizedException`을 반환합니다. GetUser요청에는 앱 클라이언트 클레임이 있는 액세스 토큰이 포함됩니다. Amazon Cognito는 앱 클라이언트가 읽을 수 있는 속성 값만 반환합니다. 앱의 사용자 ID 토큰에는 읽기 가능한 속성에 해당하는 클레임만 포함됩니다. 모든 앱 클라이언트는 사용자 풀에 필요한 속성을 작성할 수 있습니다. 아직 값이 없는 필수 속성에도 값을 제공하는 경우에만 Amazon Cognito 사용자 풀 API 요청에서 속성 값을 설정할 수 있습니다.

사용자 지정 속성에는 읽기 및 쓰기 권한에 대한 고유한 기능이 있습니다. 사용자 풀에 대해 변경 가능 또는 변경 불가능으로 만들 수 있으며 모든 앱 클라이언트에 대해 읽기 또는 쓰기 속성으로 설정할 수 있습니다.

변경 불가능한 사용자 지정 속성은 사용자 생성 중에 한 번 업데이트할 수 있습니다. 다음 방법으로 변경 불가능한 속성을 채울 수 있습니다.

- SignUp: 사용자가 변경 불가능한 사용자 지정 속성에 대한 쓰기 권한이 있는 앱 클라이언트에 가입합니다. 해당 속성에 대한 값을 제공합니다.
- 서드 파티 IdP로 로그인: 사용자가 변경 불가능한 사용자 지정 속성에 대한 쓰기 권한이 있는 앱 클라이언트에 로그인합니다. 해당 IdP의 사용자 풀 구성에는 제공된 클레임을 변경 불가능한 속성에 매핑하는 규칙이 있습니다. 사용자가 한 번만 로그인할 수 있기 때문에 가능하지만 실용적이지는 않습니다. 처음 로그인한 후 로그인을 시도하면 Amazon Cognito는 현재 쓰기 불가능한 속성에 대한 매핑 규칙으로 인해 시도를 거부합니다.
- AdminCreateUser: 변경 불가능한 속성에 값을 입력합니다.

범위가 있는 속성 권한

AWS SDK 또는 CDK, REST API 또는 로 구성하는 사용자 풀에서 OIDC 범위를 사용하여 앱 클라이언트 읽기 또는 쓰기 액세스를 구성할 AWS CLI수 있습니다oidc:profile. oidc:profile은 다음 표준 속성에 대한 읽기 또는 쓰기 액세스 권한을 부여합니다.

- name
- family_name
- given_name
- middle_name
- nickname
- preferred_username
- profile
- picture
- website
- gender
- birthdate
- zoneinfo
- locale

이 목록은 OIDC 사양의 2.4 섹션에 정의된 대로 email, phone_number, sub 및 address를 뺀 OIDC 표준 속성입니다. 앱 클라이언트에 할당할 수 있는 범위에 대한 자세한 내용은 범위, M2M 및 리소스 서버 섹션을 참조하세요.

`oidc:profile` 범위 아래의 속성에 쓰도록 앱 클라이언트를 구성하려면 CreateUserPoolClient 또는 UpdateUserPoolClient API 요청에서 WriteAttributes 값을 `oidc:profile`로 설정하고 애플리케이션이 수정하도록 허용할 다른 속성을 설정합니다. 마찬가지로 이러한 속성에 대한 읽기 액세스 권한을 부여하려면 ReadAttributes 값에 `oidc:profile`을 추가합니다.

사용자 풀을 생성한 후 속성 권한과 범위를 변경할 수 있습니다.

사용자 풀 및 앱 클라이언트 구성 업데이트

사용자 풀 또는 앱 클라이언트에서 설정을 변경하려는 경우 몇 번의 클릭만으로 Amazon Cognito 콘솔에서 업데이트를 적용할 수 있습니다. 사용자 풀 설정에서 기능 기반 탭을 탐색하고 이 안내서의 다른 부분에 설명된 대로 필드를 업데이트합니다.

많은 조직이에서 프로그래밍 방식으로 리소스 AWS CloudFormation, AWS SDKs 또는 CDK 기반 애플리케이션 및 기타 자동화 소프트웨어를 관리합니다. 이것이 리소스 관리 모델인 경우 리소스에 대한 변경을 단계적으로 수행할 때 각별히 주의해야 합니다.

API 작업 UpdateUserPool 및 UpdateUserPoolClient는 기존 사용자 풀 또는 앱 클라이언트를 업데이트합니다. API 참조에 속성에 값을 제공하지 않으면 Amazon Cognito가 기본값으로 설정합니다라는 경고가 함께 제공됩니다. 파라미터가 하나뿐인 업데이트 요청을 제출하면 Amazon Cognito는 해당 파라미터를 선택한 값으로 설정하고 다른 모든 파라미터를 기본값으로 설정합니다. 이렇게 하면 속성 스키마, Lambda 트리거, 이메일 및 SMS 메시지 구성을 포함한 구성을 재설정할 수 있습니다.

또한 일부 설정은 사용자 풀 또는 앱 클라이언트를 생성한 후 잠기며 새 리소스를 생성하지 않는 한 변경할 수 없습니다.

주제

- 변경할 수 없는 설정
- SMS 구성
- AWS SDK AWS CDK또는 REST API를 사용하여 사용자 풀 업데이트

변경할 수 없는 설정

사용자 풀이 생성된 후에는 일부 설정을 변경할 수 없습니다. 다음 설정을 변경하려면 새 사용자 풀 또는 앱 클라이언트를 생성해야 합니다.

> **Note**
>
> 이전에는 사용자 풀의 이름을 변경할 수 없었습니다. 이 기능이 변경되었습니다. 이제 사용자 풀에 새 표시 이름을 할당할 수 있습니다.

사용자 풀 ID

API 파라미터 이름: Id/UserPoolId

예를 들어 `us-east-1_EXAMPLE`과 같은 사용자 풀 ID는 Amazon Cognito에서 자동으로 생성되며 변경할 수 없습니다.

Amazon Cognito 사용자 풀 로그인 옵션

API 파라미터 이름: AliasAttributes 및 UsernameAttributes

사용자가 로그인할 때 사용자 이름으로 전달할 수 있는 속성입니다. 사용자 풀을 생성할 때 사용자 이름, 이메일 주소, 전화번호 또는 선호하는 사용자 이름으로 로그인할 수 있도록 선택할 수 있습니다. 사용자 풀 로그인 옵션을 변경하려면 새 사용자 풀을 생성합니다.

사용자 이름의 대/소문자 구분(Make user name case sensitive)

API 파라미터 이름: UsernameConfiguration

문자 대소문자를 제외한 다른 사용자 이름과 일치하는 사용자 이름을 생성하면 Amazon Cognito는 해당 사용자 이름을 동일한 사용자 또는 고유한 사용자로 취급할 수 있습니다. 자세한 내용은 사용자 풀 대/소문자 구분 섹션을 참조하세요. 대소문자 구분을 변경하려면 새 사용자 풀을 생성합니다.

클라이언트 보안 암호

API 파라미터 이름: GenerateSecret

앱 클라이언트를 만들 때 신뢰할 수 있는 소스만 사용자 풀에 요청하도록 클라이언트 보안 암호를 생성할 수 있습니다. 자세한 내용은 앱 클라이언트를 사용한 애플리케이션별 설정 섹션을 참조하세요. 클라이언트 보안 암호를 변경하려면 동일한 사용자 풀에 새 앱 클라이언트를 생성합니다.

필수 속성

API 파라미터 이름: 스키마

사용자가 가입할 때 또는 사용자를 생성할 때 값을 제공해야 하는 속성입니다. 자세한 내용은 사용자 속성 작업 섹션을 참조하세요. 필수 속성을 변경하려면 새 사용자 풀을 생성합니다.

사용자 지정 속성(삭제)

API 파라미터 이름: 스키마

사용자 지정 이름이 있는 속성입니다. 사용자의 사용자 지정 속성 값은 변경할 수 있지만, 사용자 풀에서 사용자 지정 속성을 삭제할 수는 없습니다. 자세한 내용은 사용자 속성 작업 섹션을 참조하세요. 최대 사용자 지정 속성 수에 도달하고 목록을 수정하려면 새 사용자 풀을 생성합니다.

SMS 구성

사용자 풀에서 SMS 메시지를 활성화한 후에는 비활성화할 수 없습니다.

- 사용자 풀을 생성할 때 SMS 메시지를 구성하도록 선택한 경우 설정을 완료한 후에는 SMS를 비활성화할 수 없습니다.
- 생성한 사용자 풀에서 SMS 메시지를 활성화할 수 있지만, 이후에는 SMS를 비활성화할 수 없습니다.
- Amazon Cognito는 사용자 계정 초대 및 복구, 속성 확인 및 다중 인증(MFA)에 SMS 메시지를 사용할 수 있습니다. SMS 메시지를 활성화한 후에는 언제든지 이러한 함수에 대해 SMS 메시지를 켜거나 끌 수 있습니다.
- SMS 메시지 구성에는 Amazon SNS를 통해 메시지를 전송하기 위해 Amazon Cognito에 위임하는 IAM 역할이 포함되어 있습니다. 언제든지 할당된 역할을 변경할 수 있습니다.

AWS SDK AWS CDK또는 REST API를 사용하여 사용자 풀 업데이트

Amazon Cognito 콘솔에서 사용자 풀 설정을 한 번에 하나의 파라미터로 변경할 수 있습니다. 예를 들어 Lambda 트리거를 추가하려면 Lambda 트리거 추가를 선택하고 함수와 트리거 유형을 선택합니다. Amazon Cognito 사용자 풀 API는 사용자 풀 및 앱 클라이언트에 대한 업데이트 작업에 사용자 풀에 대한 전체 파라미터 세트가 필요한 방식으로 구성됩니다. 하지만 콘솔은 다른 사용자 풀 설정으로 이 업데이트 작업을 투명하게 자동화합니다.

의 다른 위치에서 변경하면 변경하려는 설정과 관련이 없을 때 업데이트에 오류가 발생할 AWS 계정 수 있습니다. AWS WAF예를 들어 삭제된 Amazon SES 자격 증명 또는에 대한 IAM 권한 변경. 현재 파라미터 중 하나가 더 이상 유효하지 않으면 설정을 수정할 때까지 설정을 업데이트할 수 없습니다. 이러한 오류가 발생하면 오류 응답을 검사하고 언급된 설정을 검증합니다.

AWS Cloud Development Kit (AWS CDK), Amazon Cognito 사용자 풀 REST API 및 AWS SDK는 Amazon Cognito 리소스의 자동화 및 프로그래밍 구성 도구입니다. Amazon Cognito 콘솔과 같이 이러

한 도구를 사용하는 요청도 요청 본문의 전체 리소스 구성으로 설정을 업데이트해야 합니다. 높은 수준에서 다음 프로세스를 수행해야 합니다.

1. 기존 리소스의 구성을 설명하는 작업에서 출력을 캡처합니다.
2. 설정이 변경되면 출력을 수정합니다.
3. 리소스를 업데이트하는 작업에서 수정된 구성을 전송합니다.

다음 절차에서는 UpdateUserPool API 작업을 사용하여 구성을 업데이트합니다. 다른 입력 필드를 사용하는 동일한 접근 방식이 UpdateUserPoolClient에 적용됩니다.

> **Important**
>
> 기존 파라미터에 대한 값을 입력하지 않으면 Amazon Cognito에 이 값이 기본값으로 설정됩니다. 예를 들어 기존 `LambdaConfig`가 있고 `LambdaConfig`를 비어 있는 상태로 `UpdateUserPool`을 제출하면 사용자 풀 트리거에 대한 모든 Lambda 함수 할당이 삭제됩니다. 사용자 풀 구성 변경을 자동화하려는 경우 적절하게 계획합니다.

1. 사용자 풀의 기존 상태를 DescribeUserPool로 캡처합니다.
2. `DescribeUserPool`의 출력 형식을 `UpdateUserPool`의 요청 파라미터와 일치하도록 지정합니다. 출력 JSON에서 다음 최상위 필드 및 해당 하위 객체를 제거합니다.
   - `Arn`
   - `CreationDate`
   - `CustomDomain`
     - 이 필드를 UpdateUserPoolDomain API 작업으로 업데이트합니다.
   - `Domain`
     - 이 필드를 UpdateUserPoolDomain API 작업으로 업데이트합니다.
   - `EmailConfigurationFailure`
   - `EstimatedNumberOfUsers`
   - `Id`
   - `LastModifiedDate`
   - `Name`
   - `SchemaAttributes`

- `SmsConfigurationFailure`
- `Status`

3. 결과로 나타나는 JSON이 `UpdateUserPool`의 요청 파라미터와 일치하는지 확인합니다.
4. 결과로 나타나는 JSON에서 변경하려는 파라미터를 수정합니다.
5. 수정된 JSON을 요청 입력으로 하여 `UpdateUserPool` API 요청을 제출합니다.

수정된 출력은 AWS CLI에서 `update-user-pool`의 `--cli-input-json` 파라미터에도 사용할 수 있습니다.

또는 다음 AWS CLI 명령을 실행하여에 허용되는 입력 필드에 빈 값이 있는 JSON을 생성합니다`update-user-pool`. 그런 다음 이 필드를 사용자 풀의 기존 값으로 채울 수 있습니다.

```
aws cognito-idp update-user-pool --generate-cli-skeleton --output json
```

다음 명령을 실행하여 앱 클라이언트에 동일한 JSON 객체를 생성합니다.

```
aws cognito-idp update-user-pool-client --generate-cli-skeleton --output json
```

사용자 풀 사용자 지정

Cognito 사용자 풀은 확장 지점을 노출하므로 애플리케이션에 대한 가입 및 로그인 환경을 구성할 수 있습니다. Lambda 트리거를 사용하여 사용자 수명 주기의 주요 순간에 자체 코드를 실행하고 사용자 풀이 보내는 메시지 및 분석을 구성합니다.

주제

- Lambda 트리거를 사용하여 사용자 풀 워크플로 사용자 정의
- 메시징 및 분석

Lambda 트리거를 사용하여 사용자 풀 워크플로 사용자 정의

Amazon Cognito는 AWS Lambda 함수와 함께 작동하여 사용자 풀의 인증 동작을 수정합니다. 처음 가입하기 전, 인증을 완료한 후, 그리고 그 사이의 여러 단계에서 Lambda 함수를 자동으로 호출하도록 사용자 풀을 구성할 수 있습니다. 함수는 인증 흐름의 기본 동작을 수정하고, API 요청을 통해 사용자 풀 또는 기타 AWS 리소스를 수정하고, 외부 시스템과 통신할 수 있습니다. Lambda 함수의 코드는 사용자 고유의 것입니다. Amazon Cognito는 이벤트 데이터를 함수로 전송하고, 함수가 데이터를 처리할 때까지 기다리며, 대부분의 경우 세션에 적용하려는 변경 사항이 반영된 응답 이벤트를 예상합니다.

요청 및 응답 이벤트 시스템 내에서 자체 인증 챌린지를 도입하고, 사용자 풀과 다른 자격 증명 스토어 간에 사용자를 마이그레이션하고, 메시지를 사용자 지정하고, JSON 웹 토큰(JWT)을 수정할 수 있습니다.

Lambda 트리거는 사용자가 사용자 풀에서 작업을 시작한 후 Amazon Cognito가 사용자에게 전달하는 응답을 사용자 지정할 수 있습니다. 예를 들어, 다른 방법으로는 성공할 수 있는 사용자의 로그인을 막을 수 있습니다. 또한 AWS 환경, 외부 API, 데이터베이스 또는 자격 증명 스토어에 대해 런타임 작업을 수행할 수도 있습니다. 예를 들어 사용자 마이그레이션 트리거는 외부 작업과 Amazon Cognito의 변경을 결합할 수 있습니다. 즉, 외부 디렉터리에서 사용자 정보를 조회한 다음 해당 외부 정보를 기반으로 새 사용자의 속성을 설정할 수 있습니다.

사용자 풀에 Lambda 트리거를 할당하면 Amazon Cognito는 기본 흐름을 중단하여 함수에서 정보를 요청합니다. Amazon Cognito는 JSON 이벤트를 생성하여 함수에 전달합니다. 해당 이벤트에는 사용자 계정 생성, 로그인, 암호 재설정 또는 속성 업데이트를 위한 사용자의 요청 관련 정보가 포함됩니다. 그러면 함수에서 조치를 취하거나 수정되지 않은 상태로 이벤트를 다시 보낼 수 있습니다. 수정되지 않은 상태로 반환된 이벤트는 사용자 풀에 이벤트의 기본 작업을 진행하도록 알립니다. 예를 들어 사전 가입 트리거는 `PreSignUp_SignUp` 트리거 소스의 사용자를 자동으로 확인할 수 있지만 외부 및 관리자가 생성한 사용자에 대해서는 변경되지 않은 이벤트를 반환할 수 있습니다.

다음 표에는 Lambda 트리거를 사용하여 사용자 풀 작업을 사용자 지정할 수 있는 방법 중 일부가 요약되어 있습니다.

사용자 풀 흐름	연산	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”	로그인 요청 승인 및 거절에 대한 사용자 지정 검증입니다.
	the section called “사후 인증”	사용자 지정 분석을 위한 이벤트 기록
	the section called “사전 토큰 생성”	토큰 신청 증강 또는 억제
연동	the section called “인바운드 페더레이션”	Amazon Cognito 사용자 풀에서 사용자를 생성하거나 업데이트하기 전에 페더레이션 사용자 속성을 변환합니다.
가입	the section called “사전 가입”	가입 요청을 수락 또는 거부하는 사용자 지정 검증 수행
	the section called “게시 확인”	사용자 지정 분석을 위한 사용자 지정 시작 메시지 또는 이벤트 로깅 추가
	the section called “사용자 마이그레이션”	기존 사용자 디렉터리에 있는 사용자를 사용자 풀로 마이그레이션

사용자 풀 흐름	연산	설명
메시지	the section called “사용자 지정 메시지”	고급 사용자 지정 및 메시지 현지화 수행
토큰 생성	the section called “사전 토큰 생성”	ID 및 액세스 토큰에 속성 추가 또는 제거
이메일 및 SMS 서드 파티 공급자	the section called “사용자 지정 발신자”	서드 파티 공급자를 사용하여 SMS 및 이메일 메시지 보내기

주제

- Lambda 트리거에 대해 알아야 할 사항
- 사용자 풀 Lambda 트리거 추가
- 사용자 풀 Lambda 트리거 이벤트
- 사용자 풀 Lambda 트리거 공통 파라미터
- 클라이언트 메타데이터
- API 작업을 Lambda 트리거에 연결
- Lambda 트리거를 사용자 풀 기능 작업에 연결
- 사전 가입 Lambda 트리거
- 사후 확인 Lambda 트리거
- 사전 인증 Lambda 트리거
- 사후 인증 Lambda 트리거
- 인바운드 페더레이션 Lambda 트리거
- 사용자 정의 인증 챌린지 Lambda 트리거
- 사전 토큰 생성 Lambda 트리거
- 사용자 마이그레이션 Lambda 트리거
- 사용자 정의 메시지 Lambda 트리거
- 사용자 지정 발신자 Lambda 트리거

Lambda 트리거에 대해 알아야 할 사항

Lambda 함수를 위한 사용자 풀을 준비할 때는 다음을 고려하세요.

- Amazon Cognito가 Lambda 트리거로 전송하는 이벤트는 새 기능에 따라 변경될 수 있습니다. JSON 계층 구조에서 응답 및 요청 요소의 위치가 변경되거나 요소 이름이 추가될 수 있습니다. Lambda 함수에서 이 가이드에 설명된 입력 요소 키-값 페어를 수신할 것으로 예상할 수 있지만 입력 검증이 엄격해지면 함수가 실패할 수 있습니다.
- Amazon Cognito가 일부 트리거에 전송하는 여러 버전의 이벤트 중 하나를 선택할 수 있습니다. 일부 버전에서는 Amazon Cognito 요금 변경을 수락해야 할 수 있습니다. 요금에 대한 자세한 내용은 Amazon Cognito 요금 섹션을 참조하세요. 사전 토큰 생성 Lambda 트리거에서 액세스 토큰을 사용자 지정하려면 Lite가 아닌 다른 기능 플랜으로 사용자 풀을 구성하고 Lambda 트리거 구성을 업데이트하여 이벤트 버전 2를 사용해야 합니다.
- 사용자 지정 발신자 Lambda 트리거를 제외하고, Amazon Cognito는 Lambda 함수를 동기적으로 간접 호출합니다. Amazon Cognito에서 Lambda 함수를 호출하면 5초 내에 응답해야 합니다. 그렇지 않고 호출을 재시도할 수 있는 경우 Amazon Cognito는 호출을 재시도할 수 있습니다. 모든 재시도가 실패하면 함수가 시간 초과됩니다. 이 5초 제한 시간 값은 변경할 수 없습니다. 자세한 내용은 AWS Lambda 개발자 안내서의 Lambda 프로그래밍 모델을 참조하세요.

  Amazon Cognito는 HTTP 상태 코드가 500-599인 호출 오류를 반환하는 함수 호출을 재시도하지 않습니다. 이러한 코드는 Lambda가 함수를 시작할 수 없게 만드는 구성 문제를 나타냅니다. 자세한 내용은 오류 처리 및 자동 재시도를 참조하세요 AWS Lambda.
- Lambda 트리거 구성에서는 함수 버전을 선언할 수 없습니다. Amazon Cognito 사용자 풀은 기본적으로 최신 버전의 함수를 호출합니다. 하지만 CreateUserPool 또는 UpdateUserPool API 요청에서 함수 버전을 별칭과 연결하고 트리거 `LambdaArn`을 별칭 ARN으로 설정할 수 있습니다. AWS Management Console에서는 이 옵션을 사용할 수 없습니다. 별칭에 대한 자세한 내용은 AWS Lambda 개발자 안내서의 Lambda 함수 별칭을 참조하세요.
- Lambda 트리거를 삭제하는 경우 사용자 풀의 해당 트리거를 업데이트해야 합니다. 예를 들어 사후 인증 트리거를 삭제하면 해당 사용자 풀의 사후 인증 트리거를 없음으로 설정해야 합니다.
- Lambda 함수가 Amazon Cognito에 요청 및 응답 파라미터를 반환하지 않거나 오류를 반환하는 경우 인증 이벤트는 성공하지 못합니다. 함수에서 오류를 반환하여 사용자의 가입, 인증, 토큰 생성 또는 Lambda 트리거를 호출하는 인증 흐름의 다른 단계를 방지할 수 있습니다.

  관리형 로그인은 Lambda 트리거가 생성한 오류를 로그인 프롬프트 위에 오류 텍스트로 반환합니다. Amazon Cognito 사용자 풀 API는 *`[trigger]`* `failed with error` *`[error text from response]`* 형식으로 트리거 오류를 반환합니다. Lambda 함수에서 사용자에게 표시하려는 오류만 생성하는 것이 좋습니다. `print()`와 같은 출력 방법을 사용하여 민감한 정보나 디버깅 정보를 CloudWatch Logs에 로깅합니다. 예제는 사전 가입 예제: 사용자 이름의 문자가 5자 미만인 경우 가입 거부 섹션을 참조하세요.

- 다른의 Lambda 함수를 사용자 풀의 트리거 AWS 계정 로 추가할 수 있습니다. CreateUserPool 및 UpdateUserPool API 작업 또는 CloudFormation 및에서 이에 상응하는 작업을 사용하여 교차 계정 트리거를 추가해야 합니다 AWS CLI. 에서는 교차 계정 함수를 추가할 수 없습니다 AWS Management Console.
- Amazon Cognito 콘솔에서 Lambda 트리거를 추가하면 Amazon Cognito는 사용자 풀이 함수를 호출하도록 허용하는 리소스 기반 정책을 함수에 추가합니다. Amazon Cognito 콘솔 외부에서 교차 계정 함수를 비롯한 Lambda 트리거를 생성하는 경우, Lambda 함수의 리소스 기반 정책에 권한을 추가해야 합니다. 추가한 권한은 Amazon Cognito가 사용자 풀을 대신하여 함수를 호출하도록 허용해야 합니다. Lambda 콘솔에서 권한을 추가하거나 Lambda AddPermission API 작업을 사용할 수 있습니다.

  Lambda 리소스 기반 정책 예제

  다음 Lambda 리소스 기반 정책 예제는 Amazon Cognito에 Lambda 함수를 호출할 수 있는 제한적인 권한을 부여합니다. Amazon Cognito는 `aws:SourceArn` 조건의 사용자 풀과 `aws:SourceAccount` 조건의 계정을 둘 다 대신하는 경우에만 함수를 호출할 수 있습니다.

  JSON

```
{
    "Version":"2012-10-17",
    "Id": "default",
    "Statement": [
        {
            "Sid": "LambdaCognitoIdpTrust",
            "Effect": "Allow",
            "Principal": {
                "Service": "cognito-idp.amazonaws.com"
            },
            "Action": "lambda:InvokeFunction",
            "Resource": "arn:aws:lambda:us-
east-1:111122223333:function:MyFunction",
            "Condition": {
                "StringEquals": {
                    "AWS:SourceAccount": "111122223333"
                },
                "ArnLike": {
                    "AWS:SourceArn": "arn:aws:cognito-idp:us-
east-1:111122223333:userpool/us-east-1_EXAMPLE"
                }
            }
```

```
        }
    ]
}
```

사용자 풀 Lambda 트리거 추가

콘솔을 사용하여 사용자 풀 Lambda 트리거를 추가하려면

1. Lambda 콘솔을 사용하여 Lambda 함수를 생성합니다. Lambda 함수에 대한 자세한 내용은 AWS Lambda 개발자 가이드를 참조하세요.
2. Amazon Cognito 콘솔로 이동한 다음 사용자 풀(User Pools)을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. 확장 메뉴를 선택하고 Lambda 트리거를 찾습니다.
5. Lambda 트리거 추가(Add a Lambda trigger)를 선택합니다.
6. 사용자 지정할 인증 스테이지에 따라 Lambda 트리거 범주(Category)를 선택합니다.
7. Lambda 함수 할당을 선택하고 AWS 리전 사용자 풀과 동일한에서 함수를 선택합니다.

> **Note**
>
> AWS Identity and Access Management (IAM) 자격 증명에 Lambda 함수를 업데이트할 수 있는 권한이 있는 경우 Amazon Cognito는 Lambda 리소스 기반 정책을 추가합니다. 이 정책을 사용하면 선택한 함수를 Amazon Cognito에서 호출할 수 있습니다. 로그인한 자격 증명에 충분한 IAM 권한이 없는 경우 리소스 기반 정책을 별도로 업데이트해야 합니다. 자세한 내용은 the section called “알아야 할 것들” 섹션을 참조하세요.

8. 변경 사항 저장(Save changes)을 선택합니다.
9. Lambda 콘솔에서 CloudWatch를 사용하여 Lambda 함수를 로그할 수 있습니다. 자세한 내용은 Lambda에 대한 CloudWatch Logs 액세스를 참조하세요.

사용자 풀 Lambda 트리거 이벤트

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. Lambda 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. 함수가 수정 없이 입력 이벤트를 반환하면 Amazon Cognito는 기본 동작을 진행합니다. 다음은 모든 Lambda 트리거 입력 이벤트에 공통적인 파라미터를

보여줍니다. 트리거별 이벤트 구문의 경우 각 트리거에 대해 이 가이드의 섹션에 있는 이벤트 스키마를 검토합니다.

JSON

```
{
    "version": "string",
    "triggerSource": "string",
    "region": AWSRegion,
    "userPoolId": "string",
    "userName": "string",
    "callerContext":
        {
            "awsSdkVersion": "string",
            "clientId": "string"
        },
    "request":
        {
            "userAttributes": {
                "string": "string",
                ....
            }
        },
    "response": {}
}
```

사용자 풀 Lambda 트리거 공통 파라미터

버전

Lambda 함수의 버전 번호입니다.

triggerSource

Lambda 함수를 트리거한 이벤트의 이름입니다. 각 triggerSource의 설명은 Lambda 트리거를 사용자 풀 기능 작업에 연결 섹션을 참조하세요.

리전

`AWSRegion` 인스턴스 AWS 리전 로서의 .

userPoolId

사용자 풀의 ID입니다.

userName

현재 사용자의 사용자 이름입니다.

callerContext

요청 및 코드 환경에 대한 메타데이터입니다. 여기에는 awsSdkVersion 및 clientId 필드가 포함됩니다.

awsSdkVersion

요청을 생성한 AWS SDK의 버전입니다.

clientId

사용자 풀 앱 클라이언트의 ID입니다.

request

사용자의 API 요청 세부 정보입니다. 여기에는 다음 필드와 해당 트리거에 해당하는 모든 요청 파라미터가 포함됩니다. 예를 들어 Amazon Cognito가 사전 인증 트리거로 보내는 이벤트에는 `userNotFound` 파라미터도 포함됩니다. 사용자가 등록되지 않은 사용자 이름으로 로그인하려고 할 때 이 파라미터의 값을 처리하여 사용자 지정 작업을 수행할 수 있습니다.

userAttributes

사용자 속성 이름 및 값의 키 값 페어(1개 이상)입니다. 예: `"email": "john@example.com"`.

response

이 파라미터는 원래 요청의 정보를 포함하지 않습니다. Lambda 함수는 전체 이벤트를 Amazon Cognito에 반환하고 모든 반환 파라미터를 `response`에 추가해야 합니다. 함수에 포함할 수 있는 반환 파라미터를 보려면 사용하려는 트리거에 대한 설명서를 참조하세요.

클라이언트 메타데이터

API 작업 및 Token 엔드포인트 요청에서 Lambda 트리거 함수에 사용자 지정 파라미터를 제출할 수 있습니다. 클라이언트 메타데이터를 사용하면 애플리케이션이 요청이 발생한 환경에 대한 추가 정보를 수집할 수 있습니다. Lambda 함수에 클라이언트 메타데이터를 전달하면 추가 데이터를 처리하고 인증 흐름의 로깅 또는 사용자 지정에 사용할 수 있습니다. 클라이언트 메타데이터는 JSON 키-값 형식의 선택 및 설계 문자열 페어입니다.

클라이언트 메타데이터 사용 사례 예시

- 가입 시 지리 위치 데이터를 사전 가입 트리거에 전달하고 원치 않는 위치에서 로그인하지 않도록 합니다.
- 테넌트 ID 데이터를 사용자 지정 문제 트리거에 전달하고 다양한 사업부의 고객에게 다양한 문제를 발행합니다.
- 사용자의 토큰을 사전 토큰 생성 트리거에 전달하고 M2M 요청이 대신 수행된 보안 주체의 로그를 생성합니다. 요청 예시는 기본 권한이 있는 클라이언트 자격 증명 섹션을 참조하세요.

다음은 클라이언트 메타데이터를 사전 가입 트리거에 전달하는 예입니다.

SignUp request

다음은 Amazon Cognito가 사전 가입 트리거에 전달하는 클라이언트 메타데이터가 있는 SignUp 요청의 예입니다.

```
POST HTTP/1.1
Host: cognito-idp.us-east-1.amazonaws.com
X-Amz-Date: 20230613T200059Z
Accept-Encoding: gzip, deflate, br
X-Amz-Target: AWSCognitoIdentityProviderService.SignUp
User-Agent: <UserAgentString>
Authorization: AWS4-HMAC-SHA256 Credential=<Credential>, SignedHeaders=<Headers>,
 Signature=<Signature>
Content-Length: <PayloadSizeBytes>

{
    "ClientId": "1example23456789",
    "Username": "mary_major",
    "Password": "<Password>",
    "SecretHash": "<Secret hash>",
    "ClientMetadata": {
        "IpAddress" : "192.0.2.252",
        "GeoLocation" : "Netherlands (Kingdom of the) [NL]"
    }
    "UserAttributes": [
        {
            "Name": "name",
            "Value": "Mary"
        },
        {
```

```
            "Name": "email",
            "Value": "mary_major@example.com"
        },
        {
            "Name": "phone_number",
            "Value": "+12065551212"
        }
    ],
}
```

Lambda trigger input event

그러면 사전 가입 함수에 다음과 같은 요청 본문이 생성됩니다.

```
{
    "callerContext": {
        "awsSdkVersion": "aws-sdk-unknown-unknown",
        "clientId": "1example23456789"
    },
    "region": "us-west-2",
    "request": {
        "clientMetadata": {
            "GeoLocation": "Netherlands (Kingdom of the) [NL]",
            "IpAddress": "192.0.2.252"
        },
        "userAttributes": {
            "email": "mary_major@example.com",
            "name": "Mary",
            "phone_number": "+12065551212"
        },
        "validationData": null
    },
    "response": {
        "autoConfirmUser": false,
        "autoVerifyEmail": false,
        "autoVerifyPhone": false
    },
    "triggerSource": "PreSignUp_SignUp",
    "userName": "mary_major2",
    "userPoolId": "us-west-2_EXAMPLE",
    "version": "1"
}
```

M2M(머신 간) 클라이언트 자격 증명을 위한 클라이언트 메타데이터

M2M 요청에서 클라이언트 메타데이터를 전달할 수 있습니다. 클라이언트 메타데이터는 사전 토큰 생성 Lambda 트리거의 결과에 기여할 수 있는 사용자 또는 애플리케이션 환경의 추가 정보입니다. 사용자 보안 주체를 사용한 인증 작업에서 AdminRespondToAuthChallenge 및 RespondToAuthChallenge API 요청 본문의 사전 토큰 생성 트리거에 클라이언트 메타데이터를 전달할 수 있습니다. 애플리케이션은 Token 엔드포인트에 대한 직접 요청으로 M2M에 대한 액세스 토큰 생성을 위한 흐름을 수행하므로 모델이 다릅니다. 클라이언트 자격 증명에 대한 토큰 요청의 POST 본문에서 클라이언트 메타데이터 객체 URL 인코딩(`x-www-form-urlencoded`)이 포함된 `aws_client_metadata` 파라미터를 문자열에 전달합니다. 요청 예시는 기본 권한이 있는 클라이언트 자격 증명 섹션을 참조하세요. 다음은 키-값 페어 `{"environment": "dev", "language": "en-US"}`를 전달하는 예제 파라미터입니다.

```
aws_client_metadata=%7B%22environment%22%3A%20%22dev%22,%20%22language%22%3A%20%22en-US%22%7D
```

임시 사용자 속성: **`validationData`**

일부 인증 작업에는 `validationData` 파라미터도 있습니다. 클라이언트 메타데이터와 마찬가지로 Amazon Cognito가 자동으로 Lambda 트리거에 수집하지 않는 외부 정보를 전달할 수 있는 기회입니다. 검증 데이터 필드는 가입 및 로그인 작업에서 추가 사용자 컨텍스트와 함께 Lambda 함수를 제공하기 위한 것입니다. SignUp 및 AdminCreateUser는 `validationData`를 사전 가입 트리거에 전달합니다. InitiateAuth 및 AdminInitiateAuth는 입력 이벤트의 `validationData`와 같이 API 요청 본문의 `ClientMetadata`을 사전 인증 및 마이그레이션 사용자 트리거에 전달합니다.

API 작업을 클라이언트 메타데이터를 전달할 수 있는 함수에 매핑하려면 다음 트리거 소스 섹션을 참조하세요.

API 작업을 Lambda 트리거에 연결

다음 섹션에서는 Amazon Cognito가 사용자 풀의 활동에서 호출하는 Lambda 트리거를 설명합니다.

앱이 Amazon Cognito 사용자 풀 API, 관리형 로그인 또는 사용자 풀 엔드포인트를 통해 사용자를 로그인하면 Amazon Cognito는 세션 컨텍스트를 기반으로 Lambda 함수를 호출합니다. Amazon Cognito 사용자 풀 API와 사용자 풀 엔드포인트에 대한 자세한 내용은 API, OIDC 및 관리형 로그인 페이지 인증 이해을 참조하세요. 다음 섹션의 표에서는 Amazon Cognito가 함수를 호출하도록 하는 이벤트와 Amazon Cognito가 요청에 포함하는 `triggerSource` 문자열을 설명합니다.

주제

- Amazon Cognito API의 Lambda 트리거
- 관리형 로그인에서 Amazon Cognito 로컬 사용자를 위한 Lambda 트리거
- 페더레이션 사용자를 위한 Lambda 트리거

Amazon Cognito API의 Lambda 트리거

다음 표에서는 앱이 사용자 풀 로컬 사용자를 생성, 로그인 또는 업데이트할 때 Amazon Cognito가 호출할 수 있는 Lambda 트리거의 소스 문자열을 설명합니다.

Amazon Cognito API의 로컬 사용자 트리거 소스

API 작업	Lambda 트리거	트리거 소스
AdminCreateUser	사전 가입	`PreSignUp_AdminCreateUser`
	사전 토큰 생성	`TokenGeneration_NewPasswordChallenge`
	사용자 지정 메시지	`CustomMessage_AdminCreateUser`
	사용자 지정 이메일 발신자	`CustomEmailSender_AdminCreateUser`
	사용자 지정 SMS 발신자	`CustomSMSSender_AdminCreateUser`
SignUp	사전 가입	`PreSignUp_SignUp`
	사용자 지정 메시지	`CustomMessage_SignUp`
	사용자 지정 이메일 발신자	`CustomEmailSender_SignUp`
	사용자 지정 SMS 발신자	`CustomSMSSender_SignUp`
ConfirmSignUp	게시 확인	`PostConfirmation_ConfirmSignUp`

API 작업	Lambda 트리거	트리거 소스
AdminConfirmSignUp		
InitiateAuth AdminInitiateAuth	사전 인증	PreAuthentication_Authentication
	사후 인증	PostAuthentication_Authentication
	인증 문제 정의	DefineAuthChallenge_Authentication
	인증 문제 생성	CreateAuthChallenge_Authentication
	인증 문제 확인	VerifyAuthChallenge_Authentication
	사전 토큰 생성	TokenGeneration_Authentication TokenGeneration_AuthenticateDevice TokenGeneration_RefreshTokens
	사용자 마이그레이션	UserMigration_Authentication
	사용자 지정 메시지	CustomMessage_Authentication
	사용자 지정 이메일 발신자	CustomEmailSender_AccountTakeOverNotification CustomEmailSender_Authentication

API 작업	Lambda 트리거	트리거 소스
	사용자 지정 SMS 발신자	`CustomSMSSender_Authentication`
RespondToAuthChallenge AdminRespondToAuthChallenge	사후 인증	`PostAuthentication_Authentication`
	인증 문제 정의	`DefineAuthChallenge_Authentication`
	인증 문제 생성	`CreateAuthChallenge_Authentication`
	인증 문제 확인	`VerifyAuthChallenge_Authentication`

	사전 토큰 생성	`TokenGeneration_Authentication`
		
		`TokenGeneration_AuthenticateDevice`
		
		`TokenGeneration_RefreshTokens`

	사용자 지정 메시지	`CustomMessage_Authentication`

	사용자 지정 이메일 발신자	`CustomEmailSender_AccountTakeOverNotification`
		
		`CustomEmailSender_Authentication`

	사용자 지정 SMS 발신자	`CustomSMSSender_Authentication`

API 작업	Lambda 트리거	트리거 소스
ForgotPassword	사용자 마이그레이션	UserMigration_ForgotPassword
	사용자 지정 메시지	CustomMessage_ForgotPassword
	사용자 지정 이메일 발신자	CustomEmailSender_ForgotPassword
	사용자 지정 SMS 발신자	CustomSMSSender_ForgotPassword
ConfirmForgotPassword	게시 확인	PostConfirmation_ConfirmForgotPassword
UpdateUserAttributes AdminUpdateUserAttributes	사용자 지정 메시지	CustomMessage_UpdateUserAttribute
	사용자 지정 이메일 발신자	CustomEmailSender_UpdateUserAttribute
	사용자 지정 SMS 발신자	CustomSMSSender_UpdateUserAttribute
VerifyUserAttributes	사용자 지정 메시지	CustomMessage_VerifyUserAttribute
	사용자 지정 이메일 발신자	CustomEmailSender_VerifyUserAttribute
	사용자 지정 SMS 발신자	CustomSMSSender_VerifyUserAttribute
GetTokensFromRefreshToken	사전 토큰 생성	TokenGeneration_Authentication

관리형 로그인에서 Amazon Cognito 로컬 사용자를 위한 Lambda 트리거

다음 표에서는 로컬 사용자가 관리형 로그인을 사용하여 사용자 풀에 로그인할 때 Amazon Cognito가 호출할 수 있는 Lambda 트리거의 소스 문자열을 설명합니다.

관리형 로그인의 로컬 사용자 트리거 소스

관리형 로그인 URI	Lambda 트리거	트리거 소스
/signup	사전 가입	PreSignUp_SignUp
	사용자 지정 메시지	CustomMessage_SignUp
	사용자 지정 이메일 발신자	CustomEmailSender_SignUp
	사용자 지정 SMS 발신자	CustomSMSSender_SignUp
/confirmuser	게시 확인	PostConfirmation_ConfirmSignUp
/login	사전 인증	PreAuthentication_Authentication
	사전 토큰 생성	TokenGeneration_Authentication TokenGeneration_AuthenticateDevice TokenGeneration_RefreshTokens
	사용자 마이그레이션	UserMigration_Authentication
	사용자 지정 메시지	CustomMessage_Authentication

관리형 로그인 URI	Lambda 트리거	트리거 소스
/forgotpassword	사용자 지정 이메일 발신자	CustomEmailSender_AccountTakeOverNotification CustomEmailSender_Authentication
	사용자 지정 SMS 발신자	CustomSMSSender_Authentication
	사용자 마이그레이션	UserMigration_ForgotPassword
	사용자 지정 메시지	CustomMessage_ForgotPassword
	사용자 지정 이메일 발신자	CustomEmailSender_ForgotPassword
	사용자 지정 SMS 발신자	CustomSMSSender_ForgotPassword
/confirmforgotpassword	게시 확인	PostConfirmation_ConfirmForgotPassword

페더레이션 사용자를 위한 Lambda 트리거

다음 Lambda 트리거를 사용하여 페더레이션 공급자로 로그인하는 사용자에 대한 사용자 풀 워크플로를 사용자 지정할 수 있습니다.

> **Note**
>
> 페더레이션 사용자는 관리형 로그인을 사용하여 로그인할 수 있습니다. 또는 자격 증명 공급자 로그인 페이지로 해당 사용자를 자동 리디렉션하는 권한 부여 엔드포인트에 대한 요청을 귀하

가 생성하는 방법도 있습니다. Amazon Cognito 사용자 풀 API를 사용하여 페더레이션 사용자를 로그인할 수 없습니다.

페더레이션 사용자 트리거 소스

로그인 이벤트	Lambda 트리거	트리거 소스
첫 로그인	사전 가입	`PreSignUp_ExternalProvider`
	게시 확인	`PostConfirmation_ConfirmSignUp`
	사전 토큰 생성	`TokenGeneration_HostedAuth`
후속 로그인	사전 인증	`PreAuthentication_Authentication`
	사후 인증	`PostAuthentication_Authentication`
	사전 토큰 생성	`TokenGeneration_HostedAuth`

페더레이션 로그인은 사용자 풀에서 사용자 정의 인증 챌린지 Lambda 트리거, 사용자 마이그레이션 Lambda 트리거, 사용자 정의 메시지 Lambda 트리거 또는 사용자 지정 발신자 Lambda 트리거를 호출하지 않습니다.

Lambda 트리거를 사용자 풀 기능 작업에 연결

각 Lambda 트리거는 사용자 풀에서 기능적 역할을 수행합니다. 예를 들어 트리거는 가입 흐름을 수정하거나 사용자 지정 인증 문제를 추가할 수 있습니다. Amazon Cognito가 Lambda 함수로 보내는 이벤트는 해당 기능적 역할을 구성하는 여러 작업 중 하나를 반영할 수 있습니다. 예를 들어, Amazon Cognito는 사용자가 가입할 때와 귀하가 사용자를 생성할 때 가입 전 트리거를 호출합니다. 동일한 기능적 역할에 대해 서로 다른 이러한 사례는 각각 고유한 `triggerSource` 값을 지닙니다. Lambda 함수는 해당 함수를 호출한 작업에 따라 수신 이벤트를 다르게 처리할 수 있습니다.

또한 Amazon Cognito는 이벤트가 트리거 소스에 해당하는 경우 할당된 모든 함수를 호출합니다. 예를 들어, 마이그레이션 사용자 및 인증 전 트리거를 할당한 사용자 풀에 사용자가 로그인하면 두 가지가 모두 활성화됩니다.

가입, 확인 및 로그인(인증) 트리거

트리거	triggerSource 값	Event
사전 가입	`PreSignUp_SignUp`	사전 가입.
사전 가입	`PreSignUp_AdminCreateUser`	관리자가 새 사용자를 만들면 사전 가입.
사전 가입	`PreSignUp_ExternalProvider`	외부 자격 증명 공급자에 대한 사전 가입.
게시 확인	`PostConfirmation_ConfirmSignUp`	사후 가입 확인.
게시 확인	`PostConfirmation_ConfirmForgotPassword`	사후 암호 분실 확인.
사전 인증	`PreAuthentication_Authentication`	사전 인증.
사후 인증	`PostAuthentication_Authentication`	사후 인증.

사용자 지정 인증 문제 트리거

트리거	triggerSource 값	Event
인증 문제 정의	`DefineAuthChallenge_Authentication`	인증 문제 정의.
인증 문제 생성	`CreateAuthChallenge_Authentication`	인증 문제 생성.

트리거	triggerSource 값	Event
인증 문제 확인	`VerifyAuthChallengeResponse_Authentication`	인증 문제 응답 확인.

페더레이션 트리거

트리거	triggerSource 값	Event
인바운드 페더레이션	`InboundFederation_ExternalProvider`	인바운드 페더레이션.

사전 토큰 생성 트리거

트리거	triggerSource 값	Event
사전 토큰 생성	`TokenGeneration_HostedAuth`	Amazon Cognito는 관리형 로그인 페이지에서 사용자를 인증합니다.
사전 토큰 생성	`TokenGeneration_Authentication`	사용자 인증 또는 토큰 새로 고침이 완료되었습니다.
사전 토큰 생성	`TokenGeneration_NewPasswordChallenge`	관리자가 사용자를 생성합니다. Amazon Cognito는 사용자가 임시 암호를 변경해야 할 때 이를 호출합니다.
사전 토큰 생성	`TokenGeneration_AuthenticateDevice`	사용자 디바이스 인증의 끝입니다.
사전 토큰 생성	`TokenGeneration_RefreshTokens`	사용자가 자격 증명 및 액세스 토큰을 새로 고치려고 합니다.

사용자 트리거 마이그레이션

트리거	triggerSource 값	Event
사용자 마이그레이션	`UserMigration_Authentication`	로그인 시 사용자 마이그레이션입니다.
사용자 마이그레이션	`UserMigration_ForgotPassword`	암호 찾기 흐름 도중 사용자 마이그레이션.

사용자 지정 메시지 트리거

트리거	triggerSource 값	Event
사용자 지정 메시지	`CustomMessage_SignUp`	사용자가 사용자 풀에 가입할 때 사용자 지정 메시지입니다.
사용자 지정 메시지	`CustomMessage_AdminCreateUser`	관리자로서의 사용자가 생성되고 Amazon Cognito에서 이 사용자에게 임시 암호를 보낼 때 사용자 지정 메시지입니다.
사용자 지정 메시지	`CustomMessage_ResendCode`	기존 사용자가 새 확인 코드를 요청할 때 사용자 지정 메시지입니다.
사용자 지정 메시지	`CustomMessage_ForgotPassword`	사용자가 암호 재설정을 요청할 때 사용자 지정 메시지입니다.
사용자 지정 메시지	`CustomMessage_UpdateUserAttribute`	사용자가 이메일 주소 또는 전화 번호를 변경하고 Amazon Cognito에서 확인 코드를 보낼 때 사용자 지정 메시지입니다.
사용자 지정 메시지	`CustomMessage_VerifyUserAttribute`	사용자가 이메일 주소 또는 전화 번호를 추가하고 Amazon Cognito에서 확인 코드를 보낼 때 사용자 지정 메시지입니다.

트리거	triggerSource 값	Event
사용자 지정 메시지	`CustomMessage_Authentication`	SMS MFA를 구성한 사용자가 로그인할 때 사용자 지정 메시지입니다.

사용자 지정 발신자 트리거

트리거	triggerSource 값	Event

사용자 지정 발신자	`CustomEmailSender_SignUp`
	
	`CustomSmsSender_SignUp`
		사용자가 사용자 풀에 가입하는 경우.

사용자 지정 발신자	`CustomEmailSender_AdminCreateUser`
	
	`CustomSmsSender_AdminCreateUser`
		사용자를 관리자로 생성하면 Amazon Cognito에서 해당 사용자에게 임시 암호를 보냅니다.

사용자 지정 발신자	`CustomEmailSender_ForgotPassword`
	
	`CustomSmsSender_ForgotPassword`
		사용자가 암호 재설정을 요청하는 경우.

사용자 지정 발신자	`CustomEmailSender_UpdateUserAttribute`
	
	`CustomSmsSender_UpdateUserAttribute`
		사용자가 이메일 주소 또는 전화 번호를 변경하고 Amazon Cognito에서 확인 코드를 보낼 때.

사용자 지정 발신자	`CustomEmailSender_VerifyUserAttribute`
	
	`CustomSmsSender_VerifyUserAttribute`
		사용자가 이메일 주소 또는 전화 번호를 추가하고 Amazon Cognito에서 확인 코드를 보내는 경우.

트리거	triggerSource 값	Event

사용자 지정 발신자	`CustomEmailSender_Authentication`
	
	`CustomSmsSender_Authentication`
		SMS 또는 이메일을 구성한 사용자가 로그인하는 경우.

사용자 지정 발신자	`CustomEmailSender_AccountTakeOverNotification`	위협 방지 설정이 사용자의 로그인 시도에 대해 자동화된 조치를 취하고 위험 수준에 대한 조치에 알림이 포함된 경우.

사전 가입 Lambda 트리거

셀프 서비스 가입 옵션이 있는 사용자 풀에서 가입 프로세스를 사용자 지정할 수 있습니다. 가입 전 트리거의 몇 가지 일반적인 용도는 새 사용자의 사용자 지정 분석 및 기록을 수행하거나, 보안 및 거버넌스 표준을 적용하거나, 타사 IdP의 사용자를 통합 사용자 프로필에 연결하는 것입니다. 검증 및 확인을 거치지 않아도 되는 신뢰할 수 있는 사용자가 있을 수도 있습니다.

Amazon Cognito가 새로운 로컬 또는 페더레이션 사용자 생성을 완료하기 직전에 사전 가입 Lambda 함수를 활성화합니다. 이 함수로 전송된 요청 객체의 `userAttributes`에는 로컬 사용자 가입에서 제공하거나 페더레이션 사용자의 공급자 속성에서 성공적으로 매핑된 속성이 포함되어 있습니다. 사용자 풀은 SignUp을 사용한 셀프 서비스 가입 또는 신뢰할 수 있는 ID 공급업체를 사용한 최초 로그인과 AdminCreateUser를 사용한 사용자 생성 시이 트리거를 호출합니다. 가입 프로세스의 일환으로 이 함수를 사용하여 사용자 지정 로직으로 로그인 이벤트를 분석하고 새 사용자를 수정하거나 거부할 수 있습니다.

주제

- 사전 가입 Lambda 트리거 파라미터
- 사전 가입 예제: 등록된 도메인의 사용자 자동 확인
- 사전 가입 예제: 모든 사용자 자동 확인 및 자동 검증
- 사전 가입 예제: 사용자 이름의 문자가 5자 미만인 경우 가입 거부

사전 가입 Lambda 트리거 파라미터

Amazon Cognito가 이 Lambda 함수에 전달하는 요청은 아래 파라미터와 Amazon Cognito가 모든 요청에 추가하는 공통 파라미터의 조합입니다.

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
        },
        "validationData": {
            "string": "string",
            . . .
         },
        "clientMetadata": {
            "string": "string",
            . . .
         }
    },

    "response": {
        "autoConfirmUser": "boolean",
        "autoVerifyPhone": "boolean",
        "autoVerifyEmail": "boolean"
    }
}
```

사전 가입 요청 파라미터

userAttributes

사용자 속성을 나타내는 하나 이상의 이름-값 페어입니다. 속성 이름은 키입니다.

validationData

새 사용자 생성 요청 시 앱이 Amazon Cognito에 전달한 사용자 속성 데이터가 포함된 하나 이상의 키-값 페어입니다. AdminCreateUser 또는 SignUp API 요청의 ValidationData 파라미터를 통해 Lambda 함수에 이 정보를 보냅니다.

Amazon Cognito는 ValidationData 데이터를 생성된 사용자의 속성으로 설정하지 않습니다. ValidationData는 사전 가입 Lambda 트리거 목적으로 제공하는 임시 사용자 정보입니다.

clientMetadata

사전 가입 트리거에 지정하는 Lambda 함수에 사용자 지정 입력으로 제공할 수 있는 하나 이상의 키-값 페어입니다. 다음 API 작업에서 ClientMetadata 파라미터를 사용하여 이 데이터를 Lambda 함수에 전달할 수 있습니다. AdminCreateUser, AdminRespondToAuthChallenge, ForgotPassword 및 SignUp.

사전 가입 응답 파라미터

사용자를 자동으로 확인하려면 응답에서 `autoConfirmUser`를 `true`로 설정하면 됩니다. `autoVerifyEmail`을 `true`로 설정하여 사용자 이메일을 자동으로 확인할 수 있습니다. `autoVerifyPhone`을 `true`로 설정하여 사용자 전화 번호를 자동으로 확인할 수 있습니다.

> **Note**
>
> 응답 파라미터 `autoVerifyPhone`, `autoVerifyEmail` 및 `autoConfirmUser`는 `AdminCreateUser` API에 의해 사전 가입 Lambda 함수가 트리거될 때 Amazon Cognito에서 무시됩니다.

autoConfirmUser

사용자를 자동으로 확인하려면 `true`로 설정하고 그렇지 않으면 `false`로 설정합니다.

autoVerifyEmail

가입 중인 사용자의 이메일 주소를 확인한 것으로 설정하려면 `true`로 설정하고 그렇지 않으면 `false`로 설정합니다. `autoVerifyEmail`이 `true`로 설정되면 `email` 속성에 null이 아닌 유효한 값이 있어야 합니다. 그렇지 않으면 오류가 발생하고 사용자가 가입을 완료할 수 없습니다.

`email` 속성을 별칭으로 선택하면가 설정될 때 사용자의 이메일 주소에 대한 별칭`autoVerifyEmail`이 생성됩니다. 해당 이메일 주소의 별칭이 이미 있는 경우 별칭이 새 사용자로 이동되고 이전 사용자의 이메일 주소가 확인되지 않음으로 표시됩니다. 자세한 내용은 로그인 속성 사용자 지정 섹션을 참조하세요.

autoVerifyPhone

가입 중인 사용자의 전화 번호를 확인한 것으로 설정하려면 `true`로 설정하고 그렇지 않으면 `false`로 설정합니다. `autoVerifyPhone`이 `true`로 설정되면 phone_number 속성에 null이 아

닌 유효한 값이 있어야 합니다. 그렇지 않으면 오류가 발생하고 사용자가 가입을 완료할 수 없습니다.

phone_number 속성을 별칭으로 선택하면가 설정될 때 사용자의 전화번호에 대한 별칭autoVerifyPhone이 생성됩니다. 해당 전화 번호가 있는 별칭이 이미 있는 경우 별칭이 새 사용자로 이동되고 이전 사용자의 전화 번호가 확인되지 않음으로 표시됩니다. 자세한 내용은 로그인 속성 사용자 지정 섹션을 참조하세요.

사전 가입 예제: 등록된 도메인의 사용자 자동 확인

다음은 Lambda 트리거 코드의 예입니다. Amazon Cognito가 가입 요청을 처리하기 직전에 사전 가입 트리거가 호출됩니다. 이 트리거는 사용자 지정 속성 custom:domain을 사용하여 특정 이메일 도메인의 신규 사용자를 자동으로 확인할 수 있습니다. 사용자 지정 도메인에 없는 모든 신규 사용자는 사용자 풀에 추가되지만 자동으로 확인되지는 않습니다.

Node.js

```
export const handler = async (event, context, callback) => {
  // Set the user pool autoConfirmUser flag after validating the email domain
  event.response.autoConfirmUser = false;

  // Split the email address so we can compare domains
  var address = event.request.userAttributes.email.split("@");

  // This example uses a custom attribute "custom:domain"
  if (event.request.userAttributes.hasOwnProperty("custom:domain")) {
    if (event.request.userAttributes["custom:domain"] === address[1]) {
      event.response.autoConfirmUser = true;
    }
  }

  // Return to Amazon Cognito
  callback(null, event);
};
```

Python

```
def lambda_handler(event, context):
    # It sets the user pool autoConfirmUser flag after validating the email domain
    event['response']['autoConfirmUser'] = False
```

```
    # Split the email address so we can compare domains
    address = event['request']['userAttributes']['email'].split('@')

    # This example uses a custom attribute 'custom:domain'
    if 'custom:domain' in event['request']['userAttributes']:
        if event['request']['userAttributes']['custom:domain'] == address[1]:
            event['response']['autoConfirmUser'] = True

    # Return to Amazon Cognito
    return event
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그런 다음 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. Lambda 콘솔에서 해당 Lambda 트리거와 관련 있는 데이터로 테스트 이벤트를 설정할 수 있습니다. 다음은 이 코드 샘플의 테스트 이벤트입니다.

JSON

```
{
    "request": {
        "userAttributes": {
            "email": "testuser@example.com",
            "custom:domain": "example.com"
        }
    },
    "response": {}
}
```

사전 가입 예제: 모든 사용자 자동 확인 및 자동 검증

이 예에서는 모든 사용자를 자동으로 컨펌하고, 속성이 있는 경우에 확인된 것으로 사용자의 `email` 및 phone_number 속성을 설정합니다. 또한 별칭 지정이 활성화된 경우 자동 확인이 설정`email`되면 phone_number 및에 대해 별칭이 생성됩니다.

> **Note**
>
> 전화번호가 동일한 별칭이 이미 있는 경우 별칭이 새 사용자로 이동되고 이전 사용자의 phone_number가 확인되지 않은 것으로 표시됩니다. 이메일 주소도 마찬가지입니다. 사용자 풀 ListUsers API를 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. 그런 다음 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. Lambda 콘솔에서 해당 Lambda 트리거와 관련 있는 데이터로 테스트 이벤트를 설정할 수 있습니다. 다음은 이 코드 샘플의 테스트 이벤트입니다.

JSON

```
{
  "request": {
    "userAttributes": {
      "email": "user@example.com",
      "phone_number": "+12065550100"
    }
  },
  "response": {}
}
```

사전 가입 예제: 사용자 이름의 문자가 5자 미만인 경우 가입 거부

이 예제에서는 가입 요청에서 사용자 이름의 길이를 확인합니다. 이 예에서는 사용자가 5자 미만의 이름을 입력한 경우 오류를 반환합니다.

Node.js

```
export const handler = (event, context, callback) => {
    // Impose a condition that the minimum length of the username is 5 is imposed on
 all user pools.
    if (event.userName.length < 5) {
        var error = new Error("Cannot register users with username less than the
 minimum length of 5");
        // Return error to Amazon Cognito
        callback(error, event);
    }
    // Return to Amazon Cognito
    callback(null, event);
};
```

Python

```
def lambda_handler(event, context):
    if len(event['userName']) < 5:
        raise Exception("Cannot register users with username less than the minimum
 length of 5")
    # Return to Amazon Cognito
    return event
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그런 다음 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. Lambda 콘솔에서 해당 Lambda 트리거와 관련 있는 데이터로 테스트 이벤트를 설정할 수 있습니다. 다음은 이 코드 샘플의 테스트 이벤트입니다.

JSON

```
{
  "userName": "rroe",
  "response": {}
}
```

사후 확인 Lambda 트리거

Amazon Cognito는 가입한 사용자가 사용자 계정을 확인한 후 이 트리거를 호출합니다. 사후 확인 Lambda 함수에서 사용자 지정 메시지를 보내거나 사용자 지정 API 요청을 추가할 수 있습니다. 예를 들어 외부 시스템을 쿼리하고 사용자에게 추가 속성을 채울 수 있습니다. Amazon Cognito는 사용자 풀에 등록한 사용자에 대해서만 이 트리거를 호출하고 관리자 보안 인증 정보로 생성한 사용자 계정에 대해서는 호출하지 않습니다.

요청에는 확인된 사용자의 현재 속성이 포함됩니다. 사용자 풀은 ConfirmSignUp, AdminConfirmSignUp 및 ConfirmForgotPassword에서 사후 확인 함수를 호출합니다. 이 트리거는 사용자가 관리형 로그인에서 가입 또는 암호 재설정을 확인할 때도 실행됩니다.

주제

- 사후 확인 Lambda 트리거 파라미터
- 사후 확인 예제

사후 확인 Lambda 트리거 파라미터

Amazon Cognito가 이 Lambda 함수에 전달하는 요청은 아래 파라미터와 Amazon Cognito가 모든 요청에 추가하는 공통 파라미터의 조합입니다.

JSON

```
{
    "request": {
            "userAttributes": {
                "string": "string",
```

```
                . . .
            },
            "clientMetadata": {
             "string": "string",
             . . .
            }
        },
    "response": {}
}
```

사후 확인 요청 파라미터

userAttributes

사용자 속성을 나타내는 하나 이상의 키-값 페어입니다.

clientMetadata

사후 확인 트리거에 지정하는 Lambda 함수에 사용자 지정 입력으로 제공할 수 있는 하나 이상의 키-값 페어입니다. 다음 API 작업에서 ClientMetadata 파라미터를 사용하여 이 데이터를 Lambda 함수에 전달할 수 있습니다. AdminConfirmSignUp, ConfirmForgotPassword, ConfirmSignUp 및 SignUp.

사후 확인 응답 파라미터

응답에는 추가적인 반환 정보가 없습니다.

사후 확인 예제

이 예에서 Lambda 함수는 Amazon SES를 사용하여 사용자에게 확인 이메일 메시지를 전송합니다. 자세한 내용은 Amazon Simple Storage Service 개발자 가이드를 참조하세요.

Node.js

```
// Import required AWS SDK clients and commands for Node.js. Note that this requires
// the `@aws-sdk/client-ses` module to be either bundled with this code or included
// as a Lambda layer.
import { SES, SendEmailCommand } from "@aws-sdk/client-ses";
const ses = new SES();

const handler = async (event) => {
  if (event.request.userAttributes.email) {
```

```
    await sendTheEmail(
      event.request.userAttributes.email,
      `Congratulations ${event.userName}, you have been confirmed.`,
    );
  }
  return event;
};

const sendTheEmail = async (to, body) => {
  const eParams = {
    Destination: {
      ToAddresses: [to],
    },
    Message: {
      Body: {
        Text: {
          Data: body,
        },
      },
      Subject: {
        Data: "Cognito Identity Provider registration completed",
      },
    },
    // Replace source_email with your SES validated email address
    Source: "<source_email>",
  };
  try {
    await ses.send(new SendEmailCommand(eParams));
  } catch (err) {
    console.log(err);
  }
};

export { handler };
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그런 다음 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. Lambda 콘솔에서 해당 Lambda 트리거와 관련 있는 데이터로 테스트 이벤트를 설정할 수 있습니다. 다음은 이 코드 샘플의 테스트 이벤트입니다.

JSON

```
{
    "request": {
        "userAttributes": {
            "email": "user@example.com",
            "email_verified": true
        }
    },
    "response": {}
}
```

사전 인증 Lambda 트리거

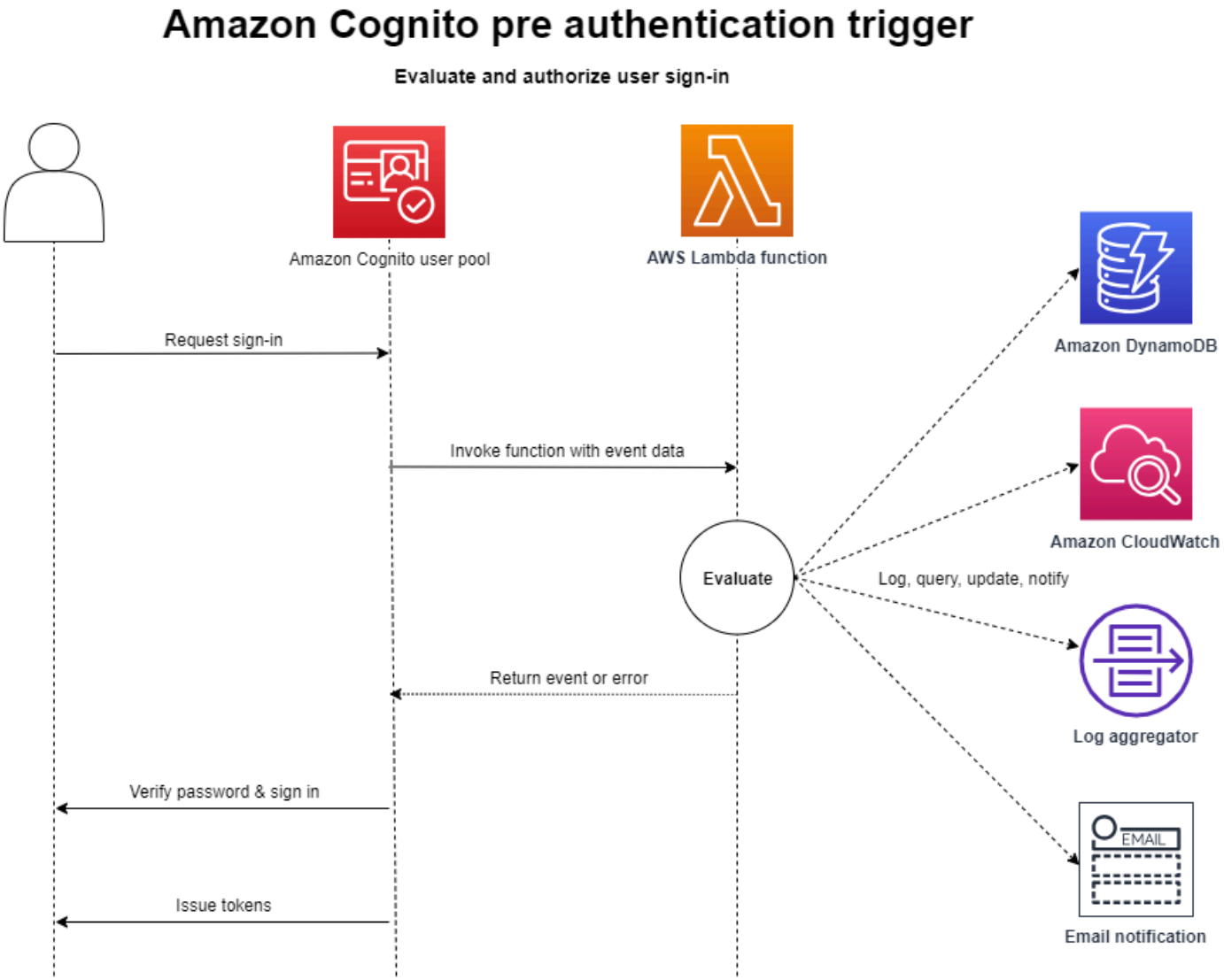
예비 작업을 수행하는 사용자 지정 검증을 생성할 수 있도록 사용자가 로그인을 시도할 때 Amazon Cognito가 이 트리거를 호출합니다. 예를 들어 인증 요청을 거부하거나 세션 데이터를 외부 시스템에 기록할 수 있습니다.

> **Note**
>
> 이 Lambda 트리거는 사용자 풀 앱 클라이언트의 `PreventUserExistenceErrors` 설정이 `ENABLED`로 설정되어 있지 않으면 사용자가 존재하지 않는 경우 활성화되지 않습니다. 기존 인증 세션을 갱신해도 이 트리거가 활성화되지 않습니다.

주제

- 흐름 개요
- 사전 인증 Lambda 트리거 파라미터
- 사전 인증 예제

흐름 개요



요청에는 앱이 사용자 풀 `InitiateAuth` 및 `AdminInitiateAuth` API 작업에 전달하는 `ClientMetadata` 값에 있는 클라이언트 검증 데이터가 포함됩니다.

자세한 내용은 인증 세션의 예 단원을 참조하십시오.

사전 인증 Lambda 트리거 파라미터

Amazon Cognito가 이 Lambda 함수에 전달하는 요청은 아래 파라미터와 Amazon Cognito가 모든 요청에 추가하는 공통 파라미터의 조합입니다.

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
        },
        "validationData": {
            "string": "string",
            . . .
        },
        "userNotFound": boolean
    },
    "response": {}
}
```

사전 인증 요청 파라미터

userAttributes

사용자 속성을 나타내는 하나 이상의 이름-값 페어입니다.

userNotFound

사용자 풀 클라이언트에 대해 `PreventUserExistenceErrors`를 `ENABLED`로 설정한 경우 Amazon Cognito에서 이 부울을 채웁니다.

validationData

사용자 로그인 요청에 검증 데이터를 포함하는 하나 이상의 키-값 페어입니다. 이 데이터를 Lambda 함수에 전달하려면 InitiateAuth 및 AdminInitiateAuth API 작업에서 ClientMetadata 파라미터를 사용합니다.

사전 인증 응답 파라미터

Amazon Cognito는 함수가 응답에서 반환하는 추가 정보를 처리하지 않습니다. 함수가 오류를 반환하여 로그인 시도를 거부하거나 API 작업을 사용하여 리소스를 쿼리하고 수정할 수 있습니다.

사전 인증 예제

이 예제 함수는 사용자가 특정 앱 클라이언트로 사용자 풀에 로그인하지 못하도록 합니다. 사전 인증 Lambda 함수는 사용자에게 기존 세션이 있을 때는 호출하지 않기 때문에 이 함수는 차단하려는 앱 클라이언트 ID가 있는 새 세션만 방지합니다.

Node.js

```
const handler = async (event) => {
  if (
    event.callerContext.clientId === "user-pool-app-client-id-to-be-blocked"
  ) {
    throw new Error("Cannot authenticate users from this user pool app client");
  }

  return event;
};

export { handler };
```

Python

```
def lambda_handler(event, context):
    if event['callerContext']['clientId'] == "<user pool app client id to be
 blocked>":
        raise Exception("Cannot authenticate users from this user pool app client")

    # Return to Amazon Cognito
    return event
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그런 다음 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. Lambda 콘솔에서 해당 Lambda 트리거와 관련 있는 데이터로 테스트 이벤트를 설정할 수 있습니다. 다음은 이 코드 샘플의 테스트 이벤트입니다.

JSON

```
{
    "callerContext": {
        "clientId": "<user pool app client id to be blocked>"
    },
    "response": {}
```

```
}
```

사후 인증 Lambda 트리거

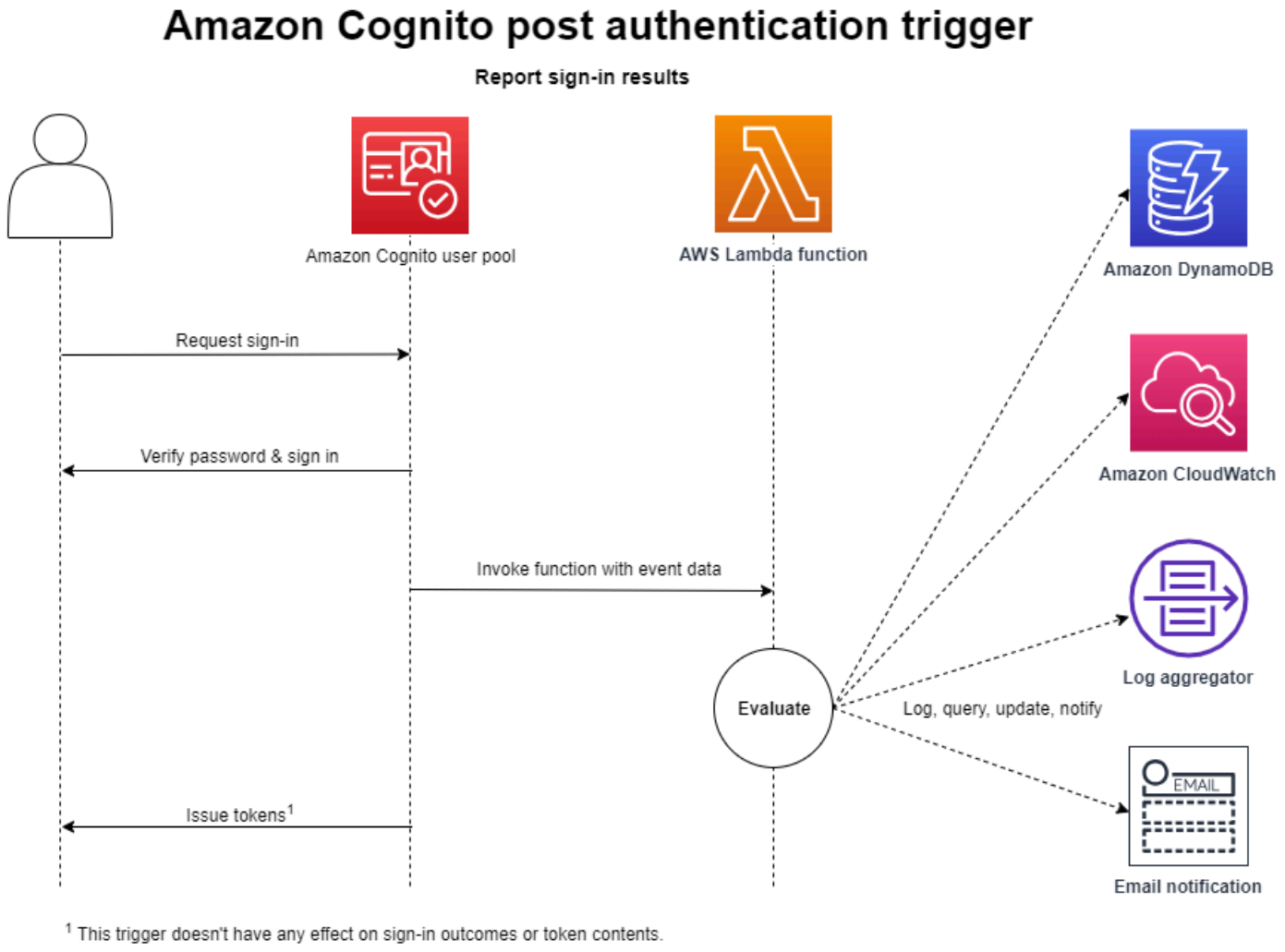
사후 인증 트리거는 사용자의 인증 흐름을 변경하지 않습니다. Amazon Cognito는 인증이 완료된 후 사용자가 토큰을 수신하기 전에 이 Lambda를 호출합니다. 다음 로그인에 반영되는 로깅 또는 사용자 프로필 조정과 같은 인증 이벤트의 사용자 지정 사후 처리를 추가하려는 경우 사후 인증 트리거를 추가합니다.

요청 본문을 Amazon Cognito에 반환하지 않는 사후 인증 Lambda를 사용하면 여전히 인증이 완료되지 않을 수 있습니다. 자세한 내용은 Lambda 트리거에 대해 알아야 할 사항 섹션을 참조하세요.

주제

- 인증 흐름 개요
- 사후 인증 Lambda 트리거 파라미터
- 사후 인증 예제

인증 흐름 개요



자세한 내용은 인증 세션의 예 섹션을 참조하세요.

사후 인증 Lambda 트리거 파라미터

Amazon Cognito가 이 Lambda 함수에 전달하는 요청은 아래 파라미터와 Amazon Cognito가 모든 요청에 추가하는 공통 파라미터의 조합입니다.

JSON

```
{
    "request": {
        "userAttributes": {
```

```
            "string": "string",
            . . .
        },
        "newDeviceUsed": boolean,
        "clientMetadata": {
            "string": "string",
            . . .
           }
       },
    "response": {}
}
```

사후 인증 요청 파라미터

newDeviceUsed

이 플래그는 사용자가 새로운 디바이스에 로그인했는지 여부를 표시합니다. 기억된 사용자 풀의 디바이스 값이 `Always` 또는 `User Opt-In`으로 설정된 경우에만 Amazon Cognito에서 이 플래그를 설정합니다.

userAttributes

사용자 속성을 나타내는 하나 이상의 이름-값 페어입니다.

clientMetadata

사후 인증 트리거에 지정하는 Lambda 함수에 사용자 지정 입력으로 제공할 수 있는 하나 이상의 키-값 페어입니다. 이 데이터를 Lambda 함수에 전달하려면 AdminRespondToAuthChallenge 및 RespondToAuthChallenge API 작업에서 ClientMetadata 파라미터를 사용합니다. Amazon Cognito는 사후 승인 함수에 전달하는 요청에 있는 AdminInitiateAuth 및 InitiateAuth API 작업의 ClientMetadata 파라미터에서 전달된 데이터를 포함하지 않습니다.

사후 인증 응답 파라미터

Amazon Cognito는 응답에서 추가 반환 정보를 기대하지 않습니다. 함수는 API 작업을 사용하여 리소스를 쿼리 및 수정하거나 이벤트 메타데이터를 외부 시스템에 기록할 수 있습니다.

사후 인증 예제

이 사후 인증 Lambda 함수 샘플은 CloudWatch Logs에 성공적으로 로그인하여 데이터를 전송합니다.

Node.js

```
const handler = async (event) => {
  // Send post authentication data to Amazon CloudWatch logs
  console.log("Authentication successful");
  console.log("Trigger function =", event.triggerSource);
  console.log("User pool = ", event.userPoolId);
  console.log("App client ID = ", event.callerContext.clientId);
  console.log("User ID = ", event.userName);

  return event;
};

export { handler };
```

Python

```
import os
def lambda_handler(event, context):

    # Send post authentication data to Cloudwatch logs
    print ("Authentication successful")
    print ("Trigger function =", event['triggerSource'])
    print ("User pool = ", event['userPoolId'])
    print ("App client ID = ", event['callerContext']['clientId'])
    print ("User ID = ", event['userName'])

    # Return to Amazon Cognito
    return event
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그런 다음 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. Lambda 콘솔에서 해당 Lambda 트리거와 관련 있는 데이터로 테스트 이벤트를 설정할 수 있습니다. 다음은 이 코드 샘플의 테스트 이벤트입니다.

JSON

```
{
  "triggerSource": "testTrigger",
  "userPoolId": "testPool",
  "userName": "testName",
  "callerContext": {
```

```
        "clientId": "12345"
    },
    "response": {}
}
```

인바운드 페더레이션 Lambda 트리거

인바운드 페더레이션 트리거는 외부 자격 증명 공급자와의 인증 프로세스 중에 페더레이션 사용자 속성을 변환합니다. 사용자가 구성된 자격 증명 공급자를 통해 인증하면이 트리거를 사용하면 인증 프로세스의 데이터를 가로채고 변환하여 외부 SAML 및 OIDC 공급자의 응답을 수정할 수 있으므로 Amazon Cognito 사용자 풀이 페더레이션 사용자 및 해당 속성을 처리하는 방법을 프로그래밍 방식으로 제어할 수 있습니다.

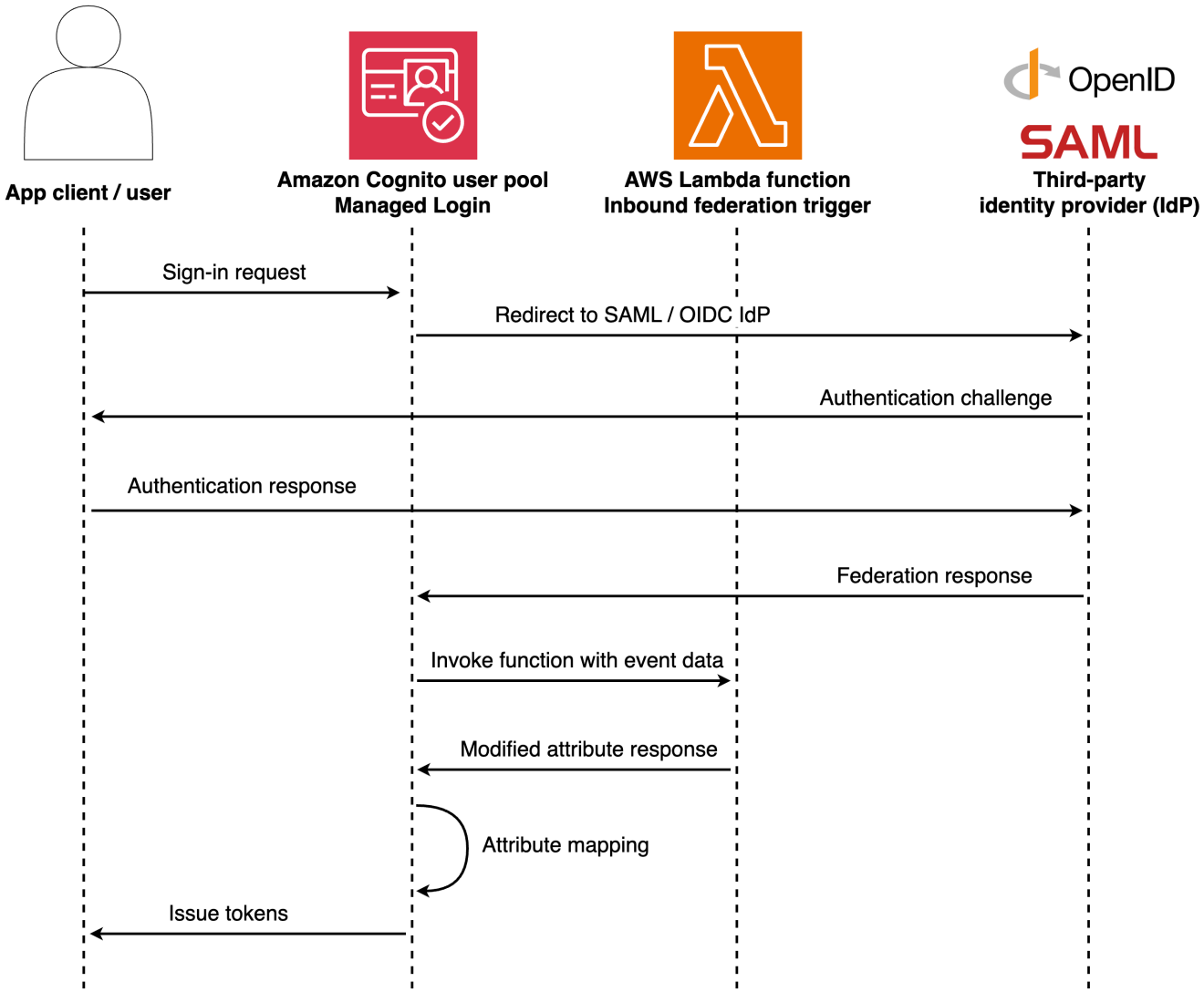
새 사용자를 생성하거나 기존 페더레이션 사용자 프로필을 업데이트하기 전에이 트리거를 사용하여 속성을 추가, 재정의 또는 억제합니다. 이 트리거는 원시 자격 증명 공급자 속성을 입력으로 수신하고 Amazon Cognito가 사용자 프로필에 적용하는 수정된 속성을 반환합니다.

주제

- 흐름 개요
- 인바운드 페더레이션 Lambda 트리거 파라미터
- 인바운드 페더레이션 예제: 그룹 멤버십 관리
- 인바운드 페더레이션 예제: 큰 속성 잘라내기
- 인바운드 페더레이션 예제: 페더레이션 이벤트 로깅

흐름 개요

사용자가 외부 자격 증명 공급자로 인증하면 Amazon Cognito는 사용자 프로필을 생성하거나 업데이트하기 전에 인바운드 페더레이션 트리거를 호출합니다. 트리거는 자격 증명 공급자로부터 원시 속성을 수신하고 Amazon Cognito가 저장하기 전에 변환할 수 있습니다. 이 흐름은 새 페더레이션 사용자와 페더레이션을 통해 다시 로그인하는 기존 사용자 모두에게 발생합니다.



인바운드 페더레이션 Lambda 트리거 파라미터

Amazon Cognito가 이 Lambda 함수에 전달하는 요청은 아래 파라미터와 Amazon Cognito가 모든 요청에 추가하는 공통 파라미터의 조합입니다.

JSON

```
{
    "version": "string",
    "triggerSource": "InboundFederation_ExternalProvider",
    "region": AWSRegion,
    "userPoolId": "string",
    "userName": "string",
```

```
    "callerContext": {
        "awsSdkVersion": "string",
        "clientId": "string"
    },
    "request": {
        "providerName": "string",
        "providerType": "string",
        "attributes": {
            "tokenResponse": {
                "access_token": "string",
                "token_type": "string",
                "expires_in": "string"
            },
            "idToken": {
                "sub": "string",
                "email": "string",
                "email_verified": "string"
            },
            "userInfo": {
                "email": "string",
                "given_name": "string",
                "family_name": "string"
            },
            "samlResponse": {
                "string": "string"
            }
        }
    },
    "response": {
        "userAttributesToMap": {
            "string": "string"
        }
    }
}
```

인바운드 페더레이션 요청 파라미터

providerName

외부 자격 증명 공급자의 이름입니다.

providerType

외부 자격 증명 공급자의 유형입니다. 유효한 값: `OIDC`, `SAML`, `Facebook`, `Google`, `SignInWithApple`, `LoginWithAmazon`.

attributes

처리 전에 자격 증명 공급자로부터 받은 원시 속성입니다. 구조는 공급자 유형에 따라 다릅니다.

attributes.tokenResponse

`/token` 엔드포인트의 OAuth 토큰 응답 데이터입니다. OIDC 및 소셜 공급자만 사용할 수 있습니다. `access_token`, `id_token`, `refresh_token`, `token_type`, 및 `expires_in`를 포함합니다`scope`.

attributes.idToken

디코딩되고 검증된 ID 토큰 JWT 클레임. OIDC 및 소셜 공급자만 사용할 수 있습니다. `sub` (고유 사용자 식별자), `email`, `name`, `iss` (발급자), `aud` (대상), `exp` (만료) 및 `iat` (발급된 시간)을 포함하여 확인된 사용자 자격 증명 정보를 포함합니다.

attributes.userInfo

UserInfo 엔드포인트의 확장 사용자 프로필 정보입니다. OIDC 및 소셜 공급자만 사용할 수 있습니다. `given_name`, , `family_name`, 및 기타 공급자별 필드`pictureaddress`와 같은 세부 프로필 속성을 포함합니다. IdP가 UserInfo 엔드포인트를 지원하지 않거나 엔드포인트 호출이 실패하는 경우 비어 있을 수 있습니다.

attributes.samlResponse

SAML 어설션 속성. SAML 공급자에만 사용할 수 있습니다. SAML 응답의 속성을 포함합니다.

인바운드 페더레이션 응답 파라미터

userAttributesToMap

사용자 프로필에 적용할 사용자 속성입니다.

> **Important**
>
> 수정하지 않는 속성을 포함하여 응답에 유지하려는 모든 사용자 속성을 포함해야 합니다. `userAttributesToMap` 응답에 포함되지 않은 속성은 삭제되고 사용자 프로필에 저장되지 않습니다. 이는 수정된 속성과 수정되지 않은 속성 모두에 적용됩니다.

빈 응답 동작

에 {} 대해 빈 객체를 반환하면 `userAttributesToMap`자격 증명 공급자의 모든 원래 속성이 변경되지 않고 유지됩니다. 이는 Lambda 함수가 실행되지 않은 것처럼 no-op 역할을 합니다. 이는 속성을 생략하여 속성을 삭제하는 것과 다릅니다.

공급자별 속성

의 구조는에 따라 `request.attributes` 달라집니다`providerType`. OIDC 및 소셜 공급자에는 `tokenResponse`, `idToken`및 `userInfo` 객체가 포함됩니다. SAML 공급자는 `samlResponse` 객체만 포함합니다.

인바운드 페더레이션 예제: 그룹 멤버십 관리

이 예제에서는 페더레이션 ID 제공업체 그룹을 Amazon Cognito 사용자 풀 그룹에 매핑하는 방법을 보여줍니다. 이 함수는 페더레이션 응답에서 그룹 멤버십을 추출하고 사용자를 해당 Amazon Cognito 그룹에 자동으로 추가하므로 인증 후 트리거가 필요하지 않습니다.

Node.js

```
exports.handler = async (event) => {
    const { providerType, attributes } = event.request;

    // Extract user attributes based on provider type
    let userAttributesFromIdp = {};
    if (providerType === 'SAML') {
        userAttributesFromIdp = attributes.samlResponse || {};
    } else {
        // For OIDC and Social providers, merge userInfo and idToken
        userAttributesFromIdp = {
            ...(attributes.userInfo || {}),
            ...(attributes.idToken || {})
        };
    }

    // Extract groups from federated response
    const federatedGroups = userAttributesFromIdp.groups?.split(',') || [];

    // Map federated groups to Cognito groups
```

```
    const groupMapping = {
        'Domain Admins': 'Administrators',
        'Engineering': 'Developers',
        'Sales': 'SalesTeam'
    };

    // Filter to only in-scope groups
    const mappedGroups = federatedGroups
        .map(group => groupMapping[group.trim()])
        .filter(group => group); // Remove undefined values

    // Pass through attributes with mapped groups as custom attribute
    const attributesToMap = {
        ...userAttributesFromIdp,
        'custom:user_groups': mappedGroups.join(',')
    };

    // Remove original groups attribute
    delete attributesToMap.groups;

    event.response.userAttributesToMap = attributesToMap;
    return event;
};
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그런 다음 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. Lambda 콘솔에서 해당 Lambda 트리거와 관련 있는 데이터로 테스트 이벤트를 설정할 수 있습니다. 다음은 이 코드 샘플의 테스트 이벤트입니다.

JSON

```
{
    "userPoolId": "us-east-1_XXXXXXXXX",
    "request": {
        "providerName": "CorporateAD",
        "providerType": "SAML",
        "attributes": {
            "samlResponse": {
                "email": "jane.smith@company.com",
                "given_name": "Jane",
                "family_name": "Smith",
                "groups": "Engineering,Domain Admins",
                "department": "Engineering"
```

```
            }
        }
    },
    "response": {
        "userAttributesToMap": {}
    }
}
```

인바운드 페더레이션 예제: 큰 속성 잘라내기

이 예제에서는 Amazon Cognito의 스토리지 제한을 초과하는 속성 값을 자르는 방법을 보여줍니다. 이 함수는 자격 증명 공급자의 각 속성을 확인합니다. 속성 값이 2048자를 초과하는 경우 값을 잘라내고 줄임표를 추가하여 잘림을 나타냅니다. 다른 모든 속성은 변경되지 않고 통과합니다.

Node.js

```
exports.handler = async (event) => {
    const MAX_ATTRIBUTE_LENGTH = 2048;

    // Get the identity provider attributes based on provider type
    const { providerType, attributes } = event.request;
    let idpAttributes = {};

    if (providerType === 'SAML') {
        idpAttributes = attributes.samlResponse || {};
    } else {
        // For OIDC and Social providers, merge userInfo and idToken
        idpAttributes = {
            ...(attributes.userInfo || {}),
            ...(attributes.idToken || {})
        };
    }

    const userAttributes = {};

    // Process each attribute
    for (const [key, value] of Object.entries(idpAttributes)) {
        if (typeof value === 'string' && value.length > MAX_ATTRIBUTE_LENGTH) {
            // Truncate the value and add ellipsis
            userAttributes[key] = value.substring(0, MAX_ATTRIBUTE_LENGTH - 3) +
 '...';
```

```
            console.log(`Truncated attribute ${key} from ${value.length} to
 ${userAttributes[key].length} characters`);
        } else {
            // Keep the original value
            userAttributes[key] = value;
        }
    }

    // Return the modified attributes
    event.response.userAttributesToMap = userAttributes;
    return event;
};
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그런 다음 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. Lambda 콘솔에서 해당 Lambda 트리거와 관련 있는 데이터로 테스트 이벤트를 설정할 수 있습니다. 다음은 이 코드 샘플의 테스트 이벤트입니다.

JSON

```
{
    "version": "string",
    "triggerSource": "InboundFederation_ExternalProvider",
    "region": "us-east-1",
    "userPoolId": "us-east-1_XXXXXXXXX",
    "userName": "ExampleProvider_12345",
    "callerContext": {
        "awsSdkVersion": "string",
        "clientId": "string"
    },
    "request": {
        "providerName": "ExampleProvider",
        "providerType": "OIDC",
        "attributes": {
            "tokenResponse": {
                "access_token": "abcDE...",
                "token_type": "Bearer",
                "expires_in": "3600"
            },
            "idToken": {
                "sub": "12345",
                "email": "user@example.com"
            },
```

```
            "userInfo": {
                "email": "user@example.com",
                "given_name": "Example",
                "family_name": "User",
                "bio": "This is a very long biography that contains more than 2048
 characters..."
            }
        }
    },
    "response": {
        "userAttributesToMap": {}
    }
}
```

인바운드 페더레이션 예제: 페더레이션 이벤트 로깅

이 예제에서는 모니터링 및 디버깅을 위해 페더레이션 인증 이벤트를 로깅하는 방법을 보여줍니다. 이 예제 함수는 페더레이션 사용자 및 해당 속성에 대한 자세한 정보를 캡처하여 인증 프로세스에 대한 가시성을 제공합니다.

Node.js

```
exports.handler = async (event) => {
    const { providerName, providerType, attributes } = event.request;

    // Extract user attributes based on provider type
    let userAttributesFromIdp = {};
    if (providerType === 'SAML') {
        userAttributesFromIdp = attributes.samlResponse || {};
    } else {
        // For OIDC and Social providers, merge userInfo and idToken
        userAttributesFromIdp = {
            ...(attributes.userInfo || {}),
            ...(attributes.idToken || {})
        };
    }

    // Log federated authentication details
    console.log(JSON.stringify({
        timestamp: new Date().toISOString(),
        providerName,
        providerType,
```

```
        userEmail: userAttributesFromIdp.email,
        attributeCount: Object.keys(userAttributesFromIdp).length,
        attributes: userAttributesFromIdp
    }));

    // Pass through all attributes unchanged
    event.response.userAttributesToMap = userAttributesFromIdp;
    return event;
};
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그런 다음 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. Lambda 콘솔에서 해당 Lambda 트리거와 관련 있는 데이터로 테스트 이벤트를 설정할 수 있습니다. 다음은 이 코드 샘플의 테스트 이벤트입니다.

JSON

```
{
    "version": "string",
    "triggerSource": "InboundFederation_ExternalProvider",
    "region": "us-east-1",
    "userPoolId": "us-east-1_XXXXXXXXX",
    "userName": "CorporateAD_john.doe",
    "callerContext": {
        "awsSdkVersion": "string",
        "clientId": "string"
    },
    "request": {
        "providerName": "CorporateAD",
        "providerType": "SAML",
        "attributes": {
            "samlResponse": {
                "email": "john.doe@company.com",
                "given_name": "John",
                "family_name": "Doe",
                "department": "Engineering",
                "employee_id": "EMP12345"
            }
        }
    },
    "response": {
        "userAttributesToMap": {}
    }
```

```
}
```

예상 CloudWatch Logs 출력:

JSON

```
{
    "timestamp": "2025-01-14T21:17:40.153Z",
    "providerName": "CorporateAD",
    "providerType": "SAML",
    "userEmail": "john.doe@company.com",
    "attributeCount": 5,
    "attributes": {
        "email": "john.doe@company.com",
        "given_name": "John",
        "family_name": "Doe",
        "department": "Engineering",
        "employee_id": "EMP12345"
    }
}
```

사용자 정의 인증 챌린지 Lambda 트리거

Amazon Cognito 사용자 풀에 대한 인증 흐름을 구축할 때 기본 제공 흐름 이상으로 인증 모델을 확장하고 싶을 수 있습니다. 사용자 지정 챌린지 트리거의 일반적인 사용 사례 중 하나는 사용자 이름, 암호 및 다중 인증(MFA) 이외의 추가 보안 검사를 구현하는 것입니다. 사용자 지정 챌린지는 Lambda 지원 프로그래밍 언어로 생성할 수 있는 모든 질문과 응답입니다. 예를 들어 인증이 허용되기 전에 사용자가 CAPTCHA를 해결하거나 보안 질문에 답하도록 요구할 수 있습니다. 또 다른 잠재적 요구 사항은 특수 인증 요소 또는 디바이스와 통합하는 것입니다. 또는 하드웨어 보안 키 또는 생체 인식 장치로 사용자를 인증하는 소프트웨어를 이미 개발했을 수도 있습니다. 사용자 지정 챌린지에 대한 인증 성공의 정의는 Lambda 함수가 올바른 것으로 받아들이는 답변(예: 고정 문자열 또는 외부 API의 만족스러운 응답)입니다.

사용자 지정 챌린지로 인증을 시작하고 인증 프로세스를 완전히 제어하거나 애플리케이션이 사용자 지정 챌린지를 수신하기 전에 사용자 이름 암호 인증을 수행할 수 있습니다.

사용자 지정 인증 챌린지 Lambda 트리거:

정의

챌린지 시퀀스를 시작합니다. 새 챌린지를 시작할지, 인증을 완료로 표시할지 또는 인증 시도를 중단할지 결정합니다.

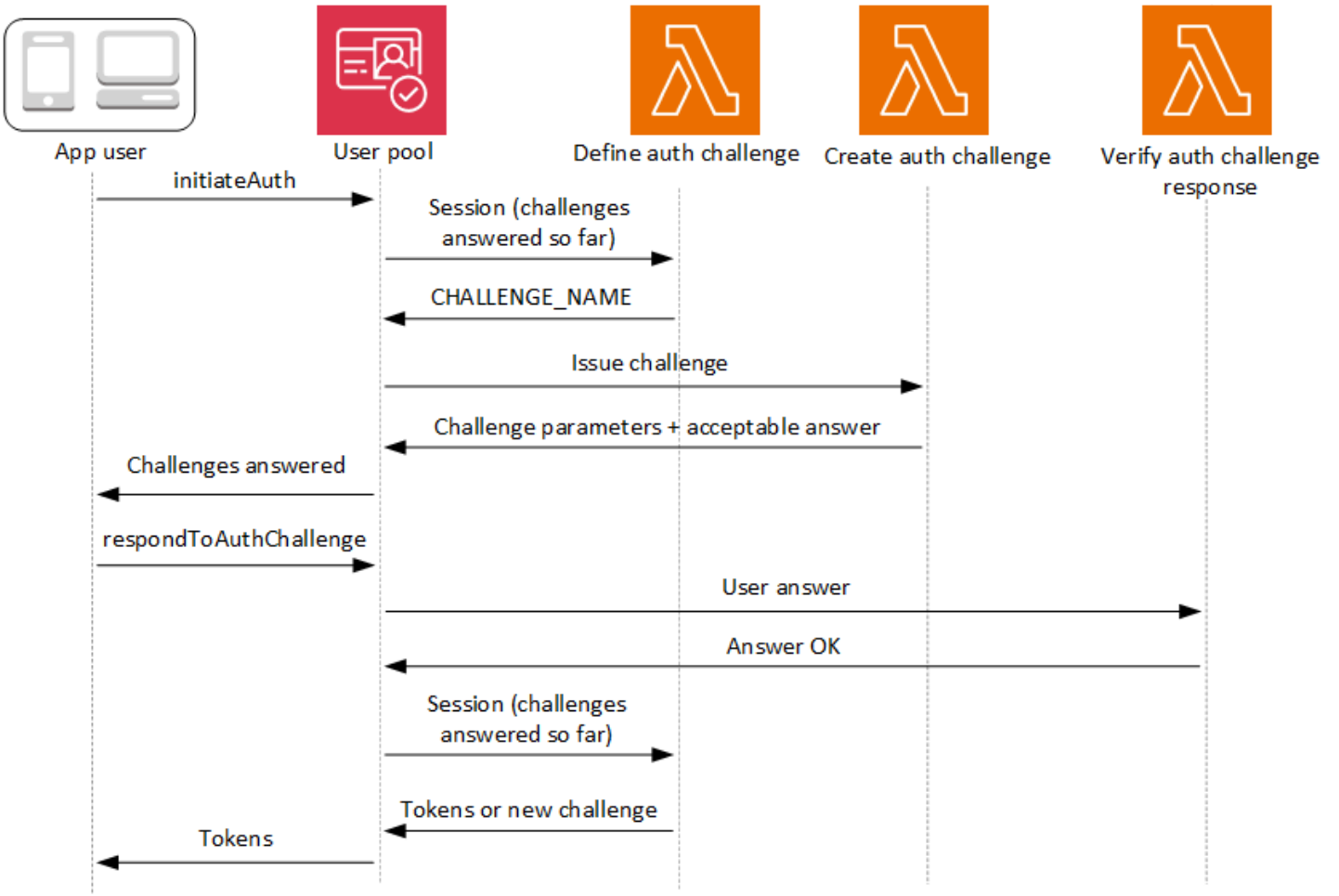
생성

사용자가 답변해야 하는 질문을 애플리케이션에 발급합니다. 이 함수는 애플리케이션이 사용자에게 표시해야 하는 보안 질문 또는 CAPTCHA에 대한 링크를 제공할 수 있습니다.

확인

예상 응답을 알고 이를 애플리케이션이 챌린지 응답에서 제공하는 응답과 비교합니다. 함수는 CAPTCHA 서비스의 API를 호출하여 사용자가 시도한 솔루션의 예상 결과를 검색할 수 있습니다.

이 세 가지 Lambda 함수는 서로 연결되어 사용자의 제어 범위 내에 있고 자체 설계에 완전히 속하는 인증 메커니즘을 제공합니다. 사용자 지정 인증에는 클라이언트 및 Lambda 함수에 애플리케이션 로직이 필요하므로 관리형 로그인 내에서 사용자 지정 인증을 처리할 수 없습니다. 이 인증 시스템에는 추가 개발자 노력이 필요합니다. 애플리케이션은 사용자 풀 API로 인증 흐름을 수행하고 사용자 지정 인증 문제의 중앙에서 질문을 렌더링하는 맞춤형 로그인 인터페이스로 결과 문제를 처리해야 합니다.



사용자 지정 인증을 구현하는 방법에 대한 자세한 내용은 사용자 지정 인증 흐름 및 챌린지 섹션을 참조하세요.

API 작업 InitiateAuth 또는 AdminInitiateAuth와 RespondToAuthChallenge 또는 AdminRespondToAuthChallenge 간의 인증. 이 흐름에서 인증이 실패하거나 사용자에게 토큰이 발행될 때까지 사용자는 연속 문제에 응답하여 인증합니다. 챌린지 응답은 새로운 챌린지일 수 있습니다. 이 경우 애플리케이션은 새로운 챌린지에 필요한 만큼 응답합니다. 인증 챌린지 정의 함수가 지금까지의 결과를 분석하여 모든 챌린지에 응답했다고 판단하고 `IssueTokens`를 반환하면 인증이 성공합니다.

> **사용자 지정 인증 트리거가 실행되지 않은 이유**
>
> 인증 챌린지 생성 또는 인증 챌린지 확인 응답 함수가 호출되지 않은 경우 원인은 일반적으로 다음 중 하나입니다.
>
> - 앱 클라이언트는 사용자 지정 인증 흐름을 허용해야 합니다. 앱 클라이언트에서 `ALLOW_CUSTOM_AUTH` (또는 선택 기반 로그인의 경우 `ALLOW_USER_AUTH`)를 활성화하고

일치하는 흐름으로 로그인을 시작합니다. SRP 또는 암호 인증만 허용하는 풀은 사용자 지정 챌린지 시퀀스에 들어가지 않습니다.

- Amazon Cognito는 세 가지 함수를 순서대로 호출합니다. 항상 인증 챌린지 정의를 먼저 호출하고 인증 챌린지 정의 함수가를 반환하는 경우에만 인증 챌린지 생성 및 인증 챌린지 응답 확인을 호출합니다`challengeName: CUSTOM_CHALLENGE`. 생성이 실행되지 않는 경우 정의가 `issueTokens: true` 또는를 설정하는 `failAuthentication: false`대신 `issueTokens: false` 및 `challengeName: CUSTOM_CHALLENGE`를 사용하여를 반환하는지 확인합니다`failAuthentication: true`.
- 세 트리거를 모두 사용자 풀에 할당해야 합니다. 할당이 누락되면 체인이 끊어집니다.

주제

- 사용자 지정 문제 흐름의 SRP 인증
- 인증 챌린지 정의 Lambda 트리거
- 인증 챌린지 생성 Lambda 트리거
- 인증 챌린지 확인 응답 Lambda 트리거

사용자 지정 문제 흐름의 SRP 인증

Amazon Cognito가 사용자 지정 문제를 표시하기 전에 사용자 암호를 확인하도록 할 수 있습니다. 요청 속도 할당량의 인증 범주에 연결된 모든 Lambda 트리거는 사용자 지정 챌린지 흐름에서 SRP 인증을 수행할 때 실행됩니다. 다음은 이 프로세스의 개요입니다.

1. 앱이 `AuthParameters` 맵으로 `InitiateAuth` 또는 `AdminInitiateAuth`를 호출하여 로그인을 시작합니다. 파라미터에는 `CHALLENGE_NAME: SRP_A,`와, `SRP_A` 및 `USERNAME`에 대한 값이 포함되어야 합니다.
2. Amazon Cognito는 `challengeName: SRP_A` 및 `challengeResult: true`를 포함하는 초기 세션과 함께 인증 문제 정의 Lambda 트리거를 호출합니다.
3. 이러한 입력을 수신한 후 Lambda 함수는 `challengeName: PASSWORD_VERIFIER`, `issueTokens: false`, `failAuthentication: false`로 응답합니다.
4. 암호 확인이 성공하면 Amazon Cognito가 `challengeName: PASSWORD_VERIFIER` 및 `challengeResult: true`가 포함된 새 세션을 사용하여 Lambda 함수를 다시 호출합니다.
5. 사용자 지정 문제를 시작하려면 Lambda 함수가 `challengeName: CUSTOM_CHALLENGE`, `issueTokens: false` 및 `failAuthentication: false`로 응답합니다. 암호 확인을 사용하여

사용자 지정 인증 흐름을 시작하지 않으려면 CHALLENGE_NAME: CUSTOM_CHALLENGE를 포함한 AuthParameters 맵으로 로그인을 시작하면 됩니다.

6. 모든 챌린지에 응답할 때까지 챌린지 루프가 반복됩니다.

다음은 SRP 흐름을 사용하여 사용자 지정 인증에 앞서 시작되는 InitiateAuth 요청의 예입니다.

```
{
    "AuthFlow": "CUSTOM_AUTH",
    "ClientId": "1example23456789",
    "AuthParameters": {
        "CHALLENGE_NAME": "SRP_A",
        "USERNAME": "testuser",
        "SRP_A": "[SRP_A]",
        "SECRET_HASH": "[secret hash]"
    }
}
```

사용자 지정 인증 SRP 흐름의 암호 재설정

사용자가 FORCE_CHANGE_PASSWORD 상태인 경우 사용자 지정 인증 흐름은 인증 문제의 무결성을 유지하면서 암호 변경 단계를 통합해야 합니다. Amazon Cognito는 NEW_PASSWORD_REQUIRED 문제 중에 인증 문제 정의 Lambda 트리거를 호출합니다. 이 시나리오에서는 사용자 지정 문제 흐름 및 SRP 인증으로 로그인하는 사용자가 암호 재설정 상태인 경우 새 암호를 설정할 수 있습니다.

사용자가 RESET_REQUIRED 또는 FORCE_CHANGE_PASSWORD 상태인 경우 NEW_PASSWORD를 사용하여 NEW_PASSWORD_REQUIRED 문제에 응답해야 합니다. SRP를 사용한 사용자 지정 인증에서 Amazon Cognito는 사용자가 SRP NEW_PASSWORD_REQUIRED 문제를 완료한 후 PASSWORD_VERIFIER 문제를 반환합니다. 인증 문제 정의 트리거는 session 배열에서 두 문제 결과를 모두 수신하며 사용자가 암호를 성공적으로 변경한 후 추가 사용자 지정 문제를 진행할 수 있습니다.

인증 문제 정의 Lambda 트리거는 SRP 인증, 암호 재설정 및 후속 사용자 지정 문제를 통해 문제 시퀀스를 관리해야 합니다. 트리거는 PASSWORD_VERIFIER 및 NEW_PASSWORD_REQUIRED 결과를 포함하여 session 파라미터에서 완료된 문제 배열을 수신합니다. 구현 예는 인증 챌린지 정의 예제 섹션을 참조하세요.

인증 흐름 단계

사용자 지정 문제 전에 암호를 확인해야 하는 사용자의 경우 프로세스는 다음 단계를 따릅니다.

1. 앱이 AuthParameters 맵으로 InitiateAuth 또는 AdminInitiateAuth를 호출하여 로그인을 시작합니다. 파라미터에는 CHALLENGE_NAME: SRP_A와, SRP_A 및 USERNAME에 대한 값이 포함되어야 합니다.
2. Amazon Cognito는 challengeName: SRP_A 및 challengeResult: true를 포함하는 초기 세션과 함께 인증 문제 정의 Lambda 트리거를 호출합니다.
3. 이러한 입력을 수신한 후 Lambda 함수는 challengeName: PASSWORD_VERIFIER, issueTokens: false, failAuthentication: false로 응답합니다.
4. 암호 확인에 성공하면 다음 두 가지 중 하나가 발생합니다.

   정상 상태인 사용자의 경우:

   Amazon Cognito가 challengeName: PASSWORD_VERIFIER 및 challengeResult: true가 포함된 새 세션을 사용하여 Lambda 함수를 다시 호출합니다.

   사용자 지정 문제를 시작하려면 Lambda 함수가 challengeName: CUSTOM_CHALLENGE, issueTokens: false 및 failAuthentication: false로 응답합니다.

   RESET_REQUIRED 또는 **FORCE_CHANGE_PASSWORD** 상태의 사용자의 경우:

   Amazon Cognito는 challengeName: PASSWORD_VERIFIER 및 challengeResult: true가 포함된 세션을 사용하여 Lambda 함수를 호출합니다.

   Lambda 함수는 challengeName: NEW_PASSWORD_REQUIRED, issueTokens: false, failAuthentication: false와 함께 응답해야 합니다.

   암호 변경에 성공하면 Amazon Cognito는 PASSWORD_VERIFIER 및 NEW_PASSWORD_REQUIRED 결과가 모두 포함된 세션으로 Lambda 함수를 호출합니다.

   사용자 지정 문제를 시작하려면 Lambda 함수가 challengeName: CUSTOM_CHALLENGE, issueTokens: false 및 failAuthentication: false로 응답합니다.
5. 모든 챌린지에 응답할 때까지 챌린지 루프가 반복됩니다.

암호 확인을 사용하여 사용자 지정 인증 흐름을 시작하지 않으려면 CHALLENGE_NAME: CUSTOM_CHALLENGE를 포함한 AuthParameters 맵으로 로그인을 시작하면 됩니다.

세션 관리

인증 흐름은 일련의 세션 IDs 및 문제 결과를 통해 세션 연속성을 유지합니다. 각 문제 응답은 세션 재사용 오류를 방지하기 위해 새 세션 ID를 생성합니다. 이는 다중 인증 흐름에 특히 중요합니다.

문제 결과는 Lambda 트리거가 수신하는 세션 배열에 시간순으로 저장됩니다.
FORCE_CHANGE_PASSWORD 상태인 사용자의 경우 세션 배열에는 다음이 포함됩니다.

1. session[0] - 초기 SRP_A 문제
2. session[1] - PASSWORD_VERIFIER 결과
3. session[2] - NEW_PASSWORD_REQUIRED 결과
4. 후속 요소 - 추가 사용자 지정 문제의 결과

인증 흐름 예시

다음 예제에서는 암호 변경과 사용자 지정 CAPTCHA 문제를 모두 완료해야 하는
FORCE_CHANGE_PASSWORD 상태의 사용자에 대한 전체 사용자 지정 인증 흐름을 보여줍니다.

1. InitiateAuth 요청

```
{
    "AuthFlow": "CUSTOM_AUTH",
    "ClientId": "1example23456789",
    "AuthParameters": {
        "CHALLENGE_NAME": "SRP_A",
        "USERNAME": "testuser",
        "SRP_A": "[SRP_A]"
    }
}
```

2. InitiateAuth 응답

```
{
    "ChallengeName": "PASSWORD_VERIFIER",
    "ChallengeParameters": {
        "USER_ID_FOR_SRP": "testuser"
    },
    "Session": "[session_id_1]"
}
```

3. **PASSWORD_VERIFIER**를 사용한 RespondToAuthChallenge 요청

```
{
    "ChallengeName": "PASSWORD_VERIFIER",
    "ClientId": "1example23456789",
    "ChallengeResponses": {
```

```
        "PASSWORD_CLAIM_SIGNATURE": "[claim_signature]",
        "PASSWORD_CLAIM_SECRET_BLOCK": "[secret_block]",
        "TIMESTAMP": "[timestamp]",
        "USERNAME": "testuser"
    },
    "Session": "[session_id_1]"
}
```

4. **NEW_PASSWORD_REQUIRED** 문제가 포함된 RespondToAuthChallenge 응답

```
{
    "ChallengeName": "NEW_PASSWORD_REQUIRED",
    "ChallengeParameters": {},
    "Session": "[session_id_2]"
}
```

5. **NEW_PASSWORD_REQUIRED**를 사용한 RespondToAuthChallenge 요청

```
{
    "ChallengeName": "NEW_PASSWORD_REQUIRED",
    "ClientId": "1example23456789",
    "ChallengeResponses": {
        "NEW_PASSWORD": "[password]",
        "USERNAME": "testuser"
    },
    "Session": "[session_id_2]"
}
```

6. CAPTCHA 사용자 지정 문제를 사용한 RespondToAuthChallenge 응답

```
{
    "ChallengeName": "CUSTOM_CHALLENGE",
    "ChallengeParameters": {
        "captchaUrl": "url/123.jpg"
    },
    "Session": "[session_id_3]"
}
```

7. CAPTCHA 사용자 지정 문제에 대한 답변이 포함된 RespondToAuthChallenge 요청

```
{
    "ChallengeName": "CUSTOM_CHALLENGE",
    "ClientId": "1example23456789",
```

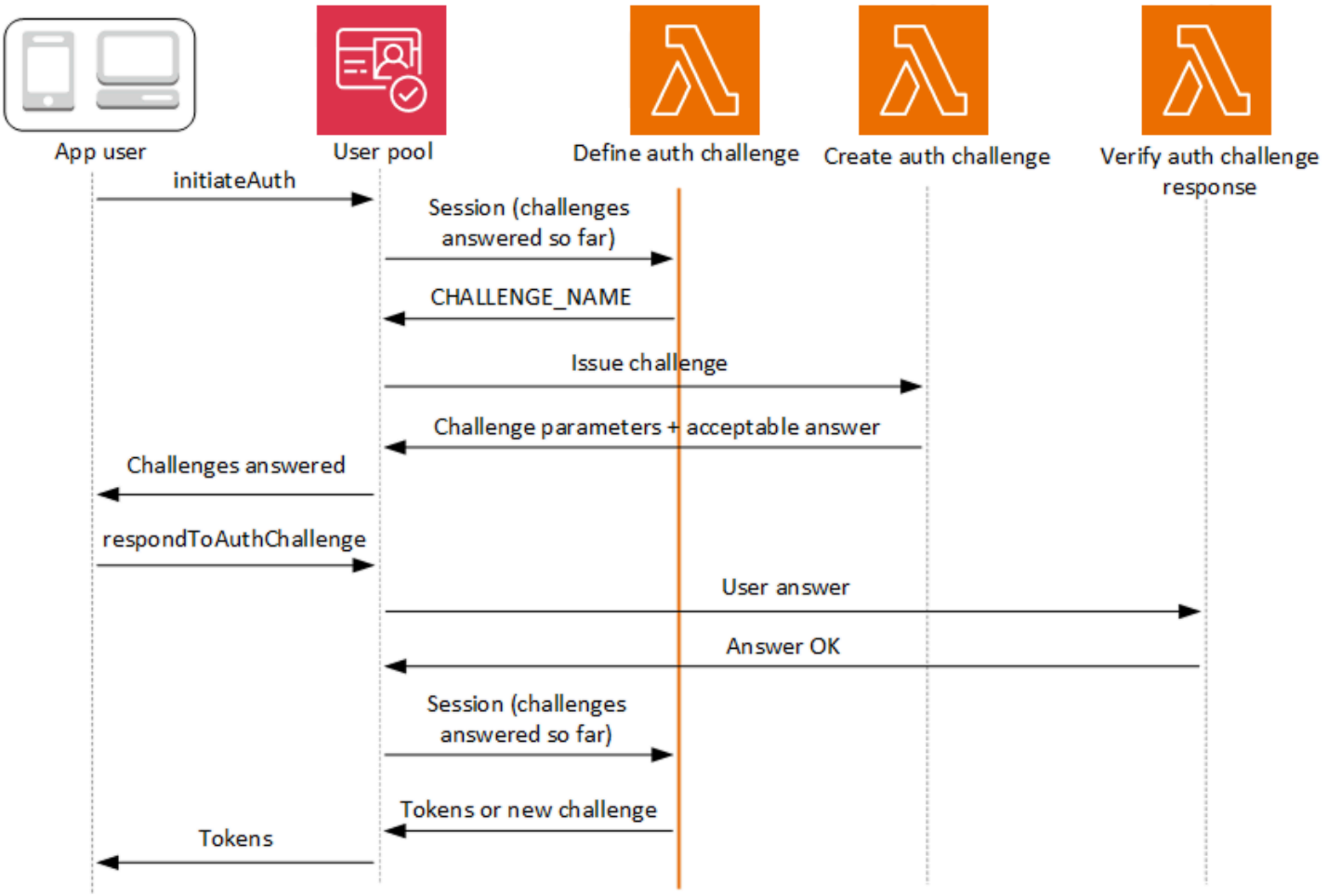
```
    "ChallengeResponses": {
        "ANSWER": "123",
        "USERNAME": "testuser"
    },
    "Session": "[session_id_3]"
}
```

6. 최종 성공 응답

```
{
    "AuthenticationResult": {
        "AccessToken": "eyJra456defEXAMPLE",
        "ExpiresIn": 3600,
        "IdToken": "eyJra789ghiEXAMPLE",
        "RefreshToken": "eyJjd123abcEXAMPLE",
        "TokenType": "Bearer"
    },
    "ChallengeParameters": {}
}
```

인증 챌린지 정의 Lambda 트리거

인증 챌린지 정의 트리거는 사용자 지정 인증 흐름에서 챌린지 시퀀스를 유지하는 Lambda 함수입니다. 챌린지 시퀀스의 성공 또는 실패를 선언하고 시퀀스가 아직 완료되지 않은 경우 다음 챌린지를 설정합니다.



인증 문제 정의

Amazon Cognito가 이 트리거를 호출하여 사용자 지정 인증 흐름을 시작합니다.

이 Lambda 트리거에 대한 요청에는 `session`이 포함됩니다. `session` 파라미터는 현재 인증 프로세스에서 사용자에게 표시되는 모든 문제가 포함된 배열입니다. 해당하는 결과를 포함하는 어레이입니다. `session` 배열에는 문제 세부 정보(`ChallengeResult`)가 시간순으로 저장됩니다. 문제 `session[0]`은 사용자가 수신하는 첫 번째 문제를 나타냅니다.

주제

- 인증 챌린지 정의 Lambda 트리거 파라미터
- 인증 챌린지 정의 예제

인증 챌린지 정의 Lambda 트리거 파라미터

Amazon Cognito가 이 Lambda 함수에 전달하는 요청은 아래 파라미터와 Amazon Cognito가 모든 요청에 추가하는 공통 파라미터의 조합입니다.

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
                . . .
        },
        "session": [
            ChallengeResult,
            . . .
        ],
        "clientMetadata": {
            "string": "string",
            . . .
        },
        "userNotFound": boolean
    },
    "response": {
        "challengeName": "string",
        "issueTokens": boolean,
        "failAuthentication": boolean
    }
}
```

인증 챌린지 정의 요청 파라미터

Amazon Cognito에서 Lambda 함수를 호출하면 Amazon Cognito가 다음과 같은 파라미터를 제공합니다.

userAttributes

사용자 속성을 나타내는 하나 이상의 이름-값 페어입니다.

userNotFound

사용자 풀 클라이언트에 대해 `PreventUserExistenceErrors`가 `ENABLED`로 설정된 경우 Amazon Cognito에서 채우는 부울입니다. 값이 `true`이면 해당 사용자 ID(사

용자 이름, 이메일 주소, 기타 세부 정보)와 일치하는 기존 사용자가 없는 것입니다. `PreventUserExistenceErrors`가 `ENABLED`로 설정된 경우 서비스는 존재하지 않는 사용자를 앱에 알리지 않습니다. Lambda 함수에서 동일한 사용자 경험을 유지하고 대기 시간을 고려하는 것이 좋습니다. 이렇게 하면 호출자가 사용자의 존재 여부에 상관없이 동일한 동작을 감지할 수 있습니다.

세션

`ChallengeResult` 요소 배열입니다. 각각 다음 요소가 포함됩니다.

challengeName

챌린지 유형 `CUSTOM_CHALLENGE`, `SRP_A`, `PASSWORD_VERIFIER`, `SMS_MFA`, `EMAIL_OTP`, `SOFTWARE_TOKEN_MFA`, `DEVICE_SRP_AUTH`, `DEVICE_PASSWORD_VERIFIER`, `ADMIN_NO_SRP_AUTH` 중 하나입니다.

인증 챌린지 정의 함수가 멀티 팩터 인증을 설정한 사용자에게 `PASSWORD_VERIFIER` 챌린지를 발급하면 Amazon Cognito는 `SMS_MFA`, `EMAIL_OTP` 또는 `SOFTWARE_TOKEN_MFA` 챌린지로 후속 작업을 실행합니다. 다음은 다중 인증 코드에 대한 프롬프트입니다. 함수에 `SMS_MFA`, `EMAIL_OTP` 및 `SOFTWARE_TOKEN_MFA` 챌린지의 입력 이벤트 처리를 포함하세요. 인증 챌린지 정의 함수에서 MFA 챌린지를 호출할 필요는 없습니다.

> **Important**
>
> 사용자가 성공적으로 인증되었고 토큰을 발급받아야 하는지 여부를 함수가 결정할 때 인증 챌린지 정의 함수에서 항상 `challengeName`을 확인하여 예상 값과 일치하는지 확인해야 합니다.

challengeResult

사용자가 챌린지를 성공적으로 완료하면 `true`로 설정하고 그렇지 않으면 `false`로 설정합니다.

challengeMetadata

사용자 지정 챌린지의 이름입니다. `challengeName`이 `CUSTOM_CHALLENGE`인 경우에만 사용됩니다.

clientMetadata

인증 문제 정의 트리거에 지정하는 Lambda 함수에 사용자 지정 입력으로 제공할 수 있는 하나 이상의 키-값 쌍입니다. 이 데이터를 Lambda 함수에 전달하려면 AdminRespondToAuthChallenge 및

RespondToAuthChallenge API 작업에서 `ClientMetadata` 파라미터를 사용합니다. 인증 문제 정의 함수를 호출하는 요청에는 AdminInitiateAuth 및 InitiateAuth API 작업의 ClientMetadata 파라미터에 전달된 데이터가 포함되지 않습니다.

인증 챌린지 정의 응답 파라미터

응답에서 인증 프로세스의 다음 단계를 반환할 수 있습니다.

challengeName

다음 문제의 이름이 포함된 문자열입니다. 사용자에게 새로운 챌린지를 표시하려면 여기에 챌린지 이름을 지정하세요.

issueTokens

사용자가 인증 문제를 충분히 완료했다고 확인한 경우 `true`로 설정합니다. 사용자가 문제를 충분히 충족하지 못한 경우 `false`로 설정합니다.

failAuthentication

현재 인증 프로세스를 종료하려는 경우 `true`로 설정합니다. 현재 인증 프로세스를 계속하려면 `false`로 설정합니다.

인증 챌린지 정의 예제

이 예에서는 인증을 위한 일련의 문제를 정의하고 사용자가 모든 문제를 성공적으로 완료한 경우에만 토큰을 발행합니다. 사용자가 `SRP_A` 및 `PASSWORD_VERIFIER` 문제로 SRP 인증을 완료하면 이 함수는 인증 문제 생성 트리거를 호출하는 `CUSTOM_CHALLENGE`를 전달합니다. 이 시퀀스는 인증 생성 문제 예제와 함께 문제 3에 대한 CAPTCHA 문제와 문제 4에 대한 보안 질문을 제공합니다.

사용자가 CAPTCHA를 해결하고 보안 질문에 답변하면 이 함수는 사용자 풀이 토큰을 발급할 수 있는지 확인합니다. SRP 인증은 필요하지 않습니다. CAPTCHA 및 보안 질문을 문제 1과 2로 설정할 수도 있습니다. 인증 문제 정의 함수가 SRP 문제를 선언하지 않는 경우 사용자의 성공은 전적으로 사용자 지정 프롬프트에 대한 응답에 따라 결정됩니다.

Node.js

```
const handler = async (event) => {
  if (
    event.request.session.length === 1 &&
    event.request.session[0].challengeName === "SRP_A"
  ) {
```

```
    event.response.issueTokens = false;
    event.response.failAuthentication = false;
    event.response.challengeName = "PASSWORD_VERIFIER";
  } else if (
    event.request.session.length === 2 &&
    event.request.session[1].challengeName === "PASSWORD_VERIFIER" &&
    event.request.session[1].challengeResult === true
  ) {
    event.response.issueTokens = false;
    event.response.failAuthentication = false;
    event.response.challengeName = "CUSTOM_CHALLENGE";
  } else if (
    event.request.session.length === 3 &&
    event.request.session[2].challengeName === "CUSTOM_CHALLENGE" &&
    event.request.session[2].challengeResult === true
  ) {
    event.response.issueTokens = false;
    event.response.failAuthentication = false;
    event.response.challengeName = "CUSTOM_CHALLENGE";
  } else if (
    event.request.session.length === 4 &&
    event.request.session[3].challengeName === "CUSTOM_CHALLENGE" &&
    event.request.session[3].challengeResult === true
  ) {
    event.response.issueTokens = true;
    event.response.failAuthentication = false;
  } else {
    event.response.issueTokens = false;
    event.response.failAuthentication = true;
  }

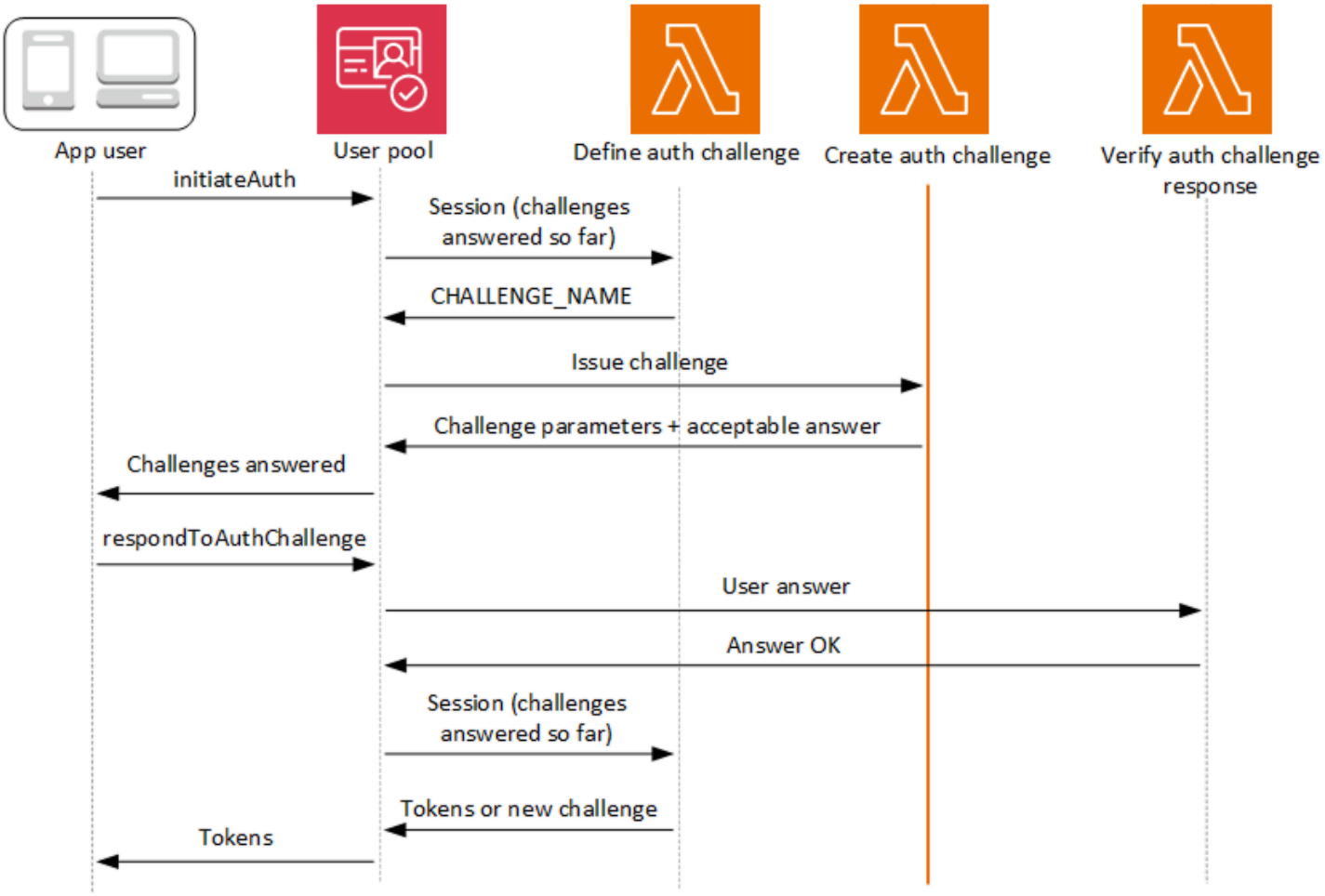
  return event;
};

export { handler };
```

인증 챌린지 생성 Lambda 트리거

인증 챌린지 생성 트리거는 인증 챌린지 정의 트리거가 선언한 각 챌린지의 세부 정보를 포함하는 Lambda 함수입니다. 인증 챌린지 정의 트리거에서 선언한 챌린지 이름을 처리하고 애플리케이션이 사용자에게 제공해야 하는 `publicChallengeParameters`를 반환합니다. 그런 다음 이 함수는 사용자 풀이 인증 챌린지 확인 트리거에 전달하는 챌린지에 대한 답변(`privateChallengeParameters`)을

사용자 풀에 제공합니다. 인증 챌린지 정의 트리거가 챌린지 시퀀스를 관리하는 경우 인증 챌린지 생성 트리거가 챌린지 콘텐츠를 관리합니다.



인증 문제 생성

> 인증 문제 정의 트리거의 일부로 사용자 지정 문제를 지정한 경우 Amazon Cognito가 인증 문제 정의 이후에 이 트리거를 호출합니다. 사용자 지정 인증 흐름을 생성합니다.

사용자에게 표시할 문제를 만들기 위해 이 Lambda 트리거가 호출됩니다. 이 Lambda 트리거에 대한 요청에는 `challengeName` 및 `session`이 포함됩니다. `challengeName`은 문자열이며 사용자에게 표시할 다음 챌린지의 이름입니다. 인증 문제 정의 Lambda 트리거에서 이 속성의 값이 설정됩니다.

모든 챌린지에 응답할 때까지 챌린지 루프가 반복됩니다.

주제

- 인증 챌린지 생성 Lambda 트리거 파라미터
- 인증 챌린지 생성 예제

인증 챌린지 생성 Lambda 트리거 파라미터

Amazon Cognito가 이 Lambda 함수에 전달하는 요청은 아래 파라미터와 Amazon Cognito가 모든 요청에 추가하는 공통 파라미터의 조합입니다.

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
        },
        "challengeName": "string",
        "session": [
            ChallengeResult,
            . . .
        ],
        "clientMetadata": {
            "string": "string",
            . . .
        },
        "userNotFound": boolean
    },
    "response": {
        "publicChallengeParameters": {
            "string": "string",
            . . .
        },
        "privateChallengeParameters": {
            "string": "string",
            . . .
        },
        "challengeMetadata": "string"
    }
}
```

인증 챌린지 생성 요청 파라미터

userAttributes

사용자 속성을 나타내는 하나 이상의 이름-값 페어입니다.

userNotFound

이 부울은 PreventUserExistenceErrors가 사용자 풀 클라이언트에 대해 ENABLED로 설정된 경우에 채워집니다.

challengeName

새로운 챌린지의 이름입니다.

세션

세션 요소는 각각 다음 요소를 포함하는 ChallengeResult 요소의 어레이입니다.

challengeName

챌린지 유형입니다. "CUSTOM_CHALLENGE", "PASSWORD_VERIFIER", "SMS_MFA", "DEVICE_SRP_AUTH", "DEVICE_PASSWORD_VERIFIER", "NEW_PASSWORD_REQUIRED" 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 사용됩니다.

clientMetadata

인증 문제 생성 트리거에 지정하는 Lambda 함수에 사용자 지정 입력으로 제공할 수 있는 하나 이상의 키-값 페어입니다. AdminRespondToAuthChallenge 및 RespondToAuthChallenge API 작업에서 ClientMetadata 파라미터를 사용하여 이 데이터를 Lambda 함수에 전달할 수 있습니다. 인증 문제 생성 함수를 호출하는 요청에는 AdminInitiateAuth 및 InitiateAuth API 작업의 ClientMetadata 파라미터에 전달된 데이터가 포함되지 않습니다.

인증 챌린지 생성 응답 파라미터

publicChallengeParameters

클라이언트 앱이 사용자에게 표시할 챌린지에 사용하기 위한 하나 이상의 키-값 페어입니다. 사용자에게 문제를 정확히 표시하는 데 필요한 모든 정보를 이 파라미터에 포함해야 합니다.

privateChallengeParameters

이 파라미터는 인증 문제 응답 확인 Lambda 트리거에만 사용됩니다. 챌린지에 대한 사용자의 응답을 확인하는 데 필요한 모든 정보를 이 파라미터에 포함해야 합니다. 즉, `publicChallengeParameters` 파라미터는 사용자에게 표시할 질문을 포함하고 `privateChallengeParameters`는 질문의 유효한 대답을 포함합니다.

challengeMetadata

이 항목이 사용자 지정 챌린지일 경우 사용자 지정 챌린지의 이름입니다.

인증 챌린지 생성 예제

이 함수에는 인증 문제 정의 예제의 문제 시퀀스에 해당하는 두 가지 사용자 지정 문제가 있습니다. 처음 두 가지 문제는 SRP 인증입니다. 세 번째 문제의 경우 이 함수는 문제 응답에서 애플리케이션에 CAPTCHA URL을 반환합니다. 애플리케이션은 지정된 URL에서 CAPTCHA를 렌더링하고 사용자의 입력을 반환합니다. CAPTCHA 이미지 URL이 퍼블릭 챌린지 파라미터에 `"captchaUrl"`로 추가되고 예상되는 답이 프라이빗 챌린지 파라미터에 추가됩니다.

네 번째 문제의 경우 이 함수는 보안 질문을 반환합니다. 애플리케이션이 질문을 렌더링하고 사용자에게 답변을 요청합니다. 사용자가 두 사용자 지정 문제를 모두 해결한 후 인증 문제 정의 트리거는 사용자 풀이 토큰을 발급할 수 있는지 확인합니다.

Node.js

```
const handler = async (event) => {
  if (event.request.challengeName !== "CUSTOM_CHALLENGE") {
    return event;
  }

  if (event.request.session.length === 2) {
    event.response.publicChallengeParameters = {};
    event.response.privateChallengeParameters = {};
    event.response.publicChallengeParameters.captchaUrl = "url/123.jpg";
    event.response.privateChallengeParameters.answer = "5";
  }

  if (event.request.session.length === 3) {
    event.response.publicChallengeParameters = {};
    event.response.privateChallengeParameters = {};
    event.response.publicChallengeParameters.securityQuestion =
```

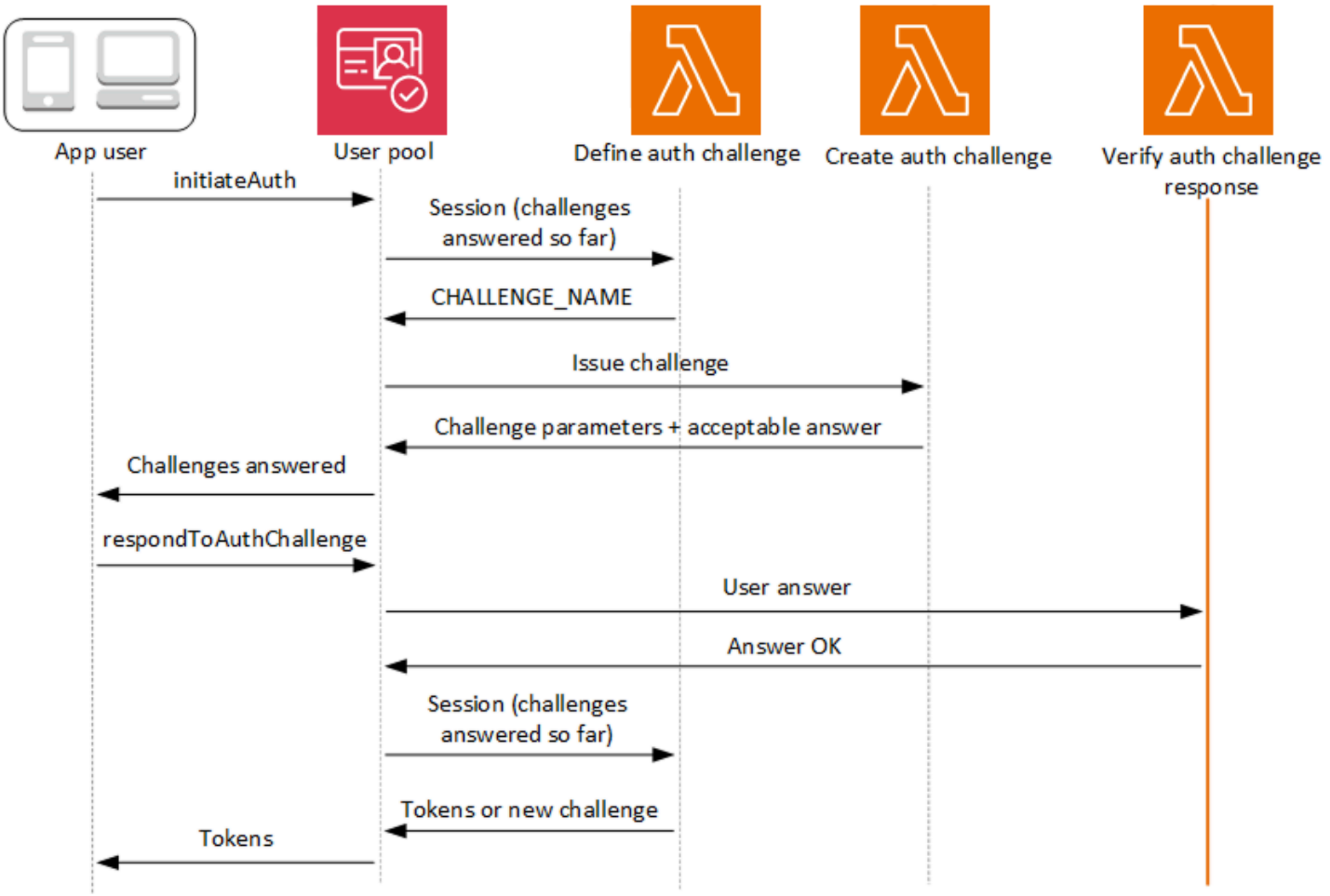
```
      "Who is your favorite team mascot?";
    event.response.privateChallengeParameters.answer = "Peccy";
  }

  return event;
};

export { handler };
```

인증 챌린지 확인 응답 Lambda 트리거

확인 인증 챌린지 트리거는 사용자가 제공한 응답을 알려진 응답과 비교하는 Lambda 함수입니다. 이 함수는 사용자가 챌린지에 올바르게 응답했는지 여부를 사용자 풀에 알려줍니다. 인증 챌린지 확인 트리거가 `true`의 `answerCorrect`로 응답하면 인증 시퀀스를 계속할 수 있습니다.



인증 문제 응답 확인

Amazon Cognito는 이 트리거를 호출하여 사용자 지정 인증 문제에 대한 사용자의 응답이 유효한지 여부를 확인합니다. 이것은 사용자 풀 사용자 지정 인증 흐름에 속합니다.

이 트리거에 대한 요청에는 `privateChallengeParameters` 및 `challengeAnswer` 파라미터가 포함됩니다. 인증 문제 생성 Lambda 트리거는 `privateChallengeParameters` 값을 반환하고 사용자의 예상 응답을 포함합니다. `challengeAnswer` 파라미터에는 챌린지에 대한 사용자의 응답이 포함됩니다.

응답에는 `answerCorrect` 속성이 포함됩니다. 사용자가 문제를 성공적으로 완료한 경우 Amazon Cognito는 속성 값을 `true`로 설정합니다. 사용자가 문제를 성공적으로 완료하지 못한 경우 Amazon Cognito는 속성 값을 `false`로 설정합니다.

사용자가 모든 문제에 응답할 때까지 문제 루프가 반복됩니다.

주제

- 인증 챌린지 확인 Lambda 트리거 파라미터
- 인증 챌린지 확인 응답 예제

인증 챌린지 확인 Lambda 트리거 파라미터

Amazon Cognito가 이 Lambda 함수에 전달하는 요청은 아래 파라미터와 Amazon Cognito가 모든 요청에 추가하는 공통 파라미터의 조합입니다.

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
        },
        "privateChallengeParameters": {
            "string": "string",
            . . .
        },
        "challengeAnswer": "string",
        "clientMetadata": {
            "string": "string",
```

```
            . . .
        },
        "userNotFound": boolean
    },
    "response": {
        "answerCorrect": boolean
    }
}
```

인증 챌린지 확인 요청 파라미터

userAttributes

이 파라미터에는 사용자 속성을 나타내는 하나 이상의 이름-값 페어가 포함됩니다.

userNotFound

Amazon Cognito에서 사용자 풀 클라이언트에 대해 `PreventUserExistenceErrors`를 `ENABLED`로 설정한 경우 Amazon Cognito는 이 부울을 채웁니다.

privateChallengeParameters

이 파라미터는 인증 문제 생성 트리거에서 가져옵니다. 사용자가 문제를 통과했는지 여부를 확인하려면 Amazon Cognito에서 이 파라미터를 사용자의 challengeAnswer과 비교합니다.

이 파라미터는 사용자의 문제에 대한 응답을 검증하는 데 필요한 모든 정보를 포함합니다. 이 정보에는 Amazon Cognito가 사용자에게 제시하는 질문(`publicChallengeParameters`) 및 질문에 대한 유효한 대답(`privateChallengeParameters`)이 포함됩니다. 인증 문제 응답 확인 Lambda 트리거만 이 파라미터를 사용합니다.

challengeAnswer

이 파라미터 값은 사용자의 문제에 대한 응답에 있는 대답입니다.

clientMetadata

이 파라미터는 인증 문제 확인 트리거에 대한 Lambda 함수에 사용자 지정 입력으로 제공될 수 있는 하나 이상의 키-값 페어를 포함합니다. 이 데이터를 Lambda 함수에 전달하려면 AdminRespondToAuthChallenge 및 RespondToAuthChallenge API 작업에서 ClientMetadata 파라미터를 사용합니다. Amazon Cognito는 승인 문제 확인 함수에 전달하는 요청에 있는 AdminInitiateAuth 및 InitiateAuth API 작업의 ClientMetadata 파라미터에서 전달된 데이터는 포함하지 않습니다.

인증 챌린지 확인 응답 파라미터

answerCorrect

사용자가 문제를 성공적으로 완료한 경우 Amazon Cognito는 이 파라미터를 `true`로 설정합니다. 사용자가 문제를 성공적으로 완료하지 못한 경우 Amazon Cognito는 이 파라미터를 `false`로 설정합니다.

인증 챌린지 확인 응답 예제

이 인증 확인 문제 기능은 사용자의 문제 응답이 예상 응답과 일치하는지 확인합니다. 사용자의 답변은 애플리케이션의 입력에 따라 정의되고, 선호하는 답변은 `privateChallengeParameters.answer`가 인증 문제 트리거 응답 생성의 응답에서 정의합니다. 정답과 주어진 답변 모두이 함수에 대한 입력 이벤트의 일부입니다.

이 예에서, 사용자의 응답과 예상 응답이 일치하는 경우 Amazon Cognito는 `answerCorrect` 파라미터를 `true`로 설정합니다.

Node.js

```
const handler = async (event) => {
  if (
    event.request.privateChallengeParameters.answer ===
    event.request.challengeAnswer
  ) {
    event.response.answerCorrect = true;
  } else {
    event.response.answerCorrect = false;
  }

  return event;
};

export { handler };
```

사전 토큰 생성 Lambda 트리거

토큰 생성 전에 Amazon Cognito가 이 트리거를 간접 호출하기 때문에 사용자 풀 토큰 신청을 사용자 지정할 수 있습니다. 버전 1 또는 `V1_0` 사전 토큰 생성 트리거 이벤트의 기본 기능을 사용하여 ID 토큰을 사용자 지정할 수 있습니다. Essentials 또는 Plus 기능 플랜이 있는 사용자 풀에서는 버전

2를 생성하거나 액세스 토큰 사용자 지정을 통해 이벤트를 `V2_0` 트리거할 수 있으며, 버전 3 또는 트리거 이벤트는 클라이언트 자격 증명 권한 부여와 `GetClientToken` API 작업을 모두 포함하여 M2M(Machinemachine-to-machine) 권한 부여를 위한 액세스 토큰 사용자 지정을 통해 이벤트를 `V3_0` 트리거할 수 있습니다.

Amazon Cognito는 ID 토큰에 쓸 데이터와 함께 `V1_0` 이벤트를 요청으로서 함수에 보냅니다. `V2_0` 또는 `V3_0` 이벤트는 Amazon Cognito가 ID 및 액세스 토큰 모두에 쓸 데이터가 포함된 단일 요청입니다. 두 토큰을 모두 사용자 지정하려면 트리거 버전 2 또는 3을 사용하도록 함수를 업데이트하고 동일한 응답에서 두 토큰에 대한 데이터를 전송해야 합니다.

Amazon Cognito는 인간 사용자가 사용자 풀에 자격 증명을 제공한 사용자 인증의 액세스 토큰에 버전 2 이벤트 응답을 적용합니다. 버전 3 이벤트 응답은 사용자 인증 및 시스템 인증의 액세스 토큰에 적용되며, 자동 시스템은 앱 클라이언트 보안 암호로 액세스 토큰 요청을 승인합니다. 결과 액세스 토큰의 상황을 제외하면 버전 2와 3 이벤트는 동일합니다.

이 Lambda 트리거는 Amazon Cognito가 앱에 클레임을 발행하기 전에 ID 및 액세스 토큰의 일부 클레임을 추가, 제거 및 수정할 수 있습니다. 이 기능을 사용하려면 Amazon Cognito 사용자 풀 콘솔에서 Lambda 함수를 연결하거나 AWS Command Line Interface (AWS CLI)를 통해 사용자 풀 `LambdaConfig`를 업데이트합니다.

이벤트 버전

사용자 풀은 다양한 버전의 사전 토큰 생성 트리거 이벤트를 Lambda 함수에 전달할 수 있습니다. `V1_0` 트리거는 ID 토큰 수정을 위한 파라미터를 제공합니다. `V2_0` 또는 `V3_0` 트리거는 다음에 대한 파라미터를 제공합니다.

1. `V1_0` 트리거의 함수.
2. 액세스 토큰을 사용자 지정하는 기능.
3. 복잡한 데이터 유형을 ID 및 액세스 토큰 클레임 값에 전달하는 기능:
   - 문자열
   - 숫자
   - 부울
   - 문자열, 숫자, 부울 또는 이러한 문자열의 조합 배열
   - JSON

> **Note**
>
> ID 토큰에서 `phone_number_verified`, `email_verified`, `updated_at` 및 `address`를 제외한 복잡한 개체를 클레임 값에 채울 수 있습니다.

사용자 풀은 기본적으로 `V1_0` 이벤트를 전달합니다. `V2_0` 이벤트를 보내도록 사용자 풀을 구성하려면 Amazon Cognito 콘솔에서 트리거를 구성할 때 사용자 ID에 대한 기본 기능 + 액세스 토큰 사용자 지정의 이벤트 트리거 버전을 선택합니다. `V3_0` 이벤트를 생성하려면 사용자 및 머신 ID에 대한 기본 기능 + 액세스 토큰 사용자 지정을 선택합니다. UpdateUserPool 또는 CreateUserPool API 요청의 LambdaConfig 파라미터에서 `LambdaVersion` 값을 설정할 수도 있습니다. 이벤트 버전 1, 2, 3은 Essentials 및 Plus 기능 플랜에서 사용할 수 있습니다. 버전 3 이벤트에 대한 M2M 작업에는 월간 활성 사용자(MAU) 공식과 별도의 요금 구조가 있습니다. 자세한 내용은 Amazon Cognito 요금을 참조하세요.

> **Note**
>
> 2024년 11월 22일 18:00 GMT 이전에 고급 보안 기능 옵션으로 작동하고 Lite 기능 계층에 남아 있는 사용자 풀은 사전 토큰 생성 트리거의 이벤트 버전 1 및 2에 액세스할 수 있습니다. 고급 보안 기능이 없는 이 레거시 계층의 사용자 풀은 이벤트 버전 1에 액세스할 수 있습니다. 버전 3은 Essentials 및 Plus에서만 사용할 수 있습니다.

클레임 및 범위 참조

Amazon Cognito는 액세스 및 자격 증명 토큰에서 추가, 수정 또는 억제할 수 있는 클레임 및 범위를 제한합니다. 다음 표에서는 Lambda 함수가 수정할 수 있는 클레임과 수정할 수 없는 클레임 및 클레임의 존재 또는 값에 영향을 미치는 트리거 이벤트 파라미터를 설명합니다.

Claim	기본 토큰 유형	추가할 수 있나요?	수정할 수 있나요?	억제할 수 있나요?	이벤트 파라미터 - 추가 또는 수정	이벤트 파라미터 - 억제	자격 증명 유형	이벤트 버전
사용자 풀 토큰 스키	없음	예	예	해당 사항 없음	`claimsTo/ ddOrOver: ide`	`claimsTo: uppress`	사용자, 머신[1]	모두[2]

Claim	기본 토큰 유형	추가할 수 있나요?	수정할 수 있나요?	억제할 수 있나요?	이벤트 파라미터 - 추가 또는 수정	이벤트 파라미터 - 억제	자격 증명 유형	이벤트 버전
마에 없는 클레임								
scope	액세스	예	예	예	scopesToA dd	scopesTo uppress	사용자, 머신[1]	v2_0, v3_0
cognito:g roups	ID, 액세스	예	예	예	groupsTo verride	claimsTo uppress	User	모두[2]
cognito:p referred_ role	ID	예	예	예	preferre Role	claimsTo uppress [3]	User	모두
cognito:r oles	ID	예	예	예	iamRoles oOverrid	claimsTo uppress [3]	User	모두
cognito:u sername	ID	아니요	아니요	아니요	해당 사항 없음	해당 사항 없음	User	해당 사항 없음
cognito: 접두사가 있는 기타 모든 클레임	없음	아니요	아니요	아니요	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음
username	액세스	아니요	아니요	아니요	해당 사항 없음	해당 사항 없음	User	v2_0, v3_0
sub	ID, 액세스	아니요	아니요	아니요	해당 사항 없음	해당 사항 없음	User	해당 사항 없음

Claim	기본 토큰 유형	추가할 수 있나요?	수정할 수 있나요?	억제할 수 있나요?	이벤트 파라미터 - 추가 또는 수정	이벤트 파라미터 - 억제	자격 증명 유형	이벤트 버전
표준 OIDC 속성	ID	예	예	예	claimsTo/ ddOrOver: ide	claimsTo: uppress	User	모두
custom: 속성	ID	예	예	예	claimsTo/ ddOrOver: ide	claimsTo: uppress	User	모두
dev: 속성	ID	아니요	아니요	예	해당 사항 없음	claimsTo: uppress	User	모두
identiti s	ID	아니요	아니요	아니요	해당 사항 없음	해당 사항 없음	User	해당 사항 없음
aud[4]	ID	아니요	아니요	아니요	해당 사항 없음	해당 사항 없음	사용자, 머신	해당 사항 없음
client_i	액세스	아니요	아니요	아니요	해당 사항 없음	해당 사항 없음	사용자, 머신	해당 사항 없음
event_id	액세스	아니요	아니요	아니요	해당 사항 없음	해당 사항 없음	사용자, 머신	해당 사항 없음
device_k y	액세스	아니요	아니요	아니요	해당 사항 없음	해당 사항 없음	User	해당 사항 없음
version	액세스	아니요	아니요	아니요	해당 사항 없음	해당 사항 없음	사용자, 머신	해당 사항 없음
acr	ID, 액세스	아니요	아니요	아니요	해당 사항 없음	해당 사항 없음	사용자, 머신	해당 사항 없음

Claim	기본 토큰 유형	추가할 수 있나요?	수정할 수 있나요?	억제할 수 있나요?	이벤트 파라미터 - 추가 또는 수정	이벤트 파라미터 - 억제	자격 증명 유형	이벤트 버전
amr	ID, 액세스	아니요	아니요	아니요	해당 사항 없음	해당 사항 없음	사용자, 머신	해당 사항 없음
at_hash	ID	아니요	아니요	아니요	해당 사항 없음	해당 사항 없음	사용자, 머신	해당 사항 없음
auth_time	ID, 액세스	아니요	아니요	아니요	해당 사항 없음	해당 사항 없음	사용자, 머신	해당 사항 없음
azp	ID, 액세스	아니요	아니요	아니요	해당 사항 없음	해당 사항 없음	사용자, 머신	해당 사항 없음
exp	ID, 액세스	아니요	아니요	아니요	해당 사항 없음	해당 사항 없음	사용자, 머신	해당 사항 없음
iat	ID, 액세스	아니요	아니요	아니요	해당 사항 없음	해당 사항 없음	사용자, 머신	해당 사항 없음
iss	ID, 액세스	아니요	아니요	아니요	해당 사항 없음	해당 사항 없음	사용자, 머신	해당 사항 없음
jti	ID, 액세스	아니요	아니요	아니요	해당 사항 없음	해당 사항 없음	사용자, 머신	해당 사항 없음
nbf	ID, 액세스	아니요	아니요	아니요	해당 사항 없음	해당 사항 없음	사용자, 머신	해당 사항 없음

Claim	기본 토큰 유형	추가할 수 있나요?	수정할 수 있나요?	억제할 수 있나요?	이벤트 파라미터 - 추가 또는 수정	이벤트 파라미터 - 억제	자격 증명 유형	이벤트 버전
nonce	ID, 액세스	아니요	아니요	아니요	해당 사항 없음	해당 사항 없음	사용자, 머신	해당 사항 없음
origin_ji	ID, 액세스	아니요	아니요	아니요	해당 사항 없음	해당 사항 없음	사용자, 머신	해당 사항 없음
token_use	ID, 액세스	아니요	아니요	아니요	해당 사항 없음	해당 사항 없음	사용자, 머신	해당 사항 없음

[1] 머신 ID에 대한 액세스 토큰은 트리거 입력 이벤트의 `v3_0`에서만 사용할 수 있습니다. 이벤트 버전 3은 Essentials 및 Plus 기능 계층에서만 사용할 수 있습니다. Lite 계층의 사용자 풀은 `v1_0` 이벤트를 수신할 수 있습니다. 고급 보안 기능이 있는 Lite 계층의 사용자 풀은 `v1_0` 및 `v2_0` 이벤트를 수신할 수 있습니다.

[2] ID 토큰만을 위한 이벤트 버전 `v1_0`, ID 및 액세스 토큰을 위한 이벤트 버전 `v2_0`, 머신 ID에 대한 기능이 있는 ID 및 액세스 토큰을 위한 이벤트 버전 `v3_0`으로 사전 토큰 생성 트리거를 구성합니다.

[3] `cognito:preferred_role` 및 `cognito:roles` 클레임을 억제하려면 `claimsToSuppress`에 `cognito:groups`를 추가합니다.

[4] 액세스 토큰에 `aud` 클레임을 추가할 수 있지만 해당 값은 현재 세션의 앱 클라이언트 ID와 일치해야 합니다. `event.callerContext.clientId`의 요청 이벤트에서 클라이언트 ID를 파생할 수 있습니다.

자격 증명 토큰 사용자 지정

사전 토큰 생성 Lambda 트리거의 모든 이벤트 버전을 사용하면 사용자 풀에서 ID 토큰의 내용을 사용자 지정할 수 있습니다. ID 토큰은 웹 또는 모바일 앱에 로그인하기 위한 신뢰할 수 있는 ID 소스의 사용자 속성을 제공합니다. ID 토큰에 대한 자세한 내용은 자격 증명(ID) 토큰 이해 섹션을 참조하세요.

사전 토큰 생성 Lambda 트리거를 ID 토큰과 함께 사용하는 방법은 다음과 같습니다.

- 사용자가 자격 증명 풀에서 요청하는 IAM 역할을 런타임에서 변경합니다.
- 외부 소스의 사용자 속성을 추가합니다.
- 기존 사용자 속성 값을 추가하거나 바꿉니다.
- 사용자의 승인된 범위와 앱 클라이언트에 부여한 속성에 대한 읽기 액세스 권한으로 인해 앱에 달리 전달될 수 있는 사용자 속성의 공개를 차단합니다.

액세스 토큰 사용자 지정

사전 토큰 생성 Lambda 트리거의 이벤트 버전 2와 3을 사용하면 사용자 풀에서 액세스 토큰의 내용을 사용자 지정할 수 있습니다. 액세스 토큰은 Amazon Cognito 토큰 인증 API 작업 및 서드 파티 API와 같은 액세스 보호 리소스에서 정보를 검색할 수 있는 권한을 사용자에게 부여합니다. M2M(Machinemachine-to-machine) 권한 부여의 경우 액세스 토큰이 클라이언트 자격 증명 권한 부여에서 왔든 `GetClientToken` API 작업에서 왔든 관계없이 Amazon Cognito는 사용자 풀이 버전 3(`V3_0`) 이벤트에 대해 구성된 경우에만 사전 토큰 생성 트리거를 호출합니다. 액세스 토큰에 대한 자세한 내용은 액세스 토큰 이해 섹션을 참조하세요.

사전 토큰 생성 Lambda 트리거를 액세스 토큰과 함께 사용하는 방법은 다음과 같습니다.

- `scope` 클레임에 범위를 추가하거나 억제합니다. 예를 들어 범위 `aws.cognito.signin.user.admin`만 할당하는 Amazon Cognito 사용자 풀 API 인증을 통해 생성된 액세스 토큰에 범위를 추가할 수 있습니다.
- 사용자 풀 그룹의 사용자 멤버십을 변경합니다.
- Amazon Cognito 액세스 토큰에 아직 존재하지 않는 클레임을 추가합니다.
- 앱으로 달리 전달될 수 있는 클레임의 공개를 차단합니다.

사용자 풀의 액세스 사용자 지정을 지원하려면 트리거 요청의 업데이트된 버전을 생성하도록 사용자 풀을 구성해야 합니다. 다음 절차에 나오는 대로 사용자 풀을 업데이트합니다.

AWS Management Console

사전 토큰 생성 Lambda 트리거에서 액세스 토큰 사용자 지정의 지원

1. Amazon Cognito 콘솔로 이동한 다음 사용자 풀(User Pools)을 선택합니다.
2. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
3. 확장 메뉴를 선택하고 Lambda 트리거를 찾습니다.
4. 사전 토큰 생성 트리거를 추가 또는 편집합니다.

5. Lambda 함수 할당에서 Lambda 함수를 선택합니다.
6. 사용자 ID에 대한 기본 기능 + 액세스 토큰 사용자 지정 또는 사용자 및 머신 ID에 대한 기본 기능 + 액세스 토큰 사용자 지정의 트리거 이벤트 버전을 선택합니다. 이 설정은 Amazon Cognito가 함수에 보내는 요청 파라미터를 업데이트하여 액세스 토큰 사용자 지정을 위한 필드를 포함합니다.

User pools API

사전 토큰 생성 Lambda 트리거에서 액세스 토큰 사용자 지정의 지원

CreateUserPool 또는 UpdateUserPool API 요청을 생성합니다. 기본값으로 설정하지 않으려는 모든 파라미터의 값을 지정해야 합니다. 자세한 내용은 사용자 풀 및 앱 클라이언트 구성 업데이트 섹션을 참조하세요.

요청의 `LambdaVersion` 파라미터에 다음 콘텐츠를 포함합니다. `V2_0`의 `LambdaVersion` 값은 사용자 풀이 액세스 토큰에 대한 파라미터를 추가하고 변경 사항을 적용하도록 합니다. `V3_0`의 `LambdaVersion` 값은 `V2_0`과 동일한 이벤트를 생성하지만 또한 사용자 풀이 M2M 액세스 토큰에 변경 사항을 적용하도록 합니다. 특정 함수 버전을 간접 호출하려면 함수 버전을 `LambdaArn`의 값으로 포함하는 Lambda 함수 ARN을 사용하세요.

```
"PreTokenGenerationConfig": {
   "LambdaArn": "arn:aws:lambda:us-west-2:123456789012:function:MyFunction",
   "LambdaVersion": "V3_0"
},
```

M2M(머신 간) 클라이언트 자격 증명을 위한 클라이언트 메타데이터

M2M 요청에서 클라이언트 메타데이터를 전달할 수 있습니다. 클라이언트 메타데이터는 사전 토큰 생성 Lambda 트리거의 결과에 기여할 수 있는 사용자 또는 애플리케이션 환경의 추가 정보입니다. 사용자 보안 주체를 사용한 인증 작업에서 AdminRespondToAuthChallenge 및 RespondToAuthChallenge API 요청 본문의 사전 토큰 생성 트리거에 클라이언트 메타데이터를 전달할 수 있습니다. 애플리케이션은 Token 엔드포인트에 대한 직접 요청으로 M2M에 대한 액세스 토큰 생성을 위한 흐름을 수행하므로 모델이 다릅니다. 클라이언트 자격 증명에 대한 토큰 요청의 POST 본문에서 클라이언트 메타데이터 객체 URL 인코딩(`x-www-form-urlencoded`)이 포함된 `aws_client_metadata` 파라미터를 문자열에 전달합니다. 요청 예시는 기본 권한이 있는 클라이언트 자격 증명 섹션을 참조하세요. 다음은 키-값 페어 `{"environment": "dev", "language": "en-US"}`를 전달하는 예제 파라미터입니다.

```
aws_client_metadata=%7B%22environment%22%3A%20%22dev%22,%20%22language%22%3A%20%22en-US%22%7D
```

추가 리소스

- Amazon Cognito 사용자 풀에서 액세스 토큰을 사용자 지정하는 방법

주제

- 사전 토큰 생성 Lambda 트리거 소스
- 사전 토큰 생성 Lambda 트리거 파라미터
- 사전 토큰 트리거 이벤트 버전 2 예시: 클레임, 범위 및 그룹 추가 및 억제
- 사전 토큰 생성 이벤트 버전 2 예제: 복잡한 개체로 클레임 추가
- 사전 토큰 생성 이벤트 버전 1 예시: 새 클레임 추가 및 기존 클레임 억제
- 사전 토큰 생성 이벤트 버전 1 예시: 사용자의 그룹 멤버십 수정

사전 토큰 생성 Lambda 트리거 소스

triggerSource 값	Event
`TokenGeneration_HostedAuth`	Amazon Cognito 관리 로그인 페이지에서 인증 중에 호출됩니다.
`TokenGeneration_Authentication`	사용자 인증 흐름이 완료된 이후 호출됩니다.
`TokenGeneration_NewPasswordChallenge`	사용자가 관리자에 의해 생성된 후 호출됩니다. 이 흐름은 사용자가 임시 비밀번호를 변경해야 할 때 호출됩니다.
`TokenGeneration_ClientCredentials`	M2M 클라이언트 자격 증명 부여 후 호출됩니다. 사용자 풀은 이벤트 버전이 `V3_0`인 경우에만 이 이벤트를 전송합니다.
`TokenGeneration_AuthenticateDevice`	사용자 디바이스 인증 마지막에 호출됩니다.

triggerSource 값	Event
TokenGeneration_RefreshTokens	사용자가 자격 증명 및 액세스 토큰을 새로 고치려 할 때 호출됩니다.

사전 토큰 생성 Lambda 트리거 파라미터

Amazon Cognito가 이 Lambda 함수에 전달하는 요청은 아래 파라미터와 Amazon Cognito가 모든 요청에 추가하는 공통 파라미터의 조합입니다. 사전 토큰 생성 Lambda 트리거를 사용자 풀에 추가할 때 트리거 버전을 선택할 수 있습니다. 이 버전은 Amazon Cognito가 액세스 토큰 사용자 지정을 위한 추가 파라미터와 함께 Lambda 함수에 요청을 전달할지 여부를 결정합니다.

Version one

버전 1 토큰은 ID 토큰에서 그룹 멤버십, IAM 역할 및 새 클레임을 설정할 수 있습니다. 그룹 멤버십 재정의는 액세스 토큰의 `cognito:groups` 클레임에도 적용됩니다.

```
{
    "request": {
        "userAttributes": {"string": "string"},
        "groupConfiguration": {
                "groupsToOverride": [
                    "string",
                    "string"
                ],
                "iamRolesToOverride": [
                    "string",
                    "string"
                ],
                "preferredRole": "string"
        },
        "clientMetadata": {"string": "string"}
    },
    "response": {
        "claimsOverrideDetails": {
            "claimsToAddOrOverride": {"string": "string"},
            "claimsToSuppress": [
                "string",
                "string"
            ],
            "groupOverrideDetails": {
```

```
                "groupsToOverride": [
                    "string",
                    "string"
                ],
                "iamRolesToOverride": [
                    "string",
                    "string"
                ],
                "preferredRole": "string"
            }
        }
    }
}
```

Versions two and three

버전 2와 3 요청 이벤트는 액세스 토큰을 사용자 지정하는 필드를 추가합니다. 사용자 풀은 버전 3 이벤트의 변경 사항을 시스템 ID에 대한 액세스 토큰에 적용합니다. 이러한 버전에서는 응답 객체에서 복잡한 `claimsToOverride` 데이터 유형에 대한 지원도 추가되었습니다. Lambda 함수는 값 `claimsToOverride`에서 다음 형식의 데이터를 반환할 수 있습니다.

- 문자열
- 숫자
- 부울
- 문자열, 숫자, 부울 또는 이러한 문자열의 조합 배열
- JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string"
        },
        "scopes": ["string", "string"],
        "groupConfiguration": {
            "groupsToOverride": ["string", "string"],
            "iamRolesToOverride": ["string", "string"],
            "preferredRole": "string"
        },
        "clientMetadata": {
            "string": "string"
```

```
        }
    },
    "response": {
        "claimsAndScopeOverrideDetails": {
            "idTokenGeneration": {
                "claimsToAddOrOverride": {
                    "string": [accepted datatype]
                },
                "claimsToSuppress": ["string", "string"]
            },
            "accessTokenGeneration": {
                "claimsToAddOrOverride": {
                    "string": [accepted datatype]
                },
                "claimsToSuppress": ["string", "string"],
                "scopesToAdd": ["string", "string"],
                "scopesToSuppress": ["string", "string"]
            },
            "groupOverrideDetails": {
                "groupsToOverride": ["string", "string"],
                "iamRolesToOverride": ["string", "string"],
                "preferredRole": "string"
            }
        }
    }
}
```

사전 토큰 생성 요청 파라미터

이름	설명	최소 트리거 이벤트 버전
userAttributes	사용자 풀에 있는 사용자의 사용자 프로필 속성입니다.	1
groupConfiguration	현재 그룹 구성을 포함하는 입력 객체입니다. 이 객체에는 `groupsToOverride` , `iamRolesToOverride` 및 `preferredRole` 이 포함됩니다.	1
groupsToOverride	사용자가 속한 사용자 풀 그룹입니다.	1

이름	설명	최소 트리거 이벤트 버전
iamRolesToOverride	사용자 풀 그룹을 AWS Identity and Access Management (IAM) 역할과 연결할 수 있습니다. 이 요소는 사용자가 속한 그룹의 모든 IAM 역할 목록입니다.	1
preferredRole	사용자 풀 그룹의 우선 순위를 설정할 수 있습니다. 이 요소에는 `groupsToOverride` 요소에서 우선 순위가 가장 높은 그룹의 IAM 역할 이름이 포함됩니다.	1
clientMetadata	사용자가 지정하고 사전 토큰 생성 트리거에 Lambda 함수에 대한 사용자 지정 입력으로 제공할 수 있는 하나 이상의 키-값 페어입니다. 이 데이터를 Lambda 함수에 전달하려면 AdminRespondToAuthChallenge 및 RespondToAuthChallenge API 작업에서 ClientMetadata 파라미터를 사용합니다. Amazon Cognito는 사전 토큰 생성 함수로 전달하는 요청 내 AdminInitiateAuth 및 InitiateAuth API 작업의 `ClientMetadata` 파라미터의 데이터는 포함하지 않습니다.	1
scopes	액세스 토큰 범위입니다. 액세스 토큰에 표시되는 범위는 사용자가 요청하고 앱 클라이언트에 발급을 승인한 사용자 풀 표준 및 사용자 지정 범위입니다.	2

사전 토큰 생성 응답 파라미터

이름	설명	최소 트리거 이벤트 버전
claimsOverrideDetails	A container for all elements in a `V1_0` trigger event.	1

이름	설명	최소 트리거 이벤트 버전
claimsAndScopeOverrideDetails	`V2_0` 또는 `V3_0` 트리거 이벤트의 모든 요소를 담는 컨테이너입니다.	2
idTokenGeneration	사용자 ID 토큰에서 재정의, 추가 또는 차단하려는 클레임입니다. 이 상위 ID 토큰 사용자 지정 값은 버전 2 이벤트에만 나타나지만 하위 요소는 버전 1 이벤트에 나타납니다.	2
accessTokenGeneration	사용자 액세스 토큰에서 재정의, 추가 또는 차단하려는 클레임 및 범위입니다. 토큰 사용자 지정 값에 액세스하는 이 상위 값은 이벤트 버전 2 이상에만 나타납니다.	2
claimsToAddOrOverride	추가 또는 수정하려는 하나 이상의 클레임 및 해당 값이 표시된 맵입니다. 그룹 관련 클레임의 경우 대신 `groupOverrideDetails` 를 사용합니다. 이벤트 버전 2 이상에서 이 요소는 `accessTokenGeneration` 및 `idTokenGeneration` 에 모두 나타납니다.	1*
claimsToSuppress	Amazon Cognito가 차단하길 원하는 클레임의 목록입니다. 함수가 신청 값을 억제하고 대체하는 경우 Amazon Cognito는 신청을 억제합니다. 이벤트 버전 2 이상에서 이 요소는 `accessTokenGeneration` 및 `idTokenGeneration` 에 모두 나타납니다.	1

이름	설명	최소 트리거 이벤트 버전
groupOverrideDetails	현재 그룹 구성을 포함하는 입력 객체입니다. 이 객체에는 `groupsToOverride` , `iamRolesToOverride` 및 `preferredRole` 이 포함됩니다. 함수가 `groupOverrideDetails` 객체를 사용자가 제공하는 객체로 바꿉니다. 응답에서 비어 있는 객체 또는 Null 객체를 제공하면 Amazon Cognito가 그룹을 억제합니다. 기존 그룹 구성을 동일하게 유지하려면 요청의 `groupConfiguration` 객체 값을 응답의 `groupOverrideDetails` 객체로 복사합니다. 그런 다음 해당 객체를 서비스로 다시 전달합니다. Amazon Cognito ID 및 액세스 토큰 둘 다에 `cognito:groups` 클레임이 포함되어 있습니다. `groupOverrideDetails` 객체는 액세스 토큰과 ID 토큰의 `cognito:groups` 클레임을 대체합니다. 그룹 재정의는 버전 1 이벤트가 수행할 수 있는 액세스 토큰에 대한 유일한 변경 사항입니다.	1
scopesToAdd	사용자의 액세스 토큰의 `scope` 클레임에 추가하려는 범위 목록입니다. 하나 이상의 공백 문자가 포함된 범위 값은 추가할 수 없습니다.	2
scopesToSuppress	사용자의 액세스 토큰의 `scope` 클레임에서 제거하려는 범위 목록입니다.	2

* 버전 1 이벤트에 대한 응답 객체는 문자열을 반환할 수 있습니다. 버전 2 및 3 이벤트에 대한 응답 객체는 복잡한 객체를 반환할 수 있습니다.

사전 토큰 트리거 이벤트 버전 2 예시: 클레임, 범위 및 그룹 추가 및 억제

이 예시에서는 사용자 토큰을 다음과 같이 수정합니다.

1. ID 토큰에서 `family_name`을 `Doe`와 동일하게 설정합니다.
2. ID 토큰에 `email` 및 `phone_number` 클레임이 나타나지 않도록 합니다.

3. ID 토큰 `cognito:roles` 클레임을 `"arn:aws:iam::123456789012:role\/sns_callerA","arn:aws:iam::123456789012:role\/sns_callerC","arn:aws:iam::123456789012:role\/sns_callerB"`로 설정합니다.
4. ID 토큰 `cognito:preferred_role` 클레임을 `arn:aws:iam::123456789012:role/sns_caller`로 설정합니다.
5. 액세스 토큰에 범위 `openid`, `email`, `solar-system-data/asteroids.add`를 추가합니다.
6. 액세스 토큰에서 범위 `phone_number` 및 `aws.cognito.signin.user.admin`을 억제합니다. `phone_number`를 제거하면 `userInfo`에서 사용자 전화 번호를 검색할 수 없습니다. `aws.cognito.signin.user.admin`를 제거하면 Amazon Cognito 사용자 풀 API로 자신의 프로필을 읽고 수정하려는 사용자의 API 요청이 방지됩니다.

> **Note**
>
> 액세스 토큰의 나머지 범위에 `openid` 및 하나 이상의 표준 범위가 포함된 경우에만 범위에서 `phone_number`를 제거하면 사용자 전화 번호를 검색할 수 없습니다. 자세한 내용은 범위에 대한 정보 섹션을 참조하세요.

7. ID 및 액세스 토큰 `cognito:groups` 클레임을 `"new-group-A","new-group-B","new-group-C"`로 설정합니다.

JavaScript

```
export const handler = function(event, context) {
  event.response = {
    "claimsAndScopeOverrideDetails": {
      "idTokenGeneration": {
        "claimsToAddOrOverride": {
          "family_name": "Doe"
        },
        "claimsToSuppress": [
          "email",
          "phone_number"
        ]
      },
      "accessTokenGeneration": {
        "scopesToAdd": [
          "openid",
          "email",
          "solar-system-data/asteroids.add"
```

```
        ],
        "scopesToSuppress": [
          "phone_number",
          "aws.cognito.signin.user.admin"
        ]
      },
      "groupOverrideDetails": {
        "groupsToOverride": [
          "new-group-A",
          "new-group-B",
          "new-group-C"
        ],
        "iamRolesToOverride": [
          "arn:aws:iam::123456789012:role/new_roleA",
          "arn:aws:iam::123456789012:role/new_roleB",
          "arn:aws:iam::123456789012:role/new_roleC"
        ],
        "preferredRole": "arn:aws:iam::123456789012:role/new_role"
      }
    }
  };
  // Return to Amazon Cognito
  context.done(null, event);
};
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그런 다음 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. Lambda 콘솔에서 해당 Lambda 트리거와 관련 있는 데이터로 테스트 이벤트를 설정할 수 있습니다. 다음은 이 코드 샘플의 테스트 이벤트입니다.

JSON

```
{
    "version": "2",
    "triggerSource": "TokenGeneration_Authentication",
    "region": "us-east-1",
    "userPoolId": "us-east-1_EXAMPLE",
    "userName": "JaneDoe",
    "callerContext": {
        "awsSdkVersion": "aws-sdk-unknown-unknown",
        "clientId": "1example23456789"
    },
    "request": {
```

```
        "userAttributes": {
            "sub": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "cognito:user_status": "CONFIRMED",
            "email_verified": "true",
            "phone_number_verified": "true",
            "phone_number": "+12065551212",
            "family_name": "Zoe",
            "email": "Jane.Doe@example.com"
        },
        "groupConfiguration": {
            "groupsToOverride": ["group-1", "group-2", "group-3"],
            "iamRolesToOverride": ["arn:aws:iam::123456789012:role/sns_caller1",
 "arn:aws:iam::123456789012:role/sns_caller2", "arn:aws:iam::123456789012:role/
sns_caller3"],
            "preferredRole": ["arn:aws:iam::123456789012:role/sns_caller"]
        },
        "scopes": [
            "aws.cognito.signin.user.admin", "openid", "email", "phone"
        ]
    },
    "response": {
        "claimsAndScopeOverrideDetails": []
    }
}
```

사전 토큰 생성 이벤트 버전 2 예제: 복잡한 개체로 클레임 추가

이 예시에서는 사용자 토큰을 다음과 같이 수정합니다.

1. ID 토큰에 숫자, 문자열, 부울 및 JSON 유형의 클레임을 추가합니다. 이는 버전 2 트리거 이벤트가 ID 토큰에 사용할 수 있도록 하는 유일한 변경 사항입니다.
2. 번호, 문자열, 부울 및 JSON 유형의 클레임을 액세스 토큰에 추가합니다.
3. 액세스 토큰에 세 가지 범위를 추가합니다.
4. ID 및 액세스 토큰에서 `email` 클레임을 억제합니다.
5. 액세스 토큰에서 `aws.cognito.signin.user.admin` 범위를 억제합니다.

JavaScript

```
export const handler = function(event, context) {
```

```
    var scopes = ["MyAPI.read", "MyAPI.write", "MyAPI.admin"]
    var claims = {}
    claims["aud"]= event.callerContext.clientId;
    claims["booleanTest"] = false;
    claims["longTest"] = 9223372036854775807;
    claims["exponentTest"] = 1.7976931348623157E308;
    claims["ArrayTest"] = ["test", 9223372036854775807, 1.7976931348623157E308,
true];
    claims["longStringTest"] = "\{\
        \"first_json_block\": \{\
            \"key_A\": \"value_A\",\
            \"key_B\": \"value_B\"\
        \},\
        \"second_json_block\": \{\
            \"key_C\": \{\
                \"subkey_D\": [\
                    \"value_D\",\
                    \"value_E\"\
                ],\
                \"subkey_F\": \"value_F\"\
            \},\
            \"key_G\": \"value_G\"\
        \}\
    \}";
    claims["jsonTest"] = {
     "first_json_block": {
      "key_A": "value_A",
      "key_B": "value_B"
     },
     "second_json_block": {
      "key_C": {
       "subkey_D": [
        "value_D",
        "value_E"
       ],
       "subkey_F": "value_F"
      },
      "key_G": "value_G"
     }
    };
    event.response = {
        "claimsAndScopeOverrideDetails": {
            "idTokenGeneration": {
                "claimsToAddOrOverride": claims,
```

```
                "claimsToSuppress": ["email"]
            },
            "accessTokenGeneration": {
                "claimsToAddOrOverride": claims,
                "claimsToSuppress": ["email"],
                "scopesToAdd": scopes,
                "scopesToSuppress": ["aws.cognito.signin.user.admin"]
            }
        }
    };
    console.info("EVENT response\n" + JSON.stringify(event, (_, v) => typeof v === 
 'bigint' ? v.toString() : v, 2))
    console.info("EVENT response size\n" + JSON.stringify(event, (_, v) => typeof v 
 === 'bigint' ? v.toString() : v).length)
    // Return to Amazon Cognito
    context.done(null, event);
};
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그런 다음 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. Lambda 콘솔에서 해당 Lambda 트리거와 관련 있는 데이터로 테스트 이벤트를 설정할 수 있습니다. 다음은 이 코드 샘플의 테스트 이벤트입니다.

JSON

```
{
    "version": "2",
    "triggerSource": "TokenGeneration_HostedAuth",
    "region": "us-west-2",
    "userPoolId": "us-west-2_EXAMPLE",
    "userName": "JaneDoe",
    "callerContext": {
        "awsSdkVersion": "aws-sdk-unknown-unknown",
        "clientId": "1example23456789"
    },
    "request": {
        "userAttributes": {
            "sub": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "cognito:user_status": "CONFIRMED",
            "email_verified": "true",
            "phone_number_verified": "true",
            "phone_number": "+12065551212",
            "email": "Jane.Doe@example.com"
```

```
        },
        "groupConfiguration": {
            "groupsToOverride": ["group-1", "group-2", "group-3"],
            "iamRolesToOverride": ["arn:aws:iam::123456789012:role/sns_caller1"],
            "preferredRole": ["arn:aws:iam::123456789012:role/sns_caller1"]
        },
        "scopes": [
            "aws.cognito.signin.user.admin",
            "phone",
            "openid",
            "profile",
            "email"
        ]
    },
    "response": {
        "claimsAndScopeOverrideDetails": []
    }
}
```

사전 토큰 생성 이벤트 버전 1 예시: 새 클레임 추가 및 기존 클레임 억제

이 예시에서는 사전 토큰 생성 Lambda 함수와 버전 1 트리거 이벤트를 사용하여 새 클레임을 추가하고 기존 클레임을 차단합니다.

Node.js

```
const handler = async (event) => {
  event.response = {
    claimsOverrideDetails: {
      claimsToAddOrOverride: {
        my_first_attribute: "first_value",
        my_second_attribute: "second_value",
      },
      claimsToSuppress: ["email"],
    },
  };

  return event;
};

export { handler };
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그런 다음 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. Lambda 콘솔에서 해당 Lambda 트리거와 관련 있는 데이터로 테스트 이벤트를 설정할 수 있습니다. 다음은 이 코드 샘플에 대한 테스트 이벤트입니다. 이 코드 예에서는 요청 파라미터를 처리하지 않기 때문에 빈 요청에 테스트 이벤트를 사용할 수 있습니다. 공통 요청 파라미터에 대한 자세한 내용은 사용자 풀 Lambda 트리거 이벤트 섹션을 참조하세요.

JSON

```
{
  "request": {},
  "response": {}
}
```

사전 토큰 생성 이벤트 버전 1 예시: 사용자의 그룹 멤버십 수정

이 예시에서는 사전 토큰 생성 Lambda 함수와 버전 1 트리거 이벤트를 사용하여 사용자의 그룹 멤버십을 수정합니다.

Node.js

```
const handler = async (event) => {
  event.response = {
    claimsOverrideDetails: {
      groupOverrideDetails: {
        groupsToOverride: ["group-A", "group-B", "group-C"],
        iamRolesToOverride: [
          "arn:aws:iam::XXXXXXXXXXX:role/sns_callerA",
          "arn:aws:iam::XXXXXXXXX:role/sns_callerB",
          "arn:aws:iam::XXXXXXXXXX:role/sns_callerC",
        ],
        preferredRole: "arn:aws:iam::XXXXXXXXXXX:role/sns_caller",
      },
    },
  };

  return event;
};

export { handler };
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그런 다음 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. Lambda 콘솔에서 해당 Lambda 트리거와 관련 있는 데이터로 테스트 이벤트를 설정할 수 있습니다. 다음은 이 코드 샘플의 테스트 이벤트입니다.

JSON

```
{
  "request": {},
  "response": {}
}
```

사용자 마이그레이션 Lambda 트리거

사용자가 암호로 로그인할 때 사용자 풀에 존재하지 않는 경우 또는 암호 찾기 흐름에 있는 경우 Amazon Cognito에서 이 트리거를 호출합니다. Lambda 함수가 성공적으로 반환된 이후 Amazon Cognito는 사용자를 사용자 풀에 추가합니다. 사용자 마이그레이션 Lambda 트리거를 사용하는 인증 흐름에 대한 자세한 내용은 사용자 마이그레이션 Lambda 트리거를 사용하여 사용자 가져오기를 참조하세요.

로그인 시 또는 암호 찾기 흐름 동안 기존 사용자 디렉터리에 있는 사용자를 Amazon Cognito 사용자 풀로 마이그레이션하려면 이 Lambda 트리거를 사용합니다.

주제

- 사용자 마이그레이션 Lambda 트리거 소스
- 사용자 마이그레이션 Lambda 트리거 파라미터
- 예제: 기존 암호가 있는 사용자 마이그레이션
- 마이그레이션 사용자 트리거 문제 해결

사용자 마이그레이션 Lambda 트리거 소스

triggerSource 값	Event
`UserMigration_Authentication` [1]	로그인 시 사용자 마이그레이션입니다.
`UserMigration_ForgotPassword`	암호 찾기 흐름 도중 사용자 마이그레이션.

[1] Amazon Cognito는 사용자가 암호 없는 로그인으로 인증할 때 이 트리거를 호출하지 않습니다.

사용자 마이그레이션 Lambda 트리거 파라미터

Amazon Cognito가 이 Lambda 함수에 전달하는 요청은 아래 파라미터와 Amazon Cognito가 모든 요청에 추가하는 공통 파라미터의 조합입니다.

JSON

```
{
    "userName": "string",
    "request": {
        "password": "string",
        "validationData": {
            "string": "string",
            . . .
        },
        "clientMetadata": {
            "string": "string",
       . . .
        }
    },
    "response": {
        "userAttributes": {
            "string": "string",
            . . .
        },
        "finalUserStatus": "string",
        "messageAction": "string",
        "desiredDeliveryMediums": [ "string", . . .],
        "forceAliasCreation": boolean,
        "enableSMSMFA": boolean
    }
}
```

사용자 마이그레이션 요청 파라미터

userName

로그인 시 사용자가 입력하는 사용자 이름입니다.

암호

로그인 시 사용자가 입력하는 암호입니다. Amazon Cognito는 암호 찾기 흐름에 의해 시작된 요청에서 이 값을 보내지 않습니다.

validationData

사용자 로그인 요청에 검증 데이터를 포함하는 하나 이상의 키-값 페어입니다. 이 데이터를 Lambda 함수에 전달하려면 InitiateAuth 및 AdminInitiateAuth API 작업에서 ClientMetadata 파라미터를 사용합니다.

clientMetadata

사용자 마이그레이션 트리거를 위한 Lambda 함수에 사용자 지정 입력으로 제공할 수 있는 하나 이상의 키-값 페어입니다. 이 데이터를 Lambda 함수에 전달하려면 AdminRespondToAuthChallenge 및 ForgotPassword API 작업에서 ClientMetadata 파라미터를 사용합니다.

사용자 마이그레이션 응답 파라미터

userAttributes

이 필드는 필수 항목입니다.

이 필드는 Amazon Cognito가 사용자 풀의 사용자 프로필에 저장하고 사용자 속성으로 사용하는 이름-값 페어를 하나 이상 포함해야 합니다. 표준 속성과 사용자 지정 속성을 모두 포함할 수 있습니다. 표준 속성과 구별하기 위해 사용자 지정 속성에 `custom:` 접두사가 필요합니다. 자세한 내용은 사용자 지정 속성을 참조하세요.

> **Note**
>
> 암호 찾기 흐름에서 사용자가 암호를 재설정하려면 확인된 이메일 주소 또는 확인된 전화번호를 입력해야 합니다. Amazon Cognito는 암호 재설정 코드가 포함된 메시지를 사용자 속성에 있는 이메일 주소 또는 전화 번호로 보냅니다.

속성	요구 사항
사용자 풀을 만들 때 필수로 표시한 모든 속성	마이그레이션 중에 필수 속성이 누락된 경우 Amazon Cognito는 기본값을 사용합니다.

속성	요구 사항
username	로그인을 위한 사용자 이름 외에 별칭 속성으로 사용자 풀을 구성하고 사용자가 유효한 별칭 값을 사용자 이름으로 입력한 경우 필요합니다. 이 별칭 값은 이메일 주소, 선호하는 사용자 이름 또는 전화 번호일 수 있습니다. 요청과 사용자 풀이 별칭 요구 사항을 충족하는 경우 함수의 응답은 수신한 username 파라미터를 별칭 속성에 할당해야 합니다. 또한 응답은 username 속성에 고유한 값을 할당해야 합니다. 사용자 풀이 수신된 username을 별칭에 매핑하는 데 필요한 조건을 충족하지 않는 경우 응답의 username 파라미터는 요청과 정확하게 일치하거나 생략되어야 합니다. **Note** username은 사용자 풀에서 고유해야 합니다.

finalUserStatus

사용자가 이전 암호로 로그인할 수 있도록 사용자를 자동 확인하기 위해 이 파라미터를 CONFIRMED로 설정할 수 있습니다. 사용자를 CONFIRMED로 설정하면 해당 사용자는 사전에 추가 작업을 수행하지 않고도 로그인할 수 있습니다. 이 속성을 CONFIRMED로 설정하지 않은 경우 해당 속성이 RESET_REQUIRED로 설정됩니다.

RESET_REQUIRED의 finalUserStatus는 사용자가 로그인 시 마이그레이션 직후에 자신의 암호를 변경해야 하며 클라이언트 앱은 인증 흐름 동안 PasswordResetRequiredException을 처리해야 함을 의미합니다.

Note

Amazon Cognito는 Lambda 트리거를 사용하여 마이그레이션하는 동안 사용자 풀에 대해 구성된 암호 강도 정책을 적용하지 않습니다. 암호가 구성된 암호 정책에 부합하지 않는 경우에도 Amazon Cognito는 여전히 암호를 수락하여 사용자를 계속 마이그레이션할 수 있습니다. 암호 강도 정책을 적용하고 정책에 부합하지 않는 암호를 거부하려면 코드에서

암호 강도를 검증합니다. 그런 다음 암호가 정책에 부합하지 않는 경우 finalUserStatus를 RESET_REQUIRED로 설정합니다.

messageAction

Amazon Cognito가 일반적으로 새 사용자에게 보내는 시작 메시지 보내기를 거부하려면 이 파라미터를 SUPPRESS로 설정합니다. 함수가 이 파라미터를 반환하지 않으면 Amazon Cognito가 시작 메시지를 보냅니다.

desiredDeliveryMediums

시작 메시지를 이메일로 보내려면 이 파라미터를 EMAIL로 설정하고, 시작 메시지를 SMS로 보내려면 이 파라미터를 SMS로 설정합니다. 함수가 이 파라미터를 반환하지 않으면 Amazon Cognito가 시작 메시지를 SMS로 보냅니다.

forceAliasCreation

이 파라미터가 TRUE로 설정되고 UserAttributes 파라미터의 전화 번호 또는 이메일 주소가 다른 사용자의 별칭으로 이미 존재하는 경우 API 호출이 이전 사용자의 별칭을 새로 생성된 사용자로 마이그레이션합니다. 이전 사용자는 더 이상 해당 별칭을 사용하여 로그인할 수 없습니다.

이 파라미터를 FALSE로 설정하고 별칭이 존재하는 경우 Amazon Cognito가 사용자를 마이그레이션하지 않고 클라이언트 앱에 오류를 반환합니다.

이 파라미터를 반환하지 않는 경우 Amazon Cognito는 해당 값이 'false'라고 가정합니다.

enableSMSMFA

이 파라미터를 true로 설정하는 경우 마이그레이션된 사용자가 로그인하려면 SMS 문자 메시지 다중 인증(MFA)을 완료해야 합니다. 사용자 풀에 MFA가 활성화되어 있어야 합니다. 요청 파라미터의 사용자 속성에 전화 번호가 포함되어야 합니다. 그렇지 않으면 해당 사용자의 마이그레이션이 실패합니다.

예제: 기존 암호가 있는 사용자 마이그레이션

이 Lambda 함수 예는 기존 암호를 가진 사용자를 마이그레이션하고 Amazon Cognito의 환영 인사 메시지를 억제합니다.

Node.js

```
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);
  }
};
```

마이그레이션 사용자 트리거 문제 해결

함수가 문서화된 응답을 반환하더라도 마이그레이션이 예상대로 작동하지 않는 경우 다음과 같은 일반적인 원인을 확인합니다.

트리거가 호출되지 않음

Amazon Cognito는 사용자가 사용자 풀에 아직 없는 경우에만 마이그레이션 사용자 트리거를 호출합니다. USER_PASSWORD_AUTH, ADMIN_USER_PASSWORD_AUTH및 USER_AUTH (암호 사용) 로그인 흐름과 암호 찾기 흐름에만 적용됩니다. 사용자의 암호가 해당 흐름의 함수에 도달하지 않거나 암호 없는 로그인의 경우 SRP(USER_SRP_AUTH) 로그인에 대해 호출되지 않습니다. 트리거가 실행되지 않는 경우 앱 클라이언트가 이러한 흐름 중 하나를 허용하고 사용자가 풀에 아직 없는지 확인합니다.

Amazon Cognito가 함수를 호출할 수 없음

Amazon Cognito 콘솔을 통하지 않고 함수에 리소스 기반 정책을 사용하여 트리거를 설정하는 경우 정책은 `cognito-idp.amazonaws.com` 서비스 보안 주체가 사용자 풀 ARN으로 `lambda:InvokeFunction`범위가 지정된를 호출하도록 허용해야 합니다. 콘솔은이 권한을 추가합니다. 코드형 인프라 또는 Lambda API와 연결된 함수는 명시적으로 추가해야 합니다. 그렇지 않으면 Amazon Cognito가 트리거를 호출할 수 없기 때문에 마이그레이션이 실패합니다.

사용자가 로그인했지만 프로필이 생성되지 않음

Amazon Cognito는 함수가 예상한 정확한 모양으로 응답을 반환하는 경우에만 마이그레이션된 사용자를 생성합니다. `userAttributes` 필드는 있어야 하며 사용자 풀에 필요한 것으로 표시된 모든 속성을 포함해야 합니다. 사용자를 인증하지만 비어 있거나 잘못된 형식의 `response` 객체를 반환하거나 필요한 속성을 생략하는 함수는 프로파일을 유지하지 않고 로그인 호출을 완료합니다. 모든 필수 속성`userAttributes`과 함께를 반환하고 필요에 `messageAction` 따라 `finalUserStatus` 및를 설정합니다.

사용자 정의 메시지 Lambda 트리거

사용자에게 전송하려는 이메일 및 SMS 메시지에 대한 외부 표준이 있거나 런타임에 사용자 메시지 형식에 자체 로직을 적용하려는 경우 사용자 풀에 사용자 지정 메시지 트리거를 추가합니다. 사용자 지정 메시지 Lambda는 사용자 풀에서 전송하기 전에 모든 이메일 및 SMS 메시지의 내용을 수신합니다. 그러면 Lambda 함수에서 메시지 내용 및 주제를 수정할 수 있습니다.

Amazon Cognito는 이메일이나 전화 확인 메시지 또는 멀티 팩터 인증(MFA) 코드를 보내기 전에 이 트리거를 호출합니다. 사용자 지정 메시지 트리거를 사용하여 메시지를 동적으로 사용자 지정할 수 있습니다.

요청에는 `codeParameter`가 포함됩니다. 이는 Amazon Cognito가 사용자에게 전달하는 코드의 자리 표시자 역할을 하는 문자열입니다. `codeParameter` 문자열을 확인 코드가 나타날 메시지 본문에 삽

입합니다. Amazon Cognito에서 응답을 수신하면 Amazon Cognito에서 `codeParameter` 문자열을 실제 확인 코드로 바꿉니다.

> **Note**
>
> `CustomMessage_AdminCreateUser` 트리거 소스가 있는 사용자 지정 메시지 Lambda 함수의 입력 이벤트에는 사용자 이름과 확인 코드가 포함됩니다. 관리자가 생성한 사용자는 사용자 이름과 코드를 모두 수신해야 하므로 함수의 응답에는 사용자 이름과 코드에 대한 자리 표시자 변수가 포함되어야 합니다. 메시지의 자리 표시자는 `request.usernameParameter` 및 `request.codeParameter`의 값입니다. 이러한 값은 일반적으로 `{username}` 및 `{####}`입니다. 가장 좋은 방법은 변수 이름을 하드 코딩하는 대신 입력 값을 참조하는 것입니다.

> **Note**
>
> Amazon Cognito 콘솔의 메시지 템플릿 메뉴에서 구성하는 메시지는 사용자 지정 메시지 함수가 설정하는 본문을 재정의할 수 있습니다. 함수의 메시지가 표시되지 않으면 콘솔 메시지 템플릿이 동일한 이벤트에 대해 구성되어 있는지 확인합니다. 자세한 내용은 MFA, 인증, 확인 및 초대 메시지 구성 섹션을 참조하세요.

주제

- 사용자 정의 메시지 Lambda 트리거 소스
- 사용자 정의 메시지 Lambda 트리거 파라미터
- 가입용 사용자 지정 메시지의 예
- 관리자로 사용자 생성용 사용자 정의 메시지 예제

사용자 정의 메시지 Lambda 트리거 소스

triggerSource 값	Event
`CustomMessage_SignUp`	사용자 지정 메시지 – 가입 후 확인 코드 전송.
`CustomMessage_AdminCreateUser`	사용자 지정 메시지 – 새 사용자에게 임시 암호 전송.

triggerSource 값	Event
CustomMessage_ResendCode	사용자 지정 메시지 – 기존 사용자에게 확인 코드 재전송.
CustomMessage_ForgotPassword	사용자 지정 메시지 – 암호 분실 요청을 위한 확인 코드 전송.
CustomMessage_UpdateUserAttribute	사용자 지정 메시지 – 사용자의 이메일 또는 전화 번호가 변경되면 이 트리거가 사용자에게 자동으로 확인 코드를 전송합니다. 다른 속성에는 사용할 수 없습니다.
CustomMessage_VerifyUserAttribute	사용자 지정 메시지 – 사용자가 새 이메일 또는 전화 번호에 대해 수동으로 요청하면 이 트리거가 사용자에게 확인 코드를 전송합니다.
CustomMessage_Authentication	사용자 지정 메시지 – 인증하는 동안 MFA 코드 전송.

사용자 정의 메시지 Lambda 트리거 파라미터

Amazon Cognito가 이 Lambda 함수에 전달하는 요청은 아래 파라미터와 Amazon Cognito가 모든 요청에 추가하는 공통 파라미터의 조합입니다.

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
        }
        "codeParameter": "####",
        "usernameParameter": "string",
        "clientMetadata": {
            "string": "string",
            . . .
        }
    },
```

```
    "response": {
        "smsMessage": "string",
        "emailMessage": "string",
        "emailSubject": "string"
    }
}
```

사용자 정의 메시지 요청 파라미터

userAttributes

사용자 속성을 나타내는 하나 이상의 이름-값 페어입니다.

codeParameter

사용자 지정 메시지에서 확인 코드의 자리 표시자로 사용할 문자열입니다.

usernameParameter

사용자 이름입니다. Amazon Cognito는 관리자가 생성한 사용자의 요청에 이 파라미터를 포함합니다.

clientMetadata

사용자 지정 메시지 트리거에 지정하는 Lambda 함수에 사용자 지정 입력으로 제공할 수 있는 하나 이상의 키-값 쌍입니다. 사용자 지정 메시지 기능을 호출하는 요청에는 AdminInitiateAuth 및 InitiateAuth API 작업의 ClientMetadata 파라미터에 전달된 데이터가 포함되지 않습니다. 이 데이터를 Lambda 함수에 전달하려면 다음 API 작업에 ClientMetadata 파라미터를 사용합니다.

- AdminResetUserPassword
- AdminRespondToAuthChallenge
- AdminUpdateUserAttributes
- ForgotPassword
- GetUserAttributeVerificationCode
- ResendConfirmationCode
- SignUp
- UpdateUserAttributes

사용자 정의 메시지 응답 파라미터

응답에서는 사용자에게 보낼 메시지에 사용할 사용자 지정 텍스트를 지정합니다. Amazon Cognito가 이러한 파라미터에 적용하는 문자열 제약 조건은 MessageTemplateType을 참조하세요.

smsMessage

사용자에게 보낼 사용자 지정 SMS 메시지입니다. 요청에서 수신한 `codeParameter` 값을 포함해야 합니다.

emailMessage

사용자에게 보낼 사용자 지정 이메일 메시지입니다. `emailMessage` 파라미터에서 HTML 서식을 사용할 수 있습니다. 요청에서 수신한 `codeParameter` 값을 변수 `{####}`으로 포함해야 합니다. Amazon Cognito는 사용자 풀의 `EmailSendingAccount` 속성이 `DEVELOPER`인 경우에만 `emailMessage` 파라미터를 사용할 수 있습니다. 사용자 풀의 `EmailSendingAccount` 속성이 `DEVELOPER`가 아니고 `emailMessage` 파라미터가 반환된 경우 Amazon Cognito에서 400오류 코드 `com.amazonaws.cognito.identity.idp.model.InvalidLambdaResponseException`을 생성합니다. Amazon Simple Email Service(Amazon SES)를 선택하여 이메일 메시지를 보내면 사용자 풀의 `EmailSendingAccount` 속성이 `DEVELOPER`입니다. 그렇지 않으면 이 값은 `COGNITO_DEFAULT`입니다.

emailSubject

사용자 지정 메시지의 제목 줄입니다. 사용자 풀의 EmailSendingAccount 속성이 `DEVELOPER`인 경우에만 `emailSubject` 파라라미터를 사용할 수 있습니다. 사용자 풀의 `EmailSendingAccount` 속성이 `DEVELOPER`가 아니고 Amazon Cognito에서 `emailSubject` 파라미터를 반환하는 경우 Amazon Cognito에서 400 오류 코드 `com.amazonaws.cognito.identity.idp.model.InvalidLambdaResponseException`을 생성합니다. Amazon Simple Email Service(Amazon SES)를 선택하여 이메일 메시지를 보내도록 선택하면 사용자 풀의 `EmailSendingAccount` 속성이 `DEVELOPER`입니다. 그렇지 않으면 이 값은 `COGNITO_DEFAULT`입니다.

가입용 사용자 지정 메시지의 예

이 Lambda 함수 예는 서비스에서 앱이 사용자에게 확인 코드를 보내야 할 때 이메일 또는 SMS 메시지를 사용자 지정합니다.

Amazon Cognito는 사후 등록, 확인 코드 재전송, 잊어버린 암호 복구 또는 사용자 속성 확인과 같은 여러 이벤트에서 Lambda 트리거를 호출할 수 있습니다. SMS와 이메일 모두의 메시지가 응답에 포함됩

니다. 메시지에는 코드 파라미터 "####"이 포함되어야 합니다. 이 파라미터는 사용자가 수신하는 확인 코드의 자리 표시자입니다.

이메일 메시지의 최대 길이는 20,000 UTF-8자입니다. 이 길이에는 확인 코드가 포함됩니다. 이러한 이메일 메시지에 HTML 태그를 사용할 수 있습니다.

SMS 메시지의 최대 길이는 UTF-8 140자입니다. 이 길이에는 확인 코드가 포함됩니다.

> **Note**
>
> 함수가 20,000 UTF-8 문자보다 긴 이메일 메시지를 반환하는 경우 Amazon Cognito는 오류를 반환하지 않습니다. 대신 사용자 지정 메시지를 자동으로 삭제하고 해당 이벤트에 대한 기본 메시지를 전송합니다. 사용자 지정 메시지가 사용자에게 도달하지 않는 경우 먼저이 제한과 비교하여 길이를 확인합니다.

Node.js

```
const handler = async (event) => {
  if (event.triggerSource === "CustomMessage_SignUp") {
    const message = `Thank you for signing up. Your confirmation code is
 ${event.request.codeParameter}.`;
    event.response.smsMessage = message;
    event.response.emailMessage = message;
    event.response.emailSubject = "Welcome to the service.";
  }
  return event;
};

export { handler };
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그런 다음 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. Lambda 콘솔에서 해당 Lambda 트리거와 관련 있는 데이터로 테스트 이벤트를 설정할 수 있습니다. 다음은 이 코드 샘플의 테스트 이벤트입니다.

JSON

```
{
 "version": "1",
 "region": "us-west-2",
```

```
 "userPoolId": "us-west-2_EXAMPLE",
 "userName": "test-user",
 "callerContext": {
  "awsSdkVersion": "aws-sdk-unknown-unknown",
  "clientId": "1example23456789"
 },
 "triggerSource": "CustomMessage_SignUp",
 "request": {
  "userAttributes": {
   "sub": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
   "cognito:user_status": "CONFIRMED",
   "email_verified": "true",
   "phone_number_verified": "true",
   "phone_number": "+12065551212",
   "email": "test-user@example.com"
  },
  "codeParameter": "{####}",
  "linkParameter": "{##Click Here##}",
  "usernameParameter": "None"
 },
 "response": {
  "smsMessage": "None",
  "emailMessage": "None",
  "emailSubject": "None"
 }
}
```

관리자로 사용자 생성용 사용자 정의 메시지 예제

이 예제 사용자 지정 메시지 Lambda 함수에 Amazon Cognito가 전송한 요청에는 `CustomMessage_AdminCreateUser`의 `triggerSource` 값과 사용자 이름 및 임시 비밀번호가 있습니다. 이 함수는 요청의 임시 비밀번호에서 `${event.request.codeParameter}`를 채우고 요청의 사용자 이름에서 `${event.request.usernameParameter}`를 채웁니다.

사용자 지정 메시지는 `codeParameter` 및 `usernameParameter`의 값을 응답 개체의 `smsMessage` 및 `emailMessage`에 삽입해야 합니다. 이 예제에서는 함수가 응답 필드 `event.response.smsMessage` 및 `event.response.emailMessage`에 동일한 메시지를 기록합니다.

이메일 메시지의 최대 길이는 UTF-8 20,000자입니다. 이 길이에는 확인 코드가 포함됩니다. 이러한 이메일에 HTML 태그를 사용할 수 있습니다. SMS 메시지의 최대 길이는 UTF-8 140자입니다. 이 길이에는 확인 코드가 포함됩니다.

> **Note**
>
> 함수가 20,000 UTF-8 문자보다 긴 이메일 메시지를 반환하는 경우 Amazon Cognito는 오류를 반환하지 않습니다. 대신 사용자 지정 메시지를 자동으로 삭제하고 해당 이벤트에 대한 기본 메시지를 전송합니다. 사용자 지정 메시지가 사용자에게 도달하지 않는 경우 먼저이 제한과 비교하여 길이를 확인합니다.

SMS와 이메일 모두의 메시지가 응답에 포함됩니다.

Node.js

```
const handler = async (event) => {
  if (event.triggerSource === "CustomMessage_AdminCreateUser") {
    const message = `Welcome to the service. Your user name is
 ${event.request.usernameParameter}. Your temporary password is
 ${event.request.codeParameter}`;
    event.response.smsMessage = message;
    event.response.emailMessage = message;
    event.response.emailSubject = "Welcome to the service";
  }
  return event;
};

export { handler };
```

Amazon Cognito는 이벤트 정보를 Lambda 함수에 전달합니다. 그런 다음 함수는 응답이 변경되면 동일한 이벤트 객체를 Amazon Cognito에 반환합니다. Lambda 콘솔에서 해당 Lambda 트리거와 관련 있는 데이터로 테스트 이벤트를 설정할 수 있습니다. 다음은 이 코드 샘플의 테스트 이벤트입니다.

JSON

```
{
  "version": 1,
  "triggerSource": "CustomMessage_AdminCreateUser",
  "region": "<region>",
```

```
  "userPoolId": "<userPoolId>",
  "userName": "<userName>",
  "callerContext": {
      "awsSdk": "<calling aws sdk with version>",
      "clientId": "<apps client id>",
      ...
  },
  "request": {
      "userAttributes": {
          "phone_number_verified": false,
          "email_verified": true,
           ...
      },
      "codeParameter": "####",
      "usernameParameter": "username"
  },
  "response": {
      "smsMessage": "<custom message to be sent in the message with code parameter
 and username parameter>"
      "emailMessage": "<custom message to be sent in the message with code parameter
 and username parameter>"
      "emailSubject": "<custom email subject>"
  }
}
```

사용자 지정 발신자 Lambda 트리거

Lambda는 사용자 풀에서 `CustomEmailSender` 및 `CustomSMSSender` 지원 타사 이메일 및 SMS 알림을 트리거합니다. SMS 및 이메일 공급자를 선택하여 Lambda 함수 코드 내에서 사용자에게 알림을 보낼 수 있습니다. Amazon Cognito가 사용자에게 초대, MFA 코드, 확인 코드, 인증 코드 및 임시 비밀번호를 전송하면 이벤트는 구성된 Lambda 함수를 활성화합니다. Amazon Cognito는 코드 및 임시 보안 암호를 활성화된 Lambda에 보냅니다. Amazon Cognito는 AWS KMS 고객 관리형 키와를 사용하여 이러한 보안 암호를 암호화합니다 AWS Encryption SDK. 는 일반 데이터를 암호화하고 해독하는 데 도움이 되는 클라이언트 측 암호화 라이브러리 AWS Encryption SDK 입니다.

CustomEmailSender

Amazon Cognito는 이 트리거를 호출하여 사용자에게 이메일 알림을 전송합니다.

CustomSMSSender

Amazon Cognito는 이 트리거를 호출하여 사용자에게 SMS 알림을 전송합니다.

암호화 개념

Amazon Cognito는 사용자 지정 발신자 트리거에 보내는 이벤트에서 일반 텍스트로 사용자 코드를 보내지 않습니다. Lambda 함수는 이벤트에서 코드를 해독해야 합니다. 다음 개념은 함수가 사용자에게 전달할 수 있는 코드를 가져오는 데 사용해야 하는 암호화 아키텍처입니다.

AWS KMS

AWS KMS 는 AWS KMS 키를 생성하고 제어하는 관리형 서비스입니다. 이러한 키는 데이터를 암호화합니다. 자세한 내용은 AWS Key Management Service란 무엇입니까?를 참조하세요.

KMS 키

KMS 키는 암호화 키를 논리적으로 표현한 것입니다. KMS 키에는 키 ID, 생성 날짜, 설명 및 키 상태와 같은 메타데이터가 포함됩니다. 또한 KMS 키에는 데이터를 암호화 및 복호화하는 데 사용되는 키 재료도 포함됩니다. 자세한 내용은 AWS KMS 키를 참조하세요.

대칭 KMS 키

대칭 KMS 키는 암호화되지 않은 상태로 AWS KMS 를 종료하지 않는 256비트 암호화 키입니다. 대칭 KMS 키를 사용하려면를 호출해야 합니다 AWS KMS. Amazon Cognito는 대칭 키를 사용합니다. 동일한 키가 암호화하고 해독합니다. 자세한 내용은 대칭 KMS 키를 참조하세요.

사용자 지정 발신자 Lambda 트리거에 대해 알아야 할 사항

- 사용자 풀에서 이러한 Lambda 트리거를 사용하도록 구성하려면 AWS CLI 또는 SDK를 사용합니다. Amazon Cognito 콘솔에서는 이러한 구성을 사용할 수 없습니다.

  `UpdateUserPool` 작업은 Lambda 구성을 설정합니다. 이 작업에 대한 요청에는 사용자 풀의 모든 파라미터 및 변경하려는 파라미터가 필요합니다. 관련 파라미터를 모두 제공하지 않으면 Amazon Cognito에서 누락된 파라미터 값을 기본값으로 설정합니다. 다음 AWS CLI 예제에 설명된 대로 사용자 풀에 추가하거나 유지하려는 모든 Lambda 함수에 대한 항목을 포함합니다. 자세한 내용은 사용자 풀 및 앱 클라이언트 구성 업데이트 단원을 참조하십시오.

```
   #Send this parameter in an 'aws cognito-idp update-user-pool' CLI command,
including any existing
   #user pool configurations. This snippet also includes a pre sign-up trigger for
syntax reference. The pre sign-up trigger
   #doesn't have a role in custom sender triggers.
```

```
    --lambda-config "PreSignUp=lambda-arn, \
                     CustomSMSSender={LambdaVersion=V1_0,LambdaArn=lambda-arn}, \
                     CustomEmailSender={LambdaVersion=V1_0,LambdaArn=lambda-arn}, \
                     KMSKeyID=key-id"
```

UpdateUserPool의 JSON 본문을 사용하는 요청의 경우 다음 LambdaConfig 스니펫은 사용자 지정 SMS 및 이메일 발신자 함수를 할당합니다.

```
"LambdaConfig": {
   "KMSKeyID": "arn:aws:kms:us-
east-1:111122223333:key/a6c4f8e2-0c45-47db-925f-87854bc9e357",
   "CustomEmailSender": {
      "LambdaArn": "arn:aws:lambda:us-east-1:111122223333:function:MyFunction",
      "LambdaVersion": "V1_0"
   },
   "CustomSMSSender": {
      "LambdaArn": "arn:aws:lambda:us-east-1:111122223333:function:MyFunction",
      "LambdaVersion": "V1_0"
   }
```

- update-user-pool AWS CLI 명령으로 사용자 지정 발신자 Lambda 트리거를 제거하려면에서 CustomSMSSender 또는 CustomEmailSender 파라미터를 생략--lambda-config하고 사용자 풀에 사용할 다른 모든 트리거를 포함합니다.

  UpdateUserPool API 요청이 포함된 사용자 지정 발신자 Lambda 트리거를 제거하려면 나머지 사용자 풀 구성을 포함하는 요청 본문에서 CustomSMSSender 또는 CustomEmailSender 파라미터를 생략합니다.

- Amazon Cognito HTML은 사용자의 임시 암호에 있는 <(<) 및>(>) 같은 예약 문자를 이스케이프 처리합니다. 이러한 문자는 Amazon Cognito가 사용자 지정 이메일 발신자 함수로 보내는 임시 암호에 표시될 수 있지만, 임시 확인 코드에는 표시되지 않습니다. 임시 암호를 전송하려면 Lambda 함수가 암호를 해독하고 이러한 문자를 이스케이프 해제한 다음 사용자에게 메시지를 전송해야 합니다.

사용자 지정 발신자 Lambda 트리거 활성화

사용자 풀에 SMS 또는 이메일 메시지를 보내기 위해 사용자 지정 로직을 사용하려면 사용자 지정 발신자 트리거를 설정합니다. 다음 절차에서는 사용자 풀에 사용자 지정 SMS 트리거, 사용자 지정 이메일 트리거 또는 둘 다를 할당합니다. 사용자 지정 발신자 트리거를 추가하면 Amazon Cognito는 SMS

또는 이메일 메시지를 보내는 기본 동작 대신 전화번호, 일회용 코드 등의 사용자 속성을 항상 Lambda 함수로 전송합니다.

1. AWS Key Management Service ()에서 대칭 암호화 키를 생성합니다AWS KMS. Amazon Cognito는 보안 암호(임시 암호, 확인 코드, 인증용 일회용 암호, 확인 코드)을 생성한 다음 이 KMS 키를 사용하여 AWS Encryption SDK로 보안 암호를 암호화합니다. 그런 다음 Lambda 함수 AWS Encryption SDK 의를 사용하여 보안 암호를 해독하고 일반 텍스트로 사용자에게 보낼 수 있습니다.
2. 사용자 풀을 생성하거나 업데이트하는 IAM 보안 주체는 Amazon Cognito가 코드를 암호화하는 데 사용하는 KMS 키에 대해 일회용 권한 부여를 생성합니다. KMS 키에 대한이 보안 주체 `CreateGrant` 권한을 부여합니다. 이 예제 KMS 키 정책이 적용되려면 사용자 풀을 업데이트하는 관리자가 IAM 역할 `arn:aws:iam::111222333444:role/my-example-administrator-role`에 대한 위임된 역할 세션으로 로그인해야 합니다.

   사용자 환경에 맞게 수정된 다음 리소스 기반 정책을 KMS 키에 적용합니다.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::111122223333:role/my-example-administrator-
role"
        },
        "Action": "kms:CreateGrant",
        "Resource": "arn:aws:kms:us-
west-2:111122223333:key/1example-2222-3333-4444-999example",
        "Condition": {
            "StringEquals": {
               "kms:EncryptionContext:userpool-id": "us-west-2_EXAMPLE"
            }
        }
    },
    {
        "Sid": "Allow Lambda to decrypt",
        "Effect": "Allow",
        "Principal": {
```

```
            "AWS": "arn:aws:iam::111122223333:role/my-lambda-function-role"
        },
        "Action": "kms:Decrypt",
        "Resource": "*"
    }]
}
```

3. 사용자 지정 SMS 발신자 트리거에 대한 Lambda 함수를 생성합니다. Amazon Cognito는 AWS 암호화 SDK를 사용하여 보안 암호, 임시 암호 및 사용자의 API 요청을 인증하는 코드를 암호화합니다.

   a. 최소한 KMS 키에 대한 `kms:Decrypt` 권한이 있는 Lambda 실행 역할을 할당합니다.

   b. Lambda 함수 코드를 작성하여 메시지를 전송합니다. 함수에 대한 입력 이벤트에는 보안 암호가 포함되어 있습니다. 함수에서를 사용하여 보안 암호를 해독 AWS Encryption SDK 하고 관련 메타데이터를 처리합니다. 그런 다음 코드, 사용자 지정 메시지 및 대상 전화번호를 메시지를 전달하는 사용자 지정 API에 전송합니다.

   c. Lambda 함수 AWS Encryption SDK 에를 추가합니다. 자세한 내용은 AWS 암호화 SDK 프로그래밍 언어를 참조하세요. Lambda 패키지를 업데이트하려면 다음 단계를 완료하세요.

      i. Lambda 함수를 AWS Management Console에 .zip 파일로 내보냅니다.

      ii. 함수를 열고를 추가합니다 AWS Encryption SDK. 자세한 내용과 다운로드 링크는 AWS Encryption SDK 개발자 안내서의 AWS Encryption SDK 프로그래밍 언어를 참조하세요.

      iii. SDK 종속성과 함께 함수를 압축하고 함수를 Lambda에 업로드합니다. 자세한 내용은 AWS Lambda 개발자 안내서의 Lambda 함수를 .zip 파일 아카이브로 배포를 참조하세요.

4. Amazon Cognito 서비스 보안 주체 `cognito-idp.amazonaws.com`에 Lambda 함수를 호출할 수 있는 권한을 부여합니다.

   다음 AWS CLI 명령은 Amazon Cognito에 Lambda 함수를 호출할 수 있는 권한을 부여합니다.

```
aws lambda add-permission --function-name lambda_arn --statement-id
 "CognitoLambdaInvokeAccess" --action lambda:InvokeFunction --principal cognito-
idp.amazonaws.com
```

5. 사용자 지정 발신자 Lambda 트리거를 추가하는 `LambdaConfig` 파라미터를 사용하여 UpdateUserPool API 요청을 생성합니다. Amazon Cognito 콘솔에서는 이 유형의 트리거를 추가할 수 없습니다. 사용자 지정 발신자 트리거에는 `KMSKeyID` 및 `CustomSMSSender` 또는 `CustomEmailSender`(또는 둘 다)의 `LambdaConfig` 파라미터가 필요합니다.

사용자 정의 이메일 발신자 Lambda 트리거

사용자 지정 이메일 발신자 트리거를 사용자 풀에 할당하면 Amazon Cognito가 사용자 이벤트로 인해 이메일 메시지를 전송해야 하는 경우 기본 동작 대신 Lambda 함수를 호출합니다. 사용자 지정 발신자 트리거를 사용하면 AWS Lambda 함수가 선택한 메서드와 공급자를 통해 사용자에게 이메일 알림을 보낼 수 있습니다. 함수의 사용자 지정 코드는 사용자 풀의 모든 이메일 메시지를 처리하고 전달해야 합니다.

이 트리거는 사용자 풀이 이메일 메시지를 보내는 방법을 더 잘 제어하려는 시나리오를 제공합니다. 예를 들어 여러 개의 인증 ID를 관리하거나 AWS 리전간에 사용하려는 경우와 같이 Lambda 함수를 사용하여 Amazon SES API 작업에 대한 호출을 사용자 지정할 수 있습니다. 함수는 메시지를 다른 전송 미디어 또는 타사 서비스로 리디렉션할 수도 있습니다.

사용자 지정 이메일 발신자 트리거를 구성하는 방법은 사용자 지정 발신자 Lambda 트리거 활성화 섹션을 참조하세요.

사용자 정의 이메일 발신자 Lambda 트리거 소스

다음 표에는 Lambda 코드의 사용자 지정 이메일 트리거 소스에 대한 트리거 이벤트가 나와 있습니다.

`TriggerSource value`	Event
`CustomEmailSender_SignUp`	사용자가 가입하면 Amazon Cognito에서 시작 메시지를 보냅니다.
`CustomEmailSender_Authentication`	사용자가 로그인하면 Amazon Cognito가 이메일 OTP 또는 MFA 코드를 보냅니다.
`CustomEmailSender_ForgotPassword`	사용자가 암호를 재설정하는 코드를 요청합니다.
`CustomEmailSender_ResendCode`	사용자가 대체 계정 확인 코드를 요청합니다.
`CustomEmailSender_UpdateUserAttribute`	사용자가 이메일 주소 또는 전화 번호 속성을 업데이트하면 Amazon Cognito에서 속성을 확인하는 코드를 보냅니다.
`CustomEmailSender_VerifyUserAttribute`	사용자가 새 이메일 주소 또는 전화 번호 속성을 생성하면 Amazon Cognito에서 속성을 확인하는 코드를 보냅니다.

TriggerSource value	Event
CustomEmailSender_AdminCreateUser	사용자 풀에 새 사용자를 생성하면 Amazon Cognito에서 해당 사용자에게 임시 암호를 보냅니다.
CustomEmailSender_AccountTakeOverNotification	Amazon Cognito는 사용자 계정을 획득하려는 시도를 감지하고 사용자에게 알림을 보냅니다.

사용자 지정 이메일 발신자 Lambda 트리거 파라미터

Amazon Cognito가 이 Lambda 함수에 전달하는 요청은 아래 파라미터와 Amazon Cognito가 모든 요청에 추가하는 공통 파라미터의 조합입니다.

JSON

```
{
    "request": {
        "type": "customEmailSenderRequestV1",
        "code": "string",
        "clientMetadata": {
            "string": "string",
             . . .
            },
        "userAttributes": {
            "string": "string",
            . . .
         }
}
```

사용자 지정 이메일 발신자 요청 파라미터

type

요청 버전입니다. 사용자 지정 이메일 발신자 이벤트의 경우 이 문자열의 값은 항상 customEmailSenderRequestV1입니다.

code

함수가 해독하여 사용자에게 보낼 수 있는 암호화된 코드입니다.

clientMetadata

사용자 지정 이메일 발신자 Lambda 함수 트리거에 대한 사용자 지정 입력으로 제공할 수 있는 하나 이상의 키-값 페어입니다. 이 데이터를 Lambda 함수에 전달하려면 AdminRespondToAuthChallenge 및 RespondToAuthChallenge API 작업에서 ClientMetadata 파라미터를 사용합니다. Amazon Cognito는 사후 승인 함수에 전달하는 요청에 있는 AdminInitiateAuth 및 InitiateAuth API 작업의 ClientMetadata 파라미터에서 전달된 데이터를 포함하지 않습니다.

> **Note**
>
> Amazon Cognito는 다음과 같은 트리거 소스가 있는 이벤트에서 사용자 지정 이메일 트리거 함수로 `ClientMetadata`를 전송합니다.
>
> - `CustomEmailSender_ForgotPassword`
> - `CustomEmailSender_SignUp`
> - `CustomEmailSender_Authentication`
>
> Amazon Cognito는 소스 `CustomEmailSender_AccountTakeOverNotification`를 사용하여 트리거 이벤트에서 `ClientMetadata`를 전송하지 않습니다.

userAttributes

사용자 속성을 나타내는 하나 이상의 키-값 페어입니다.

사용자 지정 이메일 발신자 응답 파라미터

Amazon Cognito는 사용자 지정 이메일 발신자 응답에서 추가 반환 정보를 기대하지 않습니다. Lambda 함수는 이벤트를 해석하고 코드를 복호화한 다음 메시지 내용을 전달해야 합니다. 일반적인 함수는 이메일 메시지를 조합하여 타사 SMTP 릴레이로 전달합니다.

코드 예제

다음 Node.js 예제에서는 사용자 지정 이메일 발신자 Lambda 함수에서 이메일 메시지 이벤트를 처리합니다. 이 예에서는 함수에 두 개의 환경 변수가 정의되어 있다고 가정합니다.

`KEY_ID`

사용자 코드를 암호화하고 복호화하는 데 사용하려는 KMS 키의 ID입니다.

KEY_ARN

사용자 코드를 암호화하고 복호화하는 데 사용하려는 KMS 키의 Amazon 리소스 이름(ARN)입니다.

이 함수를 배포하려면

1. 개발자 워크스페이스에 최신 버전의 NodeJS를 설치합니다.
2. 워크스페이스에서 새 NodeJS 프로젝트를 생성합니다.
3. `npm init -y`를 사용하여 프로젝트를 초기화합니다.
4. Lambda 함수 `touch index.mjs`의 스크립트를 생성합니다.
5. 아래 예제의 내용을 `index.mjs`에 붙여 넣습니다.
6. 프로젝트 종속성 다운로드 AWS Encryption SDK: `npm install @aws-crypto/client-node`.
7. 프로젝트 디렉토리를 `zip -r my_deployment_package.zip .`이라는 파일로 압축합니다.
8. ZIP 파일을 함수에 배포합니다.

이 예제 함수는 코드를 복호화하고, 가입 이벤트의 경우 사용자의 이메일 주소로 이메일 메시지를 보내는 것을 시뮬레이션합니다.

```
import { KmsKeyringNode, buildClient, CommitmentPolicy } from '@aws-crypto/client-
node';

// Configure the encryption SDK client with the KMS key from the environment variables
const { encrypt, decrypt } = buildClient(
    CommitmentPolicy.REQUIRE_ENCRYPT_ALLOW_DECRYPT
);

const generatorKeyId = process.env.KEY_ID;
const keyIds = [process.env.KEY_ARN];
const keyring = new KmsKeyringNode({ generatorKeyId, keyIds });

// Example function to simulate sending email.
// This example logs message details to CloudWatch Logs from your Lambda function.
// Update this function with custom logic that sends an email message to 'emailaddress'
 with body 'message'.
const sendEmail = async (emailAddress, message) => {
    // Log the destination with the email address masked.
```

```
    console.log(`Simulating email send to ${emailAddress.replace(/[^@.]/g, '*')}`);
    // Log the message with the code masked.
    console.log(`Message content: ${message.replace(/\b\d{6,8}\b/g, '********')}`);
    // Simulate API delay
    await new Promise(resolve => setTimeout(resolve, 100));
    console.log('Email sent successfully');
    return true;
};

export const handler = async (event) => {
    try {
        // Decrypt the secret code using encryption SDK
        let plainTextCode;
        if (event.request.code) {
            const { plaintext, messageHeader } = await decrypt(keyring,
 Buffer.from(event.request.code, 'base64'));
            plainTextCode = Buffer.from(plaintext).toString('utf-8');
        }

        // Handle different trigger sources
        if (event.triggerSource == 'CustomEmailSender_SignUp') {
            const emailAddress = event.request.userAttributes.email;
            const message = `Welcome! Your verification code is: ${plainTextCode}`;
            await sendEmail(emailAddress, message);
        }
        else if (event.triggerSource == 'CustomEmailSender_ResendCode') {
            // Handle resend code
        }
        else if (event.triggerSource == 'CustomEmailSender_ForgotPassword') {
            // Handle forgot password
        }
        else if (event.triggerSource == 'CustomEmailSender_UpdateUserAttribute') {
            // Handle update attribute
        }
        else if (event.triggerSource == 'CustomEmailSender_VerifyUserAttribute') {
            // Handle verify attribute
        }
        else if (event.triggerSource == 'CustomEmailSender_AdminCreateUser') {
            // Handle admin create user
        }
        else if (event.triggerSource == 'CustomEmailSender_Authentication') {
            // Handle authentication
        }
```

```
        else if (event.triggerSource ==
 'CustomEmailSender_AccountTakeOverNotification') {
            // Handle account takeover notification
        }

        return;
    } catch (error) {
        console.error('Error in custom email sender:', error);
        throw error;
    }
};
```

사용자 지정 SMS 발신자 Lambda 트리거

사용자 지정 SMS 발신자 트리거를 사용자 풀에 할당하면 Amazon Cognito가 사용자 이벤트로 인해 SMS 메시지를 전송해야 하는 경우 기본 동작 대신 Lambda 함수를 호출합니다. 사용자 지정 발신자 트리거를 사용하면 AWS Lambda 함수가 선택한 메서드와 공급자를 통해 사용자에게 SMS 알림을 보낼 수 있습니다. 함수의 사용자 지정 코드는 사용자 풀의 모든 SMS 메시지를 처리하고 전달해야 합니다.

이 트리거는 사용자 풀이 SMS 메시지를 보내는 방법을 더 잘 제어하려는 시나리오를 제공합니다. 예를 들어 여러 개의 발신 ID를 관리하거나 AWS 리전간에 사용하려는 경우와 같이 Lambda 함수를 사용하여 Amazon SNS API 작업에 대한 호출을 사용자 지정할 수 있습니다. 함수는 메시지를 다른 전송 미디어 또는 타사 서비스로 리디렉션할 수도 있습니다.

사용자 지정 이메일 발신자 트리거를 구성하는 방법은 사용자 지정 발신자 Lambda 트리거 활성화 섹션을 참조하세요.

사용자 지정 SMS 발신자 Lambda 트리거 소스

다음 표에는 Lambda 코드의 사용자 지정 SMS 트리거 소스에 대한 트리거 이벤트가 나와 있습니다.

`TriggerSource value`	Event
`CustomSMSSender_SignUp`	사용자가 가입하면 Amazon Cognito에서 시작 메시지를 보냅니다.
`CustomSMSSender_ForgotPassword`	사용자가 암호를 재설정하는 코드를 요청합니다.
`CustomSMSSender_ResendCode`	사용자가 새 코드를 요청하여 등록을 확인합니다.

TriggerSource value	Event
CustomSMSSender_VerifyUserAttribute	사용자가 새 이메일 주소 또는 전화 번호 속성을 생성하면 Amazon Cognito에서 속성을 확인하는 코드를 보냅니다.
CustomSMSSender_UpdateUserAttribute	사용자가 이메일 주소 또는 전화 번호 속성을 업데이트하면 Amazon Cognito에서 속성을 확인하는 코드를 보냅니다.
CustomSMSSender_Authentication	사용자가 로그인하면 Amazon Cognito가 SMS OTP 또는 MFA 코드를 전송합니다.
CustomSMSSender_AdminCreateUser	사용자 풀에 새 사용자를 생성하면 Amazon Cognito에서 해당 사용자에게 임시 암호를 보냅니다.

사용자 지정 SMS 발신자 Lambda 트리거 파라미터

Amazon Cognito가 이 Lambda 함수에 전달하는 요청은 아래 파라미터와 Amazon Cognito가 모든 요청에 추가하는 공통 파라미터의 조합입니다.

JSON

```
{
    "request": {
        "type": "customSMSSenderRequestV1",
        "code": "string",
        "clientMetadata": {
            "string": "string",
             . . .
            },
        "userAttributes": {
            "string": "string",
            . . .
         }
}
```

사용자 지정 SMS 발신자 요청 파라미터

type

요청 버전입니다. 사용자 지정 SMS 발신자 이벤트의 경우 이 문자열의 값은 항상 `customSMSSenderRequestV1`입니다.

code

함수가 해독하여 사용자에게 보낼 수 있는 암호화된 코드입니다.

clientMetadata

사용자 지정 SMS 발신자 Lambda 함수 트리거에 대한 사용자 지정 입력으로 제공할 수 있는 하나 이상의 키-값 페어입니다. 이 데이터를 Lambda 함수에 전달하려면 AdminRespondToAuthChallenge 및 RespondToAuthChallenge API 작업에서 ClientMetadata 파라미터를 사용합니다. Amazon Cognito는 사후 승인 함수에 전달하는 요청에 있는 AdminInitiateAuth 및 InitiateAuth API 작업의 ClientMetadata 파라미터에서 전달된 데이터를 포함하지 않습니다.

userAttributes

사용자 속성을 나타내는 하나 이상의 키-값 페어입니다.

사용자 지정 SMS 발신자 응답 파라미터

Amazon Cognito는 응답에서 추가 반환 정보를 기대하지 않습니다. 함수는 API 작업을 사용하여 리소스를 쿼리 및 수정하거나 이벤트 메타데이터를 외부 시스템에 기록할 수 있습니다.

코드 예제

다음 Node.js 예제에서는 사용자 지정 SMS 발신자 Lambda 함수에서 SMS 메시지 이벤트를 처리합니다. 이 예에서는 함수에 두 개의 환경 변수가 정의되어 있다고 가정합니다.

`KEY_ID`

사용자 코드를 암호화하고 복호화하는 데 사용하려는 KMS 키의 ID입니다.

`KEY_ARN`

사용자 코드를 암호화하고 복호화하는 데 사용하려는 KMS 키의 Amazon 리소스 이름(ARN)입니다.

이 함수를 배포하려면

1. 개발자 워크스페이스에 최신 버전의 NodeJS를 설치합니다.
2. 워크스페이스에서 새 NodeJS 프로젝트를 생성합니다.
3. `npm init -y`를 사용하여 프로젝트를 초기화합니다.
4. Lambda 함수 `touch index.mjs`의 스크립트를 생성합니다.
5. 아래 예제의 내용을 `index.mjs`에 붙여 넣습니다.
6. 프로젝트 종속성 다운로드 AWS Encryption SDK: `npm install @aws-crypto/client-node`.
7. 프로젝트 디렉토리를 `zip -r my_deployment_package.zip .`이라는 파일로 압축합니다.
8. ZIP 파일을 함수에 배포합니다.

```
import { KmsKeyringNode, buildClient, CommitmentPolicy } from '@aws-crypto/client-
node';

// Configure the encryption SDK client with the KMS key from the environment variables
const { encrypt, decrypt } = buildClient(
    CommitmentPolicy.REQUIRE_ENCRYPT_ALLOW_DECRYPT
);

const generatorKeyId = process.env.KEY_ID;
const keyIds = [process.env.KEY_ARN];
const keyring = new KmsKeyringNode({ generatorKeyId, keyIds });

// Example function to simulate sending SMS.
// This example logs message details to CloudWatch Logs from your Lambda function.
// Update this function with custom logic that sends an SMS message to 'phoneNumber'
 with body 'message'.
const sendSMS = async (phoneNumber, message) => {
    // Log the destination with the phone number masked.
    console.log(`Simulating SMS send to ${phoneNumber.replace(/[^+]/g, '*')}`);
    // Log the message with the code masked.
    console.log(`Message content: ${message.replace(/\b\d{6,8}\b/g, '********')}`);
    // Simulate API delay
    await new Promise(resolve => setTimeout(resolve, 100));
    console.log('SMS sent successfully');
    return true;
};
```

```
export const handler = async (event) => {
    try {
        // Decrypt the secret code using encryption SDK
        let plainTextCode;
        if (event.request.code) {
            const { plaintext, messageHeader } = await decrypt(keyring,
 Buffer.from(event.request.code, 'base64'));
            plainTextCode = Buffer.from(plaintext).toString('utf-8');
        }

        // Handle different trigger sources
        if (event.triggerSource == 'CustomSMSSender_SignUp') {
            const phoneNumber = event.request.userAttributes.phone_number;
            const message = `Welcome! Your verification code is: ${plainTextCode}`;
            await sendSMS(phoneNumber, message);
        }
        else if (event.triggerSource == 'CustomSMSSender_ResendCode') {
            // Handle resend code
        }
        else if (event.triggerSource == 'CustomSMSSender_ForgotPassword') {
            // Handle forgot password
        }
        else if (event.triggerSource == 'CustomSMSSender_UpdateUserAttribute') {
            // Handle update attribute
        }
        else if (event.triggerSource == 'CustomSMSSender_VerifyUserAttribute') {
            // Handle verify attribute
        }
        else if (event.triggerSource == 'CustomSMSSender_AdminCreateUser') {
            // Handle admin create user
        }
        return;
    } catch (error) {
        console.error('Error in custom SMS sender:', error);
        throw error;
    }
};
```

주제

- 사용자 지정 SMS 발신자 함수로 SMS 메시지 기능 평가

사용자 지정 SMS 발신자 함수로 SMS 메시지 기능 평가

사용자 지정 SMS 발신자 Lambda 함수는 사용자 풀이 보낼 SMS 메시지를 수락하고, 함수는 사용자 지정 논리를 기반으로 콘텐츠를 전달합니다. Amazon Cognito는 사용자 지정 SMS 발신자 Lambda 트리거 파라미터를 사용자의 함수에 보냅니다. 함수는 이 정보로 원하는 것을 할 수 있습니다. 예를 들어, 코드를 Amazon SNS Simple Notification Service(Amazon SNS) 주제에 보낼 수 있습니다. Amazon SNS 주제 구독자는 SMS 메시지, HTTPS 엔드포인트 또는 이메일 주소일 수 있습니다.

사용자 지정 SMS 발신자 Lambda 함수를 사용하여 Amazon Cognito SMS 메시징에 대한 테스트 환경을 생성하려면 GitHub의 aws-samples 라이브러리의 amazon-cognito-user-pool-development-and-testing-with-sms-redirected-to-email을 참조하세요. 리포지토리에는 새 사용자 풀을 생성하거나 이미 보유한 사용자 풀로 작업할 수 있는 AWS CloudFormation 템플릿이 포함되어 있습니다. 이러한 템플릿은 Lambda 함수와 Amazon SNS 주제를 생성합니다. 템플릿이 사용자 지정 SMS 발신자 트리거로 할당하는 Lambda 함수는 SMS 메시지를 Amazon SNS 주제의 구독자로 리디렉션합니다.

이 솔루션을 사용자 풀에 배포하면 Amazon Cognito가 일반적으로 SMS 메시징을 통해 보내는 모든 메시지, 즉 Lambda 함수는 대신 중앙 이메일 주소로 보냅니다. 이 솔루션을 사용하여 SMS 메시지를 사용자 지정하고 미리 보고 Amazon Cognito에서 SMS 메시지를 보내도록 하는 사용자 풀 이벤트를 테스트할 수 있습니다. 테스트를 완료한 후 CloudFormation 스택을 롤백하거나 사용자 풀에서 사용자 지정 SMS 발신자 함수 할당을 제거합니다.

> **Important**
>
> amazon-cognito-user-pool-development-and-testing-with-sms-redirected-to-email의 템플릿을 사용하여 프로덕션 환경을 구축하지 않습니다. 솔루션의 사용자 지정 SMS 발신자 Lambda 함수에서 SMS 메시지를 시뮬레이션하지만 Lambda 함수는 이 모든 메시지를 하나의 중앙 이메일 주소로 보냅니다. 프로덕션 Amazon Cognito 사용자 풀에서 SMS 메시지를 보내려면 먼저 Amazon Cognito 사용자 풀의 SMS 메시지 설정에 표시된 요구 사항을 완료해야 합니다.

메시징 및 분석

사용자 풀에 대한 이메일 및 SMS 메시징을 구성하고 사용자 참여를 추적하도록 Amazon Pinpoint 분석을 설정합니다.

주제

- Amazon Cognito 사용자 풀에 대한 이메일 설정
- Amazon Cognito 사용자 풀의 SMS 메시지 설정

- 사용자 풀 분석에 Amazon Pinpoint 사용

Amazon Cognito 사용자 풀에 대한 이메일 설정

애플리케이션의 특정 이벤트로 인해 Amazon Cognito가 사용자에게 이메일을 보낼 수 있습니다. 예를 들어 이메일 확인을 요구하도록 사용자 풀을 구성한 경우 사용자가 앱에서 새 계정을 등록하거나 암호를 다시 설정하면 Amazon Cognito에서 이메일을 보냅니다. 이메일을 시작한 작업이 무엇인지에 따라 이메일에는 확인 코드가 들어가거나 임시 암호가 들어갑니다.

이메일 전송을 처리하려면 다음 옵션 중 하나를 사용할 수 있습니다.

- Amazon Cognito 서비스에 내장된 기본 이메일 구성.
- Amazon Simple Email Service(Amazon SES) 구성

사용자 풀을 생성한 후 전송 옵션을 변경할 수 있습니다.

Amazon Cognito는 사용자가 입력할 수 있는 코드나 선택할 수 있는 URL 링크가 포함된 이메일 메시지를 사용자에게 보냅니다. 다음 표에서는 이메일 메시지를 생성할 수 있는 이벤트를 보여줍니다.

메시지 옵션

활동	API 작업	전송 옵션	포맷 옵션	사용자 지정 가능	메시지 템플릿
Forgot password	ForgotPassword, AdminResetUserPassword	Email, SMS	code	Yes	검증 메시지
Invitation	AdminCreateUser	Email, SMS	code	Yes	초대 메시지
Self-registration	SignUp, ResendConfirmationCode	Email, SMS	code, link	Yes	검증 메시지
Email address or phone	UpdateUserAttributes, AdminUpda	Email, SMS	code	Yes	검증 메시지

활동	API 작업	전송 옵션	포맷 옵션	사용자 지정 가능	메시지 템플릿
number verification	teUserAttributes, GetUserAttributeVerificationCode				
Multi-factor authentication (MFA)	AdminInitiateAuth, InitiateAuth	Email[1], SMS, authenticator app	code	Yes[2]	MFA 메시지
One-time password authentication (OTP)	AdminInitiateAuth, InitiateAuth	Email[1], SMS	code	Yes	MFA 메시지[3]

[1] Essentials 기능 플랜 이상 및 Amazon SES 이메일 구성이 필요합니다.

[2] SMS 및 이메일 메시지의 경우.

[3] 사용자 풀에 MFA가 필요하거나 선택 사항인 경우에만 MFA 메시지 템플릿을 사용자 지정할 수 있습니다. MFA가 비활성화되면 Amazon Cognito는 기본 템플릿과 함께 일회용 암호를 전송합니다.

Amazon SES는 이메일 메시지에 대해 요금을 부과합니다. 자세한 내용은 Amazon SES 요금을 참조하세요.

이메일 MFA에 대한 자세한 내용은 SMS 및 이메일 메시지 MFA 섹션을 참조하세요.

Amazon Cognito는 짧은 시간 내에 단일 대상으로 추가 이메일 또는 SMS 메시지를 전송하는 것을 방지할 수 있습니다. 사용자 풀이 영향을 받았다고 생각되면 메시지 전달 오류 로그를 구성하고 검토한 후 계정 팀에 문의하세요.

기본 이메일 구성

Amazon Cognito는 해당 기본 이메일 구성을 사용하여 이메일 전송을 처리할 수 있습니다. 기본 옵션을 사용하면 Amazon Cognito는 사용자 풀에 대해 매일 제한된 수의 이메일 보내기만 허용합니다. 서비스 제한에 대한 자세한 내용은 Amazon Cognito의 할당량 섹션을 참조하세요. 일반적인 프로덕션 환경의 경우 기본 이메일 제한은 필수 전송 볼륨보다 적습니다. 더 많은 전송량을 활성화하려면 Amazon SES 이메일 구성을 사용할 수 있습니다.

기본 구성을 사용하는 경우 AWS 에서 관리하는 Amazon SES 리소스를 사용하여 이메일 메시지를 보낼 수 있습니다. Amazon SES는 하드 바운스를 반환하는 이메일 주소를 계정 수준 억제 목록 또는 글로벌 억제 목록에 추가합니다. 전달 불가능한 이메일 주소가 나중에 전달 가능한 이메일 주소가 되는 경우 사용자 풀이 기본 구성을 사용하도록 구성되어 있는 동안에는 제외 목록에서의 이메일 주소 제거를 제어할 수 없습니다. 이메일 주소는 AWS관리형 금지 목록에 무기한 남아 있을 수 있습니다. 전달 불가능한 이메일 주소를 관리하려면 다음 섹션의 설명대로 Amazon SES 이메일 구성을 계정 수준 억제 목록과 함께 사용합니다.

기본 이메일 구성을 사용하면 다음 이메일 주소 중 하나를 발신 주소로 사용할 수 있습니다.

- 기본 이메일 주소 no-reply@verificationemail.com입니다.
- 사용자 지정 이메일 주소. 자신의 이메일 주소를 사용하려면 먼저 Amazon SES로 확인해야 하며 이 주소를 사용할 수 있는 권한을 Amazon Cognito에 부여해야 합니다.

Amazon SES 이메일 구성

애플리케이션에 기본 옵션으로 제공되는 것보다 더 많은 전송 볼륨이 필요할 수도 있습니다. 가능한 전송량을 늘리려면 Amazon SES 리소스를 사용자 풀과 함께 사용하여 사용자에게 이메일을 보냅니다. 자체 Amazon SES 구성을 사용하여 이메일 메시지를 보낼 때 이메일 보내기 활동을 모니터링할 수도 있습니다.

Amazon SES 구성을 사용하려면 먼저 Amazon SES로 하나 이상의 이메일 주소 또는 도메인을 확인해야 합니다. 확인된 이메일 주소 또는 확인된 도메인에 속한 주소를 사용자 풀에 할당한 발신 이메일 주소로 사용합니다. Amazon Cognito에서 사용자에게 이메일을 보내면 Amazon SES가 호출되고 이메일 주소를 사용합니다.

Amazon SES 구성을 사용할 때 다음 조건이 적용됩니다.

- 사용자 풀에 대한 이메일 전송 제한은 AWS 계정에서 Amazon SES 확인 이메일 주소에 적용되는 것과 동일한 제한입니다.
- 글로벌 억제 목록을 재정의하는 Amazon SES의 계정 수준 억제 목록을 사용하여 전달 불가능한 이메일 주소에 대한 메시지를 관리할 수 있습니다. 계정 수준 억제 목록을 사용할 때 이메일 메시지 반송은 발신자로서의 계정 평판에 영향을 미칩니다. 자세한 내용은 Amazon Simple Email Service 개발자 안내서의 Amazon SES 계정 수준 금지 목록 사용을 참조하세요.

Amazon SES 이메일 구성 리전

사용자 풀을 생성하는 AWS 리전 에는 Amazon SES를 사용한 이메일 메시지 구성에 대한 세 가지 요구 사항 중 하나가 있습니다. 사용자 풀과 동일한 리전, 동일한 리전을 포함한 여러 리전 또는 하나 이상의 원격 리전에 있는 Amazon SES에서 이메일 메시지를 보낼 수 있습니다. 최상의 성능을 위해 옵션이 있는 경우 사용자 풀과 동일한 리전에서 Amazon SES 인증 ID로 이메일 메시지를 전송합니다.

Amazon SES 인증 ID에 대한 리전 요구 사항 범주

리전 내 전용

사용자 풀은 AWS 리전 사용자 풀과 동일한에서 확인된 자격 증명이 있는 이메일 메시지를 보낼 수 있습니다. 사용자 지정 `FROM` 이메일 주소가 없는 기본 이메일 구성에서 Amazon Cognito는 동일한 리전에서 `no-reply@verificationemail.com` 인증 ID를 사용합니다.

이전 버전과의 호환성

사용자 풀은 동일한 AWS 리전 또는 다음 대체 리전 중 하나에서 확인된 자격 증명이 있는 이메일 메시지를 보낼 수 있습니다.

- 미국 동부(버지니아 북부)
- US West (Oregon)
- 유럽(아일랜드)

이 기능은 서비스가 시작될 때 Amazon Cognito 요구 사항에 맞게 생성했을 수 있는 사용자 풀 리소스의 연속성을 지원합니다. 해당 기간의 사용자 풀은 제한된 수의 자격 증명으로만 이메일 메시지를 보낼 수 있습니다 AWS 리전. 사용자 지정 `FROM` 이메일 주소가 없는 기본 이메일 구성에서 Amazon Cognito는 동일한 리전에서 `no-reply@verificationemail.com` 인증 ID를 사용합니다.

대체 리전

사용자 풀은 사용자 풀 리전 외부에 AWS 리전 있는 대체에서 확인된 자격 증명이 있는 이메일 메시지를 보낼 수 있습니다. 이 구성은 Amazon Cognito를 사용할 수 있는 리전에서 Amazon SES를 사용할 수 없는 경우에 발생합니다.

대체 리전의 인증 ID에 대한 Amazon SES 전송 권한 부여 정책은 발신 리전의 Amazon Cognito 서비스 보안 주체를 신뢰해야 합니다. 자세한 내용은 기본 이메일 구성을 사용할 수 있는 권한을 부여하려면 섹션을 참조하세요.

이러한 리전 중 일부에서 Amazon Cognito는 `COGNITO_DEFAULT`의 기본 이메일 구성을 위해 두 개의 대체 리전 간에 이메일 메시지를 분할합니다. 이러한 경우 사용자 지정 `FROM` 이메일 주소를 사

용하려면 각 대체 리전의 인증 ID에 대한 Amazon SES 전송 권한 부여 정책이 발신 리전의 Amazon Cognito 서비스 보안 주체를 신뢰해야 합니다. 자세한 내용은 기본 이메일 구성을 사용할 수 있는 권한을 부여하려면 섹션을 참조하세요. 이러한 리전에서 DEVELOPER의 Amazon SES 이메일 구성을 사용하는 경우 첫 번째로 나열된 리전에서 인증 ID를 사용하고 사용자 풀 리전의 Amazon Cognito 서비스 보안 주체를 신뢰하도록 구성해야 합니다. 예를 들어 중동(UAE)의 사용자 풀에서 `cognito-idp.me-central-1.amazonaws.com`을 신뢰하도록 유럽(프랑크푸르트)에서 인증 ID를 구성합니다. 사용자 지정 FROM 이메일 주소가 없는 기본 이메일 구성에서 Amazon Cognito는 각 리전에서 `no-reply@verificationemail.com` 인증 ID를 사용합니다.

> **Note**
>
> 다음 조건 조합에서는 리전 요소의 와일드카드를 사용하여 EmailConfiguration의 SourceArn 파라미터를 `arn:${Partition}:ses:*:${Account}:identity/${IdentityName}` 형식으로 지정해야 합니다. 이렇게 하면 사용자 풀이 두의에서 동일한 확인된 자격 증명으로 이메일 메시지를 보낼 수 AWS 계정 있습니다 AWS 리전.
>
> - EmailSendingAccount는 COGNITO_DEFAULT입니다.
> - 사용자 지정 FROM 주소를 사용하려고 합니다.
> - 사용자 풀은 대체 리전에서 이메일을 보냅니다.
> - 사용자 풀에는 다음 Amazon SES 지원 리전의 테이블에 지정된 두 번째[1] 대체 리전이 있습니다.

AWS SDK, Amazon Cognito API 또는 CLI AWS CDK, 또는 AWS CloudFormation를 사용하여 프로그래밍 방식으로 사용자 풀을 생성하는 경우 사용자 풀은 EmailConfiguration의 SourceArn 파라미터가 사용자 풀에 지정하는 Amazon SES 자격 증명이 포함된 이메일 메시지를 전송합니다. Amazon SES 자격 증명은 지원되는를 사용해야 합니다 AWS 리전. EmailSendingAccount가 COGNITO_DEFAULT이고 SourceArn 파라미터가 지정되지 않은 경우 Amazon Cognito는 사용자 풀이 생성된 리전의 리소스를 사용하여 `no-reply@verificationemail.com`에서 이메일 메시지를 보냅니다.

다음 표에는 Amazon Cognito에서 Amazon SES 자격 증명을 사용할 수 있는 AWS 리전 가 나와 있습니다. Amazon Cognito

사용자 풀 리전	리전 옵션	Amazon SES 지원 리전
미국 동부(버지니아 북부)	이전 버전과의 호환성	미국 서부(오레곤), 미국 동부(버지니아 북부), 유럽(아일랜드)
미국 동부(오하이오)	이전 버전과의 호환성	미국 서부(오하이오), 미국 동부(버지니아 북부), 미국 서부(오레곤), 유럽(아일랜드)
미국 서부(캘리포니아 북부)	리전 내 전용	미국 서부(캘리포니아 북부)
미국 서부(오리건)	이전 버전과의 호환성	미국 동부(버지니아 북부), 미국 서부(오레곤), 유럽(아일랜드)
캐나다(중부)	이전 버전과의 호환성	캐나다(중부), 미국 동부(버지니아 북부), 미국 서부(오레곤), 유럽(아일랜드)
캐나다 서부(캘거리)	대체 리전	캐나다(중부), 미국 서부(캘리포니아 북부)[1]
멕시코(중부)	대체 리전	미국 동부(버지니아 북부), 미국 서부(오리건)[1]
아시아 태평양(도쿄)	이전 버전과의 호환성	아시아 태평양(도쿄), 미국 동부(버지니아 북부), 미국 서부(오레곤), 유럽(아일랜드)
아시아 태평양(홍콩)	대체 리전	아시아 태평양(싱가포르), 아시아 태평양(도쿄)[1]
아시아 태평양(서울)	이전 버전과의 호환성	아시아 태평양(서울), 미국 동부(버지니아 북부), 미국 서부(오레곤), 유럽(아일랜드)

사용자 풀 리전	리전 옵션	Amazon SES 지원 리전
아시아 태평양(말레이시아)	대체 리전	아시아 태평양(시드니), 아시아 태평양(싱가포르)[1]
아시아 태평양(태국)	대체 리전	아시아 태평양(싱가포르), 아시아 태평양(뭄바이)[1]
아시아 태평양(뭄바이)	이전 버전과의 호환성	아시아 태평양(뭄바이), 미국 동부(버지니아 북부), 미국 서부(오레곤), 유럽(아일랜드)
아시아 태평양(하이데라바드)	대체 리전	아시아 태평양(뭄바이), 아시아 태평양(싱가포르)[1]
아시아 태평양(싱가포르)	이전 버전과의 호환성	아시아 태평양(싱가포르), 미국 동부(버지니아 북부), 미국 서부(오레곤), 유럽(아일랜드)
아시아 태평양(시드니)	이전 버전과의 호환성	아시아 태평양(시드니), 미국 동부(버지니아 북부), 미국 서부(오레곤), 유럽(아일랜드)
아시아 태평양(오사카)	리전 내 전용	아시아 태평양(오사카)
아시아 태평양(자카르타)	리전 내 전용	아시아 태평양(자카르타)
아시아 태평양(멜버른)	대체 리전	아시아 태평양(시드니), 아시아 태평양(싱가포르)[1]
유럽(아일랜드)	이전 버전과의 호환성	미국 서부(오레곤), 미국 동부(버지니아 북부), 유럽(아일랜드)
유럽(런던)	이전 버전과의 호환성	유럽(런던), 미국 동부(버니지아 북부), 미국 서부(오레곤), 유럽(아일랜드)
유럽(파리)	리전 내 전용	유럽(파리)

사용자 풀 리전	리전 옵션	Amazon SES 지원 리전
유럽(프랑크푸르트)	이전 버전과의 호환성	유럽(프랑크푸르트), 미국 동부(버니지아 북부), 미국 서부(오레곤), 유럽(아일랜드)
유럽(취리히)	대체 리전	유럽(프랑크푸르트), 유럽(런던)[1]
유럽(스톡홀름)	리전 내 전용	유럽(스톡홀름)
유럽(밀라노)	리전 내 전용	유럽(밀라노)
유럽(스페인)	대체 리전	유럽(파리), 유럽(스톡홀름)[1]
Middle East (Bahrain)	리전 내 전용	Middle East (Bahrain)
중동(UAE)	대체 리전	유럽(프랑크푸르트), 유럽(런던)[1]
남아메리카(상파울루)	리전 내 전용	남아메리카(상파울루)
이스라엘(텔아비브)	리전 내 전용	이스라엘(텔아비브)
아프리카(케이프타운)	리전 내 전용	아프리카(케이프타운)

[1] 기본 이메일 구성이 있는 사용자 풀에 사용됩니다. Amazon Cognito는 각 리전에서 동일한 이메일 주소를 사용하여 확인된 자격 증명 간에 이메일 메시지를 배포합니다. 사용자 지정 FROM 주소를 사용하려면 `arn:${Partition}:ses:*:${Account}:identity/${IdentityName}` 형식의 `SourceArn` 파라미터로 `EmailConfiguration`을 구성합니다.

사용자 풀에 대한 이메일 구성

사용자 풀에 대한 이메일 설정을 구성하려면 다음 단계를 완료하세요. 사용하는 설정에 따라 Amazon SES, AWS Identity and Access Management (IAM) 및 Amazon Cognito에서 IAM 권한이 필요할 수도 있습니다.

> **Note**
>
> 이러한 단계에서 생성되는 리소스는 AWS 계정간에 공유할 수 없습니다. 예를 들어 사용자 풀을 구성한 한 계정을 다른 계정의 Amazon SES 이메일 주소로 사용할 수 없습니다. Amazon Cognito를 여러 계정에서 사용하는 경우 각 계정에 대해 이러한 단계를 반복합니다.

1단계: Amazon SES로 이메일 주소 또는 도메인 확인

사용자 풀을 구성하기 전에 다음 중 하나를 수행하려면 Amazon SES로 하나 이상의 도메인 또는 이메일 주소를 확인해야 합니다.

- 자신의 이메일 주소를 발신 주소로 사용합니다.
- Amazon SES 구성을 사용하여 이메일 전송을 처리합니다.

이메일 주소나 도메인을 확인하여 이메일 주소를 소유하고 있음을 확인하면 무단 사용을 방지하는 데 도움이 됩니다.

Amazon SES에서 이메일 확인에 관한 자세한 내용은 Amazon Simple Email Service 개발자 가이드에서 Amazon SES에서 이메일 주소 확인을 참조하세요. Amazon SES로 도메인을 확인하는 방법에 대한 자세한 내용은 도메인 확인을 참조하세요.

2단계: Amazon SES 샌드박스에서 계정 이동

기본 Amazon Cognito 이메일 구성을 사용하는 경우 이 단계를 건너뜁니다.

에서 Amazon SES AWS 리전를 처음 사용하는 경우 해당 리전의 Amazon SES 샌드박스 AWS 계정 에이 배치됩니다. Amazon SES는 사기 및 부정 사용을 방지하기 위해 샌드박스를 사용합니다. Amazon SES 구성을 사용하여 이메일 전송을 처리하는 경우 먼저 AWS 계정 을 샌드박스 외부로 이동해야 Amazon Cognito가 사용자에게 이메일을 보낼 수 있습니다.

샌드박스에서 Amazon SES는 전송할 수 있는 이메일 수와 전송할 수 있는 위치를 제한합니다. Amazon SES로 확인한 주소 및 도메인에만 이메일을 보내거나 Amazon SES 메일박스 시뮬레이터 주소로 이메일을 보낼 수 있습니다. 가 샌드박스에 AWS 계정 있는 동안에는 프로덕션 환경에 있는 애플리케이션에 Amazon SES 구성을 사용하지 마십시오. 이 경우 Amazon Cognito에서 사용자의 이메일 주소로 메시지를 보낼 수 없습니다.

샌드박스 AWS 계정 에서를 제거하려면 Amazon Simple Email Service 개발자 안내서의 Amazon SES 샌드박스 밖으로 이동을 참조하세요.

3단계: Amazon Cognito에 이메일 권한 부여

사용자에게 이메일을 보내기 전에 특정 권한을 Amazon Cognito에 부여해야 할 수도 있습니다. 사용자가 부여하는 권한과 권한을 부여하는 데 사용하는 프로세스는 기본 이메일 구성을 사용하는지 아니면 Amazon SES 구성을 사용하는지에 따라 다릅니다.

기본 이메일 구성을 사용할 수 있는 권한을 부여하려면

사용자 풀을 Cognito를 사용하여 이메일 보내기로 구성하거나 `EmailSendingAccount`를 `COGNITO_DEFAULT`로 설정한 경우에만 이 단계를 완료합니다.

기본 이메일 구성을 사용하면 사용자 풀에서 다음 주소 중 하나를 사용하여 이메일 메시지를 보낼 수 있습니다.

- 기본 주소 `no-reply@verificationemail.com`.
- Amazon SES에서 확인된 이메일 주소 또는 도메인의 사용자 지정 발신 주소입니다.

사용자 정의 주소를 사용하는 경우 Amazon Cognito에 추가 권한이 있어야 해당 주소에서 사용자에게 이메일을 보낼 수 있습니다. 이러한 권한은 Amazon SES의 주소나 도메인에 대한 전송 권한 부여 정책을 통해 부여됩니다. Amazon Cognito 콘솔을 사용하여 사용자 풀에 사용자 정의 주소를 추가하는 경우 정책이 Amazon SES 확인 이메일 주소에 자동으로 연결됩니다. 그러나 AWS CLI 또는 Amazon Cognito API를 사용하는 등 콘솔 외부에서 사용자 풀을 구성하는 경우 Amazon SES 콘솔 또는 PutIdentityPolicy API를 사용하여 정책을 연결해야 합니다.

> **Note**
>
> AWS CLI 또는 Amazon Cognito API를 사용하여 확인된 도메인의 FROM 주소만 구성할 수 있습니다.

전송 권한 부여 정책은 Amazon Cognito를 사용하여 Amazon SES를 호출하는 계정 리소스를 기준으로 액세스를 허용하거나 거부합니다. 리소스 기반 정책에 대한 자세한 내용은 IAM 사용 설명서를 참조하세요. Amazon SES 개발자 가이드에서 리소스 기반 정책의 예제도 찾아볼 수 있습니다.

Example전송 권한 부여 정책

다음 전송 권한 부여 정책 예제는 Amazon Cognito에 Amazon SES 확인 자격 증명을 사용할 수 있는 제한적인 권한을 부여합니다. Amazon Cognito는 `aws:SourceArn` 조건의 사용자 풀과 `aws:SourceAccount` 조건의 계정을 둘 다 대신하는 경우에만 이메일 메시지를 전송할 수 있습니다.

Regions with Amazon SES

사용자 풀 리전 또는 대체 리전의 전송 권한 부여 정책은 Amazon Cognito 서비스 보안 주체가 이메일 메시지를 전송하도록 허용해야 합니다. 자세한 내용은 리전 표를 참조하십시오. 사용자 풀 리전이 Amazon SES 리전에서 하나 이상의 값과 일치하는 경우 다음 예제에서 글로벌 서비스 보안 주체를 사용하여 전송 권한 부여 정책을 구성합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "stmnt1234567891234",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "email.cognito-idp.amazonaws.com"
                ]
            },
            "Action": [
                "SES:SendEmail",
                "SES:SendRawEmail"
            ],
            "Resource": "arn:aws:ses:us-
east-1:111122223333:identity/support@example.com",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:cognito-idp:us-
east-1:111122223333:userpool/us-east-1_EXAMPLE"
                }
            }
        }
    ]
}
```

Opt-in Regions without Amazon SES

Amazon Cognito를 사용할 수 있는 일부 옵트인에서는 Amazon SES를 사용할 수 없습니다. AWS 리전 Amazon Cognito 중동(UAE)가 한 예이며 유럽(프랑크푸르트)(`eu-central-1`)에서 인증 ID로 이메일만 보낼 수 있습니다. 기본 이메일 구성이 있는 사용자 풀에서 Amazon Cognito는 두 리전 각각에서 인증 ID로 이메일 메시지도 전송합니다. 중동(UAE)의 경우 추가 리전은 유럽(런던)입니다. 두 리전 모두에서 전송 권한 부여 정책을 업데이트해야 합니다.

각 대체 리전의 전송 권한 부여 정책은 사용자 풀 옵트인 리전의 Amazon Cognito 서비스 보안 주체가 이메일 메시지를 전송하도록 허용해야 합니다. 자세한 내용은 리전 표를 참조하십시오. 리전이 대체 리전으로 표시된 경우 다음 예제와 같이 리전 서비스 보안 주체를 사용하여 전송 권한 부여 정책을 구성합니다. 필요에 따라 예시 리전 식별자 *me-central-1*을 필수 리전 ID로 바꿉니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "cognito-idp.me-central-1.amazonaws.com"
                ]
            },
            "Action": [
                "SES:SendEmail",
                "SES:SendRawEmail"
            ],
            "Resource": "arn:aws:ses:us-
east-1:111122223333:identity/support@example.com",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:cognito-idp:us-
east-1:111122223333:userpool/us-east-1_EXAMPLE"
                }
            }
        }
    ]
```

```
}
```

정책 구문에 대한 자세한 내용은 Amazon Simple Email Service 개발자 가이드에서 Amazon SES 전송 권한 부여 정책을 참조하세요.

자세한 내용은 Amazon Simple Email Service 개발자 가이드에서 Amazon SES 전송 권한 부여 정책 예제를 참조하세요.

Amazon SES 구성을 사용할 수 있는 권한을 부여하려면

Amazon SES 구성을 사용하도록 사용자 풀을 구성한 경우 Amazon Cognito는 사용자에게 이메일을 보낼 때 사용자 대신 Amazon SES를 호출할 수 있는 추가 권한이 필요합니다. 이 권한은 IAM 서비스를 사용하여 부여됩니다.

이 옵션으로 사용자 풀을 구성하면 Amazon Cognito가 AWS 계정에서 IAM 역할 유형인 서비스 연결 역할을 생성합니다. 이 역할에는 Amazon Cognito에서 Amazon SES에 액세스하여 귀하의 주소로 이메일 전송을 허용하는 권한이 포함되어 있습니다.

Amazon Cognito는 구성을 설정하는 사용자 세션의 AWS 자격 증명을 사용하여 서비스 연결 역할을 생성합니다. 이 세션의 IAM 권한에는 `iam:CreateServiceLinkedRole` 작업이 포함되어야 합니다. IAM의 권한에 대한 자세한 내용은 IAM 사용 설명서의 AWS 리소스에 대한 액세스 관리를 참조하세요.

Amazon Cognito가 생성하는 서비스 연결 역할에 대한 자세한 내용은 Amazon Cognito에 서비스 연결 역할 사용 섹션을 참조하세요.

4단계: 사용자 풀 구성

다음 중 하나를 사용하여 사용자 풀을 구성하려면 다음 단계를 완료하세요.

- 이메일 발신자로 표시되는 사용자 지정 발신 주소
- 사용자가 발신 주소로 보낸 메시지를 받는 사용자 지정 회신 주소
- Amazon SES 구성

> **Note**
>
> 확인된 자격 증명이 이메일 주소인 경우 Amazon Cognito는 기본적으로 해당 이메일 주소를 FROM 및 REPLY-TO 이메일 주소로 설정합니다. 하지만 인증 ID가 도메인인 경우 FROM 이메일 주소의 값을 제공해야 합니다.

기본 Amazon Cognito 이메일 구성 및 주소를 사용하는 경우 이 절차를 생략합니다.

사용자 지정 이메일 주소를 사용하도록 사용자 풀을 구성하려면

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 자격 AWS 증명을 입력합니다.
2. 사용자 풀(User Pools)을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택합니다.
4. 인증 방법 메뉴를 선택하고 이메일 구성을 찾은 다음 편집을 선택합니다.
5. [이메일 구성 편집(Edit email configuration)] 페이지에서 [Amazon SES SES에서 이메일 전송(Send email from Amazon SES)] 또는 [Amazon Cognito로 이메일 전송(Send email with Amazon Cognito)]을 선택합니다. [Amazon SES에서 이메일 전송(Send email from Amazon SES)]을 선택한 경우에만 SES 리전(SES Region), 구성 집합(Configuration Set), FROM 발신자 이름(FROM sender name)을 사용자 정의할 수 있습니다.
6. 사용자 정의 FROM 주소를 사용하려면 다음 단계를 완료합니다.
   a. [SES 리전(SES Region)]에서 확인된 이메일 주소가 포함된 리전을 선택합니다.
   b. 발신 이메일 주소(FROM email address)에서 이메일 주소를 선택합니다. Amazon SES에서 확인된 이메일 주소를 사용합니다.
   c. (선택 사항) 구성 세트(Configuration set)에서 Amazon SES에 사용할 구성 세트를 선택합니다. 이 변경 사항을 수행하고 저장하면 서비스 연결 역할이 생성됩니다.
   d. (선택 사항) 발신 발신자 주소(FROM sender address) 아래에 이메일 주소를 입력합니다. 이메일 주소만 제공하거나, 이메일 주소와 친숙한 이름을 `Jane Doe <janedoe@example.com>` 포맷으로 제공할 수 있습니다.
   e. (선택 사항) [REPLY-TO 이메일 주소(REPLY-TO email address)] 아래에 사용자가 FROM 주소로 보내는 메시지를 받으려는 이메일 주소를 입력합니다.
7. 변경 사항 저장(Save changes)을 선택합니다.

관련 주제

- 이메일 확인 메시지 사용자 정의
- 사용자 초대 메시지 사용자 정의

Amazon Cognito 사용자 풀의 SMS 메시지 설정

사용자 풀에 대한 일부 Amazon Cognito 이벤트로 인해 Amazon Cognito가 사용자에게 SMS 문자 메시지를 보낼 수 있습니다. 예를 들어 전화 확인을 요구하도록 사용자 풀을 구성한 경우 사용자가 앱에서 새 계정을 등록하거나 암호를 다시 설정하면 Amazon Cognito에서 SMS 문자 메시지를 보냅니다. SMS 문자 메시지를 유발하는 작업에 따라 메시지에는 확인 코드, 임시 암호 또는 시작 메시지가 포함됩니다.

Amazon Cognito는 다음 구성 중 하나를 사용하여 SMS 메시지를 보낼 수 있습니다.

- Amazon Simple Notification Service(Amazon SNS) - Amazon Cognito는 SMS 메시지를 라우팅하는 Amazon SNS를 호출합니다 AWS 최종 사용자 메시징 SMS.
- AWS 최종 사용자 메시징 SMS direct - Amazon Cognito는 Amazon SNS를 중개자로 사용하지 않고 AWS 최종 사용자 메시징 SMS 직접를 호출합니다. Amazon Cognito를 사용할 수 AWS 리전 있는 대부분의에서이 옵션을 사용할 수 있습니다. 예외 사항은 AWS 최종 사용자 메시징 SMS 다이렉트 경로 리전 구성를 참조하세요.

사용자 풀은 한 번에 하나의 SMS 구성을 사용합니다. Amazon SNS와 AWS 최종 사용자 메시징 SMS 직접 구성은 상호 배타적입니다. 한 구성을 설정하면 Amazon Cognito가 다른 구성을 지웁니다. 이는 CreateUserPool, UpdateUserPool 및 SetUserPoolMfaConfig 작업에 적용됩니다.

Amazon SNS SMS는에 AWS 리전 나열된에서 사용할 수 있습니다Amazon SNS 경로 리전 구성. 다른의 사용자 풀에서 SMS 메시지를 보내려면를 AWS 최종 사용자 메시징 SMS 직접 AWS 리전사용합니다.

Amazon Cognito를 통해 처음으로 문자 메시지를 보내는 경우는 사용자를 샌드박스 환경에 AWS 최종 사용자 메시징 SMS 배치합니다. 샌드박스 환경에서는 애플리케이션에서 SMS 문자 메시지를 테스트할 수 있습니다. 샌드박스에서 발신 자격 증명이 있는 경우 제한된 수의 확인된 대상 전화번호로 메시지를 보내거나 그렇지 않은 경우 메시지 전송을 시뮬레이션할 수 있습니다.

> **Note**
>
> 2024년 11월에에서 Amazon SNS SMS 메시징을 AWS 로 대체했습니다 AWS 최종 사용자 메시징 SMS. Amazon SNS 구성 경로를 사용하면 사용자 풀은 패스스루인 Amazon SNS 게시 작업을 사용하여 SMS 메시지를 시작합니다 AWS 최종 사용자 메시징 SMS. 이 경로의 경우에 대한 권한을 구성해야 합니다`sns:Publish`. 직접 AWS 최종 사용자 메시징 SMS 구성 경로를 사용하면 Amazon Cognito가 AWS 최종 사용자 메시징 SMS `SendTextMessage` 작업을 호출하고에 대한 권한을 구성해야 합니다`sms-voice:SendTextMessage`.

AWS 최종 사용자 메시징 SMS 는 SMS 문자 메시지에 대해 요금을 부과합니다. 자세한 내용은 AWS 최종 사용자 메시징 SMS 요금을 참조하십시오.

Amazon Cognito는 사용자가 입력할 수 있는 코드와 함께 SMS 메시지를 사용자에게 보냅니다. 다음 표에서는 SMS 메시지를 생성할 수 있는 이벤트를 보여줍니다.

메시지 옵션

활동	API 작업	전송 옵션	포맷 옵션	사용자 지정 가능	메시지 템플릿
Forgot password	ForgotPassword, AdminResetUserPassword	Email, SMS	code	Yes	검증 메시지
Invitation	AdminCreateUser	Email, SMS	code	Yes	초대 메시지
Self-registration	SignUp, ResendConfirmationCode	Email, SMS	code, link	Yes	검증 메시지
Email address or phone number verification	UpdateUserAttributes, AdminUpdateUserAttributes, GetUserAttributeVerificationCode	Email, SMS	code	Yes	검증 메시지
Multi-factor authentication (MFA)	AdminInitiateAuth, InitiateAuth	Email[1], SMS, authenticator app	code	Yes[2]	MFA 메시지
One-time password authentication (OTP)	AdminInitiateAuth, InitiateAuth	Email[1], SMS	code	Yes	MFA 메시지[3]

[1] Essentials 기능 플랜 이상 및 Amazon SES 이메일 구성이 필요합니다.

[2] SMS 및 이메일 메시지의 경우.

[3] 사용자 풀에 MFA가 필요하거나 선택 사항인 경우에만 MFA 메시지 템플릿을 사용자 지정할 수 있습니다. MFA가 비활성화되면 Amazon Cognito는 기본 템플릿과 함께 일회용 암호를 전송합니다.

AWS 최종 사용자 메시징 SMS 는 SMS 메시지에 대해 요금을 부과합니다. 자세한 내용은 AWS 최종 사용자 메시징 SMS 요금을 참조하십시오.

MFA에 대한 자세한 내용은 SMS 및 이메일 메시지 MFA 섹션을 참조하세요.

Amazon Cognito는 짧은 시간 내에 단일 대상으로 추가 이메일 또는 SMS 메시지를 전송하는 것을 방지할 수 있습니다. 사용자 풀이 영향을 받았다고 생각되면 메시지 전달 오류 로그를 구성하고 검토한 후 계정 팀에 문의하세요.

Amazon SNS와 중에서 선택 AWS 최종 사용자 메시징 SMS

Amazon Cognito 사용자 풀은 Amazon SNS 또는 AWS 최종 사용자 메시징 SMS 직접이라는 두 경로를 통해 SMS 메시지를 보낼 수 있습니다. 새 사용자 풀의 경우 AWS 최종 사용자 메시징 SMS 직접 경로를 사용합니다. Amazon SNS SMS는 고정된 세트에서만 사용할 수 있습니다 AWS 리전. 직접 AWS 최종 사용자 메시징 SMS 경로는 추가 리전을 지원하고 Amazon SNS 경로가 지원하지 않는 기능을 노출합니다. 두 경로 모두 메시지를 전달합니다 AWS 최종 사용자 메시징 SMS.

다음 표에서는 두 SMS 구성 경로를 비교합니다.

기능	Amazon SNS 경로	AWS 최종 사용자 메시징 SMS 직접 경로
리전 가용성	Amazon SNS 경로 리전 테이블에 AWS 리전 나열된에서 사용 가능	Amazon Cognito를 사용할 수 AWS 리전 있는 대부분의에서 사용할 수 있습니다. Amazon SNS 경로만 사용할 수 있는 리전은에 나열되어 있습니다AWS 최종 사용자 메시징 SMS 다이렉트 경로 리전 구성.
통합	Amazon Cognito는 로 전달되는 Amazon SNS 게시 작업을	Amazon Cognito가 AWS 최종 사용자 메시징 SMS

기능	Amazon SNS 경로	AWS 최종 사용자 메시징 SMS 직접 경로
	호출합니다. AWS 최종 사용자 메시징 SMS	SendTextMessage 작업을 직접 호출합니다.
IAM 권한	`sns:Publish`	`sms-voice:SendTextMessage`
구성 세트	Amazon Cognito를 통해 구성할 수 없음	이벤트 로깅 및 대상에 대한 `ConfigurationSetName` 필드 설정
발신 ID	Amazon Cognito를 통해 구성할 수 없음	`OriginationIdentity` 필드 설정
리전 간 전송	`SnsRegion` 필드 설정	지원되지 않음
인도 DLT 등록	Amazon Cognito를 통해 구성할 수 없음	`InEntityId` 및 `InTemplateId` 필드 설정
사기 방지(보호)	계정 기본 보호 구성 적용	계정 기본값이 적용됩니다. 구성 세트를 통해 특정 사용 사례에 보호 구성을 적용할 수도 있습니다.

> **Note**
>
> 기본 AWS 최종 사용자 메시징 SMS 보호 구성은 Amazon Cognito의 메시지를 포함한 모든 메시지에 적용됩니다. 특정 사용 사례에 보호 구성을 적용하려면 구성 세트에 연결하고 직접 AWS 최종 사용자 메시징 SMS 경로를 사용합니다. 자세한 내용은 의 사기 방지 AWS 최종 사용자 메시징 SMS를 참조하세요.

Amazon Cognito 사용자 풀 API의 SMS 구성

Amazon Cognito 사용자 풀 API의 SmsConfigurationType 객체에는 사용자 풀에 대한 SMS 구성이 포함되어 있습니다. 이 객체에는 다음과 같은 구성 경로가 있습니다.

Amazon SNS 경로

Amazon Cognito에 SnsCallerArn를 호출할 수 있는 권한을 부여하는 IAM 역할의 ARN으로 설정합니다sns:Publish. 선택적으로 SnsRegion 및를 설정합니다ExternalId. AWS 최종 사용자 메시징 SMS 직접 경로를 사용하는 경우 SnsCallerArn 파라미터는 선택 사항입니다.

AWS 최종 사용자 메시징 SMS 직접 경로

유형의 EumsSms 객체를 설정합니다EumsSmsConfigurationType. 이 객체에는 다음과 같은 멤버가 있습니다.

- CallerArn (필수) - Amazon Cognito에를 호출할 수 있는 권한을 부여하는 IAM 역할의 ARN입니다sms-voice:SendTextMessage.
- ExternalId (선택 사항) - IAM 역할 신뢰 정책의 외부 ID입니다. 이 값은 혼동된 대리자 방지를 제공합니다.
- OriginationIdentity (선택 사항) - 전송에 사용할 전화번호, 풀 또는 발신자 ID입니다.
- ConfigurationSetName (선택 사항) - 사용할 AWS 최종 사용자 메시징 SMS 구성 세트입니다.
- InEntityId (선택 사항) - India DLT 등록의 엔터티 등록 ID입니다. 인도의 수신자에게 보내는 데 필요합니다.
- InTemplateId (선택 사항) - India DLT 등록의 템플릿 등록 ID입니다. 인도의 수신자에게 보내는 데 필요합니다.
- Region (선택 사항) - AWS 최종 사용자 메시징 SMS 리소스 AWS 리전 의 입니다. 자세한 내용은 SMS 메시지 AWS 리전 용 선택 단원을 참조하십시오.

자세한 내용은 Amazon Cognito 사용자 풀 API 참조의 SmsConfigurationType을 참조하세요. Amazon Cognito

모범 사례

원치 않는 SMS 트래픽 양의 전 세계적 증가로 인해 일부 정부는 SMS 메시지 발신자와 수신자 사이에 장벽을 부과하고 있습니다. MFA와 사용자 업데이트에 SMS 메시지를 사용하는 경우 메시지가 전달되도록 추가 조치를 취해야 합니다. 또한 사용자가 거주할 수 있는 국가의 SMS 메시지 관련 규정을 모니터링하고 SMS 메시지 구성을 최신 상태로 유지해야 합니다. 자세한 내용은 AWS 최종 사용자 메시징 SMS 사용 설명서에서 SMS 및 MMS 국가별 기능과 제한 사항을 참조하세요.

SMS 메시지를 사용한 사용자 인증과 확인은 보안 모범 사례가 아닙니다. 전화번호는 소유자가 바뀔 수 있으며, 사용자의 가지고 있는 것 MFA 요소를 확실히 대표하지 못할 수 있습니다. 대신 앱이나 서드

파티 IdP를 사용하여 TOTP MFA를 구현하세요. 사용자 정의 인증 챌린지 Lambda 트리거를 사용하여 추가 사용자 지정 인증 요소를 생성할 수도 있습니다.

SMS 메시지 전송 아키텍처 보안에 대한 자세한 내용은 다음 링크를 검토하세요.

- Amazon Cognito 사용자 풀을 사용하여 사용자 가입 사기 및 SMS 펌프의 위험 감소
- SMS Pumping 방지: 인공적으로 팽창된 트래픽에 대처하는 데 도움이 되는 새로운 AWS 기능

Amazon Cognito 사용자 풀에서 처음으로 SMS 메시지 설정

Amazon Cognito는 Amazon SNS(및 간접적으로 AWS 최종 사용자 메시징 SMS) 또는 AWS 최종 사용자 메시징 SMS 를 사용하여 사용자 풀에서 SMS 메시지를 보냅니다. 자체 리소스를 사용하는 사용자 지정 SMS 발신자 Lambda 트리거를 사용하여 SMS 메시지를 보낼 수도 있습니다. 특정에서 SMS 문자 메시지를 처음 설정하면가 해당 리전의 SMS 샌드박스 AWS 계정 에 AWS 리전 AWS 최종 사용자 메시징 SMS 배치됩니다. AWS 최종 사용자 메시징 SMS 는 샌드박스를 사용하여 사기 및 침해를 방지하고 규정 준수 요구 사항을 충족합니다. AWS 계정 가 샌드박스에 있는 경우는 몇 가지 제한을 AWS 최종 사용자 메시징 SMS 부과합니다. 예를 들어 발신 ID가 있는 경우 최대 10개의 확인된 대상 번호로 문자 메시지를 보내거나 발신 ID 없이 메시지 전송을 시뮬레이션할 수 있습니다. 가 샌드박스에 AWS 계정 있는 동안에는 프로덕션 환경에서 SMS 메시지를 보내지 마십시오. 샌드박스에서는 Amazon Cognito가 사용자의 전화 번호로 메시지를 보낼 수 없습니다.

주제

- Amazon Cognito가 SMS 메시지를 보내는 데 사용할 수 있는 IAM 역할 준비 AWS 최종 사용자 메시징 SMS
- 사용자 풀에 대한 AWS 최종 사용자 메시징 SMS 직접 SMS를 설정하려면
- SMS 메시지 AWS 리전 용 선택
- 미국 전화 번호로 SMS 메시지를 보내기 위한 발신 자격 증명 얻기
- 환경이 SMS 샌드박스인지 확인합니다.
- 샌드박스에서 계정 이동
- 에서 시뮬레이터 번호 또는 확인된 전화번호 사용 AWS 최종 사용자 메시징 SMS
- Amazon Cognito에서 사용자 풀 설정 완료

Amazon Cognito가 SMS 메시지를 보내는 데 사용할 수 있는 IAM 역할 준비 AWS 최종 사용자 메시징 SMS

사용자 풀에서 SMS 메시지를 보낼 때 Amazon Cognito는 계정에서 IAM 역할을 맡습니다. 역할에 필요한 권한은 선택한 SMS 구성 경로에 따라 다릅니다.

- Amazon SNS 경로 - 역할은 `sns:Publish` 권한을 부여해야 합니다. Amazon Cognito는이 권한을 사용하여 Amazon SNS를 통해 SMS 메시지를 전송합니다.
- AWS 최종 사용자 메시징 SMS 직접 경로 - 역할은 `sms-voice:SendTextMessage` 권한을 부여해야 합니다. Amazon Cognito는이 권한을 사용하여 AWS 최종 사용자 메시징 SMS 직접를 호출합니다.

Amazon Cognito 콘솔의 SMS 아래에 있는 사용자 풀의 인증 방법 메뉴에서 IAM 역할 선택을 설정하거나 사용자 풀 생성 마법사 중에 선택할 수 있습니다.

다음 예제에서는 IAM 역할 신뢰 정책은 Amazon Cognito 사용자 풀에 역할을 맡을 수 있는 제한된 권한을 부여합니다. Amazon Cognito는 다음 조건을 충족하는 경우에만 해당 역할을 수행할 수 있습니다.

- assume-role 작업은 `aws:SourceArn` 조건의 사용자 풀을 대신하여 수행됩니다.
- assume-role 작업은 `aws:SourceAccount` 조건에 의해 설정된 AWS 계정 의 사용자 풀을 대신하여 수행됩니다.
- assume-role 작업에는 `sts:externalId` 조건에 외부 ID가 포함됩니다.

이러한 신뢰 정책 조건은 Amazon SNS 경로와 AWS 최종 사용자 메시징 SMS 직접 경로 모두에 적용됩니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "cognito-idp.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
```

```
            "Condition": {
                "StringEquals": {
                    "sts:ExternalId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
                    "aws:SourceAccount": "111122223333"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:cognito-idp:us-
west-2:111122223333:userpool/us-west-2_EXAMPLE"
                }
            }
        }
    ]
}
```

`aws:SourceArn` 조건 값에 정확한 사용자 풀 ARN 또는 와일드카드 ARN을 지정할 수 있습니다. DescribeUserPool API 요청을 사용하여 AWS Management Console 또는에서 사용자 풀의 ARNs을 조회합니다.

다중 인증에 대한 SMS 메시지를 보내려면 IAM 역할 신뢰 정책에 `sts:ExternalId` 조건이 있어야 합니다. 이 조건의 값은 사용자 풀의 SmsConfiguration의 `ExternalId` 속성과 일치해야 합니다. Amazon Cognito 콘솔에서 사용자 풀을 생성하는 중에 IAM 역할을 생성하면 Amazon Cognito는 역할 및 사용자 풀 설정에서 외부 ID를 구성합니다. 기존 IAM 역할을 사용할 때는 그렇지 않습니다.

UpdateUserPool API 요청에서 사용자 풀 `ExternalId` 파라미터를 업데이트하고 IAM 역할 신뢰 정책을 동일한 값의 `sts:externalId` 조건으로 업데이트해야 합니다. API를 사용하여 원래 구성을 보존하는 방식으로 사용자 풀을 업데이트하는 방법을 알아보려면 사용자 풀 및 앱 클라이언트 구성 업데이트 섹션을 참조하세요.

IAM 역할 및 신뢰 정책에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 역할 용어 및 개념을 참조하세요.

사용자 풀에 대한 AWS 최종 사용자 메시징 SMS 직접 SMS를 설정하려면

다음 절차에 따라 SMS 메시지를 직접 보내도록 사용자 풀을 구성합니다 AWS 최종 사용자 메시징 SMS. 이 구성은 SmsConfigurationType의 `EumsSms` 객체를 사용합니다.

AWS 최종 사용자 메시징 SMS 직접 SMS 메시징을 설정하려면

1. 에서 AWS 리전 전송하려는 AWS 최종 사용자 메시징 SMS 발신 자격 증명을 생성하거나 식별합니다. 발신 자격 증명은 전화번호, 발신자 ID 또는 풀일 수 있습니다. 전송 규칙을 적용하기 위한

AWS 최종 사용자 메시징 SMS 구성 세트를 생성할 수도 있습니다. 전화번호 요청에 대한 지침은 AWS 최종 사용자 메시징 SMS 사용 설명서의 전화번호 요청을 참조하세요.

2. Amazon Cognito가 호출할 수 있는 IAM 역할을 생성합니다 AWS 최종 사용자 메시징 SMS. 이 역할에 대한 신뢰 정책은 Amazon SNS 경로와 동일하며 `aws:SourceArn`, `aws:SourceAccount`및 `sts:ExternalId` 조건을 포함해야 합니다. 신뢰 정책 세부 정보는 섹션을 참조하세요Amazon Cognito가 SMS 메시지를 보내는 데 사용할 수 있는 IAM 역할 준비 AWS 최종 사용자 메시징 SMS.

3. 다음 자격 증명(권한) 정책을 역할에 연결합니다. 이 정책은 사용자 풀이 보내는 발신 자격 증명에 `sms-voice:SendTextMessage` 대해를 호출할 수 있는 권한을 Amazon Cognito에 부여합니다. 최소 권한 관행으로 `Resource` 요소의 범위를 전화번호, 전화 풀 또는 발신자 ID의 ARN으로 지정합니다. 예제 ARN을 발신 자격 증명의 ARN으로 바꿉니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sms-voice:SendTextMessage"
            ],
            "Resource": "arn:aws:sms-voice:us-east-1:123456789012:phone-number/
phone-a1b2c3d4e5f6EXAMPLE"
        }
    ]
}
```

발신 자격 증명 ARN은 자격 증명 유형에 따라 다음 형식 중 하나를 사용합니다.

- 전화번호 - `arn:`*`partition`*`:sms-voice:`*`region`*`:`*`account`*`:phone-number/`*`phoneNumberId`*
- 전화 풀 - `arn:`*`partition`*`:sms-voice:`*`region`*`:`*`account`*`:pool/`*`poolId`*
- 발신자 ID - `arn:`*`partition`*`:sms-voice:`*`region`*`:`*`account`*`:sender-id/`*`senderId`*`/`*`isoCountryCode`*

> **Important**
>
> 정책이 `Resource` 요소의 범위를 발신 자격 증명 ARN으로 지정하는 경우 `OriginationIdentity` 필드를와 같은 동일한 ARN으로 설정합니

다arn:aws:sms-voice:us-east-1:123456789012:phone-number/phone-a1b2c3d4e5f6EXAMPLE. E.164 형식의 전화번호(예: +12065550100)가 범위가 지정된 정책과 일치하지 않으므로 요청이 승인되지 않아 Amazon Cognito가 SMS 메시지를 보낼 수 없습니다.

4. 인도의 수신자에게 SMS 메시지를 보내는 경우 분산 원장 기술(DLT) 포털에 DLT 엔터티 ID와 템플릿 ID를 등록합니다. EumsSms 구성InTemplateId에서 개체 ID를 로InEntityId, 템플릿 ID를 로 제공합니다. 인도 DLT 등록에 대한 자세한 내용은 AWS 최종 사용자 메시징 SMS 사용 설명서의 인도의 수신자에게 SMS 메시지를 보내기 위한 특별 요구 사항을 참조하세요.

5. 구성한 발신 ID 및 IAM 역할을 사용하여 AWS 최종 사용자 메시징 SMS SendTextMessage API 작업을 직접 호출하여 구성을 테스트합니다. 사용자 풀에 구성을 할당하기 전에 테스트 메시지가 전달되었는지 확인합니다.

6. 사용자 풀에 EumsSms 객체를 설정합니다. CreateUserPool, UpdateUserPool 또는 SetUserPoolMfaConfig 요청에서이 구성을 설정할 수 있습니다. CallerArn 파라미터가 필요합니다. 다음 예제에서는 최소 EumsSms 구성을 보여줍니다.

```
"EumsSms": {
    "CallerArn": "arn:aws:iam::123456789012:role/service-role/CognitoEumsSmsRole",
    "ExternalId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
}
```

특정 발신 자격 증명에서 전송하거나 구성 세트를 사용하려면 OriginationIdentity 및 ConfigurationSetName 파라미터를 포함합니다.

AWS 최종 사용자 메시징 SMS 직접 SMS에 대한 IAM 권한

의 IAM 역할에는 cognito-idp.amazonaws.com 서비스 보안 주체가 이를 수임하도록 허용하는 신뢰 정책이 CallerArn 필요합니다. Amazon SNS 경로와 동일한 신뢰 정책 구조를 사용합니다(참조Amazon Cognito가 SMS 메시지를 보내는 데 사용할 수 있는 IAM 역할 준비 AWS 최종 사용자 메시징 SMS). 또한 역할에는 발신 자격 증명 ARN에 sms-voice:SendTextMessage 범위가 지정된 권한을 부여하는 권한 정책이 필요합니다(의 설정 절차 참조사용자 풀에 대한 AWS 최종 사용자 메시징 SMS 직접 SMS를 설정하려면).

SMS 메시지 AWS 리전 용 선택

> **Note**
>
> 이제의 SMS 메시지가에서 관리 AWS 됩니다AWS 최종 사용자 메시징 SMS.

Amazon SNS 경로 리전 구성

일부 리전에서는 Amazon Cognito SMS 메시지에 사용할 Amazon SNS 리소스가 포함된 리전을 선택할 AWS 리전수 있습니다. Amazon Cognito 아시아 태평양(서울) AWS 리전 을 제외하고 Amazon Cognito를 사용할 수 AWS 리전 있는 에서는 사용자 풀을 생성한에서 Amazon SNS 리소스를 사용할 수 있습니다. 리전을 선택할 때 SMS 메시징을 더 빠르고 안정적으로 만들려면 해당 사용자 풀과 동일한 리전에서 Amazon SNS 리소스를 사용합니다.

새 사용자 풀 마법사의 메시지 전송 구성(Configure message delivery) 단계에서 SMS 리소스에 대한 리전을 선택합니다. 또한 기존 사용자 풀의 인증 방법 메뉴의 SMS 아래에서 편집을 선택할 수 있습니다.

시작 시 일부의 경우 AWS 리전 Amazon Cognito는 대체 리전의 Amazon SNS 리소스와 함께 SMS 메시지를 보냈습니다. 선호하는 리전을 설정하려면 사용자 풀에 대한 SmsConfigurationType 객체의 `SnsRegion` 파라미터를 사용합니다. 다음 표의 Amazon Cognito 리전에서 Amazon Cognito 사용자 풀 리소스를 프로그래밍 방식으로 생성하고 `SnsRegion` 파라미터를 제공하지 않으면 사용자 풀이 레거시 Amazon SNS 리전에서 Amazon SNS 리소스와 함께 SMS 메시지를 보낼 수 있습니다.

아시아 태평양(서울)의 Amazon Cognito 사용자 풀 AWS 리전 은 아시아 태평양(도쿄) 리전에서 Amazon SNS 구성을 사용해야 합니다.

Amazon SNS(경유 AWS 최종 사용자 메시징 SMS)는 모든 신규 계정에 대한 지출 할당량을 매월 1.00 USD(USD)로 설정합니다. Amazon Cognito와 함께 사용할 AWS 리전 에서 지출 한도를 늘렸을 수도 있습니다. Amazon SNS SMS 메시지에 AWS 리전 대한를 변경하기 전에 AWS 지원 센터에서 할당량 증가 사례를 열어 새 리전의 한도를 늘리세요. 자세한 내용은 AWS 최종 사용자 메시징 SMS 사용 설명서의 AWS 최종 사용자 메시징 SMS MMS 및 Voice 샌드박스에서 프로덕션으로 이동을 참조하세요.

다음 표의 모든 Amazon Cognito 리전에 대해 해당 SMS 메시지 리전의 AWS 최종 사용자 메시징 SMS 리소스를 사용하여 SMS 메시지를 보낼 수 있습니다.

Amazon Cognito 리전	SMS 메시지 리전
미국 동부(오하이오)	미국 동부(오하이오), 미국 동부(버지니아 북부)

Amazon Cognito 리전	SMS 메시지 리전
미국 동부(버지니아 북부)	미국 동부(버지니아 북부)
미국 서부(캘리포니아 북부)	미국 서부(캘리포니아 북부)
미국 서부(오리건)	미국 서부(오리건)
캐나다(중부)	캐나다(중부), 미국 동부(버지니아 북부)
캐나다 서부(캘거리)	캐나다 서부(캘거리)
멕시코(중부)	멕시코(중부)
유럽(프랑크푸르트)	유럽(프랑크푸르트), 유럽(아일랜드)
유럽(런던)	유럽(런던), 유럽(아일랜드)
유럽(아일랜드)	유럽(아일랜드)
유럽(파리)	유럽(파리)
유럽(스톡홀름)	유럽(스톡홀름)
유럽(밀라노)	유럽(밀라노)
유럽(스페인)	유럽(스페인)
유럽(취리히)	유럽(취리히)
아시아 태평양(말레이시아)	아시아 태평양(싱가포르)
아시아 태평양(태국)	아시아 태평양(뭄바이)
아시아 태평양(뭄바이)	아시아 태평양(뭄바이), 아시아 태평양(싱가포르)
아시아 태평양(하이데라바드)	아시아 태평양(하이데라바드)
아시아 태평양(홍콩)	아시아 태평양(싱가포르)
아시아 태평양(서울)	아시아 태평양(도쿄)

Amazon Cognito 리전	SMS 메시지 리전
아시아 태평양(싱가포르)	아시아 태평양(싱가포르)
아시아 태평양(시드니)	아시아 태평양(시드니)
아시아 태평양(도쿄)	아시아 태평양(도쿄)
아시아 태평양(자카르타)	아시아 태평양(자카르타)
아시아 태평양(오사카)	아시아 태평양(오사카)
아시아 태평양(멜버른)	아시아 태평양(멜버른)
Middle East (Bahrain)	Middle East (Bahrain)
중동(UAE)	중동(UAE)
남아메리카(상파울루)	남아메리카(상파울루)
이스라엘(텔아비브)	이스라엘(텔아비브)
아프리카(케이프타운)	아프리카(케이프타운)

AWS 최종 사용자 메시징 SMS 다이렉트 경로 리전 구성

AWS 최종 사용자 메시징 SMS 는 자체 세트에서 사용할 수 있으며 AWS 리전, Amazon SNS 또는 Amazon Cognito 사용자 풀을 사용할 수 있는 리전과 다를 수 있습니다. 현재 목록은 AWS 최종 사용자 메시징 SMS 사용 설명서의 지원되는 리전 및 국가를 참조하세요.

SmsConfigurationType의 `EumsSms` 객체 `Region` 필드는 Amazon Cognito가를 호출 AWS 리전 하는 를 설정합니다 AWS 최종 사용자 메시징 SMS. 사용자 풀의 리전에서를 사용할 수 있어야 AWS 최종 사용자 메시징 SMS 합니다. Omit `Region` 및 Amazon Cognito는 사용자 풀의 리전 AWS 최종 사용자 메시징 SMS 에서를 사용하거나 사용자 풀의 리전`Region`으로 명시적으로 설정합니다. Amazon Cognito는 다른 모든 값을 거부합니다.

> **Note**
>
> `Region`를 지원되지 않는 값으로 설정하면 , `CreateUserPool` `UpdateUserPool`및 `SetUserPoolMfaConfig` 요청이와 함께 실패합니다`InvalidParameterException`.

> **Note**
>
> 아시아 태평양(태국), 아시아 태평양(홍콩) 및 아시아 태평양(말레이시아)에서는 AWS 최종 사용자 메시징 SMS 직접 경로를 사용할 수 없습니다 AWS 리전. 이러한 리전 중 하나의 사용자 풀에서 SMS 메시지를 보내려면 Amazon SNS 경로를 대신 사용합니다. 자세한 내용은 Amazon SNS 경로 리전 구성 단원을 참조하십시오.

미국 전화 번호로 SMS 메시지를 보내기 위한 발신 자격 증명 얻기

미국 전화 번호로 SMS 문자 메시지를 보내려는 경우 SMS 샌드박스 테스트 환경을 구축하거나 프로덕션 환경을 구축하는 경우 모두 발신 자격 증명을 얻어야 합니다.

미국 이동 통신사는 미국 전화 번호로 메시지를 보낼 때 발신 ID를 요구합니다. 발신 자격 증명이 아직 없는 경우 받아야 합니다. 발신 ID를 얻는 방법을 알아보려면 AWS 최종 사용자 메시징 SMS 사용 설명서에서 전화번호 요청을 참조하세요.

동일한에 둘 이상의 발신 자격 증명이 있는 경우는 단축 코드 AWS 리전, 10DLC, 수신자 부담 전화번호의 우선 순위에 따라 발신 자격 증명 유형을 AWS 최종 사용자 메시징 SMS 선택합니다. 이 우선 순위는 변경할 수 없습니다. 자세한 내용은 AWS 최종 사용자 메시징 SMS FAQ를 참조하세요.

환경이 SMS 샌드박스인지 확인합니다.

환경이 SMS 샌드박스인지 확인하려면 다음 절차를 따르세요. 프로덕션 Amazon Cognito 사용자 풀 AWS 리전 이 있는 각에 대해 반복합니다.

Amazon Cognito 콘솔에서 SMS 샌드박스 상태 검토

환경이 SMS 샌드박스인지 확인하려면

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 AWS 자격 증명을 입력합니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택합니다.

4. 인증 방법 메뉴를 선택합니다.
5. [SMS 구성(SMS configuration)] 섹션에서 [Amazon SNS 프로덕션 환경으로 이동(Move to Amazon SNS production environment)]을 확장합니다. 계정이 SMS 샌드박스에 있는 경우 다음 메시지가 표시됩니다.

   SMS 메시지 설정을 완료하도록 AWS 서비스 종속성 구성

   이 메시지가 표시되지 않으면 누군가 이미 계정에 SMS 메시지를 설정한 것입니다. Amazon Cognito에서 사용자 풀 설정 완료로 이동하세요.
6. Amazon SNS 프로덕션 환경으로 이동에서 Amazon SNS 링크를 선택합니다. 이렇게 하면 Amazon SNS 콘솔이 새 탭에서 열립니다.
7. 샌드박스 환경에 있는지 확인합니다. 콘솔 메시지는 다음과 같이 샌드박스 상태와 AWS 리전를 나타냅니다.

   ```
   This account is in the SMS sandbox in US East (N. Virginia).
   ```

샌드박스에서 계정 이동

프로덕션에서 앱을 사용하려면 계정을 SMS 샌드박스에서 프로덕션으로 이동합니다. Amazon Cognito에서 사용할 AWS 최종 사용자 메시징 SMS 리소스가 포함된에서 AWS 리전 발신 자격 증명을 구성한 후가 SMS 샌드박스에 AWS 계정 있는 동안 미국 전화번호를 확인할 수 있습니다. 환경이 프로덕션 상태일 때는 SMS 메시지를 보내기 전에 사용자 전화번호를 확인할 필요가 없습니다.

AWS 최종 사용자 메시징 SMS 콘솔 또는 Amazon SNS 콘솔에서 샌드박스를 종료하는 요청을 생성할 수 있습니다. 자세한 지침은 AWS 최종 사용자 메시징 SMS 사용 설명서의 SMS 샌드박스에서 이동을 참조하세요.

에서 시뮬레이터 번호 또는 확인된 전화번호 사용 AWS 최종 사용자 메시징 SMS

계정을 SMS 샌드박스 외부로 이동한 경우 이 단계를 건너뜁니다.

샌드박스에 있지만 발신 번호를 설정한 경우 확인된 대상 번호로 메시지를 보낼 수 있습니다. 확인된 대상을 설정하려면 AWS 최종 사용자 메시징 SMS 사용 설명서의 확인된 대상 전화번호 추가를 참조하세요.

시뮬레이션된 발신자 및 대상으로 메시지를 보낼 수도 있습니다. 시뮬레이터 메시지는 로그를 생성하지만 통신 사업자 네트워크를 통해 전송되지는 않습니다. 바로 가기 메뉴에서 SMS 시뮬레이터를 사용한 SMS 전송 테스트를 선택합니다. 자세한 내용은 AWS 최종 사용자 메시징 SMS 사용 설명서의 시뮬레이터 전화번호를 참조하세요.

Amazon Cognito에서 사용자 풀 설정 완료

사용자 풀을 생성 또는 편집하고 있던 브라우저 탭으로 돌아갑니다. 절차를 완료합니다. 사용자 풀에 SMS 구성을 성공적으로 추가하면 Amazon Cognito는 테스트 메시지를 내부 전화번호로 전송하여 구성이 작동하는지 확인합니다.는 각 테스트 SMS 메시지에 대해 AWS 최종 사용자 메시징 SMS 요금을 부과합니다.

사용자 풀 분석에 Amazon Pinpoint 사용

> **Note**
>
> 지원 종료 공지: 2026년 10월 30일에 AWS 는 Amazon Pinpoint에 대한 지원을 종료합니다. 2026년 10월 30일 이후에는 Amazon Pinpoint 콘솔 또는 Amazon Pinpoint 리소스(엔드포인트, 세그먼트, 캠페인, 여정, 분석)에 더 이상 액세스할 수 없습니다. 자세한 내용은 Amazon Pinpoint 지원 종료를 참조하세요. 참고: SMS, 음성, 모바일 푸시, OTP 및 전화번호 검증과 관련된 APIs는이 변경의 영향을 받지 않으며 AWS 최종 사용자 메시징에서 지원됩니다.

Amazon Cognito 사용자 풀은 Amazon Pinpoint와 통합되어 Amazon Cognito 사용자 풀에 대한 분석을 제공하고 Amazon Pinpoint 캠페인을 위한 사용자 데이터를 보강합니다. Amazon Pinpoint는 분석 및 표적화된 캠페인을 제공하여 푸시 알림을 사용하는 모바일 앱에서 사용자 참여를 유도합니다. Amazon Cognito 사용자 풀의 Amazon Pinpoint 분석 지원을 통해 Amazon Pinpoint 콘솔에서 사용자 풀 가입, 로그인, 인증 실패, 일 실사용자(DAU) 및 월 실사용자(MAU)를 추적할 수 있습니다. 디바이스 플랫폼, 디바이스 로캘 및 앱 버전과 같은 다양한 날짜 범위 또는 속성에 대한 데이터를 자세히 확인할 수 있습니다.

앱의 사용자 지정 속성을 설정할 수도 있습니다. 그런 다음 Amazon Pinpoint에서 사용자를 세분화하고 대상 푸시 알림에 전송하는데 이를 사용할 수 있습니다. Amazon Cognito 콘솔의 앱 클라이언트 메뉴에서 앱 클라이언트에 대한 분석 구성에서 Amazon Pinpoint와 사용자 속성 데이터 공유를 선택하면 Amazon Pinpoint에서 사용자 이메일 주소와 전화번호에 대한 추가 엔드포인트를 생성합니다.

Amazon Cognito 콘솔을 사용하여 사용자 풀에서 Amazon Pinpoint 분석을 활성화하면 Amazon Cognito가 사용자 풀을 위해 Amazon Pinpoint에 API 요청을 보낼 때 Amazon Cognito가 맡는 서비스 연결 역할도 생성됩니다. 분석 구성을 추가하는 IAM 보안 주체에게는 CreateServiceLinkedRole 권한이 있어야 합니다. 이 서비스 연결 역할은 AWSServiceRoleForAmazonCognitoIdp입니다. 자세한 내용은 Amazon Cognito에 서비스 연결 역할 사용 단원을 참조하십시오.

Amazon Cognito API에서 앱 클라이언트에 `AnalyticsConfiguration`을 적용하면 Amazon Pinpoint의 사용자 지정 IAM 역할과 이 역할을 맡을 외부 ID를 할당할 수 있습니다. 역할은

cognito-idp 서비스 보안 주체를 신뢰해야 하며, 역할 신뢰 정책에 외부 ID가 필요한 경우, 해당 ID가 AnalyticsConfiguration과 일치해야 합니다. Amazon Pinpoint 프로젝트를 위한 역할 cognito-idp:Describe* 권한 및 다음 권한을 부여해야 합니다.

- mobiletargeting:UpdateEndpoint
- mobiletargeting:PutEvents

Amazon Cognito 및 Amazon Pinpoint 리전 가용성

다음 표에는 다음 조건 중 하나를 충족하는 Amazon Cognito와 Amazon Pinpoint 간의 AWS 리전 매핑이 나와 있습니다.

- Amazon Pinpoint 프로젝트는 미국 동부(버지니아 북부)(us-east-1) 리전에서만 사용할 수 있습니다.
- Amazon Pinpoint 프로젝트는 동일한 리전 또는 미국 동부(버지니아 북부)(us-east-1) 리전에서만 사용할 수 있습니다.

기본적으로 Amazon Cognito는 동일한 AWS 리전에 있는 Amazon Pinpoint 프로젝트에만 분석을 전송할 수 있습니다. 다음 표에 나오는 지역과 Amazon Pinpoint를 사용할 수 없는 지역에는 이 규칙이 적용되지 않습니다.

다음 리전에서 Amazon Pinpoint를 사용할 수 없습니다. 이러한 리전에 있는 Amazon Cognito 사용자 풀은 분석을 지원하지 않습니다.

- 유럽(밀라노)
- Middle East (Bahrain)
- 아시아 태평양(오사카)
- 이스라엘(텔아비브)
- 아프리카(케이프타운)
- 아시아 태평양(자카르타)
- 아시아 태평양(말레이시아)

이 표에서는 Amazon Cognito 사용자 풀을 구축한 리전과 Amazon Pinpoint에 있는 대응하는 리전 간의 관계를 확인할 수 있습니다. Amazon Pinpoint 프로젝트를 Amazon Cognito와 통합하려면 사용 가능한 리전에서 Amazon Pinpoint 프로젝트를 구성해야 합니다.

Amazon Cognito 사용자 풀 리전	Amazon Pinpoint 프로젝트용 리전
ap-northeast-1	us-east-1
ap-northeast-2	us-east-1
ap-south-1	us-east-1, ap-south-1
ap-southeast-1	us-east-1
ap-southeast-2	us-east-1, ap-southeast-2
ca-central-1	us-east-1
eu-central-1	us-east-1, eu-central-1
eu-west-1	us-east-1, eu-west-1
eu-west-2	us-east-1
us-east-1	us-east-1
us-east-2	us-east-1
us-west-2	us-east-1, us-west-2

리전 매핑 예

- ap-northeast-1에 사용자 풀을 생성하는 경우 us-east-1에 Amazon Pinpoint 프로젝트를 생성할 수 있습니다.
- ap-south-1에 사용자 풀을 생성하는 경우 us-east-1 또는 ap-south-1에 Amazon Pinpoint 프로젝트를 생성할 수 있습니다.

> **Note**
>
> 이전 표의 프로젝트를 AWS 리전 제외한 모든 항목에 대해 Amazon Cognito는 사용자 풀과 동일한 리전의 Amazon Pinpoint 프로젝트만 사용할 수 있습니다. 사용자 풀을 구축한 리전에서 Amazon Pinpoint를 사용할 수 없다면, Amazon Cognito는 해당 리전에서 Amazon Pinpoint 분

석을 지원하지 않습니다. 자세한 AWS 리전 정보는 Amazon Pinpoint 엔드포인트 및 할당량을 참조하세요.

Amazon Pinpoint 분석 설정 지정(AWS Management Console)

분석 데이터를 Amazon Pinpoint로 보내도록 Amazon Cognito 사용자 풀을 구성할 수 있습니다. Amazon Cognito는 로컬 사용자에 대한 분석 데이터만 Amazon Pinpoint로 전송합니다. Amazon Pinpoint 프로젝트와 연결되도록 사용자 풀을 구성한 후에는 `AnalyticsMetadata`를 API 요청에 포함해야 합니다. 자세한 내용은 앱을 Amazon Pinpoint와 통합 단원을 참조하십시오.

분석 설정을 지정하려면

1. Amazon Cognito 콘솔로 이동합니다. AWS 자격 증명을 입력하라는 메시지가 나타날 수 있습니다.
2. User Pools(사용자 풀)을 선택하고 목록에서 기존 사용자 풀을 선택합니다.
3. 앱 클라이언트 메뉴를 선택하고 업데이트하려는 앱 클라이언트를 선택합니다.
4. 분석 탭의 Pinpoint 분석에서 활성화를 선택합니다.
5. Pinpoint Region(Pinpoint 리전)을 선택합니다.
6. Amazon Pinpoint project(Amazon Pinpoint 프로젝트)를 선택하거나 Create Amazon Pinpoint project(Amazon Pinpoint 프로젝트 생성)를 선택합니다.

   > **Note**
   >
   > Amazon Pinpoint 프로젝트 ID는 Amazon Pinpoint 프로젝트에 대해 고유한 32자 문자열입니다. 이 ID는 Amazon Pinpoint 콘솔에 나열됩니다.
   > Amazon Pinpoint 프로젝트에 여러 Amazon Cognito 앱을 매핑할 수 있습니다. 그러나 각 Amazon Cognito 앱은 하나의 Amazon Pinpoint 프로젝트에만 매핑될 수 있습니다.
   > Amazon Pinpoint에서 각 프로젝트는 단일 앱이어야 합니다. 예를 들어, 게임 개발자에게 두 개의 게임이 있고 두 게임 모두 동일한 Amazon Cognito 사용자 풀을 사용한다 하더라도 각 게임은 개별 Amazon Pinpoint 프로젝트입니다. Amazon Pinpoint 프로젝트에 대한 자세한 내용은 Amazon Pinpoint에서 프로젝트 생성을 참조하세요.

7. Amazon Cognito에서 이메일 주소와 전화 번호를 Amazon Pinpoint로 보내 사용자를 위한 추가 엔드포인트를 생성하게 하고 싶다면, User data sharing(사용자 데이터 공유)에서 Share user data with Amazon Pinpoint(Amazon Pinpoint와 사용자 데이터 공유)를 선택합니다. 사용자가 자신의 이메일 주소와 전화 번호를 확인하면, Amazon Cognito는 사용자 계정에 사용할 수 있는 경우에만 이를 Amazon Pinpoint와 공유합니다.

> **Note**
>
> 엔드포인트는 Amazon Pinpoint를 사용하여 푸시 알림이 전송될 수 있는 사용자 디바이스를 고유하게 식별합니다. 엔드포인트에 대한 자세한 내용은 Amazon Pinpoint 개발자 가이드에서 엔드포인트 추가를 참조하세요.

8. 변경 사항 저장을 선택합니다.

Amazon Pinpoint 분석 설정(AWS CLI 및 AWS API) 지정

다음 명령을 사용하여 사용자 풀에 대한 Amazon Pinpoint 분석 설정을 지정합니다.

앱 생성 시간에 사용자 풀의 기존 클라이언트 앱에 대한 분석 설정을 지정하는 방법

- AWS CLI: `aws cognito-idp create-user-pool-client`
- AWS API: CreateUserPoolClient

사용자 풀의 기존 클라이언트 앱에 대한 분석 설정을 업데이트하는 방법

- AWS CLI: `aws cognito-idp update-user-pool-client`
- AWS API: UpdateUserPoolClient

> **Note**
>
> Amazon Cognito는 `ApplicationArn`을 사용하는 경우 리전 내 통합을 지원합니다.

앱을 Amazon Pinpoint와 통합

사용자 풀 API에서 Amazon Cognito 로컬 사용자를 위한 분석 메타데이터를 Amazon Pinpoint에 게시할 수 있습니다.

로컬 사용자

계정에 가입했거나 서드 파티 ID 제공업체(idP)를 통해 로그인하는 대신 사용자 풀에서 생성된 사용자입니다.

사용자 풀 API

> 사용자 지정 사용자 인터페이스(UI)가 있는 앱을 사용하여 AWS SDK와 통합할 수 있는 작업입니다. 관리형 로그인을 통해 로그인하는 페더레이션 또는 로컬 사용자에 대한 분석 메타데이터를 전달할 수 없습니다. 사용자 풀 API 작업 목록은 Amazon Cognito API 참조에서 확인하세요.

캠페인에 게시하도록 사용자 풀을 구성한 후 Amazon Cognito는 다음 API 작업을 위해 메타데이터를 Amazon Pinpoint에 전달합니다.

- `AdminInitiateAuth`
- `AdminRespondToAuthChallenge`
- `ConfirmForgotPassword`
- `ConfirmSignUp`
- `ForgotPassword`
- `InitiateAuth`
- `ResendConfirmationCode`
- `RespondToAuthChallenge`
- `SignUp`

사용자 세션에 대한 메타데이터를 Amazon Pinpoint 캠페인에 전달하려면 API 요청의 `AnalyticsMetadata` 파라미터에 `AnalyticsEndpointId` 값을 포함합니다. JavaScript 예제는 AWS 지식 센터에서 Amazon Cognito 사용자 풀 분석이 Amazon Pinpoint 대시보드에 나타나지 않는 이유는 무엇입니까?를 참조하세요.

위협 방지를 통한 사용자 풀 보호

네트워크 침입, 암호 추측, 사용자 사칭, 악의적인 가입 및 로그인으로부터 애플리케이션을 보호해야 할 수도 있습니다. Amazon Cognito 사용자 풀 보안 기능의 구성은 보안 아키텍처의 주요 구성 요소가 될 수 있습니다. 애플리케이션의 보안은 AWS 공동 책임 모델에 설명된 대로 고객 책임 "클라우드의 보안"입니다. 이 장의 도구는 애플리케이션 보안 설계가 이러한 목표에 부합하는 데 도움이 됩니다.

공동 책임 모델, 데이터 보호, 규정 준수 및 인프라 보안과 같은 계정 수준 및 서비스 전반의 보안 지침은 섹션을 참조하세요Amazon Cognito의 보안.

사용자 풀을 구성할 때 결정해야 하는 중요한 사항은 공개 가입 및 로그인을 허용할지 여부입니다. 기밀 클라이언트, 사용자의 관리 생성 및 확인, 도메인이 없는 사용자 풀과 같은 일부 사용자 풀 옵션은 인터넷을 통한 공격의 영향을 덜 받습니다. 그러나 일반적인 사용 사례는 인터넷에 있는 모든 사람의 가입을 수락하고 모든 작업을 사용자 풀로 직접 보내는 퍼블릭 클라이언트입니다. 모든 구성에서, 특히 이러한 퍼블릭 구성의 경우 보안 기능을 염두에 두고 사용자 풀을 계획하고 배포하는 것이 좋습니다. 보안이 충분하지 않으면 원치 않는 소스가 새 활성 사용자를 생성하거나 기존 사용자를 악용하려고 할 때 AWS 청구서에 영향을 미칠 수도 있습니다.

MFA 및 위협 보호는 로컬 사용자에게 적용됩니다. 타사 IdP는 페더레이션 사용자의 보안 태세를 담당합니다.

사용자 풀의 보안 기능

다중 인증(MFA)

사용자 풀이 이메일(Essentials 또는 Plus 기능 플랜 사용) 또는 SMS 메시지 또는 인증 앱에서 보내는 코드를 요청하여 사용자 풀 로그인을 확인합니다.

위협 방지

로그인에서 위험 지표를 모니터링하고 MFA 또는 블록 로그인을 적용합니다. 사용자 지정 클레임 및 범위를 액세스 토큰에 추가합니다. 이메일로 MFA 코드를 전송합니다.

AWS WAF 웹 ACLs

네트워크 및 애플리케이션 레이어에서 원치 않는 활동이 있는지 사용자 풀 엔드포인트 및 인증 API로 들어오는 트래픽을 검사합니다.

대소문자 구분

문자 대소문자를 제외하고 이메일 주소 또는 기본 사용자 이름이 다른 사용자와 동일한 사용자를 생성하지 않도록 합니다.

삭제 방지

자동화된 시스템이 실수로 사용자 풀을 삭제하는 것을 방지하세요. AWS Management Console에서 사용자 풀 삭제를 추가로 확인해야 합니다.

사용자 존재 오류

사용자 풀에서 기존 사용자 이름과 별칭이 공개되지 않도록 보호합니다. 사용자 이름이 유효한지 여부에 관계없이 인증 실패에 대한 응답으로 일반 오류를 반환합니다.

주제

- Amazon Cognito 사용자 풀을 위한 보안 모범 사례
- 사용자 풀에 MFA 추가
- ACR 및 AMR 클레임을 사용한 인증 수준
- 사용자 풀의 사용자 디바이스 작업
- 위협 방지 기능이 있는 고급 보안
- AWS WAF 웹 ACL을 사용자 풀과 연결
- 사용자 풀 대/소문자 구분
- 사용자 풀 삭제 방지
- 사용자 존재 오류 응답 관리

Amazon Cognito 사용자 풀을 위한 보안 모범 사례

이 페이지에서는 일반적인 위협으로부터 보호하려는 경우 구현할 수 있는 보안 모범 사례를 설명합니다. 선택하는 구성은 각 애플리케이션의 사용 사례에 따라 다릅니다. 최소한 관리 작업에 최소 권한을 적용하고 애플리케이션 및 사용자 보안 암호를 보호하기 위한 조치를 취하는 것이 좋습니다. 수행할 수 있는 또 다른 고급이지만 효과적인 단계는 AWS WAF 웹 ACLs을 구성하고 사용자 풀에 적용하는 것입니다.

네트워크 수준에서 사용자 풀 보호

AWS WAF 웹 ACLs Amazon Cognito로 빌드하는 인증 메커니즘의 성능과 비용을 보호할 수 있습니다. 웹 ACL 사용하면 API 및 관리형 로그인 요청 앞에 가드레일을 구현할 수 있습니다. 웹 ACL 사용자가 고안한 규칙에 따라 트래픽을 삭제하거나 CAPTCHA를 요구할 수 있는 네트워크 및 애플리케이션 계층 필터를 생성합니다. 요청은 웹 ACL 규칙의 조건을 충족할 때까지 Amazon Cognito 리소스로 전달되지 않습니다. 자세한 내용은 AWS WAF 웹 ACL 섹션을 참조하세요.

SMS 메시지 침해로부터 보호

사용자 풀에서 퍼블릭 가입을 허용하는 경우 Amazon Cognito가 SMS 문자 메시지로 보내는 코드를 사용하여 계정 확인을 구성할 수 있습니다. SMS 메시지는 원치 않는 활동과 연결되어 AWS 청구서를 늘릴 수 있습니다. 사기 상황에서 SMS 메시지를 전송하지 않도록 인프라를 구성합니다. 자세한 내용은 AWS 블로그의 다음 게시물을 검토하세요.

- Amazon Cognito 사용자 풀을 사용하여 사용자 가입 사기 및 SMS 펌프의 위험 감소
- SMS Pumping 방지: 인공적으로 팽창된 트래픽에 대처하는 데 도움이 되는 새로운 AWS 기능

퍼블릭 인증 이해

Amazon Cognito 사용자 풀에는 일반 대중의 구성원이 사용자 계정에 가입하고 애플리케이션에 액세스할 수 있는 사용 사례를 지원하는 고객 ID 및 액세스 관리(CIAM) 기능이 있습니다. 사용자 풀이 셀프 서비스 가입을 허용하면 퍼블릭 인터넷에서 사용자 계정에 대한 요청을 받을 수 있습니다. 셀프 서비스 요청은 SignUp 및 InitiateAuth와 같은 API 작업과 관리형 로그인과의 사용자 상호 작용에서 발생합니다. 퍼블릭 요청에서 발생할 수 있는 침해를 완화하거나 퍼블릭 인증 작업을 완전히 비활성화하도록 사용자 풀을 구성할 수 있습니다.

다음 설정은 사용자 풀 및 앱 클라이언트에서 퍼블릭 및 내부 인증 요청을 관리할 수 있는 몇 가지 방법입니다.

퍼블릭 사용자 풀 액세스에 영향을 미치는 사용자 풀 설정의 예

설정	사용 가능한 옵션	구성 대상	퍼블릭 인증에 미치는 영향	콘솔 설정	API 작업 및 파라미터
셀프 서비스 가입	사용자가 계정에 가입하거나 사용자 계정을 관리자로 생성할 수 있도록 허용합니다.	사용자 풀	퍼블릭 가입 방지	가입 - 셀프 서비스 가입	CreateUserPool, UpdateUserPool AdminCreateUserConfig – AllowAdmi

설정	사용 가능한 옵션	구성 대상	퍼블릭 인증에 미치는 영향	콘솔 설정	API 작업 및 파라미터
					nCreateUserOnly
관리자 확인	새 사용자에게 확인 코드를 보내거나 관리자에게 확인하도록 요청합니다.	사용자 풀	관리자 작업 없는 가입 확인 방지	가입 - Cognito 지원 확인 및 확인	CreateUserPool, UpdateUserPool AccountRecoverySettings_admin_only
사용자 공개	로그인 및 암호 재설정 시 "사용자를 찾을 수 없음" 메시지를 전송하거나 공개를 방지합니다.	앱 클라이언트	로그인 이름, 이메일 주소 또는 전화 번호 추측 방지	앱 클라이언트 - 사용자 존재 오류 방지	CreateUserPoolClient, UpdateUserPoolClient PreventUserExistenceErrors
클라이언트 보안 암호 (client secret)	가입, 로그인, 암호 재설정 시 보안 암호 해시 필요 또는 필요 없음	앱 클라이언트	승인되지 않은 소스의 인증 요청으로부터 보호	앱 클라이언트 - 클라이언트 보안 암호	CreateUserPoolClient GenerateSecret

설정	사용 가능한 옵션	구성 대상	퍼블릭 인증에 미치는 영향	콘솔 설정	API 작업 및 파라미터
웹 ACLs	인증 요청에 대해 네트워크 방화벽 활성화 또는 비활성화	사용자 풀	관리자 정의 요청 특성 및 IP 주소 규칙에 따라 액세스 제한 또는 방지	AWS WAF - WAF 설정	AssociateWebACL `ResourceArn`
외부 ID 제공업체 (IdP)	타사 IdP, 사용자 풀 디렉터리 또는 둘 다에서 사용자의 로그인 허용	앱 클라이언트	가입 및 로그인에서 로컬 사용자 또는 페더레이션 사용자를 제외합니다.	앱 클라이언트 - ID 제공업체	CreateUserPoolClient, UpdateUserPoolClient `SupportedIdentityProviders`
권한 부여 서버	인증을 위해 퍼블릭 웹 페이지를 호스팅하거나 호스팅하지 않음	사용자 풀	퍼블릭 웹 페이지 비활성화 및 SDK 기반 인증만 허용	도메인	CreateUserPoolDomain 사용자 풀 도메인을 생성하면 퍼블릭 웹 페이지를 사용할 수 있습니다.

설정	사용 가능한 옵션	구성 대상	퍼블릭 인증에 미치는 영향	콘솔 설정	API 작업 및 파라미터
위협 방지	악의적인 활동 또는 안전하지 않은 암호의 징후에 대한 모니터링 활성화 또는 비활성화	사용자 풀 또는 앱 클라이언트	사용자가 손상 지표를 표시할 때 자동으로 로그인을 차단하거나 MFA를 요구할 수 있음	위협 방지 - 보호 설정	SetRiskConfiguration SetRiskConfiguration 의 파라미터는 위협 방지 설정을 정의합니다.

클라이언트 보안 암호로 기밀 클라이언트 보호

클라이언트 보안 암호는 앱 클라이언트와 연결된 선택적 문자열입니다. 클라이언트 보안 암호가 있는 앱 클라이언트에 대한 모든 인증 요청에는 사용자 이름, 클라이언트 ID 및 클라이언트 보안 암호에서 생성된 보안 암호 해시가 포함되어야 합니다. 클라이언트 보안 암호를 모르는 사용자는 처음부터 애플리케이션을 종료합니다.

그러나 클라이언트 보안 암호에는 제한이 있습니다. 클라이언트 보안 암호를 퍼블릭 클라이언트 소프트웨어에 임베드하면 클라이언트 보안 암호가 검사를 받을 수 있습니다. 그러면 사용자를 생성하고, 암호 재설정 요청을 제출하고, 앱 클라이언트에서 다른 작업을 수행할 수 있는 기능이 열립니다. 클라이언트 보안 암호는 애플리케이션이 보안 암호에 액세스할 수 있는 유일한 엔터티인 경우에만 구현해야 합니다. 일반적으로 이는 서버 측 기밀 클라이언트 애플리케이션에서 가능합니다. 이는 클라이언트 보안 암호가 필요한 M2M 애플리케이션에서도 마찬가지입니다. 클라이언트 보안 암호를 암호화된 로컬 스토리지 또는에 저장합니다 AWS Secrets Manager. 클라이언트 보안 암호가 퍼블릭 인터넷에 표시되지 않도록 합니다.

기타 보안 암호 보호

Amazon Cognito 사용자 풀을 사용하는 인증 시스템은 프라이빗 데이터, 암호 및 AWS 자격 증명을 처리할 수 있습니다. 다음은 애플리케이션이 액세스할 수 있는 보안 암호를 처리하는 몇 가지 모범 사례입니다.

암호

사용자는 애플리케이션에 로그인할 때 암호를 입력할 수 있습니다. Amazon Cognito에는 애플리케이션이 새 암호 프롬프트 없이 만료된 사용자 세션을 계속하기 위해 사용할 수 있는 새로 고침 토큰이 있습니다. 로컬 스토리지에 암호나 암호 해시를 두지 마세요. 암호를 불투명하게 취급하고 사용자 풀로만 전달하도록 애플리케이션을 설계합니다.

WebAuthn 패스키를 사용하여 암호 없는 인증을 구현하는 것이 가장 좋습니다. 암호를 구현해야 하는 경우 보안 원격 암호(SRP) 인증 흐름 및 다중 인증(MFA)을 사용합니다.

AWS 자격 증명

관리 인증 및 사용자 풀 관리 작업에는 AWS 자격 증명을 사용한 인증이 필요합니다. 애플리케이션에서 이러한 작업을 구현하려면 임시 AWS 자격 증명에 대한 보안 액세스 권한을 부여합니다. 제어하는 서버 구성 요소에서 실행되는 애플리케이션에만 자격 증명 액세스 권한을 부여합니다. 자격 AWS 증명이 있는 애플리케이션을 GitHub와 같은 퍼블릭 버전 제어 시스템에 두지 마세요. 퍼블릭 클라이언트 측 애플리케이션에서 AWS 자격 증명을 인코딩하지 마십시오.

PKCE 코드 확인자

코드 교환용 증명 키(PKCE)는 사용자 풀 권한 부여 서버에 OpenID Connect(OIDC) 권한 부여 코드를 부여하기 위한 것입니다. 애플리케이션은 권한 부여 코드를 요청할 때 사용자 풀과 코드 확인자 보안 암호를 공유합니다. 권한 부여 코드를 토큰으로 교환하려면 클라이언트가 코드 확인자를 알고 있음을 다시 확인해야 합니다. 이 방법은 가로채는 권한 부여 코드로 토큰을 발급하지 않도록 보호합니다.

클라이언트는 각 권한 부여 요청과 함께 새 무작위 코드 확인자를 생성해야 합니다. 정적 또는 예측 가능한 코드 확인자를 사용하면 공격자가 하드 코딩된 확인자와 권한 부여 코드를 가로채기만 하면 됩니다. 사용자에게 코드 확인자 값이 노출되지 않도록 애플리케이션을 설계합니다.

사용자 풀 관리 최소 권한

IAM 정책은 보안 주체가 Amazon Cognito 사용자 풀 관리 및 관리 인증 작업에 대해 갖는 액세스 수준을 정의할 수 있습니다. 예제:

- 웹 서버에 관리 API 작업을 통한 인증 권한을 부여합니다.
- 에서 사용자 풀을 관리하는 AWS IAM Identity Center 사용자에게 사용자 풀 유지 관리 및 보고 권한을 AWS 계정부여합니다.

Amazon Cognito의 리소스 세분화 수준은 IAM 정책 목적상 사용자 풀과 ID 풀이라는 두 가지 리소스 유형으로 제한됩니다. 개별 앱 클라이언트를 관리하는 권한은 적용할 수 없습니다. 부여하는 권한이 모든 앱 클라이언트에 효과적이라는 지식으로 사용자 풀을 구성합니다. 조직에 여러 애플리케이션 테넌트가 있고 보안 모델에서 테넌트 간에 관리 책임을 분리해야 하는 경우 사용자 풀당 하나의 테넌트로 다중 테넌트를 구현합니다.

`InitiateAuth`와 같은 사용자 인증 작업에 대한 권한이 있는 IAM 정책을 생성할 수 있지만 이러한 권한은 영향을 미치지 않습니다. 퍼블릭 및 토큰 인증 API 작업에는 IAM 권한이 적용되지 않습니다. 사용 가능한 사용자 풀 인증 작업 중 `AdminInitiateAuth`와 같은 관리 서버 측 작업에만 권한을 부여할 수 있습니다.

최소 권한 `Action` 목록으로 사용자 풀 관리 수준을 제한할 수 있습니다. 다음 예제 정책은 IdP, 리소스 서버, 앱 클라이언트 및 사용자 풀 도메인을 관리할 수 있지만 사용자 또는 사용자 풀은 관리할 수 없는 관리자를 위한 것입니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "UserPoolClientAdministrator",
      "Action": [
        "cognito-idp:CreateIdentityProvider",
        "cognito-idp:CreateManagedLoginBranding",
        "cognito-idp:CreateResourceServer",
        "cognito-idp:CreateUserPoolDomain",
        "cognito-idp:DeleteIdentityProvider",
        "cognito-idp:DeleteResourceServer",
        "cognito-idp:DeleteUserPoolDomain",
        "cognito-idp:DescribeIdentityProvider",
        "cognito-idp:DescribeManagedLoginBranding",
        "cognito-idp:DescribeManagedLoginBrandingByClient",
        "cognito-idp:DescribeResourceServer",
        "cognito-idp:DescribeUserPool",
        "cognito-idp:DescribeUserPoolClient",
        "cognito-idp:DescribeUserPoolDomain",
        "cognito-idp:GetIdentityProviderByIdentifier",
        "cognito-idp:GetUICustomization",
        "cognito-idp:ListIdentityProviders",
        "cognito-idp:ListResourceServers",
```

```
        "cognito-idp:ListUserPoolClients",
        "cognito-idp:ListUserPools",
        "cognito-idp:SetUICustomization",
        "cognito-idp:UpdateIdentityProvider",
        "cognito-idp:UpdateManagedLoginBranding",
        "cognito-idp:UpdateResourceServer",
        "cognito-idp:UpdateUserPoolClient",
        "cognito-idp:UpdateUserPoolDomain"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:cognito-idp:us-west-2:123456789012:userpool/us-
west-2_EXAMPLE"
    }
  ]
}
```

다음 예제 정책은 서버 측 애플리케이션에 사용자 및 그룹 관리 및 인증을 부여합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "UserAdminAuthN",
      "Action": [
        "cognito-idp:AdminAddUserToGroup",
        "cognito-idp:AdminConfirmSignUp",
        "cognito-idp:AdminCreateUser",
        "cognito-idp:AdminDeleteUser",
        "cognito-idp:AdminDeleteUserAttributes",
        "cognito-idp:AdminDisableProviderForUser",
        "cognito-idp:AdminDisableUser",
        "cognito-idp:AdminEnableUser",
        "cognito-idp:AdminForgetDevice",
        "cognito-idp:AdminGetDevice",
        "cognito-idp:AdminGetUser",
        "cognito-idp:AdminInitiateAuth",
        "cognito-idp:AdminLinkProviderForUser",
        "cognito-idp:AdminListDevices",
        "cognito-idp:AdminListGroupsForUser",
        "cognito-idp:AdminListUserAuthEvents",
```

```
        "cognito-idp:AdminRemoveUserFromGroup",
        "cognito-idp:AdminResetUserPassword",
        "cognito-idp:AdminRespondToAuthChallenge",
        "cognito-idp:AdminSetUserMFAPreference",
        "cognito-idp:AdminSetUserPassword",
        "cognito-idp:AdminSetUserSettings",
        "cognito-idp:AdminUpdateAuthEventFeedback",
        "cognito-idp:AdminUpdateDeviceStatus",
        "cognito-idp:AdminUpdateUserAttributes",
        "cognito-idp:AdminUserGlobalSignOut",
        "cognito-idp:AssociateSoftwareToken",
        "cognito-idp:ListGroups",
        "cognito-idp:ListUsers",
        "cognito-idp:ListUsersInGroup",
        "cognito-idp:RevokeToken",
        "cognito-idp:UpdateGroup",
        "cognito-idp:VerifySoftwareToken"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:cognito-idp:us-west-2:123456789012:userpool/us-
west-2_EXAMPLE"
    }
  ]
}
```

토큰 보안 및 확인

토큰에는 최종 사용자에게 공개하지 않을 수 있는 그룹 멤버십 및 사용자 속성에 대한 내부 참조가 포함될 수 있습니다. 로컬 스토리지에 ID 및 액세스 토큰을 저장하지 마세요. 새로 고침 토큰은 사용자 풀만 액세스할 수 있는 키로 암호화되며 사용자와 애플리케이션에서는 볼 수 없습니다. 사용자가 로그아웃하거나 보안상의 이유로 사용자 세션 지속이 바람직하지 않다고 판단되면 새로 고침 토큰을 취소합니다.

액세스 토큰을 사용하여 토큰이 유효하고 만료되지 않았는지 독립적으로 확인하는 시스템에 대한 액세스 권한만 부여합니다. 확인 리소스는 JSON 웹 토큰 확인을 참조하세요.

신뢰할 수 있는 ID 제공업체 확인

SAML 또는 OIDC ID 제공업체(idP)로 사용자 풀을 구성하면 IdP 새 사용자를 생성하고, 사용자 속성을 설정하고, 애플리케이션 리소스에 액세스할 수 있습니다. SAML 및 OIDC 제공업체는 일반적으로 사용

자 또는 즉각적인 고객이 제공업체의 멤버십 및 구성을 제어하는 B2B(기업 간) 또는 엔터프라이즈 시나리오에서 사용됩니다.

소셜 제공업체는 인터넷에 있는 모든 사용자에게 사용자 계정을 제공하며 엔터프라이즈 제공업체보다 사용자의 제어 권한이 낮습니다. 퍼블릭 고객이 애플리케이션의 리소스에 로그인하고 액세스할 수 있도록 허용할 준비가 된 경우에만 앱 클라이언트에서 소셜 IdP를 활성화합니다.

범위가 사용자 프로필에 대한 액세스에 미치는 영향 이해

인증 요청의 액세스 제어 범위를 사용자 풀 권한 부여 서버에 요청할 수 있습니다. 이러한 범위는 사용자에게 외부 리소스에 대한 액세스 권한을 부여하고 사용자에게 자신의 사용자 프로필을 보고 수정할 수 있는 액세스 권한을 부여할 수 있습니다. 애플리케이션 작동에 필요한 최소 범위를 지원하도록 앱 클라이언트를 구성합니다.

`aws.cognito.signin.user.admin` 범위는 InitiateAuth와 같은 작업을 통해 SDK 인증에서 발급한 모든 액세스 토큰에 있습니다. 애플리케이션의 사용자 프로필 셀프 서비스 작업을 위해 지정됩니다. 권한 부여 서버에서 이 범위를 요청할 수도 있습니다. 이 범위는 UpdateUserAttributes 및 GetUser와 같은 토큰 인증 작업에 필요합니다. 이러한 작업의 효과는 앱 클라이언트의 읽기 및 쓰기 권한에 의해 제한됩니다.

`openid`, `profile`, `email`, phone 범위는 사용자 풀 권한 부여 서버의 userInfo 엔드포인트에 대한 요청을 승인합니다. 엔드포인트가 반환할 수 있는 속성을 정의합니다. 다른 범위 없이 요청하면 `openid` 범위가 사용 가능한 모든 속성을 반환하지만 요청에서 더 많은 범위를 요청하면 응답이 추가 범위로 표시되는 속성으로 좁아집니다. `openid` 범위는 ID 토큰에 대한 요청도 나타냅니다. 요청에서 권한 부여 엔드포인트로 이 범위를 생략하면 Amazon Cognito는 액세스 토큰과 해당하는 경우 새로 고침 토큰만 발급합니다. 자세한 내용은 앱 클라이언트 용어의 OpenID Connect 범위를 참조하세요.

사용자 속성에 대한 입력 삭제

예를 들어 `email`과 같이 전송 방법 및 사용자 이름으로 사용될 수 있는 사용자 속성에는 형식 제한이 있습니다. 다른 속성에는 문자열, 부울 또는 숫자 데이터 형식이 있을 수 있습니다. 문자열 속성 값은 다양한 입력을 지원합니다. Amazon Cognito가 사용자에게 전달하는 메시지 또는 사용자 디렉터리에 원치 않는 데이터를 쓰려는 시도로부터 보호하도록 애플리케이션을 구성합니다. Amazon Cognito에 제출하기 전에 애플리케이션에서 사용자가 제출한 문자열 속성 값의 클라이언트 측 검증을 수행합니다.

사용자 풀은 사용자가 지정한 속성 매핑을 기반으로 IdP의 속성을 사용자 풀에 매핑합니다. 안전하고 예측 가능한 IdP 속성만 사용자 풀 문자열 속성에 매핑합니다.

사용자 풀에 MFA 추가

MFA는 일반적으로 사용자 이름과 암호인 초기 사용자가 알고 있는 것 인증 요소에 사용자가 가지고 있는 것 인증 요소를 추가합니다. 암호를 기본 인증 요소로 사용하는 사용자를 로그인시키기 위해 SMS 문자 메시지, 이메일 메시지 또는 시간 기반 일회용 암호(TOTP)를 추가 요소로 선택할 수 있습니다.

다중 인증(MFA)은 애플리케이션의 로컬 사용자의 보안을 강화합니다. 페더레이션 사용자의 경우 Amazon Cognito는 모든 인증 프로세스를 IdP에 위임하며 추가 인증 요소를 제공하지 않습니다.

> **Note**
>
> 새 사용자가 앱에 처음 로그인할 때 Amazon Cognito는 사용자 풀에 MFA가 필요한 경우에도 OAuth 2.0 토큰을 발급합니다. 사용자가 처음 로그인할 때 두 번째 인증 요소는 Amazon Cognito가 사용자에게 보내는 확인 메시지를 확인하는 것입니다. 사용자 풀에 MFA가 필요한 경우 Amazon Cognito는 첫 로그인 후 각 로그인 시도 중에 사용할 추가 로그인 요소를 등록하라는 메시지를 표시합니다.

적응형 인증을 통해 위험 수준이 높아졌을 때 추가 인증 요소를 요구하도록 사용자 풀을 구성할 수 있습니다. 사용자 풀에 조정 인증을 추가하는 방법은 위협 방지 기능이 있는 고급 보안 섹션을 참조하세요.

MFA를 사용자 풀에 대해 `required`로 설정하면 모든 사용자가 로그인하려면 MFA를 완료해야 합니다. 로그인하려면 각 사용자가 하나 이상의 MFA 요소를 설정해야 합니다. MFA가 필요한 경우 사용자 풀에서 사용자의 로그인을 허용하도록 사용자 온보딩에 MFA 설정을 포함해야 합니다.

MFA를 필수로 설정하면 관리형 로그인에 MFA를 설정하라는 메시지가 표시됩니다. 사용자 풀에서 MFA를 선택 사항으로 설정하면 관리형 로그인에서 사용자에게 메시지를 표시하지 않습니다. 선택적 MFA를 사용하려면 사용자에게 MFA를 설정할 것인지 선택하라는 메시지를 표시한 다음 API 입력을 통해 추가 로그인 요소를 검증하도록 안내하는 인터페이스를 앱에 구축해야 합니다.

주제

- 사용자 풀 MFA에 대해 알아야 할 사항
- 사용자 MFA 기본 설정
- 사용자 런타임 시 MFA 로직에 대한 세부 정보
- 다중 인증을 위한 사용자 풀 구성
- SMS 및 이메일 메시지 MFA
- TOTP 소프트웨어 토큰 MFA

사용자 풀 MFA에 대해 알아야 할 사항

MFA를 설정하기 전에 다음을 고려하세요.

- 사용자는 MFA를 사용하거나 암호 없는 요소로 로그인할 수 있습니다. 단, 사용자 풀`MULTI_FACTOR_WITH_USER_VERIFICATION`에서 `FactorConfiguration`를 로 설정하면 사용자 확인이 포함된 패스키가 MFA 요구 사항을 첫 번째 요소로 충족할 수 있습니다`WebAuthnConfiguration`. 패스키는 암호 로그인의 두 번째 요소로 사용할 수 없습니다.
  - 일회용 암호를 지원하는 사용자 풀에서는 MFA를 필수로 설정할 수 없습니다.
  - 사용자 풀에 MFA가 필요한 `AllowedFirstAuthFactors` 경우 `EMAIL_OTP` 또는 `SMS_OTP`를에 추가할 수 없습니다. `FactorConfiguration`가 로 설정된 `WEB_AUTHN` 경우를 추가할 수 있습니다`MULTI_FACTOR_WITH_USER_VERIFICATION`.
  - 선택 기반 로그인은 사용자 풀에 MFA가 필요한 경우 모든 앱 클라이언트에서만 `PASSWORD` 및 `PASSWORD_SRP` 요인을 제공합니다. 사용자 이름-암호 흐름에 대한 자세한 내용은 이 가이드의 인증 장에서 영구 암호로 로그인 및 영구 암호 및 보안 페이로드로 로그인 섹션을 참조하세요.
  - MFA가 선택 사항인 사용자 풀에서 MFA 요소를 구성한 사용자는 선택 기반 로그인 시 사용자 이름 암호 인증 흐름으로만 로그인할 수 있습니다. 이러한 사용자는 모든 클라이언트 기반 로그인 흐름을 사용할 수 있습니다.

  다음 표에서는 사용자 풀 MFA 설정 및 MFA 요소의 사용자 구성이 암호 없는 요소로 로그인하는 사용자의 기능에 미치는 영향을 설명합니다.

사용자 풀 MFA 설정	사용자 MFA 상태	Webauthn/OTP 사용 가능	암호 로그인 후 MFA에 대한 프롬프트 표시	WebAuthn/OTP로 로그인할 수 있음
필수	구성됨	아니요	예	아니요
필수	구성되지 않음	아니요	아니요(로그인할 수 없음)	아니요
선택 사항	구성됨	WebAuthn을 설정할 수 있지만 패스키로 로그인할 수 없음	예	아니요
선택 사항	구성되지 않음	예	아니요	예

사용자 풀 MFA 설정	사용자 MFA 상태	Webauthn/OTP 사용 가능	암호 로그인 후 MFA에 대한 프롬프트 표시	WebAuthn/OTP로 로그인할 수 있음
선택 사항(패스키 MFA가 활성화된 경우)	패스키 MFA 구성됨	예	예(사용자 확인이 포함된 패스키는 MFA를 독립적으로 충족함)	예(사용자 확인이 포함된 패스키는 MFA를 독립적으로 충족함)
필수(패스키 MFA가 활성화된 경우)	패스키 MFA 구성됨	예	예(사용자 확인이 포함된 패스키는 MFA를 독립적으로 충족함)	예(사용자 확인이 포함된 패스키는 MFA를 독립적으로 충족함)
꺼짐	임의	예	아니요	예

- 사용자가 선호하는 MFA 메서드는 암호를 복구하는 데 사용할 수 있는 메서드에 영향을 미칩니다. 선호하는 MFA가 이메일 메시지인 사용자는 이메일로 암호 재설정 코드를 받을 수 없습니다. 선호하는 MFA가 SMS 메시지인 사용자는 SMS로 암호 재설정 코드를 받을 수 없습니다.

  사용자가 선호하는 암호 재설정 방법을 사용할 수 없는 경우 암호 복구 설정에서 대체 옵션을 제공해야 합니다. 예를 들어 복구 메커니즘에 이메일이 최우선 순위일 수 있고 이메일 MFA가 사용자 풀의 옵션일 수 있습니다. 이 경우 SMS 메시지 계정 복구를 두 번째 옵션으로 추가하거나 관리 API 작업을 사용하여 해당 사용자의 암호를 재설정합니다.

  Amazon Cognito는 `InvalidParameterException` 오류 응답과 함께 유효한 복구 방법이 없는 사용자의 암호 재설정 요청에 응답합니다.

  UpdateUserPool에 대한 예제 요청 본문은 이메일 메시지 암호 재설정을 사용할 수 없는 경우 사용자가 SMS 메시지로 복구로 돌아갈 수 있는 `AccountRecoverySetting`을 보여줍니다.
- 사용자는 동일한 이메일 주소 또는 전화번호로 MFA 및 암호 재설정 코드를 받을 수 없습니다. MFA용 이메일 메시지의 일회용 암호(OTP)를 사용하는 경우 계정 복구에 SMS 메시지를 사용해야 합니다. MFA에 SMS 메시지의 OTP를 사용하는 경우 계정 복구에 이메일 메시지를 사용해야 합니다. MFA가 있는 사용자 풀에서 사용자가 이메일 주소 속성은 있지만 전화번호가 없거나, 전화번호 속성은 있지만 이메일 주소가 없는 경우 셀프 서비스 암호 복구를 완료하지 못할 수 있습니다.

사용자가 이 구성으로 사용자 풀에서 암호를 재설정할 수 없는 상태를 방지하려면 email 및 phone_number 속성을 필요에 따라 설정합니다. 또는 사용자가 가입하거나 관리자가 사용자 프로필을 생성할 때 항상 이러한 속성을 수집하고 설정하는 프로세스를 설정할 수 있습니다. 사용자에게 두 속성이 모두 있는 경우 Amazon Cognito는 사용자의 MFA 요소가 아닌 대상으로 암호 재설정 코드를 자동으로 전송합니다.

- 사용자 풀에서 MFA를 활성화하고 SMS 메시지 또는 이메일 메시지를 두 번째 요소로 선택하면 Amazon Cognito에서 확인하지 않은 전화번호 또는 이메일 속성으로 메시지를 보낼 수 있습니다. 사용자가 MFA를 완료하면 Amazon Cognito에서 phone_number_verified 또는 email_verified 속성을 true로 설정합니다.
- MFA 코드 제시 시도가 5번 실패하면 Amazon Cognito는 로그인 시도 실패에 대한 잠금 동작에 설명된 기하급수적 타임아웃 잠금 프로세스를 시작합니다.
- 계정이 사용자 풀에 대한 Amazon Simple Notification Service(Amazon SNS) 리소스가 포함된의 SMS 샌드박스에 AWS 리전 있는 경우 SMS 메시지를 보내기 전에 Amazon SNS의 전화번호를 확인해야 합니다. 자세한 내용은 Amazon Cognito 사용자 풀의 SMS 메시지 설정 섹션을 참조하세요.
- 위협 방지 기능을 사용하여 감지된 이벤트에 대한 응답으로 사용자의 MFA 상태를 변경하려면 Amazon Cognito 사용자 풀 콘솔에서 MFA를 활성화하고 선택 사항으로 설정합니다. 자세한 내용은 위협 방지 기능이 있는 고급 보안 섹션을 참조하세요.
- 이메일 및 SMS 메시지를 사용하려면 사용자에게 각각 이메일 주소와 전화번호 속성이 있어야 합니다. 사용자 풀에서 email 또는 phone_number를 필수 속성으로 설정할 수 있습니다. 이 경우 사용자는 전화번호를 제공하지 않으면 가입을 완료할 수 없습니다. 필요에 따라 이러한 속성을 설정하지 않고 이메일 또는 SMS 메시지 MFA를 수행하려는 경우 사용자가 가입할 때 이메일 주소 또는 전화번호를 묻는 메시지가 표시됩니다. 모범 사례로 이러한 속성을 검증하도록 사용자에게 자동으로 메시지를 보내도록 사용자 풀을 구성합니다.

  Amazon Cognito는 사용자가 SMS 또는 이메일 메시지로 임시 코드를 성공적으로 수신하고 VerifyUserAttribute API 요청에서 해당 코드를 반환한 경우 확인된 전화번호 또는 이메일 주소를 계산합니다. 또는 팀이 전화번호를 설정하고 AdminUpdateUserAttributes API 요청을 수행하는 관리 애플리케이션으로 확인된 것으로 표시할 수 있습니다.
- MFA를 필수로 설정하고 둘 이상의 인증 요소를 활성화한 경우 Amazon Cognito는 새 사용자에게 사용하려는 MFA 요소를 선택하라는 메시지를 표시합니다. SMS 메시지 MFA를 설정하려면 사용자에게 전화번호가 있어야 하고 이메일 메시지 MFA를 설정하려면 이메일 주소가 있어야 합니다. 사용 가능한 메시지 기반 MFA에 대해 정의된 속성이 사용자에게 없는 경우 Amazon Cognito는 TOTP MFA를 설정하라는 메시지를 표시합니다. MFA 요인(SELECT_MFA_TYPE)을 선택하고 선택한 요인(MFA_SETUP)을 설정하는 프롬프트는 InitiateAuth 및 AdminInitiateAuth API 작업에 대한 챌린지 응답으로 표시됩니다.

사용자 MFA 기본 설정

사용자는 여러 MFA 요소를 설정할 수 있습니다. 하나만 활성화할 수 있습니다. 사용자 풀 설정 또는 사용자 프롬프트에서 사용자에게 효과적인 MFA 기본 설정을 선택할 수 있습니다. 사용자 풀 설정 및 자체 사용자 수준 설정이 다음 조건을 충족할 때 사용자 풀은 사용자에게 MFA 코드를 묻는 메시지를 표시합니다.

1. MFA를 선택 사항으로 설정하거나 사용자 풀에 필요합니다.
2. 사용자에게 유효 `email` 또는 `phone_number` 속성이 있거나 TOTP용 인증자 앱을 설정했습니다.
3. 하나 이상의 MFA 요소가 활성 상태입니다.
4. 하나의 MFA 요소가 기본 설정으로 설정됩니다.

로그인 및 MFA에 동일한 요소를 사용하지 않도록 방지

하나의 로그인 요소를 일부 또는 모든 사용자가 사용할 수 있는 유일한 로그인 및 MFA 옵션으로 만드는 방식으로 사용자 풀을 구성할 수 있습니다. 이 결과는 기본 로그인 사용 사례가 이메일 메시지 또는 SMS 메시지 일회용 암호(OTPs. 다음 조건에서 사용자가 선호하는 MFA는 로그인과 동일한 유형의 팩터일 수 있습니다.

- MFA는 사용자 풀에 필요합니다.
- 이메일 및 SMS OTP는 사용자 풀에서 로그인 및 MFA 옵션을 사용할 수 있습니다.
- 사용자가 이메일 또는 SMS 메시지 OTP로 로그인합니다.
- 이메일 주소 속성은 있지만 전화번호 속성은 없거나 전화번호 속성은 있지만 이메일 주소 속성은 없습니다.

이 시나리오에서 사용자는 이메일 OTP로 로그인하고 이메일 OTP로 MFA를 완료할 수 있습니다. 이 옵션은 MFA의 필수 함수를 취소합니다. 일회용 암호로 로그인하는 사용자는 MFA와 다른 로그인 전송 방법을 사용할 수 있어야 합니다. 사용자에게 SMS 및 이메일 옵션이 모두 있는 경우 Amazon Cognito는 자동으로 다른 요소를 할당합니다. 예를 들어 사용자가 이메일 OTP로 로그인하면 선호하는 MFA는 SMS OTP입니다.

사용자 풀이 로그인 및 MFA 모두에 대해 OTP 인증을 지원하는 경우 다음 단계를 수행하여 동일 요소 인증을 해결합니다.

1. 이메일 및 SMS OTP를 모두 로그인 요인으로 활성화합니다.
2. 이메일 및 SMS OTP를 모두 MFA 요소로 활성화합니다.

3. 수집

사용자 풀 설정 및 MFA 옵션에 미치는 영향

사용자 풀의 구성은 사용자가 선택할 수 있는 MFA 메서드에 영향을 미칩니다. 다음은 MFA를 설정하는 사용자의 기능에 영향을 미치는 일부 사용자 풀 설정입니다.

- Amazon Cognito 콘솔의 로그인 메뉴에 있는 다중 인증 구성에서 MFA를 선택적 또는 필수로 설정하거나 끌 수 있습니다. 이 설정과 동등한 API는 `CreateUserPool`, `UpdateUserPool` 및 `SetUserPoolMfaConfig`의 MfaConfiguration 파라미터입니다.

  또한 다중 인증 구성에서 MFA 메서드 설정은 사용자가 설정할 수 있는 MFA 요소를 결정합니다. 이 설정과 동등한 API는 SetUserPoolMfaConfig 작업입니다.
- 사용자 계정 복구의 로그인 메뉴에서 사용자 풀이 암호를 잊어버린 사용자에게 메시지를 보내는 방법을 구성할 수 있습니다. 사용자의 MFA 메서드는 암호 찾기 코드에 대한 사용자 풀 전송 메서드와 동일한 MFA 전송 메서드를 가질 수 없습니다. 암호 찾기 전송 방법의 API 파라미터는 `CreateUserPool` 및 `UpdateUserPool`의 AccountRecoverySetting 파라미터입니다.

  예를 들어 복구 옵션이 이메일 전용인 경우 사용자는 이메일 MFA를 설정할 수 없습니다. 동일한 사용자 풀에서 이메일 MFA를 활성화하고 복구 옵션을 이메일 전용으로 설정할 수 없기 때문입니다. 이 옵션을 사용 가능한 경우 이메일, 그렇지 않은 경우 SMS로 설정하면 이메일이 복구의 우선 순위 옵션이지만, 사용자가 이메일 메시지 복구에 적합하지 않은 경우 사용자 풀이 SMS 메시지로 대체될 수 있습니다. 이 시나리오에서 사용자는 이메일 MFA를 기본 설정으로 설정하고 암호 재설정을 시도할 때만 SMS 메시지를 수신할 수 있습니다.
- 사용 가능한 MFA 메서드를 하나만 설정하면 사용자 MFA 기본 설정을 관리할 필요가 없습니다.
- 활성 SMS 구성은 SMS 메시지를 사용자 풀에서 사용 가능한 MFA 메서드로 자동 설정합니다.

  사용자 풀에 자체 Amazon SES 리소스가 포함된 활성 이메일 구성과 Essentials 또는 Plus 기능 플랜은 자동으로 이메일 메시지를 사용자 풀에서 사용 가능한 MFA 방법으로 만듭니다.
- MFA를 사용자 풀에서 필수로 설정하면 사용자는 MFA 메서드를 활성화하거나 비활성화할 수 없습니다. 선호하는 방법만 설정할 수 있습니다.
- 사용자 풀에서 MFA를 선택 사항으로 설정하면 관리형 로그인은 사용자에게 MFA를 설정하라는 메시지를 표시하지 않지만, 사용자에게 선호하는 MFA 메서드가 있는 경우 MFA 코드를 묻는 메시지를 표시합니다.
- 위협 방지를 활성화하고 전체 기능 모드에서 적응형 인증 응답을 구성하는 경우 MFA는 사용자 풀에서 선택 사항이어야 합니다. 적응형 인증을 사용하는 응답 옵션 중 하나는 로그인 시도가 위험 수준을 포함하도록 평가되는 사용자에게 MFA를 요구하는 것입니다.

콘솔의 가입 메뉴에 있는 필수 속성 설정에 따라 사용자가 애플리케이션에 가입하기 위해 이메일 주소 또는 전화번호를 제공해야 하는지 여부가 결정됩니다. 이메일 및 SMS 메시지는 사용자에게 해당 속성이 있을 때 적격 MFA 요소가 됩니다. `CreateUserPool`의 스키마 파라미터는 필요에 따라 속성을 설정합니다.

- MFA를 사용자 풀에 필수로 설정하고 사용자가 관리형 로그인으로 로그인하면 Amazon Cognito는 사용자 풀에 사용 가능한 방법에서 MFA 메서드를 선택하라는 메시지를 표시합니다. 관리형 로그인은 이메일 주소 또는 전화번호의 수집과 TOTP 설정을 처리합니다. 다음 다이어그램은 Amazon Cognito가 사용자에게 제공하는 옵션의 논리를 보여줍니다.

사용자에 대한 MFA 기본 설정 구성

액세스 토큰 권한이 있는 셀프 서비스 모델 또는 관리형 API 작업이 있는 관리자 관리형 모델의 사용자에 대해 MFA 기본 설정을 구성할 수 있습니다. 이러한 작업은 MFA 메서드를 활성화 또는 비활성화하고 여러 메서드 중 하나를 기본 옵션으로 설정합니다. 사용자가 MFA 기본 설정을 지정하면 Amazon Cognito는 로그인 시 기본 MFA 메서드의 코드를 제공하라는 메시지를 표시합니다. 기본 설정을 지정하지 않은 사용자는 `SELECT_MFA_TYPE` 챌린지에서 선호하는 방법을 선택하라는 프롬프트를 받게 됩니다.

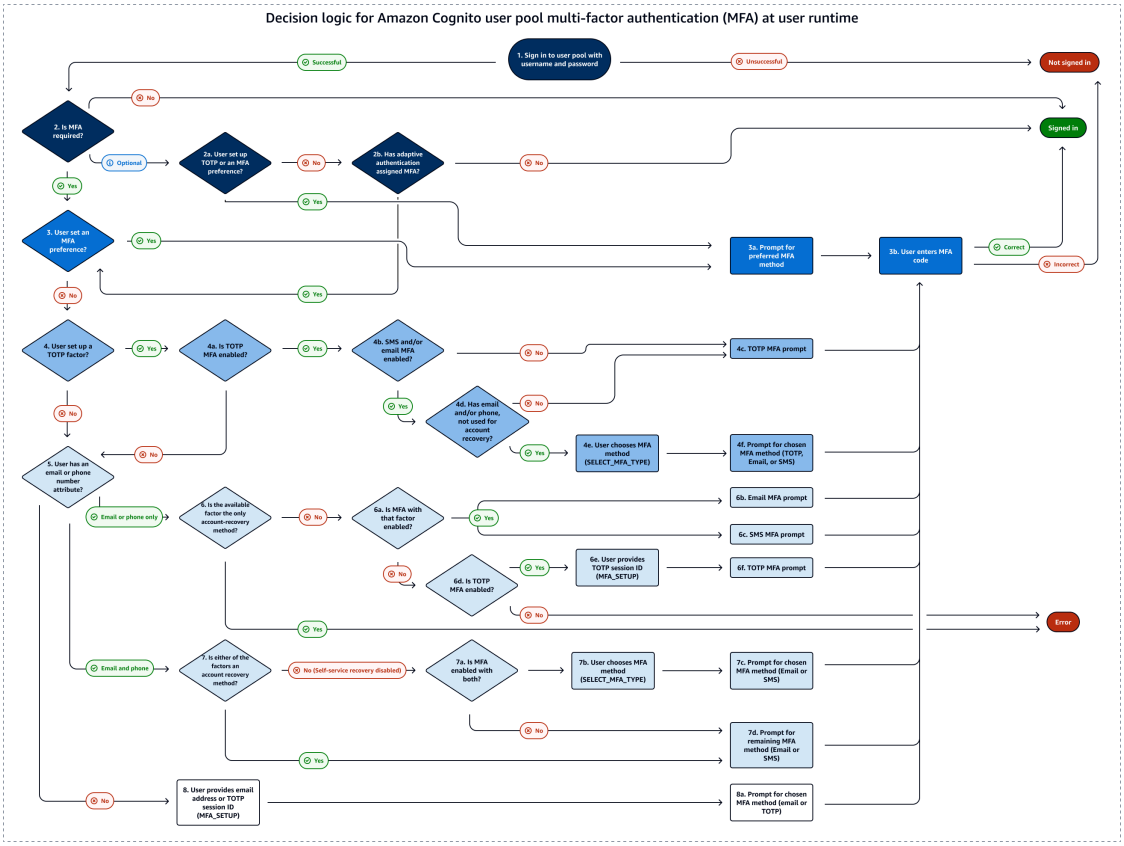
- 사용자 셀프 서비스 모델 또는 퍼블릭 애플리케이션에서 로그인한 사용자의 액세스 토큰으로 승인된 SetUserMfaPreference는 MFA 구성을 설정합니다.
- 관리자 관리형 또는 기밀 애플리케이션에서 관리 자격 AWS 증명으로 권한이 부여된 AdminSetUserPreference는 MFA 구성을 설정합니다.

Amazon Cognito 콘솔의 사용자 메뉴에서 사용자 MFA 기본 설정을 지정할 수도 있습니다. Amazon Cognito 사용자 풀 API의 공개 및 기밀 인증 모델에 대한 자세한 내용은 API, OIDC 및 관리형 로그인 페이지 인증 이해 섹션을 참조하십시오.

사용자 런타임 시 MFA 로직에 대한 세부 정보

사용자가 로그인할 때 수행할 단계를 결정하기 위해 사용자 풀은 사용자 MFA 기본 설정, 사용자 속성, 사용자 풀 MFA 설정, 위협 방지 조치 및 셀프 서비스 계정 복구 설정을 평가합니다. 그런 다음 사용자에게 로그인하거나, MFA 메서드를 선택하라는 메시지를 표시하거나, MFA 메서드를 설정하라는 메시지를 표시하거나, MFA를 입력하라는 메시지를 표시합니다. MFA 메서드를 설정하려면 사용자가 이메일 주소 또는 전화번호를 제공하거나 TOTP 인증자를 등록해야 합니다. 또한 MFA 옵션을 설정하고 미리 선호하는 옵션을 등록할 수 있습니다. 다음 다이어그램은 초기 가입 직후 로그인 시도에 대한 사용자 풀 구성의 자세한 영향을 나열합니다.

여기에 설명된 로직은 SDK 기반 애플리케이션 및 관리형 로그인 로그인에 적용되지만 관리형 로그인에는 표시되지 않습니다. MFA 문제를 해결할 때 사용자 결과에서 시작하여 해당 결정에 영향을 미친 사용자 프로필 및 사용자 풀 구성까지 역으로 작업합니다.



다음 목록은 결정 로직 다이어그램의 번호 지정에 해당하며 각 단계를 자세히 설명합니다.

는 성공한 인증과 흐름의 결론을 나타냅니다.

는 인증 실패를 나타냅니다.

1. 사용자가 로그인 화면에 사용자 이름 또는 사용자 이름과 암호를 표시합니다. 유효한 자격 증명을 제시하지 않으면 로그인 요청이 거부됩니다.
2. 사용자 이름-암호 인증에 성공한 경우 MFA가 필요한지, 선택 사항인지 또는 꺼져 있는지 확인합니다. 꺼져 있으면 올바른 사용자 이름과 암호로 인증이 성공합니다.

   a. MFA가 선택 사항인 경우 사용자가 이전에 TOTP 인증자를 설정했는지 확인합니다. TOTP를 설정한 경우 TOTP MFA를 입력하라는 메시

지를 표시합니다. MFA 문제에 성공적으로 응답하면 로그인됩니다.



b. 위협 방지의 적응형 인증 기능에 사용자가 MFA를 설정해야 하는지 확인합니다. MFA가 할당되지 않은 경우 사용자가 로그인됩니다.



3. MFA가 필요하거나 적응형 인증에 MFA가 할당된 경우 사용자가 MFA 요소를 활성화 및 기본 설정으로 설정했는지 확인합니다. 설정한 경우 해당 요소로 MFA를 입력하라는 메시지가 표시됩니다. MFA 문제에 성공적으로 응답하면 로그인됩니다.



4. 사용자가 MFA 기본 설정을 지정하지 않은 경우 사용자가 TOTP 인증자를 등록했는지 확인합니다.

   a. 사용자가 TOTP 인증자를 등록한 경우 사용자 풀에서 TOTP MFA를 사용할 수 있는지 확인합니다(사용자가 이전에 인증자를 설정한 후 TOTP MFA를 비활성화할 수 있음).

   b. 사용자 풀에서도 이메일 메시지나 SMS 메시지 MFA를 사용할 수 있는지 확인합니다.

   c. 이메일 또는 SMS MFA를 사용할 수 없는 경우 사용자에게 TOTP MFA를 입력하라는 메시지를 표시합니다. MFA 문제에 성공적으로 응답하면 로그인됩니다.



   d. 이메일 또는 SMS MFA를 사용할 수 있는 경우 사용자에게 해당 `email` 또는 `phone_number` 속성이 있는지 확인합니다. 그렇다면 셀프 서비스 계정 복구의 기본 방법이 아니고 MFA에 대해 활성화된 모든 속성을 사용할 수 있습니다.

   e. TOTP와 사용 가능한 SMS 또는 이메일 MFA 요소를 포함하는 `MFAS_CAN_SELECT` 옵션을 사용하여 사용자에게 `SELECT_MFA_TYPE` 문제 메시지를 표시합니다.

   f. 사용자에게 `SELECT_MFA_TYPE` 문제에 대한 응답으로 선택한 요소를 묻는 메시지를 표시합니다. MFA 문제에 성공적으로 응답하면 로그인됩니다.



5. 사용자가 TOTP 인증자를 등록하지 않았거나 TOTP MFA가 현재 비활성화되어 있는 경우 사용자에게 `email` 또는 `phone_number` 속성이 있는지 확인합니다.

6. 사용자에게 이메일 주소만 있거나 전화번호만 있는 경우 해당 속성이 사용자 풀이 암호 재설정을 위해 계정 복구 메시지를 보내기 위해 구현하는 방법인지 확인합니다. true인 경우 MFA가 필요한 로그인을 완료할 수 없으며 Amazon Cognito는 오류를 반환합니다. 이 사용자의 로그인을 활성화하려면 복구되지 않는 속성을 추가하거나 TOTP 인증자를 등록해야 합니다.



a. 사용 가능한 복구 불가능한 이메일 주소나 전화번호가 있는 경우 해당 이메일이나 SMS MFA 요소가 활성화되어 있는지 확인하세요.

b. 복구되지 않는 이메일 주소 속성이 있고 이메일 MFA가 활성화된 경우 `EMAIL_OTP` 문제를 표시합니다. MFA 문제에 성공적으로 응답하면 로그인됩니다.



c. 복구되지 않는 전화번호 속성이 있고 SMS MFA가 활성화된 경우 `SMS_MFA` 문제를 입력하라는 메시지를 표시합니다. MFA 문제에 성공적으로 응답하면 로그인됩니다.



d. 활성화된 이메일 또는 SMS MFA 요소에 적합한 속성이 없는 경우 TOTP MFA가 활성화되어 있는지 확인합니다. TOTP MFA가 비활성화된 경우 MFA가 필요한 로그인을 완료할 수 없으며 Amazon Cognito는 오류를 반환합니다. 이 사용자의 로그인을 활성화하려면 복구되지 않는 속성을 추가하거나 TOTP 인증자를 등록해야 합니다.



> **Note**
>
> 사용자에게 TOTP 인증자가 있지만 TOTP MFA가 비활성화된 경우 이 단계는 이미 아니요로 평가되었습니다.

e. TOTP MFA가 활성화된 경우 `MFAS_CAN_SETUP` 옵션에서 `SOFTWARE_TOKEN_MFA`를 사용하여 사용자에게 `MFA_SETUP` 문제를 제시합니다. 이 문제를 완료하려면 사용자에 대한 TOTP 인증자를 별도로 등록하고 `"ChallengeName": "MFA_SETUP", "ChallengeResponses": {"USERNAME": "[username]", "SESSION": "[Session ID from VerifySoftwareToken]}"`로 응답해야 합니다.

f. 사용자가 VerifySoftwareToken 요청의 세션 토큰으로 `MFA_SETUP` 문제에 응답한 후 `SOFTWARE_TOKEN_MFA` 문제 메시지를 표시합니다. MFA 문제에 성공적으로 응답하면 로그인됩니다.



7. 사용자에게 이메일 주소와 전화번호가 모두 있는 경우 암호 재설정을 위한 계정 복구 메시지의 기본 방법이 되는 속성을 결정합니다.

a. 셀프 서비스 계정 복구가 비활성화된 경우 두 속성 중 하나를 MFA에 사용할 수 있습니다. 이메일 및 SMS MFA 요소 중 하나 또는 둘 모두가 활성화되어 있는지 확인합니다.

b. 두 속성이 모두 MFA 요소로 활성화된 경우 MFAS_CAN_SELECT 옵션 SMS_MFA 및 EMAIL_OTP에 대한 SELECT_MFA_TYPE 문제를 사용자에게 표시합니다.

c. SELECT_MFA_TYPE 문제에 대한 응답으로 선택한 요소를 묻는 메시지를 표시합니다. MFA 문제에 성공적으로 응답하면 로그인됩니다.



d. 하나의 속성만 적격 MFA 요소인 경우 나머지 요소에 대한 문제를 표시합니다. MFA 문제에 성공적으로 응답하면 로그인됩니다.



이러한 결과는 다음과 같은 시나리오에서 발생합니다.

i. email 및 phone_number 속성이 있는 경우 SMS 및 이메일 MFA가 활성화되고 기본 계정 복구 방법은 이메일 또는 SMS 메시지입니다.

ii. email 및 phone_number 속성이 있는 경우 SMS MFA 또는 이메일 MFA만 활성화되고 셀프 서비스 계정 복구는 비활성화됩니다.

8. 사용자가 TOTP 인증자를 등록하지 않았고 email 또는 phone_number 속성이 없는 경우 MFA_SETUP 문제를 입력하라는 메시지를 표시합니다. MFAS_CAN_SETUP의 목록에는 기본 계정 복구 옵션이 아닌 사용자 풀에서 활성화된 모든 MFA 요소가 포함됩니다. 이메일 또는 TOTP MFA에 대해 ChallengeResponses를 사용하여 이 문제에 응답할 수 있습니다. SMS MFA를 설정하려면 전화번호 속성을 별도로 추가하고 인증을 다시 시작합니다.

TOTP MFA의 경우 `"ChallengeName": "MFA_SETUP", "ChallengeResponses": {"USERNAME": "[username]", "SESSION": "[Session ID from VerifySoftwareToken]"}`로 응답합니다.

이메일 MFA의 경우 `"ChallengeName": "MFA_SETUP", "ChallengeResponses": {"USERNAME": "[username]", "email": "[user's email address]"}`로 응답합니다.

a. SELECT_MFA_TYPE 문제에 대한 응답으로 선택한 요소를 묻는 메시지를 표시합니다. MFA 문제에 성공적으로 응답하면 로그인됩니다.

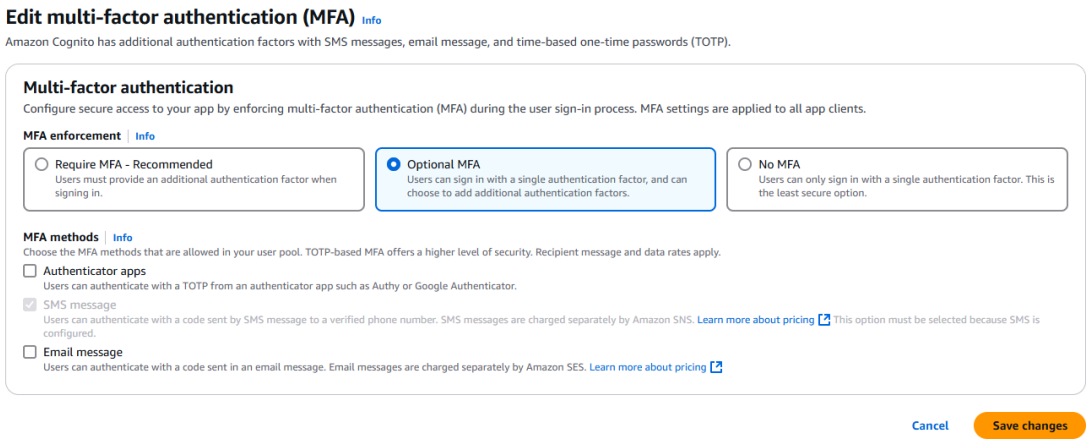


다중 인증을 위한 사용자 풀 구성

Amazon Cognito 콘솔 또는 SetUserPoolMfaConfig API 작업 및 SDK 메서드를 사용하여 MFA를 구성할 수 있습니다.

Amazon Cognito 콘솔에서 MFA를 구성하려면

1. Amazon Cognito 콘솔에 로그인합니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. 로그인 메뉴를 선택합니다. 다중 인증을 찾아서 편집을 선택합니다.
5. 사용자 풀에 사용하려는 MFA 시행(MFA enforcement) 방법을 선택합니다.



   a. MFA 필수(Require MFA). 사용자 풀에 있는 모든 사용자는 추가 인증 요소로 SMS, 이메일 또는 시간 기반 일회용 암호(TOTP) 코드를 사용하여 로그인해야 합니다.

   b. 선택적 MFA. 사용자에게 추가 로그인 팩터를 등록하는 옵션을 제공할 수 있지만 MFA를 구성하지 않은 사용자의 로그인도 허용할 수 있습니다. 조정 인증을 사용하는 경우 이 옵션을 선택합니다. 조정 인증에 대한 자세한 내용은 위협 방지 기능이 있는 고급 보안 섹션을 참조하세요.

   c. MFA 없음(No MFA). 사용자는 추가 로그인 요소를 등록할 수 없습니다.

6. 앱에서 지원하는 MFA 방법(MFA methods)을 선택합니다. 이메일 메시지, SMS 메시지 또는 TOTP 생성 인증자 앱을 두 번째 요소로 설정할 수 있습니다.
7. SMS 문자 메시지를 두 번째 팩터로 사용하는데 SMS 메시지용 Amazon Simple Notification Service(Amazon SNS)와 함께 사용하도록 구성된 IAM 역할이 없는 경우 콘솔에서 이러한 역할을 생성합니다. 사용자 풀의 인증 방법 메뉴에서 SMS를 찾아 편집을 선택합니다. Amazon Cognito가 사용자에게 SMS 메시지를 전송하도록 허용하는 기존 역할을 사용할 수도 있습니다. 자세한 내용은 IAM 역할을 참조하세요.

   이메일 메시지를 두 번째 요소로 사용하고 Amazon Simple Email Service(Amazon SES)에서 이메일 메시지에 사용할 발신 ID를 구성하지 않은 경우 콘솔에서 ID를 하나 만듭니다. SES로 이메일 전송 옵션을 선택해야 합니다. 사용자 풀의 인증 방법 메뉴에서 이메일을 찾아 편집을 선택합니다.

목록에 있는 사용 가능한 확인된 ID에서 보낸 사람 이메일 주소를 선택합니다. 확인된 도메인(예: `example.com`)을 선택하는 경우 `admin-noreply@example.com`과 같이 확인된 도메인에서 보낸 사람 발신자 이름도 구성해야 합니다.

8. 변경 사항 저장을 선택합니다.

SMS 및 이메일 메시지 MFA

SMS 및 이메일 MFA 메시지는 사용자가 로그인하기 전에 메시지 대상에 액세스할 수 있음을 확인합니다. 암호뿐만 아니라 원래 사용자의 SMS 메시지 또는 이메일 받은 편지함에도 액세스할 수 있는지 확인합니다. Amazon Cognito는 사용자가 사용자 이름과 암호를 성공적으로 제공한 후 사용자 풀이 보낸 단축 코드를 제공하도록 요청합니다.

SMS 및 이메일 메시지 MFA는 사용자가 프로필에 이메일 주소 또는 전화번호를 추가한 후 추가 구성이 필요하지 않습니다. Amazon Cognito는 확인되지 않은 이메일 주소와 전화번호로 메시지를 보낼 수 있습니다. 사용자가 첫 번째 MFA를 완료하면 Amazon Cognito는 이메일 주소 또는 전화번호를 확인된 것으로 표시합니다.

MFA 인증은 MFA를 사용하는 사용자가 애플리케이션에 사용자 이름과 암호를 입력할 때 시작됩니다. 애플리케이션은 InitiateAuth 또는 AdminInitiateAuth API 요청을 호출하는 SDK 메서드에 이러한 초기 파라미터를 제출합니다. API 응답의 `ChallengeParameters`에는 인증 코드가 전송된 위치를 나타내는 `CODE_DELIVERY_DESTINATION` 값이 포함되어 있습니다. 애플리케이션에서 사용자에게 휴대폰을 확인하라는 메시지를 표시하고 코드에 대한 입력 요소를 포함합니다. 코드를 입력하면 챌린지 응답 API 요청에 제출하여 로그인 프로세스를 완료합니다.

MFA를 사용하는 사용자가 관리형 로그인 페이지에서 사용자 이름과 암호로 로그인하면 MFA 코드를 자동으로 입력하라는 메시지가 표시됩니다.

사용자 풀은 AWS 계정의 Amazon Simple Notification Service(Amazon SNS) 리소스를 사용하여 MFA 및 기타 Amazon Cognito 알림에 대한 SMS 메시지를 보냅니다. 마찬가지로 사용자 풀은 계정의 Amazon Simple Email Service(Amazon SES) 리소스로 이메일 메시지를 전송합니다. 이러한 연결된 서비스에는 메시지 전송 AWS 청구서에 대한 자체 비용이 발생합니다. 또한 프로덕션 볼륨에서 메시지를 보내기 위한 추가 요구 사항도 있습니다. 자세한 내용은 다음 링크를 참조하십시오.

- Amazon Cognito 사용자 풀의 SMS 메시지 설정
- 전 세계 SMS 요금
- Amazon Cognito 사용자 풀에 대한 이메일 설정
- Amazon SES 요금

SMS 및 이메일 메시지 MFA 고려 사항

- 사용자가 이메일 MFA로 로그인하도록 허용하려면 사용자 풀에 다음 구성 옵션이 있어야 합니다.
  1. 사용자 풀에 Plus 또는 Essentials 기능 플랜이 있습니다. 자세한 내용은 사용자 풀 기능 플랜 섹션을 참조하세요.
  2. 사용자 풀은 자체 Amazon SES 리소스와 함께 이메일 메시지를 전송합니다. 자세한 내용은 Amazon SES 이메일 구성 섹션을 참조하세요.
- MFA 코드는 앱 클라이언트에 대해 설정한 인증 흐름 세션 기간 동안 유효합니다.

  앱 클라이언트를 편집할 때 앱 클라이언트 메뉴에서 Amazon Cognito 콘솔에서 인증 흐름 세션의 기간을 설정합니다. `CreateUserPoolClient` 또는 `UpdateUserPoolClient` API 요청에서 인증 흐름 세션 기간을 설정할 수도 있습니다. 자세한 내용은 인증 세션의 예 섹션을 참조하세요.
- 사용자가 Amazon Cognito가 확인되지 않은 전화번호 또는 이메일 주소로 보낸 SMS 또는 이메일 메시지의 코드를 성공적으로 제공하면 Amazon Cognito는 해당 속성을 확인된 것으로 표시합니다.
- 사용자가 MFA와 연결된 전화번호 또는 이메일 주소의 값을 셀프 서비스 방식으로 변경하려면 액세스 토큰으로 로그인하여 요청을 승인해야 합니다. 현재 전화번호 또는 이메일 주소에 액세스할 수 없는 경우 로그인할 수 없습니다. 팀은 AdminUpdateUserAttributes API 요청에서 관리자 AWS 자격 증명을 사용하여 이러한 값을 변경해야 합니다.
- 사용자 풀에서 SMS를 구성한 후에는 SMS 메시지를 사용 가능한 MFA 요인으로 비활성화할 수 없습니다.

TOTP 소프트웨어 토큰 MFA

사용자 풀에서 TOTP 소프트웨어 토큰 MFA를 설정하면 사용자가 사용자 이름 및 암호로 로그인한 다음 TOTP를 사용하여 인증을 완료합니다. 사용자가 사용자 이름 및 암호를 설정하고 확인하면 MFA에서 TOTP 소프트웨어 토큰을 활성화할 수 있습니다. 앱이 Amazon Cognito 관리형 로그인을 사용하여 사용자를 로그인하는 경우 사용자는 자신의 사용자 이름 및 암호를 제출한 다음 추가 로그인 페이지에서 TOTP 암호를 제출합니다.

Amazon Cognito 콘솔에서 사용자 풀에 대해 TOTP MFA를 활성화하거나 Amazon Cognito API 작업을 사용할 수 있습니다. 사용자 풀 수준에서 SetUserPoolMfaConfig를 호출하여 MFA를 구성하고 TOTP MFA를 활성화할 수 있습니다.

> **Note**
>
> 사용자 풀에 대해 TOTP 소프트웨어 토큰 MFA를 활성화하지 않는 경우 Amazon Cognito는 토큰을 사용하여 사용자를 연결하거나 확인할 수 없습니다. 이 경우 사용자는

`SoftwareTokenMFANotFoundException` 설명과 함께 `Software Token MFA has not been enabled by the userPool` 예외를 수신합니다. 나중에 사용자 풀에 대해 소프트웨어 토큰 MFA를 비활성화하면 이전에 TOTP 토큰을 연결하고 확인한 사용자는 해당 토큰을 MFA에 계속 사용할 수 있습니다.

사용자에 대해 TOTP를 구성하는 것은 사용자가 일회용 암호를 입력하여 유효성을 검사하는 시크릿 코드를 받는 다단계 프로세스입니다. 그런 다음 사용자에 대해 TOTP MFA를 활성화하거나 TOTP를 사용자에 대한 기본 MFA 방법으로 설정할 수 있습니다.

TOTP MFA를 요구하도록 사용자 풀을 구성하고 사용자가 관리형 로그인에서 앱에 가입하면 Amazon Cognito가 사용자 프로세스를 자동화합니다. Amazon Cognito는 사용자에게 MFA 방식을 선택하라는 메시지를 표시하고 QR 코드를 표시하여 인증 앱을 설정하고 MFA 등록을 확인합니다. 사용자가 SMS와 TOTP MFA 중에서 선택할 수 있도록 허용한 사용자 풀에서 Amazon Cognito는 사용자에게 방식을 선택할 수 있는 기회도 제공합니다.

Important

사용자 풀과 연결된 AWS WAF 웹 ACL이 있고 웹 ACL의 규칙에 CAPTCHA가 표시되는 경우 관리형 로그인 TOTP 등록에서 복구할 수 없는 오류가 발생할 수 있습니다. CAPTCHA 작업이 있고 관리형 로그인 TOTP에 영향을 주지 않는 규칙을 생성하려면 관리형 로그인 TOTP MFA를 위한 AWS WAF 웹 ACL 구성 섹션을 참조하세요. AWS WAF 웹 ACL 및 Amazon Cognito에 대한 자세한 내용은 AWS WAF 웹 ACL을 사용자 풀과 연결 단원을 참조하시기 바랍니다.

AWS SDK 및 Amazon Cognito 사용자 풀 API를 사용하여 사용자 지정 빌드 UI에서 TOTP MFA를 구현하려면 섹션을 참조하세요사용자에 대한 TOTP MFA 구성.

사용자 풀에 MFA를 추가하는 방법은 사용자 풀에 MFA 추가 섹션을 참조하세요.

TOTP MFA 고려 사항 및 제한 사항

1. Amazon Cognito는 TOTP 코드를 생성하는 인증자 앱을 통해 소프트웨어 토큰 MFA를 지원합니다. Amazon Cognito는 하드웨어 기반 MFA를 지원하지 않습니다.
2. 사용자 풀이 TOTP를 구성하지 않은 사용자에 대해 TOTP를 요구하는 경우 사용자는 앱이 사용자의 TOTP MFA를 활성화하는 데 사용할 수 있는 일회성 액세스 토큰을 수신합니다. 사용자가 추가 TOTP 로그인 팩터를 등록할 때까지 후속 로그인 시도는 실패합니다.

- `SignUp` API 작업과 함께 또는 관리형 로그인을 통해 사용자 풀에 가입하는 사용자는 가입을 완료할 때 일회성 토큰을 수신합니다.
- 사용자를 생성하고 해당 사용자가 자신의 초기 암호를 설정하면 Amazon Cognito가 관리형 로그인에서 해당 사용자에게 일회성 토큰을 발행합니다. 사용자에 대한 영구 암호를 설정하면 사용자가 처음 로그인할 때 Amazon Cognito가 일회성 토큰을 발급합니다.
- Amazon Cognito는 InitiateAuth 또는 AdminInitiateAuth API 작업으로 로그인하는 관리자가 생성한 사용자에게 일회성 토큰을 발행하지 않습니다. 사용자가 자신의 초기 암호를 설정하는 문제에 성공한 후 또는 사용자에 대한 영구 암호가 설정된 경우 Amazon Cognito는 즉시 해당 사용자에게 MFA를 설정하는 문제를 냅니다.

3. MFA가 필요한 사용자 풀의 사용자가 이미 일회성 액세스 토큰을 수신했지만 TOTP MFA를 설정하지 않은 경우 사용자는 MFA를 설정할 때까지 관리형 로그인으로 로그인할 수 없습니다. 액세스 토큰 대신 AssociateSoftwareToken 요청의 InitiateAuth 또는 AdminInitiateAuth에 대해 `MFA_SETUP` 문제의 `session` 응답 값을 사용할 수 있습니다.
4. 사용자가 TOTP를 설정한 경우 나중에 사용자 풀에 대해 TOTP가 비활성화되어도 해당 TOTP를 MFA에 사용할 수 있습니다.
5. Amazon Cognito는 HMAC-SHA1 해시 함수를 사용하여 코드를 생성하는 인증 앱의 TOTP만 허용합니다. SHA-256 해싱으로 생성된 코드는 `Code mismatch` 오류를 반환합니다.

사용자에 대한 TOTP MFA 구성

사용자가 처음 로그인하면 앱은 일회성 액세스 토큰을 사용하여 TOTP 프라이빗 키를 생성하고 해당 키를 텍스트 또는 QR 코드 형식으로 사용자에게 제공합니다. 사용자가 인증자 앱을 구성하고 후속 로그인 시도에 TOTP를 제공합니다. 앱 또는 관리형 로그인은 MFA 문제 응답에서 Amazon Cognito에 TOTP를 제공합니다.

경우에 따라 관리형 로그인은 새 사용자에게 TOTP 인증자를 설정하라는 메시지를 표시합니다. 자세한 내용은 사용자 런타임 시 MFA 로직에 대한 세부 정보 섹션을 참조하세요.

주제

- TOTP 소프트웨어 토큰 연결
- TOTP 토큰 확인
- TOTP MFA로 로그인
- TOTP 토큰 제거

TOTP 소프트웨어 토큰 연결

TOTP 토큰을 연결하려면 일회용 암호로 검증해야 하는 시크릿 코드를 사용자에게 보냅니다. 토큰 연결에는 세 가지 단계가 필요합니다.

1. 사용자가 TOTP 소프트웨어 토큰 MFA를 선택하면 AssociateSoftwareToken을 호출하여 해당 사용자 계정에 대해 생성된 고유한 공유 시크릿 키 코드를 반환합니다. 액세스 토큰 또는 세션 문자열로 AssociateSoftwareToken을 인증할 수 있습니다.
2. 앱은 사용자에게 프라이빗 키 또는 프라이빗 키에서 생성한 QR 코드를 제공합니다. 사용자는 애플리케이션이 프라이빗 키에서 생성하는 QR 코드를 스캔하거나 키를 수동으로 입력하여 Google Authenticator와 같은 TOTP 생성 앱에 키를 입력해야 합니다.
3. 사용자가 키를 입력하거나 Google Authenticator와 같은 인증자 앱에 QR 코드를 스캔하면 앱에서 코드 생성을 시작합니다.

TOTP 토큰 확인

다음으로 TOTP 토큰을 확인합니다. 다음과 같이 사용자의 샘플 코드를 요청하고 해당 코드를 Amazon Cognito 서비스에 제공하여 사용자가 TOTP 코드를 성공적으로 생성하고 있는지 확인합니다.

1. 앱에서 사용자에게 인증자 앱을 올바르게 설정했음을 보여주는 코드를 입력하라는 메시지를 표시합니다.
2. 사용자의 인증자 앱에서 임시 암호를 표시합니다. 인증자 앱은 이 암호를 사용자에게 제공된 시크릿 키를 기반으로 생성합니다.
3. 사용자가 자신의 임시 암호를 입력합니다. `VerifySoftwareToken` API 요청에서 앱이 임시 암호를 Amazon Cognito에 전달합니다.
4. Amazon Cognito는 사용자와 연결된 시크릿 키를 유지하고 TOTP를 생성하여 사용자가 제공한 TOTP와 비교합니다. 일치하는 경우 `VerifySoftwareToken`에서 `SUCCESS` 응답을 반환합니다.
5. Amazon Cognito에서 TOTP 팩터를 사용자와 연결합니다.
6. `VerifySoftwareToken` 작업에서 `ERROR` 응답을 반환하는 경우 사용자의 클록이 정확하고 최대 재시도 횟수를 초과하지 않았는지 확인합니다. Amazon Cognito는 낮은 클록 스큐를 고려해서 시도 전후 30초 이내 TOTP 토큰은 허용합니다. 문제를 해결했으면 VerifySoftwareToken 작업을 다시 시도합니다.

TOTP MFA로 로그인

이 시점에서 사용자는 시간 기반 일회용 암호를 사용하여 로그인합니다. 프로세스는 다음과 같습니다.

1. 사용자는 사용자 이름과 암호를 입력하여 클라이언트 앱에 로그인합니다.
2. TOTP MFA 문제가 호출되고, 앱은 사용자에게 임시 암호를 입력하라는 메시지를 표시합니다.
3. 사용자는 연결된 TOTP 생성 앱에서 임시 암호를 가져옵니다.
4. 사용자는 클라이언트 앱에 TOTP 코드를 입력합니다. 앱이 이를 확인하라고 Amazon Cognito 서비스에 알립니다. 각 로그인에 대해 RespondToAuthChallenge를 호출하여 새 TOTP 인증 챌린지에 대한 응답을 받아야 합니다.
5. Amazon Cognito에서 토큰을 확인하면 로그인이 성공하고 사용자가 인증 흐름을 계속합니다.

TOTP 토큰 제거

마지막으로, 앱에서 사용자가 TOTP 구성을 비활성화할 수 있도록 허용해야 합니다. 사용자의 TOTP 소프트웨어 토큰을 삭제하려면 AdminDeleteSoftwareToken을 사용합니다. 사용자의 소프트웨어 토큰을 삭제한 후에는 더 이상 TOTP MFA로 로그인할 수 없으며 새 소프트웨어 토큰을 등록할 수 있습니다. 전화 분실 또는 재설정과 같이 사용자가 TOTP 생성 디바이스에 액세스할 수 없는 경우이 작업을 사용합니다. 사용자 풀에 MFA가 필요하고 다른 요소를 사용할 수 없는 경우 Amazon Cognito는 사용자의 다음 로그인 시 `MFA_SETUP` 챌린지를 반환합니다. SMS 메시지 MFA와 같은 다른 요소를 사용할 수 있는 경우 로그인은 해당 요소로 돌아갑니다.

사용자가 여전히 TOTP 생성 디바이스에 액세스할 수 있는 경우 소프트웨어 토큰을 삭제할 필요가 없습니다. Amazon Cognito는 사용자가 새 소프트웨어 토큰을 확인할 때 기존 소프트웨어 토큰의 연결을 해제합니다. 사용자의 TOTP MFA를 비활성화하려면 SetUserMFAPreference를 호출하여 MFA를 사용하지 않거나 SMS MFA만 사용하도록 사용자를 수정합니다.

1. MFA를 재설정하려는 사용자를 위한 인터페이스를 앱에 생성합니다. 이 인터페이스에서 사용자에게 암호를 입력하라는 메시지를 표시합니다.
2. Amazon Cognito가 TOTP MFA 챌린지를 반환하는 경우 SetUserMFAPreference로 사용자의 MFA 기본 설정을 업데이트합니다.
3. 앱에서 사용자에게 MFA를 비활성화했음을 알리고 다시 로그인하라는 메시지를 표시합니다.

관리형 로그인 TOTP MFA를 위한 AWS WAF 웹 ACL 구성

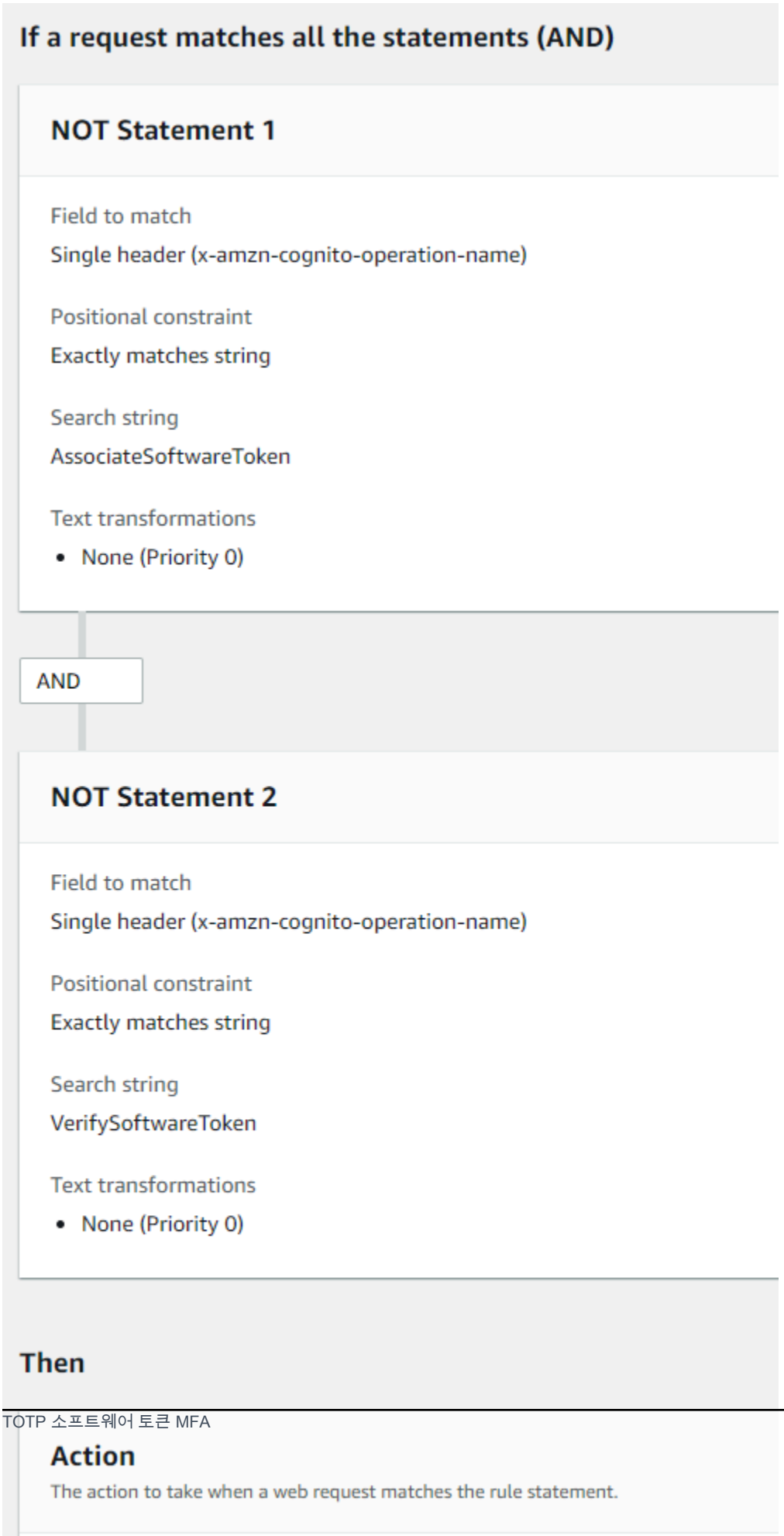
사용자 풀과 연결된 AWS WAF 웹 ACL이 있고 웹 ACL의 규칙에 CAPTCHA가 있는 경우 관리형 로그인 TOTP 등록에서 복구할 수 없는 오류가 발생할 수 있습니다. AWS WAF CAPTCHA 규칙은 관리형 로그인 및 클래식 호스팅 UI의 TOTP MFA에만 이러한 영향을 미칩니다. SMS MFA는 영향을 받지 않습니다.

CAPTCHA 규칙으로 인해 사용자가 TOTP MFA 설정을 완료할 수 없으면 Amazon Cognito에서 다음 오류를 표시합니다.

Request not allowed due to WAF captcha.

이 오류는가 사용자 풀이 백그라운드에서 수행하는 AssociateSoftwareToken 및 VerifySoftwareToken API 요청에 대한 응답으로 CAPTCHA를 입력하라는 AWS WAF 메시지를 표시할 때 발생합니다. 관리형 로그인 페이지에서 TOTP에 영향을 미치지 않고 CAPTCHA 작업이 있는 규칙을 만들려면 규칙의 CAPTCHA 작업에서 `AssociateSoftwareToken` 및 `VerifySoftwareToken`의 `x-amzn-cognito-operation-name` 헤더 값을 제외합니다.

다음 스크린샷은 `x-amzn-cognito-operation-name` 헤더 값이 `AssociateSoftwareToken` 또는가 아닌 모든 요청에 CAPTCHA 작업을 적용하는 예제 AWS WAF 규칙을 보여줍니다`VerifySoftwareToken`.


If a request matches all the statements (AND)
NOT Statement 1
Field to match
Single header (x-amzn-cognito-operation-name)
Positional constraint
Exactly matches string
Search string
AssociateSoftwareToken
Text transformations
None (Priority 0)
AND
NOT Statement 2
Field to match
Single header (x-amzn-cognito-operation-name)
Positional constraint
Exactly matches string
Search string
VerifySoftwareToken
Text transformations
None (Priority 0)
Then
Action
The action to take when a web request matches the rule statement.

AWS WAF 웹 ACLsAWS WAF 웹 ACL을 사용자 풀과 연결. Amazon Cognito

ACR 및 AMR 클레임을 사용한 인증 수준

Amazon Cognito는 사용자가 얼마나 강력하게 인증했는지, 어떤 방법을 사용했는지 기록할 수 있습니다. 사용자가 로그인할 때 대상 인증 수준을 입력으로 제공하고 Amazon Cognito는 사용자가 도달한 수준과 사용자가 완료한 메서드를 발급하는 토큰에 클레임으로 보고합니다. 이러한 클레임은 사용자의 초기 로그인을 포함한 모든 로그인에 적용됩니다.

관리형 로그인 권한 부여 및 로그인 엔드포인트의 `acr_values` 파라미터 또는 USER_AUTH API 흐름의 동등한 TARGET_ACR_VALUES 파라미터를 사용하여 인증 수준을 요청합니다. 둘 다 대상 인증 수준의 정렬된 목록을 수락합니다. 선택적 `max_age` 파라미터(MAX_AGEAPI의)를 추가하여 사용자가 최근 기간 내에 인증하도록 요구할 수 있습니다. 이러한 입력에 대한 응답으로 Amazon Cognito는 발급하는 토큰에 두 가지 클레임을 추가합니다.

- `acr` 클레임은 사용자가 만족한 인증 수준을 나타내는 문자열입니다. 예: `urn:cognito:loa:3`.
- `amr` 클레임은 사용자가 완료한 인증 방법을 나타내는 문자열 배열입니다. 예: `["pwd", "otp", "mfa"]`.

`acr` 및 `amr` 클레임은 OpenID Connect Core 사양의 정의를 따릅니다. Amazon Cognito는 자체 ACR 수준 URIs 사용합니다. Amazon Cognito 수준을 애플리케이션에 필요한 표준으로 변환할 수 있습니다.

이러한 클레임의 일반적인 사용 사례는 단계적 인증입니다. 단계적 인증에서 애플리케이션은 사용자가 민감한 작업을 수행하기 전에 더 강력한 방법으로 인증해야 합니다. 프롬프트를 표시할지 여부를 결정하기 위해 애플리케이션은 사용자 ID 토큰의 `acr` 수준 및 `auth_time` 클레임을 작업에 필요한 수준 및 최신성과 비교하고 아직 충족되지 않은 경우에만 단계적 증가를 요청합니다. 단계적 증가가 필요한 경우 사용자를를 사용하여 관리형 로그인 권한 부여 또는 로그인 엔드포인트로 리디렉션`acr_values`하거나를 사용하여 USER_AUTH API 흐름을 호출합니다TARGET_ACR_VALUES. 예제는 단계적 인증 시나리오를 참조하세요.

모든 Amazon Cognito 로그인 진입점에서 ACR 및 AMR을 요청할 수 있습니다. 각 진입점의 메커니즘은 해당 진입점과 함께 문서화됩니다.

- SDK 및 API 로그인 - TARGET_ACR_VALUES 및 MAX_AGE 파라미터를 사용하여 USER_AUTH 흐름의 대상 수준을 요청합니다. 파라미터 및 단계적 동작은 섹션을 참조하세요USER_AUTH 흐름을 사용한 단계적 인증.
- 관리형 로그인 및 호스팅 UI - 권한 부여 `acr_values` 및 로그인 엔드포인트에 및 `max_age` 파라미터를 전달합니다. 자세한 내용은 권한 부여 엔드포인트 및 로그인 엔드포인트를 참조하세요.

- 페더레이션 로그인 - Amazon Cognito는 ID 제공업체(IdP) 응답에서 페더레이션 사용자의 ACR 및 AMR 값을 도출하고 `AcrMapping` 필드와 매핑합니다. OIDC 자격 증명 공급자는 섹션을 참조하세요페더레이션 사용자를 위한 단계적 인증.

단계적 인증을 요청하려면 Essentials 또는 Plus 기능 계획이 필요합니다. 자세한 내용은 기능 계획 요구 사항 섹션을 참조하세요.

주제

- 인증 수준 및 방법
- ACR 레벨 이름 구성
- 기능 계획 요구 사항
- 단계적 인증에 대한 책임

인증 수준 및 방법

Amazon Cognito는 사용자가 완료하는 인증 요소를 평가하고 결과를 `acr` 클레임의 인증 수준 및 `amr` 클레임의 인증 방법 목록으로 표시합니다.

ACR 인증 수준

Amazon Cognito는 네 가지 고정 인증 수준을 정의합니다. 각 레벨을 충족하는 인증 요소 조합은 고정되며 변경할 수 없습니다. 사용자 풀에 대한 각 수준의 URI 이름만 사용자 지정할 수 있습니다. 자세한 내용은 ACR 레벨 이름 구성 섹션을 참조하세요.

Amazon Cognito는 사용자가 완료한 요소가 완전히 충족하는 가장 높은 인증 수준을 할당합니다.

ACR 수준 및 충족 인자 조합

ACR 수준	기본 ACR 값	만족스러운 팩터 조합
1	`urn:cognito:loa:1`	암호만
2	`urn:cognito:loa:2`	SMS 또는 이메일로 전달되는 단일 일회용 암호입니다. 레벨 2에는 패스키가 포함되지 않음
3	`urn:cognito:loa:3`	자체 패스키(플랫폼 또는 로밍) 또는 SMS 일회용 코드가 있는 암호 또는 이메일 일회용 코드가 있는 암호

ACR 수준	기본 ACR 값	만족스러운 팩터 조합
4	`urn:cognito:loa:4`	인증 앱의 시간 기반 일회용 암호(TOTP)가 포함된 암호

레벨 1은 암호 전용 인증을 나타냅니다. 페더레이션은이 테이블을 통해 레벨을 충족하지 않습니다. 대신 Amazon Cognito는 ID 제공업체(IdP)의 응답에서 페더레이션 사용자의 인증 수준을 도출합니다. 자세한 내용은 페더레이션 사용자를 위한 단계적 인증 섹션을 참조하세요.

AMR 값

`amr` 클레임은 문자열 목록입니다. 사용자가 완료하는 각 인증 요소는 해당 값을 목록에 추가합니다. Amazon Cognito는 다음 값을 추가할 수 있습니다.

AMR 값

인증 방법	AMR 값	참고
암호	`pwd`	보안 원격 암호(SRP)를 포함한 암호 패밀리 로그인
인증 앱의 TOTP	`otp`	인증 앱과 같은 앱의 시간 기반 일회용 암호
SMS 일회용 코드	`sms`	SMS로 전달되는 일회용 코드
이메일 일회용 코드	`email_otp`	이메일로 전달되는 일회용 코드
플랫폼 인증자가 있는 패스키	`swk`	디바이스 생체 인식으로 보호되는 패스키와 같은 소프트웨어 바인딩 키
로밍 인증자가 있는 패스키	`hwk`	보안 키와 같은 하드웨어 바운드 키
멀티 팩터 조합	`mfa`	사용자가 두 개 이상의 개별 요소를 완료할 때 추가됨
페더레이션 로그인	`fed`	사용자가 ACR 및 AMR을 지원하지 않는 IdP를 통해 페더레이션하거나 Amazon Cognito가 인식하는 값을 반환하지 않을 때 Amazon Cognito가 추가하는 기본 자리 표시자

ACR 및 AMR 토큰 클레임

Amazon Cognito는 액세스 토큰 및 ID 토큰에 `acr` 및 `amr` 클레임을 추가합니다. 애플리케이션은 액세스 토큰 또는 ID 토큰의 `acr` 및 `amr` 클레임을 검사하여 인증 수준을 평가합니다. 사용자가 토큰을 새로 고치면 Amazon Cognito는 새 액세스 `acr` 및 ID 토큰에 대해 동일한 및 `amr` 값을 유지하므로 새로 고칠 때마다 다시 시작하라는 메시지가 사용자에게 표시되지 않습니다. 토큰 콘텐츠에 대한 자세한 내용은 ID 토큰 사용 및 액세스 토큰 사용을 참조하세요.

다음 예제는 사용자가 암호와 TOTP로 인증한 후의 ID 토큰을 보여줍니다.

```
{
  "sub": "a1b2c3d4-5678-90ab-cdef-1234567890ab",
  "iss": "https://cognito-idp.us-east-1.amazonaws.com/us-east-1_ExAmPlE",
  "aud": "abc123def456",
  "exp": 1759348800,
  "iat": 1759345200,
  "auth_time": 1759345200,
  "token_use": "id",
  "acr": "urn:cognito:loa:4",
  "amr": ["pwd", "otp", "mfa"],
  "email": "user@example.com",
  "cognito:username": "a1b2c3d4-5678-90ab-cdef-1234567890ab"
}
```

사용자가 토큰을 새로 고치면 Amazon Cognito는 새 액세스 `acr` 및 ID 토큰에 동일한 및 `amr` 값을 설정합니다. 값은 수정 없이 이월됩니다. 이 동작은 사용자가 새로 고칠 때마다 다시 시작하라는 요청을 받지 않기 때문에 더 원활한 경험을 제공합니다.

토큰을 새로 고쳐도 `auth_time` 클레임이 재설정되지 않습니다. 일정 기간 후에 사용자가 다시 인증하도록 하려면 사용자가 토큰을 새로 고친 횟수에 `auth_time` 관계없이 현재 시간을와 비교하는 `max_age` 파라미터를 사용합니다.

사용자가 단계적 인증을 완료하면 Amazon Cognito는 업데이트된 `acr` 및 `amr` 클레임과 새로운 유효 기간이 포함된 새 액세스, ID 및 새로 고침 토큰을 발급합니다.

ACR 레벨 이름 구성

`AcrConfiguration` 필드를 사용하여 사용자 풀에 대한 각 인증 수준의 URI 이름을 사용자 지정할 수 있습니다. CreateUserPool 또는 UpdateUserPool 작업을 사용하여 사용자 풀을 생성하거나 업데이트할 때이 필드를 구성합니다. 각 레벨을 충족하는 팩터 조합은 고정 상태로 유지됩니다. 이름만 사용자 지정합니다.

AcrConfiguration 필드를 사용하여 사용자 지정 ACR 수준 이름을 구성하려면 Essentials 또는 Plus 기능 계획이 필요합니다. 자세한 내용은 기능 계획 요구 사항 섹션을 참조하세요.

AcrConfiguration 필드는를 Level1 통한 키가 있는 맵입니다Level4. 각 값은 단일 AcrValue 문자열이 있는 객체입니다. 사용자 지정하려는 수준만 재정의할 수 있습니다. 다음 예제 요청은 레벨 2 및 3의 이름을 사용자 지정합니다.

```
{
  "AcrConfiguration": {
    "Level2": { "AcrValue": "urn:my-standard:loa:2" },
    "Level3": { "AcrValue": "urn:my-standard:loa:3" }
  }
}
```

응답은 항상 유효 구성을 반환하며, 기본 이름은 사용자 지정하지 않은 모든 수준에 대해에 병합됩니다.

```
{
  "AcrConfiguration": {
    "Level1": { "AcrValue": "urn:cognito:loa:1" },
    "Level2": { "AcrValue": "urn:my-standard:loa:2" },
    "Level3": { "AcrValue": "urn:my-standard:loa:3" },
    "Level4": { "AcrValue": "urn:cognito:loa:4" }
  }
}
```

다음 검증 규칙은 각에 적용됩니다AcrValue.

- 각는 네 가지 수준 모두에서 고유해야 AcrValue 합니다. 여기에는 사용자 지정하지 않은 수준에 대해 입력되는 기본 이름이 포함됩니다. 사용자 지정되지 않은 레벨의 기본값과 충돌하는 값은 거부됩니다.
- 각 AcrValue는 1~64자의 문자열입니다.
- 각 에는 공백, 큰따옴표 또는 백슬래시를 포함할 AcrValue 수 없습니다.

ACR 레벨 이름을 변경하면 나중에 사용자가 acr 값이 더 이상 구성된 레벨과 일치하지 않는 액세스 토큰을 제공할 수 있습니다. Amazon Cognito는 오래된 값을 존중하지 않고 인식할 수 없는 수준으로 취급하지만 오류가 아닙니다. Amazon Cognito는 여전히 토큰의 amr 값을 준수하며 특정 수준에 크레딧을 부여하지 않고 amr 클레임에 아직 나열되지 않은 요소에 대해서만 프롬프트를 표시합니다.

기능 계획 요구 사항

다음 기능에는 Essentials 또는 Plus 기능 계획이 필요합니다.

- 단계적 인증 요청. 이는의 TARGET_ACR_VALUES 파라미터InitiateAuth와 관리형 로그인의 acr_values 파라미터에 적용됩니다.
- CreateUserPool 또는의 AcrConfiguration 필드를 사용하여 사용자 지정 ACR 수준 이름을 구성합니다UpdateUserPool.

기능 부족 계획에 대한 단계적 인증을 요청할 때의 동작은 진입점에 따라 다릅니다.

- InitiateAuth API는를 반환합니다FeatureUnavailableInTierException.
- 관리형 로그인은 엔드포인트를 자동으로 무시acr_values하고 오류 없이 일반 인증을 진행합니다.

기능 계획에 대한 자세한 내용은 섹션을 참조하세요사용자 풀 기능 플랜.

단계적 인증에 대한 책임

Amazon Cognito는 사용자가 달성한 인증 수준으로 토큰을 발급합니다. 애플리케이션은 다음을 담당합니다.

- 리디렉션하기 전에 사용자의 현재 ID 토큰을 확인합니다. 애플리케이션이 이미 보유한 토큰max_age을 사용하여 작업에 필요한 acr 수준에 대해 auth_time , ID 토큰의 수준을 평가합니다. 필요한 수준 또는 최신성이 아직 충족되지 않은 경우에만 사용자를 Amazon Cognito로 리디렉션하여 단계적으로 진행합니다.
- 올바른 TARGET_ACR_VALUES 또는 acr_values, 올바른 MAX_AGE 또는를 설정합니다max_age. Amazon Cognito는 사용자가 제공한 내용에 대해서만 작동합니다. 의도한 수준이나 신선도를 유추하지 않습니다. 잘못된 레벨을 요청하거나 파라미터를 생략하면 Amazon Cognito는 의도한 대로 사용자를 단계적으로 늘리지 않습니다.
- 애플리케이션에서 acr 클레임을 검증합니다. Amazon Cognito가 토큰을 반환한 후 애플리케이션은 액세스 권한을 부여하기 전에 acr 클레임이 작업에 필요한 수준과 일치하는지 확인해야 합니다. Amazon Cognito는 달성된 수준으로 토큰을 발급합니다. 작업에 지정된 수준을 요구하는 것은 애플리케이션의 책임입니다.
- Amazon Cognito 레벨을 토큰 외부의 다른 표준으로 변환합니다. Amazon Cognito는 인증 흐름에서 acr 및 amr 클레임을 계산합니다. 사전 토큰 생성 Lambda 트리거 또는 기타 Lambda 트리거를 사용

하여 이러한 클레임을 사용자 지정, 재정의, 추가 또는 억제할 수 없습니다. Amazon Cognito 수준을 NIST 또는 eIDAS와 같은 다른 표준으로 변환해야 하는 경우 애플리케이션 계층에서 변환합니다.

사용자 풀의 사용자 디바이스 작업

Amazon Cognito 사용자 풀 API로 로컬 사용자 풀 사용자를 로그인하면 위협 방지의 사용자 활동 로그를 각 디바이스와 연결할 수 있으며, 선택적으로 사용자가 신뢰할 수 있는 디바이스를 사용하는 경우 다중 인증(MFA)을 건너뛰도록 허용할 수 있습니다. Amazon Cognito는 디바이스 정보가 아직 포함되어 있지 않은 모든 로그인에 대한 응답에 디바이스 키를 포함시킵니다. 디바이스 키의 형식은 *Region*_*UUID*입니다. 디바이스 키, SRP(Secure Remote Password) 라이브러리, 디바이스 인증을 허용하는 사용자 풀을 사용하면, 앱 사용자에게 현재 디바이스를 신뢰하라는 메시지를 표시하고 로그인 시 MFA 코드를 묻는 메시지를 더 이상 표시하지 않을 수 있습니다.

주제

- 기억된 디바이스 설정
- 디바이스 키 가져오기
- 디바이스로 로그인
- 디바이스 보기, 업데이트 및 지우기

기억된 디바이스 설정

Amazon Cognito 사용자 풀을 사용하면 각 사용자의 디바이스를 고유한 디바이스 식별자인 디바이스 키와 연결할 수 있습니다. 로그인 시 디바이스 키를 제시하고 디바이스 인증을 수행할 때 신뢰할 수 있는 디바이스 인증 흐름으로 애플리케이션을 구성할 수 있습니다. 이 흐름에서 애플리케이션은 앱의 보안 요구 사항 또는 사용자의 기본 설정에 따라 나중에 MFA 없이 로그인할 수 있는 선택 사항을 사용자에게 제공할 수 있습니다. 해당 기간이 끝나면 애플리케이션은 디바이스 상태를 기억하지 않음으로 변경하고 사용자는 디바이스 기억을 확인할 때까지 MFA로 로그인해야 합니다. 예를 들어, 애플리케이션은 사용자에게 30일, 60일 또는 90일 동안 디바이스를 신뢰하라는 메시지를 표시할 수 있습니다. 이 날짜를 사용자 지정 속성에 저장할 수 있으며 해당 날짜에 디바이스의 기억 상태를 변경할 수 있습니다. 그런 다음 사용자에게 MFA 코드를 제출하도록 다시 프롬프트하고 인증에 성공한 후 디바이스를 다시 기억하도록 설정해야 합니다.

1. 기억된 디바이스는 MFA가 활성화된 사용자 풀에서만 MFA를 재정의할 수 있습니다.

사용자가 기억된 디바이스로 로그인하면 인증 흐름에서 추가 디바이스 인증을 수행해야 합니다. 자세한 내용은 디바이스로 로그인 단원을 참조하십시오.

디바이스 추적 내 사용자 풀의 로그인 메뉴에서 사용자 풀이 디바이스를 기억하도록 구성합니다. Amazon Cognito 콘솔을 통해 기억된 디바이스 기능을 설정할 때, [항상(Always)], [사용자 옵트인(User Opt-In)] 및 [아니요(No)]라는 세 가지 옵션이 제공됩니다.

기억하지 않음

사용자 풀이 사용자가 로그인할 때 디바이스를 기억하라는 메시지를 표시하지 않습니다.

항상 기억

앱이 사용자의 디바이스를 확인하면 사용자 풀이 항상 디바이스를 기억하고 향후 디바이스 로그인에 성공할 때 MFA 챌린지를 반환하지 않습니다.

사용자 옵트인

앱이 사용자의 디바이스를 확인하더라도 사용자 풀이 MFA 챌린지를 자동으로 억제하지 않습니다. 사용자에게 디바이스를 기억할지 여부를 선택하라는 메시지를 표시해야 합니다.

항상 기억 또는 사용자 옵트인을 선택하는 경우, Amazon Cognito는 사용자가 미확인 디바이스에서 로그인할 때마다 디바이스 식별자 키와 보안 암호를 생성합니다. 디바이스 키는 사용자가 디바이스 인증을 수행할 때 앱이 사용자 풀에 보내는 초기 식별자입니다.

자동으로 기억되든 옵트인되든 관계없이 확인된 각 사용자 디바이스에서 디바이스 식별자 키와 보안 암호를 사용하여 사용자가 로그인할 때마다 디바이스를 인증할 수 있습니다.

또한 CreateUserPool 또는 UpdateUserPool API 요청에서 사용자 풀의 기억된 디바이스 설정을 구성할 수 있습니다. 자세한 내용은 DeviceConfiguration 속성을 참조하세요.

Amazon Cognito 사용자 풀 API에는 기억된 디바이스에 대한 추가 작업이 있습니다.

1. ListDevices 및 AdminListDevices는 사용자에 대한 디바이스 키 및 해당 메타데이터 목록을 반환합니다.
2. GetDevice 및 AdminGetDevice는 단일 디바이스에 대한 디바이스 키와 메타데이터를 반환합니다.
3. UpdateDeviceStatus 및 AdminUpdateDeviceStatus는 사용자의 디바이스를 기억됨 또는 기억되지 않음으로 설정합니다.
4. ForgetDevice 및 AdminForgetDevice는 프로필에서 사용자의 확인된 디바이스를 제거합니다.

이름이 Admin으로 시작하는 API 작업은 서버측 앱에서 사용하기 위한 것이며 IAM 보안 인증 정보로 승인되어야 합니다. 자세한 내용은 API, OIDC 및 관리형 로그인 페이지 인증 이해 단원을 참조하십시오.

디바이스 키 가져오기

사용자가 사용자 풀 API를 사용하여 로그인하고 인증 파라미터에 DEVICE_KEY로 디바이스 키를 포함하지 않을 때마다 Amazon Cognito는 응답으로 새 디바이스 키를 반환합니다. 퍼블릭 클라이언트측 앱에서는 향후 요청에 포함할 수 있도록 앱 스토리지에 디바이스 키를 저장합니다. 기밀 서버측 앱에서는 사용자의 디바이스 키로 브라우저 쿠키 또는 다른 클라이언트측 토큰을 설정합니다.

사용자가 신뢰할 수 있는 디바이스로 로그인하려면 먼저 앱에서 디바이스 키를 확인하고 추가 정보를 제공해야 합니다. Amazon Cognito에 ConfirmDevice 요청을 생성하여 디바이스 키, 친숙한 이름, 암호 확인 도구 및 솔트를 사용해 사용자의 디바이스를 확인합니다. 사용자 풀에 옵트인 디바이스 인증을 구성한 경우, Amazon Cognito는 ConfirmDevice 요청에 응답하여 사용자가 현재 디바이스를 기억할지 여부를 선택해야 한다는 메시지를 표시합니다. UpdateDeviceStatus 요청에서 사용자의 선택 사항으로 응답합니다.

사용자의 디바이스를 확인하지만 기억하도록 설정하지 않은 경우 Amazon Cognito는 연결을 저장하지만 디바이스 키를 제공할 때 디바이스 이외 로그인을 진행합니다. 디바이스는 사용자 보안 및 문제 해결에 유용한 로그를 생성할 수 있습니다. 확인되었지만 기억되지 않은 디바이스는 로그인 기능을 활용하지 않지만 보안 모니터링 로그 기능은 활용합니다. 앱 클라이언트에 대한 위협 방지 기능을 활성화하고 요청에 디바이스 지문을 인코딩하면 Amazon Cognito가 사용자 이벤트를 확인된 디바이스와 연결합니다.

새 디바이스 키를 얻으려면

1. InitiateAuth API 요청으로 사용자의 로그인 세션을 시작합니다.
2. 사용자의 로그인 세션이 완료되었음을 표시하는 JSON 웹 토큰(JWT)을 받을 때까지 RespondToAuthChallenge를 사용하여 모든 인증 챌린지에 응답합니다.
3. 앱에서 Amazon Cognito가 RespondToAuthChallenge 또는 InitiateAuth 응답의 NewDeviceMetadata에 반환하는 값(DeviceGroupKey 및 DeviceKey)을 기록합니다.
4. 사용자를 위한 새 SRP 보안 암호(솔트 및 암호 검증자)를 생성합니다. 이 기능은 SRP 라이브러리를 제공하는 SDK에서 사용할 수 있습니다.
5. 사용자에게 디바이스 이름을 입력하라는 메시지를 표시하거나 사용자의 디바이스 특성에서 이름을 생성합니다.
6. ConfirmDevice API 요청에서 사용자의 액세스 토큰, 디바이스 키, 디바이스 이름 및 SRP 보안 암호를 제공합니다. 사용자 풀이 디바이스를 항상 기억하도록 설정된 경우 사용자 등록이 완료됩니다.

7. Amazon Cognito가 "UserConfirmationNecessary": true로 ConfirmDevice에 응답한 경우, 사용자에게 디바이스를 기억할지 선택하라는 메시지를 표시합니다. 사용자가 디바이스를 기억하겠다고 확인하면 사용자의 액세스 토큰, 디바이스 키, "DeviceRememberedStatus": "remembered" 등을 사용하여 UpdateDeviceStatus API 요청을 생성합니다.
8. Amazon Cognito에 디바이스를 기억하도록 지시한 경우, 다음에 로그인할 때 MFA 챌린지 대신 DEVICE_SRP_AUTH 챌린지가 표시됩니다.

디바이스로 로그인

사용자의 디바이스를 기억하도록 구성한 후에는 사용자가 동일한 디바이스 키로 로그인할 때 Amazon Cognito에서 더 이상 MFA 코드 제출을 요구하지 않습니다. 디바이스 인증은 MFA 인증 챌린지를 디바이스 인증 챌린지로 대체할 뿐입니다. 디바이스 인증만으로는 사용자가 로그인될 수 없습니다. 사용자는 먼저 암호 또는 사용자 지정 챌린지로 인증을 완료해야 합니다. 기억된 디바이스에서 사용자를 인증하는 프로세스는 다음과 같습니다.

사용자 지정 인증 챌린지 Lambda 트리거를 사용하는 흐름에서 디바이스 인증을 수행하려면 InitiateAuth API 요청에 DEVICE_KEY 파라미터를 전달합니다. 사용자가 모든 챌린지에 성공하고 CUSTOM_CHALLENGE 챌린지가 true인 issueTokens 값을 반환하면 Amazon Cognito는 하나의 최종 DEVICE_SRP_AUTH 챌린지를 반환합니다.

디바이스로 로그인하려면

1. 클라이언트 스토리지에서 사용자의 디바이스 키를 검색합니다.
2. InitiateAuth API 요청으로 사용자의 로그인 세션을 시작합니다. USER_SRP_AUTH, REFRESH_TOKEN_AUTH, USER_PASSWORD_AUTH 또는 CUSTOM_AUTH의 AuthFlow를 선택합니다. AuthParameters에서 사용자의 디바이스 키를 DEVICE_KEY 파라미터에 추가하고 선택한 로그인 흐름에 필요한 기타 파라미터를 포함합니다.
   a. 인증 챌린지에 대한 PASSWORD_VERIFIER 응답의 파라미터에 DEVICE_KEY를 전달할 수도 있습니다.
3. 응답으로 DEVICE_SRP_AUTH 챌린지를 받을 때까지 챌린지 응답을 완료합니다.
4. RespondToAuthChallenge API 요청에서 DEVICE_SRP_AUTH의 ChallengeName과 USERNAME, DEVICE_KEY 및 SRP_A에 대한 파라미터를 전송합니다.
5. Amazon Cognito가 DEVICE_PASSWORD_VERIFIER 챌린지로 응답합니다. 이 챌린지 응답에는 SECRET_BLOCK 및 SRP_B 값이 포함됩니다.

6. SRP 라이브러리를 사용하여 PASSWORD_CLAIM_SIGNATURE, PASSWORD_CLAIM_SECRET_BLOCK, TIMESTAMP, USERNAME 및 DEVICE_KEY 파라미터를 생성하고 제출합니다. 추가 RespondToAuthChallenge 요청으로 제출해야 합니다.
7. 사용자의 JWT를 받을 때까지 추가 챌린지를 완료합니다.

다음 유사 코드는 DEVICE_PASSWORD_VERIFIER 챌린지 응답의 값을 계산하는 방법을 보여줍니다. 디바이스를 사용한 SRP 인증의 경우 새로운 고엔트로피 암호 DeviceSecret, 솔트 및 관련 암호 확인자 등 사용자를 위한 새 SRP 암호를 생성합니다. 이러한 값은 사용자의 SRP 인증에 사용되는 암호, 솔트 및 확인자와 다릅니다. 디바이스 인증에만 사용되며 디바이스에만 저장됩니다. 사용자의 디바이스에 대한 SRP 보안 암호를 생성하는 함수는 다양한 SDK에서 사용할 수 있는 SRP 라이브러리에서 사용할 수 있습니다.

```
PASSWORD_CLAIM_SECRET_BLOCK = SECRET_BLOCK
TIMESTAMP = "Tue May 7 00:09:40 UTC 2025"
k = SHA256(N || g) as a non-negative integer in big-endian
u = SHA256(SRP_A || SRP_B) as a non-negative integer in big-endian
x = SHA256(salt || SHA256(DeviceGroupKey || DeviceKey || ":" || DeviceSecret)) as a
 non-negative integer in big-endian
S_USER = (SRP_B - k * g^x)^(a + u * x) % N
K_USER = HKDF_HMAC_SHA256(salt=u, ikm=S_USER, info="Caldera Derived Key", length=16
 bytes)
PASSWORD_CLAIM_SIGNATURE = Base64(HMAC_SHA256(key=K_USER, message=(DeviceGroupKey ||
 DeviceKey || PASSWORD_CLAIM_SECRET_BLOCK || TIMESTAMP)))
```

디바이스 보기, 업데이트 및 지우기

Amazon Cognito API를 사용하여 앱에서 다음 기능을 구현할 수 있습니다.

1. 사용자의 현재 디바이스에 대한 정보를 표시합니다.
2. 모든 사용자 디바이스 목록을 표시합니다.
3. 디바이스를 지웁니다.
4. 디바이스 기억 상태를 업데이트합니다.

다음 설명의 API 요청을 승인하는 액세스 토큰에는 aws.cognito.signin.user.admin 범위가 포함되어야 합니다. Amazon Cognito는 Amazon Cognito 사용자 풀 API로 생성하는 모든 액세스 토큰에 이 범위에 대한 클레임을 추가합니다. 서드 파티 IdP는 Amazon Cognito에 인증하는 사용자의 디바이스와 MFA를 별도로 관리해야 합니다. 관리형 로그인에서는 aws.cognito.signin.user.admin

범위를 요청할 수 있지만, 관리형 로그인은 자동으로 디바이스 정보를 고급 보안 사용자 로그에 추가하며, 디바이스를 기억하는 기능을 제공하지 않습니다.

디바이스에 대한 정보 표시

사용자 디바이스에 대한 정보를 쿼리하여 디바이스가 현재 사용 중인지 확인할 수 있습니다. 예를 들어 90일 동안 로그인하지 않은 기억된 디바이스를 비활성화할 수 있습니다.

- 퍼블릭 클라이언트 앱에 사용자의 디바이스 정보를 표시하려면 GetDevice API 요청에서 사용자의 액세스 키와 디바이스 키를 제출합니다.
- 기밀 클라이언트 앱에 사용자의 디바이스 정보를 표시하려면 자격 AWS 증명으로 AdminGetDevice API 요청에 서명하고 사용자의 사용자 이름, 디바이스 키 및 사용자 풀을 제출합니다.

모든 사용자 디바이스 목록 표시

모든 사용자 디바이스 및 속성 목록을 표시할 수 있습니다. 예를 들어 현재 디바이스가 기억된 디바이스와 일치하는지 확인할 수 있습니다.

- 퍼블릭 클라이언트 앱에서는 ListDevices API 요청으로 사용자의 액세스 토큰을 제출합니다.
- 기밀-클라이언트 앱에서 자격 AWS 증명으로 AdminListDevices API 요청에 서명하고 사용자의 사용자 이름과 사용자 풀을 제출합니다.

디바이스 지우기

사용자의 디바이스 키를 삭제할 수 있습니다. 사용자가 더 이상 디바이스를 사용하지 않는 것으로 확인되거나 비정상적인 활동을 탐지하여 사용자에게 MFA를 다시 완료하도록 요구하려는 경우 이 방법을 사용하는 것이 좋습니다. 나중에 디바이스를 다시 등록하려면 새 디바이스 키를 생성하여 저장해야 합니다.

- 퍼블릭 클라이언트 앱에서는 ForgetDevice API 요청으로 사용자의 디바이스 키와 액세스 토큰을 제출합니다.
- 기밀 클라이언트 앱에서는 AdminForgetDevice API 요청으로 사용자의 디바이스 키와 액세스 토큰을 제출합니다.

위협 방지 기능이 있는 고급 보안

사용자 풀을 생성한 후에는 Amazon Cognito 콘솔의 탐색 메뉴에서 위협 방지에 액세스할 수 있습니다. 위협 방지 기능을 켜고 다양한 위험에 대응하여 취해지는 조치를 사용자 지정할 수 있습니다. 또는 감

사 모드를 사용하여 보안 완화를 적용하지 않고 탐지된 위험에 대한 지표를 수집할 수 있습니다. 감사 모드에서 위협 방지는 Amazon CloudWatch에 지표를 게시합니다. Amazon Cognito가 첫 번째 이벤트를 생성한 후에 지표를 볼 수 있습니다. 위협 방지 지표 보기 섹션을 참조하세요.

위협 방지는 이전에 고급 보안 기능으로 불렸으며, 사용자 풀에서 원치 않는 활동에 대한 모니터링 도구 세트이며 잠재적으로 악의적인 활동을 자동으로 종료하는 구성 도구입니다. 위협 방지에는 표준 인증 작업과 사용자 지정 인증 작업에 대한 다양한 구성 옵션이 있습니다. 예를 들어, 추가 보안 요소를 설정했지만 기본 사용자 이름-암호 인증으로 동일한 위험 수준의 사용자를 차단한 의심스러운 사용자 지정 인증 로그인이 있는 사용자에게 알림을 보낼 수 있습니다.

위협 방지는 Plus 기능 플랜에서 사용할 수 있습니다. 자세한 내용은 사용자 풀 기능 플랜 섹션을 참조하세요.

다음 사용자 풀 옵션은 위협 방지의 구성 요소입니다.

손상된 보안 인증

사용자는 여러 사용자 계정에 암호를 재사용합니다. Amazon Cognito의 손상된 자격 증명 기능은 사용자 이름과 암호의 공개 유출로부터 데이터를 컴파일하고 사용자의 자격 증명을 유출된 자격 증명 목록과 비교합니다. 손상된 자격 증명 탐지는 일반적으로 추측되는 암호도 확인합니다. 사용자 풀의 사용자 이름 및 암호 표준 인증 흐름에서 손상된 자격 증명을 확인할 수 있습니다. Amazon Cognito는 보안 원격 암호(SRP) 또는 사용자 지정 인증에서 손상된 자격 증명을 감지하지 않습니다.

손상된 보안 인증을 확인하도록 요청하는 사용자 작업과 이에 대한 응답으로 Amazon Cognito가 수행하기를 원하는 작업을 선택할 수 있습니다. 로그인, 가입, 암호 변경 이벤트의 경우, Amazon Cognito는 로그인을 차단하거나 로그인을 허용할 수 있습니다. 두 경우 모두 Amazon Cognito가 생성하는 사용자 활동 로그에서 이벤트에 대한 추가 정보를 찾을 수 있습니다.

자세히 알아보기

손상된 자격 증명 감지 작업

조정 인증

Amazon Cognito는 사용자의 로그인 요청에서 위치 및 디바이스 정보를 검토하고 자동 응답을 적용하여 의심스러운 활동으로부터 사용자 풀의 사용자 계정을 보호할 수 있습니다. 사용자 활동을 모니터링하고 사용자 이름-암호 및 SRP의 감지된 위험 수준에 대한 응답과 사용자 지정 인증을 자동화할 수 있습니다.

위협 방지를 활성화하면 Amazon Cognito가 사용자 활동에 위험 점수를 할당합니다. 의심스러운 활동에 자동 응답을 할당할 수 있습니다. MFA를 요구하거나, 로그인을 차단하거나, 활동 세부 정보

및 위험 점수를 로깅할 수 있습니다. 또한 의심스러운 활동을 사용자에게 알리는 이메일 메시지를 자동으로 전송하여 사용자가 암호를 재설정하거나 기타 자체 안내 조치를 취할 수 있습니다.

자세히 알아보기

적응형 인증 작업

IP 주소 허용 목록 및 거부 목록

전체 기능 모드의 Amazon Cognito 위협 방지를 사용하면 IP 주소 항상 차단 및 항상 허용 예외를 생성할 수 있습니다. 항상 차단(Always block) 예외 목록에 있는 IP 주소의 세션에는 조정 인증별 위험 수준이 할당되지 않으며 이러한 세션은 사용자 풀에 로그인할 수 없습니다.

IP 주소 허용 목록 및 차단 목록에 대해 알아야 할 사항

- 항상 차단 및 항상 허용은 CIDR 형식으로 표현해야 합니다. 예를 들어 `192.0.2.0/24`는 24비트 마스크, `192.0.2.252/32`는 단일 IP 주소입니다.
- IP 주소가 항상 차단 IP 범위에 있는 디바이스는 SDK 기반 또는 관리형 로그인 애플리케이션으로 가입하거나 로그인할 수 없지만 타사 IdP로 로그인할 수 있습니다.
- 항상 허용 및 항상 차단 목록은 토큰 새로 고침에 영향을 주지 않습니다.
- Amazon Cognito는 항상 허용 IP 범위의 디바이스에 적응형 인증 MFA 규칙을 적용하지 않지만 손상된 자격 증명 규칙을 적용합니다.

로그 내보내기

위협 방지는 사용자 풀에 대한 사용자 인증 요청의 세부 정보를 세부적으로 기록합니다. 이러한 로그에는 위협 평가, 사용자 정보, 위치 및 디바이스와 같은 세션 메타데이터가 포함됩니다. 보존 및 분석을 위해 이러한 로그의 외부 아카이브를 생성할 수 있습니다. Amazon Cognito 사용자 풀은 위협 방지 로그를 Amazon S3, CloudWatch Logs 및 Amazon Data Firehose로 내보냅니다. 자세한 내용은 사용자 이벤트 기록 보기 및 내보내기 섹션을 참조하세요.

자세히 알아보기

위협 방지 사용자 활동 로그 내보내기

주제

- 위협 방지를 위한 고려 사항 및 제한 사항
- 사용자 풀에서 위협 방지 켜기
- 위협 방지 시행 개념

- 표준 인증 및 사용자 지정 인증을 위한 위협 방지
- 위협 방지 전제 조건
- 위협 방지 설정
- 손상된 자격 증명 감지 작업
- 적응형 인증 작업
- 애플리케이션에서 위협 방지를 위한 데이터 수집

위협 방지를 위한 고려 사항 및 제한 사항

위협 방지 옵션은 인증 흐름마다 다릅니다.

Amazon Cognito는 인증 흐름 `USER_PASSWORD_AUTH` 및 `ADMIN_USER_PASSWORD_AUTH`를 통해 적응형 인증과 손상된 자격 증명 탐지를 모두 지원합니다. `USER_SRP_AUTH`에 대한 적응형 인증만 활성화할 수 있습니다. 페더레이션 로그인에는 위협 방지를 사용할 수 없습니다.

상시 차단 IP 요청 할당량에 기여합니다.

사용자 풀의 항상 차단(Always block) 예외 목록에 있는 IP 주소의 차단된 요청은 사용자 풀에 대한 요청 비율 할당량에 기여합니다.

위협 방지는 속도 제한을 적용하지 않습니다.

일부 악성 트래픽은 분산 서비스 거부(DDoS) 공격과 같은 대량 요청의 특성을 갖습니다. Amazon Cognito가 들어오는 트래픽에 적용하는 위험 등급은 요청당이며 요청 볼륨을 고려하지 않습니다. 대량 이벤트의 개별 요청은 볼륨 공격에서의 역할과 관련이 없는 애플리케이션 계층 이유로 위험 점수와 자동 응답을 받을 수 있습니다. 사용자 풀에서 볼륨 측정 공격에 대한 방어를 구현하려면 AWS WAF 웹 ACLs 추가합니다. 자세한 내용은 AWS WAF 웹 ACL을 사용자 풀과 연결 섹션을 참조하세요.

위협 방지는 M2M 요청에 영향을 주지 않습니다.

클라이언트 보안 인증 부여는 사용자 계정과 연결되지 않은 M2M(Machine-to-Machine) 인증을 위한 것입니다. 위협 방지는 사용자 풀의 사용자 계정과 암호만 모니터링합니다. M2M 활동으로 보안 기능을 구현하려면 요청 속도 및 콘텐츠를 모니터링하기 AWS WAF 위한의 기능을 고려하세요. 자세한 내용은 AWS WAF 웹 ACL을 사용자 풀과 연결 섹션을 참조하세요.

사용자 풀에서 위협 방지 켜기

Amazon Cognito user pools console

사용자 풀에 대한 위협 방지를 활성화하려면

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 자격 AWS 증명을 입력합니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. 아직 활성화하지 않은 경우 설정 메뉴에서 Plus 기능 플랜을 활성화합니다.
5. 위협 방지 메뉴를 선택하고 활성화를 선택합니다.
6. 변경 사항 저장을 선택합니다.

API

CreateUserPool 또는 UpdateUserPool API 요청에서 기능 플랜을 Plus로 설정합니다. 다음 부분 예제 요청 본문은 위협 방지를 전체 기능 모드로 설정합니다. 전체 예제 요청은 예제 섹션을 참조하세요.

```
"UserPoolAddOns": {
      "AdvancedSecurityMode": "ENFORCED"
   }
```

위협 방지는 계정 탈취 징후에 대해 사용자 작업을 모니터링하고 영향을 받는 사용자 계정을 보호하기 위해 자동으로 대응하는 기능을 통칭하는 용어입니다. 사용자가 표준 및 사용자 지정 인증 흐름으로 로그인할 때 위협 방지 설정을 적용할 수 있습니다.

위협 방지는 사용자의 로그인, 로그아웃 및 기타 활동을 자세히 설명하는 로그를 생성합니다. 이러한 로그를 타사 시스템으로 내보낼 수 있습니다. 자세한 내용은 사용자 이벤트 기록 보기 및 내보내기 섹션을 참조하세요.

위협 방지 시행 개념

위협 방지는 사용자 풀이 사용자 활동을 모니터링하고 위험 수준을 할당하며 로그를 생성하는 감사 전용 모드에서 시작됩니다. 전체 기능 모드를 활성화하기 전에 2주 이상 감사 전용 모드에서 실행하는 것이 가장 좋습니다. 전체 기능 모드에는 감지된 위험한 활동 및 손상된 암호에 대한 자동 대응 세트가 포

함됩니다. 감사 전용 모드를 사용하면 Amazon Cognito가 수행하는 위협 평가를 모니터링할 수 있습니다. 가양성 및 부정 사항에 대해 기능을 훈련하는 피드백을 제공할 수도 있습니다.

사용자 풀 수준의 위협 방지 적용은 사용자 풀의 모든 앱 클라이언트와 개별 앱 클라이언트 수준에서 구성할 수 있습니다. 앱 클라이언트 위협 방지 구성은 사용자 풀 구성을 재정의합니다. 앱 클라이언트에 대한 위협 방지를 구성하려면 Amazon Cognito 콘솔의 사용자 풀에 있는 앱 클라이언트 메뉴에서 앱 클라이언트 설정으로 이동합니다. 여기서 클라이언트 수준 설정을 사용하고 앱 클라이언트 전용으로 적용을 구성할 수 있습니다.

또한 표준 인증 유형과 사용자 지정 인증 유형에 대해 위협 방지를 별도로 구성할 수 있습니다.

표준 인증 및 사용자 지정 인증을 위한 위협 방지

위협 방지를 구성할 수 있는 방법은 사용자 풀 및 앱 클라이언트에서 수행하는 인증 유형에 따라 다릅니다. 다음 각 유형의 인증에는 자체 적용 모드와 자동 응답이 있을 수 있습니다.

표준 인증

표준 인증은 관리형 로그인에서 사용자 이름-암호 흐름을 사용한 사용자 로그인, 로그아웃 및 암호 관리입니다. Amazon Cognito 위협 방지는 관리형 로그인으로 로그인하거나 다음 API `AuthFlow` 파라미터를 사용할 때 작업에서 위험 지표를 모니터링합니다.

InitiateAuth

`USER_PASSWORD_AUTH`, `USER_SRP_AUTH`. 손상된 자격 증명 기능은 `USER_SRP_AUTH` 로그인에서 암호에 액세스할 수 없으며 이 흐름을 사용하여 이벤트를 모니터링하거나 작업하지 않습니다.

AdminInitiateAuth

`ADMIN_USER_PASSWORD_AUTH`, `USER_SRP_AUTH`. 손상된 자격 증명 기능은 `USER_SRP_AUTH` 로그인에서 암호에 액세스할 수 없으며 이 흐름을 사용하여 이벤트를 모니터링하거나 작업하지 않습니다.

표준 인증의 적용 모드를 감사 전용 또는 전체 함수로 설정할 수 있습니다. 표준 인증에 대한 위협 모니터링을 비활성화하려면 위협 방지를 적용 없음으로 설정합니다.

사용자 지정 인증

사용자 지정 인증은 사용자 지정 챌린지 Lambda 트리거를 사용한 사용자 로그인입니다. 관리형 로그인에서는 사용자 지정 인증을 수행할 수 없습니다. Amazon Cognito 위협 방지는 `InitiateAuth` 및 `AdminInitiateAuth`의 API `AuthFlow` 파라미터 `CUSTOM_AUTH`를 사용하여 로그인할 때 위험 지표에 대한 작업을 모니터링합니다.

사용자 지정 인증의 적용 모드를 감사 전용, 전체 함수 또는 적용 없음으로 설정할 수 있습니다. 적용 없음 옵션은 다른 위협 방지 기능에 영향을 주지 않고 사용자 지정 인증에 대한 위협 모니터링을 비활성화합니다.

위협 방지 전제 조건

시작하려면 다음이 필요합니다.

- 앱 클라이언트가 포함된 사용자 풀. 자세한 내용은 Amazon Cognito 콘솔에서 새 애플리케이션 생성 섹션을 참조하세요.
- Amazon Cognito 콘솔에서 멀티 팩터 인증(MFA)을 [선택 사항(Optional)]으로 설정하여 위험 기반 조정 인증 기능을 사용합니다. 자세한 내용은 사용자 풀에 MFA 추가 섹션을 참조하세요.
- 이메일 알림을 사용하고 있는 경우에는 Amazon SES 콘솔로 이동하여 이메일 알림에서 사용할 이메일 주소 또는 도메인을 구성하고 확인합니다. Amazon SES에 대한 자세한 내용은 Amazon SES에서 자격 증명 확인을 참조하세요.

위협 방지 설정

다음 지침에 따라 사용자 풀 위협 방지를 설정합니다.

> **Note**
>
> Amazon Cognito 사용자 풀 콘솔에서 앱 클라이언트에 대해 다른 위협 방지 구성을 설정하려면 앱 클라이언트 메뉴에서 앱 클라이언트를 선택하고 클라이언트 수준 설정 사용을 선택합니다.

AWS Management Console

사용자 풀에 대한 위협 방지를 구성하려면

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 자격 AWS 증명을 입력합니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. 위협 방지 메뉴를 선택하고 활성화를 선택합니다.

5. 구성하려는 위협 방지 방법인 표준 및 사용자 지정 인증을 선택합니다. 사용자 지정 및 표준 인증에 대해 다양한 적용 모드를 설정할 수 있지만 전체 함수 모드에서 자동 응답의 구성을 공유합니다.
6. 편집을 선택합니다.
7. 적용 모드를 선택합니다. 감지된 위험에 즉시 대응하려면 전체 함수를 선택하고 손상된 자격 증명 및 적응형 인증에 대한 자동 응답을 구성합니다. 사용자 수준 로그 및 CloudWatch에서 정보를 수집하려면 감사만 선택합니다.

   작업을 활성화하기 전에 2주 동안 위협 방지를 감사 모드로 유지하는 것이 좋습니다. 이 기간 동안 Amazon Cognito는 앱 사용자의 사용 패턴을 학습하고 응답을 조정하기 위한 이벤트 피드백을 제공할 수 있습니다.
8. 감사 전용(Audit only)을 선택한 경우 변경 사항 저장(Save changes)을 선택합니다. [전체 기능(Full function)]을 선택한 경우 다음을 수행합니다.

   a. 사용자 지정 작업을 수행할지 아니면 Cognito 기본값을 사용하여 손상된 것으로 의심되는 자격 증명에 응답할지 선택합니다. [Cognito 기본값(Cognito defaults)]은 다음과 같습니다.

      i. [로그인(Sign-in)], [가입(Sign-up)], 암호 변경(Password change)]에서 손상된 자격 증명을 탐지합니다.

      ii. [로그인 차단(Block sign-in)] 작업으로 손상된 자격 증명에 대응합니다.

   b. 손상된 자격 증명에 대한 사용자 지정 작업을 선택한 경우 Amazon Cognito가 이벤트 감지에 사용하는 사용자 풀 작업과 Amazon Cognito가 수행할 손상된 자격 증명 응답을 선택합니다. 의심스러운 손상된 자격 증명으로 [로그인 차단(Block sign-in)] 또는 [로그인 허용(Allow sign-in)]을 선택할 수 있습니다.

   c. 조정 인증(Adaptive authentication)에서 악의적인 로그인 시도에 대응하는 방법을 선택합니다. 사용자 지정 작업을 수행할지 아니면 Cognito 기본값을 사용하여 의심스러운 악성 활동에 대응할지 선택합니다. Cognito 기본값(Cognito defaults)을 선택한 경우 Amazon Cognito는 모든 위험 수준에서 로그인을 차단하고 사용자에게 알리지 않습니다.

   d. 적응형 인증을 위한 사용자 지정 작업을 선택한 경우 심각도 수준에 따라 탐지된 위험에 대응하여 Amazon Cognito가 수행하는 자동 위험 대응 작업을 선택합니다. Amazon Cognito 위험 수준에 응답을 할당하면 더 높은 위험 수준에 덜 제한적인 응답을 할당할 수 없습니다. 위험 수준에 할당할 수 있는 대응은 다음과 같습니다.

      i. [로그인 허용(Allow sign-in)] - 예방 조치를 수행하지 않습니다.

ii. 선택적 MFA - 사용자에게 MFA가 구성된 경우 Amazon Cognito는 사용자가 로그인할 때 항상 추가 SMS 또는 시간 기반 일회용 암호(TOTP) 팩터를 제공하도록 요구합니다. MFA가 구성되어 있지 않은 사용자는 정상적으로 로그인을 계속할 수 있습니다.

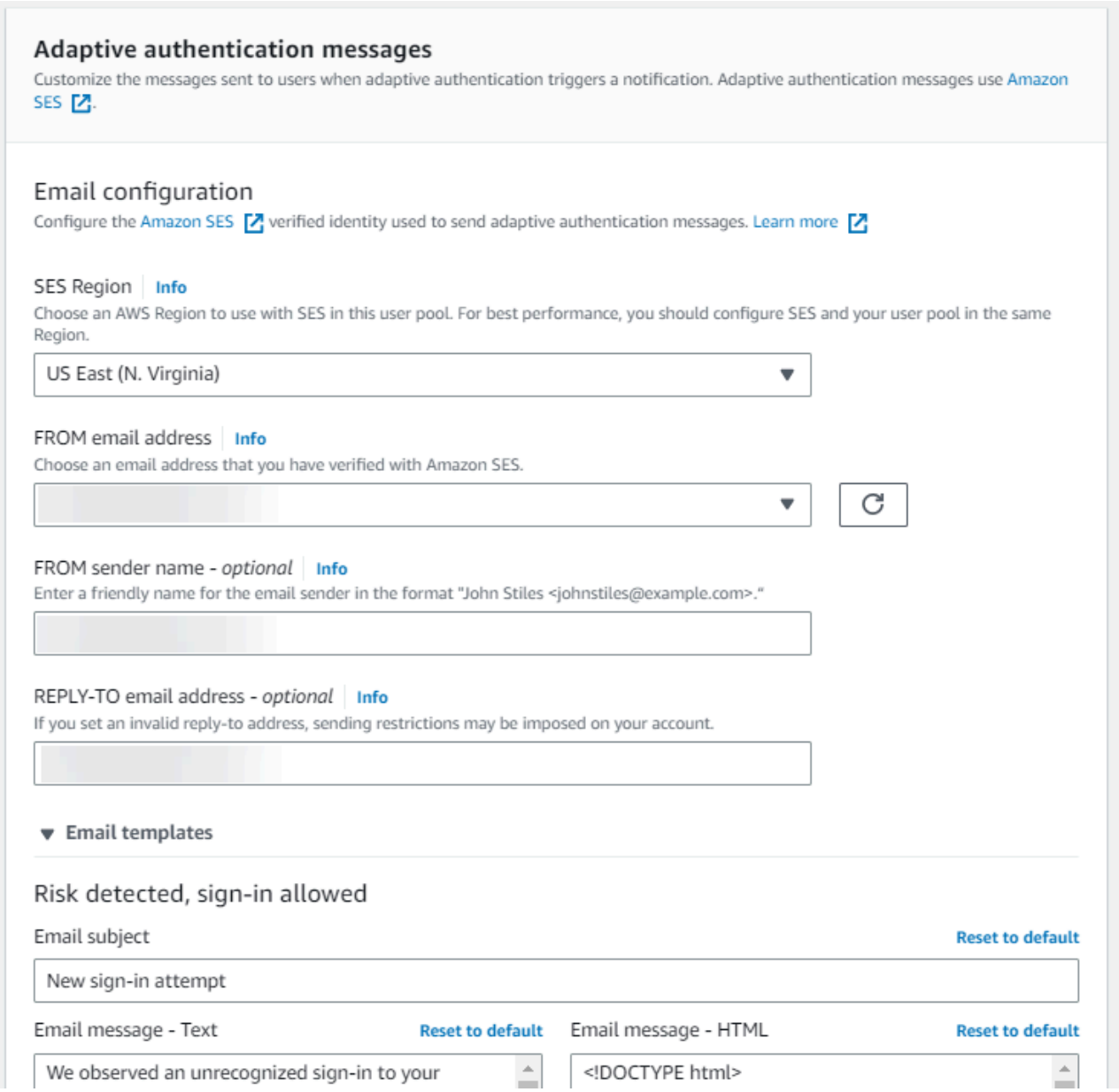
iii. MFA 필요 - 사용자에게 MFA가 구성된 경우 Amazon Cognito는 사용자가 로그인할 때 항상 추가 SMS 또는 TOTP 팩터를 제공하도록 요구합니다. 사용자에게 MFA가 구성되지 않은 경우 Amazon Cognito는 MFA를 설정하라는 메시지를 표시합니다. 사용자에게 MFA를 자동으로 요구하기 전에 SMS MFA용 전화 번호를 캡처하거나 TOTP MFA용 인증 앱을 등록하는 메커니즘을 앱에 구성합니다.

iv. 로그인 차단(Block sign-in) - 사용자가 로그인할 수 없도록 합니다.

v. 사용자 알림(Notify user) - Amazon Cognito가 탐지한 위험과 수행된 대응 작업에 대한 정보가 포함된 이메일 메시지를 사용자에게 보냅니다. 보내는 메시지의 이메일 메시지 템플릿을 사용자 지정할 수 있습니다.

9. 이전 단계에서 사용자 알림(Notify user)을 선택한 경우 조정 인증용 이메일 전송 설정 및 이메일 템플릿을 사용자 지정할 수 있습니다.

   a. 이메일 구성에서 적응형 인증과 함께 사용할 SES 리전, 보낸 사람 이메일 주소, 보낸 사람 발신자 이름 및 회신 이메일 주소를 선택합니다. 사용자 풀 이메일 메시지를 Amazon Simple Email Service와 통합하는 방법에 대한 자세한 내용은 Amazon Cognito 사용자 풀에 대한 이메일 설정을 참조하세요.



b. 이메일 템플릿(Email templates)을 확장하여 HTML 버전의 이메일 메시지와 일반 텍스트 버전의 이메일 메시지 모두에서 조정 인증 알림을 사용자 지정합니다. 이메일 메시지 템플릿에 대한 자세한 내용은 메시지 템플릿 섹션을 참조하세요.

10. IP 주소 예외를 확장하여 위협 방지 위험 평가에 관계없이 항상 허용되거나 차단되는 IPv4 또는 IPv6 주소 범위의 항상 허용 또는 항상 차단 목록을 생성합니다. CIDR 표기법(예: 192.168.100.0/24)으로 IP 주소 범위를 지정합니다.
11. 변경 사항 저장을 선택합니다.

API (user pool)

사용자 풀에 대한 위협 방지 구성을 설정하려면 `ClientId` 파라미터가 아닌 `UserPoolId` 파라미터가 포함된 SetRiskConfiguration API 요청을 보냅니다. 다음은 사용자 풀에 대한 요청 본문의 예입니다. 이 위험 구성은 위험의 심각도에 따라 일련의 작업을 에스컬레이션하고 모든 위험 수준의 사용자에게 알립니다. 가입 작업에 손상된 자격 증명 블록을 적용합니다.

이 구성을 적용하려면 별도의 CreateUserPool 또는 UpdateUserPool API 요청에서 `AdvancedSecurityMode`를 `ENFORCED`로 설정해야 합니다. 이 예제의 `{username}`와 같은 자리 표시자 템플릿에 대한 자세한 내용은 MFA, 인증, 확인 및 초대 메시지 구성 섹션을 참조하십시오.

```
{
   "AccountTakeoverRiskConfiguration": {
      "Actions": {
         "HighAction": {
            "EventAction": "MFA_REQUIRED",
            "Notify": true
         },
         "LowAction": {
            "EventAction": "NO_ACTION",
            "Notify": true
         },
         "MediumAction": {
            "EventAction": "MFA_IF_CONFIGURED",
            "Notify": true
         }
      },
      "NotifyConfiguration": {
         "BlockEmail": {
            "Subject": "You have been blocked for suspicious activity",
            "TextBody": "We blocked {username} at {login-time} from {ip-address}."
         },
         "From": "admin@example.com",
         "MfaEmail": {
            "Subject": "Suspicious activity detected, MFA required",
            "TextBody": "Unexpected sign-in from {username} on device {device-name}.
 You must use MFA."
         },
         "NoActionEmail": {
            "Subject": "Suspicious activity detected, secure your user account",
            "TextBody": "We noticed suspicious sign-in activity by {username} from
 {city}, {country} at {login-time}. If this was not you, reset your password."
         },
```

```
         "ReplyTo": "admin@example.com",
         "SourceArn": "arn:aws:ses:us-west-2:123456789012:identity/
admin@example.com"
      }
   },
   "CompromisedCredentialsRiskConfiguration": {
      "Actions": {
         "EventAction": "BLOCK"
      },
      "EventFilter": [ "SIGN_UP" ]
   },
   "RiskExceptionConfiguration": {
      "BlockedIPRangeList": [ "192.0.2.0/24","198.51.100.0/24" ],
      "SkippedIPRangeList": [ "203.0.113.0/24" ]
   },
   "UserPoolId": "us-west-2_EXAMPLE"
}
```

API (app client)

앱 클라이언트에 대한 위협 방지 구성을 설정하려면 `UserPoolId` 파라미터와 `ClientId` 파라미터를 포함하는 SetRiskConfiguration API 요청을 보냅니다. 다음은 앱 클라이언트에 대한 요청 본문의 예입니다. 이 위험 구성은 사용자 풀 구성보다 더 심각하여 고위험 항목을 차단합니다. 또한 손상된 자격 증명 블록을 가입, 로그인 및 암호 재설정 작업에 적용합니다.

이 구성을 적용하려면 별도의 CreateUserPool 또는 UpdateUserPool API 요청에서 `AdvancedSecurityMode`를 `ENFORCED`로 설정해야 합니다. 이 예제의 `{username}`와 같은 자리 표시자 템플릿에 대한 자세한 내용은 MFA, 인증, 확인 및 초대 메시지 구성 섹션을 참조하십시오.

```
{
   "AccountTakeoverRiskConfiguration": {
      "Actions": {
         "HighAction": {
            "EventAction": "BLOCK",
            "Notify": true
         },
         "LowAction": {
            "EventAction": "NO_ACTION",
            "Notify": true
         },
         "MediumAction": {
            "EventAction": "MFA_REQUIRED",
            "Notify": true
```

```
         }
      },
      "NotifyConfiguration": {
         "BlockEmail": {
            "Subject": "You have been blocked for suspicious activity",
            "TextBody": "We blocked {username} at {login-time} from {ip-address}."
         },
         "From": "admin@example.com",
         "MfaEmail": {
            "Subject": "Suspicious activity detected, MFA required",
            "TextBody": "Unexpected sign-in from {username} on device {device-name}.
 You must use MFA."
         },
         "NoActionEmail": {
            "Subject": "Suspicious activity detected, secure your user account",
            "TextBody": "We noticed suspicious sign-in activity by {username} from
 {city}, {country} at {login-time}. If this was not you, reset your password."
         },
         "ReplyTo": "admin@example.com",
         "SourceArn": "arn:aws:ses:us-west-2:123456789012:identity/
admin@example.com"
      }
   },
   "ClientId": "1example23456789",
   "CompromisedCredentialsRiskConfiguration": {
      "Actions": {
         "EventAction": "BLOCK"
      },
      "EventFilter": [ "SIGN_UP", "SIGN_IN", "PASSWORD_CHANGE" ]
   },
   "RiskExceptionConfiguration": {
      "BlockedIPRangeList": [ "192.0.2.1/32","192.0.2.2/32" ],
      "SkippedIPRangeList": [ "192.0.2.3/32","192.0.2.4/32" ]
   },
   "UserPoolId": "us-west-2_EXAMPLE"
}
```

손상된 자격 증명 감지 작업

Amazon Cognito는 사용자의 사용자 이름과 암호가 다른 곳에서 손상되었는지 탐지할 수 있습니다. 사용자가 자격 증명을 둘 이상의 사이트에서 재사용하거나 안전하지 않은 암호를 사용할 때 이러한 문제

가 발생할 수 있습니다. Amazon Cognito는 사용자 이름과 암호로 관리형 로그인 및 Amazon Cognito API로 로그인하는 로컬 사용자를 확인합니다.

Amazon Cognito 콘솔의 위협 방지 메뉴에서 손상된 자격 증명을 구성할 수 있습니다. 이벤트 감지(Event detection)를 구성하여 손상된 보안 인증 정보에 대해 모니터링할 사용자 이벤트를 선택합니다. 손상된 보안 인증 정보 응답(Compromised credentials responses)을 구성하여 손상된 보안 인증 정보가 감지된 경우 사용자를 허용할지 또는 차단할지 선택합니다. 로그인, 가입 및 암호 변경을 진행하는 동안 Amazon Cognito에서 손상된 보안 인증 정보를 확인할 수 있습니다.

로그인 허용(Allow sign-in)을 선택한 경우 Amazon CloudWatch Logs를 검토하여 사용자 이벤트에서 Amazon Cognito가 수행하는 평가를 모니터링할 수 있습니다. 자세한 내용은 위협 방지 지표 보기 섹션을 참조하세요. 로그인 차단(Block sign-in)을 선택하면 Amazon Cognito는 손상된 보안 인증 정보를 이용하는 사용자의 로그인을 방지합니다. Amazon Cognito가 사용자의 로그인을 차단하면 사용자의 `UserStatus`가 `RESET_REQUIRED`로 설정됩니다. `RESET_REQUIRED` 상태의 사용자는 암호를 변경해야 다시 로그인할 수 있습니다.

손상된 자격 증명은 다음 사용자 활동에 대한 암호를 확인할 수 있습니다.

가입

사용자 풀은 사용자가 SignUp 작업 및 관리형 로그인의 가입 페이지에서 전송하는 암호에 손상 지표가 있는지 확인합니다.

로그인

사용자 풀은 사용자가 암호 기반 로그인에서 제출하는 암호에 손상 지표가 있는지 확인합니다. Amazon Cognito는 AdminInitiateAuth의 `ADMIN_USER_PASSWORD_AUTH` 흐름, InitiateAuth의 `USER_PASSWORD_AUTH` 흐름 및 둘 다의 `USER_AUTH` 흐름 `PASSWORD` 옵션을 검토할 수 있습니다.

현재 Amazon Cognito는 SRP(Secure Remote Password) 흐름을 사용한 로그인 작업에서 손상된 보안 인증을 확인하지 않습니다. SRP는 로그인 시 해시된 암호 증명을 전송합니다. Amazon Cognito는 내부적으로 암호에 액세스할 수 없으므로 클라이언트가 일반 텍스트로 전달하는 암호만 평가할 수 있습니다.

암호 재설정

사용자 풀은 ConfirmForgotPassword 셀프 서비스 암호 재설정 작업을 사용하여 새 사용자 암호를 설정하는 작업의 손상 지표를 확인합니다. 이 작업에 필요한 코드는 ForgotPassword 및 AdminResetUserPassword에서 생성됩니다.

손상된 자격 증명은 AdminSetUserPassword로 설정된 임시 또는 영구 관리자 설정 암호를 확인하지 않습니다. 그러나 임시 암호를 사용하면 사용자 풀이 RespondToAuthChallenge 및

AdminRespondToAuthChallenge의 NEW_PASSWORD_REQUIRED 문제에 대한 응답의 암호를 확인합니다.

사용자 풀에 손상된 자격 증명 차단을 추가하는 방법은 위협 방지 기능이 있는 고급 보안 섹션을 참조하세요.

적응형 인증 작업

조정 인증을 사용하면 의심스러운 로그인을 차단하거나 위험 수준이 높아졌을 때 두 번째 인증 요소를 추가하도록 사용자 풀을 구성할 수 있습니다. Amazon Cognito는 각 로그인 시도에 대해 손상된 소스로부터 로그인 요청이 나오게 될 가능성에 대해 위험 점수를 계산합니다. 이 위험 점수는 애플리케이션이 제공하는 디바이스 및 사용자 요인과 Amazon Cognito가 요청에서 파생된 기타 요인을 기반으로 합니다. Amazon Cognito의 위험 평가에 영향을 미치는 몇 가지 요인은 IP 주소, 사용자 에이전트 및 다른 로그인 시도와의 지리적 거리입니다. 적응형 인증은 Amazon Cognito가 사용자 세션에서 위험을 감지하고 사용자가 아직 MFA 방법을 선택하지 않은 경우 사용자 풀의 사용자에 대해 다중 인증(MFA)을 켜거나 요구할 수 있습니다. 사용자에 대해 MFA를 활성화하면 적응형 인증이 구성된 방식과 관계없이 인증 중에 두 번째 요소를 제공하거나 설정하라는 인증 문제가 항상 수신됩니다. 사용자의 관점에서 보면 앱이 MFA 설정을 지원하며, 선택적으로 Amazon Cognito는 추가 요소를 구성할 때까지 사용자가 다시 로그인하지 못하도록 합니다.

Amazon Cognito는 로그인 시도, 위험 수준 및 실패한 챌린지에 대한 지표를 Amazon CloudWatch에 게시합니다. 자세한 내용은 위협 방지 지표 보기 섹션을 참조하세요.

사용자 풀에 조정 인증을 추가하는 방법은 위협 방지 기능이 있는 고급 보안 섹션을 참조하세요.

주제

- 조정 인증 개요
- API 요청에 사용자 디바이스 및 세션 데이터 추가
- 사용자 이벤트 기록 보기 및 내보내기
- 이벤트 피드백 제공
- 알림 메시지 전송

조정 인증 개요

Amazon Cognito 콘솔의 위협 방지 메뉴에서 다양한 위험 수준에서 수행할 작업과 사용자에 대한 알림 메시지 사용자 지정을 포함하여 적응형 인증에 대한 설정을 선택할 수 있습니다. 모든 앱 클라이언트에

글로벌 위협 방지 구성을 할당할 수 있지만 개별 앱 클라이언트에는 클라이언트 수준 구성을 적용할 수 있습니다.

Amazon Cognito 조정 인증은 각 사용자 세션에 높음, 중간, 낮음 또는 위험 없음 위험 수준 중 하나를 할당합니다.

Enforcement method(적용 방법)를 Audit-only(감사 전용)에서 Full-function(전체 기능)으로 변경할 때는 옵션을 신중하게 고려하세요. 위험 수준에 적용하는 자동 응답은 Amazon Cognito가 동일한 특성을 가진 후속 사용자 세션에 할당하는 위험 수준에 영향을 줍니다. 예를 들어, Amazon Cognito가 처음에 고위험으로 평가한 사용자 세션에 조치를 취하지 않거나 Allow(허용)를 선택하면 Amazon Cognito는 유사한 세션을 위험이 낮은 것으로 간주합니다.

위험 수준별로 다음 옵션 중에서 선택할 수 있습니다.

옵션	작업
허용	사용자는 추가 팩터 없이 로그인할 수 있습니다.
선택적 MFA	두 번째 팩터가 구성된 사용자는 로그인을 위해 두 번째 팩터 문제를 완료해야 합니다. SMS 전화 번호 및 TOTP 소프트웨어 토큰이 사용 가능한 두 번째 팩터입니다. 두 번째 인증 요소를 구성하지 않은 사용자는 하나의 자격 증명 세트로만 로그인할 수 있습니다.
MFA 필요	두 번째 팩터가 구성된 사용자는 로그인을 위해 두 번째 팩터 문제를 완료해야 합니다. Amazon Cognito는 두 번째 팩터가 구성되지 않은 사용자의 로그인을 차단합니다.
차단	Amazon Cognito는 지정된 위험 수준의 모든 로그인 시도를 차단합니다.

> **Note**
>
> 두 번째 인증 팩터인 SMS에 사용하기 위해 전화 번호를 인증할 필요가 없습니다.

API 요청에 사용자 디바이스 및 세션 데이터 추가

API를 사용하여 가입하고, 로그인하고, 암호를 재설정할 때 사용자의 세션 정보를 수집하여 Amazon Cognito 위협 방지로 전달할 수 있습니다. 이 정보에는 사용자의 IP 주소와 고유 디바이스 식별자가 포함됩니다.

프록시 서비스나 애플리케이션 서버와 같이 사용자와 Amazon Cognito 간에 중간 네트워크 디바이스가 있을 수 있습니다. 적응형 인증이 서버나 프록시 대신 사용자 엔드포인트의 특성을 기반으로 위험을 계산하도록 사용자의 컨텍스트 데이터를 수집하여 Amazon Cognito에 전달할 수 있습니다. 클라이언트측 앱이 Amazon Cognito API 작업을 직접 호출하는 경우 적응형 인증은 소스 IP 주소를 자동으로 기록합니다. 그러나 디바이스 지문을 수집하지 않는 한 `user-agent`와 같은 다른 디바이스 정보는 기록하지 않습니다.

Amazon Cognito 컨텍스트 데이터 수집 라이브러리로 이 데이터를 생성하여 ContextData 및 UserContextData 파라미터와 함께 Amazon Cognito 위협 방지에 제출합니다. 컨텍스트 데이터 수집 라이브러리는 AWS SDKs에 포함됩니다. 자세한 내용은 웹 및 모바일 앱과 Amazon Cognito 인증 및 권한 부여 통합 섹션을 참조하세요. Plus 기능 플랜이 있는 경우 `ContextData`를 제출할 수 있습니다. 자세한 내용은 위협 방지 설정 섹션을 참조하세요.

애플리케이션 서버에서 다음 Amazon Cognito 인증 API 작업을 호출할 때 사용자 디바이스의 IP를 `ContextData` 파라미터에 전달합니다. 또한 서버 이름, 서버 경로 및 인코딩된 디바이스 지문 데이터를 전달합니다.

- AdminInitiateAuth
- AdminRespondToAuthChallenge

Amazon Cognito에서 인증하지 않은 API 작업을 호출하면 `UserContextData`를 Amazon Cognito 위협 방지에 제출할 수 있습니다. 이 데이터에는 `EncodedData` 파라미터에 디바이스 지문이 포함되어 있습니다. 다음 조건을 충족하는 경우 `UserContextData`에 `IpAddress` 파라미터를 제출할 수도 있습니다.

- 사용자 풀은 Plus 기능 플랜에 있습니다. 자세한 내용은 사용자 풀 기능 플랜 섹션을 참조하세요.
- 앱 클라이언트에 클라이언트 보안 암호가 있습니다. 자세한 내용은 앱 클라이언트를 사용한 애플리케이션별 설정 섹션을 참조하세요.
- 앱 클라이언트에서 추가 사용자 컨텍스트 데이터 허용(Accept additional user context data)을 활성화했습니다. 자세한 내용은 추가 사용자 컨텍스트 데이터 수락(AWS Management Console) 섹션을 참조하세요.

앱은 다음 Amazon Cognito에서 인증하지 않은 API 작업의 인코딩된 디바이스 지문 데이터와 사용자 디바이스 IP 주소로 `UserContextData` 파라미터를 채울 수 있습니다.

- InitiateAuth
- RespondToAuthChallenge
- SignUp
- ConfirmSignUp
- ForgotPassword
- ConfirmForgotPassword
- ResendConfirmationCode

추가 사용자 컨텍스트 데이터 수락(AWS Management Console)

추가 사용자 컨텍스트 데이터 허용(Accept additional user context data) 기능을 활성화하면 사용자 풀에서 `UserContextData` 파라미터의 IP 주소를 수락합니다. 다음과 같은 경우에는 이 기능을 활성화할 필요가 없습니다.

- 사용자가 AdminInitiateAuth와 같은 인증 API 작업으로만 로그인하고, 본인은 `ContextData` 파라미터를 사용합니다.
- 인증되지 않은 API 작업에서 IP 주소가 아닌 디바이스 지문만 Amazon Cognito 위협 방지로 전송하려고 합니다.

Amazon Cognito 콘솔에서 다음과 같이 앱 클라이언트를 업데이트하여 추가 사용자 컨텍스트 데이터에 대한 지원을 추가합니다.

1. Amazon Cognito 콘솔에 로그인합니다.
2. 탐색 창에서 사용자 풀 관리(Manage your User Pools)를 선택한 다음 편집할 사용자 풀을 선택합니다.
3. 앱 클라이언트 메뉴를 선택합니다.
4. 앱 클라이언트를 선택하거나 생성합니다. 자세한 내용은 사용자 풀 앱 클라이언트 구성을 참조하세요.
5. 앱 클라이언트 정보(App client information) 컨테이너에서 편집(Edit)을 선택합니다.
6. 앱 클라이언트의 고급 인증 설정(Advanced authentication settings)에서 추가 사용자 컨텍스트 데이터 허용(Accept additional user context data)을 선택합니다.

7. 변경 사항 저장을 선택합니다.

Amazon Cognito API에서 사용자 컨텍스트 데이터를 수락하도록 앱 클라이언트를 구성하려면CreateUserPoolClient 또는 UpdateUserPoolClient 요청에서 `EnablePropagateAdditionalUserContextData`를 `true`로 설정합니다. 웹 또는 모바일 앱에서 위협 방지 기능을 사용하는 방법에 대한 자세한 내용은 애플리케이션에서 위협 방지를 위한 데이터 수집 섹션을 참조하세요. 앱이 서버에서 Amazon Cognito를 호출하면 클라이언트 측에서 사용자 컨텍스트 데이터를 수집합니다. 다음은 JavaScript SDK 메서드 `getData`를 사용하는 예제입니다.

```
var EncodedData =
 AmazonCognitoAdvancedSecurityData.getData(username, userPoolId, clientId);
```

앱에서 조정 인증을 사용하도록 설계하는 경우 최신 Amazon Cognito SDK를 앱에 통합하는 것이 좋습니다. SDK의 최신 버전은 디바이스 ID, 모델 및 표준 시간대와 같은 디바이스 지문 정보를 수집합니다. Amazon Cognito SDK에 대한 자세한 내용은 사용자 풀 SDK 설치를 참조하세요. Amazon Cognito 위협 방지는 앱이 올바른 형식으로 제출하는 이벤트에만 위험 점수를 저장하고 할당합니다. Amazon Cognito에서 오류 응답을 반환하는 경우 요청에 유효한 시크릿 해시가 포함되어 있는지, `IPaddress` 파라미터가 유효한 IPv4 또는 IPv6 주소인지 확인합니다.

`ContextData` 및 `UserContextData` 리소스

- AWS Amplify Android용 SDK: GetUserContextData
- AWS Amplify iOS용 SDK: userContextData
- JavaScript: amazon-cognito-advanced-security-data.min.js

사용자 이벤트 기록 보기 및 내보내기

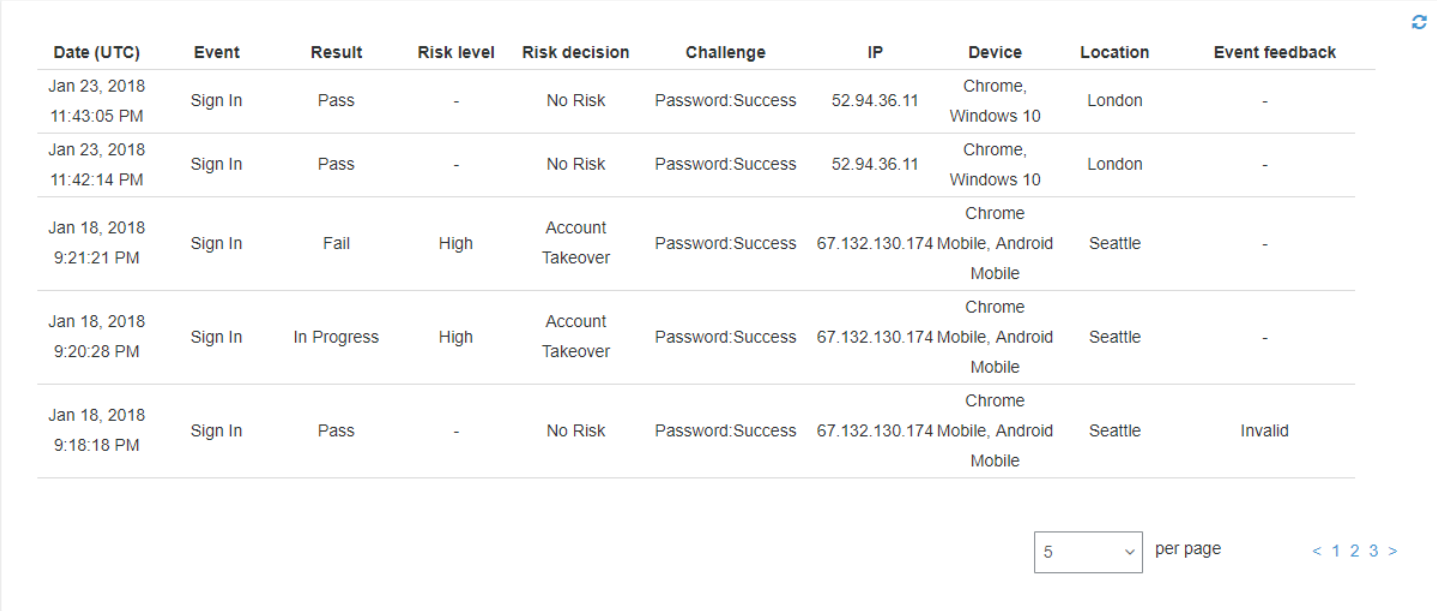
Amazon Cognito는 위협 방지를 활성화하면 사용자가 각 인증 이벤트에 대한 로그를 생성합니다. 기본적으로 Amazon Cognito 콘솔의 사용자 메뉴 또는 AdminListUserAuthEvents API 작업에서 사용자 로그를 볼 수 있습니다. 이러한 이벤트를 CloudWatch Logs, Amazon S3 또는 Amazon Data Firehose와 같은 외부 시스템으로 내보낼 수도 있습니다. 내보내기 기능을 사용하면 애플리케이션의 사용자 활동에 대한 보안 정보를 자체 보안 분석 시스템에 더 쉽게 액세스할 수 있습니다.

주제

- 사용자 이벤트 기록 보기(AWS Management Console)
- 사용자 이벤트 기록 보기(API/CLI)
- 사용자 인증 이벤트 내보내기

사용자 이벤트 기록 보기(AWS Management Console)

사용자의 로그인 기록을 보려면 Amazon Cognito 콘솔의 사용자 메뉴에서 사용자를 선택합니다. Amazon Cognito는 사용자 이벤트 기록을 2년 동안 보관합니다.



Date (UTC)	Event	Result	Risk level	Risk decision	Challenge	IP	Device	Location	Event feedback
Jan 23, 2018 11:43:05 PM	Sign In	Pass	-	No Risk	Password:Success	52.94.36.11	Chrome, Windows 10	London	-
Jan 23, 2018 11:42:14 PM	Sign In	Pass	-	No Risk	Password:Success	52.94.36.11	Chrome, Windows 10	London	-
Jan 18, 2018 9:21:21 PM	Sign In	Fail	High	Account Takeover	Password:Success	67.132.130.174	Chrome Mobile, Android Mobile	Seattle	-
Jan 18, 2018 9:20:28 PM	Sign In	In Progress	High	Account Takeover	Password:Success	67.132.130.174	Chrome Mobile, Android Mobile	Seattle	-
Jan 18, 2018 9:18:18 PM	Sign In	Pass	-	No Risk	Password:Success	67.132.130.174	Chrome Mobile, Android Mobile	Seattle	Invalid

5 per page < 1 2 3 >

각 로그인 이벤트에는 이벤트 ID가 있습니다. 이벤트에는 위치, 디바이스 세부 정보 및 위험 탐지 결과와 같은 해당 컨텍스트 데이터도 있습니다.

이벤트 ID를 이벤트를 기록한 시점에 Amazon Cognito가 발행한 토큰과 연관시킬 수도 있습니다. ID 및 액세스 토큰은 페이로드에 이 이벤트 ID를 포함합니다. Amazon Cognito는 새로 고침 토큰 사용을 원래 이벤트 ID와 연관시킵니다. 원래 이벤트 ID를 Amazon Cognito 토큰을 발급한 로그인 이벤트의 이벤트 ID로 다시 추적할 수 있습니다. 시스템 내에서 특정 인증 이벤트에 대한 토큰 사용을 추적할 수 있습니다. 자세한 내용은 사용자 풀 JSON 웹 토큰(JWT) 이해 섹션을 참조하세요.

사용자 이벤트 기록 보기(API/CLI)

Amazon Cognito API 작업 AdminListUserAuthEvents 또는 admin-list-user-auth-events가 있는 AWS Command Line Interface (AWS CLI)를 사용하여 사용자 이벤트 기록을 쿼리할 수 있습니다.

AdminListUserAuthEvents request

`AdminListUserAuthEvents`에 대한 다음 요청 본문은 한 사용자에 대한 최신 활동 로그를 반환합니다.

```
{
  "UserPoolId": "us-west-2_EXAMPLE",
```

```
  "Username": "myexampleuser",
  "MaxResults": 1
}
```

admin-list-user-auth-events request

`admin-list-user-auth-events`에 대한 다음 요청은 한 사용자에 대한 최신 활동 로그를 반환합니다.

```
aws cognito-idp admin-list-user-auth-events --max-results 1 --username myexampleuser
 --user-pool-id us-west-2_EXAMPLE
```

Response

Amazon Cognito는 두 요청에 동일한 JSON 응답 본문을 반환합니다. 다음은 위험 요소를 포함하지 않은 관리형 로그인 이벤트에 대한 예시 응답입니다.

```
{
    "AuthEvents": [
        {
            "EventId": "[event ID]",
            "EventType": "SignIn",
            "CreationDate": "[Timestamp]",
            "EventResponse": "Pass",
            "EventRisk": {
                "RiskDecision": "NoRisk",
                "CompromisedCredentialsDetected": false
            },
            "ChallengeResponses": [
                {
                    "ChallengeName": "Password",
                    "ChallengeResponse": "Success"
                }
            ],
            "EventContextData": {
                "IpAddress": "192.168.2.1",
                "DeviceName": "Chrome 125, Windows 10",
                "Timezone": "-07:00",
                "City": "Bellevue",
                "Country": "United States"
            }
        }
```

```
    ],
    "NextToken": "[event ID]#[Timestamp]"
}
```

사용자 인증 이벤트 내보내기

위협 방지에서 외부 시스템으로 사용자 이벤트를 내보내도록 사용자 풀을 구성합니다. 지원되는 외부 시스템인 Amazon S3, CloudWatch Logs 및 Amazon Data Firehose는 전송하거나 검색하는 데이터에 대한 비용을 AWS 청구서에 추가할 수 있습니다. 자세한 내용은 위협 방지 사용자 활동 로그 내보내기 섹션을 참조하세요.

AWS Management Console

1. Amazon Cognito 콘솔에 로그인합니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. 로그 스트리밍 메뉴를 선택합니다. 편집을 선택합니다.
5. 로깅 상태에서 사용자 활동 로그 내보내기 활성화 옆의 확인란을 선택합니다.
6. 로깅 대상에서 로그를 처리할 AWS 서비스 (CloudWatch 로그 그룹, Amazon Data Firehose 스트림 또는 S3 버킷)를 선택합니다.
7. 선택하면 리소스 선택기가 해당 리소스 유형으로 채워집니다. 목록에서 로그 그룹, 스트림 또는 버킷을 선택합니다. 생성 버튼을 선택하여 선택한 서비스의 AWS Management Console 로 이동하고 새 리소스를 생성할 수도 있습니다.
8. 변경 사항 저장을 선택합니다.

API

사용자 활동 로그의 대상 유형 하나를 선택합니다.

다음은 Firehose 스트림을 로그 대상으로 설정하는 `SetLogDeliveryConfiguration` 요청 본문의 예입니다.

```
{
   "LogConfigurations": [
      {
         "EventSource": "userAuthEvents",
```

```
         "FirehoseConfiguration": {
            "StreamArn": "arn:aws:firehose:us-west-2:123456789012:deliverystream/
example-user-pool-activity-exported"
         },
         "LogLevel": "INFO"
      }
   ],
   "UserPoolId": "us-west-2_EXAMPLE"
}
```

다음은 Amazon S3 버킷을 로그 대상으로 설정하는 SetLogDeliveryConfiguration 요청 본문의 예입니다.

```
{
   "LogConfigurations": [
      {
         "EventSource": "userAuthEvents",
         "S3Configuration": {
            "BucketArn": "arn:aws:s3:::amzn-s3-demo-logging-bucket"
         },
         "LogLevel": "INFO"
      }
   ],
   "UserPoolId": "us-west-2_EXAMPLE"
}
```

다음은 CloudWatch 로그 그룹을 로그 대상으로 설정하는 SetLogDeliveryConfiguration 요청 본문의 예입니다.

```
{
   "LogConfigurations": [
      {
         "EventSource": "userAuthEvents",
         "CloudWatchLogsConfiguration": {
            "LogGroupArn": "arn:aws:logs:us-west-2:123456789012:log-group:DOC-
EXAMPLE-LOG-GROUP"
         },
         "LogLevel": "INFO"
      }
   ],
   "UserPoolId": "us-west-2_EXAMPLE"
}
```

이벤트 피드백 제공

이벤트 피드백은 실시간으로 위험 평가에 영향을 미치고 시간이 지남에 따라 위험 평가 알고리즘을 개선합니다. 또한 Amazon Cognito 콘솔 및 API 작업을 통해 로그인 시도의 유효성에 대한 피드백을 제공할 수 있습니다.

> **Note**
>
> 이벤트 피드백은 Amazon Cognito가 동일한 특성을 가진 후속 사용자 세션에 할당하는 위험 수준에 영향을 미칩니다.

Amazon Cognito 콘솔의 사용자 메뉴에서 사용자를 선택하고 이벤트 피드백 제공을 선택합니다. 이벤트 세부 정보를 검토하고 Set as valid(유효한 것으로 설정) 또는 Set as invalid(잘못된 것으로 설정)를 선택할 수 있습니다.

콘솔은 사용자 메뉴의 사용자 세부 정보에 로그인 기록을 나열합니다. 항목을 선택한 경우 이벤트를 유효하거나 유효하지 않음으로 표시할 수 있습니다. 사용자 풀 API 작업 AdminUpdateAuthEventFeedback과 AWS CLI 명령 admin-update-auth-event-feedback을 통해 피드백을 제공할 수도 있습니다.

Amazon Cognito 콘솔에서 Set as valid(유효한 것으로 설정)를 선택하거나 API에서 `valid FeedbackValue` 값을 제공하면 Amazon Cognito가 일정 수준의 위험을 평가한 사용자 세션을 신뢰한다고 Amazon Cognito에 알리는 것입니다. Amazon Cognito 콘솔에서 Set as invalid(잘못된 것으로 설정)를 선택하거나 API에서 `invalid FeedbackValue` 값을 제공하면 사용자 세션을 신뢰하지 않거나 Amazon Cognito가 충분히 높은 수준의 위험을 평가했다고 믿지 않는다고 Amazon Cognito에 알리는 것입니다.

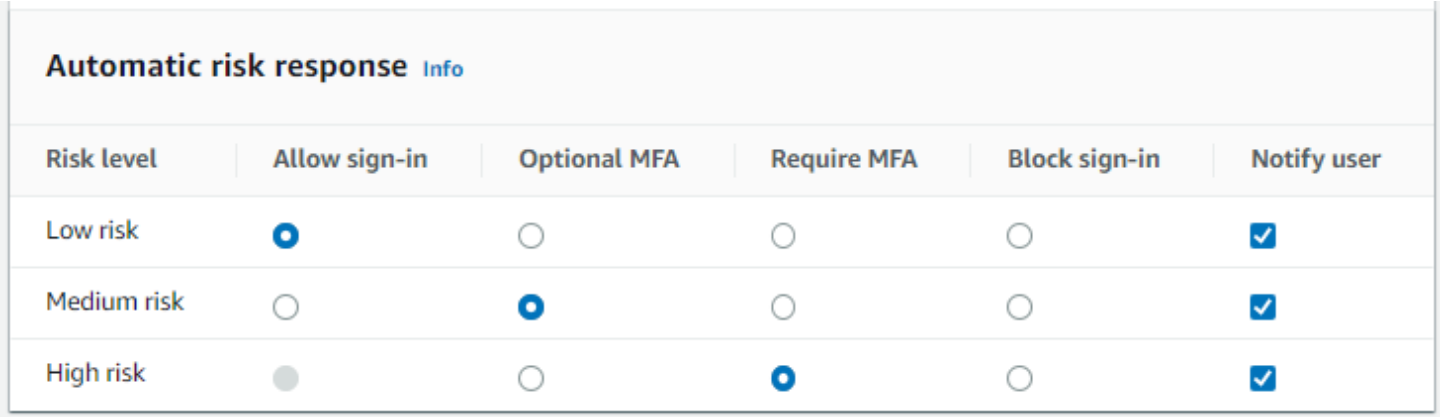
알림 메시지 전송

위협 방지를 통해 Amazon Cognito는 사용자에게 위험한 로그인 시도를 알릴 수 있습니다. 또한 Amazon Cognito는 사용자에게 로그인이 유효한지 여부를 나타내는 링크를 선택하라는 메시지를 표시할 수 있습니다. Amazon Cognito는 이 피드백을 사용하여 사용자 풀의 위험 탐지 정확도를 개선합니다.

> **Note**
>
> Amazon Cognito는 작업이 로그인 차단, 로그인 허용, MFA를 선택 사항으로 설정 또는 MFA 요구와 같은 자동 위험 응답을 생성하는 경우에만 사용자에게 알림 메시지를 보냅니다. 일부 요청에는 위험 수준이 할당될 수 있지만 적응형 인증 자동 위험 응답을 생성하지 않습니다. 이 경

우 사용자 풀은 알림을 보내지 않습니다. 예를 들어 잘못된 암호는 위험 등급과 함께 로깅될 수 있지만 Amazon Cognito의 응답은 적응형 인증 규칙을 적용하지 않고 로그인에 실패하는 것입니다.

Automatic risk response(자동 위험 대응) 섹션에서 낮음, 중간 또는 높은 위험 사례에 대해 Notify Users(사용자 알림)를 선택합니다.



Amazon Cognito는 이메일 주소 확인 여부와 관계없이 사용자에게 이메일 알림을 보냅니다.

알림 이메일 메시지를 사용자 지정하고, 이러한 메시지의 일반 텍스트 버전과 HTML 버전을 모두 제공할 수 있습니다. 이메일 알림을 사용자 지정하려면 위협 방지 구성의 적응형 인증 메시지에서 이메일 템플릿을 엽니다. 이메일 템플릿에 대한 자세한 내용은 메시지 템플릿 섹션을 참조하세요.

애플리케이션에서 위협 방지를 위한 데이터 수집

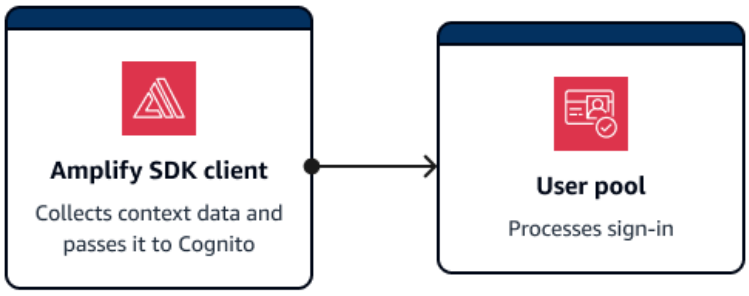
Amazon Cognito 적응형 인증은 사용자의 로그인 시도에 대한 컨텍스트 세부 정보에서 시도된 계정 탈취에 대한 위험 수준을 평가합니다. Amazon Cognito 위협 방지가 위험을 보다 정확하게 평가할 수 있도록 애플리케이션은 API 요청에 컨텍스트 데이터를 추가해야 합니다. 컨텍스트 데이터는 IP 주소, 브라우저 에이전트, 디바이스 정보 및 요청 헤더와 같은 정보로, 사용자가 사용자 풀에 연결된 방식에 대한 컨텍스트 정보를 제공합니다.

Amazon Cognito에 이 컨텍스트를 제출하는 애플리케이션의 핵심 책임은 사용자 풀에 대한 인증 요청의 `EncodedData` 파라미터입니다. 요청에 이 데이터를 추가하려면 이 정보를 자동으로 생성하는 SDK를 사용하여 Amazon Cognito를 구현하거나 이 데이터를 수집하는 JavaScript, iOS 또는 Android용 모듈을 구현할 수 있습니다. Amazon Cognito에 직접 요청하는 클라이언트 전용 애플리케이션은 AWS Amplify SDKs 구현해야 합니다. 중간 서버 또는 API 구성 요소가 있는 클라이언트 서버 애플리케이션은 별도의 SDK 모듈을 구현해야 합니다.

다음 시나리오에서 인증 프런트 엔드는 추가 구성 없이 사용자 컨텍스트 데이터 수집을 관리합니다.

- 관리형 로그인은 컨텍스트 데이터를 자동으로 수집하여 위협 방지에 제출합니다.
- 모든 AWS Amplify 라이브러리에는 인증 방법에 컨텍스트 데이터 수집이 내장되어 있습니다.

Amplify를 사용하여 클라이언트 전용 애플리케이션에서 사용자 컨텍스트 데이터 제출



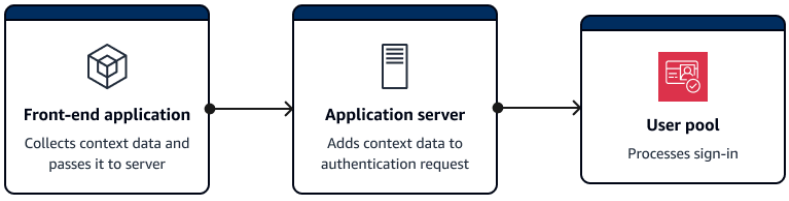
Amplify SDK는 Amazon Cognito로 직접 인증하는 모바일 클라이언트를 지원합니다. 이러한 유형의 클라이언트는 Amazon Cognito 퍼블릭 API 작업에 직접 API를 요청합니다. Amplify 클라이언트는 기본적으로 위협 방지를 위해 컨텍스트 데이터를 자동으로 수집합니다.

JavaScript를 사용한 Amplify 애플리케이션은 예외입니다. 사용자 컨텍스트 데이터를 수집하는 JavaScript 모듈을 추가해야 합니다.

일반적으로 이 구성의 애플리케이션은 InitiateAuth 및 RespondToAuthChallenge와 같은 인증되지 않은 API 작업을 사용합니다. UserContextData 개체는 이러한 작업에 대한 위험을 보다 정확하게 평가하는 데 도움이 됩니다. Amplify SDK는 디바이스 및 세션 정보를 `UserContextData`의 `EncodedData` 파라미터에 추가합니다.

클라이언트-서버 애플리케이션에서 컨텍스트 데이터 수집

일부 애플리케이션에는 사용자 인증 데이터를 수집하는 프런트 엔드 계층과 Amazon Cognito에 인증 요청을 제출하는 애플리케이션 백 엔드 계층이 있습니다. 이는 마이크로서비스가 지원하는 웹서버 및 애플리케이션의 일반적인 아키텍처입니다. 이러한 애플리케이션에서는 퍼블릭 컨텍스트 데이터 수집 라이브러리를 가져와야 합니다.



일반적으로 이 구성의 애플리케이션 서버는 AdminInitiateAuth 및 AdminRespondToAuthChallenge와 같은 인증된 API 작업을 사용합니다. ContextData 개체는 Amazon Cognito가 이러한 작업에 대한 위험을 보다 정확하게 평가하는 데 도움이 됩니다. ContextData의 내용은 프런트 엔드가 서버에 전달한 인코딩된 데이터와 서버에 대한 사용자의 HTTP 요청에서 추가 세부 정보입니다. HTTP 헤더 및 IP 주소와 같은 이러한 추가 컨텍스트 세부 정보는 애플리케이션 서버에 사용자 환경의 특성을 제공합니다.

애플리케이션 서버는 InitiateAuth 및 RespondToAuthChallenge와 같은 인증되지 않은 API 작업으로 로그인할 수도 있습니다. UserContextData 객체는 이러한 작업에서 위협 방지 위험 분석을 알립니다. 사용 가능한 퍼블릭 컨텍스트 데이터 수집 라이브러리의 작업은 인증 요청의 EncodedData 파라미터에 보안 정보를 추가합니다. 또한 추가 컨텍스트 데이터를 허용하도록 사용자 풀을 구성하고 사용자의 소스 IP를 UserContextData의 IpAddress 파라미터에 추가합니다.

클라이언트-서버 애플리케이션에 컨텍스트 데이터를 추가하는 방법

1. 프런트엔드 애플리케이션에서 iOS, Android 또는 JavaScript 모듈을 사용하여 클라이언트에서 인코딩된 컨텍스트 데이터를 수집합니다.
2. 인코딩된 데이터와 인증 요청의 세부 정보를 애플리케이션 서버에 전달합니다.
3. 애플리케이션 서버에서 HTTP 요청에서 사용자의 IP 주소, 관련 HTTP 헤더, 요청된 서버 이름 및 요청된 경로를 추출합니다. Amazon Cognito에 대한 API 요청의 ContextData 파라미터에 이러한 값을 채웁니다.
4. API 요청에서 ContextData의 EncodedData 파라미터를 SDK 모듈이 수집한 인코딩된 디바이스 데이터로 채웁니다. 이 컨텍스트 데이터를 인증 요청에 추가합니다.

클라이언트-서버 애플리케이션을 위한 컨텍스트 데이터 라이브러리

JavaScript

amazon-cognito-advanced-security-data.min.js 모듈은 애플리케이션 서버에 전달할 수 있는 EncodedData를 수집합니다.

JavaScript 구성에 amazon-cognito-advanced-security-data.min.js 모듈을 추가합니다. 를 us-east-1, , us-east-2, us-west-2, eu-west-2또는 AWS 리전 목록에서 eu-west-1<region>로 바꿉니다eu-central-1.

```
<script src="https://amazon-cognito-assets.<region>.amazoncognito.com/amazon-cognito-advanced-security-data.min.js"></script>
```

EncodedData 파라미터에서 사용할 수 있는 encodedContextData 개체를 생성하려면 JavaScript 애플리케이션 소스에 다음을 추가합니다.

```
var encodedContextData = AmazonCognitoAdvancedSecurityData.getData(_username,
 _userpoolId, _userPoolClientId);
```

iOS/Swift

컨텍스트 데이터를 생성하기 위해 iOS 애플리케이션은 Mobile SDK for iOS 모듈 AWSCognitoIdentityProviderASF를 통합할 수 있습니다.

위협 방지를 위해 인코딩된 컨텍스트 데이터를 수집하려면 애플리케이션에 다음 조각을 추가합니다.

```
import AWSCognitoIdentityProviderASF

let deviceId = getDeviceId()
let encodedContextData = AWSCognitoIdentityProviderASF.userContextData(
                            userPoolId,
                            username: username,
                            deviceId: deviceId,
                            userPoolClientId: userPoolClientId)

/**
 * Reuse DeviceId from keychain or generate one for the first time.
 */
func getDeviceId() -> String {
    let deviceIdKey = getKeyChainKey(namespace: userPoolId, key:
 "AWSCognitoAuthAsfDeviceId")

   if let existingDeviceId = self.keychain.string(forKey: deviceIdKey) {
        return existingDeviceId
    }

    let newDeviceId = UUID().uuidString
    self.keychain.setString(newDeviceId, forKey: deviceIdKey)
    return newDeviceId
}

/**
 * Get a namespaced keychain key given a namespace and key
 */
func getKeyChainKey(namespace: String, key: String) -> String {
    return "\(namespace).\(key)"
```

```
}
```

Android

컨텍스트 데이터를 생성하기 위해 Android 애플리케이션은 Mobile SDK for Android 모듈 aws-android-sdk-cognitoidentityprovider-asf를 통합할 수 있습니다.

위협 방지를 위해 인코딩된 컨텍스트 데이터를 수집하려면 애플리케이션에 다음 조각을 추가합니다.

```
UserContextDataProvider provider = UserContextDataProvider.getInstance();
// context here is android application context.
String encodedContextData = provider.getEncodedContextData(context, username,
 userPoolId, userPoolClientId);
```

AWS WAF 웹 ACL을 사용자 풀과 연결

AWS WAF 는 웹 애플리케이션 방화벽입니다. AWS WAF 웹 ACL(웹 액세스 제어 목록)을 사용하면 클래식 호스팅 UI, 관리형 로그인 및 Amazon Cognito API 서비스 엔드포인트에 대한 원치 않는 요청으로부터 사용자 풀을 보호할 수 있습니다. 웹 ACL을 사용하면 사용자 풀이 응답하는 모든 HTTPS 웹 요청을 세부적으로 제어할 수 있습니다. AWS WAF 웹 ACL에 대한 자세한 내용은 AWS WAF 개발자 안내서의 웹 액세스 제어 목록(웹 ACL) 관리 및 사용을 참조하세요.

사용자 풀과 연결된 AWS WAF 웹 ACL이 있는 경우 Amazon Cognito는 선택한 비기밀 헤더 및 사용자의 요청 내용을에 전달합니다 AWS WAF.는 요청 내용을 AWS WAF 검사하고 웹 ACL에 지정한 규칙과 비교하여 Amazon Cognito에 응답을 반환합니다. Amazon Cognito가에 전달하는 요청 구성 요소는 요청 유형에 AWS WAF 따라 다릅니다. 관리형 로그인 및 클래식 호스팅 UI의 경우 Amazon Cognito는 요청 본문을 전달하지 않습니다. 자세한 내용은 AWS WAF 웹 ACLs 및 Amazon Cognito에 대해 알아야 할 사항 섹션을 참조하세요.

AWS WAF 웹 ACLs 및 Amazon Cognito에 대해 알아야 할 사항

- 사용자 이름, 암호, 전화번호 또는 이메일 주소와 같은 사용자 풀 요청의 개인 식별 정보(PII)와 일치하도록 웹 ACL 규칙을 구성할 수 없습니다. 이 데이터는에서 사용할 수 없습니다 AWS WAF. 대신 IP 주소, 브라우저 에이전트 및 요청된 API 작업과 같은 헤더, 경로 및 본문의 세션 데이터와 일치하도록 웹 ACL 규칙을 구성합니다.
- Amazon Cognito는 사용자 풀 API에 대한 요청에 대해서만 본문 기반 웹 ACL 규칙을 지원합니다. 관리형 로그인 및 클래식 호스팅 UI의 경우 Amazon Cognito는 요청 본문을 전달하지 않고 헤더 및 기타 비 바디 구성 요소만 검사합니다.

- 웹 ACL 규칙은 브라우저가 관리형 로그인 페이지를 로드하는 경우에만 사용자 지정 블록 응답을 반환할 수 있습니다. 사용자가 로그인 양식을 제출할 때와 같이 페이지가 다시 로드하지 않고 보내는 요청은 대신 기본 403 블록 응답을 받습니다.
- 에 의해 차단된 요청은 모든 요청 유형에 대한 요청 속도 할당량에 포함되지 AWS WAF 않습니다. AWS WAF 핸들러는 API 수준 제한 핸들러보다 먼저 호출됩니다.
- 웹 ACL을 생성할 때 웹 ACL이 완전히 전파되어 Amazon Cognito에서 사용할 수 있게 되기까지 약간의 시간이 걸립니다. 전파 시간은 몇 초에서 몇 분 사이일 수 있습니다. 웹 ACL이 완전히 전파되기 전에 연결을 시도`WAFUnavailableEntityException`하면가를 AWS WAF 반환합니다.
- 각 사용자 풀에 하나의 웹 ACL을 연결할 수 있습니다.
- 요청으로 인해 AWS WAF 가 검사할 수 있는 한계보다 더 큰 페이로드가 발생할 수 있습니다. 가 Amazon Cognito의 과대 요청을 처리하는 방법을 구성하는 방법은 개발자 안내서의 과대 요청 구성 요소 처리를 참조하세요. AWS WAF AWS WAF
- AWS WAF Fraud Control 계정 탈취 방지(ATP)를 사용하는 웹 ACL을 Amazon Cognito 사용자 풀과 연결할 수 없습니다. ATP 기능은 `AWS-AWSManagedRulesATPRuleSet` 관리형 규칙 그룹에 있습니다. 웹 ACL을 사용자 풀과 연결하기 전에 해당 관리형 규칙 그룹을 사용하지 않는지 확인합니다.
- 사용자 풀과 연결된 AWS WAF 웹 ACL이 있고 웹 ACL의 규칙에 CAPTCHA가 표시되는 경우 관리형 로그인 TOTP 등록에서 복구할 수 없는 오류가 발생할 수 있습니다. CAPTCHA 작업이 있고 관리형 로그인 TOTP에 영향을 주지 않는 규칙을 생성하려면 관리형 로그인 TOTP MFA를 위한 AWS WAF 웹 ACL 구성 섹션을 참조하세요.
- 웹 ACL의 규칙이 사용자 지정 블록 응답을 반환하면 Amazon Cognito는 해당 응답에 Content-Security-Policy 헤더를 추가합니다. 이 정책은 사용자 풀 도메인과 동일한 오리진의 스크립트와 Amazon Cognito가 AWS WAF 제공하는 CAPTCHA 및 챌린지 스크립트의 스크립트만 허용합니다. 사용자 지정 응답 본문이 사용자를 리디렉션하는 데 사용하는 스크립트를 포함하여 다른 인라인 또는 타사 스크립트를 차단합니다. 자세한 내용은 사용자 지정 블록 응답에 대한 Content-Security-Policy 섹션을 참조하세요.
- 웹 ACL의 규칙이 관리형 로그인 또는 클래식 호스팅 UI 요청에 대한 응답으로 HTTP 상태 코드가 500 이상인 사용자 지정 블록 응답을 반환하면 Amazon Cognito는 대신 `403 (Forbidden)` 상태 코드를 반환합니다. 자세한 내용은 사용자 지정 블록 응답의 상태 코드 섹션을 참조하세요.

AWS WAF 는 다음 엔드포인트에 대한 요청을 검사합니다.

관리형 로그인 및 클래식 호스팅 UI

사용자 풀 엔드포인트 및 관리형 로그인 참조의 모든 엔드포인트에 대한 요청입니다.

퍼블릭 API 작업

AWS 자격 증명을 사용하여 권한을 부여하지 않는 앱의 Amazon Cognito API에 대한 요청입니다. 여기에는 InitiateAuth, RespondToAuthChallenge 및 GetUser와 같은 API 작업이 포함됩니다. 범위 내에 있는 API 작업은 AWS 자격 증명으로 인증 AWS WAF 할 필요가 없습니다. 인증되지 않았거나 세션 문자열 또는 액세스 토큰으로 권한이 부여되었습니다. 자세한 내용은 권한 부여 모델별로 그룹화된 API 작업 목록 섹션을 참조하세요.

Count(개수), Allow(허용), Block(차단) 또는 규칙과 일치하는 요청에 대한 응답으로 CAPTCHA를 표시하는 규칙 작업을 사용하여 웹 ACL에서 규칙을 구성할 수 있습니다. 자세한 내용은 AWS WAF 개발자 안내서의 AWS WAF 규칙을 참조하세요. 규칙 작업에 따라 Amazon Cognito가 사용자에게 반환하는 응답을 사용자 지정할 수 있습니다.

Important

오류 응답을 사용자 지정하는 옵션은 API 요청 방법에 따라 달라집니다.

- 관리형 로그인 요청의 오류 코드와 응답 본문을 사용자 지정할 수 있습니다. Amazon Cognito는 500 이상의 사용자 지정 오류 코드를 로 반환합니다403. 자세한 내용은 사용자 지정 블록 응답의 상태 코드 섹션을 참조하세요. 관리형 로그인에서는 사용자가 해결할 수 있는 CAPTCHA만 제시할 수 있습니다.
- Amazon Cognito 사용자 풀 API로 요청하는 경우 차단 응답을 수신하는 요청의 응답 본문을 사용자 지정할 수 있습니다. 400~499 범위의 사용자 지정 오류 코드를 지정할 수도 있습니다.
- AWS Command Line Interface (AWS CLI) 및 AWS SDKs 반환합니다. `ForbiddenException`

사용자 지정 블록 응답에 대한 Content-Security-Policy

사용자 지정 블록 응답이 스크립트에 의존하는 경우이 정책은 스크립트의 동작 방식에 영향을 미칠 수 있습니다. 웹 ACL 규칙이 관리형 로그인 또는 클래식 호스팅 UI에 대한 사용자 지정 블록 응답을 반환하면 Amazon Cognito는 해당 응답에 Content-Security-Policy(CSP) 헤더를 추가합니다. CSP는 페이지가 로드할 수 있는 스크립트 및 기타 리소스를 제한하는 브라우저 보안 메커니즘입니다. 정책은 다음과 같이 응답을 제한합니다.

- 응답에는 사용자 풀 도메인과 동일한 오리진의 스크립트 및 기타 리소스가 포함될 수 있습니다.

- 응답에는 Amazon Cognito가 제공하는 AWS WAF CAPTCHA 및 챌린지 스크립트가 포함될 수 있습니다. Amazon Cognito는 AWS WAF CAPTCHA 및 챌린지 규칙 작업이 계속 작동하도록 이러한 스크립트를 허용합니다.

다른 인라인 스크립트와 사용자 풀 도메인 이외의 도메인에서 호스팅되는 스크립트는 실행되지 않습니다. 여기에는 사용자 지정 응답 본문이 사용자를 리디렉션하거나, 양식을 제출하거나, 네트워크 요청을 보내는 데 사용하는 스크립트가 포함됩니다.

사용자 지정 응답에서 사용자를 리디렉션해야 하는 경우 스크립트 기반 리디렉션 대신 HTTP 3xx 상태 코드와 `Location` 헤더를 반환합니다. 이 패턴은 스크립트 실행에 의존하지 않으며 CSP의 영향을 받지 않습니다.

이 동작은 웹 ACL을 사용자 풀과 연결할 수 AWS 리전 있는 모든에 적용됩니다.

사용자 지정 블록 응답의 상태 코드

Amazon Cognito는 AWS WAF 사용자 지정 블록 응답에서 `5xx` 상태 코드를 반환하지 않습니다. 웹 ACL 규칙이 상태 코드가 500 이상인 사용자 지정 블록 응답을 반환하면 Amazon Cognito는 대신 `403 (Forbidden)` 상태 코드로 응답합니다. 응답에는 여전히 사용자 지정 응답의 헤더와 본문이 포함됩니다. 이 동작은 다음과 같이 적용됩니다.

- 관리형 로그인 페이지 - 브라우저가 관리형 로그인 페이지를 로드하면 Amazon Cognito는 사용자 지정 응답의 헤더 및 본문과 `403` 함께를 반환합니다. 양식 제출과 같이 페이지가 다시 로드하지 않고 보내는 요청은 사용자 지정 헤더 또는 본문 없이 기본 `403` 블록 응답을 받습니다.
- 클래식 호스팅 UI 페이지 및 기타 사용자 풀 엔드포인트 - Amazon Cognito는 클래식 호스팅 UI 페이지에 대한 사용자 지정 응답의 헤더 및 본문과 `403` 함께를 반환합니다. ID 제공업체와 신뢰 당사자 엔드포인트는 사용자 풀이 관리형 로그인을 사용하든 클래식 호스팅 UI 브랜딩을 사용하든 동일한 방식으로 작동합니다.
- 사용자 풀 API - 사용자 풀 API는 사용자 지정 상태 코드가 400~499 범위에 있는 경우에만 사용자 지정 상태 코드를 반환하고 다른 상태 코드에 `403` 대해를 반환합니다.

웹 ACL을 사용자 풀과 연결

사용자 풀에서 웹 ACL을 사용하려면 AWS Identity and Access Management (IAM) 보안 주체에 다음과 같은 Amazon Cognito 및 AWS WAF 권한이 있어야 합니다. AWS WAF 권한에 대한 자세한 내용은 AWS WAF 개발자 안내서의 AWS WAF API 권한을 참조하세요.

JSON

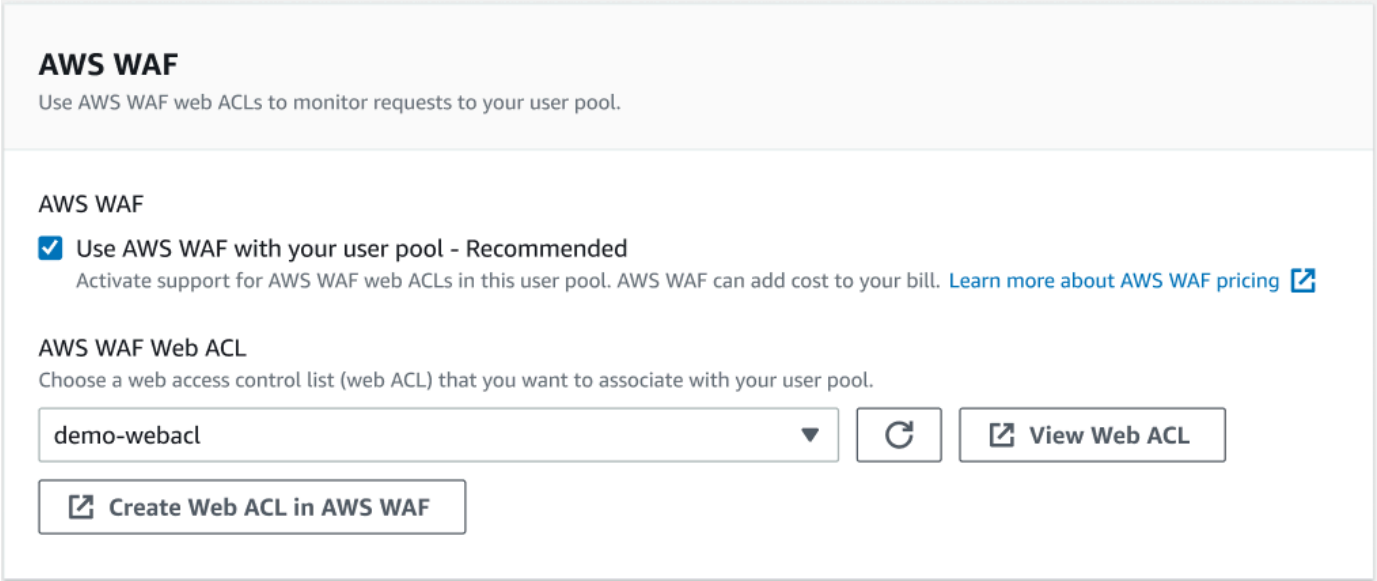
```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Sid": "AllowWebACLUserPool",
   "Effect": "Allow",
   "Action": [
    "cognito-idp:ListResourcesForWebACL",
    "cognito-idp:GetWebACLForResource",
    "cognito-idp:AssociateWebACL"
   ],
   "Resource": [
    "arn:aws:cognito-idp:*:123456789012:userpool/*"
   ]
  },
  {
   "Sid": "AllowWebACLUserPoolWAFv2",
   "Effect": "Allow",
   "Action": [
    "wafv2:ListResourcesForWebACL",
    "wafv2:AssociateWebACL",
    "wafv2:DisassociateWebACL",
    "wafv2:GetWebACLForResource"
   ],
   "Resource": "arn:aws:wafv2:*:123456789012:*/webacl/*/*"
  },
  {
   "Sid": "DisassociateWebACL1",
   "Effect": "Allow",
   "Action": "wafv2:DisassociateWebACL",
   "Resource": "*"
  },
  {
   "Sid": "DisassociateWebACL2",
   "Effect": "Allow",
   "Action": [
    "cognito-idp:DisassociateWebACL"
   ],
   "Resource": [
    "arn:aws:cognito-idp:*:123456789012:userpool/*"
   ]
```

```
    }
  ]
}
```

IAM 권한을 부여해야 하지만 나열된 작업은 권한 전용이며 API 작업에 해당하지 않습니다.

사용자 풀 AWS WAF 에 대해를 활성화하고 웹 ACL을 연결하려면

1. Amazon Cognito 콘솔에 로그인합니다.
2. 탐색 창에서 [사용자 풀(User Pools)]을 선택한 다음 편집할 사용자 풀을 선택합니다.
3. 보안 섹션에서 AWS WAF 탭을 선택합니다.
4. 편집을 선택합니다.
5. 사용자 풀 AWS WAF 과 함께 사용을 선택합니다.



6. 이미 생성한 AWS WAF 웹 ACL을 선택하거나 에서 웹 ACL 생성을 선택하여의 새 세션에 웹 ACL AWS WAF을 생성합니다 AWS Management Console. AWS WAF
7. 변경 사항 저장을 선택합니다.

웹 ACL을 AWS Command Line Interface 또는 SDK의 사용자 풀과 프로그래밍 방식으로 연결하려면 AWS WAF API의 AssociateWebACL을 사용합니다. Amazon Cognito에는 웹 ACL을 연결하는 별도의 API 작업이 없습니다.

AWS WAF 웹 ACLs 테스트 및 로깅

웹 ACL에서 규칙 작업을 개수로 설정하면가 규칙과 일치하는 요청 수에 요청을 AWS WAF 추가합니다. 사용자 풀로 웹 ACL을 테스트하려면 규칙 작업을 Count(개수)로 설정하고 각 규칙과 일치하는 요청 볼륨을 고려합니다. 예를 들어 Block(차단) 작업으로 설정하려는 규칙이 일반 사용자 트래픽으로 판단되는 많은 수의 요청과 일치하는 경우 규칙을 재구성해야 할 수 있습니다. 자세한 내용은 AWS WAF 개발자 안내서의 AWS WAF 보호 기능 테스트 및 조정을 참조하세요.

Amazon CloudWatch Logs 로그 그룹, Amazon Simple Storage Service(Amazon S3) 버킷 또는 Amazon Data Firehose에 요청 헤더를 로깅 AWS WAF 하도록를 구성할 수도 있습니다. `x-amzn-cognito-client-id` 및 `x-amzn-cognito-operation-name`을 통해 사용자 풀 API로 수행한 Amazon Cognito 요청을 식별할 수 있습니다. 관리형 로그인 요청에는 `x-amzn-cognito-client-id` 헤더만 포함됩니다. 자세한 내용은 AWS WAF 개발자 안내서의 웹 ACL 트래픽 로깅을 참조하세요.

AWS WAF 웹 ACLs은 모든 사용자 풀 기능 플랜에서 사용할 수 있습니다. AWS WAF 의 보안 기능은 Amazon Cognito 위협 방지를 보완합니다. 사용자 풀에서 두 기능을 모두 활성화할 수 있습니다. AWS WAF 는 사용자 풀 요청 검사에 대해 별도로 요금을 청구합니다. 자세한 내용은 AWS WAF 요금 섹션을 참조하세요.

로깅 AWS WAF 요청 데이터에는 로그를 대상으로 하는 서비스의 추가 요금이 부과됩니다. 자세한 내용은 AWS WAF 개발자 안내서의 웹 ACL 트래픽 정보 로깅 요금을 참조하세요.

사용자 풀 대/소문자 구분

에서 생성하는 Amazon Cognito 사용자 풀 AWS Management Console 은 기본적으로 대/소문자를 구분하지 않습니다. 사용자 풀이 대/소문자를 구분하지 않는 경우 user@example.com과 User@example.com은 동일한 사용자를 지칭합니다. 사용자 풀의 사용자 이름이 대/소문자를 구분하지 않는 경우 `preferred_username` 및 `email` 속성도 대/소문자를 구분하지 않습니다.

대소문자 구분은 속성 입력뿐만 아니라 출력에도 적용됩니다. 대소문자를 구분하지 않는 사용자 풀의 대소문자가 혼합된 속성 값은 사용자 풀 텍스트 출력에서 소문자로 변환됩니다. 사용자 풀 텍스트 출력의 예로는 userInfo 응답, GetUser 출력과 같은 사용자 쿼리 응답, Lambda 트리거에 대한 입력 이벤트가 있습니다.

사용자 풀 대/소문자 구분 설정을 고려하여 앱 코드에서 대체 사용자 속성을 기반으로 사용자를 식별합니다. 사용자 이름, 기본 설정 사용자 이름 또는 이메일 주소 속성의 대/소문자는 사용자 프로필마다 다를 수 있으므로 `sub` 속성을 참조하세요. 사용자 풀에 변경 불가능한 사용자 지정 속성을 생성하고 각 새 사용자 프로필의 속성에 고유한 식별자 값을 할당할 수도 있습니다. 사용자를 처음 생성할 때 생성된 변경 불가능한 사용자 지정 속성에 값을 쓸 수 있습니다.

> **Note**
>
> 사용자 풀의 대/소문자 구분 설정에 관계없이 Amazon Cognito에서는 SAML 자격 증명 공급자(IdP)의 페더레이션 사용자가 고유하고 대/소문자를 구분하는 `NameId` 또는 `sub` 클레임을 전달해야 합니다. 고유 식별자 대/소문자 구분 및 SAML IdP에 대한 자세한 내용은 SP 시작 SAML 로그인 구현 섹션을 참조하세요.

대/소문자를 구분하는 사용자 풀 생성

CreateUserPool과 같은 API 작업 및 AWS Command Line Interface (AWS CLI)를 사용하여 리소스를 생성하는 경우 부울 `CaseSensitive` 파라미터를 로 설정해야 합니다`false`. 이 설정은 대/소문자를 구분하지 않는 사용자 풀을 생성합니다. 값을 지정하지 않으면 `CaseSensitive`가 `true`를 기본값으로 사용합니다. Amazon Cognito 콘솔에서 생성하는 사용자 풀은 대/소문자를 구분하지 않습니다. 대/소문자를 구분하는 사용자 풀을 생성하려면 `CreateUserPool` 작업을 사용해야 합니다. 2020년 2월 12일 이전에는 플랫폼과 관계없이 사용자 풀이 기본적으로 대/소문자를 구분하도록 설정되었습니다.

AWS Management Console 의 로그인 메뉴와 DescribeUserPool의 `UsernameConfiguration` 속성에서 계정의 각 사용자 풀에 대한 대소문자 구분 설정을 검토할 수 있습니다.

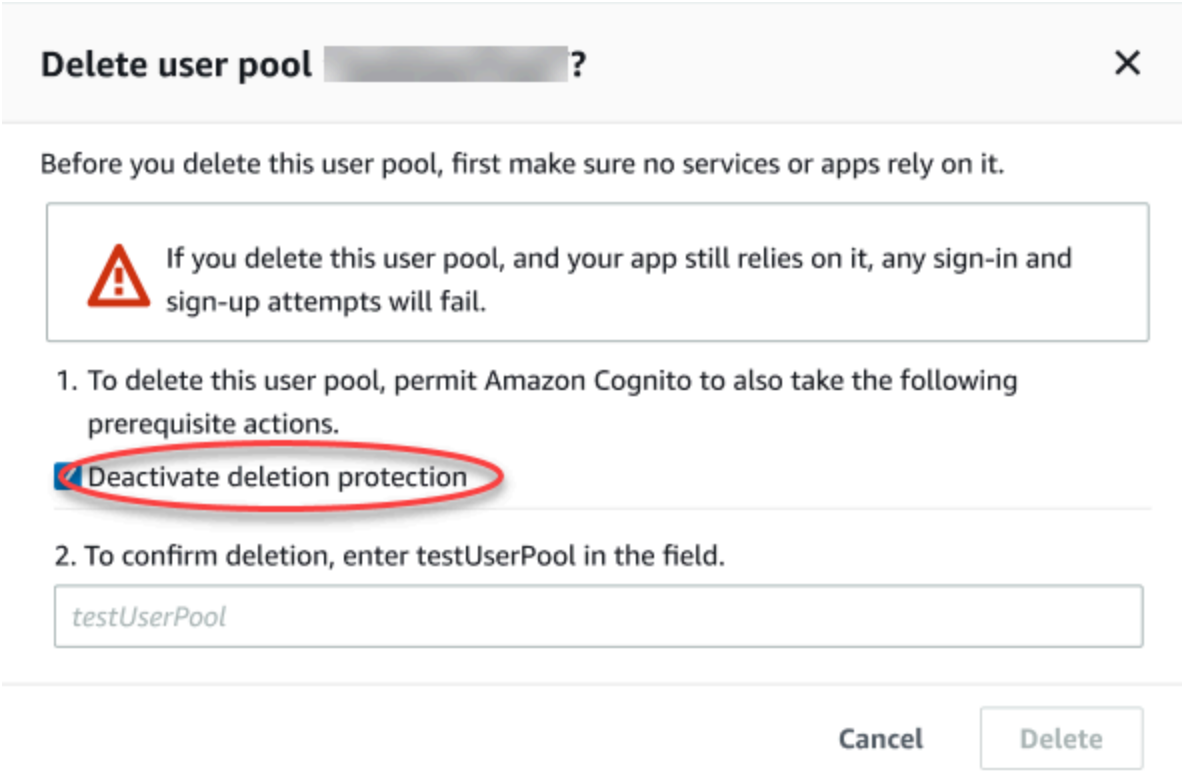
새 사용자 풀로 마이그레이션

사용자 프로필 간에 충돌이 발생할 수 있으므로 Amazon Cognito 사용자 풀을 대/소문자 구분에서 대/소문자 구분 안 함으로 변경할 수 없습니다. 대신 사용자를 새 사용자 풀로 마이그레이션할 수 있습니다. 대/소문자 관련 충돌을 해결하려면 마이그레이션 코드를 작성해야 합니다. 이 코드는 충돌이 감지될 때 고유한 새 사용자를 반환하거나 로그인 시도를 거부해야 합니다. 대/소문자를 구분하지 않는 새 사용자 풀에서 사용자 마이그레이션 Lambda 트리거를 할당합니다. AWS Lambda 함수는 대/소문자를 구분하지 않는 새 사용자 풀에서 사용자를 생성할 수 있습니다. 사용자가 대/소문자를 구분하지 않는 사용자 풀로 로그인에 실패하면 Lambda 함수가 대/소문자를 구분하는 사용자 풀에서 사용자를 찾아서 복제합니다. ForgotPassword 이벤트에 대해 사용자 마이그레이션 Lambda 트리거를 활성화할 수도 있습니다. Amazon Cognito는 로그인 또는 암호 복구 작업의 사용자 정보 및 이벤트 메타데이터를 Lambda 함수로 전달합니다. 함수가 대/소문자를 구분하지 않는 사용자 풀에서 새 사용자를 생성할 때 이벤트 데이터를 사용하여 사용자 이름과 이메일 주소 간의 충돌을 관리할 수 있습니다. 이러한 충돌은 대/소문자를 구분하는 사용자 풀에서는 고유하지만 대/소문자를 구분하지 않는 사용자 풀에서는 동일한 사용자 이름과 이메일 주소 간에 발생합니다.

Amazon Cognito 사용자 풀 간에 사용자 Lambda 마이그레이션 트리거를 사용하는 방법에 대한 자세한 내용은 AWS 블로그의 사용자에서 Amazon Cognito 사용자 풀로 마이그레이션을 참조하세요.

사용자 풀 삭제 방지

관리자가 실수로 사용자 풀을 삭제하지 않도록 하려면 삭제 방지를 활성화합니다. 삭제 방지 기능이 활성화된 상태에서는 사용자 풀을 삭제할 것인지 확인한 후에만 사용자 풀을 삭제할 수 있습니다. 에서 사용자 풀을 삭제하면 삭제 방지 기능을 동시에 비활성화AWS Management Console할 수 있습니다. 다음 이미지와 같이 삭제 방지를 비활성화하라는 메시지를 수락하고 삭제 의사를 확인하면 Amazon Cognito가 사용자 풀을 삭제합니다.



Amazon Cognito API 요청을 사용하여 사용자 풀을 삭제하려면 먼저 UpdateUserPool 요청에서 `DeletionProtection`을 `Inactive`로 변경해야 합니다. 삭제 방지를 비활성화하지 않으면 Amazon Cognito에서 `InvalidParameterException` 오류를 반환합니다. 삭제 방지를 비활성화한 후 DeleteUserPool 요청에서 사용자 풀을 삭제할 수 있습니다.

AWS Management Console에서 새 사용자 풀을 생성하면 기본적으로 Amazon Cognito가 Deletion protection(삭제 방지)을 활성화합니다. `CreateUserPool` API를 이용해 사용자 풀을 생성하면 기본적으로 삭제 방지가 비활성화됩니다. AWS CLI또는 AWSSDK를 사용하여 생성한 사용자 풀에서이 기능을 사용하려면 `DeletionProtection` 파라미터를 로 설정합니다`True`.

Amazon Cognito 콘솔의 설정 메뉴에 있는 삭제 방지 컨테이너에서 삭제 방지 상태를 활성화하거나 비활성화할 수 있습니다.

삭제 방지를 구성하려면

1. Amazon Cognito 콘솔로 이동합니다. 자격 AWS증명을 입력하라는 메시지가 표시될 수 있습니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. 설정 메뉴를 선택하고 사용자 풀 보안 탭으로 이동한 다음 삭제 방지를 찾습니다. 활성화 또는 비활성화를 선택합니다.
5. 다음 대화 상자에서 선택을 확인합니다.

사용자 존재 오류 응답 관리

Amazon Cognito는 사용자 풀에서 반환하는 오류 응답을 사용자 지정하도록 지원합니다. 사용자 지정 오류 응답은 사용자 생성, 인증, 암호 복구 및 확인 작업에 사용할 수 있습니다.

사용자 풀 앱 클라이언트의 `PreventUserExistenceErrors` 설정을 사용하여 사용자 존재 여부와 관련한 오류를 사용하거나 사용하지 않도록 설정합니다. Amazon Cognito 사용자 풀 API로 새로운 앱 클라이언트를 생성할 때 기본적으로 `PreventUserExistenceErrors`는 `LEGACY`이거나 비활성화됩니다. Amazon Cognito 콘솔에서는 `PreventUserExistenceErrors`의 `ENABLED` 설정인 사용자 존재 오류 방지 옵션이 기본적으로 선택되어 있습니다. `PreventUserExistenceErrors` 구성을 업데이트하려면 다음 중 하나를 수행합니다.

- UpdateUserPoolClient API 요청에서 `ENABLED`와 `LEGACY` 사이의 `PreventUserExistenceErrors` 값을 변경합니다.
- Amazon Cognito 콘솔에서 앱 클라이언트를 편집하고 선택한 (`ENABLED`)와 선택 취소된 (`LEGACY`) 사이에서 사용자 존재 오류 방지의 상태를 변경합니다.

이 속성의 값이 `LEGACY`인 경우 사용자가 사용자 풀에 없는 사용자 이름으로 로그인을 시도하면 앱 클라이언트가 `UserNotFoundException` 오류 응답을 반환합니다.

이 속성의 값이 `ENABLED`인 경우 앱 클라이언트는 `UserNotFoundException` 오류와 함께 사용자 풀에 사용자 계정이 존재하지 않음을 공개하지 않습니다. `ENABLED`의 `PreventUserExistenceErrors` 구성은 존재하지 않는 사용자 이름에 대한 요청을 제출할 때 다음과 같은 영향을 미칩니다.

- Amazon Cognito는 API 요청에 대해 비특정 정보로 응답하며, 그렇지 않으면 유효한 사용자가 존재한다는 사실이 공개될 수 있습니다.

- Amazon Cognito는 암호를 잊어버린 경우와 선택 기반 인증(USER_AUTH)을 제외한 인증 흐름이 있는 인증 요청에 대해 일반적인 인증 실패 응답을 반환합니다(예: USER_SRP_AUTH 또는 CUSTOM_AUTH). 오류 응답은 사용자 이름 또는 암호가 잘못되었음을 알려줍니다.
- Amazon Cognito는 사용자 풀에 허용되는 문제 유형 중에서 임의로 선택하여 선택 기반 인증 요청에 응답합니다. 사용자 풀은 패스키, 일회용 암호 또는 암호 문제를 반환할 수 있습니다.
- Amazon Cognito 계정 확인 및 암호 복구 API의 동작은 코드가 시뮬레이션된 전송 매체로 전송되었음을 나타내는 응답을 반환하고 `InvalidParameterException` 오류를 반환하는 것을 번갈아 수행합니다.

다음 정보는 `PreventUserExistenceErrors`가 `ENABLED`로 설정된 경우 사용자 풀 작업의 동작을 자세히 설명합니다.

인증 및 사용자 생성 작업

사용자 이름-암호 및 SRP(보안 원격 암호) 인증에서 오류 응답을 구성할 수 있습니다. 사용자 지정 인증을 통해 반환되는 오류를 사용자 지정할 수도 있습니다. 선택 기반 인증은 `PreventUserExistenceErrors` 구성의 영향을 받지 않습니다.

인증 흐름의 사용자 존재 공개 세부 정보

선택 기반 인증

> USER_AUTH 선택 기반 인증 흐름에서 Amazon Cognito는 사용자 풀 구성 및 사용자의 속성에 따라 사용 가능한 기본 인증 요소에서 문제를 반환합니다. 이 인증 흐름은 암호, 보안 원격 암호(SRP), WebAuthn(패스키), SMS 일회용 암호(OTP) 또는 이메일 OTP 문제를 반환할 수 있습니다. `PreventUserExistenceErrors` 활성 상태에서 Amazon Cognito는 존재하지 않는 사용자에게 사용 가능한 인증 형식 중 하나 이상을 완료하기 위한 문제를 발행합니다. `PreventUserExistenceErrors` 비활성 상태인 경우 Amazon Cognito는 `UserNotFound` 예외를 반환합니다.

사용자 이름 및 암호 인증

> 인증 흐름 `ADMIN_USER_PASSWORD_AUTH`, `USER_PASSWORD_AUTH` 및 `USER_AUTH`의 `PASSWORD` 흐름은 `PreventUserExistenceErrors`이 활성 상태일 때 메시지 `Incorrect username or password`와 함께를 `NotAuthorizedException`을 반환합니다. `PreventUserExistenceErrors`가 비활성 상태이면 이러한 흐름은 `UserNotFoundException`를 반환합니다.

Secure Remote Password(SRP) 기반 인증

가장 좋은 방법은 이메일 주소, 전화번호 또는 선호하는 사용자 이름 별칭 속성 없이 USER_SRP_AUTH를 사용하는 PreventUserExistenceErrors 또는 사용자 풀에서 USER_AUTH의 PASSWORD_SRP 흐름으로만 구현하는 것입니다. 별칭 속성이 있는 사용자는 SRP 인증 흐름에서 사용자 존재 억제가 적용되지 않을 수 있습니다. 사용자 이름 암호 인증 흐름(ADMIN_USER_PASSWORD_AUTH, USER_PASSWORD_AUTH, 및 USER_AUTH PASSWORD 문제)은 별칭 속성에서 사용자의 존재를 완전히 억제합니다.

누군가 앱 클라이언트에 알려지지 않은 사용자 이름으로 SRP 로그인을 시도하면 Amazon Cognito는 RFC 5054에 설명된 대로 첫 번째 단계에서 시뮬레이션된 응답을 반환합니다. Amazon Cognito가 동일한 사용자 이름 및 사용자 풀 조합에 대해 동일한 솔트와 내부 사용자 ID(UUID 형식)를 반환합니다. RespondToAuthChallenge 요청을 암호 증명과 함께 보내면, Amazon Cognito는 사용자 이름 또는 암호가 올바르지 않을 때 일반 NotAuthorizedException 오류를 반환합니다. SRP 인증 구현에 대한 자세한 내용은 영구 암호 및 보안 페이로드로 로그인 섹션을 참조하세요.

> **Note**
>
> 검증 기반 별칭 속성을 사용하고 있으며 변경 불가능한 사용자 이름의 형식이 UUID로 지정되지 않았다면, 사용자 이름 및 암호 인증을 사용하여 일반 응답을 시뮬레이션할 수 있습니다.

사용자 지정 인증 문제 Lambda 트리거

Amazon Cognito는 사용자가 CUSTOM_AUTH 인증 흐름으로 로그인하려고 하지만 사용자 이름을 찾을 수 없는 경우 Lambda가 트리거하는 사용자 지정 인증 문제를 호출합니다. 입력 이벤트에는 존재하지 않는 사용자의 값이 true인 UserNotFound라는 부울 파라미터가 포함됩니다. 이 파라미터는 사용자 풀이 사용자 지정 인증 아키텍처를 구성하는 인증 문제 Lambda 함수를 생성, 정의 및 확인하는 데 보내는 요청 이벤트에 나타납니다. Lambda 함수의 로직에서 이 지표를 검사하면 존재하지 않는 사용자의 사용자 지정 인증 문제를 시뮬레이션할 수 있습니다.

사전 인증 Lambda 트리거

Amazon Cognito는 사용자가 로그인을 시도했지만 사용자 이름을 찾을 수 없는 경우 사전 인증 트리거를 호출합니다. 입력 이벤트에는 존재하지 않는 사용자의 값이 true인 UserNotFound 파라미터가 포함됩니다.

다음 목록은 PreventUserExistenceErrors가 사용자 계정 생성에 미치는 영향을 설명합니다.

사용자 생성 흐름의 사용자 존재 공개 세부 정보

SignUp

SignUp 작업은 이미 사용 중인 사용자 이름인 경우 항상 UsernameExistsException을 반환합니다. 앱에서 사용자를 등록할 때 Amazon Cognito가 이메일 주소 및 전화번호에 대한 UsernameExistsException 오류를 반환하지 않게 하려면, 확인 기반 별칭 속성을 사용해야 합니다. 별칭에 대한 자세한 내용은 로그인 속성 사용자 지정을 참조하십시오.

Amazon Cognito가 SignUp API 요청을 사용하여 사용자 풀에서 사용자를 검색하지 못하게 하는 예시는 가입 시 이메일 주소 및 전화 번호 관련 UsernameExistsException 오류 방지에서 확인할 수 있습니다.

가져온 사용자

PreventUserExistenceErrors를 사용하는 경우 가져온 사용자를 인증하는 중에 PasswordResetRequiredException이 반환되지 않고 대신 사용자 이름 또는 암호가 잘못되었음을 나타내는 일반 NotAuthorizedException 오류가 반환됩니다. 자세한 내용은 가져온 사용자에게 암호 재설정 요구 섹션을 참조하세요.

사용자 마이그레이션 Lambda 트리거

Amazon Cognito는 Lambda 트리거에서 원래 이벤트 컨텍스트에 빈 응답이 설정된 경우 존재하지 않는 사용자에 대해 시뮬레이션된 응답을 반환합니다. 자세한 내용은 사용자 마이그레이션 Lambda 트리거를 사용하여 사용자 가져오기 섹션을 참조하세요.

가입 시 이메일 주소 및 전화 번호 관련 **UsernameExistsException** 오류 방지

다음 예제는 사용자 풀에서 별칭 속성을 구성할 때 중복된 이메일 주소와 전화번호 때문에 SignUp API 요청에 대한 응답으로 UsernameExistsException 오류가 생성되지 않게 하는 방법을 보여줍니다. 이메일 주소나 전화 번호를 별칭 속성으로 사용하여 사용자 풀을 생성해야 합니다. 자세한 내용은 사용자 풀 속성의 로그인 특성 사용자 지정 섹션을 참조하세요.

1. Jie는 새 사용자 이름을 등록하고 이메일 주소(jie@example.com)도 입력합니다. Amazon Cognito가 이 이메일 주소로 확인 코드를 전송합니다.

   예제 AWS CLI 명령

   ```
   aws cognito-idp sign-up --client-id 1234567890abcdef0 --username jie --password
    PASSWORD --user-attributes Name="email",Value="jie@example.com"
   ```

응답의 예

```
{
    "UserConfirmed": false,
    "UserSub": "<subId>",
    "CodeDeliveryDetails": {
        "AttributeName": "email",
        "Destination": "j****@e****",
        "DeliveryMedium": "EMAIL"
    }
}
```

2. Jie는 전송된 코드를 입력하여 이메일 주소의 소유권을 확인합니다. 이로써 사용자 등록이 완료됩니다.

   예제 AWS CLI 명령

   ```
   aws cognito-idp confirm-sign-up --client-id 1234567890abcdef0 --username=jie --
   confirmation-code xxxxxx
   ```

3. Shirley는 새 사용자 계정을 등록하고 이메일 주소(`jie@example.com`)를 입력합니다. Amazon Cognito는 `UsernameExistsException` 오류를 반환하지 않고, Jie의 이메일 주소로 확인 코드를 보냅니다.

   예제 AWS CLI 명령

   ```
   aws cognito-idp sign-up --client-id 1234567890abcdef0 --username shirley --password
    PASSWORD --user-attributes Name="email",Value="jie@example.com"
   ```

   응답의 예

   ```
   {
       "UserConfirmed": false,
       "UserSub": "<new subId>",
       "CodeDeliveryDetails": {
           "AttributeName": "email",
           "Destination": "j****@e****",
           "DeliveryMedium": "EMAIL"
       }
   }
   ```

4. 다른 시나리오에서는 Shirley 가 jie@example.com에 대한 소유권을 가집니다. 셜리는 Amazon Cognito가 Jie의 이메일 주소로 보낸 코드를 검색하고 계정 확인을 시도합니다.

   예제 AWS CLI 명령

   ```
   aws cognito-idp confirm-sign-up --client-id 1234567890abcdef0 --username=shirley --
   confirmation-code xxxxxx
   ```

   응답의 예

   ```
   An error occurred (AliasExistsException) when calling the ConfirmSignUp operation: An
    account with the email already exists.
   ```

jie@example.com이 기존 사용자에게 할당되었지만 Amazon Cognito는 Shirley의 aws cognito-idp sign-up 요청에 대해 오류를 반환하지 않습니다. Shirley는 Amazon Cognito가 오류 응답을 반환하기 전에 이메일 주소의 소유권을 입증해야 합니다. 별칭 속성이 있는 사용자 풀에서는 이 동작 때문에, 지정된 이메일 주소나 전화 번호를 가진 사용자의 존재 여부를 공개 SignUp API로 확인할 수 없습니다.

이 동작은 다음 예에서처럼 Amazon Cognito가 기존 사용자 이름과 함께 SignUp 요청에 반환하는 응답과는 다릅니다. Shirley는 이 응답을 통해 사용자 이름이 jie인 사용자가 이미 존재함을 알게 되지만, 이 사용자와 관련된 이메일 주소나 전화번호는 알지 못합니다.

CLI 명령 예

```
aws cognito-idp sign-up --client-id 1example23456789 --username jie --password PASSWORD
      --user-attributes Name="email",Value="shirley@example.com"
```

응답의 예

```
An error occurred (UsernameExistsException) when calling the SignUp operation: User
 already exists
```

암호 재설정 작업

Amazon Cognito는 사용자 존재 오류를 방지하면 사용자 암호 재설정 작업에 다음과 같은 응답을 반환합니다.

ForgotPassword

사용자를 찾을 수 없거나, 사용자가 비활성화되었거나, 암호를 복구할 수 있는 확인된 전달 메커니즘이 없는 경우 Amazon Cognito는 사용자에 대해 시뮬레이션된 전송 미디어와 함께 `CodeDeliveryDetails`를 반환합니다. 시뮬레이션된 전송 미디어는 사용자 풀의 입력 사용자 이름 형식 및 검증 설정에 따라 결정됩니다.

ConfirmForgotPassword

Amazon Cognito는 존재하지 않거나 사용 중지된 사용자에 대해 `CodeMismatchException` 오류를 반환합니다. `ForgotPassword`를 사용할 때 코드가 요청되지 않은 경우 Amazon Cognito가 `ExpiredCodeException` 오류를 반환합니다.

확인 작업

Amazon Cognito는 사용자 존재 오류를 방지하는 경우 사용자 확인 및 검증 작업에 다음과 같은 응답을 반환합니다.

ResendConfirmationCode

Amazon Cognito는 사용 중지되거나 존재하지 않는 사용자에 대해 `CodeDeliveryDetails` 오류를 반환합니다. Amazon Cognito는 기존 사용자의 이메일 또는 전화 번호로 확인 코드를 보냅니다.

ConfirmSignUp

`ExpiredCodeException`은 코드가 만료된 경우 반환됩니다. Amazon Cognito는 사용자에게 권한이 부여되지 않은 경우 `NotAuthorizedException`을 반환합니다. 서버에서 예상하는 코드와 일치하지 않는 경우 Amazon Cognito는 `CodeMismatchException`을 반환합니다.

B2B 멀티 테넌트 애플리케이션 구축

이 섹션에서는 동일한 리전과 내의 Amazon Cognito 리소스 간에 테넌트를 분리하기 위해 구현할 수 있는 방법을 설명합니다 AWS 계정. 테넌트를 둘 이상의 AWS 계정 또는 리전으로 분할하고 각 테넌트에 고유한 할당량을 부여할 수도 있습니다. 다중 리전 다중 테넌시의 다른 이점으로는 가능한 최고 수준의 격리, 전 세계에 분산된 사용자를 위한 가장 짧은 네트워크 전송 시간, 조직의 기존 배포 모델 준수 등이 있습니다.

Amazon Cognito 사용자 풀은 Amazon Cognito 할당량 내에 유지되어야 하는 대량의 요청을 생성하는 멀티 테넌트 애플리케이션과 함께 작동합니다. 고객 기반이 증가할 때 이 용량을 스케일 업하려면 추가 할당량 용량을 구매할 수 있습니다.

> **Note**
>
> Amazon Cognito 할당량은 AWS 계정 및 별로 적용됩니다 AWS 리전. 이러한 할당량은 애플리케이션의 모든 테넌트 간에 공유됩니다. Amazon Cognito 서비스 할당량을 검토하고 할당량이 애플리케이션의 예상 볼륨 및 예상 테넌트 수를 충족하는지 확인합니다.

단일 리전 다중 테넌시도 고객과 관리자에게 이점을 제공할 수 있습니다.

다음 목록에서는 공유 리소스를 사용하는 다중 테넌시의 몇 가지 이점에 대해 설명합니다.

다중 테넌시의 이점

일반 사용자 디렉터리

다중 테넌시는 고객이 둘 이상의 애플리케이션에 계정을 가지고 있는 모델을 지원합니다. 일관된 단일 사용자 풀 프로파일에 타사 공급자의 ID를 연결할 수 있습니다. 사용자 프로필이 테넌트에 고유한 경우 단일 사용자 풀을 사용하는 모든 멀티 테넌시 전략에는 사용자 관리에 대한 하나의 진입점이 있습니다.

일반 보안

공유 사용자 풀에서는 단일 보안 표준을 생성하고 모든 테넌트에 동일한 위협 방지, 다중 인증(MFA) 및 AWS WAF 표준을 적용할 수 있습니다. AWS WAF 웹 ACL은 웹 ACL을 연결하는 리소스와 AWS 리전 동일한에 있어야 하므로 다중 테넌시는 복잡한 리소스에 대한 공유 액세스를 제공합니다. 다중 리전 Amazon Cognito 애플리케이션에서 일관된 보안 구성을 유지하려면 리소스 간에 구성을 복제하는 운영 표준을 적용해야 합니다.

일반 사용자 지정

를 사용하여 사용자 풀 및 자격 증명 풀을 사용자 지정할 수 있습니다 AWS Lambda. 사용자 풀의 Lambda 트리거 구성과 ID 풀의 Amazon Cognito 이벤트는 복잡해질 수 있습니다. Lambda 함수는 사용자 풀 또는 자격 증명 풀 AWS 리전 과 동일한에 있어야 합니다. 공유 Lambda 함수는 리전 내 사용자 지정 인증 흐름, 사용자 마이그레이션, 토큰 생성 및 기타 함수에 대한 표준을 적용할 수 있습니다.

일반 메시징

Amazon Simple Notification Service(Amazon SNS)는 SMS 메시지를 사용자에게 전송하기 전에 리전에서 추가 구성이 필요합니다. 리전 내에 포함된 Amazon Simple Email Service(Amazon SES) 확인 ID 및 도메인을 사용하여 이메일 메시지를 보낼 수 있습니다.

다중 테넌시를 사용하면 모든 테넌트 간에 이 구성 및 유지 관리 오버헤드를 공유할 수 있습니다. Amazon SNS 및 Amazon SES를 모든 AWS 리전에서 사용할 수 있는 것은 아니기 때문에 리전 간에 리소스를 분할하려면 추가 고려 사항이 필요합니다.

사용자 지정 메시징 공급자를 사용하면 단일 Lambda 함수의 일반적인 사용자 지정을 통해 메시지 전송을 관리할 수 있습니다.

관리형 로그인은 이미 인증된 사용자를 인식하도록 브라우저에 세션 쿠키를 설정합니다. 사용자 풀에서 로컬 사용자를 인증할 때 이러한 클라이언트의 세션 쿠키가 해당 사용자를 동일한 사용자 풀의 모든 앱 클라이언트에 대해 인증합니다. 로컬 사용자는 외부 IdP를 통한 페더레이션 없이 사용자 풀 디렉터리에만 존재합니다. 이러한 세션 쿠키는 1시간 동안 유효합니다. 세션 쿠키 기간은 변경할 수 없습니다.

호스팅 UI 세션 쿠키를 사용하여 앱 클라이언트 간에 로그인을 방지하는 두 가지 방법이 있습니다.

- 사용자를 테넌트당 사용자 풀로 구분합니다.
- 호스팅 UI 로그인을 Amazon Cognito 사용자 풀 API 로그인으로 바꿉니다.

주제

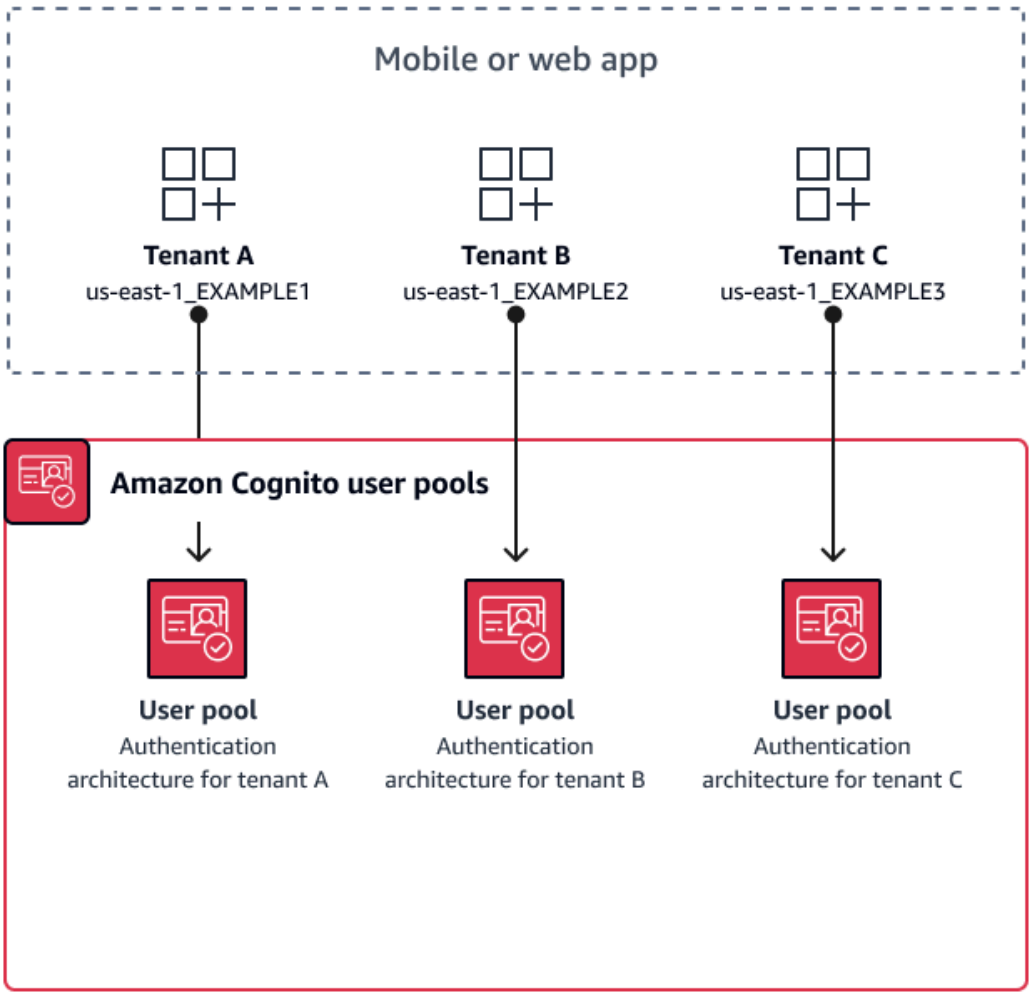
- 사용자 풀 다중 테넌시 모범 사례
- 앱 클라이언트 다중 테넌시 모범 사례
- 사용자 그룹 다중 테넌시 모범 사례
- 사용자 지정 속성 다중 테넌시 모범 사례
- 사용자 지정 범위 다중 테넌시 모범 사례
- 멀티 테넌시 보안 권장 사항

사용자 풀 다중 테넌시 모범 사례

앱의 각 테넌트에 대한 사용자 풀을 생성합니다. 이 접근 방식은 각 테넌트별로 최대한의 격리를 제공합니다. 각 테넌트별로 서로 다른 구성을 구현할 수 있습니다. 사용자 풀별 테넌트 격리를 통해 사용자와 테넌트 간의 매핑을 유연하게 설정할 수 있습니다. 동일한 사용자에 대해 여러 개의 프로필을 생성할 수 있습니다. 그러나 각 사용자는 액세스 권한이 있는 각 테넌트별로 개별적으로 가입해야 합니다.

이 접근 방식을 사용하면 각 테넌트에 대해 호스팅 UI를 독립적으로 설정하고 사용자를 테넌트별 애플리케이션 인스턴스로 리디렉션할 수 있습니다. 이 접근 방식을 사용하여 Amazon API Gateway와 같은 백엔드 서비스와 통합할 수도 있습니다.

다음 다이어그램은 전용 사용자 풀이 있는 각 테넌트를 보여줍니다.



사용자 풀 다중 테넌시를 구현해야 하는 경우

격리 및 사용자 지정이 주요 관심사인 경우. 여러 사용자 풀이 있는 아키텍처에서는 사용자와 테넌트 간의 관계가 복잡할 수 있습니다. 두 개의 교육 테넌트가 있는 예를 생각해 보세요. 동일한 사용자가 한 앱에서는 액세스가 제한된 학생이고 다른 앱에서는 높은 수준의 권한을 가진 교사일 수 있습니다. 한 앱에는 MFA가 필요하지만 다른 앱에는 필요하지 않거나 암호 정책이 다를 수 있습니다. 로컬 사용자는 관리형 로그인을 사용하여 사용자 풀의 여러 앱 클라이언트에 로그인할 수 있으므로 둘 이상의 테넌트가 관리형 로그인으로 로그인하기를 원하는 경우에도 사용자 풀 다중 테넌시가 이상적입니다.

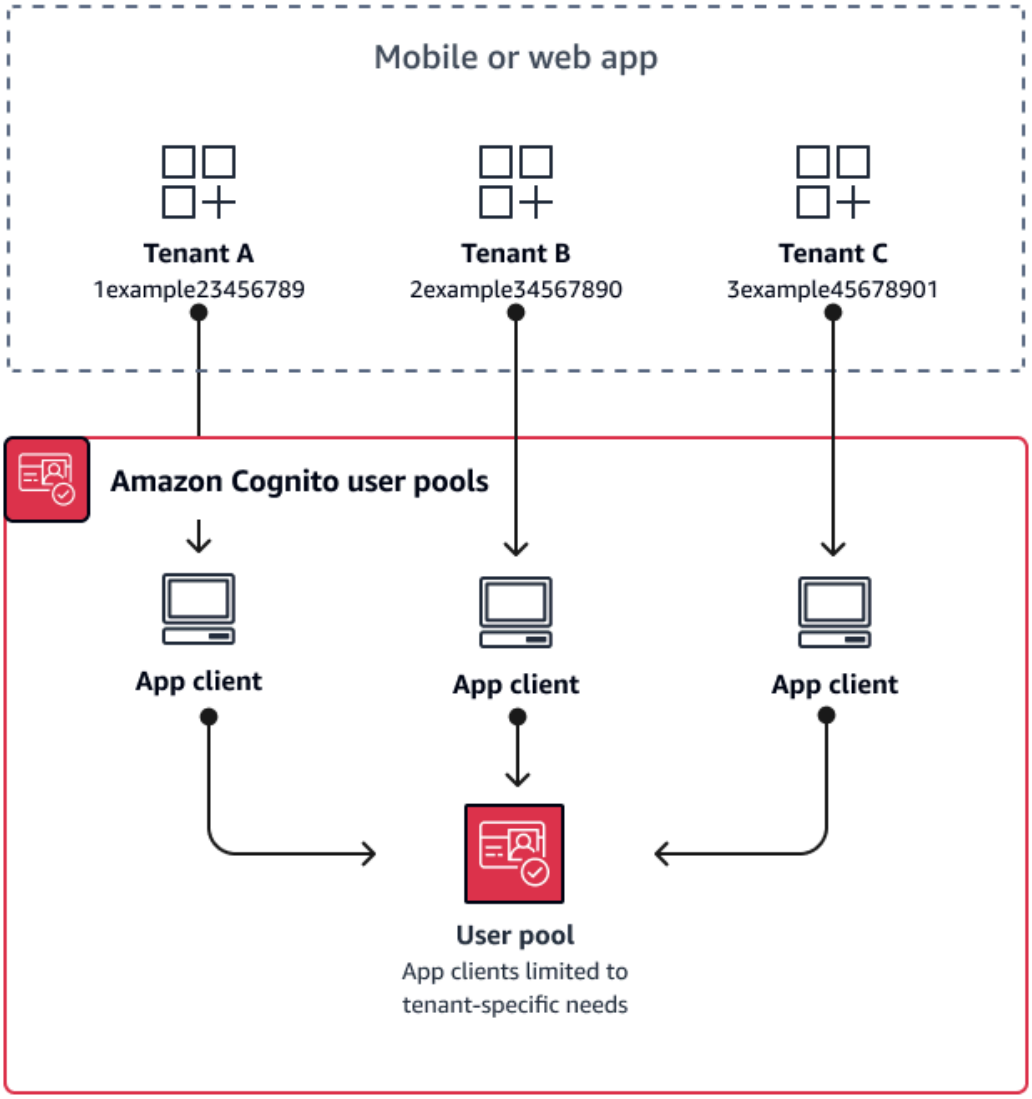
노력 수준

이 방법을 사용하기 위해서는 많은 개발 및 운영상의 작업이 요구됩니다. 앱 제품군에 대해 일관되고 예측 가능한 결과를 보장하려면 Amazon Cognito 리소스를 자동화 도구와 통합하고 인증 아키텍처가 점점 더 복잡해짐에 따라 기준을 유지 관리해야 합니다. 하나의 앱 시작 위치를 생성하려면 사용자를 올바른 리소스로 라우팅하는 초기 결정을 캡처하기 위해 사용자 인터페이스(UI) 요소를 구축해야 합니다.

앱 클라이언트 다중 테넌시 모범 사례

앱에서 각 테넌트에 대한 앱 클라이언트를 생성합니다. 앱 클라이언트 다중 테넌시를 사용하면 테넌트 연결 앱 클라이언트에 모든 사용자를 할당하고 단일 사용자 프로필을 유지할 수 있습니다. 사용자 풀의 ID 제공업체(IdPs) 중 일부 또는 전부를 앱 클라이언트에 할당할 수 있으므로 테넌트 앱 클라이언트는 테넌트별 IdP를 사용하여 로그인을 허용할 수 있습니다. 사용자가 여러 테넌트에 있는 경우 일관된 사용자 환경을 위해 해당 프로필을 여러 IdP와 연결할 수 있습니다.

다음 다이어그램은 공유 사용자 풀에 전용 앱 클라이언트가 있는 각 테넌트를 보여줍니다.



앱 클라이언트 다중 테넌시를 구현해야 하는 경우

Lambda 트리거, 암호 정책, 이메일 및 SMS 메시지의 콘텐츠 및 전송 방법과 같은 사용자 풀 수준에서 설정에 대한 범용 구성을 선택할 수 있는 경우. 공유 사용자 풀의 사용자는 모든 앱 클라이언트에 로그인할 수 있으므로 앱 클라이언트 다중 테넌시는 앱 클라이언트별 IdP 또는 Amazon Cognito 사용자 풀 API로 로그인하는 데 적합합니다. 또한 앱 클라이언트 다중 테넌시는 사용자가 여러 애플리케이션 간에 전환할 수 있도록 허용하려는 일대다 환경에 적합합니다.

노력 수준

앱 클라이언트 다중 테넌시에는 적당한 노력이 필요합니다. 앱 클라이언트 다중 테넌시의 주요 과제는 테넌트가 호스팅 UI 쿠키를 제시하고 앱 간에 전환할 수 있는 기능입니다. 앱 클라이언트 다중 테넌시 아키텍처에서는 격리가 필요한 호스팅 UI 로그인을 피합니다. 앱 클라이언트 로직이 내장된 상태에서 모바일 앱 또는 링크를 웹 앱에 배포하거나 사용자의 테넌시를 결정하는 초기 UI 요소를 구축할 수 있습니다. 여러 사용자 풀 및 ID 풀에서 구성을 표준화하고 유지할 필요가 없으므로 노력 수준이 낮습니다.

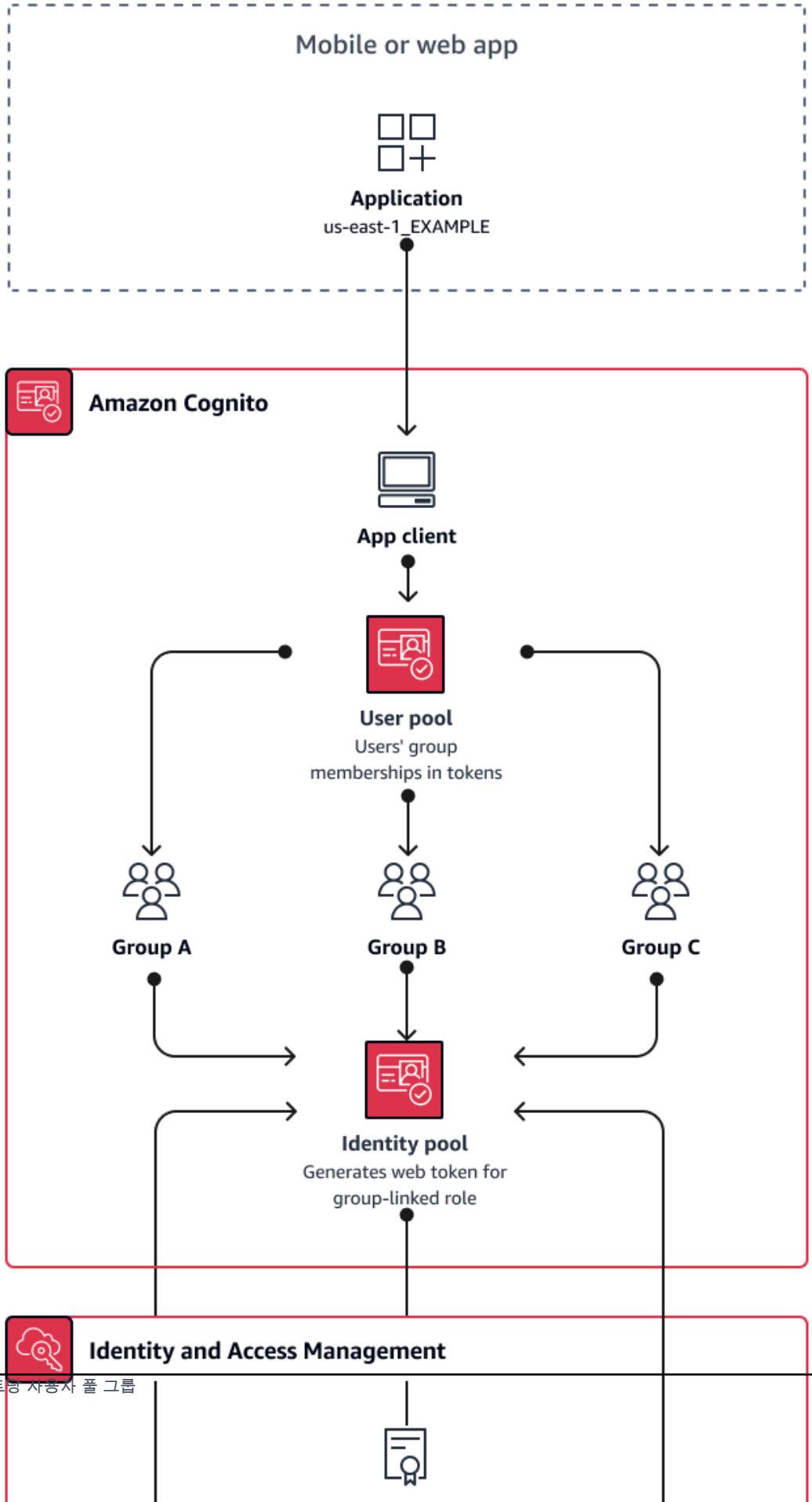
사용자 그룹 다중 테넌시 모범 사례

그룹 기반 다중 테넌시는 아키텍처에 ID 풀이 있는 Amazon Cognito 사용자 풀이 필요한 경우에 가장 적합합니다.

사용자 풀 ID 및 액세스 토큰에는 `cognito:groups` 클레임이 포함됩니다. 또한 ID 토큰에는 `cognito:roles` 및 `cognito:preferred_role` 클레임이 포함됩니다. 앱의 기본 인증 결과가 ID 풀의 AWS 임시 자격 증명인 경우 사용자의 그룹 멤버십은 사용자가 받는 IAM 역할 및 권한을 결정할 수 있습니다.

예를 들어, 각 애플리케이션 자산이 자체 Amazon S3 버킷에 저장하는 세 개의 테넌트를 고려해 보세요. 각 테넌트의 사용자를 연결된 그룹에 할당하고, 그룹에 대한 기본 역할을 구성하고, 해당 역할에 버킷에 대한 읽기 액세스 권한을 부여합니다.

다음 다이어그램은 IAM 역할에 대한 자격을 결정하는 사용자 풀의 전용 그룹과 함께 앱 클라이언트와 사용자 풀을 공유하는 테넌트를 보여줍니다.


Mobile or web app
Application
us-east-1_EXAMPLE
Amazon Cognito
App client
User pool
Users' group
memberships in tokens
Group A
Group B
Group C
Identity pool
Generates web token for
group-linked role
Identity and Access Management

그룹 다중 테넌시를 구현해야 하는 경우

AWS 리소스에 대한 액세스가 주요 관심사인 경우. Amazon Cognito 사용자 풀의 그룹은 역할 기반 액세스 제어(RBAC)를 위한 메커니즘입니다. 사용자 풀에서 여러 그룹을 구성하고 그룹 우선 순위로 복잡한 RBAC 결정을 내릴 수 있습니다. ID 풀은 우선 순위가 가장 높은 역할, 그룹 클레임의 모든 역할 또는 사용자 토큰의 다른 클레임에 대한 자격 증명을 할당할 수 있습니다.

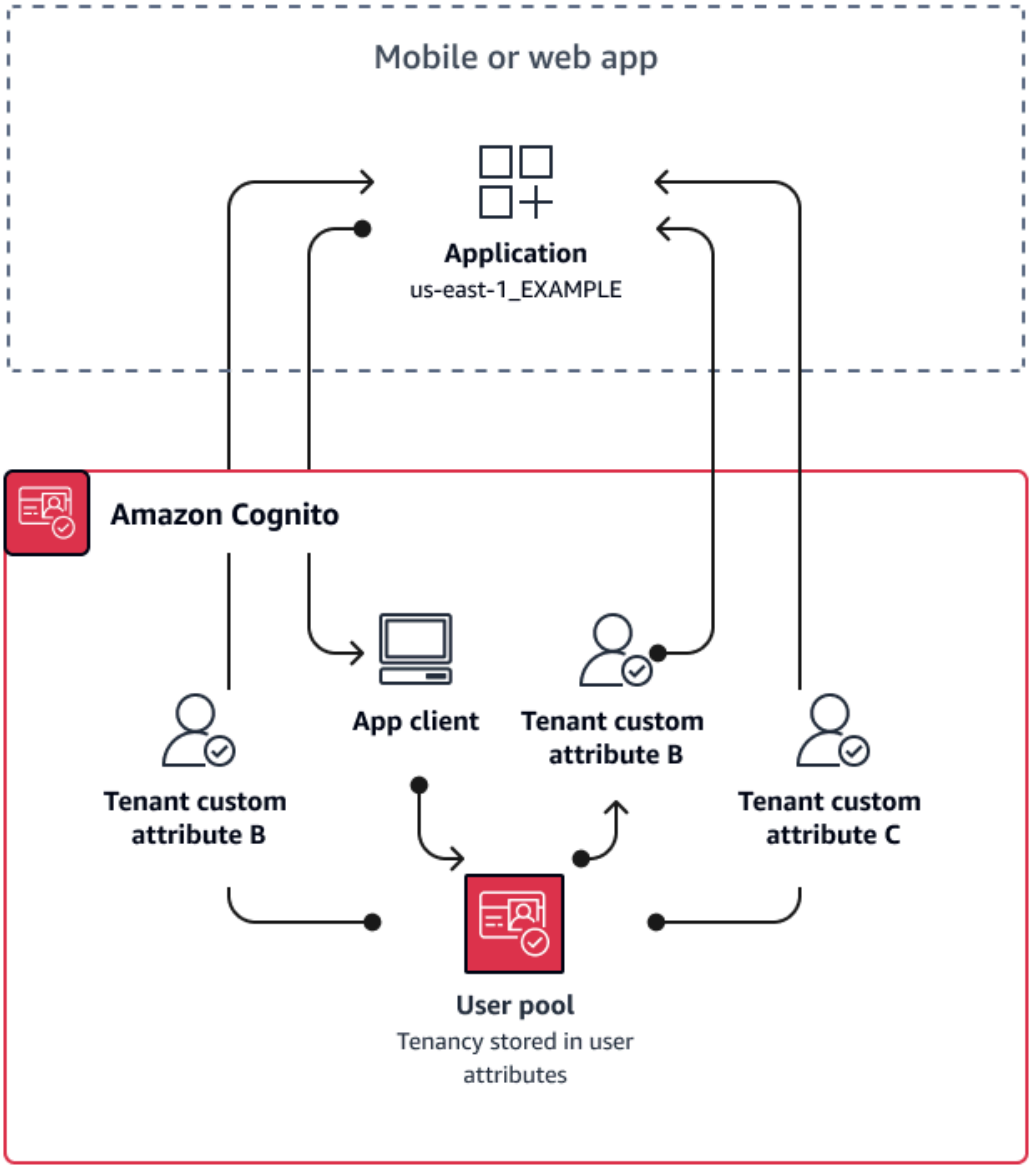
노력 수준

그룹 멤버십만으로는 다중 테넌시를 유지하기 위한 노력의 수준의 낮습니다. 그러나 사용자 풀 그룹의 역할을 기본 제공 IAM 역할 선택 용량 이상으로 확장하려면 사용자 토큰의 그룹 멤버십을 처리하는 애플리케이션 로직을 구축하고 클라이언트에서 수행할 작업을 결정해야 합니다. Amazon Verified Permissions를 앱과 통합하여 클라이언트 측 권한 부여 결정을 내릴 수 있습니다. 그룹 식별자는 현재 Verified Permissions IsAuthorizedWithToken API 작업에서 처리되지 않지만 그룹 멤버십 클레임을 포함하여 토큰의 내용을 구문 분석하는 사용자 지정 코드를 개발할 수 있습니다.

사용자 지정 속성 다중 테넌시 모범 사례

Amazon Cognito는 사용자가 선택한 이름으로 사용자 지정 속성을 지원합니다. 사용자 지정 속성이 유용한 한 가지 시나리오는 공유 사용자 풀의 사용자 테넌시를 구분하는 것입니다. 사용자에게 `custom:tenantID`와 같은 속성에 대한 값을 할당하면 그에 따라 앱이 테넌트별 리소스에 대한 액세스를 할당할 수 있습니다. 테넌트 ID를 정의하는 사용자 지정 속성은 앱 클라이언트에서 변경할 수 없거나 읽기 전용이어야 합니다.

다음 다이어그램은 앱 클라이언트와 사용자 풀을 공유하는 테넌트를 보여주며, 사용자 풀의 사용자 지정 속성은 해당 테넌트가 속해 있는 테넌트를 나타냅니다.



사용자 지정 속성에 따라 테넌시가 결정되면 단일 애플리케이션 또는 로그인 URL을 배포할 수 있습니다. 사용자가 로그인하면 앱이 `custom:tenantID` 클레임을 처리하여 로드할 자산, 적용할 브랜딩 및 표시할 기능을 결정할 수 있습니다. 사용자 속성에서 고급 액세스 제어 결정을 내리려면 Amazon Verified Permissions에서 사용자 풀을 ID 제공업체로 설정하고 ID 또는 액세스 토큰의 콘텐츠에서 액세스 결정을 생성합니다.

사용자 지정 속성 다중 테넌시를 구현해야 하는 경우

테넌시가 표면 수준인 경우. 테넌트 속성은 브랜딩 및 레이아웃 결과에 영향을 미칠 수 있습니다. 테넌트 간에 상당한 격리를 실현하려는 경우 사용자 지정 속성은 최선의 선택이 아닙니다. MFA 또는 호스팅 UI 브랜딩과 같이 사용자 풀 또는 앱 클라이언트 수준에서 구성해야 하는 테넌트 간의 모든 차이점은 사용자 지정 속성이 제공하지 않는 방식으로 테넌트 간에 구분을 생성해야 합니다. ID 풀을 사용하면 ID 토큰의 사용자 지정 속성 클레임에서 사용자로부터 IAM 역할을 선택할 수도 있습니다.

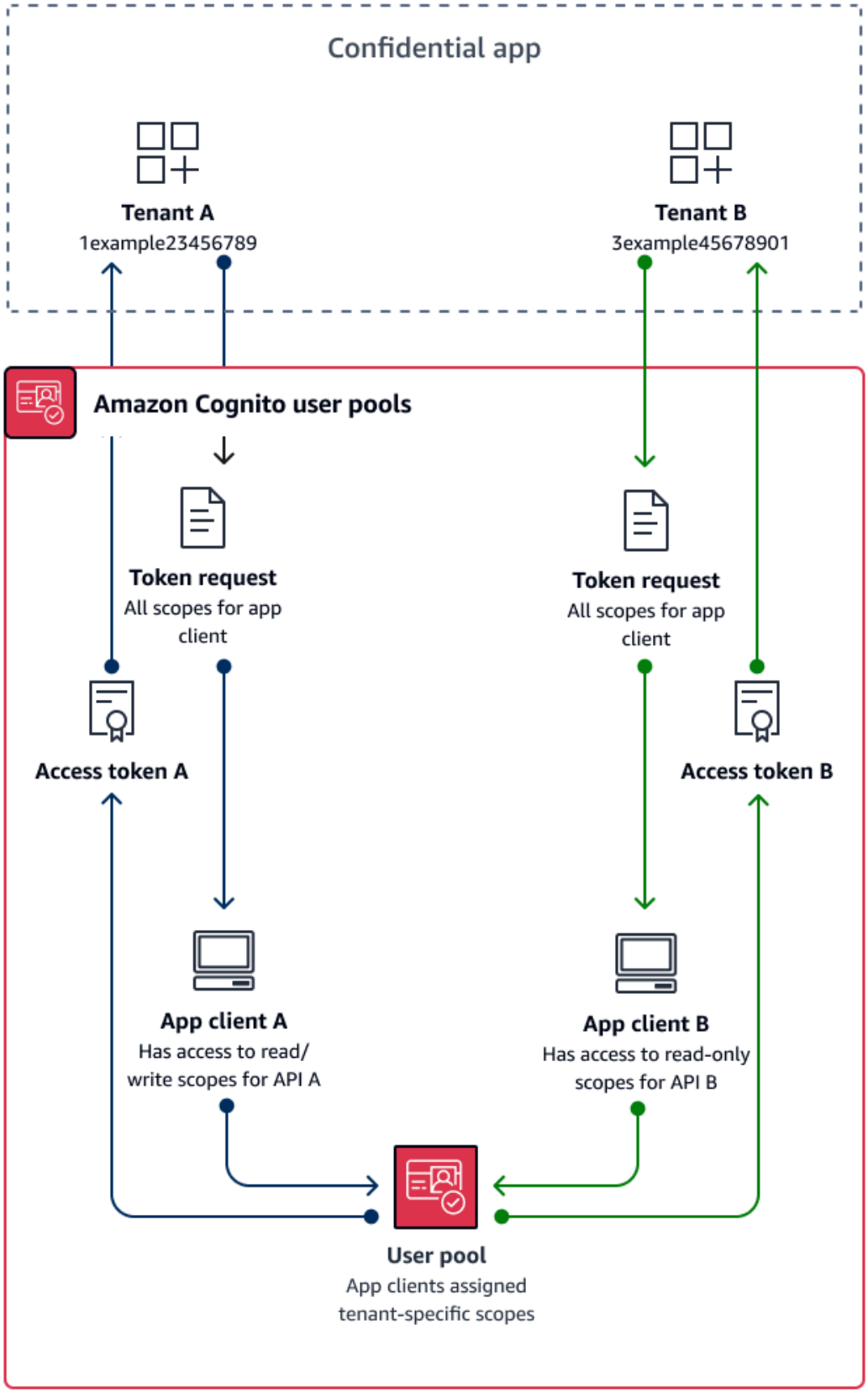
노력 수준

사용자 지정 속성 다중 테넌시가 앱에서 테넌트 기반 권한 부여 결정의 의무를 이전하기 때문에 노력 수준이 높은 경향이 있습니다. OIDC 클레임을 구문 분석하는 클라이언트 구성 또는 Amazon Verified Permissions에 이미 정통한 경우 이 접근 방식에 가장 낮은 수준의 노력이 필요할 수 있습니다.

사용자 지정 범위 다중 테넌시 모범 사례

Amazon Cognito는 리소스 서버에 대한 사용자 지정 OAuth 2.0 범위를 지원합니다. 사용자 지정 범위가 있는 M2M(Machinemachine-to-machine) 권한 부여 모델의 사용자 풀에서 앱 클라이언트 다중 테넌시를 구현할 수 있습니다. 범위 기반 다중 테넌시는 앱 클라이언트 또는 애플리케이션 구성에서 액세스를 정의하여 M2M 다중 테넌시를 구현하는 데 필요한 노력을 줄입니다.

다음 다이어그램은 사용자 지정 범위 다중 테넌시에 대한 한 가지 옵션을 보여줍니다. 사용자 풀의 관련 범위에 액세스할 수 있는 전용 앱 클라이언트가 있는 각 테넌트를 보여줍니다.


Confidential app
Tenant A
1example23456789
Tenant B
3example45678901
Amazon Cognito user pools
Token request
All scopes for app client
Token request
All scopes for app client
Access token A
Access token B
App client A
Has access to read/ write scopes for API A
App client B
Has access to read-only scopes for API B
User pool
App clients assigned tenant-specific scopes

사용자 지정 범위 다중 테넌시를 구현해야 하는 경우

기밀 클라이언트에서 클라이언트 자격 증명을 사용한 M2M 권한 부여를 사용하는 경우. 가장 좋은 방법은 앱 클라이언트 전용 리소스 서버를 생성하는 것입니다. 사용자 지정 범위 다중 테넌시는 요청에 따라 달라지거나 클라이언트에 따라 달라질 수 있습니다.

요청에 따라 달라짐

애플리케이션 로직을 구현하여 테넌트의 요구 사항과 일치하는 범위만 요청합니다. 예를 들어 앱 클라이언트는 API A 및 API B에 대한 읽기 및 쓰기 액세스를 발급할 수 있지만 테넌트 애플리케이션 A는 API A에 대한 읽기 범위와 테넌시를 나타내는 범위만 요청합니다. 이 모델을 사용하면 테넌트 간에 공유 범위를 더 복잡하게 조합할 수 있습니다.

클라이언트에 따라 달라짐

권한 부여 요청에서 앱 클라이언트에 할당된 모든 범위를 요청합니다. 이렇게 하려면 Token 엔드포인트에 대한 요청에서 `scope` 요청 파라미터를 생략합니다. GetClientToken API 작업으로 M2M 액세스 토큰을 가져오는 경우 동일한 결과에 대한 `Scopes` 파라미터를 생략합니다. 이 모델을 사용하면 앱 클라이언트가 사용자 지정 범위에 추가하려는 액세스 표시기를 저장할 수 있습니다.

어떤 경우든 애플리케이션은 종속 데이터 소스에 대한 권한을 나타내는 범위가 있는 액세스 토큰을 수신합니다. 범위는 애플리케이션에 다른 정보를 제공할 수도 있습니다.

- 테넌시 지정
- 로깅 요청에 기여
- 애플리케이션이 쿼리할 수 있는 권한이 있는 API 표시
- 활성 고객에 대한 초기 확인을 알립니다.

노력 수준

사용자 지정 범위 다중 테넌시에는 애플리케이션 규모에 따라 다양한 수준의 노력이 필요합니다. 애플리케이션이 액세스 토큰을 구문 분석하고 적절한 API 요청을 할 수 있도록 애플리케이션 로직을 고안해야 합니다.

예를 들어 리소스 서버 범위는 `[resource server identifier]/[name]` 형식으로 제공됩니다. 리소스 서버 식별자는 테넌트 범위의 권한 부여 결정과 관련이 있을 가능성이 낮으므로 범위 이름을 일관되게 구문 분석해야 합니다.

리소스 예시

다음 AWS CloudFormation 템플릿은 하나의 리소스 서버와 앱 클라이언트를 사용하여 사용자 지정 범위 다중 테넌시용 사용자 풀을 생성합니다.

```
AWSTemplateFormatVersion: "2010-09-09"
Description: A sample template illustrating scope-based multi-tenancy
Resources:
  MyUserPool:
    Type: "AWS::Cognito::UserPool"
  MyUserPoolDomain:
    Type: AWS::Cognito::UserPoolDomain
    Properties:
      UserPoolId: !Ref MyUserPool
      # Note that the value for "Domain" must be unique across all of AWS.
      # In production, you may want to consider using a custom domain.
      # See: https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-
pools-add-custom-domain.html#cognito-user-pools-add-custom-domain-adding
      Domain: !Sub "example-userpool-domain-${AWS::AccountId}"
  MyUserPoolResourceServer:
    Type: "AWS::Cognito::UserPoolResourceServer"
    Properties:
      Identifier: resource1
      Name: resource1
      Scopes:
        - ScopeDescription: Read-only access
          ScopeName: readScope
      UserPoolId: !Ref MyUserPool
  MyUserPoolTenantBatch1ResourceServer:
    Type: "AWS::Cognito::UserPoolResourceServer"
    Properties:
      Identifier: TenantBatch1
      Name: TenantBatch1
      Scopes:
        - ScopeDescription: tenant1 identifier
          ScopeName: tenant1
        - ScopeDescription: tenant2 identifier
          ScopeName: tenant2
      UserPoolId: !Ref MyUserPool
  MyUserPoolClientTenant1:
    Type: "AWS::Cognito::UserPoolClient"
    Properties:
      AllowedOAuthFlows:
```

```
        - client_credentials
      AllowedOAuthFlowsUserPoolClient: true
      AllowedOAuthScopes:
        - !Sub "${MyUserPoolTenantBatch1ResourceServer}/tenant1"
        - !Sub "${MyUserPoolResourceServer}/readScope"
      GenerateSecret: true
      UserPoolId: !Ref MyUserPool
Outputs:
  UserPoolClientId:
    Description: User pool client ID
    Value: !Ref MyUserPoolClientTenant1
  UserPoolDomain:
    Description: User pool domain
    Value: !Sub "https://${MyUserPoolDomain}.auth.${AWS::Region}.amazoncognito.com"
```

멀티 테넌시 보안 권장 사항

보다 안전한 애플리케이션을 만들려면 다음을 권장합니다.

- Amazon Verified Permissions를 사용하여 앱에서 테넌시를 검증합니다. 애플리케이션에서 사용자의 요청을 허용하기 전에 사용자 풀, 앱 클라이언트, 그룹 또는 사용자 지정 속성 권한을 검사하는 정책을 구축합니다.는 Amazon Cognito 사용자 풀을 염두에 두고 Verified Permissions 자격 증명 소스를 AWS 생성했습니다. Verified Permissions에는 다중 테넌시 관리에 대한 추가 지침이 있습니다.
- 도메인 일치를 기반으로 테넌트에 대한 사용자 액세스 권한을 부여하는 데 확인된 이메일 주소만 사용합니다. 앱에서 확인하거나 외부 IdP가 확인 증명을 제공하는 경우에만 이메일 주소와 전화 번호를 신뢰하세요. 이러한 권한 설정에 대한 자세한 내용은 속성 권한 및 범위를 참조하세요.
- 테넌트를 식별하는 사용자 프로필 속성에 변경 불가능 또는 읽기 전용 사용자 지정 속성을 사용합니다. 사용자를 생성하거나 사용자 풀에 가입한 경우에만 변경할 수 없는 속성 값을 설정할 수 있습니다. 또한 앱 클라이언트에 이러한 속성에 대한 읽기 전용 권한을 부여합니다.
- 테넌트의 외부 IdP와 애플리케이션 클라이언트 간에 1:1 매핑을 사용하여 무단 테넌트 간 액세스를 방지합니다. 외부 IdP에서 인증되었으며 유효한 Amazon Cognito 세션 쿠키가 있는 사용자는 동일한 IdP를 신뢰하는 다른 테넌트 앱에 액세스할 수 있습니다.
- 애플리케이션에서 테넌트 일치 및 권한 부여 로직을 구현할 때 테넌트에 대한 사용자 액세스 권한을 부여하는 기준을 사용자가 직접 수정할 수 없도록 합니다. 또한 페더레이션에 외부 IdP 가 사용되는 경우 테넌트 ID 공급자 관리자가 사용자 액세스를 수정할 수 없도록 합니다.

웹 및 모바일 앱과 Amazon Cognito 인증 및 권한 부여 통합

Amazon Cognito 구현은 AWS Management Console 또는 AWS SDK 관리 도구와 애플리케이션의 SDK 라이브러리를 혼합한 것입니다. Amazon Cognito 콘솔은 Amazon Cognito 사용자 풀과 자격 증명 풀을 설정하고 관리하기 위한 시각적 인터페이스입니다.

Amazon Cognito 사용자 풀을 사용하여 생성할 수 있는 가장 낮은 노력의 통합은 관리형 로그인을 사용하는 것입니다. 관리형 로그인은 Amazon Cognito 사용자 풀의 빠른 테스트 및 배포를 위해 바로 사용할 수 있는 웹 기반 로그인 애플리케이션입니다. 관리형 로그인을 사용한 사용자 풀 인증에는 사용자를 호스팅된 로그인 페이지로 안내하는 OpenID Connect(OIDC) 라이브러리가 필요합니다. 이 사용자 상호 작용 및 리디렉션 웹 엔드포인트 시리즈에서 Amazon Cognito는 타사 로그인, 다중 인증(MFA) 및 인증 흐름 선택을 포함한 인증 흐름을 처리합니다. 애플리케이션은 Amazon Cognito가 응답에서 반환하는 인증 결과만 처리하면 됩니다.

또한 애플리케이션에 AWS SDK를 추가하고, 사용자 지정 빌드 인증 인터페이스를 추가하고, 사용자의 인증 및 권한 부여를 위한 API 작업을 호출할 수 있습니다. AWS Amplify는 백엔드에서 Amazon Cognito 인증을 사용하여 풀 스택 애플리케이션을 빌드하기 AWS 서비스 위한 입니다.

예를 들어 앱은 사용자 로그인을 위해 관리형 로그인을 간접적으로 호출한 다음 앱 코드에서 토큰 엔드포인트를 직접적으로 호출하여 사용자의 인증 코드를 토큰으로 교환할 수 있습니다. 그런 다음 앱은 사용자 토큰을 해석 및 저장하고 인증 및 권한 부여를 위해 적절한 컨텍스트에서 토큰을 제시해야 합니다. Amplify는 이러한 프로세스를 위한 내장 함수가 포함된 안내 통합 도구를 추가합니다.

또한 Amazon Cognito 리소스를 완전히 코드로 구축할 수도 있습니다. 자격 증명 풀에는 사용자 풀과 동일한 관리형 인증 옵션이 없습니다. 애플리케이션의 AWS 자격 증명에 액세스하려면 가져온 SDK 모듈에서 자격 증명 풀 작업을 구현합니다. 맞춤형 애플리케이션 코드를 시작하려면 AWS SDK용 Amazon Cognito 코드 예제를 참조하세요. OpenID Connect 자격 증명 공급자로서 Amazon Cognito와 통합하려면 OpenID Connect 개발자 도구를 사용합니다.

Amazon Cognito 인증 및 권한 부여를 사용하기 전에 앱 플랫폼을 선택하고 서비스와 통합할 코드를 준비합니다. AWS SDKsAWS SDKs 사용한 인증. AWS CLI 는 Amazon Cognito 및 기타용 명령줄 SDK AWS 서비스이며 Amazon Cognito API 작업 및 해당 구문을 익히기 위한 유용한 곳입니다.

> **Note**
>
> Amazon Cognito의 일부 구성 요소는 API만을 사용하여 구성할 수 있습니다. 예를 들어 CreateUserPool 또는 UpdateUserPool API 요청에서 UserPool 클래스의 `LambdaConfig` 속

성을 업데이트하는 요청으로만 사용자 풀 사용자 지정 SMS 또는 이메일 발신자 Lambda 트리거를 설정할 수 있습니다.

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

주제

- 를 사용한 인증 AWS Amplify
- AWS SDKs 사용한 인증
- Amazon Cognito에서 인증이 작동하는 방식
- AWS SDK에서이 서비스 사용
- Amazon Verified Permissions를 통한 권한 부여

를 사용한 인증 AWS Amplify

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

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

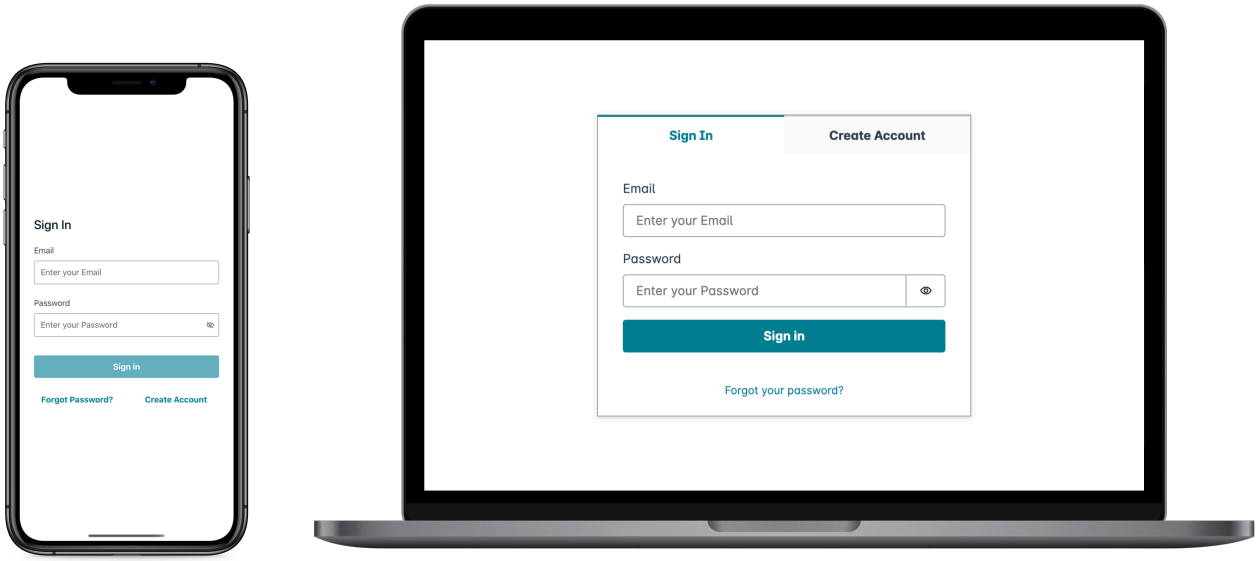
- React에 대한 Amplify 인증
- React Native에 대한 Amplify 인증
- Swift(iOS)에 대한 Amplify 인증
- Android에 대한 Amplify 인증
- Flutter에 대한 Amplify 인증

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

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

Amplify로 사용자 인터페이스(UI) 생성

사용자 풀 관리형 로그인는 웹 또는 모바일 앱에 대한 인증 프런트 엔드의 필수 요구 사항을 충족할 수 있습니다. 관리형 로그인이 수용하는 파라미터를 넘어 사용자 인터페이스(UI)를 사용자 지정하려면 애플리케이션을 사용자 지정하여 빌드하세요. Amplify UI는 다양한 언어로 된 사용자 지정 가능한 프런트 엔드 구성 요소 모음입니다.



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

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

- Swift용 Authenticator
- Vue용 Authenticator

AWS SDKs 사용한 인증

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

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

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS Java V2용 SDK
- AWS SDK for JavaScript
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

Amazon Cognito에서 인증이 작동하는 방식

고객이 Amazon Cognito 사용자 풀에 로그인하면 애플리케이션은 JSON 웹 토큰(JWT)을 수신합니다.

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

사용자 풀 로그인을 사용하면 AWS SDK로 인증 및 권한 부여를 완전히 구현할 수 있습니다. 자체 사용자 인터페이스(UI) 구성 요소를 빌드하지 않으려면 미리 빌드된 웹 UI(관리형 로그인) 또는 타사 ID 제공업체(idP)의 로그인 페이지를 호출할 수 있습니다.

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

주제

- 관리형 로그인을 사용한 사용자 풀 인증
- AWS SDK를 사용한 사용자 풀 API 인증 및 권한 부여
- 타사 ID 제공업체를 통한 사용자 풀 인증
- ID 풀 인증

관리형 로그인을 사용한 사용자 풀 인증

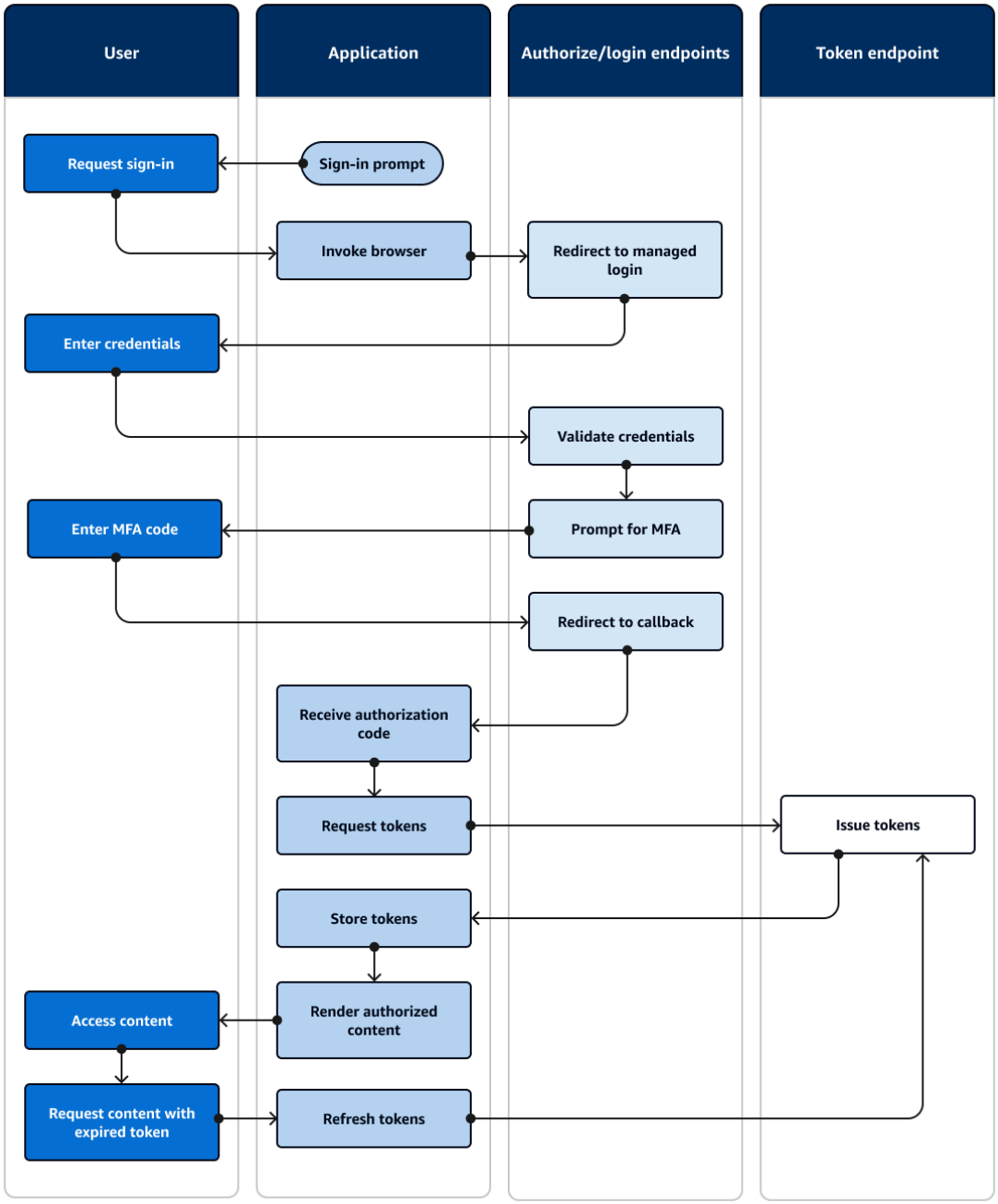
관리형 로그인은 사용자 풀 및 앱 클라이언트에 연결된 웹 사이트입니다. 사용자의 로그인, 가입 및 비밀번호 재설정 작업을 수행할 수 있습니다. 인증을 위해 관리형 로그인 구성 요소가 있는 애플리케이션은 구현하는 데 개발자의 노력이 덜 필요할 수 있습니다. 애플리케이션은 인증을 위해 UI 구성 요소를 건너뛰고 사용자 브라우저에서 관리형 로그인을 호출할 수 있습니다.

애플리케이션은 웹 또는 앱 리디렉션 위치를 사용하여 사용자의 JWT를 수집합니다. 관리형 로그인을 구현하는 애플리케이션은 OIDC(OpenID Connect) IdP인 것처럼 인증을 위해 사용자 풀에 연결할 수 있습니다.

관리형 로그인은 애플리케이션에 OIDC 인증 서버의 인증 서비스가 필요하지만 사용자 지정 인증, ID 풀 통합 또는 사용자 속성 셀프 서비스와 같은 기능이 즉시 필요하지 않은 모델에 적합합니다. 이러한 고급 옵션 중 일부를 사용하려는 경우 SDK의 사용자 풀 구성 요소를 사용하여 구현할 수 있습니다.

OIDC 구현에 주로 의존하는 관리형 로그인 및 타사 IdP 인증 모델은 OAuth 2.0 범위가 있는 고급 권한 부여 모델에 가장 적합합니다.

다음 다이어그램은 관리형 로그인 인증을 위한 일반적인 로그인 세션을 보여줍니다.



관리형 로그인 인증 흐름

1. 사용자가 애플리케이션에 액세스합니다.
2. “로그인” 링크를 선택합니다.
3. 애플리케이션은 사용자를 사용자 풀 도메인의 관리형 로그인 페이지에 있는 로그인 프롬프트로 안내합니다.
4. 사용자 이름과 비밀번호를 입력합니다.

5. 사용자 풀은 사용자의 자격 증명을 검증하고 사용자가 다중 인증(MFA)을 활성화했는지 확인합니다.
6. 관리형 로그인 페이지는 사용자에게 MFA 코드를 입력하라는 메시지를 표시합니다.
7. 사용자가 MFA 코드를 입력합니다.
8. 사용자 풀은 사용자를 애플리케이션 URL로 리디렉션합니다.
9. 애플리케이션은 관리형 로그인이 콜백 URL에 추가한 URL 요청 파라미터에서 권한 부여 코드를 수집합니다.
10. 애플리케이션은 권한 부여 코드가 있는 토큰을 요청합니다.
11. 토큰 엔드포인트는 애플리케이션에 JWT를 반환합니다.
12. 애플리케이션은 사용자의 JWT를 디코딩, 유효성 검사 및 저장하거나 캐싱합니다.
13. 애플리케이션에 요청된 액세스 제어 구성 요소가 표시됩니다.
14. 사용자가 콘텐츠를 봅니다.
15. 나중에 사용자의 액세스 토큰이 만료되어 액세스 제어된 구성 요소를 보도록 요청합니다.
16. 애플리케이션에서 사용자의 세션이 지속되어야 한다고 판단합니다. 새로 고침 토큰을 사용하여 토큰 엔드포인트에서 새 토큰을 요청합니다.

변형 및 사용자 지정

전체 사용자 풀의 브랜딩 편집기를 사용하거나 앱 클라이언트 수준에서 관리형 로그인 페이지의 디자인과 느낌을 사용자 지정할 수 있습니다. 자체 ID 제공업체, 범위, 사용자 속성에 대한 액세스 및 고급 보안 구성으로 앱 클라이언트를 구성할 수도 있습니다.

관련 리소스

- 사용자 풀 관리형 로그인
- 범위, M2M 및 리소스 서버
- 사용자 풀 엔드포인트 및 관리형 로그인 참조

AWS SDK를 사용한 사용자 풀 API 인증 및 권한 부여

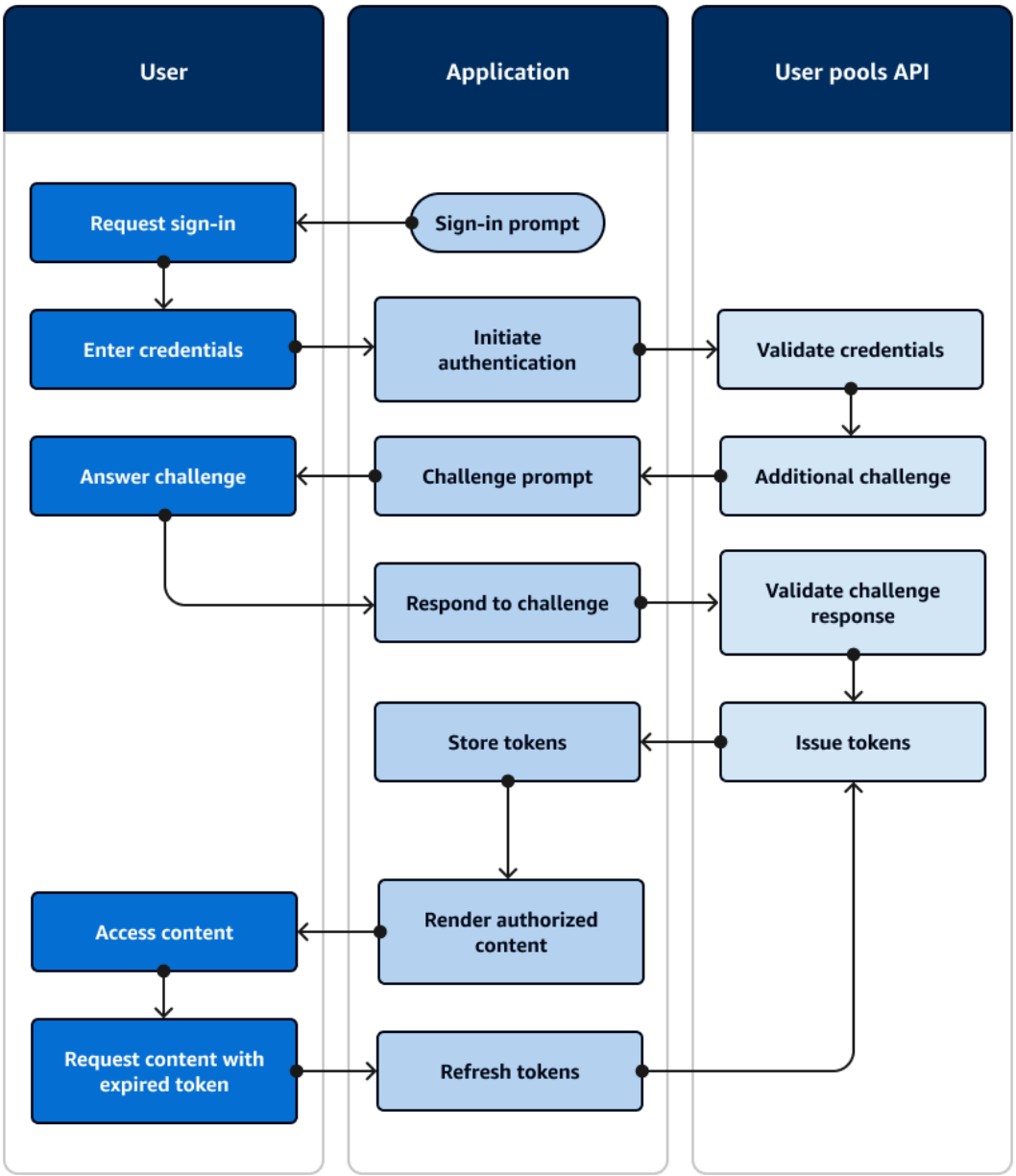
AWS 는 다양한 개발자 프레임워크에서 Amazon Cognito 사용자 풀 또는 Amazon Cognito 자격 증명 공급자를 위한 구성 요소를 개발했습니다. ??? 이러한 SDK를 Amazon Cognito 사용자 풀 API라고 합니다. 동일한 사용자 풀 API 네임스페이스에는 사용자 풀 구성 및 사용자 인증 작업이 있습니다. 보다 자세한 개요는 API, OIDC 및 관리형 로그인 페이지 인증 이해 섹션을 참조하세요.

API 인증은 애플리케이션에 기존 UI 구성 요소가 있고 주로 사용자 풀을 사용자 디렉터리로 사용하는 모델에 적합합니다. 이 설계는 Amazon Cognito를 더 큰 애플리케이션 내의 구성 요소로 추가합니다. 복잡한 챌린지 및 응답 체인을 처리하려면 프로그래밍 로직이 필요합니다.

이 애플리케이션은 전체 OIDC(OpenID Connect) 신뢰 당사자 구현을 구현할 필요가 없습니다. 대신 JWT를 디코딩하고 사용할 수 있습니다. 로컬 사용자를 위한 전체 사용자 풀 기능 세트에 액세스하려면 개발 환경에서 Amazon Cognito SDK로 인증을 빌드하세요.

사용자 지정 OAuth 범위를 사용한 API 인증은 외부 API 권한 부여에 대한 방향성이 떨어집니다. API 인증에서 액세스 토큰에 사용자 지정 범위를 추가하려면 런타임 시 사전 토큰 생성 Lambda 트리거을 사용하여 토큰을 수정합니다.

다음 다이어그램은 API 인증을 위한 일반적인 로그인 세션을 보여줍니다.



API 인증 흐름

1. 사용자가 애플리케이션에 액세스합니다.
2. “로그인” 링크를 선택합니다.
3. 사용자 이름과 비밀번호를 입력합니다.
4. 애플리케이션은 InitiateAuth API 요청을 수행하는 메서드를 간접적으로 호출합니다. 요청은 사용자의 자격 증명을 사용자 풀에 전달합니다.

5. 사용자 풀은 사용자의 자격 증명을 검증하고 사용자가 다중 인증(MFA)을 활성화했는지 확인합니다.
6. 사용자 풀은 MFA 코드를 요청하는 챌린지로 응답합니다.
7. 애플리케이션에서 사용자로부터 MFA 코드를 수집하는 프롬프트를 생성합니다.
8. 애플리케이션은 RespondToAuthChallenge API 요청을 수행하는 메서드를 호출합니다. 요청이 사용자의 MFA 코드를 전달합니다.
9. 사용자 풀은 사용자의 MFA 코드를 검증합니다.
10. 사용자 풀은 사용자의 JWT로 응답합니다.
11. 애플리케이션은 사용자의 JWT를 디코딩, 유효성 검사 및 저장하거나 캐싱합니다.
12. 애플리케이션에 요청된 액세스 제어 구성 요소가 표시됩니다.
13. 사용자가 콘텐츠를 봅니다.
14. 나중에 사용자의 액세스 토큰이 만료되어 액세스 제어된 구성 요소를 보도록 요청합니다.
15. 애플리케이션에서 사용자의 세션이 지속되어야 한다고 판단합니다. 새로 고침 토큰을 사용하여 InitiateAuth 메서드를 다시 호출하고 새 토큰을 검색합니다.

변형 및 사용자 지정

사용자 지정 인증 챌린지와 같은 추가 챌린지를 통해 이 흐름을 강화할 수 있습니다. 암호가 손상되었거나 예기치 않은 로그인 특성이 악의적인 로그인 시도를 나타낼 수 있는 사용자의 액세스를 자동으로 제한할 수 있습니다. 이 흐름은 가입, 사용자 속성 업데이트, 비밀번호 재설정 작업에서도 거의 동일하게 나타납니다. 이러한 흐름의 대부분은 퍼블릭(클라이언트 측) 및 기밀(서버 측) API 작업이 중복됩니다.

관련 리소스

- Amazon Cognito 사용자 풀 API
- Amazon Cognito 콘솔에서 새 애플리케이션 생성
- 웹 및 모바일 앱과 Amazon Cognito 인증 및 권한 부여 통합
- API, OIDC 및 관리형 로그인 페이지 인증 이해

타사 ID 제공업체를 통한 사용자 풀 인증

외부 ID 제공업체(idP) 또는 페더레이션 인증을 사용한 로그인은 관리형 로그인과 유사한 모델입니다. 애플리케이션은 사용자 풀의 OIDC 신뢰 당사자인 반면 사용자 풀은 IdP에 대한 패스스루 역할을 합니

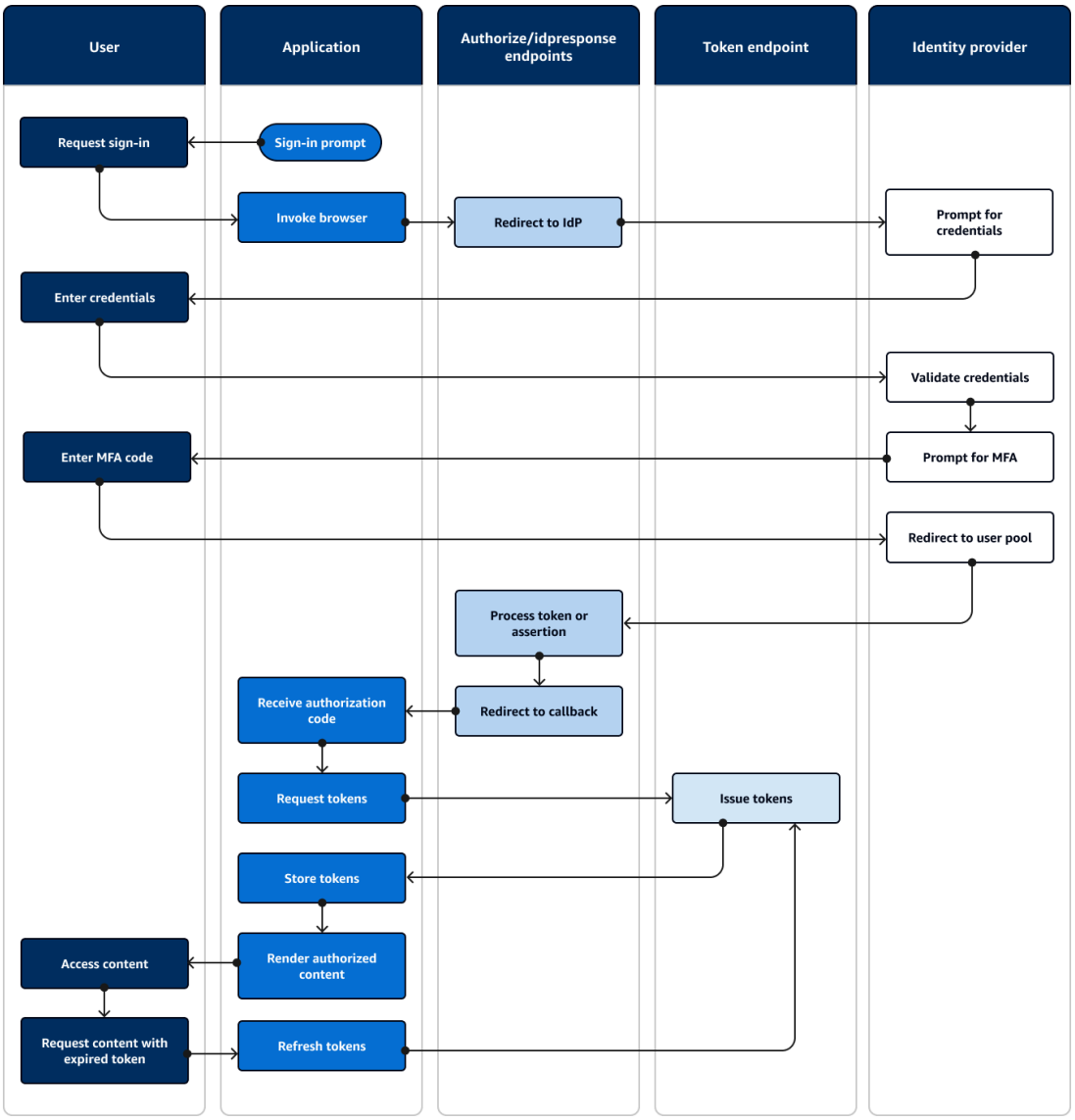
다. IdP는 Facebook 또는 Google과 같은 소비자 사용자 디렉터리이거나 Azure와 같은 SAML 2.0 또는 OIDC 엔터프라이즈 디렉터리일 수 있습니다.

사용자의 브라우저에서 관리형 로그인 대신 애플리케이션은 사용자 풀 권한 부여 서버에서 리디렉션 엔드포인트를 호출합니다. 사용자의 보기에서 애플리케이션에서 로그인 버튼을 선택합니다. 그런 다음 IdP에 로그인하라는 메시지가 표시됩니다. 관리형 로그인 인증과 마찬가지로 애플리케이션은 앱의 리디렉션 위치에서 JWT를 수집합니다.

타사 IdP를 사용한 인증은 사용자가 애플리케이션에 가입할 때 새 비밀번호를 생성하지 않으려는 모델에 적합합니다. 관리형 로그인 인증을 구현한 애플리케이션에는 적은 노력으로 타사 인증을 추가할 수 있습니다. 실제로 관리형 로그인과 타사 IdP는 사용자 브라우저에서 호출하는 항목에 약간의 차이가 있어도 일관된 인증 결과를 생성합니다.

관리형 로그인 인증과 마찬가지로 페더레이션 인증은 OAuth 2.0 범위가 있는 고급 권한 부여 모델에 가장 적합합니다.

다음 다이어그램은 페더레이션 인증을 위한 일반적인 로그인 세션을 보여줍니다.



페더레이션 인증 흐름

1. 사용자가 애플리케이션에 액세스합니다.
2. “로그인” 링크를 선택합니다.
3. 애플리케이션은 IdP를 사용하여 사용자를 로그인 프롬프트로 안내합니다.
4. 사용자 이름과 비밀번호를 입력합니다.
5. IdP는 사용자의 자격 증명을 검증하고 사용자가 다중 인증(MFA)을 활성화했는지 확인합니다.
6. IdP는 사용자에게 MFA 코드를 입력하라는 메시지를 표시합니다.
7. 사용자가 MFA 코드를 입력합니다.

8. IdP는 SAML 응답 또는 권한 부여 코드를 사용하여 사용자를 사용자 풀로 리디렉션합니다.
9. 사용자가 권한 부여 코드를 통과하면 사용자 풀은 코드를 IdP 토큰으로 자동 교환합니다. 사용자 풀은 IdP 토큰을 검증하고 새 권한 부여 코드를 사용하여 사용자를 애플리케이션으로 리디렉션합니다.
10. 애플리케이션은 사용자 풀이 콜백 URL에 추가한 URL 요청 파라미터에서 권한 부여 코드를 수집합니다.
11. 애플리케이션은 권한 부여 코드가 있는 토큰을 요청합니다.
12. 토큰 엔드포인트는 애플리케이션에 JWT를 반환합니다.
13. 애플리케이션은 사용자의 JWT를 디코딩, 유효성 검사 및 저장하거나 캐싱합니다.
14. 애플리케이션에 요청된 액세스 제어 구성 요소가 표시됩니다.
15. 사용자가 콘텐츠를 봅니다.
16. 나중에 사용자의 액세스 토큰이 만료되어 액세스 제어된 구성 요소를 보도록 요청합니다.
17. 애플리케이션에서 사용자의 세션이 지속되어야 한다고 판단합니다. 새로 고침 토큰을 사용하여 토큰 엔드포인트에서 새 토큰을 요청합니다.

변형 및 사용자 지정

관리형 로그인에서 페더레이션 인증을 시작할 수 있으며 사용자는 앱 클라이언트에 할당된 IdP 목록에서 선택할 수 있습니다. 관리형 로그인은 이메일 주소를 입력하라는 메시지를 표시하고 해당 SAML IdP로 사용자의 요청을 자동으로 라우팅할 수도 있습니다. 타사 ID 제공업체를 사용한 인증에는 관리형 로그인과의 사용자 상호 작용이 필요하지 않습니다. 애플리케이션은 사용자의 권한 부여 서버 요청에 요청 파라미터를 추가하고 사용자가 자동으로 IdP 로그인 페이지로 리디렉션하도록 할 수 있습니다.

관련 리소스

- 타사 ID 제공업체를 통한 사용자 풀 로그인
- 범위, M2M 및 리소스 서버
- 사용자 풀 엔드포인트 및 관리형 로그인 참조

ID 풀 인증

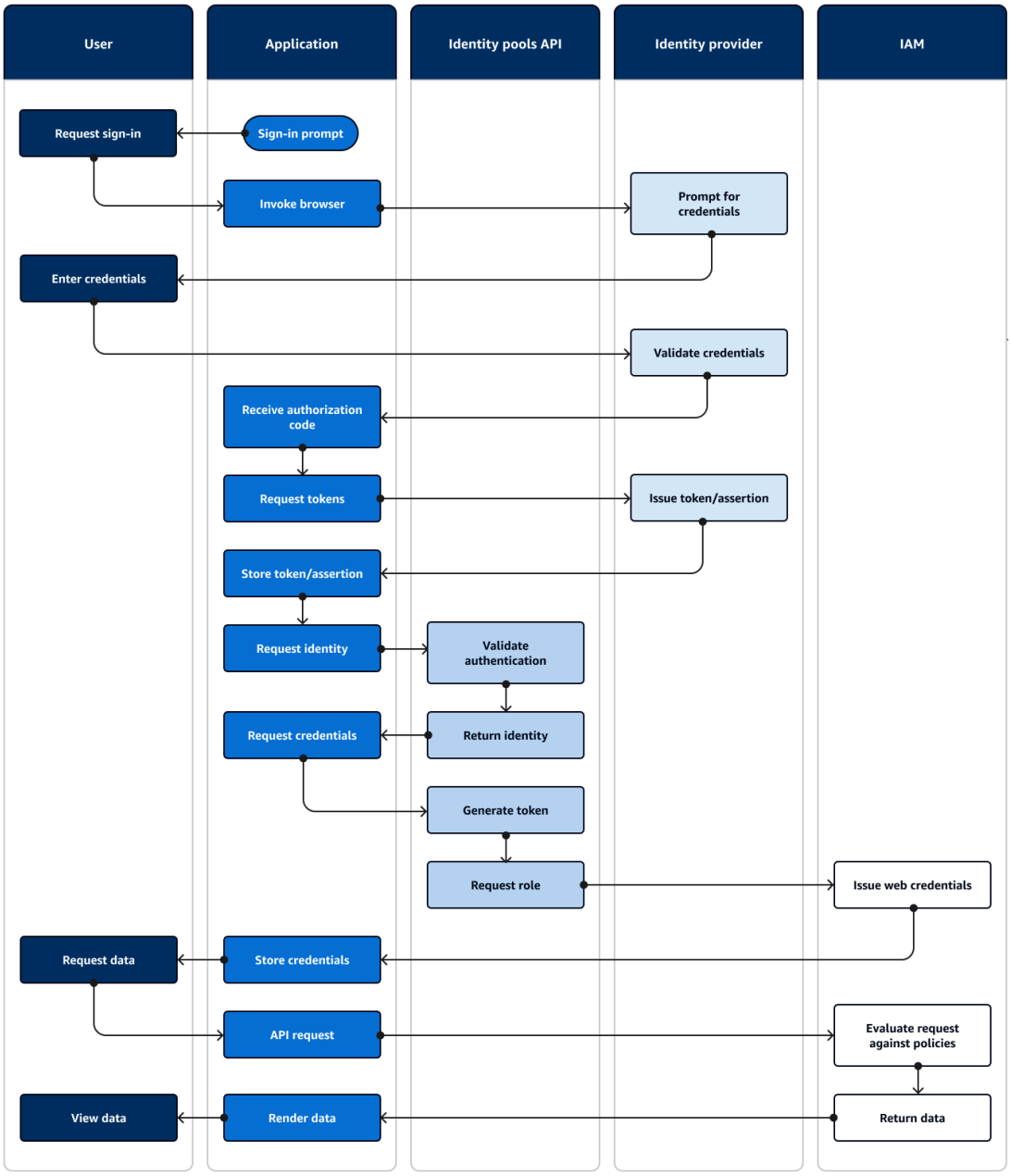
ID 풀은 함수, API 네임스페이스 및 SDK 모델의 사용자 풀과 구별되는 애플리케이션의 구성 요소입니다. 사용자 풀이 토큰 기반 인증 및 권한 부여를 제공하는 경우 자격 증명 풀은 AWS Identity and Access Management (IAM)에 대한 권한 부여를 제공합니다.

ID 풀에 IdP 세트를 할당하고 이를 사용하여 사용자를 로그인할 수 있습니다. 사용자 풀은 ID 풀 IdP로 긴밀하게 통합되며 ID 풀에 액세스 제어를 위한 가장 많은 옵션을 제공합니다. 동시에 ID 풀에 대한 다양한 인증 옵션이 있습니다. 사용자 풀은 SAML, OIDC, 소셜, 개발자 및 게스트 자격 증명 소스를 자격 증명 풀에서 임시 AWS 자격 증명으로 가는 경로로 조인합니다.

ID 풀을 사용한 인증은 외부 인증입니다. 이전에 설명한 사용자 풀 흐름 중 하나 또는 다른 IdP를 사용하여 독립적으로 개발하는 흐름을 따릅니다. 애플리케이션이 초기 인증을 수행한 후 ID 풀에 증거를 전달하고 그 대가로 임시 세션을 수신합니다.

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

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



ID 풀 인증 흐름

1. 사용자가 애플리케이션에 액세스합니다.
2. “로그인” 링크를 선택합니다.
3. 애플리케이션은 IdP를 사용하여 사용자를 로그인 프롬프트로 안내합니다.
4. 사용자 이름과 비밀번호를 입력합니다.
5. IdP는 사용자의 자격 증명을 검증합니다.

6. IdP는 SAML 응답 또는 권한 부여 코드를 사용하여 사용자를 애플리케이션으로 리디렉션합니다.
7. 사용자가 권한 부여 코드를 전달하면 애플리케이션은 코드를 IdP 토큰으로 교환합니다.
8. 애플리케이션은 사용자의 JWT 또는 어설션을 디코딩, 유효성 검사 및 저장하거나 캐싱합니다.
9. 애플리케이션은 GetId API 요청을 수행하는 메서드를 호출합니다. 사용자의 토큰 또는 어설션을 전달하고 자격 증명 ID를 요청합니다.
10. ID 풀은 구성된 ID 제공업체에 대해 토큰 또는 어설션을 검증합니다.
11. ID 풀은 자격 증명 ID를 반환합니다.
12. 애플리케이션은 GetCredentialsForIdentity API 요청을 수행하는 메서드를 호출합니다. 사용자의 토큰 또는 어설션을 전달하고 IAM 역할을 요청합니다.
13. ID 풀은 새 JWT를 생성합니다. 새 JWT에는 IAM 역할을 요청하는 클레임이 포함되어 있습니다. ID 풀은 사용자의 요청과 IdP 에 대한 ID 풀 구성의 역할 선택 기준을 기반으로 역할을 결정합니다.
14. AWS Security Token Service (AWS STS)는 자격 증명 풀의 AssumeRoleWithWebIdentity 요청에 응답합니다. 응답에는 IAM 역할이 있는 임시 세션의 API 자격 증명이 포함되어 있습니다.
15. 애플리케이션은 세션 자격 증명을 저장합니다.
16. 사용자가 앱에서 AWS의 액세스 보호 리소스가 필요한 작업을 수행합니다.
17. 애플리케이션은 임시 자격 증명을 필수에 대한 API 요청에 서명으로 적용합니다 AWS 서비스.
18. IAM은 자격 증명의 역할에 연결된 정책을 평가합니다. 이를 요청과 비교합니다.
19. 는 요청된 데이터를 AWS 서비스 반환합니다.
20. 애플리케이션은 사용자 인터페이스에서 데이터를 렌더링합니다.
21. 사용자가 데이터를 봅니다.

변형 및 사용자 지정

사용자 풀을 사용한 인증을 시각화하려면 문제 토큰/어설션 단계 뒤에 이전 사용자 풀 개요 중 하나를 삽입합니다. 개발자 인증은 ID 요청 이전의 모든 단계를 개발자 자격 증명으로 서명된 요청으로 바꿉니다. 또한 게스트 인증은 ID 요청으로 바로 건너뛰고, 인증을 검증하지 않으며, 액세스가 제한된 IAM 역할에 대한 자격 증명을 반환합니다.

관련 리소스

- Amazon Cognito 자격 증명 풀
- 사용자 IAM 역할
- 자격 증명 풀 인증 흐름

AWS SDK에서이 서비스 사용

AWS 소프트웨어 개발 키트(SDKs)는 널리 사용되는 많은 프로그래밍 언어에 사용할 수 있습니다. 각 SDK는 개발자가 선호하는 언어로 애플리케이션을 쉽게 구축할 수 있도록 하는 API, 코드 예제 및 설명서를 제공합니다.

SDK 설명서	코드 예제
AWS SDK for C++	AWS SDK for C++ 코드 예제
AWS CLI	AWS CLI 코드 예제
AWS SDK for Go	AWS SDK for Go 코드 예제
AWS SDK for Java	AWS SDK for Java 코드 예제
AWS SDK for JavaScript	AWS SDK for JavaScript 코드 예제
AWS SDK for Kotlin	AWS SDK for Kotlin 코드 예제
AWS SDK for .NET	AWS SDK for .NET 코드 예제
AWS SDK for PHP	AWS SDK for PHP 코드 예제
AWS Tools for PowerShell	AWS Tools for PowerShell 코드 예제
AWS SDK for Python v4	AWS SDK for Python v4 코드 예제
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 코드 예제
AWS SDK for Ruby	AWS SDK for Ruby 코드 예제
AWS SDK for Rust	AWS SDK for Rust 코드 예제
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP 코드 예제
AWS SDK for Swift	AWS SDK for Swift 코드 예제

> 예제 사용 가능 여부
>
> 필요한 예제를 찾을 수 없습니까? 이 페이지 하단의 피드백 제공 링크를 사용하여 코드 예제를 요청합니다.

Amazon Verified Permissions를 통한 권한 부여

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

앱에서 사용자의 신원 또는 액세스 토큰을 IsAuthorizedWithToken 또는 BatchIsAuthorizedWithToken API 요청의 확인된 권한에 제공할 수 있습니다. 이러한 API 작업은 사용자를 `Principal`로 수락하고 사용자가 액세스하려는 `Resource`의 `Action`에 대한 권한 부여 결정을 내립니다. 추가 사용자 지정 `Context`는 세부적인 액세스 결정에 기여할 수 있습니다.

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

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

취소된 토큰 및 삭제된 사용자

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

정책 평가

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

토큰 사용자 지정

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

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

추가 리소스

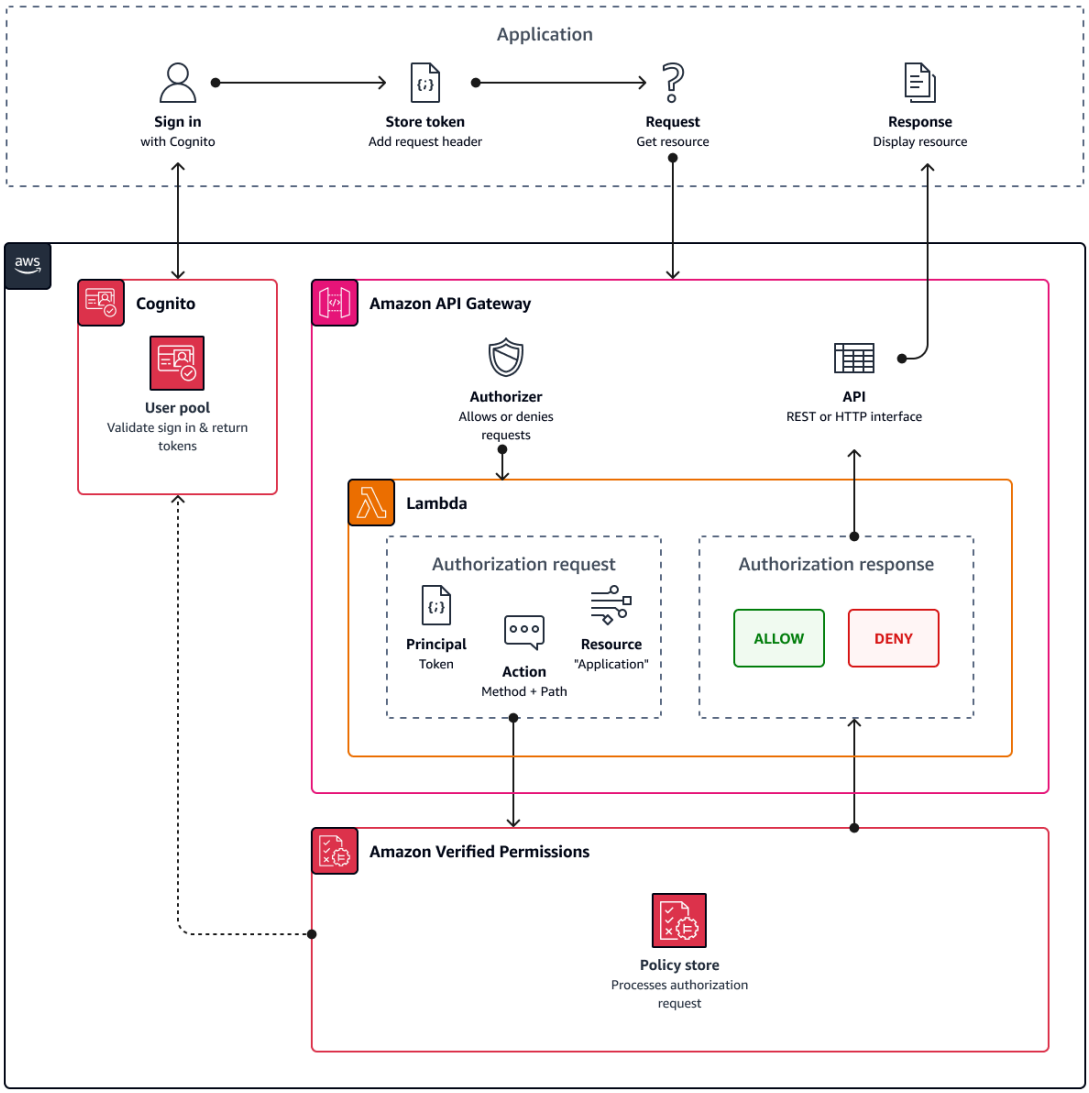
- Amazon Cognito 토큰 Verified Permissions 스키마 매핑
- Amazon Verified Permissions 및 Amazon Cognito를 사용하여 API Gateway API 권한 부여
- 워크숍: Amazon Cognito 및 Verified Permissions를 사용한 인증 및 권한 부여

Verified Permissions를 사용한 API 권한 부여

ID 또는 액세스 토큰은 Verified Permissions를 사용하여 Amazon API Gateway REST APIs 백엔드에 대한 요청을 승인할 수 있습니다. 사용자 풀 및 API에 대한 즉각적인 링크가 포함된 정책 저장

소를 생성할 수 있습니다. API Gateway 및 ID 소스를 사용하여 설정 시작 옵션을 사용하면 Verified Permissions가 정책 저장소에 사용자 풀 ID 소스를 추가하고 API에 Lambda 권한 부여자를 추가합니다. 애플리케이션이 사용자 풀 보유자 토큰을 API에 전달하면 Lambda 권한 부여자가 Verified Permissions를 호출합니다. 권한 부여자는 토큰을 보안 주체로, 요청 경로 및 메서드를 작업으로 전달합니다.

다음 다이어그램은 확인된 권한이 있는 API Gateway API의 권한 부여 흐름을 보여줍니다. 자세한 내용은 Amazon Verified Permissions 사용 설명서의 API 링크 정책 저장소를 참조하세요.



Verified Permissions는 사용자 풀 그룹에 대한 API 권한 부여를 구성합니다. ID 및 액세스 토큰 모두에 `cognito:groups` 클레임이 포함되어 있으므로 정책 저장소는 다양한 애플리케이션 컨텍스트에서 API에 대한 역할 기반 액세스 제어(RBAC)를 관리할 수 있습니다.

정책 저장소 설정 선택

정책 저장소에서 ID 소스를 구성할 때 액세스 또는 ID 토큰을 처리할지 여부를 선택해야 합니다. 이 결정은 정책 엔진이 작동하는 방식에 중요합니다. ID 토큰에는 사용자 속성이 포함됩니다. 액세스 토큰에는 사용자 액세스 제어 정보인 OAuth 범위가 포함되어 있습니다. 두 토큰 유형 모두 그룹 멤버쉽 정보가 있지만 일반적으로 Verified Permissions 정책 저장소가 있는 RBAC용 액세스 토큰을 사용하는 것이 좋습니다. 액세스 토큰은 권한 부여 결정에 기여할 수 있는 범위를 가진 그룹 멤버십에 추가됩니다. 액세스 토큰의 클레임은 권한 부여 요청의 컨텍스트가 됩니다.

사용자 풀을 ID 소스로 구성할 때 사용자 및 그룹 엔터티 유형도 구성해야 합니다. 엔터티 유형은 Verified Permissions 정책에서 참조할 수 있는 보안 주체, 작업 및 리소스 식별자입니다. 정책 저장소의 엔터티는 멤버십 관계를 가질 수 있으며, 여기서 한 엔터티는 상위 엔터티의 멤버일 수 있습니다. 멤버십을 사용하면 보안 주체 그룹, 작업 그룹 및 리소스 그룹을 참조할 수 있습니다. 사용자 풀 그룹의 경우 지정한 사용자 엔터티 유형은 그룹 엔터티 유형의 멤버여야 합니다. API 링크 정책 저장소를 설정하거나 Verified Permissions 콘솔의 안내형 설정을 수행하면 정책 저장소는 자동으로 이 상위-멤버 관계를 갖습니다.

ID 토큰은 RBAC를 속성 기반 액세스 제어(ABAC)와 결합할 수 있습니다. API 링크 정책 저장소를 생성한 후 사용자 속성 및 그룹 멤버십으로 정책을 개선할 수 있습니다. ID 토큰의 속성 클레임은 권한 부여 요청의 보안 주체 속성이 됩니다. 정책은 보안 주체 속성을 기반으로 권한 부여 결정을 내릴 수 있습니다.

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

역할 기반 API 권한 부여에 대한 정책 예시

다음 예시 정책은 PetStore 예시 REST API에 대한 Verified Permissions 정책 저장소를 설정하여 생성되었습니다.

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

Verified Permissions는 다음과 같은 경우 애플리케이션의 권한 요청에 대해 `Allow` 결정을 반환합니다.

1. 애플리케이션이 `Authorization` 헤더의 ID 또는 액세스 토큰을 보유자 토큰으로 전달한 경우.

2. 애플리케이션에서 문자열 `MyGroup`가 포함된 `cognito:groups` 클레임이 있는 토큰을 전달한 경우.
3. 애플리케이션이 `https://myapi.example.com/pets` 또는 `https://myapi.example.com/pets/scrappy` 등에 `HTTP GET` 요청을 한 경우.

Amazon Cognito 사용자에 대한 정책 예시

사용자 풀은 API 요청 이외의 조건에서 Verified Permissions에 대한 권한 부여 요청을 생성할 수도 있습니다. 애플리케이션의 모든 액세스 제어 결정을 정책 저장소에 제출할 수 있습니다. 예를 들어 요청을 통해 네트워크를 전송하기 전에 속성 기반 액세스 제어를 사용하여 Amazon DynamoDB 또는 Amazon S3 보안을 보완하여 할당량 사용을 줄일 수 있습니다.

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

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

다음 요청 본문은 `Allow` 응답을 생성합니다.

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

```
      "entityType": "Photo"
   }
}
```

Verified Permissions 정책에 보안 주체를 지정하려면 다음과 같은 형식을 사용합니다.

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

다음은 ID가 `us-east-1_Example`이고 sub 또는 사용자 ID가 `973db890-092c-49e4-a9d0-912a4c0a20c7`인 사용자 풀의 사용자에 대한 보안 주체의 예입니다.

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

Verified Permissions 정책에서 사용자 그룹을 지정하려면 다음 형식을 사용하세요.

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

속성 기반 액세스 제어

앱에 대한 Verified Permissions 권한 부여와 AWS 자격 증명을 위한 Amazon Cognito 자격 증명 풀의 액세스 제어 기능에 대한 속성은 모두 속성 기반 액세스 제어(ABAC)의 형태입니다. 다음은 Verified Permissions와 Amazon Cognito ABAC의 기능을 비교한 것입니다. ABAC에서 시스템은 엔터티의 속성을 검사하고 사용자가 정의한 조건에 따라 권한 부여 결정을 내립니다.

서비스	프로세스	결과
Amazon Verified Permissions	Returns an ## or ## decision from analysis of a user pool JWT.	Access to application resources succeeds or fails based on Cedar policy evaluation.

서비스	프로세스	결과
Amazon Cognito identity pools (attributes for access control)	Assigns 세션 태그 to your user based on their attributes. IAM policy conditions can check tags ## or ## user access to AWS 서비스.	A tagged session with temporary AWS credentials for an IAM role.

사용자 풀 엔드포인트 및 관리형 로그인 참조

Amazon Cognito에는 사용자 풀 API와 OAuth 2.0 권한 부여 서버의 두 가지 사용자 풀 인증 모델이 있습니다. 애플리케이션 백엔드에서 AWS SDK를 사용하여 OpenID Connect(OIDC) 토큰을 검색하려는 경우 API를 사용합니다. 사용자 풀을 OIDC 공급자로 구현하려는 경우 권한 부여 서버를 사용합니다. 권한 부여 서버는 페더레이션 로그인, API 및 액세스 토큰 범위를 사용한 M2M 권한 부여, 및 관리형 로그인과 같은 기능을 추가합니다. 사용자 풀 수준 또는 앱 클라이언트 수준에서 구성된 API 및 OIDC 모델을 각각 단독으로 또는 함께 사용할 수 있습니다. 이 섹션은 OIDC 모델 구현을 위한 참조 자료입니다. 두 가지 인증 모드에 대한 자세한 내용은 API, OIDC 및 관리형 로그인 페이지 인증 이해 섹션을 참조하세요.

Amazon Cognito는 도메인을 사용자 풀에 할당할 때 여기에 나열된 공개 웹 페이지를 활성화합니다. 도메인은 모든 앱 클라이언트의 중앙 액세스 지점 역할을 합니다. 여기에는 사용자가 가입 및 로그인(Login 엔드포인트)하고 로그아웃(Logout 엔드포인트)할 수 있는 관리형 로그인이 포함됩니다. 이러한 리소스에 대한 자세한 내용은 사용자 풀 관리형 로그인 섹션을 참조하세요.

이 페이지에는 사용자 풀이 타사 SAML, OpenID Connect(OIDC), OAuth 2.0 아이덴티티 제공업체(IdP)와 통신할 수 있는 공개 웹 리소스도 포함되어 있습니다. 페더레이션 ID 제공업체를 사용하여 사용자를 로그인하려면 사용자가 관리형 로그인 Login 엔드포인트 또는 OIDC 권한 부여 엔드포인트에 대한 대화형 요청을 시작해야 합니다. 권한 부여 엔드포인트는 사용자를 관리형 로그인 또는 IdP 로그인 페이지로 리디렉션합니다.

앱은 Amazon Cognito 사용자 풀 API를 사용하여 로컬 사용자를 로그인할 수도 있습니다. 로컬 사용자는 외부 IdP를 통한 페더레이션 없이 사용자 풀 디렉터리에만 존재합니다.

Amazon Cognito는 관리형 로그인 외에도 Android, iOS, JavaScript용 SDK 등과 통합됩니다. SDK는 Amazon Cognito API 서비스 엔드포인트를 사용하여 사용자 풀 API 엔드포인트 작업을 수행할 수 있는 도구를 제공합니다. 서비스 엔드포인트에 대한 자세한 내용은 Amazon Cognito Identity endpoints and quotas(Amazon Cognito 자격 증명 엔드포인트 및 할당량)를 참조하세요.

> **Warning**
>
> Amazon Cognito 도메인에 대한 최종 엔터티 또는 중간 전송 계층 보안(TLS) 인증서를 고정하지 마십시오.는 모든 사용자 풀 엔드포인트 및 접두사 도메인에 대한 모든 인증서를 AWS 관리합니다. Amazon Cognito 인증서를 지원하는 신뢰 체인의 인증 기관(CA)은 동적으로 교체되고 갱신됩니다. 앱을 중간 또는 리프 인증서에 고정하면가 인증서를 AWS 교체할 때 앱이 예고 없이 실패할 수 있습니다.

애플리케이션을 사용 가능한 Amazon 루트 인증서에 고정하세요. 자세한 내용은 AWS Certificate Manager 사용 설명서의 인증서 고정에서 모범 사례 및 권장 사항을 참조하세요.

사용자 풀 API의 서비스 엔드포인트 목록은 AWS 일반 AWS 리전참조의 서비스 엔드포인트를 참조하세요.

주제

- 사용자 대화형 관리형 로그인 및 클래식 호스팅 UI 엔드포인트
- ID 제공업체 및 신뢰 당사자 엔드포인트
- OAuth 2.0 권한 부여
- 권한 부여 코드 부여에서 PKCE 사용
- 관리형 로그인 및 페더레이션 오류 응답

사용자 대화형 관리형 로그인 및 클래식 호스팅 UI 엔드포인트

도메인을 사용자 풀에 추가하면 Amazon Cognito는 이 섹션의 관리형 로그인 엔드포인트를 활성화합니다. 이 엔드포인트는 사용자가 사용자 풀의 핵심 인증 작업을 완료할 수 있는 웹페이지입니다. 여기에는 암호 관리, 다중 인증(MFA), 속성 확인 페이지가 포함되어 있습니다.

관리형 로그인을 구성하는 웹 페이지는 고객과의 대화형 사용자 세션을 위한 프런트 엔드 웹 애플리케이션입니다. 앱은 사용자 브라우저에서 관리형 로그인을 호출해야 합니다. Amazon Cognito는 이 장의 웹 페이지에 대한 프로그래밍 방식의 액세스를 지원하지 않습니다. JSON 응답을 반환하는 ID 제공업체 및 신뢰 당사자 엔드포인트의 페더레이션 엔드포인트는 앱 코드에서 직접 쿼리할 수 있습니다. 권한 부여 엔드포인트는 관리형 로그인 또는 IdP 로그인 페이지로 리디렉션되며 사용자 브라우저에서도 열어야 합니다.

모든 사용자 풀 엔드포인트는 IPv4 및 IPv6 원본 IP 주소의 트래픽을 허용합니다.

이 가이드의 주제에서는 자주 사용되는 관리형 로그인 및 클래식 호스팅 UI 엔드포인트에 대해 자세히 설명합니다. 관리형 로그인과 호스팅 UI의 차이는 표시되지만 작동하지 않습니다. `/passkeys/add`를 제외한 모든 경로는 관리형 로그인 브랜딩의 두 버전 간에 공유됩니다.

Amazon Cognito는 도메인을 사용자 풀에 할당할 때 다음 웹 페이지를 사용할 수 있도록 합니다.

관리형 로그인 엔드포인트

엔드포인트 URL	설명	액세스 방법
https://### # ###/login	사용자 풀 로컬 및 페더레이션 사용자를 로그인합니다.	권한 부여 엔드포인트, /logout 및 /confirmforgotPassword 같은 엔드포인트에서 리디렉션합니다. Login 엔드포인트 섹션을 참조하세요.
https://### # ###/logout	사용자 풀 사용자를 로그아웃합니다.	직접 링크. Logout 엔드포인트 섹션을 참조하세요.
https://### # ###/confirmUser	사용자 계정을 확인하기 위해 이메일 링크를 선택한 사용자를 확인합니다.	이메일 메시지에서 사용자가 선택한 링크.
https://### # ###/signup	새 사용자를 등록합니다. /login 페이지에서 사용자가 Sign up(가입)을 선택하면 /signup으로 이동하게 됩니다.	/oauth2/authorize 과 파라미터가 동일한 직접 링크.
https://### # ###/confirm	가입한 사용자에게 사용자 풀이 확인 코드를 보낸 후 코드를 입력하라는 메시지가 사용자에게 표시됩니다.	/signup에서 리디렉션만.
https://### # ###/forgotPassword	사용자에게 사용자 이름을 묻는 메시지를 표시하고 암호 재설정 코드를 보냅니다. /login 페이지에서 사용자가 Forgot your password?(암호가 생각나지 않는 경우)를 선택하면 /forgotPassword 로 이동하게 됩니다.	1. /login에 있는 암호 찾기 링크에서. 2. /oauth2/authorize 과 파라미터가 동일한 직접 링크.
https://### # ###/confirmforgotPassword	사용자에게 암호 재설정 코드와 새 암호를 입력하라	/forgotPassword 에서 리디렉션만.

엔드포인트 URL	설명	액세스 방법
	는 메시지가 표시됩니다. /forgotPassword 페이지에서 사용자가 Reset your password(암호 재설정)를 선택하면 /confirmforgotPassword 로 이동하게 됩니다. 사용자가 유효한 재설정 코드와 새 암호를 제출하면는 새 암호로 로그인/login하도록 리/confirmforgotPassword 디렉션합니다.	
https://### # ###/resendcode	사용자 풀의 가입한 사용자에게 새 확인 코드를 보냅니다.	/confirm에 있는 새 코드 보내기 링크에서 리디렉션만.
https://### # ###/passkeys/add	새 패스키를 등록합니다. 관리형 로그인에서만 사용할 수 있습니다.	• 패스키 인증을 지원하는 앱 클라이언트에서 확인 후 가입 흐름. • /oauth2/authorize 과 파라미터가 동일한 직접 링크.

주제

- 관리형 로그인의 로그인 엔드포인트: /login
- 관리형 로그인 로그아웃 엔드포인트: /logout

관리형 로그인의 로그인 엔드포인트: **/login**

로그인 엔드포인트는 인증 서버이자 권한 부여 엔드포인트의 리디렉션 대상입니다. ID 제공업체를 지정하지 않는 경우 관리형 로그인의 진입점이 됩니다. 로그인 엔드포인트에 대한 리디렉션을 생성하는 경우, 로그인 엔드포인트는 로그인 페이지를 로드하고 클라이언트에 대해 구성된 인증 옵션을 사용자에게 표시합니다.

> **Note**
>
> 로그인 엔드포인트는 관리형 로그인의 구성 요소입니다. 앱에서 로그인 엔드포인트로 리디렉션되는 페더레이션 및 관리형 로그인 페이지를 호출합니다. 사용자가 로그인 엔드포인트에 직접 액세스하는 것은 모범 사례가 아닙니다.

/login 획득

`/login` 엔드포인트는 사용자의 초기 요청에서만 `HTTPS GET`을 지원합니다. 앱은 Chrome 또는 Firefox와 같은 브라우저에서 페이지를 호출합니다. 권한 부여 엔드포인트에서 `/login`으로 리디렉션하는 경우 초기 요청에서 제공한 모든 파라미터가 함께 전달됩니다. 로그인 엔드포인트는 권한 부여 엔드포인트의 모든 요청 파라미터를 지원합니다. 로그인 엔드포인트에 직접 액세스할 수도 있습니다. 가장 좋은 방법은 모든 사용자 세션을 `/oauth2/authorize`에서 시작하는 것입니다.

예 - 사용자에게 로그인하라는 메시지를 표시합니다.

이 예제는 로그인 화면에 표시됩니다.

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/login?
                response_type=code&
                client_id=1example23456789&
                redirect_uri=https://YOUR_APP/redirect_uri&
                state=STATE&
                scope=openid+profile+aws.cognito.signin.user.admin
```

예 - 응답

인증 서버는 권한 부여 코드 및 상태와 함께 사용자의 앱으로 리디렉션합니다. 이 서버는 코드 및 상태를 조각이 아닌 쿼리 문자열 파라미터로 반환해야 합니다.

```
HTTP/1.1 302 Found
                    Location: https://YOUR_APP/redirect_uri?
code=AUTHORIZATION_CODE&state=STATE
```

사용자가 시작한 로그인 요청

사용자는 /login 엔드포인트를 로드한 후 사용자 이름과 암호를 입력하고 로그인을 선택할 수 있습니다. 이렇게 하면 GET 요청과 동일한 헤더 요청 파라미터와 사용자 이름, 암호 및 디바이스 지문이 포함된 요청 본문을 갖는 HTTPS POST 요청이 생성됩니다.

단계적 인증 요청

/login 엔드포인트는 단계적 인증을 위해 다음 URL 파라미터를 허용합니다. 이러한 파라미터는 권한 부여 엔드포인트와 동작이 동일합니다. ACR 수준 및 AMR 값에 대한 자세한 내용은 섹션을 참조하세요the section called “ACR 및 AMR 클레임을 사용한 인증 수준”.

acr_values

가장 높은 것부터 가장 낮은 것까지 대상 ACR 인증 수준 URIs의 공백으로 구분된 URL 인코딩 목록입니다. Amazon Cognito는 목록을 왼쪽에서 오른쪽으로 평가하고, 사용자가 충족할 수 있는 첫 번째 유효 수준을 적용하고, 인식하지 못하는 값을 무시합니다. 이 파라미터는 TARGET_ACR_VALUES API 파라미터와 동일하게 동작합니다. Essentials 또는 Plus 기능 플랜이 필요합니다.

max_age

사용자가 마지막으로 인증된 이후 최대 초 수를 지정하는 음수가 아닌 정수입니다. 현재 시간에서를 뺀 값이 보다 큰 경우 max_age사용자는 한 단계만 올라가auth_time는 대신 처음부터 다시 인증해야 합니다. 이 동작은 MAX_AGE API 파라미터와 동일합니다.

이러한 파라미터를 사용하는 예제 요청은 권한 부여 엔드포인트를 참조하세요.

acr_values와 결합하여 민감한 작업에 대한 최신 인증과 특정 인증 수준을 max_age 요구합니다.

기능 부족 계획의 동작은 API와 다릅니다. 관리형 로그인 /login 엔드포인트는 오류 없이 정상 인증을 자동으로 무시acr_values하고 진행합니다. API는를 반환합니다FeatureUnavailableInTierException. 자세한 내용은 기능 계획 요구 사항 섹션을 참조하세요.

관리형 로그인 로그아웃 엔드포인트: /logout

/logout 엔드포인트는 리디렉션 엔드포인트입니다. 사용자를 로그아웃하고 앱 클라이언트의 권한 부여된 로그아웃 URL 또는 /login 엔드포인트로 리디렉션합니다. /logout 엔드포인트에 대한 GET 요청에서 사용 가능한 파라미터는 Amazon Cognito 관리형 로그인 사용 사례에 맞게 조정됩니다.

로그아웃 엔드포인트는 고객과의 대화형 사용자 세션을 위한 프런트 엔드 웹 애플리케이션입니다. 앱은 사용자 브라우저에서 이 엔드포인트와 다른 관리형 로그인 엔드포인트를 호출해야 합니다.

다시 로그인할 수 있도록 사용자를 관리형 로그인으로 리디렉션하려면 요청에 `redirect_uri` 파라미터를 추가하세요. `redirect_uri` 파라미터가 있는 `logout` 요청에는`client_id`, `response_type`, `scope` 등 Login 엔드포인트에 대한 후속 요청을 위한 파라미터도 포함되어야 합니다.

선택한 페이지로 사용자를 리디렉션하려면 허용된 로그아웃 URL을 앱 클라이언트에 추가하세요. `logout` 엔드포인트에 대한 사용자의 요청에 `logout_uri` 및 `client_id` 파라미터를 추가합니다. `logout_uri` 값이 앱 클라이언트의 허용된 로그아웃 URL 중 하나인 경우 Amazon Cognito는 사용자를 해당 URL로 리디렉션합니다.

SAML 2.0 IdPs용 단일 로그아웃(SLO)을 사용하면 Amazon Cognito는 먼저 사용자를 IdP 구성에서 정의한 SLO 엔드포인트로 리디렉션합니다. IdP가 사용자를 `saml2/logout`으로 리디렉션하면 Amazon Cognito는 사용자의 요청에 따라 `redirect_uri` 또는 `logout_uri`로 한 번 더 리디렉션하여 응답합니다. 자세한 내용은 단일 로그아웃으로 SAML 사용자 로그아웃 섹션을 참조하세요.

로그아웃 엔드포인트는 OIDC 또는 소셜 자격 ID 제공업체(idP)에서 사용자를 로그아웃시키지 않습니다. 외부 IdP를 사용하여 세션에서 사용자를 로그아웃하려면 해당 공급자의 로그아웃 페이지로 안내합니다.

GET /logout

`/logout` 엔드포인트는 `HTTPS GET`만 지원합니다. 사용자 풀 클라이언트는 일반적으로 시스템 브라우저를 통해 이 요청을 수행합니다. 브라우저는 일반적으로 Android의 Custom Chrome Tab 또는 iOS의 Safari View Control입니다.

요청 파라미터

client_id

앱에 대한 앱 클라이언트 ID입니다. 앱 클라이언트 ID를 얻으려면 해당 앱을 사용자 풀에 등록해야 합니다. 자세한 내용은 앱 클라이언트를 사용한 애플리케이션별 설정 섹션을 참조하세요.

필수 사항입니다.

logout_uri

logout_uri 파라미터가 있는 사용자 지정 로그아웃 페이지로 사용자를 리디렉션합니다. 로그아웃한 사용자를 리디렉션하려는 앱 클라이언트 로그아웃 URL로 값을 설정합니다. logout_uri는 client_id

파라미터와 함께만 사용합니다. 자세한 내용은 앱 클라이언트를 사용한 애플리케이션별 설정 섹션을 참조하세요.

logout_uri 파라미터를 사용하여 사용자를 다른 앱 클라이언트의 로그인 페이지로 리디렉션할 수도 있습니다. 다른 앱 클라이언트의 로그인 페이지를 앱 클라이언트의 허용된 콜백 URL로 설정합니다. `/logout` 엔드포인트에 대한 요청에서 logout_uri 파라미터의 값을 URL 인코딩된 로그인 페이지로 설정합니다.

Amazon Cognito에서는 `/logout` 엔드포인트를 요청할 때 logout_uri 또는 redirect_uri 파라미터가 필요합니다. logout_uri 파라미터는 사용자를 다른 웹 사이트로 리디렉션합니다. logout_uri 파라미터와 redirect_uri 파라미터가 모두 `/logout` 엔드포인트에 대한 요청에 포함된 경우 Amazon Cognito는 logout_uri 파라미터를 독점적으로 활용하여 redirect_uri 파라미터를 재정의합니다.

nonce

(선택 사항) 요청에 추가할 수 있는 임의 값입니다. 제공한 임시 값은 Amazon Cognito가 발행하는 ID 토큰에 포함되어 있습니다. 재생 공격을 방지하기 위해 앱은 ID 토큰의 `nonce` 클레임을 검사하고 생성한 것과 비교할 수 있습니다. `nonce` 클레임에 대한 자세한 내용은 OpenID Connect 표준의 ID 토큰 유효성 검사를 참조하세요.

redirect_uri

사용자를 로그인 페이지로 리디렉션하여 redirect_uri 파라미터를 사용하여 인증합니다. 다시 로그인한 사용자를 리디렉션하려는 앱 클라이언트 허용된 콜백 URL로 값을 설정합니다. `/login` 엔드포인트에 전달하려는 client_id, scope, state, response_type 파라미터를 추가합니다.

Amazon Cognito에서는 `/logout` 엔드포인트를 요청할 때 logout_uri 또는 redirect_uri 파라미터가 필요합니다. 사용자를 `/login` 엔드포인트로 리디렉션하여 토큰을 다시 인증하고 앱에 전달하려면 redirect_uri 파라미터를 추가하세요. `/logout` 엔드포인트에 대한 요청에 logout_uri 및 redirect_uri 파라미터가 모두 포함된 경우 Amazon Cognito는 redirect_uri 파라미터를 재정의하고 logout_uri 파라미터만 단독으로 처리합니다.

response_type

사용자가 로그인한 후 Amazon Cognito로부터 수신할 OAuth 2.0 응답입니다. `code` 및 `token`은 response_type 파라미터에 대한 유효한 값입니다.

redirect_uri 파라미터를 사용하는 경우 필수입니다.

state

애플리케이션이 요청에 state 파라미터를 추가하면 `/oauth2/logout` 엔드포인트가 사용자를 리디렉션할 때 Amazon Cognito가 해당 값을 앱으로 반환합니다.

이 값을 요청에 추가하여 CSRF 공격으로부터 보호할 수 있습니다.

`state` 파라미터의 값을 URL 인코딩 JSON 문자열로 설정할 수 없습니다. `state` 파라미터에서 이 형식과 일치하는 문자열을 전달하려면 문자열을 base64로 인코딩한 다음 애플리케이션에서 디코딩하면 됩니다.

redirect_uri 파라미터를 사용하는 경우 적극 권장됩니다.

범위

redirect_uri 파라미터를 사용하여 로그아웃한 후 Amazon Cognito로부터 요청할 OAuth 2.0 범위입니다. Amazon Cognito가 `/logout` 엔드포인트에 대한 요청에 scope 파라미터를 사용하여 사용자를 `/login` 엔드포인트로 리디렉션합니다.

redirect_uri 파라미터를 사용하는 경우에 선택 사항입니다. scope 파라미터를 포함하지 않는 경우 Amazon Cognito가 scope 파라미터를 사용하여 사용자를 `/login` 엔드포인트로 리디렉션합니다. Amazon Cognito가 사용자를 리디렉션하고 `scope`를 자동으로 채우는 경우 이 파라미터에 앱 클라이언트의 권한 부여된 모든 범위가 포함됩니다.

요청 예시

예 – 로그아웃하고 클라이언트로 리디렉션

요청에 `logout_uri` 및 `client_id`가 포함된 경우 Amazon Cognito는 다른 모든 요청 파라미터를 무시하고 사용자 세션을 `logout_uri`의 값에 속한 URL로 리디렉션합니다. 이 URL은 앱 클라이언트의 승인된 로그아웃 URL이어야 합니다.

다음은 로그아웃 및 `https://www.example.com/welcome` 리디렉션 요청의 예시입니다.

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/logout?
  client_id=1example23456789&
  logout_uri=https%3A%2F%2Fwww.example.com%2Fwelcome
```

예 – 로그아웃하고 사용자에게 다른 사용자로 로그인하라는 메시지 표시

요청에 `logout_uri`가 없지만 승인 엔드포인트에 올바른 형식의 요청을 구성하는 파라미터를 제공하는 경우 Amazon Cognito는 사용자를 관리형 로그인으로 리디렉션합니다. 로그아웃 엔드포인트는 원래 요청의 파라미터를 리디렉션 대상에 추가합니다.

로그아웃 요청에 추가하는 추가 파라미터는 요청 파라미터의 목록에 있어야 합니다. 예를 들어 로그아웃 엔드포인트는 `identity_provider` 또는 `idp_identifier` 파라미터를 사용한 자동 IdP 리디렉

션을 지원하지 않습니다. 로그아웃 엔드포인트에 대한 요청의 파라미터 `redirect_uri`는 로그아웃 URL이 아니라 승인 엔드포인트로 전달하려는 로그인 후 URL입니다.

다음은 사용자를 로그아웃시키고, 로그인 페이지로 리디렉션하고, 로그인 후 `https://www.example.com`으로 인증 코드를 제공하는 요청의 예시입니다.

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/logout?
  response_type=code&
  client_id=1example23456789&
  redirect_uri=https%3A%2F%2Fwww.example.com&
  state=example-state-value&
  nonce=example-nonce-value&
  scope=openid+profile+aws.cognito.signin.user.admin
```

ID 제공업체 및 신뢰 당사자 엔드포인트

페더레이션 엔드포인트는 사용자 풀에서 사용하는 인증 표준 중 하나의 목적을 제공하는 사용자 풀 엔드포인트입니다. 여기에는 ID 제공업체 및 신뢰 당사자 모두의 사용자 풀 역할을 위한 SAML ACS URLs, OIDC 검색 엔드포인트 및 서비스 엔드포인트가 포함됩니다. 페더레이션 엔드포인트는 인증 흐름을 시작하고, IdP로부터 인증 증명을 수신하고, 클라이언트에 토큰을 발급합니다. IdP, 애플리케이션 및 관리자와 상호 작용하지만 사용자와는 상호 작용하지 않습니다.

이 페이지 이후의 전체 페이지 주제에는 사용자 풀에 도메인을 추가할 때 사용할 수 있는 OAuth 2.0 및 OIDC 공급자 엔드포인트에 대한 세부 정보가 나와 있습니다. 다음 차트는 모든 페더레이션 엔드포인트의 목록입니다.

사용자 풀 도메인의 예는 다음과 같습니다.

1. 접두사 도메인: `mydomain.auth.us-east-1.amazoncognito.com`
2. 사용자 지정 도메인`auth.example.com`

사용자 풀 페더레이션 엔드포인트

엔드포인트 URL	설명	액세스 방법
https://### # ###/oauth2/authorize	사용자를 관리형 로그인으로 리디렉션하거나 IdP를 사용하여 로그인합니다.	사용자 인증을 시작하기 위해 고객 브라우저에서 호출됩니다. 권한 부여 엔드포인트 섹션을 참조하세요.

엔드포인트 URL	설명	액세스 방법
https://*### # ###*/oauth2/token	인증 코드 또는 클라이언트 보안 인증 요청에 따라 토큰을 반환합니다.	앱에서 토큰을 검색하도록 요청했습니다. Token 엔드포인트 섹션을 참조하세요.
https://*### # ###*/oauth2/userInfo	액세스 토큰의 OAuth 2.0 범위 및 사용자 자격 증명을 기반으로 사용자 속성을 반환합니다.	앱에서 사용자 프로필을 검색하도록 요청했습니다. userInfo 엔드포인트 섹션을 참조하세요.
https://*### # ###*/oauth2/revoke	새로 고침 토큰 및 연결된 액세스 토큰을 취소합니다.	앱에서 토큰을 취소하도록 요청했습니다. 취소 엔드포인트 섹션을 참조하세요.
https://cognito-idp.*Region*.amazonaws.com/*your user pool ID*/.well-known/openid-configuration	사용자 풀의 OIDC 아키텍처 디렉터리입니다.[1]	앱에서 사용자 풀 발급자 메타데이터를 찾도록 요청했습니다.
https://cognito-idp.*Region*.amazonaws.com/*your user pool ID*/.well-known/jwks.json	Amazon Cognito 토큰을 검증하는 데 사용할 수 있는 퍼블릭 키입니다.[2]	앱에서 JWT를 확인하도록 요청했습니다.
https://*Your user pool domain*/oauth2/idpresponse	소셜 아이덴티티 제공업체는 권한 부여 코드를 통해 사용자를 이 엔드포인트로 리디렉션해야 합니다. Amazon Cognito는 페더레이션 사용자를 인증할 때 토큰의 코드를 사용합니다.	OIDC IdP 로그인에서 IdP 클라이언트 콜백 URL로 리디렉션됩니다.
https://*Your user pool domain*/saml2/idpresponse	SAML 2.0 ID 제공업체와의 통합을 위한 ACS(Assertion Consumer Response) URL입니다.	SAML 2.0 IdP에서 ACS URL 또는 IdP 시작 로그인의 시작 지점[3]으로 리디렉션됩니다.

엔드포인트 URL	설명	액세스 방법
https://### # ###/saml2/logout	SAML 2.0 ID 제공업체와의 통합을 위한 단일 로그아웃(SLO) URL입니다.	SAML 2.0 IdP에서 단일 로그아웃(SLO) URL로 리디렉션됩니다. POST 바인딩만 허용합니다.

[1] `openid-configuration` 문서는 엔드포인트가 OIDC 및 OAuth2 사양을 준수하도록 유지하는 추가 정보로 언제든지 업데이트될 수 있습니다.

[2] `jwks.json`JSON 파일은 언제든지 새 퍼블릭 토큰 서명 키로 업데이트할 수 있습니다.

[3] IdP 시작 SAML 로그인에 대한 자세한 내용은 섹션을 참조하세요IdP 시작 SAML 로그인 구현.

OpenID Connect 및 OAuth 표준에 대한 자세한 내용은 OpenID Connect 1.0 및 OAuth 2.0을 참조하세요.

OIDC 발급자로서의 Amazon Cognito 사용자 풀

Amazon Cognito 사용자 풀은 OpenID Connect(OIDC) 자격 증명 공급자 역할을 하며 애플리케이션 라이브러리가 OIDC 페더레이션에 사용할 수 있는 발급자 역할을 합니다. OIDC 페더레이션용 애플리케이션 라이브러리는 아래에 설명된 두 가지 경로를 자동 검색 엔드포인트로 참조할 수 있습니다. 이 엔드포인트는의 JSON 웹 키 세트(JWKS) `/.well-known/jwks.json` 및의 OIDC 검색 메타데이터에 대한 액세스를 제공합니다. `/.well-known/openid-configuration`여기서 애플리케이션은 권한 부여, 토큰 및 userInfo 엔드포인트를 검색할 수 있습니다.

OIDC 자동 검색을 지원하는 애플리케이션은 이러한 잘 알려진 엔드포인트를 쿼리하여 자동으로 구성할 수 있습니다. 자동 검색을 지원하지 않는 애플리케이션의 경우 이전 섹션에 나열된 특정 OIDC 엔드포인트로 애플리케이션을 하드코딩할 수 있습니다.

사용자 풀 발급자 유형

원래 발급자

사용자 풀에 대한 현재 기본 발급자 구성입니다. 발급자 URL은 사용자 풀의 리전에서 호스팅되며 해당 리전에 고유한 OIDC 엔드포인트를 제공합니다.

원래 발급자는 형식을 사용합니다`https://cognito-idp.us-east-1.amazonaws.com/us-east-1_EXAMPLE`.

업데이트된 발급자

다중 리전 복제를 포함하여 모든 사용자 풀에 권장됩니다. 업데이트된 발급자는 여러 리전에서 동일한 JWKS 콘텐츠를 호스팅하여 복원력과 효율성을 향상시킵니다.

업데이트된 발급자는 형식을 사용합니다. `https://issuer-cognito-idp.us-east-1.amazonaws.com/us-east-1_EXAMPLE`여기서 `##`은 사용자 풀 AWS 리전 의 기본입니다.

> **Note**
>
> 업데이트된 발급자 유형은 Application Load Balancer 인증(`authenticate-cognito`작업)을 위한 기본 Amazon Cognito 통합과 호환되지 않습니다. Application Load Balancer에서 업데이트된 발급자를 사용하려면 대신 일반 OIDC 통합(`authenticate-oidc`작업)을 구성하고 사용자 풀의 OIDC 엔드포인트를 제공합니다. 자세한 내용은 Application Load Balancer를 사용하여 사용자 인증을 참조하세요.

검색 문서 변경 준비

`openid-configuration` 문서는 표준 OpenID Connect 검색 문서입니다. Amazon Cognito는 시간이 지남에 따라 메타데이터 항목을 추가하여 OpenID Connect 및 OAuth 2.0 사양을 준수하고 OIDC 클라이언트 라이브러리와의 호환성을 개선할 수 있습니다. 이러한 추가 사항은 이전 버전과 호환됩니다. Amazon Cognito는 새 멤버를 추가하지만 애플리케이션이 이미 읽은 멤버의 이름을 바꾸거나, 유형을 변경하거나, 제거하지 않습니다. 문서가 증가함에 따라 계속 작동하도록 클라이언트를 작성합니다.

인식할 수 없는 멤버 무시

OpenID Connect Discovery 사양에 따라 이해되지 않는 메타데이터 멤버를 선택 사항으로 취급합니다. 문서에 애플리케이션이 사용하지 않는 멤버가 포함된 경우 거부하거나 실패하거나 오류를 발생시키지 마세요.

멤버의 집합 또는 순서를 하드 코딩하지 마세요.

문서를 고정 구조가 아닌 일반 JSON 객체로 구문 분석합니다. 멤버는 어떤 순서로든 표시될 수 있으며, 멤버 세트는 요청 간에 시간 경과에 따라 변경될 수 있습니다. 각 멤버를 이름으로 처리하고 각 멤버를 사양에서 정의하는 유형으로 읽습니다.

현재 문서에 대해 엄격한 클라이언트 테스트

클라이언트가 검색 문서를 엄격하게 검증하는 경우 사용자 풀의 `/.well-known/openid-configuration` 엔드포인트에서 현재 문서를 가져오고 특정 멤버 집합에 의존하기 전에 클라이언트가 이를 수락하는지 확인합니다.

Amazon Cognito 토큰은 동일한 애디티브 전달 호환 접근 방식을 따릅니다. ID 또는 액세스 토큰에서 클레임을 구문 분석하는 방법에 대한 자세한 내용은 섹션을 참조하세요사용자 풀 토큰을 안전하게 구문 분석.

주제

- 리디렉션 및 권한 부여 엔드포인트
- 토큰 엔드포인트
- 사용자 속성 엔드포인트
- 토큰 취소 엔드포인트
- IdP SAML 어설션 엔드포인트

리디렉션 및 권한 부여 엔드포인트

`/oauth2/authorize` 엔드포인트는 두 개의 리디렉션 대상을 지원하는 리디렉션 엔드포인트입니다. URL에 `identity_provider` 또는 `idp_identifier` 파라미터를 포함하면 사용자를 해당 ID 제공업체(IdP)의 로그인 페이지로 자동 리디렉션합니다. 그렇지 않으면 요청에 포함된 것과 동일한 URL 파라미터를 사용하여 Login 엔드포인트로 리디렉션됩니다.

권한 부여 엔드포인트는 관리형 로그인이나 IdP 로그인 페이지로 리디렉션됩니다. 이 엔드포인트에서 사용자 세션의 대상은 사용자가 브라우저에서 직접 상호 작용해야 하는 웹 페이지입니다.

권한 부여 엔드포인트를 사용하려면 사용자 풀에 다음 사용자 풀 세부 정보에 대한 정보를 제공하는 매개변수를 사용하여 `/oauth2/authorize`에서 사용자 브라우저를 호출하세요.

- 로그인할 앱 클라이언트입니다.
- 리디렉션할 콜백 URL입니다.
- 사용자의 액세스 토큰에서 요청할 OAuth 2.0 범위입니다.
- 필요에 따라 로그인하는 데 사용할 서드 파티 IdP입니다.

Amazon Cognito가 수신 클레임을 검증하는 데 사용하는 `state` 및 `nonce` 파라미터를 제공할 수도 있습니다.

GET **`/oauth2/authorize`**

`/oauth2/authorize` 엔드포인트는 `HTTPS GET`만 지원합니다. 대체로 앱은 사용자의 브라우저에서 이 요청을 시작합니다. HTTPS를 통해서만 `/oauth2/authorize` 엔드포인트에 요청할 수 있습니다.

권한 부여 엔드포인트에서 OpenID Connect(OIDC) 표준의 권한 부여 엔드포인트 정의에 대해 자세히 알아볼 수 있습니다.

요청 파라미터

`response_type`

필수 사항입니다.

응답 유형이며 `code` 또는 `token`이어야 합니다.

`code`의 `response_type`이 있는 성공적인 요청은 권한 부여 코드 부여를 반환합니다. 권한 부여 코드 부여는 Amazon Cognito가 리디렉션 URL에 추가하는 `code` 파라미터입니다. 앱에서는 액세스, ID 및 새로 고침 토큰을 위해 Token 엔드포인트와 코드를 교환할 수 있습니다. 보안 모범 사례로 사용자를 위한 새로 고침 토큰을 받으려면 앱에서 권한 부여 코드 부여를 사용하세요.

`token`의 `response_type`이 있는 성공적인 요청은 암시적 권한 부여를 반환합니다. 암시적 권한 부여는 Amazon Cognito가 리디렉션 URL에 추가하는 ID 및 액세스 토큰입니다. 암시적 권한 부여는 토큰과 잠재적인 식별 정보를 사용자에게 노출하기 때문에 덜 안전합니다. 앱 클라이언트 구성에서 암시적 권한 부여에 대한 지원을 비활성화할 수 있습니다.

`client_id`

필수 사항입니다.

앱 클라이언트 ID입니다.

`client_id` 값은 요청한 사용자 풀에 있는 앱 클라이언트의 ID여야 합니다. 앱 클라이언트는 Amazon Cognito 로컬 사용자 또는 하나 이상의 서드 파티 IdP 로그인을 지원해야 합니다.

`redirect_uri`

필수 사항입니다.

Amazon Cognito가 사용자에게 권한을 부여한 후 인증 서버에서 브라우저를 리디렉션하는 URL입니다.

리디렉션 URI(Uniform Resource Identifier)의 속성은 다음과 같아야 합니다.

- 절대 URI이어야 합니다.
- 클라이언트를 사용하여 URI를 미리 등록했어야 합니다.
- 여기에는 조각 구성 요소가 없어야 합니다.

OAuth 2.0 - Redirection Endpoint 섹션을 참조하세요.

Amazon Cognito를 사용하려면 리디렉션 URI에서 테스트 목적으로 콜백 URL로 설정할 수 있는 HTTPS를 사용해야 합니다(`http://localhost` 제외).

또한 Amazon Cognito는 `myapp://example`과 같은 앱 콜백 URL을 지원합니다.

`state`

선택 사항, 권장됨.

앱이 요청에 state 파라미터를 추가하면 `/oauth2/authorize` 엔드포인트가 사용자를 리디렉션할 때 Amazon Cognito가 해당 값을 앱으로 반환합니다.

이 값을 요청에 추가하여 CSRF 공격으로부터 보호할 수 있습니다.

`state` 파라미터의 값을 URL 인코딩 JSON 문자열로 설정할 수 없습니다. `state` 파라미터에서 이 형식과 일치하는 문자열을 전달하려면 문자열을 base64로 인코딩한 다음 앱에서 디코딩하면 됩니다.

`identity_provider`

선택 사항.

관리형 로그인을 무시하고 사용자를 공급자 로그인 페이지로 리디렉션하려면 이 파라미터를 추가합니다. identity_provider 파라미터의 값은 사용자 프로필에 나타나는 대로 자격 증명 공급자(IdP)의 이름입니다.

- 소셜 공급자의 경우 identity_provider 값 `Facebook`, `Google`, `LoginWithAmazon` 및 `SignInWithApple`를 사용할 수 있습니다.
- Amazon Cognito 사용자 풀의 경우 `COGNITO` 값을 사용합니다.
- SAML 2.0 및 OpenID Connect(OIDC) ID 제공업체(idP)의 경우 사용자 풀의 IdP에 할당한 이름을 사용합니다.

`idp_identifier`

선택 사항.

identity_provider 이름에 대한 대체 이름을 가진 공급자로 리디렉션할 이 파라미터를 추가합니다. Amazon Cognito 콘솔의 소셜 및 외부 공급자 메뉴에서 SAML 2.0 및 OIDC IdP의 식별자를 입력할 수 있습니다.

`scope`

선택 사항.

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 연결되어 있어야 합니다. 그렇지 않으면 런타임 시 무시됩니다.

클라이언트가 범위를 요청하지 않은 경우 인증 서버에서는 클라이언트와 연결된 모든 범위를 사용합니다.

`openid` 범위가 요청될 경우에만 ID 토큰이 반환됩니다. `aws.cognito.signin.user.admin` 범위가 요청된 경우에만 Amazon Cognito 사용자 풀에 대해 액세스 토큰을 사용할 수 있습니다. `phone` 범위도 요청된 경우에만 `email`, `profile` 및 `openid` 범위를 요청할 수 있습니다. 이러한 범위는 ID 토큰 내부로 들어가는 클레임을 지정합니다.

`code_challenge_method`

선택 사항.

문제를 생성하는 데 사용한 해싱 프로토콜입니다. PKCE RFC는 S256 및 일반의 두 가지 메서드를 정의하지만 Amazon Cognito 인증 서버는 S256만 지원합니다.

`code_challenge`

선택 사항.

`code_verifier`에서 생성한 키 코드 교환(PKCE) 문제의 증거입니다. 자세한 내용은 권한 부여 코드 부여에서 PKCE 사용 섹션을 참조하세요.

`code_challenge_method` 파라미터를 지정하는 경우에만 필수입니다.

`nonce`

선택 사항.

요청에 추가할 수 있는 임의 값입니다. 제공한 임시 값은 Amazon Cognito가 발행하는 ID 토큰에 포함되어 있습니다. 재생 공격을 방지하기 위해 앱은 ID 토큰의 `nonce` 클레임을 검사하고 생성한 것과 비교할 수 있습니다. `nonce` 클레임에 대한 자세한 내용은 OpenID Connect 표준의 ID 토큰 유효성 검사를 참조하세요.

`lang`

선택 사항.

사용자 대화형 페이지를 표시할 언어입니다. 관리형 로그인 페이지는 현지화할 수 있지만 호스팅 UI(클래식) 페이지는 현지화할 수 없습니다. 자세한 내용은 관리형 로그인 현지화 섹션을 참조하세요.

`login_hint`

선택 사항.

권한 부여 서버에 전달하려는 사용자 이름 프롬프트입니다. 사용자로부터 사용자 이름, 이메일 주소 또는 전화번호를 수집하고 대상 공급자가 사용자의 로그인 이름을 미리 입력하도록 허용할 수 있습니다. `oauth2/authorize` 엔드포인트에 `login_hint` 파라미터를 제출하고 `idp_identifier` 또는 `identity_provider` 파라미터는 제출하지 않으면 관리형 로그인은 사용자 이름 필드에 힌트 값을 채웁니다. 이 파라미터를 Login 엔드포인트에 전달하고 사용자 이름 값을 자동으로 채울 수도 있습니다.

권한 부여 요청이 OIDC IdPs에 대한 리디렉션을 호출하면 Amazon Cognito는 해당 타사 권한 부여자에게 요청에 `login_hint` 파라미터를 추가합니다. 로그인 힌트를 SAML, Apple, Login With Amazon, Google 또는 Facebook(Meta) IdPs에 전달할 수 없습니다.

`prompt`

선택 사항.

기존 세션의 인증 동작을 제어하는 OIDC 파라미터입니다. 관리형 로그인 브랜딩 버전에서만 사용할 수 있으며 클래식 호스팅 UI에서는 사용할 수 없습니다. OIDC 사양에 대한 자세한 내용은 인증 요청을 참조하세요. `none` 및 `login` 값은 사용자 풀 인증 동작에 영향을 미칩니다.

사용자가 타사 제공업체를 통한 인증을 선택하면 Amazon Cognito는 `none`을 제외한 모든 `prompt` 값을 IdP에 전달합니다. 사용자가 액세스하는 URL에 `identity_provider` 또는 `idp_identifier` 파라미터가 포함되거나 권한 부여 서버가 해당 URL을 로 리디렉션Login 엔드포인트하고 사용 가능한 버튼에서 IdP를 선택하는 경우에도 마찬가지입니다.

프롬프트 파라미터 값

`prompt=none`

Amazon Cognito는 유효한 인증 세션이 있는 사용자에 대해 자동으로 인증을 계속합니다. 이 프롬프트를 사용하면 사용자는 사용자 풀의 여러 앱 클라이언트 간에 자동으로 인증할 수 있습니

다. 사용자가 아직 인증되지 않은 경우 권한 부여 서버는 `login_required` 오류를 반환합니다.

`prompt=login`

Amazon Cognito에서는 사용자가 기존 세션이 있더라도 다시 인증해야 합니다. 사용자의 ID를 다시 확인하려는 경우 이 값을 전송합니다. 기존 세션이 있는 인증된 사용자는 해당 세션을 무효화하지 않고 로그인으로 돌아갈 수 있습니다. 기존 세션이 있는 사용자가 다시 로그인하면 Amazon Cognito에서 새 세션 쿠키를 할당합니다. 이 파라미터는 IdP로 전달할 수도 있습니다. 이 파라미터를 수락하는 IdP도 사용자에게 새 인증 시도를 요청합니다.

`prompt=select_account`

이 값은 로컬 로그인에 영향을 주지 않으며 IdP로 리디렉션되는 요청에서 제출해야 합니다. 권한 부여 요청에 포함된 경우 이 파라미터는 IdP 리디렉션 대상의 URL 경로에 `prompt=select_account`를 추가합니다. IdP가 파라미터를 지원할 때 사용자에게 로그인하려는 계정을 선택하도록 요청합니다.

`prompt=consent`

이 값은 로컬 로그인에 영향을 주지 않으며 IdP로 리디렉션되는 요청에서 제출해야 합니다. 권한 부여 요청에 포함된 경우 이 파라미터는 IdP 리디렉션 대상의 URL 경로에 `prompt=consent`를 추가합니다. IdP가 파라미터를 지원하는 경우 사용자 풀로 다시 리디렉션하기 전에 사용자 동의를 요청합니다.

요청에서 `prompt` 파라미터를 생략하면 관리형 로그인은 기본 동작을 따릅니다. 브라우저에 유효한 관리형 로그인 세션 쿠키가 없는 한 사용자는 로그인해야 합니다. `prompt`에 대해 여러 값을 공백 문자 구분 기호로 결합할 수 있습니다(예: `prompt=login consent`).

`resource`

선택 사항.

aud 클레임의 액세스 토큰에 바인딩하려는 리소스의 식별자입니다. 이 파라미터를 포함하면 Amazon Cognito는 값이 URL인지 확인하고 결과 액세스 토큰의 대상을 요청된 리소스로 설정합니다. URL 형식의 식별자 또는 원하는 URL을 사용하여 사용자 풀 리소스 서버를 요청할 수 있습니다. 이 파라미터의 값은 `https://`, `http://localhost`또는 `myapp://`과 같은 사용자 지정 URL 체계로 시작해야 합니다.

리소스 바인딩은 RFC 8707에 정의되어 있습니다. 리소스 서버 및 리소스 바인딩에 대한 자세한 내용은 리소스 바인딩을 참조하세요.

예: 권한 부여 코드 부여

이는 권한 부여 코드 부여에 대한 요청의 예입니다.

다음 요청은 사용자가 `redirect_uri` 대상의 앱에 전달하는 권한 부여 코드를 검색하는 세션을 시작합니다. 이 세션은 사용자 속성 및 Amazon Cognito 셀프 서비스 API 작업에 대한 액세스 범위를 요청합니다.

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&
client_id=1example23456789&
redirect_uri=https://www.example.com&
state=abcdefg&
scope=openid+profile+aws.cognito.signin.user.admin
```

Amazon Cognito 인증 서버는 권한 부여 코드 및 상태를 통해 앱으로 다시 리디렉션합니다. 권한 부여 코드는 5분 동안 유효합니다.

```
HTTP/1.1 302 Found
Location: https://www.example.com?code=a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111&state=abcdefg
```

예: PKCE를 통한 권한 부여 코드 부여

이 예제 흐름은 PKCE를 사용하여 권한 부여 코드를 부여합니다.

이 요청은 `code_challenge` 파라미터를 추가합니다. 토큰에 대한 코드 교환을 완료하려면 `/oauth2/token` 엔드포인트에 대한 요청에 `code_verifier` 파라미터를 포함해야 합니다.

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&
client_id=1example23456789&
redirect_uri=https://www.example.com&
state=abcdefg&
scope=aws.cognito.signin.user.admin&
code_challenge_method=S256&
code_challenge=a1b2c3d4...
```

권한 부여 서버는 권한 부여 코드 및 상태를 통해 애플리케이션으로 다시 리디렉션합니다. 애플리케이션은 권한 부여 코드를 처리하고 토큰으로 교환합니다.

```
HTTP/1.1 302 Found
Location: https://www.example.com?code=a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111&state=abcdefg
```

예: **prompt=login**를 사용하여 재인증 필요

다음 요청은 기존 세션이 있더라도 사용자가 다시 인증해야 하는 prompt=login 파라미터를 추가합니다.

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&
client_id=1example23456789&
redirect_uri=https://www.example.com&
state=abcdefg&
scope=openid+profile+aws.cognito.signin.user.admin&
prompt=login
```

권한 부여 서버는 로그인 엔드포인트로 리디렉션되므로 재인증이 필요합니다.

```
HTTP/1.1 302 Found Location: https://mydomain.auth.us-east-1.amazoncognito.com/
login?response_type=code&client_id=1example23456789&redirect_uri=https://
www.example.com&state=abcdefg&scope=openid+profile
+aws.cognito.signin.user.admin&prompt=login
```

예: **prompt=none**를 사용한 자동 인증

다음 요청은 사용자에게 유효한 세션이 있는지 자동으로 확인하는 prompt=none 파라미터를 추가합니다.

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&
client_id=1example23456789&
redirect_uri=https://www.example.com&
state=abcdefg&
scope=openid+profile+aws.cognito.signin.user.admin&
prompt=none
```

유효한 세션이 없는 경우 권한 부여 서버는 리디렉션 URI에 오류를 반환합니다.

```
HTTP/1.1 302 Found Location: https://www.example.com?error=login_required&state=abcdefg
```

유효한 세션이 있으면 권한 부여 서버가 권한 부여 코드를 반환합니다.

```
HTTP/1.1 302 Found Location: https://www.example.com?
code=AUTHORIZATION_CODE&state=abcdefg
```

예: 리소스 바인딩을 사용한 권한 부여 코드 부여

다음 요청은 액세스 토큰을 특정 리소스 서버에 바인딩하는 `resource` 파라미터를 추가합니다. 결과 액세스 토큰은 대상 API가 인증된 사용자 요청의 대상인지 확인하기 위한 조건을 생성합니다.

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&
client_id=1example23456789&
redirect_uri=https://www.example.com&
state=abcdefg&
scope=solar-system-data-api.example.com/asteroids.add&
resource=https://solar-system-data-api.example.com
```

인증 서버는 `https://solar-system-data-api.example.com`의 `aud` 클레임이 포함된 액세스 토큰을 생성하는 인증 코드를 반환합니다.

```
HTTP/1.1 302 Found Location: https://www.example.com?
code=AUTHORIZATION_CODE&state=abcdefg
```

예: **openid** 범위가 없는 토큰(암시적) 권한 부여

이 예제 흐름은 암묵적 허가를 생성하고 JWT를 사용자 세션에 직접 반환합니다.

이 요청은 권한 부여 서버의 암시적 권한 부여에 대한 것입니다. 사용자 프로필 셀프 서비스 작업을 승인하는 액세스 토큰의 범위를 요청합니다.

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=token&
client_id=1example23456789&
redirect_uri=https://www.example.com&
state=abcdefg&
scope=aws.cognito.signin.user.admin
```

인증 서버는 액세스 토큰만을 사용하여 애플리케이션으로 다시 리디렉션합니다. `openid` 범위가 요청되지 않았기 때문에 Amazon Cognito에서 ID 토큰을 반환하지 않습니다. 또한 Amazon Cognito는 이 흐름에서 새로 고침 토큰을 반환하지 않습니다.

```
HTTP/1.1 302 Found
Location: https://example.com/
callback#access_token=eyJra456defEXAMPLE&token_type=bearer&expires_in=3600&state=STATE
```

예: **openid** 범위가 있는 토큰(암시적) 권한 부여

이 예제 흐름은 암시적 권한 부여를 생성하고 사용자 브라우저에 토큰을 반환합니다.

이 요청은 권한 부여 서버의 암시적 권한 부여에 대한 것입니다. 이는 사용자 속성과 셀프 서비스 작업에 대한 액세스를 승인하는 액세스 토큰의 범위를 요청합니다.

```
GET
https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=token&
client_id=1example23456789&
redirect_uri=https://www.example.com&
state=abcdefg&
scope=aws.cognito.signin.user.admin+openid+profile
```

권한 부여 서버는 액세스 토큰 및 ID 토큰을 통해 애플리케이션으로 다시 리디렉션합니다(openid 범위가 포함되어 있기 때문).

```
HTTP/1.1 302 Found
Location: https://
www.example.com#id_token=eyJra67890EXAMPLE&access_token=eyJra12345EXAMPLE&token_type=bearer&exp
```

부정 응답 예제

Amazon Cognito가 요청을 거부할 수 있습니다. 부정 요청에는 HTTP 오류 코드와 요청 파라미터를 수정하는 데 사용할 수 있는 설명이 함께 제공됩니다. 다음은 부정 응답의 예입니다.

- `client_id` 및 `redirect_uri`는 유효하지만 요청 파라미터의 형식이 올바르게 지정되지 않은 경우 인증 서버가 오류를 클라이언트의 `redirect_uri`로 리디렉션하고 URL 파라미터에 오류 메시지를 추가합니다. 다음은 잘못된 형식의 예입니다.
  - 요청에 `response_type` 파라미터가 포함되지 않습니다.
  - 권한 부여 요청은 `code_challenge` 파라미터를 제공했지만 `code_challenge_method` 파라미터는 제공하지 않았습니다.

  - code_challenge_method 파라미터의 값이 S256이 아닙니다.

  다음은 잘못된 형식의 예시 요청에 대한 응답입니다.

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?error=invalid_request
```

- 클라이언트가 response_type에서 code 또는 token을 요청했는데 이러한 요청에 대한 권한이 없는 경우 Amazon Cognito 권한 부여 서버에서 다음과 같이 unauthorized_client를 클라이언트의 redirect_uri에 반환합니다.

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?error=unauthorized_client
```

- 클라이언트가 알 수 없거나, 형식이 잘못되었거나, 유효하지 않은 범위를 요청한 경우 Amazon Cognito 권한 부여 서버에서 다음과 같이 invalid_scope를 클라이언트의 redirect_uri에 반환합니다.

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?error=invalid_scope
```

- 서버에 예상치 못한 오류가 있는 경우 인증 서버는 클라이언트의 redirect_uri에 server_error를 반환합니다. HTTP 500 오류는 클라이언트로 전송되지 않기 때문에 사용자의 브라우저에 오류가 표시되지 않습니다. 권한 부여 서버가 다음 오류를 반환합니다.

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?error=server_error
```

- Amazon Cognito가 서드 파티 IdP로의 페더레이션을 통해 인증할 때 Amazon Cognito에서 다음과 같은 연결 문제가 발생할 수 있습니다.
  - IdP에게 토큰을 요청하는 동안 연결 제한 시간이 발생하면 인증 서버가 다음과 같이 오류를 클라이언트의 redirect_uri로 리디렉션합니다.

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?
error=invalid_request&error_description=Timeout+occurred+in+calling+IdP+token
+endpoint
```

  - ID 토큰 검증을 위해 jwks_uri 엔드포인트를 호출하는 동안 연결 제한 시간이 발생하면 인증 서버가 다음과 같이 오류를 클라이언트의 redirect_uri로 리디렉션합니다.

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?
error=invalid_request&error_description=error_description=Timeout+in+calling+jwks
+uri
```

- 타사 IdP에 페더레이션하여 인증할 때 공급자는 오류 응답을 반환할 수 있습니다. 이는 구성 오류 또는 다음과 같은 기타 이유 때문일 수 있습니다.
  - 다른 공급자로부터 오류 응답이 수신되면 인증 서버가 다음과 같이 오류를 클라이언트의 `redirect_uri`로 리디렉션합니다.

    ```
    HTTP 1.1 302 Found Location: https://client_redirect_uri?
    error=invalid_request&error_description=[IdP name]+Error+-+[status code]+error
     getting token
    ```

  - Google로부터 오류 응답이 수신되면 인증 서버가 다음과 같이 오류를 클라이언트의 `redirect_uri`로 리디렉션합니다.

    ```
    HTTP 1.1 302 Found Location: https://client_redirect_uri?
    error=invalid_request&error_description=Google+Error+-+[status code]+[Google-
    provided error code]
    ```

- Amazon Cognito가 외부 IdP에 연결할 때 통신 예외가 발생하면 인증 서버는 다음 메시지 중 하나를 사용하여 오류와 `redirect_uri` 함께 클라이언트의 로 리디렉션합니다.
  - ```
    HTTP 1.1 302 Found Location: https://client_redirect_uri?
    error=invalid_request&error_description=Connection+reset
    ```
  - ```
    HTTP 1.1 302 Found Location: https://client_redirect_uri?
    error=invalid_request&error_description=Read+timed+out
    ```

예: 단계적 인증 요청

Amazon Cognito는 관리형 로그인을 통한 단계적 인증을 지원합니다. 호스팅 UI에서는 지원되지 않습니다. 애플리케이션이 ACR 및 AMR을 사용하는 경우 관리형 로그인을 사용합니다.

ACR 수준 및 AMR 값에 대한 자세한 내용은 섹션을 참조하세요the section called “ACR 및 AMR 클레임을 사용한 인증 수준”.

`/oauth2/authorize` 엔드포인트는 단계적 인증을 위해 다음 URL 파라미터를 허용합니다.

`acr_values`

가장 높은 것부터 가장 낮은 것까지 대상 ACR 인증 수준 URIs의 공백으로 구분된 URL 인코딩 목록입니다. Amazon Cognito는 목록을 왼쪽에서 오른쪽으로 평가하고, 사용자가 충족할 수 있는 첫 번째 유효 수준을 적용하고, 인식하지 못하는 값을 무시합니다. 이 파라미터는

TARGET_ACR_VALUES API 파라미터와 동일하게 동작합니다. Essentials 또는 Plus 기능 플랜이 필요합니다.

`max_age`

사용자가 마지막으로 인증된 이후 최대 초 수를 지정하는 음수가 아닌 정수입니다. 현재 시간에서를 뺀 값이 보다 큰 경우 `max_age`사용자는 한 단계만 올라가`auth_time`는 대신 처음부터 다시 인증해야 합니다. 이 동작은 MAX_AGE API 파라미터와 동일합니다.

`acr_values`와 결합하여 민감한 작업에 대한 최신 인증과 특정 인증 수준을 `max_age` 요구합니다.

다음 예제 `/oauth2/authorize` 요청은 Amazon Cognito에 먼저 레벨 4를 시도하도록 요청합니다. 예를 들어 TOTP가 구성되어 있지 않아 사용자가 레벨 4를 충족할 수 없는 경우 Amazon Cognito는 사용자에게 SMS 또는 이메일 일회용 암호가 구성되어 있다고 가정하여 레벨 3으로 돌아갑니다. 또한이 요청을 수행하려면 사용자가 지난 3,600초 이내에 인증해야 합니다. 예제 클라이언트 ID *`1example23456789`* 및 기타 기울임꼴 값은 자리 표시자입니다. 이들을 사용자의 고유한 값으로 교체합니다.

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&client_id=1example23456789&redirect_uri=https://YOUR_APP/
redirect_uri&state=STATE&scope=openid+profile
+aws.cognito.signin.user.admin&acr_values=urn%3Acognito%3Aloa%3A4+urn%3Acognito%3Aloa
%3A3&max_age=3600
```

기능 부족 계획의 동작은 API와 다릅니다. 관리형 로그인은 엔드포인트를 자동으로 무시`acr_values`하고 오류 없이 일반 인증을 진행합니다. API는를 반환합니다`FeatureUnavailableInTierException`. 자세한 내용은 기능 계획 요구 사항 섹션을 참조하세요.

토큰 엔드포인트

`/oauth2/token`의 OAuth 2.0 토큰 엔드포인트는 권한 부여 코드 및 클라이언트 자격 증명 부여 흐름을 완료하려는 애플리케이션에 JSON 웹 토큰(JWTs)을 발급합니다. 이러한 토큰은 사용자 풀을 사용한 인증의 최종 결과입니다. 여기에는 사용자(ID 토큰), 사용자의 액세스 수준(액세스 토큰) 및 로그인 세션을 유지할 수 있는 사용자의 권한(토큰 새로 고침)에 대한 정보가 포함되어 있습니다. OIDC(OpenID Connect) 신뢰 당사자 라이브러리는 이 엔드포인트의 요청 및 응답 페이로드를 처리합니다. 토큰은 검증 가능한 인증 증명, 프로필 정보 및 백엔드 시스템에 대한 액세스 메커니즘을 제공합니다.

토큰 엔드포인트는 `Access-Control-Allow-Origin: *` 응답 헤더를 반환합니다. 브라우저 기반 애플리케이션에서 토큰 엔드포인트 크로스 오리진을 호출할 수 있습니다. 예를 들어 퍼블릭 클라이언트에서 코드 교환용 증명 키(PKCE)를 사용하여 권한 부여 코드 부여를 완료할 수 있습니다. Amazon Cognito는이 엔드포인트에서 사용자 지정 CORS(Cross-Origin Resource Sharing) 오리진 정책을 지원하지 않습니다. 자세한 내용은의 CORS 정책 섹션을 참조하세요사용자 풀 관리형 로그인.

사용자 풀 OAuth 2.0 권한 부여 서버는 토큰 엔드포인트에서 다음과 같은 유형의 세션에 JWT(JSON 웹 토큰)을 발급합니다.

1. 권한 부여 코드 부여 요청을 완료한 사용자. 코드를 성공적으로 사용하면 ID, 액세스 및 새로 고침 토큰이 반환됩니다.
2. 클라이언트 보안 인증 권한 부여를 완료한 M2M(Machine-to-Machine) 세션. 클라이언트 보안 암호로 인증에 성공하면 액세스 토큰이 반환됩니다.
3. 이전에 로그인하고 새로 고침 토큰을 받은 사용자. 새로 고침 토큰 인증은 새 ID와 액세스 토큰을 반환합니다.

> **Note**
>
> 관리형 로그인 또는 페더레이션을 통해 권한 부여 코드 부여로 로그인하는 사용자는 항상 토큰 엔드포인트에서 토큰을 새로 고칠 수 있습니다. 사용자 풀에서 기억된 디바이스가 활성화되어 있지 않은 경우 API 작업 `InitiateAuth` 및 `AdminInitiateAuth`로 로그인한 사용자는 토큰 엔드포인트를 통해 토큰을 새로 고칠 수 있습니다. 기억된 디바이스가 활성 상태인 경우 앱 클라이언트에 대한 관련 API 또는 SDK 토큰 새로 고침 작업을 사용하여 토큰을 새로 고칩니다.

도메인을 사용자 풀에 추가하면 토큰 엔드포인트를 공개적으로 사용할 수 있게 됩니다. 토큰 엔드포인트는 HTTP POST 요청을 수락합니다. 애플리케이션 보안을 위해 인증 코드 로그인 이벤트와 함께 PKCE를 사용하세요. PKCE는 인증 코드를 전달하는 사용자가 인증을 받은 사용자와 동일한지 확인합니다. PKCE에 대한 자세한 내용은 IETF RFC 7636을 참조하세요.

사용자 풀 앱 클라이언트와 해당 권한 부여 유형, 클라이언트 보안 암호, 허용된 범위 및 클라이언트 ID에 대한 자세한 내용은 앱 클라이언트를 사용한 애플리케이션별 설정에서 확인할 수 있습니다. M2M 권한 부여, 클라이언트 자격 증명 부여, 액세스 토큰 범위를 사용한 권한 부여에 대한 자세한 내용은 범위, M2M 및 리소스 서버에서 확인할 수 있습니다.

액세스 토큰에서 사용자 관련 정보를 검색하려면 토큰을 userInfo 엔드포인트 또는 GetUser API 요청으로 전달합니다. 액세스 토큰에는 이러한 요청에 대한 적절한 범위가 포함되어야 합니다.

Amazon Cognito와 OIDC 사양의 차이점

Amazon Cognito는 새로 고침 토큰 발급을 제어하기 위한 OpenID Connect `offline_access` 범위 모델을 구현하지 않습니다. 대신 토큰 엔드포인트는 `grant_type`가 일 때마다 새로 고침 토큰을 반환하고 `authorization_code`를 요청하는 대신 앱 클라이언트의 새로 고침 토큰 유효성 설정을 통해 새로 고침 토큰 수명을 제어합니다`offline_access`. `offline_access` 범위를 요청해도 아무런 효과가 없으며, 범위를 생략해도 새로 고침 토큰이 억제되지 않습니다.

POST 요청을 토큰 엔드포인트로 형식 지정

`/oauth2/token` 엔드포인트는 `HTTPS POST`만 지원합니다. 이 엔드포인트는 사용자 대화형이 아닙니다. 애플리케이션에서 OpenID Connect(OIDC) 라이브러리를 사용하여 토큰 요청을 처리합니다.

토큰 엔드포인트는 `client_secret_basic` 및 `client_secret_post` 인증을 지원합니다. OIDC 사양에 대한 자세한 내용은 클라이언트 인증을 참조하세요. OpenID Connect 사양의 Token 엔드포인트에 대한 자세한 내용은 토큰 엔드포인트를 참조하세요.

헤더의 요청 파라미터

요청의 헤더에 있는 다음 파라미터를 토큰 엔드포인트에 전달할 수 있습니다.

`Authorization`

클라이언트에 보안 암호가 발급된 경우 클라이언트는 권한 부여 헤더에서 `client_secret_basic` HTTP 권한 부여로 해당 `client_id` 및 `client_secret`을 전달할 수 있습니다. 요청 본문에 `client_secret_post` 권한 부여로 `client_id` 및 `client_secret`을 포함할 수도 있습니다.

인증 헤더 문자열은 기본 `Base64Encode(client_id:client_secret)` 입니다.
다음 예제는 앱 클라이언트 `djc98u3jiedmi283eu928`에 대한 권한 부여 헤더로 `djc98u3jiedmi283eu928:abcdef01234567890` 문자열의 Base64 인코딩 버전을 사용하는 클라이언트 보안 암호 `abcdef01234567890`을 포함합니다.

```
Authorization: Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw
```

`Content-Type`

이 파라미터의 값을 `'application/x-www-form-urlencoded'`으로 설정합니다.

본문의 요청 파라미터

다음은 토큰 엔드포인트에 대한 요청 본문에서 `x-www-form-urlencoded` 형식으로 요청할 수 있는 파라미터입니다.

`grant_type`

필수 여부.

요청하려는 OIDC 권한 부여 유형입니다.

`authorization_code`, `refresh_token` 또는 `client_credentials`가 있습니다. 다음 조건에서 토큰 엔드포인트에서 사용자 지정 범위에 대한 액세스 토큰을 요청할 수 있습니다.

- 앱 클라이언트 구성에서 요청 범위를 활성화했습니다.
- 클라이언트 보안 암호로 앱 클라이언트를 구성했습니다.
- 클라이언트 자격 증명 부여에 대해 앱 클라이언트를 활성화했습니다. 다음 방법 중 하나로이 권한 부여를 활성화할 수 있습니다.
  - 호스팅 도메인 machine-to-machine) 인증의 경우 앱 클라이언트의 `client_credentials`에를 추가합니다`AllowedOAuthFlows`.
  - 도메인 없는 M2M 인증의 경우 앱 클라이언트의 `ALLOW_CLIENT_TOKEN_AUTH`에를 추가합니다`ExplicitAuthFlows`. 이 흐름에는 클라이언트 보안 암호가 필요하며 앱 클라이언트의 유일한 인증 흐름이어야 합니다.

이러한 옵션 중 어느 것도 활성화하지 않으면 토큰 엔드포인트가 무단 클래스 오류를 반환합니다.

> **Note**
>
> 토큰 엔드포인트는 `grant_type`이 `authorization_code`일 때만 새로 고침 토큰을 반환합니다.

`client_id`

선택 사항. *`Authorization`* 헤더에 앱 클라이언트 ID를 제공할 때는 필요하지 않습니다.

사용자 풀에서 앱 클라이언트의 ID입니다. 사용자를 인증한 것과 동일한 앱 클라이언트를 지정합니다.

클라이언트가 공개이고 보안 암호가 없거나 client_secret_post 권한 부여에서 client_secret를 사용하는 경우 이 파라미터를 제공해야 합니다.

client_secret

선택 사항. *Authorization* 헤더에 클라이언트 보안 암호를 제공하고 앱 클라이언트에 보안 암호가 없는 경우 필요하지 않습니다.

앱 클라이언트에 client_secret_post 권한이 있는 경우 앱 클라이언트 보안 암호입니다.

scope

선택 사항.

앱 클라이언트와 연결된 범위를 조합하여 사용할 수 있습니다. 요청에 리소스 서버에 정의된 범위가 포함된 경우 Amazon Cognito는 해당 범위가 요청된 앱 클라이언트에 허용되는지 여부를 확인합니다. 범위가 허용되지 않는 경우 Amazon Cognito는 나머지 범위에 대해 토큰을 발급하지 않고 전체 요청을 거부합니다. 허용 범위와 허용되지 않는 범위를 결합하는 요청은 동일한 이유로 실패합니다.

Amazon Cognito는 사용할 수 없는 범위를 삭제하고 나머지에 대한 토큰을 반환하는 대신 거부합니다. 거부된 요청은 응답 본문에서 오류HTTP 400와 함께를 반환합니다. 오류 값은 범위를 사용할 수 없는 이유에 따라 달라집니다. 범위가 존재하지만 앱 클라이언트에 대해 허용되지 않는 경우 Amazon Cognito는를 반환합니다invalid_grant. 범위가 존재하지 않는 리소스 서버 또는 사용자 지정 범위를 참조하면 Amazon Cognito는를 반환합니다invalid_scope. 이 요청 파라미터를 제공하지 않으면 권한 부여 서버는 앱 클라이언트 구성에서 활성화한 모든 권한 부여 범위가 포함된 액세스 토큰 scope 클레임을 반환합니다. 표준 범위, 리소스 서버의 사용자 지정 범위, aws.cognito.signin.user.admin 사용자 셀프 서비스 범위 등 요청된 앱 클라이언트에 허용되는 모든 범위를 요청할 수 있습니다.

redirect_uri

선택 사항. 클라이언트 자격 증명 부여에는 필요하지 않습니다.

/oauth2/authorize에서 authorization_code를 얻기 위해 사용했던 redirect_uri와 같아야 합니다.

grant_type가 authorization_code인 경우 이 파라미터를 제공해야 합니다.

refresh_token

선택 사항. 사용자에게 이미 새로 고침 토큰이 있고 새 ID 및 액세스 토큰이 필요한 경우에만 사용됩니다.

사용자 세션에 대한 새로운 액세스 및 ID 토큰을 생성하려면 `refresh_token` 값을 요청된 앱 클라이언트가 발행한 유효한 새로 고침 토큰으로 설정합니다.

새로 고침 토큰 교체가 활성 상태일 때 새 ID 및 액세스 토큰이 있는 새 새로 고침 토큰을 반환하고, 그렇지 않으면 ID 및 액세스 토큰만 반환합니다. 원래 액세스 토큰이 API 리소스에 바인딩된 경우 새 액세스 토큰은 요청된 API URL을 `aud` 클레임에 유지합니다.

`code`

선택 사항. 권한 부여 코드 부여에만 필요합니다.

권한 부여 코드 부여의 권한 부여 코드입니다. 권한 부여 요청에 `authorization_code`의 `grant_type`가 포함된 경우 이 파라미터를 제공해야 합니다.

`aws_client_metadata`

선택 사항.

M2M(머신 간) 권한 부여 흐름에서 사전 토큰 생성 Lambda 트리거에 전달하려는 정보입니다. 애플리케이션은 세션에 대한 컨텍스트 정보를 수집하여 이 파라미터에 전달할 수 있습니다. URL로 인코딩된 JSON 형식으로 `aws_client_metadata`를 전달하면 Amazon Cognito는 이를 트리거 Lambda 함수에 대한 입력 이벤트에 포함합니다. 사전 토큰 트리거 이벤트 버전 또는 글로벌 Lambda 트리거 버전은 버전 3 이상에 맞게 구성해야 합니다. Amazon Cognito는 권한 부여 코드 및 클라이언트 자격 증명 M2M 흐름에서 이 엔드포인트에 대한 요청을 수락하지만 사용자 풀은 `aws_client_metadata`를 클라이언트 자격 증명 요청의 사전 토큰 생성 트리거에만 전달합니다.

`code_verifier`

선택 사항. 초기 권한 부여 요청에서 `code_challenge_method` 및 `code_challenge` 파라미터를 제공한 경우에만 필요합니다.

애플리케이션이 PKCE를 통한 권한 부여 코드 부여 요청에서 `code_challenge`를 계산한 생성된 코드 확인자입니다.

인증 코드를 토큰으로 교환

다음 요청은 권한 부여 코드 부여로 인증한 후 ID, 액세스 및 새로 고침 토큰을 성공적으로 생성합니다. 요청은 `Authorization` 헤더에 클라이언트 보안 암호를 `client_secret_basic` 형식으로 전달합니다.

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token&
```

```
Content-Type='application/x-www-form-urlencoded'&
Authorization=Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw

grant_type=authorization_code&
client_id=1example23456789&
code=AUTHORIZATION_CODE&
redirect_uri=com.myclientapp://myclient/redirect
```

응답은 추가 메타데이터와 함께 사용자에게 새 ID, 액세스 및 새로 고침 토큰을 발급합니다.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "access_token": "eyJra1example",
    "id_token": "eyJra2example",
    "refresh_token": "eyJj3example",
    "token_type": "Bearer",
    "expires_in": 3600
}
```

기본 권한이 있는 클라이언트 자격 증명

M2M 애플리케이션의 다음 요청은 클라이언트 자격 증명 부여를 요청합니다. 클라이언트 자격 증명에는 클라이언트 보안 암호가 필요하므로 요청은 앱 클라이언트 ID 및 보안 암호에서 파생된 Authorization 헤더로 승인됩니다. 요청으로 인해 요청된 두 범위가 있는 액세스 토큰이 생성됩니다. 이 요청에는 IP 주소 정보를 제공하는 클라이언트 메타데이터와 이 권한 부여가 대리하는 사용자에게 발급된 토큰도 포함됩니다. Amazon Cognito는 클라이언트 메타데이터를 사전 토큰 생성 Lambda 트리거에 전달합니다.

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token >
Content-Type='application/x-www-form-urlencoded'&
Authorization=Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw

grant_type=client_credentials&
client_id=1example23456789&
scope=resourceServerIdentifier1%2Fscope1%20resourceServerIdentifier2%2Fscope2&
&aws_client_metadata=%7B%22onBehalfOfToken%22%3A%22eyJra789ghiEXAMPLE%22,
%20%22ClientIpAddress%22%3A%22192.0.2.252%22%7D
```

Amazon Cognito는 다음 입력 이벤트를 사전 토큰 생성 Lambda 트리거에 전달합니다.

```
{
    version: '3',
    triggerSource: 'TokenGeneration_ClientCredentials',
    region: 'us-east-1',
    userPoolId: 'us-east-1_EXAMPLE',
    userName: 'ClientCredentials',
    callerContext: {
        awsSdkVersion: 'aws-sdk-unknown-unknown',
        clientId: '1example23456789'
    },
    request: {
        userAttributes: {},
        groupConfiguration: null,
        scopes: [
           'resourceServerIdentifier1/scope1',
           'resourceServerIdentifier2/scope2'
        ],
        clientMetadata: {
            'onBehalfOfToken': 'eyJra789ghiEXAMPLE',
            'ClientIpAddress': '192.0.2.252'
        }
    },
    response: { claimsAndScopeOverrideDetails: null }
}
```

응답은 액세스 토큰을 반환합니다. 클라이언트 자격 증명 부여는 M2M(머신 간) 인증을 위한 것이며 액세스 토큰만 반환합니다.

```
HTTP/1.1 200 OK
Content-Type: application/json
{
    "access_token": "eyJra1example",
    "token_type": "Bearer",
    "expires_in": 3600
}
```

POST 본문 권한이 있는 클라이언트 자격 증명

다음 클라이언트 자격 증명 권한 부여 요청은 요청 본문에 `client_secret` 파라미터를 포함하고 `Authorization` 헤더를 포함하지 않습니다. 이 요청은 `client_secret_post` 권한 부여 구문을 사용합니다. 요청으로 인해 요청된 범위가 있는 액세스 토큰이 생성됩니다. 이 요청에는 IP 주소 정보를

제공하는 클라이언트 메타데이터와 이 권한 부여가 대리하는 사용자에게 발급된 토큰도 포함됩니다. Amazon Cognito는 클라이언트 메타데이터를 사전 토큰 생성 Lambda 트리거에 전달합니다.

```
POST /oauth2/token HTTP/1.1
Content-Type: application/x-www-form-urlencoded
X-Amz-Target: AWSCognitoIdentityProviderService.Client credentials request
User-Agent: USER_AGENT
Accept: /
Accept-Encoding: gzip, deflate, br
Content-Length: 177
Referer: http://auth.example.com/oauth2/token
Host: auth.example.com
Connection: keep-alive

grant_type=client_credentials&
client_id=1example23456789&
scope=my_resource_server_identifier%2Fmy_custom_scope&
client_secret=9example87654321&
aws_client_metadata=%7B%22onBehalfOfToken%22%3A%22eyJra789ghiEXAMPLE%22,
%20%22ClientIpAddress%22%3A%22192.0.2.252%22%7D
```

Amazon Cognito는 다음 입력 이벤트를 사전 토큰 생성 Lambda 트리거에 전달합니다.

```
{
    version: '3',
    triggerSource: 'TokenGeneration_ClientCredentials',
    region: 'us-east-1',
    userPoolId: 'us-east-1_EXAMPLE',
    userName: 'ClientCredentials',
    callerContext: {
        awsSdkVersion: 'aws-sdk-unknown-unknown',
        clientId: '1example23456789'
    },
    request: {
        userAttributes: {},
        groupConfiguration: null,
        scopes: [
           'resourceServerIdentifier1/my_custom_scope'
        ],
        clientMetadata: {
            'onBehalfOfToken': 'eyJra789ghiEXAMPLE',
            'ClientIpAddress': '192.0.2.252'
        }
```

```
    },
    response: { claimsAndScopeOverrideDetails: null }
}
```

응답은 액세스 토큰을 반환합니다. 클라이언트 자격 증명 부여는 M2M(머신 간) 인증을 위한 것이며 액세스 토큰만 반환합니다.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Tue, 05 Dec 2023 16:11:11 GMT
x-amz-cognito-request-id: 829f4fe2-a1ee-476e-b834-5cd85c03373b

{
    "access_token": "eyJra12345EXAMPLE",
    "expires_in": 3600,
    "token_type": "Bearer"
}
```

PKCE를 통한 인증 코드 권한 부여

다음 예제 요청은 PKCE를 사용한 권한 부여 코드 부여 요청에 code_challenge_method 및 code_challenge 파라미터를 포함한 권한 부여 요청을 완료합니다.

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token
Content-Type='application/x-www-form-urlencoded'&
Authorization=Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw

grant_type=authorization_code&
client_id=1example23456789&
code=AUTHORIZATION_CODE&
code_verifier=CODE_VERIFIER&
redirect_uri=com.myclientapp://myclient/redirect
```

응답은 애플리케이션의 성공적인 PKCE 확인에서 ID, 액세스 및 새로 고침 토큰을 반환합니다.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "access_token": "eyJra1example",
    "id_token": "eyJra2example",
```

```
    "refresh_token": "eyJj3example",
    "token_type": "Bearer",
    "expires_in": 3600
}
```

새로 고침 토큰 교체 없이 토큰 새로 고침

다음 예제 요청은 새로 고침 토큰 교체가 비활성화된 앱 클라이언트에 새로 고침 토큰을 제공합니다. 앱 클라이언트에는 클라이언트 보안 암호가 있으므로 요청은 Authorization 헤더를 제공합니다.

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token >
Content-Type='application/x-www-form-urlencoded'&
Authorization=Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw

grant_type=refresh_token&
client_id=1example23456789&
refresh_token=eyJj3example
```

응답은 새 ID와 액세스 토큰을 반환합니다.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "access_token": "eyJra1example",
    "id_token": "eyJra2example",
    "token_type": "Bearer",
    "expires_in": 3600
}
```

새로 고침 토큰 교체를 통한 토큰 새로 고침

다음 예제 요청은 새로 고침 토큰 교체가 활성화된 앱 클라이언트에 새로 고침 토큰을 제공합니다. 앱 클라이언트에는 클라이언트 보안 암호가 있으므로 요청은 Authorization 헤더를 제공합니다.

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token >
Content-Type='application/x-www-form-urlencoded'&
Authorization=Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw

grant_type=refresh_token&
client_id=1example23456789&
```

```
refresh_token=eyJj3example
```

응답은 새로운 ID, 액세스 및 새로 고침 토큰을 반환합니다.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "access_token": "eyJra1example",
    "id_token": "eyJra2example",
    "refresh_token": "eyJj4example",
    "token_type": "Bearer",
    "expires_in": 3600
}
```

부정 응답 예제

형식이 잘못된 요청은 토큰 엔드포인트에서 오류를 생성합니다. 다음은 토큰 요청에서 오류가 발생할 때 응답 본문의 일반적인 맵입니다.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error":"invalid_request|invalid_client|invalid_grant|unauthorized_client|
unsupported_grant_type"
}
```

`invalid_request`

요청에 필수 파라미터가 누락되거나, 지원되지 않는 파라미터 값(`unsupported_grant_type` 아님)이 포함되거나, 잘못되었습니다. 예를 들어, `grant_type`이 `refresh_token`이지만 `refresh_token`이 포함되어 있지 않습니다.

`invalid_client`

클라이언트 인증에 실패했습니다. 예를 들어, 클라이언트가 권한 부여 헤더에 `client_id` 및 `client_secret`을 포함하지만 `client_id` 및 `client_secret`이 있는 클라이언트가 없는 경우입니다.

`invalid_grant`

새로 고침 토큰이 취소되었습니다.

권한 부여 코드를 이미 사용했거나 해당 코드가 존재하지 않습니다.

앱 클라이언트에는 요청한 범위의 일부 속성에 대한 읽기 액세스 권한이 없습니다. 예를 들어, 앱이 email 범위를 요청하면 앱 클라이언트는 email 속성을 읽을 수 있지만, email_verified 속성은 읽을 수 없습니다.

unauthorized_client

클라이언트가 코드 부여 흐름이나 토큰 새로 고침에 대해 허용되지 않습니다.

권한 부여 요청의 값과 일치하지 redirect_uri 않는이 있는 토큰 요청은 error_description의와 unauthorized_client 함께를 반환합니다invalid_redirect.

unsupported_grant_type

grant_type이 authorization_code 또는 refresh_token 또는 client_credentials 이외의 것인 경우 반환됩니다.

사용자 속성 엔드포인트

OIDC가 사용자 속성을 포함하는 ID 토큰을 발급하는 경우 OAuth 2.0은 /oauth2/userInfo 엔드포인트를 구현합니다. 인증된 사용자 또는 클라이언트는 scopes 클레임이 포함된 액세스 토큰을 받습니다. 이 클레임은 권한 부여 서버가 반환해야 하는 속성을 결정합니다. 애플리케이션이 userInfo 엔드포인트에 액세스 토큰을 제시하면 권한 부여 서버는 액세스 토큰 범위에 의해 설정된 경계 내에 있는 사용자 속성이 포함된 응답 본문을 반환합니다. 애플리케이션은 최소한 openid 범위 클레임이 있는 유효한 액세스 토큰을 보유하는 한 userInfo 엔드포인트에서 사용자에 대한 정보를 검색할 수 있습니다.

userInfo 엔드포인트는 OpenID Connect(OIDC) userInfo 엔드포인트입니다. 이 엔드포인트는 서비스 공급자가 토큰 엔드포인트에서 발급한 액세스 토큰을 제시하면 사용자 속성을 이용해 응답합니다. 사용자 액세스 토큰의 범위는 userInfo 엔드포인트가 응답에서 반환하는 사용자 특성을 정의합니다. openid 범위는 액세스 토큰 클레임 중 하나여야 합니다.

Amazon Cognito는 InitiateAuth 같은 사용자 풀 API 요청에 대한 응답으로 액세스 토큰을 발급합니다. 범위가 없기 때문에 userInfo 엔드포인트는 이러한 액세스 토큰을 수락하지 않습니다. 대신 토큰 엔드포인트에서 액세스 토큰을 제시해야 합니다.

OAuth 2.0 타사 ID 제공업체(IdP)도 userInfo 엔드포인트를 호스팅합니다. 사용자가 이 IdP로 인증하면 Amazon Cognito는 권한 부여 코드를 자동으로 IdP token 엔드포인트와 교환합니다. 사용자 풀은 IdP 액세스 토큰을 전달하여 IdP userInfo 엔드포인트에서 사용자 정보 검색을 승인합니다.

사용자 액세스 토큰의 범위는 인증 요청의 scopes 요청 파라미터 또는 사전 토큰 생성 Lambda 트리거가 추가하는 범위에 따라 결정됩니다. 액세스 토큰을 디코딩하고 scope 클레임을 검사하여 포함된 액세스 제어 범위를 확인할 수 있습니다. 다음은 userInfo 엔드포인트에서 반환되는 데이터에 영향을 미치는 몇 가지 범위 조합입니다. 예약된 Amazon Cognito 범위 aws.cognito.signin.user.admin은 이 엔드포인트에서 반환되는 데이터에 영향을 주지 않습니다.

액세스 토큰의 범위 예제 및 **userInfo** 응답에 미치는 영향

openid

앱 클라이언트가 읽을 수 있는 모든 사용자 속성이 포함된 응답을 반환합니다.

openid profile

사용자 속성 name, family_name, given_name, middle_name, nickname, preferred_username, profile, picture, website, gender, birthdate, zoneinfo, locale, updated_at를 반환합니다. 사용자 지정 속성도 반환합니다. 각 속성에 대한 읽기 액세스 권한이 없는 앱 클라이언트에서 이 범위에 대한 응답은 앱 클라이언트가 읽기 액세스 권한이 있는 사양 내의 모든 속성입니다.

openid email

기본 프로필 정보와 email 및 email_verified 속성을 반환합니다.

openid phone

기본 프로필 정보와 phone_number 및 phone_number_verified 속성을 반환합니다.

GET /oauth2/userInfo

애플리케이션은 브라우저를 통하지 않고 이 엔드포인트에 직접 요청을 생성합니다.

자세한 내용은 OpenID Connect(OIDC) 사양의 UserInfoEndpoint(엔드포인트)를 참조하세요.

주제

- 헤더의 요청 파라미터
- 예 – 요청
- 예 – 긍정 응답
- 예 부정 응답

헤더의 요청 파라미터

Authorization: Bearer *<access_token>*

권한 부여 헤더 필드에 액세스 토큰을 전달합니다.

필수 사항입니다.

예 – 요청

```
GET /oauth2/userInfo HTTP/1.1
Content-Type: application/x-amz-json-1.1
Authorization: Bearer eyJra12345EXAMPLE
User-Agent: [User agent]
Accept: */*
Host: auth.example.com
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

예 – 긍정 응답

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: [Integer]
Date: [Timestamp]
x-amz-cognito-request-id: [UUID]
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
X-Frame-Options: DENY
Server: Server
Connection: keep-alive
{
    "sub": "[UUID]",
    "email_verified": "true",
    "custom:mycustom1": "CustomValue",
    "phone_number_verified": "true",
    "phone_number": "+12065551212",
    "email": "bob@example.com",
```

```
    "username": "bob"
}
```

OIDC 신청 목록은 표준 신청을 참조하세요. 현재 Amazon Cognito는 `email_verified` 및 `phone_number_verified`의 값을 문자열로 반환합니다.

예 부정 응답

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 누락되었거나 지원되지 않는 파라미터 값이 포함되어 있거나 기타 잘못된 형식의 요청입니다.

예 - 잘못된 토큰

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: error="invalid_token",
error_description="Access token is expired, disabled, or deleted, or the user has
 globally signed out."
```

`invalid_token`

액세스 토큰이 만료되었거나, 취소되었거나, 형식이 잘못되었거나, 유효하지 않습니다.

토큰 취소 엔드포인트

세션에 새로 고침 토큰을 보유한 사용자는 브라우저 쿠키와 유사한 기능을 갖습니다. 새로 고침 토큰이 유효한 한 기존 세션을 갱신할 수 있습니다. ID 또는 액세스 토큰이 만료된 후 사용자에게 로그인하라는 메시지를 표시하는 대신 애플리케이션은 새로 고침 토큰을 사용하여 새롭고 유효한 토큰을 가져올 수 있습니다. 그러나 외부에서 사용자의 세션을 종료해야 하는지 또는 사용자가 현재 세션을 잊어버리기로 선택할 수 있습니다. 이 시점에서 해당 새로 고침 토큰을 취소하여 더 이상 세션을 유지할 수 없게 할 수 있습니다.

/oauth2/revoke 엔드포인트는 사용자가 제공한 새로 고침 토큰으로 Amazon Cognito가 처음에 발급한 사용자의 액세스 토큰을 취소합니다. 또한 이 엔드포인트는 동일한 새로 고침 토큰에서 새로 고침 토큰 자체와 이후의 모든 액세스 및 자격 증명 토큰을 취소합니다. 엔드포인트가 토큰을 취소한 후에는 취소된 액세스 토큰을 사용하여 Amazon Cognito 토큰이 인증하는 API에 액세스할 수 없습니다.

POST /oauth2/revoke

/oauth2/revoke 엔드포인트는 HTTPS POST만 지원합니다. 사용자 풀 클라이언트는 시스템 브라우저를 통해서가 아닌 직접 이 엔드포인트를 요청합니다.

헤더의 요청 파라미터

Authorization

앱 클라이언트에 클라이언트 보안 암호가 있는 경우 애플리케이션은 기본 HTTP 권한 부여를 통해 권한 부여 헤더에서 해당 client_id 및 client_secret를 전달해야 합니다. 보안 암호는 기본 Base64Encode(client_id:client_secret)입니다.

Content-Type

항상 'application/x-www-form-urlencoded'여야 합니다.

본문의 요청 파라미터

token

(필수) 클라이언트가 취소하려고 하는 새로 고침 토큰입니다. 이 요청은 Amazon Cognito가 이 새로 고침 토큰에서 발급한 모든 액세스 토큰도 취소합니다.

필수 사항입니다.

client_id

(선택 사항) 철회할 토큰의 앱 클라이언트 ID입니다.

클라이언트가 퍼블릭이고 보안 암호가 없는 경우 필수 사항입니다.

철회 요청 예제

이 취소 요청은 클라이언트 보안 암호가 없는 앱 클라이언트에 대한 새로 고침 토큰을 취소합니다. 요청 본문에 있는 client_id 파라미터에 주의하세요.

```
POST /oauth2/revoke HTTP/1.1
Host: mydomain.auth.us-east-1.amazoncognito.com
Accept: application/json
Content-Type: application/x-www-form-urlencoded
token=2YotnFZFEjr1zCsicMWpAA&
client_id=1example23456789
```

이 취소 요청은 클라이언트 보안 암호가 있는 앱 클라이언트의 새로 고침 토큰을 취소합니다. 인코딩된 클라이언트 ID와 클라이언트 보안 암호가 포함되어 있지만 요청 본문에 `client_id`가 없는 `Authorization` 헤더를 확인합니다.

```
POST /oauth2/revoke HTTP/1.1
Host: mydomain.auth.us-east-1.amazoncognito.com
Accept: application/json
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
token=2YotnFZFEjr1zCsicMWpAA
```

오류 응답 철회

성공적인 응답에 빈 본문이 포함됩니다. 오류 응답은 `error` 필드를 포함하고 경우에 따라 `error_description` 필드도 포함하는 JSON 객체입니다.

엔드포인트 오류

- 요청에 토큰이 없는 경우 또는 앱 클라이언트에 대해 기능이 비활성화된 경우 HTTP 400 및 오류 `invalid_request`가 발생합니다.
- Amazon Cognito가 취소 요청에서 보낸 토큰이 새로 고침 토큰이 아닌 경우 HTTP 400 및 오류 `unsupported_token_type`이 발생합니다.
- 클라이언트 자격 증명이 유효하지 않은 경우 HTTP 401 및 오류 `invalid_client`가 발생합니다.
- 토큰이 취소된 경우 또는 클라이언트가 유효하지 않은 토큰을 제출한 경우 HTTP 200 OK가 발생합니다.

IdP SAML 어설션 엔드포인트

`/saml2/idpresponse`는 SAML 어설션을 수신합니다. SP 시작(service-provider-initiated) 로그인에서 애플리케이션은 이 엔드포인트와 직접 상호 작용하지 않으며, SAML 2.0 ID 제공업체(idP)는 SAML

응답과 함께 사용자를 여기로 리디렉션합니다. SP 시작 로그인의 경우, `saml2/idpresponse` 경로를 ACS(어설션 소비자 서비스) URL로 사용하여 IdP를 구성합니다. 세션 시작에 대한 자세한 내용은 Amazon Cognito 사용자 풀에서 SAML 세션 시작 섹션을 참조하세요.

IdP 시작 로그인에서 SAML 2.0 공급자로 사용자에 로그인한 후 애플리케이션의 이 엔드포인트에 요청을 호출합니다. 사용자가 브라우저에서 IdP로 로그인하면 애플리케이션이 SAML 어설션을 수집하여 이 엔드포인트에 제출합니다. HTTPS를 통해 `HTTP POST` 요청 본문에 SAML 어설션을 제출해야 합니다. `POST` 요청 본문은 `SAMLResponse` 파라미터와 `Relaystate` 파라미터여야 합니다. 자세한 내용은 IdP 시작 SAML 로그인 구현 섹션을 참조하세요.

`saml2/idpresponse` 엔드포인트는 최대 100,000자의 SAML 어설션을 수락할 수 있습니다.

POST **`/saml2/idpresponse`**

IdP 시작 로그인에서 `/saml2/idpresponse` 엔드포인트를 사용하려면 사용자 풀에 사용자 세션에 대한 정보를 제공하는 파라미터로 POST 요청을 생성합니다.

- 로그인하려는 앱 클라이언트입니다.
- 리디렉션할 콜백 URL입니다.
- 사용자의 액세스 토큰에서 요청하려는 OAuth 2.0 범위입니다.
- 로그인 요청을 시작한 IdP입니다.

IdP 시작 요청 본문 파라미터

SAMLResponse

사용자 풀의 유효한 앱 클라이언트 및 IdP 구성과 연결된 IdP의 Base64-encoded SAML 어설션입니다.

RelayState

`RelayState` 파라미터에는 `oauth2/authorize` 엔드포인트에 전달할 요청 파라미터가 포함됩니다. 이러한 파라미터에 대한 자세한 내용은 권한 부여 엔드포인트 섹션을 참조하세요.

response_type

OAuth 2.0 권한 부여 유형입니다.

client_id

앱 클라이언트 ID입니다.

redirect_uri

Amazon Cognito가 사용자에게 권한을 부여한 후 인증 서버에서 브라우저를 리디렉션하는 URL입니다.

identity_provider

사용자를 리디렉션하려는 ID 제공업체의 이름입니다.

idp_identifier

사용자를 리디렉션하려는 ID 제공업체의 식별자입니다.

범위

사용자가 권한 부여 서버에서 요청할 OAuth 2.0 범위입니다.

긍정 응답이 있는 요청 예제

예 – POST 요청

다음 요청은 앱 클라이언트 `1example23456789`의 IdP `MySAMLIdP`에서 사용자에 대한 권한 부여 코드 부여를 요청하는 것입니다. 사용자는 권한 부여 코드를 사용하여 `https://www.example.com`로 리디렉션되며, 이 권한 부여 코드는 OAuth 2.0 범위 `openid`, `email` 및 phone의 액세스 토큰이 포함된 토큰으로 교환할 수 있습니다.

```
POST /saml2/idpresponse HTTP/1.1
User-Agent: USER_AGENT
Accept: */*
Host: example.auth.us-east-1.amazoncognito.com
Content-Type: application/x-www-form-urlencoded

SAMLResponse=[Base64-encoded SAML assertion]&RelayState=identity_provider
%3DMySAMLIdP%26client_id%3D1example23456789%26redirect_uri%3Dhttps%3A%2F
%2Fwww.example.com%26response_type%3Dcode%26scope%3Demail%2Bopenid%2Bphone
```

예 - 응답

다음은 이전 요청에 대한 응답입니다.

```
HTTP/1.1 302 Found
Date: Wed, 06 Dec 2023 00:15:29 GMT
Content-Length: 0
x-amz-cognito-request-id: 8aba6eb5-fb54-4bc6-9368-c3878434f0fb
```

```
Location: https://www.example.com?code=[Authorization code]
```

OAuth 2.0 권한 부여

Amazon Cognito 사용자 풀 OAuth 2.0 권한 부여 서버는 세 가지 유형의 OAuth 2.0 권한 부여에 대한 응답으로 토큰을 발급합니다. 사용자 풀의 각 앱 클라이언트에 대해 지원되는 권한 부여 유형을 설정할 수 있습니다. 암시적 또는 인증 코드 권한 부여와 동일한 앱 클라이언트에서 클라이언트 보안 인증 권한 부여를 활성화할 수 없습니다. 암시적 및 인증 코드 권한 부여에 대한 요청은 권한 부여 엔드포인트에서 시작되고, 클라이언트 보안 인증 권한 부여 요청은 Token 엔드포인트에서 시작됩니다.

인증 코드 권한 부여

성공적인 인증 요청에 대한 응답으로 권한 부여 서버는 콜백 URL에 `code` 파라미터의 인증 코드를 추가합니다. 그런 다음 ID, 액세스 및 새로 고침 토큰에 대한 코드를 Token 엔드포인트와 교환해야 합니다. 인증 코드 권한 부여를 요청하려면 요청에서 `response_type`을 `code`로 설정합니다. 요청 예시는 예: 권한 부여 코드 부여 섹션을 참조하세요. Amazon Cognito는 권한 부여 코드 부여에서 PKCE(Proof Key for Code Exchange)를 지원합니다.

인증 코드 권한 부여는 가장 안전한 형태의 권한 부여입니다. 이는 토큰 내용을 사용자에게 직접 표시하지 않습니다. 대신 앱이 사용자 토큰을 검색하고 안전하게 저장하는 역할을 합니다. Amazon Cognito에서는 인증 코드 권한 부여가 권한 부여 서버에서 세 가지 토큰 유형(ID, 액세스 및 새로 고침)을 모두 얻는 유일한 방법입니다. Amazon Cognito 사용자 풀 API를 통한 인증에서도 세 가지 토큰 유형을 모두 얻을 수 있지만 API는 `aws.cognito.signin.user.admin` 이외의 범위로 액세스 토큰을 발급하지 않습니다.

암시적 허용

성공적인 인증 요청에 대한 응답으로 권한 부여 서버는 `access_token` 파라미터의 액세스 토큰과 `id_token` 파라미터의 ID 토큰을 콜백 URL에 추가합니다. 암시적 권한 부여에는 Token 엔드포인트와의 추가 상호 작용이 필요하지 않습니다. 암시적 권한 부여를 요청하려면 요청에서 `response_type`을 `token`으로 설정합니다. 암시적 권한 부여는 ID와 액세스 토큰만 생성합니다. 요청 예시는 예: `openid` 범위가 없는 토큰(암시적) 권한 부여 섹션을 참조하세요.

암시적 권한 부여는 레거시 권한 부여입니다. 인증 코드 권한 부여와 달리 사용자가 토큰을 가로채고 검사할 수 있습니다. 암시적 권한 부여를 통한 토큰 전달을 방지하려면 인증 코드 권한 부여만 지원하도록 앱 클라이언트를 구성합니다.

클라이언트 자격 증명

클라이언트 보안 인증은 시스템 간 액세스에 대한 인증 전용 권한 부여입니다. 클라이언트 보안 인증 권한 부여를 받으려면 권한 부여 엔드포인트를 우회하고 Token 엔드포인트에 직접 요청을 생성

합니다. 앱 클라이언트는 클라이언트 보안 암호를 가져야 하며 클라이언트 자격 증명 권한 부여만 지원해야 합니다. 성공적인 요청에 대한 응답으로 권한 부여 서버는 액세스 토큰을 반환합니다.

클라이언트 보안 인증 권한 부여의 액세스 토큰은 OAuth 2.0 범위를 포함하는 권한 부여 메커니즘입니다. 일반적으로 토큰에는 액세스 보호된 API에 대한 HTTP 작업을 승인하는 사용자 지정 범위 클레임이 포함됩니다. 자세한 내용은 범위, M2M 및 리소스 서버 섹션을 참조하세요.

클라이언트 자격 증명 부여는 AWS 청구서에 비용을 추가합니다. 자세한 내용은 Amazon Cognito 요금을 참조하세요.

새로 고침 토큰

Token 엔드포인트에서 직접 새로 고침 토큰 부여를 요청할 수 있습니다. 이 권한 부여는 유효한 새로 고침 토큰에 대한 대가로 새 ID 및 액세스 토큰을 반환합니다.

이러한 권한 부여 및 구현에 대한 자세한 내용은 AWS 보안 블로그에서 Amazon Cognito의 OAuth 2.0: 다양한 OAuth 2.0 권한 부여에 대해 알아보기를 참조하세요.

권한 부여 코드 부여에서 PKCE 사용

Amazon Cognito는 권한 부여 코드 부여에서 PKCE(Proof Key for Code Exchange) 인증을 지원합니다. PKCE는 퍼블릭 클라이언트에 대한 OAuth 2.0 권한 부여 코드 권한 부여의 확장입니다. PKCE는 가로채기된 권한 부여 코드의 사용을 방지합니다.

Amazon Cognito가 PKCE를 사용하는 방법

PKCE로 인증을 시작하려면 애플리케이션이 고유한 문자열 값을 생성해야 합니다. 이 문자열은 Amazon Cognito가 초기 권한 부여를 요청하는 클라이언트를 토큰에 대한 권한 부여 코드를 교환하는 클라이언트와 비교하는 데 사용하는 보안 암호 값인 코드 검증자입니다.

앱은 코드 검증자 문자열에 SHA256 해시를 적용하고 결과를 base64로 인코딩해야 합니다. 해시된 문자열을 요청 본문의 `code_challenge` 파라미터로 권한 부여 엔드포인트에 전달합니다. 앱이 토큰에 대한 권한 부여 코드를 교환하는 경우 Token 엔드포인트에 대한 요청 본문에 `code_verifier` 파라미터로 일반 텍스트의 코드 검증자 문자열을 포함해야 합니다. Amazon Cognito는 코드 검증자에서 동일한 해시 및 인코딩 작업을 수행합니다. Amazon Cognito는 코드 검증자가 권한 부여 요청에서 받은 것과 동일한 코드 챌린지를 초래하는 것으로 확인되는 경우에만 ID, 액세스 및 새로 고침 토큰을 반환합니다.

PKCE를 사용하여 권한 부여 흐름을 구현하려면

1. Amazon Cognito 콘솔을 엽니다. 메시지가 표시되면 자격 AWS 증명을 입력합니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다. 사용자 풀을 생성하는 경우 마법사 중에 앱 클라이언트를 설정하고 관리형 로그인을 구성하라는 메시지가 표시됩니다.

   a. 새로운 사용자 풀을 생성하는 경우, 가이드 설정 중에 앱 클라이언트를 설정하고 관리형 로그인을 구성합니다.

   b. 기존 사용자 풀을 구성하는 경우 도메인과 퍼블릭 앱 클라이언트를 아직 추가하지 않았다면 추가합니다.
4. 무작위 영숫자 문자열, 일반적으로 UUID(범용 고유 식별자)를 생성하여 PKCE에 대한 코드 챌린지를 생성합니다. 이 문자열은 요청에서 Token 엔드포인트에 제출할 `code_verifier` 파라미터의 값입니다.
5. SHA256 알고리즘으로 `code_verifier` 문자열을 해시합니다. 해싱 작업의 결과를 base64로 인코딩합니다. 이 문자열은 요청에서 권한 부여 엔드포인트에 제출할 `code_challenge` 파라미터의 값입니다.

   다음 Python 예시에서는 `code_verifier`를 생성하고 `code_challenge`를 계산합니다.

```
#!/usr/bin/env python3

import secrets
from base64 import urlsafe_b64encode
from hashlib import sha256
from string import ascii_letters
from string import digits

# use the secrets module for cryptographically strong random values
alphabet = ascii_letters + digits
code_verifier = ''.join(secrets.choice(alphabet) for _ in range(128))
code_verifier_hash = sha256(code_verifier.encode()).digest()
code_challenge = urlsafe_b64encode(code_verifier_hash).decode().rstrip('=')

print(f"code challenge: {code_challenge}")
print(f"code verifier: {code_verifier}")
```

   다음은 Python 스크립트의 출력 예시입니다.

```
code challenge: Eh0mg-OZv7BAyo-tdv_vYamx1boOYDulDklyXoMDtLg
code verifier: 9D-aW_iygXrgQcWJd0y0tNVMPSXSChIc2xceDhvYVdGLCBk-
JWFTmBNjvKSdOrjTTYazOFbUmrFERrjWx6oKtK2b6z_x4_gHBDlr4K1mRFGyE8yA-05-_v7Dxf3EIYJH
```

6. PKCE를 통한 권한 부여 코드 부여 요청으로 관리형 로그인을 완료합니다. 다음은 URL의 예입니다.

```
https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&client_id=1example23456789&redirect_uri=https://
www.example.com&code_challenge=Eh0mg-OZv7BAyo-
tdv_vYamx1boOYDulDklyXoMDtLg&code_challenge_method=S256
```

7. 권한 부여 code를 수집하고 토큰 엔드포인트를 사용하여 토큰으로 교환합니다. 다음은 요청 예제입니다.

```
POST /oauth2/token HTTP/1.1
Host: mydomain.auth.us-east-1.amazoncognito.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 296

redirect_uri=https%3A%2F%2Fwww.example.com&
client_id=1example23456789&
code=7378f445-c87f-400c-855e-0297d072ff03&
grant_type=authorization_code&
code_verifier=9D-aW_iygXrgQcWJd0y0tNVMPSXSChIc2xceDhvYVdGLCBk-
JWFTmBNjvKSdOrjTTYazOFbUmrFERrjWx6oKtK2b6z_x4_gHBDlr4K1mRFGyE8yA-05-_v7Dxf3EIYJH
```

8. 응답을 검토합니다. 여기에는 ID, 액세스 및 새로 고침 토큰이 포함됩니다. Amazon Cognito 사용자 풀 토큰 사용에 대한 자세한 내용은 사용자 풀 JSON 웹 토큰(JWT) 이해 섹션을 참조하세요.

관리형 로그인 및 페더레이션 오류 응답

관리형 로그인이나 페더레이션 로그인 프로세스에서 오류가 반환될 수 있습니다. 인증이 오류와 함께 종료될 수 있는 몇 가지 조건은 다음과 같습니다.

- 사용자가 사용자 풀이 수행할 수 없는 작업을 수행합니다.
- Lambda 트리거가 예상 구문으로 응답하지 않습니다.
- ID 제공업체(idP)가 오류를 반환합니다.
- Amazon Cognito가 사용자가 제공한 속성 정보를 검증하지 못했습니다.

- IdP가 필수 속성에 매핑되는 클레임을 보내지 않았습니다.

Amazon Cognito는 오류가 발생하면 다음 방법 중 하나로 이를 알립니다.

1. Amazon Cognito가 요청 파라미터에 오류가 있는 리디렉션 URL을 전송합니다.
2. Amazon Cognito는 관리형 로그인에 오류를 표시합니다.

Amazon Cognito가 요청 파라미터에 추가하는 오류의 형식은 다음과 같습니다.

```
https://<Callback URL>/?error_description=error+description&error=error+name
```

작업을 수행할 수 없는 사용자의 오류 정보 제출을 도와주는 경우 페이지의 URL 및 텍스트 또는 스크린샷을 캡처하도록 요청하세요.

> **Note**
>
> Amazon Cognito 오류 설명은 고정된 문자열이 아니므로 고정된 패턴 또는 형식을 사용하는 로직을 사용해서는 안 됩니다.

OIDC 및 소셜 ID 제공업체 오류 메시지

ID 제공업체가 오류를 반환할 수 있습니다. OIDC 또는 OAuth 2.0 IdP가 표준을 준수하는 오류를 반환하면 Amazon Cognito는 사용자를 콜백 URL로 리디렉션하고 오류 요청 파라미터에 공급자 오류 응답을 추가합니다. Amazon Cognito는 기존 오류 문자열에 공급자 이름과 HTTP 오류 코드를 추가합니다.

다음 URL은 오류를 Amazon Cognito에 반환한 IdP에서의 리디렉션 예입니다.

```
https://www.amazon.com/?error_description=LoginWithAmazon+Error+-+400+invalid_request
+The+request+is+missing+a+required+parameter+%3A+client_secret&error=invalid_request
```

Amazon Cognito는 공급자로부터 받은 것만 반환하므로 사용자가 보는 정보는 이 정보의 일부일 수 있습니다.

IdP를 통한 사용자의 최초 로그인에 문제가 발생하면 IdP는 오류 메시지를 사용자에게 직접 전달합니다. Amazon Cognito는 IdP에 대한 사용자 세션 검증 요청을 생성할 때 사용자에게 오류 메시지를 전달합니다. Amazon Cognito는 다음 엔드포인트의 OAuth 및 OIDC IdP 오류 메시지를 전달합니다.

`/token`

Amazon Cognito는 IdP 권한 부여 코드를 액세스 토큰으로 교환합니다.

`/.well-known/openid-configuration`

Amazon Cognito는 발급자 엔드포인트 경로를 검색합니다.

`/.well-known/jwks.json`

사용자의 JSON 웹 토큰(JWTs)을 확인하기 위해 Amazon Cognito는 IdP가 토큰에 서명하는 데 사용하는 JSON 웹 키(JWKs)를 검색합니다.

Amazon Cognito는 HTTP 오류를 반환할 수 있는 SAML 2.0 공급자에 대한 아웃바운드 세션을 시작하지 않으므로 SAML 2.0 IdP 세션 중에 발생한 사용자 오류에는 이러한 형태의 공급자 오류 메시지가 포함되지 않습니다.

배포 및 운영

Amazon Cognito 설계가 적용되면 프로덕션 서비스의 일부로 운영합니다. 이 섹션에서는 요청 속도 및 리소스 할당량, 프로비저닝된 용량, 다중 리전 복제, 리소스 태그 지정, 문제 해결 등 대규모 Amazon Cognito 실행의 운영 문제를 다룹니다. 로깅 및 모니터링, 즉 CloudWatch 지표 및 CloudTrail 이벤트 기록은 보안 Amazon Cognito의 로깅 및 모니터링 장의 섹션을 참조하세요.

주제

- Amazon Cognito의 할당량
- 사용자 풀에 대한 다중 리전 복제
- Amazon Cognito 리소스 태깅
- Amazon Cognito 문제 해결

Amazon Cognito의 할당량

Amazon Cognito에는 계정에서 수행할 수 있는 최대 작업 수에 대한 기본 할당량(구 명칭: 한도)이 있습니다. Amazon Cognito에는 또한 Amazon Cognito 리소스의 최대 수와 크기에 대한 할당량이 지정되어 있습니다.

각 Amazon Cognito 할당량은 하나의에 있는 요청의 최대 볼륨 AWS 리전 을 나타냅니다 AWS 계정. 예를 들어 앱은 미국 동부(버지니아 북부)의 모든 사용자 풀을 대상으로 하는 `UserAuthentication` 작업에 최대 기본 할당량(RPS) 요율로 API 요청을 보낼 수 있습니다. 아시아 태평양(도쿄)의 앱은 자체 리전의 모든 사용자 풀에 대해 동일한 양의 요청을 생성할 수 있습니다.는 한 번에 한 리전에서만 할당량 증가 요청을 부여할 AWS 수 있습니다. 미국 동부(버지니아 북부)의 할당량 증가는 아시아 태평양(도쿄)의 최대 요청률에 영향을 주지 않습니다.

주제

- API 요청률 할당량 이해
- API 요청률 할당량 관리
- Amazon Cognito 사용자 풀 API 작업 범주 및 요청 속도 할당량
- Amazon Cognito 자격 증명 풀(페더레이션 자격 증명) API연산 요청 속도 할당량
- 리소스 수 및 크기에 대한 할당량

API 요청률 할당량 이해

할당량 분류

Amazon Cognito는 API 작업에 대한 최대 요청 속도를 적용합니다. Amazon Cognito에서 사용할 수 있는 API 작업에 대한 자세한 내용은 사용자 풀 및 ID 풀에 대한 API 참조 가이드를 참조하십시오. 사용자 풀의 경우 이러한 작업은 `UserAuthentication` 또는 `UserCreation`과 같은 일반적인 사용 사례의 범주로 그룹화됩니다. 범주별 사용자 풀 API 작업 목록은 Amazon Cognito 사용자 풀 API 작업 범주 및 요청 속도 할당량 섹션을 참조하세요.

Service Quotas 콘솔에서 범주 사용자 풀 및 ID 풀별로 할당량 사용량을 추적할 수 있습니다. Amazon Cognito 사용자 풀의 요청 속도가 할당량을 초과하거나 할당량을 초과하는 경우 추가 용량을 구매할 수 있습니다. Service Quotas 콘솔에서 카테고리별 사용자 풀 할당량 사용량 및 구매 할당량 증가를 추적할 수 있습니다.

작업 할당량은 범주 내의 모든 작업에 대한 최대 초당 요청 수(RPS)로 정의됩니다. Amazon Cognito 사용자 풀 서비스는 각 범주의 모든 작업에 할당량을 적용합니다. 예를 들어 `UserCreation` 범주에는 `SignUp`, `ConfirmSignUp`, `AdminCreateUser`, `AdminConfirmSignUp`이라는 네 가지 작업이 포함됩니다. 총 할당량을 기준으로 50RPS가 할당됩니다. 동시에 여러 작업이 수행되는 경우, 이 범주 내의 각 작업은 최대 50개의 RPS를 개별적으로 호출하거나 조합할 수 있습니다.

> **Note**
>
> 범주 할당량은 사용자 풀에만 적용됩니다. Amazon Cognito는 각 ID 풀 할당량을 단일 작업에 적용합니다. 범주별 및 작업별 요청 속도 할당량 모두에 대해는 AWS 계정 한 리전의에 있는 모든 사용자 풀 또는 자격 증명 풀의 모든 요청의 집계 비율을 AWS 측정합니다.

특별 요청 속도 처리가 포함된 Amazon Cognito 사용자 풀 API 작업

작업 할당량은 범주 수준에서 합산된 전체 요청을 기준으로 측정되고 적용됩니다. 단, 특수 처리 규칙이 적용되는 `AdminRespondToAuthChallenge` 및 `RespondToAuthChallenge` 작업은 예외입니다.

`UserAuthentication` 범주에는 Amazon Cognito 사용자 풀 API의 4가지 작업(`AdminInitiateAuth`, `InitiateAuth`, `AdminRespondToAuthChallenge` 및 `RespondToAuthChallenge`)이 포함됩니다. 또한 호스팅 UI의 사용자 인증은 이 할당량에 기여합니다. `InitiateAuth` 및 `AdminInitiateAuth` 작업은 범주 할당량에 따라 측정되고 적

용됩니다. RespondToAuthChallenge과 AdminRespondToAuthChallenge의 매칭 작업은 UserAuthentication 범주 제한의 세 배에 해당하는 별도 할당량이 적용됩니다. 이 높은 할당량은 사용자의 앱에 설정된 여러 인증 문제를 수용합니다. 해당 할당량은 대다수의 사용 사례를 수용하기에 충분합니다. 앱이 인증 문제에 대해 최대 3개까지 응답하면 추가 요청이 UserAuthentication 범주 할당량에 포함됩니다. 다중 인증(MFA), 디바이스 인증 및 사용자 지정 인증은 모두 사용자 풀에 엔지니어링할 수 있는 챌린지 프롬프트의 예입니다.

예를 들어 UserAuthentication 범주의 할당량이 80RPS인 경우 RespondToAuthChallenge 또는 AdminRespondToAuthChallenge를 최대 240RPS(3*80RPS)의 속도로 호출할 수 있습니다. 사용자 풀이 인증당 4차례의 챌린지를 요청하고 초당 70명의 사용자가 로그인하는 경우 총 RespondToAuthChallenge는 280RPS(70x4)로, 할당량보다 40RPS 높습니다. 추가 40RPS가 70건의 InitiateAuth 호출에 추가되어 UserAuthentication 범주의 총 사용량은 110RPS(40+70)가 됩니다. 이 값은 80RPS로 설정된 범주 할당량을 30RPS 초과하므로 Amazon Cognito는 앱의 요청을 제한합니다.

월별 활성 사용자)

Amazon Cognito가 사용자 풀 청구를 계산하면 월간 활성 사용자(MAU)에 대한 요금이 부과됩니다. 할당량 증가 요청을 계획할 때 현재 및 예상 MAU 수를 고려합니다. 한 달 이내에 관련된 자격 증명 작업이 있는 사용자는 MAU로 계산됩니다. SAML 또는 OIDC 페더레이션을 사용하여 페더레이션 사용자를 로컬 사용자에 연결하면 사용자가 직접 로그인하든 페더레이션을 통해 로그인하든 EnterpriseMAU상관없이 로컬 사용자는 엔터프라이즈 디렉터리 MAU 또는 로 계산됩니다. 자세한 내용은 Amazon Cognito 요금을 참조하세요.

- 사용자 가입 또는 관리 생성. 사용자 CSV 가져오기는 MAU 수에 영향을 주지 않습니다.
- 사용자 계정 확인 또는 속성 검증.
- 로그인 및 문제 응답. 현재 로그인한 사용자의 액세스 토큰으로 권한을 부여하는 작업은 MAU 수에 영향을 주지 않지만 로그인 시 액세스 토큰이 생성되므로 이러한 작업은 연결된 사용자가 MAU임을 나타냅니다.
- 로그아웃 및 토큰 취소.
- 암호 셀프 서비스 재설정 및 관리자로서 사용자 암호 설정. 관리자(AdminResetUserPassword)로 사용자 암호를 재설정해도 MAU 수는 포함되지 않습니다.
- 사용자 속성이나 그룹 멤버십을 변경합니다.
- 관리자로서 사용자의 세부 속성을 쿼리합니다.

> **Note**
>
> 관리자로서 사용자의 세부 속성 쿼리 범주에는 API 작업 AdminGetUser가 포함되지만 ListUsers는 포함되지 않습니다. 대규모 사용자 풀의 user-by-user 세부 쿼리는 AWS 청구서에 상당한 영향을 미칠 수 있습니다. 추가 비용을 방지하려면 `ListUsers`를 사용하여 사용자 데이터를 수집하거나 외부 데이터베이스에 사용자 정보를 저장하세요.

활성 사용자 또는 한 달 동안 활성 상태가 아닌 사용자의 추가 세션에 대해서는 요금이 부과되지 않습니다. Lite, Essentials 및 Plus의 사용 가능한 옵션 간에 사용자 풀 기능 플랜을 변경한 달에는 해당 월의 청구서가 각 계층의 월별 활성 사용자(MAU) 합계를 기준으로 계산되며, 각 MAU는 사용자가 활성 상태일 때 가장 높은 가격이 할당된 계층에 할당됩니다. 예제:

1. 월초에는 사용자 풀이 Plus 기능 플랜에 있습니다.
2. 사용자 A는 매월 1일에 로그인합니다.
3. 사용자 B는 해당 월의 첫 번째 및 마지막 날에 로그인합니다.
4. 매월 10일에 기능 플랜을 Essentials로 전환합니다.
5. 사용자 C는 해당 월의 마지막 날에 로그인합니다.

이 시나리오에서 사용자 A와 사용자 B는 Plus MAU 사용자 C는 Essentials MAU입니다.

Lite MAU

: 사용자 풀이 Lite 기능 플랜에 있을 때 한 달에 한 번 이상 활성화되었고 사용자 풀이 Essentials 또는 Plus 플랜에 있을 때는 활성화되지 않은 사용자입니다.

Essentials MAU

: 사용자 풀이 Essentials 기능 플랜에 있을 때 한 달에 한 번 이상 활성화되었고 사용자 풀이 Plus 플랜에 있을 때는 활성화되지 않은 사용자입니다.

Plus MAU

: 사용자 풀이 Plus 플랜에 있을 때 한 달에 한 번 이상 활성 상태였던 사용자입니다.

자세한 내용은 사용자 풀 기능 플랜 섹션을 참조하세요.

API 요청률 할당량 관리

할당량 요구 사항 파악

> **Important**
>
> `UserAuthentication`, `UserCreation`, `AccountRecovery` 등의 범주에 대한 Amazon Cognito 할당량을 늘리는 경우 다른 AWS 서비스의 할당량도 늘려야 할 수 있습니다. 예를 들어 Amazon Cognito가 Amazon Simple Notification Service(Amazon SNS) 및 Amazon Simple Email Service(Amazon SES)를 통해 보내는 메시지는 해당 서비스의 요청 속도 할당량이 충분하지 않은 경우 실패할 수 있습니다.

할당량 요구 사항을 계산하려면 특정 기간 동안 애플리케이션과 상호 작용할 활성 사용자 수를 구합니다. 예를 들어 애플리케이션에서 평균 100만 명의 활성 사용자가 8시간 내에 로그인할 것으로 예상되는 경우 초당 평균 35명의 사용자를 인증할 수 있어야 합니다.

또한 평균 사용자 세션이 2시간이라고 가정하고 토큰이 1시간 후에 만료되도록 구성한 경우 각 사용자가 세션 중에 토큰을 한 번 새로 고쳐야 합니다. 이 경우 `UserAuthentication` 범주에서 부하를 지원하는 데 필요한 평균 할당량은 70RPS입니다.

8시간 동안 사용자 로그인 빈도의 편차를 고려한 피크 대 평균 비율이 3:1이라고 가정할 경우 `UserAuthentication` 할당량으로 200RPS가 필요합니다.

> **Note**
>
> 각 사용자 작업에 대해 여러 작업을 호출하는 경우 범주 수준에서 개별 작업 호출 속도를 합산해야 합니다.

할당량 한도에 대한 요청 속도 최적화

API 속도 제한을 늘리면 AWS 청구서에 비용이 추가되므로 할당량 증가를 요청하기 전에 사용량 모델을 조정하는 것이 좋습니다. 다음은 요청 속도를 최적화하는 앱 아키텍처의 몇 가지 예입니다.

백오프 대기 기간 후에 다시 시도

각 API 호출에서 오류를 포착한 다음 백오프 기간 후에 다시 시도할 수 있습니다. 비즈니스 요구 사항과 로드에 따라 백오프 알고리즘을 조정할 수 있습니다. Amazon SDK에는 재시도 로직이 내장되어 있습니다. 자세한 내용은 빌드 기반 도구를 참조하세요 AWS.

자주 업데이트되는 속성에 외부 데이터베이스 사용

애플리케이션에서 사용자 지정 속성을 읽거나 쓰기 위해 사용자 풀을 여러 번 호출해야 하는 경우 외부 스토리지를 사용합니다. 기본 설정 데이터베이스를 사용하여 사용자 지정 속성을 저장하거나 로그인 중에 캐시 계층을 사용하여 사용자 프로파일을 로드할 수 있습니다. 필요한 경우 사용자 풀에서 사용자 프로파일을 다시 로드하는 대신, 캐시에서 이 프로파일을 참조할 수 있습니다.

클라이언트 측에서 JSON 웹 토큰(JWT) 유효성 검사

애플리케이션은 JWT 토큰을 신뢰하기 전에 먼저 검증해야 합니다. API 요청을 사용자 풀에 보내지 않고도 클라이언트 측에서 토큰의 서명과 유효성을 확인할 수 있습니다. 토큰을 검증한 후에는 토큰의 클레임을 신뢰하고 클레임을 더 많은 `getUser` API 호출을 하는 대신 클레임을 사용할 수 있습니다. 자세한 내용은 JSON 웹 토큰 확인을 참조하세요.

대기실에서 웹 애플리케이션에 대한 트래픽 제한

시험 응시나 라이브 이벤트 참석 등 제한 시간이 있는 이벤트 중에 로그인한 많은 사용자의 트래픽이 예상되는 경우 자체 조절 메커니즘을 사용하여 요청 트래픽을 최적화할 수 있습니다. 예를 들어, 사용자가 세션을 사용할 수 있을 때까지 대기실을 설정하여 사용 가능한 용량이 있을 때 요청을 처리할 수 있습니다. 대기실 참조 아키텍처에 대해서는 AWS 가상 대기실 솔루션을 참조하세요.

JWT 캐시

만료될 때까지 액세스 토큰을 재사용합니다. API Gateway에서 토큰 캐싱을 사용하는 예시 프레임워크는 사용자 풀 토큰 만료 및 캐싱 관리 섹션을 참조하세요. 사용자 정보를 쿼리하는 API 요청을 생성하는 대신 만료될 때까지 ID 토큰을 캐시하고 캐시에서 사용자 속성을 읽습니다.

에서 API 요청 속도를 사용하는 방법에 대한 자세한 내용은 워크로드에서 API 제한 관리 및 모니터링을 AWS참조하세요. https://aws.amazon.com/blogs/mt/managing-monitoring-api-throttling-in-workloads/ AWS 청구서에 비용을 추가하는 Amazon Cognito 작업 최적화에 대한 자세한 내용은 섹션을 참조하세요비용 관리.

할당량 사용량 추적

Amazon Cognito가 계정 수준에서 각 API 작업 범주에 대한 Amazon CloudWatch의 `CallCount` 및 `ThrottleCount` 지표를 생성합니다. `CallCount`를 사용하여 범주와 관련된 고객의 총 호출 수를 추적할 수 있습니다. `ThrottleCount`를 사용하여 범주와 관련된 총 제한된 호출 수를 추적할 수 있습니다. `CallCount` 및 `ThrottleCount` 지표와 `Sum` 통계를 사용하여 범주의 총 호출 수를 계산할 수 있습니다. 자세한 내용은 CloudWatch 사용량 지표를 참조하세요.

서비스 할당량을 모니터링할 때 사용률은 사용 중인 서비스 할당량의 백분율입니다. 예를 들어 할당량 값이 리소스 200개이고, 150개의 리소스가 사용 중인 경우 사용률은 75%입니다. 사용량은 서비스 할당량으로 사용 중인 리소스 또는 작업의 수입니다.

CloudWatch 지표를 통한 사용량 추적

CloudWatch를 사용하여 Amazon Cognito 사용자 풀 사용률 지표를 추적하고 수집할 수 있습니다. CloudWatch 대시보드에는 사용하는 모든에 대한 지표 AWS 서비스 가 표시됩니다. CloudWatch 를 사용하면 지표 경보를 생성하여 모니터링 중인 특정 리소스를 알리거나 변경할 수 있습니다. CloudWatch 사용량 지표에 대한 자세한 내용은 CloudWatch 사용량 지표 추적을 참조하세요.

Service Quotas 지표를 통한 사용률 추적

Amazon Cognito 사용자 풀은 서비스 할당량 사용량을 표시하고 관리하기 위한 콘솔 인터페이스인 Service Quotas와 통합됩니다. Service Quotas 콘솔에서 특정 할당량 값을 조회하고, 모니터링 정보를 보고, 할당량 증가를 요청하고, CloudWatch 경보를 설정할 수 있습니다. 계정이 한동안 활성화된 상태로 지난 경우 리소스 사용률 그래프를 볼 수 있습니다.

Amazon Cognito 사용자 풀 및 Amazon Cognito ID 풀에 대한 Service Quotas 콘솔의 적용된 계정 수준 할당량 값 열에는 현재 할당량이 표시됩니다. 사용률 열에는 현재 할당량의 사용률이 표시됩니다. 조정 가능한 Amazon Cognito 사용자 풀 RPS(초당 요청 수) 할당량은 현재 사용량을 표시합니다. Service Quotas 콘솔에서는 CloudWatch 지표로 이동하여 선택한 할당량 지표를 자세히 살펴볼 수도 있습니다. Service Quotas 콘솔에서 할당량을 확인하는 방법에 대한 자세한 내용은 Service Quotas 보기를 참조하세요.

월별 활성 사용자(MAU) 추적

사용자 풀의 월간 활성 사용자 수(MAU)는 요청 속도 할당량 증가를 위한 계획에 중요한 데이터를 기여합니다. API 요청 속도를 지정된 기간 동안 활성 상태였던 사용자 수와 비교할 수 있습니다. 이러한 지식을 바탕으로 애플리케이션의 활성 사용자 증가가 사용 모델의 할당량에 미치는 영향을 계산할 수 있습니다. 예를 들어 미국 서부(오리건)의 통합 애플리케이션에서 한 달에 활성 사용자가 2백만 명이고 `UserAuthentication` 범주에 초당 요청 수(RPS) 120개의 기본 할당량에서 가끔 스로틀링 오류가 발생했다고 가정해 보겠습니다. 지난 달에 광고 캠페인이 성공하기 전에는 MAU가 100만 명이었고 애플리케이션은 80RPS를 넘지 않았습니다. 새로운 TV 스팟으로 인해 비슷한 급증이 발생할 것으로 예상되는 경우 조정된 할당량이 160RPS인 다음 백만 명의 사용자를 수용할 수 있도록 40RPS를 추가로 구매할 수 있습니다.

MAU를 검토하는 방법

AWS Billing 콘솔에 액세스하고 최근 청구서를 검토합니다. 서비스별 요금에서 Cognito를 필터링하여 해당 청구 기간의 MAU 내역을 볼 수 있습니다.

할당량 증가 요청

Amazon Cognito에는 각의 사용자 풀 및 자격 증명 풀에서 수행할 수 있는 초당 최대 작업 수에 대한 할당량이 있습니다 AWS 리전. 조정 가능한 Amazon Cognito 사용자 풀 API 요청 속도 할당량의 증가를 구매할 수 있습니다. 조정 가능한 사용자 풀 범주의 경우 Service Quotas에서 더 높은 계정 수준 최대 제한을 요청한 다음 Amazon Cognito 프로비저닝 API를 사용하여 원하는 속도를 설정할 수 있습니다. 자세한 내용은 프로비저닝된 제한 관리 섹션을 참조하세요.

더 높은 계정 수준 최대 한도를 요청하려면 Service Quotas 콘솔 또는 Service Quotas API 작업 `ListAWSDefaultServiceQuotas` 및를 사용합니다`RequestServiceQuotaIncrease`.

- Service Quotas 콘솔을 사용하여 계정 수준 최대 한도 증가를 요청하려면 Service Quotas 사용 설명서의 API 할당량 증가 요청을 참조하세요.
- 일부 계정 수준 최대 한도 증가는 자동 승인됩니다. 그렇지 않으면 10일 이내에 할당량 증가 요청을 완료하는 것을 AWS 목표로 합니다. 그러나 몇 가지 고려 사항으로 인해 요청 처리 시간이 10일을 초과할 수 있습니다. 예를 들어 일부 요청의 경우 Amazon Cognito에서 추가 하드웨어 용량을 프로비저닝해야 할 수 있으며 요청 볼륨의 계절적 증가로 인해 지연이 발생할 수 있습니다.
- Service Quotas에서 아직 할당량을 사용할 수 없는 경우 서비스 한도 증가 양식을 사용합니다.

> **Note**
>
> 조정 가능한 할당량만 늘릴 수 있습니다.

> **Note**
>
> 할당량 증가는 리전 구성입니다. 하나의 할당량 증가는 다른 리전의 할당량에 영향을 AWS 리전 주지 않습니다. 여러 리전에서 더 높은 API 할당량이 필요한 경우 각 리전에서 별도로 할당량 증가를 요청해야 합니다.

> **Note**
>
> 계정 수준 최대 한도 증가를 요청하는 것은 첫 번째 단계일 뿐입니다. 요청이 승인되면 Amazon Cognito 콘솔을 호출`UpdateProvisionedLimit`하거나 사용하여 프로비저닝된 한도를 설정해야 합니다. 자세한 내용은 프로비저닝된 제한 관리 섹션을 참조하세요.

프로비저닝된 제한 관리

Amazon Cognito 사용자 풀은 조정 가능한 API 속도 할당량을 위한 2계층 할당량 모델을 지원합니다. 이 모델을 사용하면 AWS Support에 문의하지 않고도 프로비저닝된 요청 속도를 프로그래밍 방식으로 늘리거나 줄일 수 있습니다. 두 계층은 다음과 같이 작동합니다.

계정 수준 최대 한도

: 프로비저닝할 수 있는 최대 속도는 Service Quotas를 통해 설정됩니다. 계정 수준 최대 한도를 늘리려면 Service Quotas 콘솔에서 할당량 증가 요청을 제출합니다. 일부 요청은 자동 승인됩니다. 자세한 내용은 할당량 증가 요청 섹션을 참조하세요.

프로비저닝된 제한

: Amazon Cognito가 현재 계정에 적용하는 초당 요청 수(RPS)의 속도입니다. Amazon Cognito 프로비저닝 API 작업을 사용하여이 값을 계정 수준 최대 한도까지 조정할 수 있습니다.

예를 들어 `UserAuthentication` 계정 수준 최대 한도가 500RPS인 경우 `UpdateProvisionedLimit` API 작업을 사용하여 프로비저닝된 한도를 300RPS로 설정할 수 있습니다. 나중에 프로비저닝된 제한을 기본값으로 다시 줄일 수 있으며 더 높은 요금으로 해당 시간에 대해서만 요금이 청구됩니다.

> **Important**
>
> 기본 요금보다 높은 프로비저닝된 용량에 대한 요금이 실제 사용 시간을 기준으로 비례 배분하여 청구됩니다. 요금은 Amazon Cognito 요금을 참조하세요.

콘솔에서 프로비저닝된 제한 관리

Amazon Cognito 콘솔에서 프로비저닝된 제한을 보고 관리할 수 있습니다. 사용자 풀 페이지를 열고 프로비저닝된 한도 탭을 선택합니다. 이 탭에서 조정 가능한 각 API 범주에 대해 현재 프로비저닝된 제한

및 계정 수준 최대 제한을 볼 수 있습니다. 승인된 계정 수준 최대 한도 내에서 프로비저닝된 속도를 늘리거나 줄일 수 있습니다.

API 작업 프로비저닝

Amazon Cognito는 프로비저닝된 제한을 관리하기 위해 다음과 같은 API 작업을 제공합니다. 이러한 작업에는 인증 및 해당 IAM 권한이 필요합니다.

GetProvisionedLimit

특정 API 범주에 대해 현재 프로비저닝된 제한 값을 반환합니다. 응답에는 프로비저닝된 제한 값과 기본(무료) 제한 값이 포함됩니다.

UpdateProvisionedLimit

특정 API 범주에 대해 프로비저닝된 제한 값을 설정합니다. 요청된 값은 기본 한도와 Service Quotas 계정 수준 최대 한도 사이여야 합니다. 이 작업은 즉시 적용됩니다.

두 작업 모두 관리할 제한을 식별하는 `LimitDefinition` 파라미터를 사용합니다. 사용자 풀 API 속도 제한의 경우 제한 정의에는의 클래스`API_CATEGORY`와 `Category` 키가 있는 속성 맵이 있습니다. 예를 들어 `UserAuthentication` 제한을 관리하려면를 사용하여 제한 정의를 지정합니다`{"Category": "UserAuthentication"}`.

Example GetProvisionedLimit 요청 및 응답 예제

다음 예제에서는 `UserAuthentication` API 범주에 대해 프로비저닝된 제한을 반환합니다.

요청

```
{
    "LimitDefinition": {
        "LimitClass": "API_CATEGORY",
        "Attributes": {"Category": "UserAuthentication"}
    }
}
```

응답

```
{
    "Limit": {
```

```
        "LimitDefinition": {
            "LimitClass": "API_CATEGORY",
            "Attributes": {"Category": "UserAuthentication"}
        },
        "ProvisionedLimitValue": 120,
        "FreeLimitValue": 120
    }
}
```

Example UpdateProvisionedLimit 요청 및 응답 예제

다음 예제에서는 UserAuthentication API 범주의 프로비저닝된 제한을 300RPS로 설정합니다.

요청

```
{
    "LimitDefinition": {
        "LimitClass": "API_CATEGORY",
        "Attributes": {"Category": "UserAuthentication"}
    },
    "RequestedLimitValue": 300
}
```

응답

```
{
    "Limit": {
        "LimitDefinition": {
            "LimitClass": "API_CATEGORY",
            "Attributes": {"Category": "UserAuthentication"}
        },
        "ProvisionedLimitValue": 300,
        "FreeLimitValue": 120
    }
}
```

다음 IAM 작업은 이러한 작업에 대한 액세스를 제어합니다.

- cognito-idp:GetProvisionedLimit
- cognito-idp:UpdateProvisionedLimit

> **Note**
>
> 교차 계정 액세스는 지원되지 않습니다. 소유 계정만 프로비저닝된 제한을 관리할 수 있습니다.

프로비저닝 워크플로

다음 워크플로에서는 프로비저닝된 제한을 늘리는 방법을 설명합니다.

1. 필요한 요금을 결정합니다. 자세한 내용은 할당량 요구 사항 파악 섹션을 참조하세요.
2. 필요한 요금이 현재 Service Quotas 계정 수준 최대 한도를 초과하는 경우 Service Quotas 콘솔에서 증가를 요청합니다.
3. 계정 수준 최대 한도가 승인되면 원하는 속도로 `UpdateProvisionedLimit`를 호출합니다.
4. 프로비저닝된 요금을 줄이고 추가 결제`UpdateProvisionedLimit`를 중지하려면 기본 제한 값 또는 더 낮은 값으로를 호출합니다.

고려 사항

- 프로비저닝된 제한은 계정 수준 리소스입니다. 이는에 있는 하나의에 있는 모든 사용자 풀의 모든 요청의 집계 속도 AWS 리전 에 적용됩니다 AWS 계정.
- `GetProvisionedLimit` 및를 포함하는 `LimitManagement` 범주`UpdateProvisionedLimit`는 계정당 초당 1개의 요청으로 속도가 제한됩니다.
- 조정 가능한 할당량 범주만 프로비저닝을 지원합니다. 조정 가능한 범주 목록은 섹션을 참조하세요Amazon Cognito 사용자 풀 API 작업 범주 및 요청 속도 할당량.
- 사용자 풀에서 관리형 로그인을 사용하는 경우 프로비저닝 API를 사용하여 `UserAuthentication` 또는 `UserFederation` 범주를 조정할 수 없습니다. 이러한 할당량을 늘리려면 Service Quotas 콘솔을 통해 Service Quotas 증가 요청을 제출하거나 서비스 한도 증가 양식을 사용합니다.
- 프로비저닝된 할당량에 대한 변경 사항은 리전별로 적용됩니다. 한의 프로비저닝된 할당량 변경은 다른 리전의 할당량에 영향을 AWS 리전 주지 않습니다.
- 프로비저닝 API 작업에 대한 모든 호출이 로그인됩니다 AWS CloudTrail.

Amazon Cognito 사용자 풀 API 작업 범주 및 요청 속도 할당량

Amazon Cognito에는 권한 부여 모델이 서로 다른 API 작업 클래스가 중복되므로, 각 작업은 단일 범주에 속합니다. 각 범주에는 계정의 단일 AWS 리전 에 있는 모든 사용자 풀에 대한 모든 멤버 API 작업에

대한 자체 폴링된 할당량이 있습니다. 조정 가능한 범주 할당량 증가만 요청할 수 있습니다. 자세한 내용은 할당량 증가 요청 섹션을 참조하세요. 할당량 조정은 단일 리전의 계정에 있는 사용자 풀에 적용됩니다. Amazon Cognito는 일부 범주[3]의 작업을 사용자 풀당 초당 요청(RPS) 5개로 제한합니다. 기본 할당량(RPS)은 AWS 계정의 모든 사용자 풀에 추가로 적용됩니다.

> **Note**
>
> 각 범주의 할당량은 월간 활성 사용자(MAU) 수로 측정됩니다. MAU가 200만 개 미만인 AWS 계정 는 기본 할당량 내에서 작동할 수 있습니다. MAU가 100만 명 미만이고 Amazon Cognito가 요청을 제한하는 경우 앱을 최적화하는 것이 좋습니다. 자세한 내용은 할당량 한도에 대한 요청 속도 최적화 섹션을 참조하세요.

범주 작업 할당량은 단일 AWS 리전내 모든 사용자 풀의 모든 사용자에게 적용됩니다. 또한 Amazon Cognito는 앱이 사용자 한 명에 대해 생성할 수 있는 요청 수에 대한 할당량을 유지합니다. 사용자별 API 요청을 다음 표와 같이 제한해야 합니다.

Amazon Cognito 사용자 풀 사용자별 요청률 할당량

연산	사용자별 초당 연산
사용자 프로필 읽기 예: `GetUser`, `GetDevice` , `InitiateAuth` , `RespondToAuthChallenge`	10
사용자 프로필 쓰기 예: `UpdateUserAttributes` , `SetUserSettings`	10

범주별 API 요청을 다음 표와 같이 제한해야 합니다.

Amazon Cognito 사용자 풀 범주별 요청률 할당량

범주	설명	기본 할당량(RPS)	조정 가능
`UserAuthentication`	사용자를 인증(로그인)하는 작업입니다.	120	예

범주	설명	기본 할당량(RPS)	조정 가능
• InitiateAuth • `InitiateAuth` 또는 토큰 엔드포인트를 사용한 토큰 새로 고침 • RespondToAuthChallenge[1] • AdminInitiateAuth • AdminRespondToAuthChallenge[1] • GetTokensFromRefreshToken • 권한 부여 코드 또는 암시적 권한[2] 부여에서 로컬 사용자를 위한 관리형 로그인 또는 클래식 호스팅 UI 로그인 및 MFA	이러한 작업에는 특별 요청 속도 처리가 포함된 Amazon Cognito 사용자 풀 API 작업이(가) 적용됩니다.		
`UserCreation` • SignUp • ConfirmSignUp • AdminCreateUser • AdminConfirmSignUp	Amazon Cognito 로컬 사용자를 생성하거나 확인하는 작업입니다. Amazon Cognito 사용자 풀에서 직접 생성 및 검증하는 사용자입니다.	50	예

범주	설명	기본 할당량(RPS)	조정 가능
`UserFederation` 서드 파티 ID 공급자를 사용하여 사용자를 Amazon Cognito 사용자 풀로 페더레이션(인증)하는 작업입니다.	사용자 풀 페더레이션 엔드포인트에 IdP 응답을 제출하는 작업입니다. IdP 토큰을 생성하는 OIDC 또는 소셜 제공업체 작업과 모든 SAML 요청이 이 할당량에 기여합니다.	25	예
`UserAccountRecovery` • ChangePassword • ConfirmForgotPassword • ForgotPassword • AdminResetUserPassword • AdminSetUserPassword • RespondToAuthChallenge[1] • AdminRespondToAuthChallenge[1] • 관리형 로그인 암호 재설정	사용자 계정을 복구하거나 사용자 암호를 변경 또는 업데이트하는 작업입니다.	30	아니요

범주	설명	기본 할당량(RPS)	조정 가능
`UserRead` • AdminGetUser • AdminGetUserAuthFactors • GetUser	사용자 풀에서 사용자를 검색하는 작업입니다.	120	예

범주	설명	기본 할당량(RPS)	조정 가능
`UserUpdate` • AdminAddUserToGroup • AdminDeleteUserAttributes • AdminUpdateUserAttributes • AdminDeleteUser • AdminDisableUser • AdminEnableUser • AdminLinkProviderForUser • AdminDisableProviderForUser • VerifyUserAttribute • DeleteUser • DeleteUserAttributes • UpdateUserAttributes • AdminUserGlobalSignOut • GlobalSignOut • AdminRemoveUserFromGroup	사용자 및 사용자 속성을 관리하는 데 사용하는 작업입니다.	25	아니요
`UserToken` • RevokeToken	토큰 관리 작업	120	예

범주	설명	기본 할당량(RPS)	조정 가능
`UserResourceRead` • AdminGetDevice • AdminListGroupsForUser • AdminListDevices • GetDevice • ListDevices • GetUserAttributeVerificationCode • ResendConfirmationCode • AdminListUserAuthEvents	Amazon Cognito에서 저장된 디바이스 또는 그룹 멤버십과 같은 사용자 리소스 정보를 검색하는 작업입니다.	50	예

범주	설명	기본 할당량(RPS)	조정 가능
`UserResourceUpdate` • AdminForgetDevice • AdminUpdateAuthEventFeedback • AdminSetUserMFAPreference • AdminDeleteSoftwareToken • AdminSetUserSettings • AdminUpdateDeviceStatus • UpdateDeviceStatus • UpdateAuthEventFeedback • ConfirmDevice • SetUserMFAPreference • SetUserSettings • VerifySoftwareToken • AssociateSoftwareToken • ForgetDevice	저장된 디바이스 또는 그룹 멤버십과 같은 사용자에 대한 리소스 정보를 업데이트하는 작업입니다.	25	아니요

범주	설명	기본 할당량(RPS)	조정 가능
`UserList` • ListUsers • ListUsersInGroup	사용자 목록을 반환하는 작업입니다.	30	아니요
`UserPoolRead` • DescribeUserPool • ListUserPools	사용자 풀을 읽는 작업입니다.	15	아니요
`UserPoolUpdate` • CreateUserPool • UpdateUserPool • DeleteUserPool	사용자 풀을 생성, 업데이트 또는 삭제하는 작업입니다.	15	아니요

범주	설명	기본 할당량(RPS)	조정 가능
`UserPoolResourceRead` • DescribeIdentityProvider • DescribeResourceServer • DescribeUserImportJob • DescribeUserPoolDomain • GetCSVHeader • GetGroup • GetSigningCertificate • GetIdentityProviderByIdentifier • GetUserPoolMfaConfig • ListGroups • ListIdentityProviders • ListResourceServers • ListTagsForResource • ListUserImportJobs • DescribeRiskConfiguration • GetUICustomization	사용자 풀에서 그룹 또는 리소스 서버와 같은 리소스에 대한 정보를 검색하는 작업입니다.[3]	20	아니요

범주	설명	기본 할당량(RPS)	조정 가능
• DescribeManagedLoginBranding • DescribeManagedLoginBrandingByClient • ListUserPoolReplicas			

범주	설명	기본 할당량(RPS)	조정 가능
`UserPoolResourceUpdate` • AddCustomAttributes • CreateGroup • CreateIdentityProvider • CreateResourceServer • CreateUserImportJob • CreateUserPoolDomain • DeleteGroup • DeleteIdentityProvider • DeleteResourceServer • DeleteUserPoolDomain • SetUserPoolMfaConfig • StartUserImportJob • StopUserImportJob • UpdateGroup • UpdateIdentityProvider • UpdateResourceServer • UpdateUserPoolDomain	사용자 풀에서 그룹 또는 리소스 서버와 같은 리소스를 수정하는 작업입니다.[3]	15	아니요

범주	설명	기본 할당량(RPS)	조정 가능
• SetRiskConfiguration • SetUICustomization • TagResource • UntagResource • CreateManagedLoginBranding • UpdateManagedLoginBranding • DeleteManagedLoginBranding • CreateUserPoolReplica • UpdateUserPoolReplica • DeleteUserPoolReplica			
`UserPoolClientRead` • DescribeUserPoolClient • ListUserPoolClients	사용자 풀 클라이언트에 대한 정보를 검색하는 작업입니다.[3]	15	아니요

범주	설명	기본 할당량(RPS)	조정 가능
`UserPoolClientUpdate` • CreateUserPoolClient • DeleteUserPoolClient • UpdateUserPoolClient	사용자 풀 클라이언트를 생성, 업데이트 및 삭제하는 작업입니다.[3]	15	아니요
`ClientAuthentication` • `client_credentials` 토큰 엔드포인트에 대한 권한 부여 유형 요청 • GetClientToken	시스템간 요청 인증에 사용할 자격 증명을 생성하는 작업	150	아니요
`LimitManagement` • GetProvisionedLimit • UpdateProvisionedLimit	프로비저닝된 API 속도 제한을 검색하거나 업데이트하는 작업	1	아니요

[1] A `RespondToAuthChallenge` 또는 `AdminRespondToAuthChallenge` 응답(`NEW_PASSWORD_REQUIRED`의 `ChallengeName`)은 `UserAccountRecovery` 범주에 포함됩니다. 다른 모든 챌린지 응답은 `UserAuthentication` 범주에 포함됩니다.

[2] 로그인 시 각 관리형 로그인 또는 클래식 호스팅 UI 작업은 할당량에서 요청 1개를 사용합니다. 예를 들어, 로그인하고 MFA 코드를 제공하는 사용자는 요청 2개를 사용합니다. 권한 부여 코드 부여에서 토큰 사용은 `UserAuthentication` 범주의 할당량과 동일한 비율로 추가 할당량 할당이 적용됩니다.

[3] 이 범주의 모든 개별 작업에는 단일 사용자 풀에 대해 5RPS 이상의 속도로 작업이 호출되지 않도록 하는 제약 조건이 있습니다.

사용자 풀 도메인에 대한 대량 요청 비율 제한

다음 할당량은 사용자 풀 도메인에 대한 전체 요청량에 적용됩니다.

연산	설명	기본 할당량(RPS)	조정 가능
소스 IP의 요청	한 IP 주소에서 한 도메인으로의 요청 볼륨	300	아니요
앱 클라이언트에 대한 요청	하나의 도메인에 있는 하나의 앱 클라이언트 ID에 대한 요청 볼륨	300	아니요
도메인에 대한 요청	한 사용자 풀 도메인의 서비스에 대한 전체 요청량	500	아니요
JSON 웹 키 문서에 대한 요청	에 대한 요청 볼륨 AWS 계정 을 일대일`jwks.json` 로 AWS 리전	50,000	아니요

Amazon Cognito 자격 증명 풀(페더레이션 자격 증명) API연산 요청 속도 할당량

연산	설명	기본 할당량 (RPS)[1]	조정 가능	할당량 증가 자격 여부
`GetId`	자격 증명 풀에서 자격 증명 ID를 검색합니다.	25	예	계정 팀에 문의하세요.
`GetOpenIdToken`	클래식 워크플로의 자격 증명 풀에서 OpenID 토큰을 검색합니다.	200	예	계정 팀에 문의하세요.

연산	설명	기본 할당량 (RPS)[1]	조정 가능	할당량 증가 자격 여부
GetCredentialsForIdentity	향상된 워크플로의 자격 증명 풀에서 AWS 자격 증명을 검색합니다.	200	예	계정 팀에 문의하세요.
GetOpenIdTokenForDeveloperIdentity	개발자 워크플로의 자격 증명 풀에서 OpenID 토큰을 검색합니다.	50	예	계정 팀에 문의하세요.
ListIdentities	자격 증명 풀의 자격 증명 ID 목록을 검색합니다.	5	예	계정 팀에 문의하세요.
DeleteIdentities	자격 증명 풀에서 하나 이상의 등록된 자격 증명을 삭제합니다.	10	예	계정 팀에 문의하세요.
TagResource	자격 증명 풀에 태그를 적용합니다.	5	예	계정 팀에 문의하세요.
UntagResource	자격 증명 풀에서 태그를 제거합니다.	5	예	계정 팀에 문의하세요.
ListTagsForResource	자격 증명 풀에 적용된 태그 목록을 표시합니다.	10	예	계정 팀에 문의하세요.

[1] 기본 할당량은의에 있는 자격 증명 풀에 대한 최소 요청 속도 할당량입니다 AWS 리전 AWS 계정. 일부 리전에서는 RPS 할당량이 더 높을 수 있습니다.

리소스 수 및 크기에 대한 할당량

리소스 할당량은 Amazon Cognito의 리소스, 입력 필드, 기간 및 기타 기능의 최대 수 또는 크기입니다.

Service Quotas 콘솔 또는 서비스 한도 증가 양식에서 일부 리소스 할당량에 대한 조정을 요청할 수 있습니다. Service Quotas 콘솔에서 할당량을 요청하려면 Service Quotas 사용 설명서에서 할당량 증가 요청을 참조하세요. Service Quotas에서 아직 할당량을 사용할 수 없는 경우 서비스 한도 증가 양식을 사용합니다.

> **Note**
>
> 리전당 사용자 풀과 같은 AWS 계정 수준의 리소스 할당량은 각의 Amazon Cognito 리소스에 적용됩니다 AWS 리전. 예를 들어 미국 동부(버지니아 북부)에 사용자 풀 1,000개가 있고 유럽(스톡홀름)에 1,000개가 있을 수 있습니다.

다음 표에는 기본 리소스 할당량과 조정 가능 여부가 나와 있습니다.

Amazon Cognito 사용자 풀 리소스 할당량

다음 할당량은 사용자 풀에서 생성할 수 있는 최대 항목 수 또는 길이를 설명합니다.

Resource	할당량	조정 가능	최대 할당량
사용자 풀당 앱 클라이언트	1,000	예	10,000
리전당 사용자 풀	1,000	예	10,000
사용자 풀당 자격 증명 공급자	300	예	1,000
사용자 풀당 리소스 서버	100	예	300
사용자 풀당 사용자	40,000,000	예	계정 팀에 문의하세요.
사전 토큰 생성 Lambda 트리거의 변경 총합[1]	5,000	예	계정 팀에 문의하세요.

Resource	할당량	조정 가능	최대 할당량
사용자 풀당 관리형 로그인 브랜딩 스타일	20	아니요	해당 사항 없음
사용자 풀당 관리형 로그인 용어 문서	40	아니요	해당 사항 없음
사용자 풀당 사용자 지정 속성	50	아니요	해당 사항 없음
속성당 글자 수	2,048바이트	아니요	해당 사항 없음
사용자 지정 속성 이름의 글자 수	20	아니요	해당 사항 없음
암호 정책에서 필요한 최소 암호 글자 수	6~99	아니요	해당 사항 없음
에 따라 매일 전송되는 이메일 메시지 AWS 계정[2]	50	아니요	해당 사항 없음
요청자 IP 주소당 시간당 하나의 이메일 주소로 전송된 이메일 MFA 메시지	5-20	아니요	해당 사항 없음
이메일 제목의 글자 수	140	아니요	해당 사항 없음
이메일 메시지의 글자 수	20,000건	아니요	해당 사항 없음
SMS 확인 메시지의 글자 수	140	아니요	해당 사항 없음
암호의 글자 수	256	아니요	해당 사항 없음
자격 증명 공급자 이름의 글자 수	32	아니요	해당 사항 없음

Resource	할당량	조정 가능	최대 할당량
SAML 응답의 문자	100,000건	아니요	해당 사항 없음
자격 증명 공급자당 식별자	50	아니요	해당 사항 없음
사용자에 연결된 자격 증명	5	아니요	해당 사항 없음
사용자당 패스키/WebAuthn 인증자	20	아니요	해당 사항 없음
앱 클라이언트당 콜백 URL	100	아니요	해당 사항 없음
앱 클라이언트당 로그아웃 URL	100	아니요	해당 사항 없음
리소스 서버당 범위	100	아니요	해당 사항 없음
앱 클라이언트당 범위	50	아니요	해당 사항 없음
리전당 사용자 지정 도메인	4	아니요	해당 사항 없음
각 사용자가 속할 수 있는 그룹	100	아니요	해당 사항 없음
사용자 풀당 그룹	10,000	아니요	해당 사항 없음

[1] 이 할당량은 사전 토큰 생성 Lambda 트리거의 토큰에서 발생할 수 있습니다. 한 트랜잭션에서 기존 클레임 및 추가된 클레임 수와 액세스 및 자격 증명 토큰의 범위를 더한 수를 합하면 이 할당량보다 작거나 같아야 합니다. 차단된 클레임 및 범위는 이 할당량에 포함되지 않습니다.

[2] 이 할당량은 Amazon Cognito 사용자 풀에 기본 이메일 기능을 사용하는 경우에만 적용됩니다. 더 많은 이메일 전송 볼륨을 활성화하려면 Amazon SES 이메일 구성을 사용하도록 사용자 풀을 구성합니다. 이 제한은 매일 0900UTC로 재설정됩니다. 자세한 내용은 Amazon Cognito 사용자 풀에 대한 이메일 설정 섹션을 참조하세요.

Amazon Cognito 사용자 풀 세션 유효성 파라미터

다음 할당량은 사용자 풀의 인증 아티팩트 및 사용자 세션 기간 동안 사용 가능한 설정을 설명합니다.

토큰	할당량
ID 토큰	5분~1일
새로 고침 토큰	1시간~3,650일
액세스 토큰	5분~1일
호스트된 UI 세션 쿠키	1시간
인증 세션 토큰	3분~15분

Amazon Cognito 사용자 풀 코드 보안 리소스 할당량(조정 불가능)

다음 할당량은 로그인, 가입 및 암호 재설정 코드와 관련하여 사용 가능한 기간을 설명합니다.

Resource	할당량
가입 확인 코드 유효 기간	24시간
사용자 속성 확인 코드 유효 기간	24시간
다중 인증(MFA) 코드 유효 기간	3~15분
암호 찾기 코드 유효 기간	1시간
시간당 사용자별 최대 `ConfirmForgotPassword` 및 `ForgotPassword` 요청 수[1]	5~20
시간당 사용자별 최대 `ResendConfirmationCode` 요청 수	5
시간당 사용자별 최대 `ConfirmSignUp` 요청 수	15

Resource	할당량
시간당 사용자별 최대 ChangePassword 요청 수	5
시간당 사용자별 최대 GetUserAttributeVerificationCode 요청 수	5
시간당 사용자별 최대 VerifyUserAttribute 요청 수	15

[1] Amazon Cognito는 암호 업데이트 요청의 위험 요소를 평가하고 평가된 위험 수준에 맞는 할당량을 할당합니다. 자세한 내용은 암호 찾기 동작 섹션을 참조하세요.

Amazon Cognito 사용자 풀 사용자 가져오기 작업 리소스 할당량

다음 할당량은 사용자 가져오기 작업에 사용할 수 있는 리소스 및 제한에 대해 설명합니다.

Resource	할당량	조정 가능	최대 할당량
사용자 풀당 사용자 가져오기 작업	1,000	예	계정 팀에 문의하세요.
사용자 가져오기 CSV 행당 최대 문자 수	16,000	아니요	해당 사항 없음
최대 CSV 파일 크기	100MB	아니요	해당 사항 없음
CSV 파일당 최대 사용자 수	500,000	아니요	해당 사항 없음

Amazon Cognito ID 풀(페더레이션 자격 증명) 리소스 할당량

다음 할당량은 ID 풀에서 생성할 수 있는 최대 항목 수 또는 길이를 설명합니다.

Resource	할당량	조정 가능	최대 할당량
계정당 자격 증명 풀	1,000	예	해당 사항 없음

Resource	할당량	조정 가능	최대 할당량
자격 증명 풀당 Amazon Cognito 사용자 풀 공급자	50	예	1000
자격 증명 풀 이름에 대한 글자 길이	128B	아니요	해당 사항 없음
로그인 공급자 이름에 대한 글자 길이	2,048바이트	아니요	해당 사항 없음
자격 증명 풀당 자격 증명	무제한	아니요	해당 사항 없음
역할 매핑을 지정할 수 있는 자격 증명 공급자	10	아니요	해당 사항 없음
단일 list 또는 lookup 호출의 결과	60	아니요	해당 사항 없음
역할 기반 액세스 제어(RBAC) 규칙	25	아니요	해당 사항 없음

Amazon Cognito Sync 리소스 할당량

다음 할당량은 Amazon Cognito Sync에서 생성할 수 있는 최대 항목 수 또는 길이를 설명합니다.

Resource	할당량	조정 가능	최대 할당량
자격 증명별 데이터 집합	20	예	계정 팀에 문의하세요.
데이터 집합당 레코드	1,024	예	계정 팀에 문의하세요.
단일 데이터 집합의 크기	1MB	예	계정 팀에 문의하세요.

Resource	할당량	조정 가능	최대 할당량
데이터 집합 이름의 글자 수	128바이트	아니요	해당 사항 없음
요청 성공 후 대량 게시를 위한 대기 시간	24시간	아니요	N/A

사용자 풀에 대한 다중 리전 복제

다중 리전 복제(MRR)를 사용하면 추가에 복제본 사용자 풀을 생성 AWS 리전 하여 인증 인프라에 비즈니스 연속성 및 재해 복구 기능을 제공할 수 있습니다. MRR을 사용하면 리전의 리소스에 대한 연결이 끊어지더라도 등록된 사용자가 애플리케이션을 계속 인증하여 애플리케이션을 계속 사용할 수 있습니다.

MRR을 구성하면 Amazon Cognito는 공유 사용자 풀 ID로 별도의 사용자 풀을 생성합니다. 각 복제본 사용자 풀은 공유 사용자 디렉터리에 대한 인증 서비스를 호스팅합니다. 기본 사용자 풀은 암호 재설정 및 사용자 가입과 같은 관리 구성 및 쓰기 작업을 위한 신뢰할 수 있는 소스 역할을 합니다. 보조 사용자 풀은 사용자를 생성할 수 없습니다. 기본 사용자 풀에서 대부분의 설정을 상속하며 장애 조치 상태에서는 사용자 로그인 및 토큰 생성과 같은 인증 작업을 처리할 수 있습니다.

다중 리전 복제에 대해 알아야 할 사항

- 다중 리전 복제에는 별도의 추가 비용이 있으며 사용자 풀이 Essentials 또는 Plus 기능 플랜에 있어야 합니다. Lite 기능 계획을 사용하여 사용자 풀에서 MRR을 활성화할 수 없습니다.
- 복제를 활성화하기 AWS KMS 전에의 다중 리전 고객 관리형 키로 사용자 풀을 구성해야 합니다. 키는 사용자 풀 복제본 AWS 리전 이 있는 모든에서 사용할 수 있어야 합니다. 자세한 내용은 the section called “데이터 암호화” 섹션을 참조하세요.
- 리전 간에 일관된 토큰 검증을 위해 업데이트된 발급자로 사용자 풀을 구성하는 것이 좋습니다. 자세한 내용은 the section called “OIDC 발급자” 섹션을 참조하세요.
- 새 보조 사용자 풀은 `INACTIVE` 상태에서 시작됩니다. 프로덕션용으로 사용자 풀을 활성화하기 전에 리전 설정을 검토하고 구성합니다.
- 리전 구성은 복제본마다 다를 수 있습니다. 복제본에서 다음 설정을 독립적으로 구성할 수 있습니다. 다른 모든 설정은 기본 사용자 풀에 설정되며 보조 사용자 풀과 자동으로 동기화됩니다.
  - 이메일 구성
  - 위협 방지 알림을 위한 이메일 구성

  - SMS 구성
  - Lambda 트리거
  - Tags
  - 로그 내보내기 구성
  - AWS WAF 웹 ACLs
- 리전 간 데이터 복제로 인해 짧은 지연이 발생할 수 있습니다. 기본 사용자 풀은 설정 및 사용자 디렉터리 업데이트를 보조 사용자 풀과 동기화하며,이 프로세스는 최종적으로 일관됩니다.

다중 리전 복제의 제한 사항

- 가입 또는 관리자 생성을 통해 보조 사용자 풀에서 새 사용자를 생성할 수 없습니다. 페더레이션 사용자는 이전에 기본 사용자 풀에 로그인한 경우에만 장애 조치 상태의 보조 사용자 풀에 로그인할 수 있습니다.
- 사용자는 보조 사용자 풀에서 암호를 재설정하거나 프로필을 수정할 수 없습니다. 장애 조치 상태에서는 사용자 인터페이스에서 이러한 작업을 비활성화하고 상태 확인이 기본 사용자 풀에 대한 액세스를 복원한 후 사용할 수 있도록 합니다.
- 사용자 디렉터리당 추가 리전에 최대 1개의 보조 복제본을 보유할 수 있습니다. 모든 적격 사용자 풀에는 보조 복제본이 있을 수 있습니다.
- TOTP MFA는 보조 복제본에서 지원되지 않습니다. TOTP MFA가 구성된 사용자는 기본 리전의 사용자 풀이 요청을 처리할 때 인증해야 합니다.
- 잠금 전 암호 기반 인증 시도 횟수는 리전 간에 동기화되지 않습니다. 각 복제본은 실패한 인증 시도 횟수를 자체적으로 유지합니다.

다중 리전 복제 구성

다중 리전 복제를 활성화하려면 먼저 사용자 풀이 필수 또는 Plus 기능 계획 및 다중 리전 고객 관리형 KMS 키의 사전 조건을 충족하는지 확인합니다.

AWS Management Console

사용자 풀에 대한 다중 리전 복제를 구성하려면

1. Amazon Cognito 콘솔에 로그인합니다.
2. [사용자 풀(User pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 새 사용자 풀을 생성합니다.

4. 설정 탭을 선택합니다.
5. 왼쪽 탐색 메뉴에서 다중 리전 복제를 선택합니다.
6. 복제본 사용자 풀 생성을 선택합니다.
7. 리전에서 복제본 사용자 풀을 생성할 AWS 리전 을 선택합니다.
8. 구성 요약을 검토하고 복제본 생성을 선택합니다.
9. 복제본을 생성한 후 비교 테이블에서 리전 구성 설정을 검토합니다. 복제본 리전에 필요한 경우 이메일 구성, SMS 설정 또는 Lambda 트리거와 같은 리전별 설정을 구성합니다.
10. 도메인에 대해 Route 53 상태 확인을 구성하려면 도메인 서비스 메뉴로 이동하여 사용자 지정 도메인을 편집 또는 추가하고 Route 53 상태 확인 ID를 구성합니다.
11. 프로덕션 트래픽에 복제본을 사용할 준비가 되면 복제본 상태를 비활성에서 활성으로 변경합니다.

API

복제본 사용자 풀을 생성하려면 CreateUserPoolReplica 작업을 사용합니다. 다음 예시에서는 `us-west-2` 리전에서의 기본 사용자 풀에 대한 복제본을 생성합니다`us-east-1`.

```
{
 "UserPoolId": "us-east-1_EXAMPLE",
 "RegionName": "us-west-2",
 "UserPoolTags": {
    "Environment": "Production",
    "Application": "MyApp"
 }
}
```

응답에는 복제본 정보가 포함됩니다.

```
{
 "Replica": {
    "RegionName": "us-west-2",
    "UserPoolArn": "arn:aws:cognito-idp:us-west-2:111122223333:userpool/us-
east-1_EXAMPLE",
    "Status": "PENDING_CREATE",
    "Role": "SECONDARY"
 }
}
```

또한 장애 조치를 위해 도메인을 구성해야 합니다. Route 53에서 상태 확인을 설정하고 UpdateUserPoolDomain 요청의 도메인에 적용합니다.

```
{
 "CustomDomainConfig": {
    "CertificateArn": "arn:aws:acm:us-
east-1:111122223333:certificate/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
 },
 "Domain": "auth.example.com",
 "ManagedLoginVersion": 2,
 "Routing": {
    "Failover": {
       "SecondaryRegion": "us-west-2",
       "PrimaryRoute53HealthCheckId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
    }
 },
 "UserPoolId": "us-east-1_EXAMPLE"
}
```

프로덕션용으로 복제본을 활성화하려면 UpdateUserPoolReplica 작업을 사용합니다.

```
{
 "UserPoolId": "us-east-1_EXAMPLE",
 "RegionName": "us-west-2",
 "Status": "ACTIVE"
}
```

응답은 업데이트된 복제본 상태를 확인합니다.

```
{
 "Replica": {
    "RegionName": "us-west-2",
    "UserPoolArn": "arn:aws:cognito-idp:us-west-2:111122223333:userpool/us-
east-1_EXAMPLE",
    "Status": "ACTIVE",
    "Role": "SECONDARY"
 }
}
```

보조 리전에서 지원되는 API 작업

Amazon Cognito는 보조 리전에서 API 작업의 하위 집합을 지원합니다. 사용 가능한 작업은 복제본의 상태에 따라 다릅니다. `INACTIVE` 상태의 복제본은 제한된 읽기 및 구성 작업 세트를 지원합니다. `ACTIVE` 상태의 복제본은 추가 인증 및 세션 관리 작업을 지원합니다. 여기에 나열되지 않은 작업은 기본 리전에서만 사용할 수 있습니다.

비활성 보조 리전에 대한 작업

`INACTIVE` 상태의 보조 리전에 있는 복제본 사용자 풀은 다음과 같은 Amazon Cognito API 작업을 허용합니다.

- AdminGetDevice
- AdminGetUser
- AdminListDevices
- AdminListGroupsForUser
- AdminListUserAuthEvents
- AssociateWebACL
- CreateUserPoolDomain
- DeleteUserPoolDomain
- DeleteUserPoolReplica
- DescribeIdentityProvider
- DescribeManagedLoginBranding
- DescribeManagedLoginBrandingByClient
- DescribeResourceServer
- DescribeRiskConfiguration
- DescribeTerms
- DescribeUserPool
- DescribeUserPoolClient
- DescribeUserPoolDomain
- DisassociateWebACL
- GetGroup
- GetLogDeliveryConfiguration

- GetSigningCertificate
- GetUICustomization
- GetUserPoolMfaConfig
- ListIdentityProviders
- ListGroups
- ListResourceServers
- ListTagsForResource
- ListTerms
- ListUserPoolClients
- ListUserPoolClientSecrets
- ListUserPoolReplicas
- ListUserPools
- ListUsers
- ListUsersInGroup
- SetLogDeliveryConfiguration
- SetRiskConfiguration
- SetUserPoolMfaConfig
- TagResource
- UntagResource
- UpdateUserPool
- UpdateUserPoolDomain
- UpdateUserPoolReplica

ACTIVE 보조 리전에 대한 추가 작업

`ACTIVE` 상태의 보조 리전에 있는 복제본 사용자 풀은 위의 모든 작업과 다음 인증 및 세션 관리 작업을 허용합니다.

- AdminInitiateAuth
- AdminRespondToAuthChallenge
- AdminUpdateAuthEventFeedback

- AdminUserGlobalSignOut
- GetDevice
- GetIdentityProviderByIdentifier
- GetTokensFromRefreshToken
- GetUser
- GetUserAuthFactors
- GlobalSignOut
- InitiateAuth
- ListDevices
- ListWebAuthnCredentials
- RespondToAuthChallenge
- RevokeToken
- UpdateAuthEventFeedback

다중 리전 사용자 풀의 장애 조치

다중 리전 사용자 풀을 사용하면 관리형 로그인, 페더레이션 로그인 및 둘 사이의 직접 API 호출을 장애 조치할 수 있습니다 AWS 리전. 관리형 로그인 및 페더레이션 장애 조치는 사용자 풀로 구성된 사용자 지정 도메인 또는 접두사(Cognito) 도메인에서 사용할 수 있습니다. 복제본 사용자 풀을 사용하여 다른 사용자 지정 도메인을 구성할 수 없습니다.

> **Note**
>
> Amazon Cognito는 현재 AWS GovCloud(미국) 리전에서 Route 53 상태 확인을 통한 자동 장애 조치를 지원하지 않습니다. 트래픽을 in AWS GovCloud(미국)의 보조 리전으로 라우팅하는 방법에 대한 자세한 내용은 섹션을 참조하세요the section called “AWS GovCloud(미국) 리전에서의 장애 조치”.

관리형 로그인, 페더레이션 및 machine-to-machine 권한 부여를 위한 장애 조치

기본 사용자 풀에 사용자 지정 도메인 또는 접두사 도메인이 있는 경우 장애 조치를 사용할 수 있습니다. 사용자 풀 도메인은 권한 부여 및 토큰 엔드포인트를 포함한 OAuth 2.0 리소스를 제공하고 OIDC, SAML 및 소셜 공급자를 포함한 타사 페더레이션 공급자의 IdP 응답을 처리합니다.

장애 조치를 활성화하려면 Route 53에서 상태 확인을 설정하고 도메인에 `Routing` 필드를 설정합니다. 무엇이 정상 상태 또는 비정상 상태를 트리거하는지 결정합니다. 상태 확인이 비정상 상태인 경우 Amazon Cognito는 보조 복제본 사용자 풀에서 관리형 로그인 페이지 및 인증 작업을 제공합니다. 상태 확인이 정상 상태가 되면 Amazon Cognito는 트래픽을 기본 복제본으로 다시 라우팅하기 시작합니다.

사용자 지정 도메인의 경우 DNS 레코드는 Route 53 또는 타사 DNS 공급자를 사용할 수 있습니다. DNS 공급자에 대상 별칭, 즉 CloudFront 배포를 가리키는 유효한 CNAME 레코드가 있는지 확인합니다. Amazon Cognito 콘솔의 도메인 페이지에서 별칭 대상을 찾을 수 있습니다.

콘솔에서 상태 확인 ID를 업데이트하려면

1. Amazon Cognito 콘솔에 로그인합니다.
2. 사용자 풀을 선택한 다음 사용자 풀을 선택합니다.
3. 메뉴에서 브랜딩에서 도메인을 선택합니다.
4. 사용자 지정 도메인 섹션에서 편집 옵션을 선택하고 다중 리전 장애 조치 편집을 선택합니다.
5. 다중 리전 장애 조치 활성화 옵션을 전환합니다.
6. 사용 가능한 상태 확인에서 Route 53 상태 확인 ID를 선택합니다.
7. 변경 사항 저장을 선택합니다.

Amazon Cognito APIs 및 SDKs에 대한 장애 조치

Amazon Cognito APIs 또는 SDKs를 사용하는 경우 사용자 지정 도메인은 사용되지 않으며 애플리케이션은 인증 및 기타 API 호출을 처리하기 위해 Amazon Cognito 서비스 리전 엔드포인트로 트래픽을 라우팅할 책임이 있습니다.

단일 페이지 애플리케이션(SPA) 또는 모바일 앱과 같은 퍼블릭 클라이언트를 사용하는 애플리케이션 프런트엔드만 있는 경우 그에 따라 API 호출을 라우팅하려면 애플리케이션이 동적이어야 합니다. Amazon Cognito를 사용한 리전 인증을 결정하는 데 도움이 되는 서버리스 애플리케이션 백엔드를 고려해 보세요.

백엔드가 있는 애플리케이션이 있는 경우 인증할 사용자 풀을 결정하는 로직은 여기에서 확인할 수 있습니다.

관리형 로그인 엔드포인트와 APIs 모두 사용하는 경우 동일한 Route 53 상태 확인을 사용하여 애플리케이션이 Amazon Cognito API 호출을 보내는 리전을 결정합니다.

AWS GovCloud(미국) 리전에서의 장애 조치

Amazon Cognito는 현재 AWS GovCloud(미국) 리전에서 Route 53 상태 확인을 통한 자동 장애 조치를 지원하지 않습니다. 사용자 풀 도메인에는 `Routing` 필드를 설정할 수 없으며 사용자 지정 도메인은 사용할 수 없습니다. 보조 리전에서 관리형 로그인, 페더레이션 및 OAuth 2.0 엔드포인트를 계속 제공하려면 각 복제본 사용자 풀에 대한 접두사 도메인을 구성하고 도메인 간에 트래픽을 직접 라우팅합니다.

장애 조치가 필요하기 전에 다음 단계를 완료하세요. 시작하기 전에 보조 리전에 복제본 사용자 풀을 생성합니다.

AWS GovCloud(미국) 리전에서 장애 조치를 준비하려면

1. 보조 리전에서 관리형 로그인을 사용하여 복제본 사용자 풀의 접두사 도메인을 생성합니다. `CreateUserPoolDomain` 요청에서를 `ManagedLoginVersion`로 설정2하거나 Amazon Cognito 콘솔의 도메인 메뉴에서 관리형 로그인을 선택합니다. 복제본 사용자 풀은 클래식 호스팅 UI가 있는 도메인을 지원하지 않습니다. 각 도메인에는 리전별 URL이 있습니다.
2. 사용자 풀에 페더레이션 ID 제공업체(IdPs가 있는 경우 보조 도메인의 리디렉션 URL을 각 IdP에 추가합니다. OIDC 및 소셜 IdPs 경우를 허용된 리디렉션 URI`https://`*`secondary-domain`*`/oauth2/idpresponse`로 추가합니다. SAML IdPs 경우를 어설션 소비자 서비스(ACS) URL`https://`*`secondary-domain`*`/saml2/idpresponse`로 추가합니다. 이 단계를 건너뛰면 트래픽을 보조 도메인으로 전환한 후 페더레이션 로그인이 실패합니다.
3. 트래픽을 제공하는 리전의 도메인에 있는 권한 부여, 토큰, 로그아웃 및 기타 OAuth 2.0 엔드포인트로 요청을 보내도록 애플리케이션을 구성합니다. 클라이언트 자격 증명 부여와 함께 토큰을 요청하는 M2M(Machinemachine-to-machine) 클라이언트를 포함합니다.
4. 트래픽을 보조 리전으로 보내기 전에 복제본 사용자 풀 상태를 로 변경합니다`ACTIVE`.

트래픽을 보조 도메인으로 라우팅할 때는 다음 사항을 고려하세요.

- 패스키 - 신뢰 당사자 ID가 도메인마다 다르기 때문에 기본 도메인에 등록된 사용자가 보조 도메인에서 작업하지 않는 패스키입니다. 보조 도메인에서 관리형 로그인에 로그인하려면 사용자가 다른 로그인 방법을 사용해야 합니다. 이는 SDKs를 사용하고 자체 도메인을 신뢰 당사자 ID로 설정하는 애플리케이션에는 영향을 주지 않습니다.
- 장애 복구 - Amazon Cognito는 트래픽을 기본 리전으로 다시 라우팅하지 않습니다. 기본 리전을 다시 사용할 수 있게 되면 애플리케이션을 업데이트하여 트래픽을 기본 도메인으로 전송합니다.

API 및 SDK 트래픽 라우팅에 대한 자세한 내용은 섹션을 참조하세요the section called “Amazon Cognito APIs 및 SDKs에 대한 장애 조치”.

Amazon Cognito 리소스 태깅

태그는 사용자 또는가 리소스에 AWS 할당하는 메타데이터 레이블입니다 AWS . 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 구성합니다. 많은 AWS 서비스가 태그 지정을 지원하므로 다른 서비스의 리소스에 동일한 태그를 할당할 수 있습니다. 이렇게 하면 어떤 리소스가 관련되어 있는지 알 수 있습니다. 예를 들어 Amazon DynamoDB 테이블에 할당한 것과 동일한 태그를 Amazon Cognito 사용자 풀에 할당할 수 있습니다.
- AWS 비용을 추적합니다. AWS 결제 및 비용 관리 대시보드에서 이러한 태그를 활성화할 수 있습니다. AWS 는 비용 할당 태그를 사용하여 비용을 분류하고 월별 비용 할당 보고서를 제공합니다. 자세한 내용은 AWS Billing 사용 설명서의 비용 할당 태그 사용을 참조하세요.
- 할당된 태그를 기반으로 리소스에 대한 액세스를 제어합니다. AWS Identity and Access Management (IAM) 정책 조건에서 태그 키와 값을 지정하여 액세스를 제어할 수 있습니다. 예를 들어 사용자 풀에 값이 해당 사용자의 이름인 `owner` 태그가 있을 때만 사용자가 사용자 풀을 업데이트하도록 허용할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 태그를 사용한 액세스 제어를 참조하십시오.

AWS Command Line Interface 또는 Amazon Cognito API를 사용하여 사용자 및 자격 증명 풀 모두에 대한 태그를 추가, 편집 또는 삭제할 수 있습니다. Amazon Cognito 콘솔을 사용하여 사용자 풀에 대한 태그를 관리할 수도 있습니다.

태그 사용 방법에 대한 팁은 AWS Answers 블로그의 AWS 태깅 전략 게시글을 참조하세요.

다음 섹션에서는 Amazon Cognito의 태그에 대한 추가 정보를 제공합니다.

Amazon Cognito에서 지원되는 리소스

Amazon Cognito의 다음 리소스는 태깅을 지원합니다.

- 사용자 풀
- ID 풀

태그 제한 사항

Amazon Cognito 리소스에 대한 태그에 적용되는 제한 사항은 다음과 같습니다.

- 리소스에 할당할 수 있는 최대 태그 수 - 50개
- 최대 키 길이 - 유니코드 128자
- 최대 값 길이 - 유니코드 256자
- 키 및 값에 사용할 수 있는 문자 - a-z, A-Z, 0-9, 공백, _ . : / = + - @ 문자
- 키와 값은 대/소문자를 구분합니다.
- 키 접두사로 `aws:`를 사용하지 마세요. AWS 전용입니다.

Amazon Cognito 콘솔을 사용하여 태그 관리

Amazon Cognito 콘솔을 사용하여 사용자 풀에 할당된 태그를 관리할 수 있습니다.

사용자 풀에 태그를 추가하려면

1. Amazon Cognito 콘솔로 이동합니다. 메시지가 표시되면 자격 AWS 증명을 입력합니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. 설정 메뉴를 선택하고 태그 탭을 찾습니다.
5. [태그 추가(Add tags)]를 선택하여 첫 번째 태그를 추가합니다. 이전에 이 사용자 풀에 태그를 할당한 경우 [태그 관리(Manage tags)]에서 [다른 태그 추가(Add another)]를 선택합니다.
6. 태그 키(Tag Key)와 태그 값(Tag Value)의 값을 지정합니다.
7. 추가하려는 각 태그에 대해 [다른 태그 추가(Add another)]를 선택합니다.
8. 태그 추가가 완료되면 변경 사항 저장(Save changes)을 선택합니다.

ID 풀에 태그를 지정하려면 ID 풀 메뉴로 이동하여 ID 풀을 선택하거나 생성합니다. ID 풀 속성 탭에서 태그를 찾습니다. 태그 추가를 선택합니다.

AWS CLI 예제

는 Amazon Cognito 사용자 풀 및 자격 증명 풀에 할당하는 태그를 관리하는 데 도움이 되는 명령을 AWS CLI 제공합니다.

태그 할당

다음 명령을 사용하여 기존 사용자 풀 및 자격 증명 풀에 태그를 할당합니다.

Example사용자 풀에 대한 tag-resource 명령

`cognito-idp` 명령 세트에서 `tag-resource`를 사용하여 사용자 풀에 태그를 할당합니다.

```
$ aws cognito-idp tag-resource \
> --resource-arn user-pool-arn \
> --tags Stage=Test
```

이 명령에는 다음 파라미터가 포함되어 있습니다.

- `resource-arn` - 태그를 적용할 사용자 풀의 Amazon 리소스 이름(ARN)입니다. ARN을 조회하려면 Amazon Cognito 콘솔에서 사용자 풀을 선택하고 일반 설정(General settings) 탭에서 풀 ARN(Pool ARN) 값을 확인합니다.
- `tags` - 태그의 키-값 페어(*key*=*value* 포맷)입니다.

여러 태그를 한 번에 할당하려면 쉼표로 구분된 목록으로 지정합니다.

```
$ aws cognito-idp tag-resource \
> --resource-arn user-pool-arn \
> --tags Stage=Test,CostCenter=80432,Owner=SysEng
```

Example자격 증명 풀에 대한 tag-resource 명령

`cognito-identity` 명령 세트에서 `tag-resource`를 사용하여 자격 증명 풀에 태그를 할당합니다.

```
$ aws cognito-identity tag-resource \
> --resource-arn identity-pool-arn \
> --tags Stage=Test
```

이 명령에는 다음 파라미터가 포함되어 있습니다.

- `resource-arn` - 태그를 적용할 자격 증명 풀의 Amazon 리소스 이름(ARN)입니다. ARN을 조회하려면 Amazon Cognito 콘솔에서 자격 증명 풀을 선택하고 [자격 증명 풀 편집(Edit identity pool)]을 선택합니다. 그런 다음 자격 증명 풀 ID(Identity pool ID)에서 ARN 표시(Show ARN)를 선택합니다.
- `tags` - 태그의 키-값 페어(*key*=*value* 포맷)입니다.

여러 태그를 한 번에 할당하려면 쉼표로 구분된 목록으로 지정합니다.

```
$ aws cognito-identity tag-resource \
> --resource-arn identity-pool-arn \
> --tags Stage=Test,CostCenter=80432,Owner=SysEng
```

태그 보기

다음 명령을 사용하여 사용자 풀 및 자격 증명 풀에 할당한 태그를 봅니다.

Example사용자 풀에 대한 list-tags-for-resource 명령

`cognito-idp` 명령 세트에서 `list-tags-for-resource`를 사용하여 사용자 풀에 할당한 태그를 봅니다.

```
$ aws cognito-idp list-tags-for-resource --resource-arn user-pool-arn
```

Example자격 증명 풀에 대한 list-tags-for-resource 명령

`cognito-identity` 명령 세트에서 `list-tags-for-resource`를 사용하여 자격 증명 풀에 할당한 태그를 봅니다.

```
$ aws cognito-identity list-tags-for-resource --resource-arn identity-pool-arn
```

태그 제거

다음 명령을 사용하여 사용자 풀 및 자격 증명 풀에서 태그를 제거합니다.

Example사용자 풀에 대한 untag-resource 명령

`cognito-idp` 명령 세트에서 `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 태그 키를 하나 이상 지정합니다. 태그 값을 포함해서는 안 됩니다. 키를 공백으로 구분합니다.

Example자격 증명 풀에 대한 untag-resource 명령

`cognito-identity` 명령 세트에서 `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 태그 키를 하나 이상 지정합니다. 태그 값을 포함해서는 안 됩니다.

> **Important**
>
> 사용자 또는 자격 증명 풀을 삭제한 후에도 삭제된 풀과 관련한 태그가 삭제 후 최대 30일 동안 콘솔 또는 API 호출에 계속 표시될 수 있습니다.

리소스를 생성할 때 태그 적용

다음 명령을 사용하여 사용자 풀 또는 자격 증명 풀을 생성할 때 태그를 할당합니다.

Example태그가 있는 create-user-pool 명령

`create-user-pool` 명령을 사용하여 사용자 풀을 생성할 때 `--user-pool-tags` 파라미터로 태그를 지정할 수 있습니다.

```
$ aws cognito-idp create-user-pool \
> --pool-name user-pool-name \
> --user-pool-tags Stage=Test,CostCenter=80432,Owner=SysEng
```

태그의 키-값 페어는 *key*=*value* 포맷이어야 합니다. 여러 태그를 추가하는 경우 태그를 쉼표로 구분된 목록으로 지정합니다.

Example태그가 있는 create-identity-pool 명령

`create-identity-pool` 명령을 사용하여 자격 증명 풀을 생성할 때 `--identity-pool-tags` 파라미터로 태그를 지정할 수 있습니다.

```
$ aws cognito-identity create-identity-pool \
> --identity-pool-name identity-pool-name \
> --allow-unauthenticated-identities \
```

```
> --identity-pool-tags Stage=Test,CostCenter=80432,Owner=SysEng
```

태그의 키-값 페어는 *key*=*value* 포맷이어야 합니다. 여러 태그를 추가하는 경우 태그를 쉼표로 구분된 목록으로 지정합니다.

Amazon Cognito API를 사용하여 태그 관리

Amazon Cognito API에서 다음 작업을 사용하여 사용자 풀 및 자격 증명 풀의 태그를 관리할 수 있습니다.

사용자 풀 태그에 대한 API 작업

다음 API 작업을 사용하여 사용자 풀에 대한 태그를 할당하고 보고 제거합니다.

- TagResource
- ListTagsForResource
- UntagResource
- CreateUserPool

자격 증명 풀 태그에 대한 API 작업

다음 API 작업을 사용하여 자격 증명 풀에 대한 태그를 할당하고 보고 제거합니다.

- TagResource
- ListTagsForResource
- UntagResource
- CreateIdentityPool

CloudFormation 권한 태그 지정

를 사용하여 Amazon Cognito 사용자 풀 또는 자격 증명 풀 리소스를 생성하거나 업데이트할 때 AWS CloudFormation CloudFormation 는 스택 수준 태그와 시스템 수준 태그를 리소스에 전파할 수 있습니다. 이 태그 전파를 허용하려면 스택을 배포하는 IAM 보안 주체에 리소스 생성 권한 외에도 태그 지정 권한이 있어야 합니다.

예를 들어를 사용하여 사용자 풀을 생성하려면 호출`cognito-idp:CreateUserPool`자에게 다음 권한도 있어야 합니다.

- cognito-idp:TagResource
- cognito-idp:UntagResource
- cognito-idp:ListTagsForResource

마찬가지로를 사용하여 자격 증명 풀cognito-identity:CreateIdentityPool을 생성하려면 호출자에게 다음 권한도 있어야 합니다.

- cognito-identity:TagResource
- cognito-identity:UntagResource
- cognito-identity:ListTagsForResource

이러한 권한이 없으면가 리소스에 태그를 적용하려고 CloudFormation 할 때 스택 생성 또는 업데이트가 실패할 수 있습니다. 스택 작업 중에 태그 지정과 관련된 액세스 거부 오류가 표시되면 IAM 정책에 이러한 권한이 포함되어 있는지 확인합니다.

Amazon Cognito 문제 해결

이 장에서는 Amazon Cognito로 작업하는 동안 발생할 수 있는 일반적인 문제에 대한 솔루션을 제공합니다. Amazon Cognito 구현은 인증 흐름, 사용자 풀 구성 및 ID 페더레이션 설정 전반에 걸쳐 다양한 문제에 직면할 수 있습니다. 새 애플리케이션을 개발하든 기존 애플리케이션을 유지하든 이 문제 해결 안내서는 일반적인 문제를 신속하게 식별하고 해결하는 데 도움이 됩니다.

사용자 지정 도메인 구성 오류

Amazon Cognito에서 사용자 지정 도메인 이름을 구성할 때 오류 메시지가 표시될 수 있습니다. 일반적인 오류에는 검증 문제, 인증서 문제 또는 도메인 충돌이 포함됩니다.

Custom domain is not a valid subdomain

이 오류는 상위 도메인의 DNS 확인 문제를 나타냅니다. Amazon Cognito는 최상위 도메인을 지원하지 않으며 검증을 위해 상위 도메인에 DNS A 레코드가 있어야 합니다.

문제

이 오류는 상위 도메인의 DNS 확인 문제를 나타냅니다. Amazon Cognito는 최상위 도메인을 지원하지 않으며 검증을 위해 상위 도메인에 DNS A 레코드가 있어야 합니다.

Solution

- 상위 도메인에 대한 레코드 생성: 사용자 지정 도메인의 상위 도메인에 대한 DNS 구성에서 A 레코드를 생성해야 합니다.
  - 예: 사용자 지정 도메인이 `auth.xyz.yourdomain.com`인 경우 상위 도메인은 `xyz.yourdomain.com`입니다. `xyz.yourdomain.com`을 사용자 지정 도메인으로 구성하려는 경우 상위 항목은 `yourdomain.com`입니다.
  - 상위 도메인은 유효한 IP 주소로 확인되어야 합니다. 실제 IP 주소를 가리키지 않는 경우 `8.8.8.8`과 같은 더미 IP 주소를 사용할 수 있습니다.
- DNS 전파 확인(선택 사항이지만 권장됨): DNS 공급자가 변경 사항을 전파했는지 확인하려면 `dig` 명령을 실행할 수 있습니다.
  - `auth.xyz.yourdomain.com`을 사용자 지정 도메인으로 사용하는 경우: `dig A xyz.yourdomain.com +short`
  - `xyz.yourdomain.com`을 사용자 지정 도메인으로 사용하는 경우: `dig A yourdomain.com +short`
  - 명령은 구성한 IP 주소를 반환해야 합니다. 그렇지 않으면 변경 사항이 완전히 전파될 때까지 기다립니다.
- 상위 도메인 A 레코드 제거: Amazon Cognito에서 사용자 지정 도메인을 성공적으로 생성한 후 더미 도메인인 경우 상위 도메인에 대해 생성한 A 레코드를 제거할 수 있습니다.

자세한 내용은 호스팅 UI에 자체 도메인 사용을 참조하세요.

`Domain already associated with another user pool`

사용자 지정 도메인 이름은 모든 AWS 계정 및 리전에서 고유해야 합니다.

문제

사용자 지정 도메인 이름은 모든 AWS 계정 및 리전에서 고유해야 합니다.

Solution

- 새 사용자 풀에 도메인 이름을 사용하려면 현재 연결된 사용자 풀에서 사용자 지정 도메인을 삭제해야 합니다.
- 삭제 후 대기: 사용자 지정 도메인이 첫 번째 사용자 풀에서 완전히 삭제되는 데 시간이 걸립니다. 삭제 직후 동일한 이름으로 새 사용자 지정 도메인을 생성해도 여전히 이 오류가 발생할 수 있습니다. 몇 분 기다렸다가 다시 시도합니다.

One or more of the CNAMEs that you provided are already associated with a different resource

사용자 지정 도메인을 생성하면 Amazon Cognito가 AWS관리형 Amazon CloudFront 배포를 생성합니다. 도메인 이름은 하나의 Amazon CloudFront 배포에만 사용할 수 있습니다. 이 오류는 도메인 이름이 이미 다른 Amazon CloudFront 배포의 대체 도메인으로 사용 중인 경우에 발생합니다.

문제

사용자 지정 도메인을 생성하면 Amazon Cognito가 AWS관리형 Amazon CloudFront 배포를 생성합니다. 도메인 이름은 하나의 Amazon CloudFront 배포에만 사용할 수 있습니다. 이 오류는 도메인 이름이 이미 다른 Amazon CloudFront 배포의 대체 도메인으로 사용 중인 경우에 발생합니다.

Solution

- 옵션 1: Amazon Cognito 사용자 지정 도메인에 다른 도메인 이름을 사용합니다.
- 옵션 2: Amazon Cognito의 도메인 이름을 사용하는 경우 다른 Amazon CloudFront 배포판과 함께 사용하지 마세요.

The specified SSL certificate doesn't exist

Amazon Cognito는 사용자 풀의 리전에 AWS 리전관계없이 AWS Certificate Manager (ACM) 인증서가 `us-east-1` (버지니아 북부)에 있어야 하는 Amazon CloudFront를 사용합니다.

문제

Amazon Cognito는 사용자 풀의 리전에 AWS 리전관계없이 AWS Certificate Manager (ACM) 인증서가 `us-east-1` (버지니아 북부)에 있어야 하는 Amazon CloudFront를 사용합니다.

Solution

- 인증서 리전 확인: ACM 인증서가 `us-east-1` 리전에 있는지 확인합니다.
- 인증서 유효성 확인: 선택한 인증서가 만료되지 않았는지 확인합니다.
- 가져온 인증서: 인증서를 ACM으로 가져온 경우 다음을 확인합니다.
  - 공인 인증 기관에서 발급한 것입니다.
  - 여기에는 올바른 인증서 체인이 포함됩니다.
- AWS KMS 정책 확인: 도메인을 생성하는 IAM 사용자 또는 역할에 대한 AWS Key Management Service (AWS KMS) 정책의 명시적 deny 문으로 인해이 오류가 발생할 수 있습니다. 특히 `kms:DescribeKey`, `kms:CreateGrant` 또는 `kms:*` 작업에 대한 명시적 거부가 있는지 확인합니다.

`Invalid refresh token` 오류

문제

새로 고침 토큰을 사용하여 `AdminInitiateAuth` 또는 `InitiateAuth` API 작업을 사용하여 Amazon Cognito 사용자 풀에서 새 액세스 및 ID 토큰을 가져오려고 할 때 `Invalid refresh token` 오류가 발생합니다.

Solution

사용자 풀 구성 및 API 사용량에 따라 다음 문제 해결 단계를 구현합니다.

- 일관된 앱 클라이언트 ID 보장: 토큰 새로 고침을 위해 `AdminInitiateAuth` 또는 `InitiateAuth` API를 호출할 때 새로 고침 토큰을 생성한 초기 인증 중에 사용된 것과 정확히 동일한 앱 클라이언트 ID를 사용해야 합니다.
- 디바이스 확인: 사용자 풀에서 디바이스 추적이 활성화되었지만 사용자의 디바이스가 확인되지 않은 경우 먼저 ConfirmDevice API를 호출해야 합니다. 사용자가 디바이스를 확인한 후 새로 고침 토큰을 교환할 수 있습니다.
- 새로 고침 요청에 디바이스 키 포함: 디바이스 추적이 활성화된 경우 `REFRESH_TOKEN_AUTH` 흐름을 사용할 때 고유한 디바이스 키를 `AuthParameter`로 포함합니다.

```
{
  "AuthFlow": "REFRESH_TOKEN_AUTH",
  "AuthParameters": {
    "REFRESH_TOKEN": "example_refresh_token",
    "SECRET_HASH": "example_secret_hash", // Required if your app client uses a
 client secret
    "DEVICE_KEY": "example_device_key"
  }
}
```

- 디바이스 추적에 **`USER_SRP_AUTH`** 사용: 디바이스 추적을 사용하는 경우 초기 인증 흐름은 `USER_SRP_AUTH`여야 합니다.

자세한 내용은 사용자 풀의 사용자 디바이스 작업 섹션을 참조하세요.

페더레이션의 잘못된 SAML 응답 오류

SAML 2.0을 사용하여 Amazon Cognito에 페더레이션하려고 하면 사용자에게 다양한 `Invalid SAML response` 및 유사한 오류가 발생합니다. 이러한 오류는 속성 매핑 문제, 인증서 문제 또는 구성 불일치로 인해 발생할 수 있습니다.

Invalid user attributes: Required attribute

문제

사용자가 사용자 풀에 필요한 속성 값을 누락했거나 IdP가 변경 불가능한 속성을 제거하거나 업데이트하려고 합니다.

Solution

- 사용자 풀 구성에 정의된 필수 속성을 확인합니다.
- 브라우저에서 네트워크 캡처 도구를 사용하여 SAML 응답을 검색합니다. URL 및 base64 디코딩을 수행해야 할 수 있습니다. 속성이 SAML 어설션에 있는지 확인합니다.
- ID 제공업체에 로그인하고 Amazon Cognito로 보내는 속성을 검토합니다. IdP가 올바른 이름을 사용하여 필요한 속성을 전송하도록 구성되어 있는지 확인합니다.
- 변경할 수 없는 속성의 경우 AWS CLI 명령을 실행하여 식별합니다`aws cognito-idp describe-user-pool --user-pool-id USER-POOL-ID --query 'UserPool.SchemaAttributes[?Mutable==`false`].Name'`.
- IdP의 SAML 속성 매핑에서 변경 불가능한 Amazon Cognito 속성을 대상으로 하는 매핑을 삭제합니다. 또는 대상 속성을 변경 가능한 다른 속성으로 업데이트합니다.

Invalid SAML response received: SAML Response signature is invalid

문제

IdP가 SAML 서명 인증서를 업데이트하여 SAML 응답의 인증서와 Amazon Cognito에 저장된 메타데이터 파일이 일치하지 않습니다.

Solution

1. IdP에서 최신 메타데이터 파일을 다운로드합니다.
2. Amazon Cognito 콘솔에서 사용자 풀의 소셜 및 외부 공급자로 이동하여 SAML 공급자를 편집하고 기존 메타데이터 파일을 새로 다운로드한 파일로 바꿉니다.

Audience restriction 또는 Application with identifier not found

문제

IdP에 잘못된 엔터티 ID가 구성되었거나 어설션이 다른 사용자 풀의 URN(Uniform Resource Name)을 사용합니다.

Solution

1. 콘솔의 개요 섹션에서 Amazon Cognito 사용자 풀 ID를 가져옵니다.
2. IdP의 관리 콘솔에서 사용자 풀의 SAML 애플리케이션에서 엔터티 ID를 업데이트합니다. `urn:amazon:cognito:sp:`*`USER_POOL_ID`* 형식과 일치하도록 객체 ID를 구성합니다. USER_POOL_ID을 전 단계의 사용자 풀 ID로 바꿉니다.

An error was encountered with the requested page

문제

IdP에 등록된 어설션 소비자 서비스(ACS) URL이 잘못 구성되었거나 IdP가 필요한 POST 바인딩을 사용하여 SAML 응답을 전송하지 않습니다.

Solution

- IdP의 관리 콘솔에서 올바른 ACS URL 형식으로 애플리케이션을 업데이트하여 HTTP POST 바인딩을 사용하는지 확인합니다.
- 기본 도메인 형식: `https://cognito-idp.`*`Region`*`.amazonaws.com/`*`your user pool ID`*`/saml2/idpresponse`
- 사용자 지정 도메인 형식: `https://`*`auth.example.com`*`/saml2/idpresponse`

Invalid relayState from identity provider

문제

`RelayState` 파라미터가 누락되었거나 유효하지 않거나 IdP와 Amazon Cognito 간에 URL이 일치하지 않습니다.

Solution

- 서비스 공급자 시작(SP 시작) 흐름의 경우: 항상 사용자 풀 `/oauth2/authorize` 엔드포인트에서 인증을 시작합니다. 사용자가 Amazon Cognito를 처음 방문할 때 `RelayState` 파라미터를 반환하지 않는 SAML 어설션은 유효한 SP 시작 요청이 아닙니다.

- ID 제공업체 시작(IdP 시작) 흐름의 경우: IdP에는 필수 형식 `redirect_uri=REDIRECT_URI&state=STATE`를 사용하여 `/saml2/idpresponse` 엔드포인트에 대한 SAML 어설션과 함께 `RelayState` 파라미터가 포함되어야 합니다.

자세한 내용은 사용자 풀에서 SAML ID 제공업체 사용 섹션을 참조하세요.

관리형 로그인 사용자는 MFA 요소를 선택할 수 없습니다.

문제

사용자는 관리형 로그인을 통해 로그인할 때 선호하는 MFA 방법을 선택할 수 없거나 예상한 MFA 방법을 입력하라는 메시지가 표시되지 않습니다.

Solution

Amazon Cognito는 특정 로직을 따라 사용자 풀 설정, 사용자 속성 및 계정 복구 구성을 기반으로 사용자가 사용할 수 있는 MFA 요소를 결정합니다. 예를 들어 이메일이 기본 계정 복구 방법으로 구성된 경우 사용자는 이메일 MFA를 사용할 수 없습니다.

기본 MFA 팩터 선택을 재정의하려면 다음 방법 중 하나를 구현할 수 있습니다.

- 퍼블릭 애플리케이션의 경우: 유효한 액세스 토큰과 함께 SetUserMFAPreference를 사용하여 사용자가 자신의 MFA 기본 설정을 설정하도록 허용
- 관리자 관리형 애플리케이션의 경우: 자격 AWS 증명과 함께 AdminSetUserMFAPreference를 사용하여 사용자 MFA 기본 설정을 구성합니다.

두 작업을 모두 사용하면 개별 사용자에 대해 SMS, 이메일 및 TOTP MFA 메서드를 활성화 또는 비활성화하고 원하는 대로 하나의 메서드를 설정할 수 있습니다.

자세한 내용은 사용자 풀에 MFA 추가 섹션을 참조하세요.

암호 없는 사용자 및 패스키 사용자는 MFA를 사용할 수 없습니다.

문제

일회용 암호(OTP) 인증 방법으로 로그인하는 사용자는 멀티 팩터 인증을 추가하거나 사용할 수 없습니다.

Solution

OTP 로그인 흐름은 MFA를 지원하지 않습니다. 그러나 사용자 확인을 통한 패스키 인증은 MFA 요구 사항을 충족할 수 있습니다. 패스키`FactorConfiguration`가 다중 인증으로 계산되도록 허

용WebAuthnConfiguration하려면 사용자 풀MULTI_FACTOR_WITH_USER_VERIFICATION에서를 로 설정합니다.

자세한 내용은 사용자 풀로 인증을 참조하세요.

이메일/SMS를 통해 암호 재설정 코드를 수신할 수 없음

문제

사용자는 암호 찾기 워크플로 중에 이메일 또는 SMS를 통해 확인 코드를 받을 수 없습니다.

Solution

확인 코드가 수신되지 않는 이유는 다양합니다. 문제를 해결하려면 다음 체크리스트를 따르세요.

- 사용자의 이메일 스팸 및 정크 폴더를 확인합니다.
- 사용자가 사용자 풀에 있는지 확인합니다.
- 사용자의 상태가 FORCE_CHANGE_PASSWORD가 아닌지 확인합니다. 이 경우 관리자가 사용자를 생성했으며 Amazon Cognito는 임시 암호로 로그인할 때 암호를 설정하라는 메시지를 표시합니다.
- 사용자에게 확인된 이메일 또는 전화 번호 속성이 있는지 확인합니다.
- SMS 메시징에 대한 계정 사용 한도 확인
- Amazon Simple Email Service(Amazon SES) 메시지 전송 할당량을 확인합니다.
- SMS와 Amazon SES(사용자 풀이 Amazon SES로 이메일을 전송하도록 구성된 경우)가 모두 샌드박스에서 이동되었는지 확인하세요. 샌드박스에서 이동하지 않은 경우 Amazon SES에서 확인되지 않았거나 암호 재설정 코드를 수신할 수 없는 이메일 주소 또는 전화번호 AWS 최종 사용자 메시징 SMS 입니다.
- 사용자에게 활성 MFA 요소가 있는 경우 동일한 요소로 메시지를 생성하려고 하지 않는지 확인합니다. 예를 들어 이메일 MFA가 활성화된 사용자는 이메일로 암호 재설정 코드를 전송할 수 없습니다.

자세한 내용은 Amazon Cognito 사용자 풀에 대한 이메일 설정 및 Amazon Cognito 사용자 풀의 SMS 메시지 설정 섹션을 참조하세요.

확인되지 않은 복구 속성으로 암호 재설정이 실패합니다. **Could not reset password for the account, please contact support or try again**

문제

확인되지 않은 복구 방법 또는 MFA 구성 충돌로 인해 암호를 재설정하려고 하면 사용자에게이 오류가 발생합니다. 오류는 사용자의 이메일 또는 전화 번호 속성이 확인되지 않거나 MFA 설정으로 인해 구성된 복구 방법을 사용할 수 없는 경우에 발생합니다.

Solution

사용자 풀의 계정 복구 구성과 영향을 받는 사용자의 확인 상태를 검토합니다.

- 복구 설정 확인: 사용자 풀에서 로그인 > 사용자 계정 복구로 이동합니다. 셀프 서비스 계정 복구가 활성화되어 있는지 확인하고 복구 메시지 전송 방법 설정을 검토합니다.
- 사용자 속성 확인: 사용자에게 복구 방법 구성과 일치하는 확인된 이메일 주소 또는 전화번호가 있는지 확인합니다. 콘솔에서 사용자 프로필로 이동하여 사용자 속성 > 편집을 선택하고 해당 속성을 확인된 것으로 표시합니다.
- MFA 충돌 해결: 기본 MFA 메서드가 이메일인 사용자는 이메일로 암호 재설정 코드를 수신할 수 없으며 SMS MFA 사용자는 SMS로 코드를 수신할 수 없습니다. 복구 메시지 전송 방법을 업데이트하여 사용 가능한 경우 SMS, 사용 가능한 경우 이메일 또는 이메일, 사용 가능한 경우 SMS와 같은 대체 옵션을 제공합니다.
- 관리 확인: API 작업 AdminUpdateUserAttributes를 사용하여 콘솔 액세스를 사용할 수 없는 경우 사용자 속성을 프로그래밍 방식으로 확인합니다.

자세한 내용은 암호, 계정 복구 및 암호 정책 섹션을 참조하세요.

SECRET_HASH 오류

문제

클라이언트 보안 암호가 있는 앱 클라이언트에 대한 인증 API 요청은 `An error occurred (NotAuthorizedException) when calling the ForgotPassword operation: Client 1example23456789 is configured with secret but SECRET_HASH was not received.`와 같은 오류를 반환합니다.

Solution

애플리케이션은 현재 사용자, 앱 클라이언트 및 클라이언트 보안 암호에 대한 SECRET_HASH를 계산해야 합니다. 계산 방법은 다음과 같습니다.

```
Base64 ( HMAC_SHA256 ( "client secret", "Username" + "Client Id" ) )
```

클라이언트 보안 암호를 가져오려면 다음을 수행할 수 있습니다.

1. Amazon Cognito 콘솔을 열고 앱 클라이언트 메뉴에서 앱 클라이언트로 이동합니다. 앱 클라이언트 정보 섹션에서 클라이언트 보안 암호를 찾습니다. 클라이언트 보안 암호 표시를 선택하면 앱 클라이언트 보안 암호가 나타납니다.
2. DescribeUserPoolClient 요청을 생성합니다. 클라이언트 보안 암호가 응답에 포함됩니다.

자세한 내용은 보안 암호 해시 값 컴퓨팅 섹션을 참조하세요.

Amazon Cognito 콘솔에서 새 사용자 풀에 대한 기본 구성 선택

문제

콘솔에서 새 사용자 풀을 설정하면 Amazon Cognito가 몇 가지 기본 설정을 자동으로 선택합니다. 일부 설정은 사용자 풀이 생성된 후에는 변경할 수 없습니다. 정보에 입각한 선택을 하고 Amazon Cognito가 자동으로 선택한 항목을 이해하려면 어떻게 해야 하나요?

Solution

Amazon Cognito 콘솔의 새로운 사용자 풀 설정은 신속한 테스트 및 프로토타이핑을 위해 설계되었습니다. 콘솔은 사용자 풀 생성 후 변경할 수 없는 가장 중요한 구성 선택 사항만 제공합니다. Amazon Cognito가 자동으로 구성하는 다른 모든 설정은 나중에 수정할 수 있습니다.

다음과 같이 하는 것이 좋습니다.

1. 구현을 구체화하는 동안 콘솔을 사용하여 테스트 사용자 풀을 생성합니다.
2. 프로덕션 구성을 결정한 후 해당 설정을 테스트 사용자 풀에 적용합니다.
3. DescribeUserPool 및 DescribeUserPoolClient API 작업을 사용하여 테스트된 구성의 JSON 템플릿을 생성합니다.
4. 이러한 템플릿을 AWS SDKs, CDK, REST API 또는와 같은 배포 도구와 함께 사용하여 프로덕션 리소스를 CloudFormation 생성합니다.

자세한 내용은 Amazon Cognito 콘솔에서 새 애플리케이션 생성 섹션을 참조하세요.

추가 문제 해결 리소스

추가 문제 해결 지침 및 커뮤니티 기여 솔루션을 보려면 다음 외부 리소스를 탐색할 수도 있습니다.

- AWS re:Post Amazon Cognito 커뮤니티 - 커뮤니티 질문 및 솔루션 찾아보기
- AWS 지식 센터 Amazon Cognito 문서 - 선별된 문제 해결 문서

AWS SDK를 사용한 Amazon Cognito 코드 예시

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

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

코드 예시

- AWS SDKs를 사용한 Amazon Cognito 자격 증명의 코드 예제
  - AWS SDKs를 사용한 Amazon Cognito 자격 증명의 기본 예제
    - AWS SDKs 사용한 Amazon Cognito 자격 증명에 대한 작업
      - AWS SDK 또는 CLI와 CreateIdentityPool 함께 사용
      - AWS SDK 또는 CLI와 DeleteIdentityPool 함께 사용
      - CLI로 DescribeIdentityPool 사용
      - AWS SDK와 GetCredentialsForIdentity 함께 사용
      - CLI로 GetIdentityPoolRoles 사용
      - AWS SDK 또는 CLI와 ListIdentityPools 함께 사용
      - CLI로 SetIdentityPoolRoles 사용
      - CLI로 UpdateIdentityPool 사용
  - AWS SDKs를 사용한 Amazon Cognito 자격 증명 시나리오
    - Amazon Textract 탐색기 애플리케이션 생성
- AWS SDKs를 사용하는 Amazon Cognito 자격 증명 공급자의 코드 예제
  - AWS SDKs 사용하는 Amazon Cognito 자격 증명 공급자의 기본 예제
    - Hello Amazon Cognito
    - AWS SDKs를 사용하는 Amazon Cognito 자격 증명 공급자에 대한 작업
      - AWS SDK 또는 CLI와 AdminCreateUser 함께 사용
      - AWS SDK 또는 CLI와 AdminGetUser 함께 사용
      - AWS SDK 또는 CLI와 AdminInitiateAuth 함께 사용
      - AWS SDK 또는 CLI와 AdminRespondToAuthChallenge 함께 사용
      - AWS SDK 또는 CLI와 AdminSetUserPassword 함께 사용
      - AWS SDK 또는 CLI와 AssociateSoftwareToken 함께 사용

- AWS SDK 또는 CLI와 ConfirmDevice 함께 사용
- AWS SDK 또는 CLI와 ConfirmForgotPassword 함께 사용
- AWS SDK 또는 CLI와 ConfirmSignUp 함께 사용
- AWS SDK 또는 CLI와 CreateUserPool 함께 사용
- AWS SDK 또는 CLI와 CreateUserPoolClient 함께 사용
- AWS SDK 또는 CLI와 DeleteUser 함께 사용
- AWS SDK와 DescribeTermsByClient 함께 사용
- AWS SDK 또는 CLI와 DescribeUserPoolClient 함께 사용
- AWS SDK 또는 CLI와 ForgotPassword 함께 사용
- AWS SDK와 GetClientToken 함께 사용
- AWS SDK 또는 CLI와 InitiateAuth 함께 사용
- AWS SDK 또는 CLI와 ListUserPools 함께 사용
- AWS SDK 또는 CLI와 ListUsers 함께 사용
- AWS SDK 또는 CLI와 ResendConfirmationCode 함께 사용
- AWS SDK 또는 CLI와 RespondToAuthChallenge 함께 사용
- AWS SDK 또는 CLI와 SignUp 함께 사용
- AWS SDK 또는 CLI와 UpdateUserPool 함께 사용
- AWS SDK 또는 CLI와 VerifySoftwareToken 함께 사용

- AWS SDKs를 사용하는 Amazon Cognito 자격 증명 공급자에 대한 시나리오
  - AWS SDK를 사용하여 Lambda 함수로 알려진 Amazon Cognito 사용자를 자동으로 확인
  - AWS SDK를 사용하여 Lambda 함수를 사용하여 알려진 Amazon Cognito 사용자를 자동으로 마이그레이션
  - AWS SDK를 사용하여 M2M 클라이언트 토큰을 가져오고 Amazon Cognito에서 용어 검색
  - 사용자 풀 시작하기
  - AWS SDK를 사용하여 MFA가 필요한 Amazon Cognito 사용자 풀에 사용자 등록
  - Amazon Cognito ID 풀 및 인증 흐름 사용
  - AWS SDK를 사용하여 Amazon Cognito 사용자 인증 후 Lambda 함수로 사용자 지정 활동 데이터 쓰기
- AWS SDKs를 사용한 Amazon Cognito Sync의 코드 예제
  - AWS SDKs 사용한 Amazon Cognito Sync의 기본 예제
    - AWS SDKs를 사용한 Amazon Cognito Sync 작업

- AWS SDK와 ListIdentityPoolUsage 함께 사용

AWS SDKs를 사용한 Amazon Cognito 자격 증명의 코드 예제

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

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

Amazon Cognito 자격 증명

- AWS SDKs를 사용한 Amazon Cognito 자격 증명의 기본 예제
  - AWS SDKs 사용한 Amazon Cognito 자격 증명에 대한 작업
    - AWS SDK 또는 CLI와 CreateIdentityPool 함께 사용
    - AWS SDK 또는 CLI와 DeleteIdentityPool 함께 사용
    - CLI로 DescribeIdentityPool 사용
    - AWS SDK와 GetCredentialsForIdentity 함께 사용
    - CLI로 GetIdentityPoolRoles 사용
    - AWS SDK 또는 CLI와 ListIdentityPools 함께 사용
    - CLI로 SetIdentityPoolRoles 사용
    - CLI로 UpdateIdentityPool 사용
- AWS SDKs를 사용한 Amazon Cognito 자격 증명 시나리오
  - Amazon Textract 탐색기 애플리케이션 생성

AWS SDKs를 사용한 Amazon Cognito 자격 증명의 기본 예제

다음 코드 예제는 Amazon Cognito ID의 기본 사항을 AWS SDK와 함께 사용하는 방법을 보여줍니다.

예제

- AWS SDKs 사용한 Amazon Cognito 자격 증명에 대한 작업
  - AWS SDK 또는 CLI와 CreateIdentityPool 함께 사용
  - AWS SDK 또는 CLI와 DeleteIdentityPool 함께 사용
  - CLI로 DescribeIdentityPool 사용
  - AWS SDK와 GetCredentialsForIdentity 함께 사용
  - CLI로 GetIdentityPoolRoles 사용
  - AWS SDK 또는 CLI와 ListIdentityPools 함께 사용
  - CLI로 SetIdentityPoolRoles 사용
  - CLI로 UpdateIdentityPool 사용

AWS SDKs 사용한 Amazon Cognito 자격 증명에 대한 작업

다음 코드 예제에서는 AWS SDKs를 사용하여 개별 Amazon Cognito 자격 증명 작업을 수행하는 방법을 보여줍니다. 각 예제에는 GitHub에 대한 링크가 포함되어 있습니다. 여기에서 코드 설정 및 실행에 대한 지침을 찾을 수 있습니다.

이들 발췌문은 Amazon Cognito Identity API를 직접적으로 호출하며, 컨텍스트에서 실행되어야 하는 더 큰 프로그램에서 발췌한 코드입니다. AWS SDKs를 사용한 Amazon Cognito 자격 증명 시나리오에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

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

예제

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

AWS SDK 또는 CLI와 **CreateIdentityPool** 함께 사용

다음 코드 예시는 CreateIdentityPool의 사용 방법을 보여 줍니다.

CLI

AWS CLI

Cognito ID 풀 공급자를 사용하여 자격 증명 풀 생성

이 예시에서는 MyIdentityPool이라는 ID 풀을 생성합니다. Cognito ID 풀 공급자가 있습니다. 인증되지 않은 자격 증명은 허용되지 않습니다.

명령:

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

출력:

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

- API 세부 정보는 AWS CLI 명령 참조의 CreateIdentityPool을 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateIdentityPool {
    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <identityPoolName>\s

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

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

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

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

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

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

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

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateIdentityPool을 참조하세요.

PowerShell

Tools for PowerShell V4

예제 1: 인증되지 않은 ID를 허용하는 새 ID 풀을 생성합니다.

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

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateIdentityPool을 참조하세요.

Tools for PowerShell V5

예제 1: 인증되지 않은 ID를 허용하는 새 ID 풀을 생성합니다.

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

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 CreateIdentityPool을 참조하세요.

Swift

SDK for Swift

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import AWSCognitoIdentity


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

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

            return poolId
        } catch {
            print("ERROR: createIdentityPool:", dump(error))
            throw error
        }
    }
```

- 자세한 내용은 AWS SDK for Swift 개발자 안내서를 참조하세요.
- API 세부 정보는 AWS SDK for Swift API 참조의 CreateIdentityPool을 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **DeleteIdentityPool** 함께 사용

다음 코드 예시는 DeleteIdentityPool의 사용 방법을 보여 줍니다.

CLI

AWS CLI

자격 증명 풀 삭제

다음 delete-identity-pool 예시에서는 지정된 ID 풀을 삭제합니다.

명령:

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

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 참조에서 DeleteIdentityPool을 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

```
import
 software.amazon.awssdk.services.cognitoidentity.model.DeleteIdentityPoolRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteIdentityPool {

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

                Usage:
                    <identityPoolId>\s

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

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

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

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

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

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

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

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

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteIdentityPool을 참조하세요.

PowerShell

Tools for PowerShell V4

예제 1: 특정 ID 풀을 삭제합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteIdentityPool을 참조하세요.

Tools for PowerShell V5

예제 1: 특정 ID 풀을 삭제합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 DeleteIdentityPool을 참조하세요.

Swift

SDK for Swift

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import AWSCognitoIdentity


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

            _ = try await cognitoIdentityClient.deleteIdentityPool(input: input)
        } catch {
            print("ERROR: deleteIdentityPool:", dump(error))
            throw error
        }
    }
```

- 자세한 내용은 AWS SDK for Swift 개발자 안내서를 참조하세요.
- API 세부 정보는 AWS SDK for Swift API 참조의 DeleteIdentityPool을 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 **`DescribeIdentityPool`** 사용

다음 코드 예시는 `DescribeIdentityPool`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

ID 풀을 설명하는 방법

이 예제에서는 ID 풀을 설명합니다.

명령:

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

출력:

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

- API 세부 정보는 AWS CLI 명령 참조의 DescribeIdentityPool 섹션을 참조하세요.

PowerShell

Tools for PowerShell V4

예제 1: ID별로 특정 ID 풀에 대한 정보를 검색합니다.

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

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeIdentityPool을 참조하세요.

Tools for PowerShell V5

예제 1: ID별로 특정 ID 풀에 대한 정보를 검색합니다.

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

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 DescribeIdentityPool을 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 **GetCredentialsForIdentity** 함께 사용

다음 코드 예시는 GetCredentialsForIdentity의 사용 방법을 보여 줍니다.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetIdentityCredentials {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <identityId>\s

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

```
            """;

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

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

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

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

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

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

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetCredentialsForIdentity 섹션을 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 **GetIdentityPoolRoles** 사용

다음 코드 예시는 GetIdentityPoolRoles의 사용 방법을 보여 줍니다.

CLI

AWS CLI

ID 풀 역할을 가져오는 방법

이 예제에서는 ID 풀이 나열되어 있습니다.

명령:

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

출력:

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

- API 세부 정보는 AWS CLI 명령 참조의 GetIdentityPoolRoles 섹션을 참조하세요.

PowerShell

Tools for PowerShell V4

예제 1: 특정 ID 풀의 역할에 대한 정보를 가져옵니다.

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

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetIdentityPoolRoles을 참조하세요.

Tools for PowerShell V5

예제 1: 특정 ID 풀의 역할에 대한 정보를 가져옵니다.

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

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 GetIdentityPoolRoles을 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **`ListIdentityPools`** 함께 사용

다음 코드 예시는 `ListIdentityPools`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

ID 풀 나열

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

명령:

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

출력:

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

- API 세부 정보는 AWS CLI 명령 참조의 ListIdentityPools를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListIdentityPools {
    public static void main(String[] args) {
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

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

    public static void listIdPools(CognitoIdentityClient cognitoClient) {
        try {
            ListIdentityPoolsRequest poolsRequest =
 ListIdentityPoolsRequest.builder()
                    .maxResults(15)
                    .build();

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

        } catch (CognitoIdentityProviderException e) {
```

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

- API 세부 정보는AWS SDK for Java 2.x API 참조의 ListIdentityPools를 참조하세요.

PowerShell

Tools for PowerShell V4

예제 1: 기존 ID 풀 목록을 검색합니다.

```
Get-CGIIdentityPoolList
```

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListIdentityPools을 참조하세요.

Tools for PowerShell V5

예제 1: 기존 ID 풀 목록을 검색합니다.

```
Get-CGIIdentityPoolList
```

출력:

```
IdentityPoolId
 IdentityPoolName
--------------
 ----------------
```

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 ListIdentityPools을 참조하세요.

Swift

SDK for Swift

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import AWSCognitoIdentity


    /// Return the ID of the identity pool with the specified name.
    ///
    /// - Parameters:
    ///   - name: The name of the identity pool whose ID should be returned.
    ///
    /// - Returns: A string containing the ID of the specified identity pool
    ///   or `nil` on error or if not found.
    ///
    func getIdentityPoolID(name: String) async throws -> String? {
        let listPoolsInput = ListIdentityPoolsInput(maxResults: 25)
        // Use "Paginated" to get all the objects.
        // This lets the SDK handle the 'nextToken' field in
 "ListIdentityPoolsOutput".
        let pages = cognitoIdentityClient.listIdentityPoolsPaginated(input:
 listPoolsInput)

        do {
            for try await page in pages {
                guard let identityPools = page.identityPools else {
                    print("ERROR: listIdentityPoolsPaginated returned nil
 contents.")
```

```
                    continue
                }

                /// Read pages of identity pools from Cognito until one is found
                /// whose name matches the one specified in the `name` parameter.
                /// Return the matching pool's ID.

                for pool in identityPools {
                    if pool.identityPoolName == name {
                        return pool.identityPoolId!
                    }
                }
            }
        } catch {
            print("ERROR: getIdentityPoolID:", dump(error))
            throw error
        }

        return nil
    }
```

기존 ID 풀의 ID를 가져오거나 존재하지 않으면 생성합니다.

```
import AWSCognitoIdentity


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

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

```
            return poolId
        } catch {
            print("ERROR: getOrCreateIdentityPoolID:", dump(error))
            throw error
        }
    }
```

- 자세한 내용은 AWS SDK for Swift 개발자 안내서를 참조하세요.
- API 세부 정보는 AWS SDK for Swift API 참조의 ListIdentityPools를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 **SetIdentityPoolRoles** 사용

다음 코드 예시는 SetIdentityPoolRoles의 사용 방법을 보여 줍니다.

CLI

AWS CLI

ID 풀 역할을 설정하는 방법

다음 set-identity-pool-roles 예제에서는 ID 풀의 역할을 설정합니다.

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

- API 세부 정보는 AWS CLI 명령 참조의 SetIdentityPoolRoles 섹션을 참조하세요.

PowerShell

Tools for PowerShell V4

예제 1: 인증되지 않은 IAM 역할을 갖도록 특정 ID 풀을 구성합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 SetIdentityPoolRoles을 참조하세요.

Tools for PowerShell V5

예제 1: 인증되지 않은 IAM 역할을 갖도록 특정 ID 풀을 구성합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 SetIdentityPoolRoles을 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

CLI로 **UpdateIdentityPool** 사용

다음 코드 예시는 UpdateIdentityPool의 사용 방법을 보여 줍니다.

CLI

AWS CLI

ID 풀을 생성하는 방법

이 예제에서는 ID 풀을 업데이트합니다. 이름을 MyIdentityPool로 설정합니다. 자격 증명 공급자로 Cognito를 추가합니다. 인증되지 않은 자격 증명은 허용되지 않습니다.

명령:

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

출력:

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

- API 세부 정보는 AWS CLI 명령 참조의 UpdateIdentityPool 섹션을 참조하세요.

PowerShell

Tools for PowerShell V4

예제 1: ID 풀 속성 중 일부를 업데이트합니다. 이 케이스에서는 ID 풀의 이름을 업데이트합니다.

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

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateIdentityPool을 참조하세요.

Tools for PowerShell V5

예제 1: ID 풀 속성 중 일부를 업데이트합니다. 이 케이스에서는 ID 풀의 이름을 업데이트합니다.

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

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 UpdateIdentityPool을 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDKs를 사용한 Amazon Cognito 자격 증명 시나리오

다음 코드 예제에서는 Amazon Cognito Identity AWS SDKs에서 일반적인 시나리오를 구현하는 방법을 보여줍니다. 이러한 시나리오에서는 Amazon Cognito ID 내에서 여러 함수를 호출하거나 다른 AWS 서비스와 결합하여 특정 작업을 수행하는 방법을 보여줍니다. 각 시나리오에는 전체 소스 코드에 대한 링크가 포함되어 있습니다. 여기에서 코드를 설정 및 실행하는 방법에 대한 지침을 찾을 수 있습니다.

시나리오는 컨텍스트에 맞는 서비스 작업을 이해하는 데 도움이 되도록 중급 수준의 경험을 대상으로 합니다.

예시

- Amazon Textract 탐색기 애플리케이션 생성

Amazon Textract 탐색기 애플리케이션 생성

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

JavaScript

JavaScript용 SDK(v3)

AWS SDK for JavaScript 를 사용하여 Amazon Textract를 사용하여 문서 이미지에서 데이터를 추출하고 대화형 웹 페이지에 표시하는 React 애플리케이션을 빌드하는 방법을 보여줍니다. 이 예제는 웹 브라우저에서 실행되며 자격 증명을 위해 인증된 Amazon Cognito 자격 증명이 필요합니다. 이 애플리케이션은 스토리지로 Amazon Simple Storage Service(Amazon S3)를 사용하고 알림을 위해 Amazon Simple Notification Service(Amazon SNS) 주제를 구독하는 Amazon Simple Queue Service(Amazon SQS) 대기열을 폴링합니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

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

Python

SDK for Python(Boto3)

Amazon Textract와 AWS SDK for Python (Boto3) 함께를 사용하여 문서 이미지에서 텍스트, 양식 및 테이블 요소를 감지하는 방법을 보여줍니다. 입력 이미지와 Amazon Textract 출력은 탐지된 요소를 탐색할 수 있는 Tkinter 애플리케이션에 표시됩니다.

- 문서 이미지를 Amazon Textract에 제출하고 감지된 요소의 출력을 탐색합니다.
- Amazon Textract로 직접, 또는 Amazon Simple Storage Service(Amazon S3) 버킷을 통해 이미지를 제출합니다.
- 비동기식 API를 사용하여 작업이 완료되면 Amazon Simple Notification Service(Amazon SNS) 주제에 알림을 게시하는 작업을 시작합니다.

- Amazon Simple Queue Service(Amazon SQS) 대기열에서 작업 완료 메시지를 폴링하고 결과를 표시합니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

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

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDKs를 사용하는 Amazon Cognito 자격 증명 공급자의 코드 예제

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

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

Amazon Cognito 자격 증명 공급자

- AWS SDKs 사용하는 Amazon Cognito 자격 증명 공급자의 기본 예제
  - Hello Amazon Cognito
  - AWS SDKs를 사용하는 Amazon Cognito 자격 증명 공급자에 대한 작업
    - AWS SDK 또는 CLI와 AdminCreateUser 함께 사용
    - AWS SDK 또는 CLI와 AdminGetUser 함께 사용

    - AWS SDK 또는 CLI와 AdminInitiateAuth 함께 사용
    - AWS SDK 또는 CLI와 AdminRespondToAuthChallenge 함께 사용
    - AWS SDK 또는 CLI와 AdminSetUserPassword 함께 사용
    - AWS SDK 또는 CLI와 AssociateSoftwareToken 함께 사용
    - AWS SDK 또는 CLI와 ConfirmDevice 함께 사용
    - AWS SDK 또는 CLI와 ConfirmForgotPassword 함께 사용
    - AWS SDK 또는 CLI와 ConfirmSignUp 함께 사용
    - AWS SDK 또는 CLI와 CreateUserPool 함께 사용
    - AWS SDK 또는 CLI와 CreateUserPoolClient 함께 사용
    - AWS SDK 또는 CLI와 DeleteUser 함께 사용
    - AWS SDK와 DescribeTermsByClient 함께 사용
    - AWS SDK 또는 CLI와 DescribeUserPoolClient 함께 사용
    - AWS SDK 또는 CLI와 ForgotPassword 함께 사용
    - AWS SDK와 GetClientToken 함께 사용
    - AWS SDK 또는 CLI와 InitiateAuth 함께 사용
    - AWS SDK 또는 CLI와 ListUserPools 함께 사용
    - AWS SDK 또는 CLI와 ListUsers 함께 사용
    - AWS SDK 또는 CLI와 ResendConfirmationCode 함께 사용
    - AWS SDK 또는 CLI와 RespondToAuthChallenge 함께 사용
    - AWS SDK 또는 CLI와 SignUp 함께 사용
    - AWS SDK 또는 CLI와 UpdateUserPool 함께 사용
    - AWS SDK 또는 CLI와 VerifySoftwareToken 함께 사용
- AWS SDKs를 사용하는 Amazon Cognito 자격 증명 공급자에 대한 시나리오
    - AWS SDK를 사용하여 Lambda 함수로 알려진 Amazon Cognito 사용자를 자동으로 확인
    - AWS SDK를 사용하여 Lambda 함수를 사용하여 알려진 Amazon Cognito 사용자를 자동으로 마이그레이션
    - AWS SDK를 사용하여 M2M 클라이언트 토큰을 가져오고 Amazon Cognito에서 용어 검색
    - 사용자 풀 시작하기
    - AWS SDK를 사용하여 MFA가 필요한 Amazon Cognito 사용자 풀에 사용자 등록
    - Amazon Cognito ID 풀 및 인증 흐름 사용

- AWS SDK를 사용하여 Amazon Cognito 사용자 인증 후 Lambda 함수로 사용자 지정 활동 데이터 쓰기

AWS SDKs 사용하는 Amazon Cognito 자격 증명 공급자의 기본 예제

다음 코드 예제는 Amazon Cognito ID 제공업체의 기본 사항을 AWS SDK와 함께 사용하는 방법을 보여줍니다.

예제

- Hello Amazon Cognito
- AWS SDKs를 사용하는 Amazon Cognito 자격 증명 공급자에 대한 작업
  - AWS SDK 또는 CLI와 AdminCreateUser 함께 사용
  - AWS SDK 또는 CLI와 AdminGetUser 함께 사용
  - AWS SDK 또는 CLI와 AdminInitiateAuth 함께 사용
  - AWS SDK 또는 CLI와 AdminRespondToAuthChallenge 함께 사용
  - AWS SDK 또는 CLI와 AdminSetUserPassword 함께 사용
  - AWS SDK 또는 CLI와 AssociateSoftwareToken 함께 사용
  - AWS SDK 또는 CLI와 ConfirmDevice 함께 사용
  - AWS SDK 또는 CLI와 ConfirmForgotPassword 함께 사용
  - AWS SDK 또는 CLI와 ConfirmSignUp 함께 사용
  - AWS SDK 또는 CLI와 CreateUserPool 함께 사용
  - AWS SDK 또는 CLI와 CreateUserPoolClient 함께 사용
  - AWS SDK 또는 CLI와 DeleteUser 함께 사용
  - AWS SDK와 DescribeTermsByClient 함께 사용
  - AWS SDK 또는 CLI와 DescribeUserPoolClient 함께 사용
  - AWS SDK 또는 CLI와 ForgotPassword 함께 사용
  - AWS SDK와 GetClientToken 함께 사용
  - AWS SDK 또는 CLI와 InitiateAuth 함께 사용
  - AWS SDK 또는 CLI와 ListUserPools 함께 사용
  - AWS SDK 또는 CLI와 ListUsers 함께 사용
  - AWS SDK 또는 CLI와 ResendConfirmationCode 함께 사용
  - AWS SDK 또는 CLI와 RespondToAuthChallenge 함께 사용

- AWS SDK 또는 CLI와 SignUp 함께 사용
- AWS SDK 또는 CLI와 UpdateUserPool 함께 사용
- AWS SDK 또는 CLI와 VerifySoftwareToken 함께 사용

Hello Amazon Cognito

다음 코드 예제에서는 Amazon Cognito 사용을 시작하는 방법을 보여줍니다.

C++

SDK for C++

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

CMakeLists.txt CMake 파일의 코드입니다.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS cognito-idp)

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

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
```

```
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_cognito.cpp)

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

hello_cognito.cpp 소스 파일의 코드입니다.

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

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

```
int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

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

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

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

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

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

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

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

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

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

- API 세부 정보는 AWS SDK for C++ API 참조의 ListUserPools를 참조하십시오.

Go

SDK for Go V2

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

```
package main

import (
 "context"
 "fmt"
 "log"

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

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
```

```
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 cognitoClient := cognitoidentityprovider.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the user pools for your account.")
 var pools []types.UserPoolDescriptionType
 paginator := cognitoidentityprovider.NewListUserPoolsPaginator(
  cognitoClient, &cognitoidentityprovider.ListUserPoolsInput{MaxResults:
 aws.Int32(10)})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get user pools. Here's why: %v\n", err)
  } else {
   pools = append(pools, output.UserPools...)
  }
 }
 if len(pools) == 0 {
  fmt.Println("You don't have any user pools!")
 } else {
  for _, pool := range pools {
   fmt.Printf("\t%v: %v\n", *pool.Name, *pool.Id)
  }
 }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 ListUserPools를 참조하세요.

Java

SDK for Java 2.x

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

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

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

    public static void listAllUserPools(CognitoIdentityProviderClient
 cognitoClient) {
```

```
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                    .maxResults(10)
                    .build();

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

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

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListUserPools를 참조하세요.

JavaScript

SDK for JavaScript (v3)

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

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

const client = new CognitoIdentityProviderClient({});

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

```
  const userPoolNames = [];

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

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

- API 세부 정보는 AWS SDK for JavaScript API 참조의 ListUserPools를 참조하세요.

Python

SDK for Python(Boto3)

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

```
import boto3

# Create a Cognito Identity Provider client
cognitoidp = boto3.client("cognito-idp")

# Initialize a paginator for the list_user_pools operation
paginator = cognitoidp.get_paginator("list_user_pools")

# Create a PageIterator from the paginator
page_iterator = paginator.paginate(MaxResults=10)

# Initialize variables for pagination
user_pools = []

# Handle pagination
for page in page_iterator:
```

```
    user_pools.extend(page.get("UserPools", []))

# Print the list of user pools
print("User Pools for the account:")
if user_pools:
    for pool in user_pools:
        print(f"Name: {pool['Name']}, ID: {pool['Id']}")
else:
    print("No user pools found.")
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 ListUserPools를 참조하세요.

Ruby

SDK for Ruby

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

```
require 'aws-sdk-cognitoidentityprovider'
require 'logger'

# CognitoManager is a class responsible for managing AWS Cognito operations
# such as listing all user pools in the current AWS account.
class CognitoManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all user pools associated with the AWS account.
  def list_user_pools
    paginator = @client.list_user_pools(max_results: 10)
    user_pools = []
    paginator.each_page do |page|
      user_pools.concat(page.user_pools)
```

```
    end

    if user_pools.empty?
      @logger.info('No Cognito user pools found.')
    else
      user_pools.each do |user_pool|
        @logger.info("User pool ID: #{user_pool.id}")
        @logger.info("User pool name: #{user_pool.name}")
        @logger.info("User pool status: #{user_pool.status}")
        @logger.info('---')
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  cognito_client = Aws::CognitoIdentityProvider::Client.new
  manager = CognitoManager.new(cognito_client)
  manager.list_user_pools
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListUserPools를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDKs를 사용하는 Amazon Cognito 자격 증명 공급자에 대한 작업

다음 코드 예제에서는 AWS SDKs를 사용하여 개별 Amazon Cognito 자격 증명 공급자 작업을 수행하는 방법을 보여줍니다. 각 예제에는 GitHub에 대한 링크가 포함되어 있습니다. 여기에서 코드 설정 및 실행에 대한 지침을 찾을 수 있습니다.

이들 발췌문은 Amazon Cognito Identity Provider API를 직접적으로 호출하며, 컨텍스트에서 실행되어야 하는 더 큰 프로그램에서 발췌한 코드입니다. AWS SDKs를 사용하는 Amazon Cognito 자격 증명 공급자에 대한 시나리오에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

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

예제

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

AWS SDK 또는 CLI와 **`AdminCreateUser`** 함께 사용

다음 코드 예제는 `AdminCreateUser`의 사용 방법을 보여 줍니다.

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

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

CLI

AWS CLI

사용자 생성

다음 admin-create-user 예시에서는 지정된 설정 이메일 주소와 전화번호를 사용하여 사용자를 생성합니다.

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

출력:

```
{
    "User": {
        "Username": "diego",
        "Attributes": [
            {
                "Name": "sub",
                "Value": "7325c1de-b05b-4f84-b321-9adc6e61f4a2"
            },
            {
                "Name": "phone_number",
                "Value": "+15555551212"
            },
            {
                "Name": "email",
                "Value": "diego@example.com"
            }
        ],
        "UserCreateDate": 1548099495.428,
        "UserLastModifiedDate": 1548099495.428,
        "Enabled": true,
```

```
        "UserStatus": "FORCE_CHANGE_PASSWORD"
    }
}
```

- API 세부 정보는 AWS CLI 명령 참조의 AdminCreateUser를 참조하세요.

Go

SDK for Go V2

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

```
import (
 "context"
 "errors"
 "log"

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

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId
 string, userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
```

```
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 AdminCreateUser를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **AdminGetUser** 함께 사용

다음 코드 예제는 AdminGetUser의 사용 방법을 보여 줍니다.

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

- MFA가 필요한 사용자 풀에 사용자 가입시키기

.NET

SDK for .NET

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

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

        var response = await _cognitoService.AdminGetUserAsync(userRequest);

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

- API 세부 정보는 AWS SDK for .NET API 참조의 AdminGetUser를 참조하십시오.

C++

SDK for C++

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

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

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

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

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

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

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

- API 세부 정보는 AWS SDK for C++ API 참조의 AdminGetUser를 참조하세요.

CLI

AWS CLI

사용자 가져오기

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

명령:

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

출력:

```
{
```

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

- API 세부 정보는 AWS CLI 명령 참조에서 AdminGetUser를 참조하세요.

Java

SDK for Java 2.x

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

```
    public static void getAdminUser(CognitoIdentityProviderClient
identityProviderClient, String userName,
            String poolId) {
        try {
            AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
                    .username(userName)
                    .userPoolId(poolId)
                    .build();

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

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

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

JavaScript

SDK for JavaScript (v3)

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

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

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

  return client.send(command);
```

```
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 AdminGetUser를 참조하십시오.

Kotlin

SDK for Kotlin

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

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

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

- API 세부 정보는 AWS SDK for Kotlin API 참조의 AdminGetUser를 참조하십시오.

Python

SDK for Python (Boto3)

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

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

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


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

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

        :param user_name: The user name that identifies the new user.
```

```
        :param password: The password for the new user.
        :param user_email: The email address for the new user.
        :return: True when the user is already confirmed with Amazon Cognito.
                 Otherwise, false.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "Password": password,
                "UserAttributes": [{"Name": "email", "Value": user_email}],
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.sign_up(**kwargs)
            confirmed = response["UserConfirmed"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "UsernameExistsException":
                response = self.cognito_idp_client.admin_get_user(
                    UserPoolId=self.user_pool_id, Username=user_name
                )
                logger.warning(
                    "User %s exists and is %s.", user_name,
response["UserStatus"]
                )
                confirmed = response["UserStatus"] == "CONFIRMED"
            else:
                logger.error(
                    "Couldn't sign up %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return confirmed
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 AdminGetUser를 참조하세요.

Swift

SDK for Swift

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

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Get information about a specific user in a user pool.
    ///
    /// - Parameters:
    ///   - cipClient: The Amazon Cognito Identity Provider client to use.
    ///   - userName: The user to retrieve information about.
    ///   - userPoolId: The user pool to search for the specified user.
    ///
    /// - Returns: `true` if the user's information was successfully
    ///   retrieved. Otherwise returns `false`.
    func adminGetUser(cipClient: CognitoIdentityProviderClient, userName: String,
                      userPoolId: String) async -> Bool {
        do {
            let output = try await cipClient.adminGetUser(
                input: AdminGetUserInput(
                    userPoolId: userPoolId,
                    username: userName
                )
            )

            guard let userStatus = output.userStatus else {
                print("*** Unable to get the user's status.")
                return false
            }

            print("User status: \(userStatus)")
            return true
        } catch {
            return false
        }
```

```
    }
```

- API 세부 정보는 AWS SDK for Swift API 참조의 AdminGetUser를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **AdminInitiateAuth** 함께 사용

다음 코드 예제는 AdminInitiateAuth의 사용 방법을 보여 줍니다.

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

- MFA가 필요한 사용자 풀에 사용자 가입시키기

.NET

SDK for .NET

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

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

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

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

- API 세부 정보는 AWS SDK for .NET API 참조의 AdminInitiateAuth를 참조하십시오.

C++

SDK for C++

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

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

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

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

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

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

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

- API 세부 정보는 AWS SDK for C++ API 참조의 AdminInitiateAuth를 참조하세요.

CLI

AWS CLI

관리 사용자로 로그인하려면

다음 admin-initiate-auth 예제에서는 사용자 diego@example.com에 로그인합니다. 이 예제에는 위협 방지를 위한 메타데이터와 Lambda 트리거를 위한 ClientMetadata도 포함되어 있습니다. 사용자는 TOTP MFA에 대해 구성되어 있으며 인증을 완료하기 전에 인증 앱에서 코드를 제공하라는 질문을 받습니다.

```
aws cognito-idp admin-initiate-auth \
    --user-pool-id us-west-2_EXAMPLE \
    --client-id 1example23456789 \
    --auth-flow ADMIN_USER_PASSWORD_AUTH \
    --auth-parameters USERNAME=diego@example.com,PASSWORD="My@Example
$Password3!",SECRET_HASH=ExampleEncodedClientIdSecretAndUsername= \
    --context-data="{\"EncodedData\":\"abc123example\",\"HttpHeaders\":
[{\"headerName\":\"UserAgent\",\"headerValue\":\"Mozilla/5.0 (Windows NT
 6.1; Win64; x64; rv:47.0) Gecko/20100101 Firefox/47.0\"}],\"IpAddress\":
\"192.0.2.1\",\"ServerName\":\"example.com\",\"ServerPath\":\"/login\"}" \
    --client-metadata="{\"MyExampleKey\": \"MyExampleValue\"}"
```

출력:

```
{
    "ChallengeName": "SOFTWARE_TOKEN_MFA",
    "Session": "AYABeExample...",
    "ChallengeParameters": {
        "FRIENDLY_DEVICE_NAME": "MyAuthenticatorApp",
        "USER_ID_FOR_SRP": "diego@example.com"
    }
}
```

자세한 내용은 Amazon Cognito 개발자 안내서의 관리자 인증 흐름 섹션을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조에서 AdminInitiateAuth를 참조하세요.

Java

SDK for Java 2.x

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

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

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

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

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

        return null;
    }
```

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

JavaScript

SDK for JavaScript (v3)

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

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

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

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

- API 세부 정보는 AWS SDK for JavaScript API 참조의 AdminInitiateAuth를 참조하십시오.

Kotlin

SDK for Kotlin

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

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

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

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

- API 세부 정보는 AWS SDK for Kotlin API 참조의 AdminInitiateAuth를 참조하십시오.

Python

SDK for Python (Boto3)

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

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

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


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

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

        :param user_name: The name of the user to sign in.
```

```
        :param password: The user's password.
        :return: The result of the sign-in attempt. When sign-in is successful,
this
                 returns an access token that can be used to get AWS credentials.
Otherwise,
                 Amazon Cognito returns a challenge to set up an MFA application,
                 or a challenge to enter an MFA code from a registered MFA
application.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "AuthFlow": "ADMIN_USER_PASSWORD_AUTH",
                "AuthParameters": {"USERNAME": user_name, "PASSWORD": password},
            }
            if self.client_secret is not None:
                kwargs["AuthParameters"]["SECRET_HASH"] =
self._secret_hash(user_name)
            response = self.cognito_idp_client.admin_initiate_auth(**kwargs)
            challenge_name = response.get("ChallengeName", None)
            if challenge_name == "MFA_SETUP":
                if (
                    "SOFTWARE_TOKEN_MFA"
                    in response["ChallengeParameters"]["MFAS_CAN_SETUP"]
                ):
                    response.update(self.get_mfa_secret(response["Session"]))
                else:
                    raise RuntimeError(
                        "The user pool requires MFA setup, but the user pool is
not "
                        "configured for TOTP MFA. This example requires TOTP
MFA."
                    )
        except ClientError as err:
            logger.error(
                "Couldn't start sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
```

```
            return response
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 AdminInitiateAuth를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

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

```
    TRY.
        " Set up authentication parameters
        DATA(lt_auth_params) = VALUE /aws1/
cl_cgpauthparamstype_w=>tt_authparameterstype(
          ( VALUE /aws1/cl_cgpauthparamstype_w=>ts_authparameterstype_maprow(
              key = 'USERNAME'
              value = NEW /aws1/cl_cgpauthparamstype_w( iv_user_name ) ) )
          ( VALUE /aws1/cl_cgpauthparamstype_w=>ts_authparameterstype_maprow(
              key = 'PASSWORD'
              value = NEW /aws1/cl_cgpauthparamstype_w( iv_password ) ) )
        ).

        " Add SECRET_HASH if provided
        IF iv_secret_hash IS NOT INITIAL.
          INSERT VALUE #(
            key = 'SECRET_HASH'
            value = NEW /aws1/cl_cgpauthparamstype_w( iv_secret_hash )
          ) INTO TABLE lt_auth_params.
        ENDIF.

        oo_result = lo_cgp->admininitiateauth(
          iv_userpoolid = iv_user_pool_id
          iv_clientid = iv_client_id
          iv_authflow = 'ADMIN_USER_PASSWORD_AUTH'
          it_authparameters = lt_auth_params
        ).
```

```
        DATA(lv_challenge) = oo_result->get_challengename( ).

        IF lv_challenge IS INITIAL.
          MESSAGE 'User successfully signed in.' TYPE 'I'.
        ELSE.
          MESSAGE |Authentication challenge required: { lv_challenge }.| TYPE
 'I'.
        ENDIF.

      CATCH /aws1/cx_cgpusernotfoundex INTO DATA(lo_user_ex).
        MESSAGE |User { iv_user_name } not found.| TYPE 'E'.

      CATCH /aws1/cx_cgpnotauthorizedex INTO DATA(lo_auth_ex).
        MESSAGE 'Not authorized. Check credentials.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 SDK for SAP ABAP API 참조의 AdminInitiateAuth를 참조하세요. AWS

Swift

SDK for Swift

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

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Begin an authentication session.
    ///
    /// - Parameters:
    ///   - cipClient: The `CongitoIdentityProviderClient` to use.
    ///   - clientId: The app client ID to use.
    ///   - userName: The username to check.
    ///   - password: The user's password.
    ///   - userPoolId: The user pool to use.
    ///
```

```
    /// - Returns: The session token associated with this authentication
    ///   session.
    func initiateAuth(cipClient: CognitoIdentityProviderClient, clientId: String,
                      userName: String, password: String,
                      userPoolId: String) async -> String? {
        var authParams: [String: String] = [:]

        authParams["USERNAME"] = userName
        authParams["PASSWORD"] = password

        do {
            let output = try await cipClient.adminInitiateAuth(
                input: AdminInitiateAuthInput(
                    authFlow:
CognitoIdentityProviderClientTypes.AuthFlowType.adminUserPasswordAuth,
                    authParameters: authParams,
                    clientId: clientId,
                    userPoolId: userPoolId
                )
            )

            guard let challengeName = output.challengeName else {
                print("*** Invalid response from the auth service.")
                return nil
            }

            print("=====> Response challenge is \(challengeName)")

            return output.session
        } catch _ as UserNotFoundException {
            print("*** The specified username, \(userName), doesn't exist.")
            return nil
        } catch _ as UserNotConfirmedException {
            print("*** The user \(userName) has not been confirmed.")
            return nil
        } catch {
            print("*** An unexpected error occurred.")
            return nil
        }
    }
```

- API 세부 정보는 AWS SDK for Swift API 참조의 AdminInitiateAuth를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **AdminRespondToAuthChallenge** 함께 사용

다음 코드 예제는 AdminRespondToAuthChallenge의 사용 방법을 보여 줍니다.

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

- MFA가 필요한 사용자 풀에 사용자 가입시키기

.NET

SDK for .NET

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

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

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

```
        challengeResponses.Add("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

        var respondToAuthChallengeRequest = new
AdminRespondToAuthChallengeRequest
        {
            ChallengeName = ChallengeNameType.SOFTWARE_TOKEN_MFA,
            ClientId = clientId,
            ChallengeResponses = challengeResponses,
            Session = session,
            UserPoolId = userPoolId,
        };

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

- API 세부 정보는 AWS SDK for .NET API 참조의 AdminRespondToAuthChallenge를 참조하십시오.

C++

SDK for C++

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

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

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

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

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

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

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

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

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

- API 세부 정보는 AWS SDK for C++ API 참조의 AdminRespondToAuthChallenge를 참조하세요.

CLI

AWS CLI

인증 질문에 응답하려면

인증 흐름, 사용자 풀 구성 및 사용자 설정에 따라 다양한 인증 질문에 대응할 수 있는 다양한 방법이 있습니다. 다음 admin-respond-to-auth-challenge 예제에서는 diego@example.com에 대한 TOTP MFA 코드를 제공하고 로그인을 완료합니다. 이 사용자 풀에는 디바이스 기억 기능이 켜져 있으므로 인증 결과도 새 디바이스 키를 반환합니다.

```
aws cognito-idp admin-respond-to-auth-challenge \
    --user-pool-id us-west-2_EXAMPLE \
    --client-id 1example23456789 \
    --challenge-name SOFTWARE_TOKEN_MFA \
    --challenge-
responses USERNAME=diego@example.com,SOFTWARE_TOKEN_MFA_CODE=000000 \
    --session AYABeExample...
```

출력:

```
{
    "ChallengeParameters": {},
    "AuthenticationResult": {
        "AccessToken": "eyJra456defEXAMPLE",
        "ExpiresIn": 3600,
        "TokenType": "Bearer",
        "RefreshToken": "eyJra123abcEXAMPLE",
        "IdToken": "eyJra789ghiEXAMPLE",
        "NewDeviceMetadata": {
            "DeviceKey": "us-west-2_a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "DeviceGroupKey": "-ExAmPlE1"
        }
    }
}
```

자세한 내용은 Amazon Cognito 개발자 안내서의 관리자 인증 흐름 섹션을 참조하세요.

- API 세부 정보는 AWS CLI API 참조의 AdminRespondToAuthChallenge를 참조하세요.

Java

SDK for Java 2.x

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

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

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

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

       AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
               .adminRespondToAuthChallenge(respondToAuthChallengeRequest);

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

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

JavaScript

SDK for JavaScript (v3)

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

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

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

- API 세부 정보는 AWS SDK for JavaScript API 참조의 AdminRespondToAuthChallenge를 참조하십시오.

Kotlin

SDK for Kotlin

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

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

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

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

- API 세부 정보는 AWS SDK for Kotlin API 참조의 AdminRespondToAuthChallenge를 참조하십시오.

Python

SDK for Python (Boto3)

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

연결된 MFA 애플리케이션에서 생성한 코드를 제공하여 MFA 문제에 응답하세요.

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

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


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

        :param user_name: The name of the user who is signing in.
        :param session: Session information returned from a previous call to
 initiate
                        authentication.
```

```
        :param mfa_code: A code generated by the associated MFA application.
        :return: The result of the authentication. When successful, this contains
an
                 access token for the user.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "ChallengeName": "SOFTWARE_TOKEN_MFA",
                "Session": session,
                "ChallengeResponses": {
                    "USERNAME": user_name,
                    "SOFTWARE_TOKEN_MFA_CODE": mfa_code,
                },
            }
            if self.client_secret is not None:
                kwargs["ChallengeResponses"]["SECRET_HASH"] = self._secret_hash(
                    user_name
                )
            response =
self.cognito_idp_client.admin_respond_to_auth_challenge(**kwargs)
            auth_result = response["AuthenticationResult"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ExpiredCodeException":
                logger.warning(
                    "Your MFA code has expired or has been used already. You
might have "
                    "to wait a few seconds until your app shows you a new code."
                )
            else:
                logger.error(
                    "Couldn't respond to mfa challenge for %s. Here's why: %s:
%s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return auth_result
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 AdminRespondToAuthChallenge를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

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

```
    TRY.
        " Build challenge responses
        DATA(lt_challenge_responses) = VALUE /aws1/
cl_cgpchallengerspstyp00=>tt_challengeresponsestype(
          ( VALUE /aws1/cl_cgpchallengerspstyp00=>ts_challengerspstype_maprow(
              key = 'USERNAME'
              value = NEW /aws1/cl_cgpchallengerspstyp00( iv_user_name ) ) )
          ( VALUE /aws1/cl_cgpchallengerspstyp00=>ts_challengerspstype_maprow(
              key = 'SOFTWARE_TOKEN_MFA_CODE'
              value = NEW /aws1/cl_cgpchallengerspstyp00( iv_mfa_code ) ) )
        ).

        " Add SECRET_HASH if provided
        IF iv_secret_hash IS NOT INITIAL.
          INSERT VALUE #(
            key = 'SECRET_HASH'
            value = NEW /aws1/cl_cgpchallengerspstyp00( iv_secret_hash )
          ) INTO TABLE lt_challenge_responses.
        ENDIF.

        DATA(lo_result) = lo_cgp->adminrespondtoauthchallenge(
          iv_userpoolid = iv_user_pool_id
          iv_clientid = iv_client_id
          iv_challengename = 'SOFTWARE_TOKEN_MFA'
          it_challengeresponses = lt_challenge_responses
          iv_session = iv_session
        ).
```

```
        oo_auth_result = lo_result->get_authenticationresult( ).

        IF oo_auth_result IS BOUND.
          MESSAGE 'MFA challenge completed successfully.' TYPE 'I'.
        ELSE.
          " Another challenge might be required
          DATA(lv_next_challenge) = lo_result->get_challengename( ).
          MESSAGE |Additional challenge required: { lv_next_challenge }.| TYPE
'I'.
        ENDIF.

      CATCH /aws1/cx_cgpcodemismatchex INTO DATA(lo_code_ex).
        MESSAGE 'Invalid MFA code provided.' TYPE 'E'.

      CATCH /aws1/cx_cgpexpiredcodeex INTO DATA(lo_expired_ex).
        MESSAGE 'MFA code has expired.' TYPE 'E'.

      CATCH /aws1/cx_cgpnotauthorizedex INTO DATA(lo_auth_ex).
        MESSAGE 'Not authorized. Check MFA configuration.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 SDK for SAP ABAP API 참조의 AdminRespondToAuthChallenge를 참조하세요. AWS

Swift

SDK for Swift

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

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Respond to the authentication challenge received from Cognito after
    /// initiating an authentication session. This involves sending a current
    /// MFA code to the service.
```

```
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - userName: The user's username.
    ///   - clientId: The app client ID.
    ///   - userPoolId: The user pool to sign into.
    ///   - mfaCode: The 6-digit MFA code currently displayed by the user's
    ///     authenticator.
    ///   - session: The authentication session to continue processing.
    func adminRespondToAuthChallenge(cipClient: CognitoIdentityProviderClient,
userName: String,
                                     clientId: String, userPoolId: String,
mfaCode: String,
                                     session: String) async {
        print("=====> SOFTWARE_TOKEN_MFA challenge is generated...")

        var challengeResponsesOb: [String: String] = [:]
        challengeResponsesOb["USERNAME"] = userName
        challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

        do {
            let output = try await cipClient.adminRespondToAuthChallenge(
                input: AdminRespondToAuthChallengeInput(
                    challengeName:
CognitoIdentityProviderClientTypes.ChallengeNameType.softwareTokenMfa,
                    challengeResponses: challengeResponsesOb,
                    clientId: clientId,
                    session: session,
                    userPoolId: userPoolId
                )
            )

            guard let authenticationResult = output.authenticationResult else {
                print("*** Unable to get authentication result.")
                return
            }

            print("=====> Authentication result (JWTs are redacted):")
            print(authenticationResult)
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return
        } catch _ as CodeMismatchException {
            print("*** The specified MFA code doesn't match the expected value.")
```

```
            return
        } catch _ as UserNotFoundException {
            print("*** The specified username, \(userName), doesn't exist.")
            return
        } catch _ as UserNotConfirmedException {
            print("*** The user \(userName) has not been confirmed.")
            return
        } catch let error as NotAuthorizedException {
            print("*** Unauthorized access. Reason: \(error.properties.message ??
"<unknown>")")
        } catch {
            print("*** Error responding to the MFA challenge.")
            return
        }
    }
```

- API 세부 정보는 AWS SDK for Swift API 참조의 AdminRespondToAuthChallenge를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **AdminSetUserPassword** 함께 사용

다음 코드 예제는 AdminSetUserPassword의 사용 방법을 보여 줍니다.

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

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

CLI

AWS CLI

사용자 암호를 관리자로 설정하려면

다음 admin-set-user-password 예제에서는 diego@example.com의 암호를 영구적으로 설정합니다.

```
aws cognito-idp admin-set-user-password \
```

```
    --user-pool-id us-west-2_EXAMPLE \
    --username diego@example.com \
    --password MyExamplePassword1! \
    --permanent
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon Cognito 개발자 안내서의 암호, 암호 복구 및 암호 정책을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 AdminSetUserPassword를 참조하세요.

Go

SDK for Go V2

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

```
import (
 "context"
 "errors"
 "log"

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

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
```

```
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 AdminSetUserPassword를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **`AssociateSoftwareToken`** 함께 사용

다음 코드 예제는 `AssociateSoftwareToken`의 사용 방법을 보여 줍니다.

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

- MFA가 필요한 사용자 풀에 사용자 가입시키기

.NET

SDK for .NET

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

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

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

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

        return tokenResponse.Session;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 AssociateSoftwareToken을 참조하십시오.

C++

SDK for C++

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

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

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

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

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

        if (outcome.IsSuccess()) {
            std::cout
                    << "Enter this setup key into an authenticator app, for
 example Google Authenticator."
                    << std::endl;
            std::cout << "Setup key: " << outcome.GetResult().GetSecretCode()
                      << std::endl;
#ifdef USING_QR
            printAsterisksLine();
            std::cout << "\nOr scan the QR code in the file '" << QR_CODE_PATH <<
 "."
                      << std::endl;

            saveQRCode(std::string("otpauth://totp/") + userName + "?secret=" +
                       outcome.GetResult().GetSecretCode());
#endif // USING_QR
            session = outcome.GetResult().GetSession();
```

```
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::AssociateSoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 AssociateSoftwareToken을 참조하세요.

CLI

AWS CLI

MFA 인증 앱의 보안 키를 생성하려면

다음 `associate-software-token` 예제에서는 로그인하고 액세스 토큰을 받은 사용자를 위해 TOTP 프라이빗 키를 생성합니다. 생성된 프라이빗 키는 인증 앱에 수동으로 입력하거나 애플리케이션이 사용자가 스캔할 수 있는 QR 코드로 렌더링할 수 있습니다.

```
aws cognito-idp associate-software-token \
    --access-token eyJra456defEXAMPLE
```

출력:

```
{
    "SecretCode": "QWERTYUIOP123456EXAMPLE"
}
```

자세한 내용은 Amazon Cognito 개발자 안내서의 TOTP 소프트웨어 토큰 MFA를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 AssociateSoftwareToken을 참조하세요.

Java

SDK for Java 2.x

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

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

       AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
               .associateSoftwareToken(softwareTokenRequest);
       String secretCode = tokenResponse.secretCode();
       System.out.println("Enter this token into Google Authenticator");
       System.out.println(secretCode);
       return tokenResponse.session();
   }
```

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

JavaScript

SDK for JavaScript (v3)

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

```
const associateSoftwareToken = (session) => {
```

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

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

- API 세부 정보는 AWS SDK for JavaScript API 참조의 AssociateSoftwareToken을 참조하십시오.

Kotlin

SDK for Kotlin

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

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

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

- API 세부 정보는 AWS SDK for Kotlin API 참조의 AssociateSoftwareToken을 참조하십시오.

Python

SDK for Python (Boto3)

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

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

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


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

        :param session: Session information returned from a previous call to
 initiate
                        authentication.
        :return: An MFA token that can be used to set up an MFA application.
        """
        try:
            response =
 self.cognito_idp_client.associate_software_token(Session=session)
```

```
        except ClientError as err:
            logger.error(
                "Couldn't get MFA secret. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 AssociateSoftwareToken을 참조하세요.

SAP ABAP

SDK for SAP ABAP API

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

```
TRY.
    DATA(lo_result) = lo_cgp->associatesoftwaretoken(
      iv_session = iv_session
    ).

    ov_secret_code = lo_result->get_secretcode( ).

    MESSAGE 'MFA secret code generated successfully.' TYPE 'I'.

  CATCH /aws1/cx_cgpresourcenotfoundex INTO DATA(lo_ex).
    MESSAGE 'Session not found or expired.' TYPE 'E'.

  CATCH /aws1/cx_cgpnotauthorizedex INTO DATA(lo_auth_ex).
    MESSAGE 'Not authorized to associate software token.' TYPE 'E'.
ENDTRY.
```

- API 세부 정보는 SDK for SAP ABAP API 참조의 AssociateSoftwareToken을 참조하세요. AWS

Swift

SDK for Swift

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

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Request and display an MFA secret token that the user should enter
    /// into their authenticator to set it up for the user account.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - authSession: The authentication session to request an MFA secret
    ///     for.
    ///
    /// - Returns: A string containing the MFA secret token that should be
    ///   entered into the authenticator software.
    func getSecretForAppMFA(cipClient: CognitoIdentityProviderClient,
 authSession: String?) async -> String? {
        do {
            let output = try await cipClient.associateSoftwareToken(
                input: AssociateSoftwareTokenInput(
                    session: authSession
                )
            )

            guard let secretCode = output.secretCode else {
                print("*** Unable to get the secret code")
                return nil
            }
```

```
            print("=====> Enter this token into Google Authenticator:
\(secretCode)")
            return output.session
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return nil
        } catch {
            print("*** An unexpected error occurred getting the secret for the
app's MFA.")
            return nil
        }
    }
```

- API 세부 정보는 AWS SDK for Swift API 참조의 AssociateSoftwareToken을 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **ConfirmDevice** 함께 사용

다음 코드 예제는 ConfirmDevice의 사용 방법을 보여 줍니다.

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

- MFA가 필요한 사용자 풀에 사용자 가입시키기

.NET

SDK for .NET

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

```
    /// <summary>
    /// Initiates and confirms tracking of the device.
```

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

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

- API 세부 정보는 AWS SDK for .NET API 참조의 ConfirmDevice를 참조하세요.

CLI

AWS CLI

사용자 디바이스를 확인하려면

다음 `confirm-device` 예제에서는 현재 사용자에 대해 기억된 새 디바이스를 추가합니다.

```
aws cognito-idp confirm-device \
   --access-token eyJra456defEXAMPLE \
   --device-key us-west-2_a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \
   --device-secret-verifier-
config PasswordVerifier=TXlWZXJpZmllclN0cmluZw,Salt=TXlTUlBTYWx0
```

출력:

```
{
     "UserConfirmationNecessary": false
```

```
}
```

자세한 내용은 Amazon Cognito 개발자 안내서의 사용자 풀에서 사용자 디바이스 작업 섹션을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조에서 ConfirmDevice를 참조하세요.

JavaScript

SDK for JavaScript (v3)

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

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

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

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

- API 세부 정보는 AWS SDK for JavaScript API 참조의 ConfirmDevice를 참조하십시오.

Python

SDK for Python (Boto3)

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

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

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


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

```
        MFA code from the MFA application.

        :param user_name: The user that is associated with the device.
        :param device_key: The key of the device, returned by Amazon Cognito.
        :param device_group_key: The group key of the device, returned by Amazon
Cognito.
        :param device_password: The password that is associated with the device.
        :param access_token: The user's access token.
        :param aws_srp: A class that helps with Secure Remote Password (SRP)
                        calculations. The scenario associated with this example
uses
                        the warrant package.
        :return: True when the user must confirm the device. Otherwise, False.
When
                 False, the device is automatically confirmed and tracked.
        """
        srp_helper = aws_srp.AWSSRP(
            username=user_name,
            password=device_password,
            pool_id="_",
            client_id=self.client_id,
            client_secret=None,
            client=self.cognito_idp_client,
        )
        device_and_pw = f"{device_group_key}{device_key}:{device_password}"
        device_and_pw_hash = aws_srp.hash_sha256(device_and_pw.encode("utf-8"))
        salt = aws_srp.pad_hex(aws_srp.get_random(16))
        x_value = aws_srp.hex_to_long(aws_srp.hex_hash(salt +
device_and_pw_hash))
        verifier = aws_srp.pad_hex(pow(srp_helper.val_g, x_value,
srp_helper.big_n))
        device_secret_verifier_config = {
            "PasswordVerifier": base64.standard_b64encode(
                bytearray.fromhex(verifier)
            ).decode("utf-8"),
            "Salt":
base64.standard_b64encode(bytearray.fromhex(salt)).decode("utf-8"),
        }
        try:
            response = self.cognito_idp_client.confirm_device(
                AccessToken=access_token,
                DeviceKey=device_key,
                DeviceSecretVerifierConfig=device_secret_verifier_config,
            )
```

```
            user_confirm = response["UserConfirmationNecessary"]
        except ClientError as err:
            logger.error(
                "Couldn't confirm mfa device %s. Here's why: %s: %s",
                device_key,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return user_confirm
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 ConfirmDevice를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **`ConfirmForgotPassword`** 함께 사용

다음 코드 예제는 `ConfirmForgotPassword`의 사용 방법을 보여 줍니다.

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

- Lambda 함수를 사용하여 알려진 사용자를 자동으로 마이그레이션

CLI

AWS CLI

잊어버린 암호를 확인하는 방법

이 예시에서는 사용자 이름 diego@example.com의 잊어버린 암호를 확인합니다.

명령:

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

- API 세부 정보는 AWS CLI 명령 참조의 ConfirmForgotPassword를 참조하세요.

Go

SDK for Go V2

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

```
import (
 "context"
 "errors"
 "log"

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

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
```

```
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 ConfirmForgotPassword를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **ConfirmSignUp** 함께 사용

다음 코드 예제는 ConfirmSignUp의 사용 방법을 보여 줍니다.

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

- MFA가 필요한 사용자 풀에 사용자 가입시키기

.NET

SDK for .NET

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

```
    /// <summary>
    /// Confirm that the user has signed up.
    /// </summary>
    /// <param name="clientId">The Id of this application.</param>
    /// <param name="code">The confirmation code sent to the user.</param>
```

```
    /// <param name="userName">The username.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ConfirmSignupAsync(string clientId, string code,
string userName)
    {
        var signUpRequest = new ConfirmSignUpRequest
        {
            ClientId = clientId,
            ConfirmationCode = code,
            Username = userName,
        };

        var response = await _cognitoService.ConfirmSignUpAsync(signUpRequest);
        if (response.HttpStatusCode == HttpStatusCode.OK)
        {
            Console.WriteLine($"{userName} was confirmed");
            return true;
        }
        return false;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 ConfirmSignUp을 참조하십시오.

C++

SDK for C++

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

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

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

```
        Aws::CognitoIdentityProvider::Model::ConfirmSignUpRequest request;
        request.SetClientId(clientID);
        request.SetConfirmationCode(confirmationCode);
        request.SetUsername(userName);

        Aws::CognitoIdentityProvider::Model::ConfirmSignUpOutcome outcome =
                client.ConfirmSignUp(request);

        if (outcome.IsSuccess()) {
            std::cout << "ConfirmSignup was Successful."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::ConfirmSignUp. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 ConfirmSignUp을 참조하세요.

CLI

AWS CLI

가입 확인

이 예시는 사용자 이름 diego@example.com 가입을 확인합니다.

명령:

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

- API 세부 정보는 AWS CLI 명령 참조에서 ConfirmSignUp을 참조하세요.

Java

SDK for Java 2.x

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

```
   public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
           String userName) {
       try {
           ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
                   .clientId(clientId)
                   .confirmationCode(code)
                   .username(userName)
                   .build();

           identityProviderClient.confirmSignUp(signUpRequest);
           System.out.println(userName + " was confirmed");

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

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

JavaScript

SDK for JavaScript (v3)

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

```
const confirmSignUp = ({ clientId, username, code }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ConfirmSignUpCommand({
    ClientId: clientId,
    Username: username,
    ConfirmationCode: code,
  });

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

- API 세부 정보는 AWS SDK for JavaScript API 참조의 ConfirmSignUp을 참조하십시오.

Kotlin

SDK for Kotlin

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

```
suspend fun confirmSignUp(
    clientIdVal: String?,
    codeVal: String?,
    userNameVal: String?,
) {
    val signUpRequest =
        ConfirmSignUpRequest {
            clientId = clientIdVal
            confirmationCode = codeVal
            username = userNameVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
```

```
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ConfirmSignUp을 참조하십시오.

Python

SDK for Python (Boto3)

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

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

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


    def confirm_user_sign_up(self, user_name, confirmation_code):
        """
        Confirms a previously created user. A user must be confirmed before they
        can sign in to Amazon Cognito.

        :param user_name: The name of the user to confirm.
```

```
        :param confirmation_code: The confirmation code sent to the user's
 registered
                                  email address.
        :return: True when the confirmation succeeds.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "ConfirmationCode": confirmation_code,
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            self.cognito_idp_client.confirm_sign_up(**kwargs)
        except ClientError as err:
            logger.error(
                "Couldn't confirm sign up for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return True
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 ConfirmSignUp을 참조하세요.

Swift

SDK for Swift

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

```
import AWSClientRuntime
import AWSCognitoIdentityProvider
```

```
    /// Submit a confirmation code for the specified user. This is the code as
    /// entered by the user after they've received it by email or text
    /// message.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The app client ID the user is signing up for.
    ///   - userName: The username of the user whose code is being sent.
    ///   - code: The user's confirmation code.
    ///
    /// - Returns: `true` if the code was successfully confirmed; otherwise
`false`.
    func confirmSignUp(cipClient: CognitoIdentityProviderClient, clientId:
String,
                       userName: String, code: String) async -> Bool {
        do {
            _ = try await cipClient.confirmSignUp(
                input: ConfirmSignUpInput(
                    clientId: clientId,
                    confirmationCode: code,
                    username: userName
                )
            )

            print("=====> \(userName) has been confirmed.")
            return true
        } catch {
            print("=====> \(userName)'s code was entered incorrectly.")
            return false
        }
    }
```

- API 세부 정보는 AWS SDK for Swift API 참조의 ConfirmSignUp을 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **`CreateUserPool`** 함께 사용

다음 코드 예제는 `CreateUserPool`의 사용 방법을 보여 줍니다.

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

- 사용자 풀 시작하기

CLI

AWS CLI

최소 구성 사용자 풀 생성

이 예시에서는 기본값을 사용하여 MyUserPool이라는 사용자 풀을 만듭니다. 필수 속성도 없고 애플리케이션 클라이언트도 없습니다. MFA 및 고급 보안이 비활성화되었습니다.

명령:

```
aws cognito-idp create-user-pool --pool-name MyUserPool
```

출력:

```
{
  "UserPool": {
      "SchemaAttributes": [
          {
              "Name": "sub",
              "StringAttributeConstraints": {
                  "MinLength": "1",
                  "MaxLength": "2048"
              },
              "DeveloperOnlyAttribute": false,
              "Required": true,
              "AttributeDataType": "String",
              "Mutable": false
          },
          {
              "Name": "name",
              "StringAttributeConstraints": {
                  "MinLength": "0",
                  "MaxLength": "2048"
              },
              "DeveloperOnlyAttribute": false,
              "Required": false,
```

```
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "given_name",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "family_name",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "middle_name",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "nickname",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
```

```
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "preferred_username",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "profile",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "picture",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "website",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
```

```
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "email",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "AttributeDataType": "Boolean",
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "Name": "email_verified",
                "Mutable": true
            },
            {
                "Name": "gender",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "birthdate",
                "StringAttributeConstraints": {
                    "MinLength": "10",
                    "MaxLength": "10"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
```

```
                "Name": "zoneinfo",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "locale",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "phone_number",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "AttributeDataType": "Boolean",
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "Name": "phone_number_verified",
                "Mutable": true
            },
            {
                "Name": "address",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
```

```
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "updated_at",
            "NumberAttributeConstraints": {
                "MinValue": "0"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "Number",
            "Mutable": true
        }
    ],
    "MfaConfiguration": "OFF",
    "Name": "MyUserPool",
    "LastModifiedDate": 1547833345.777,
    "AdminCreateUserConfig": {
        "UnusedAccountValidityDays": 7,
        "AllowAdminCreateUserOnly": false
    },
    "EmailConfiguration": {},
    "Policies": {
        "PasswordPolicy": {
            "RequireLowercase": true,
            "RequireSymbols": true,
            "RequireNumbers": true,
            "MinimumLength": 8,
            "RequireUppercase": true
        }
    },
    "CreationDate": 1547833345.777,
    "EstimatedNumberOfUsers": 0,
    "Id": "us-west-2_aaaaaaaaa",
    "LambdaConfig": {}
  }
}
```

두 개의 필수 속성으로 사용자 풀을 생성하는 방법

이 예시에서는 사용자 풀 MyUserPool을 생성합니다. 풀은 이메일을 사용자 이름 속성으로 받아들이도록 구성되어 있습니다. 또한 Amazon Simple Email Service를 사용하여 이메일 소스 주소를 검증된 주소로 설정합니다.

명령:

```
aws cognito-idp create-user-pool --pool-name MyUserPool --username-
attributes "email" --email-configuration=SourceArn="arn:aws:ses:us-
east-1:111111111111:identity/
jane@example.com",ReplyToEmailAddress="jane@example.com"
```

출력:

```
{
  "UserPool": {
      "SchemaAttributes": [
          {
              "Name": "sub",
              "StringAttributeConstraints": {
                  "MinLength": "1",
                  "MaxLength": "2048"
              },
              "DeveloperOnlyAttribute": false,
              "Required": true,
              "AttributeDataType": "String",
              "Mutable": false
          },
          {
              "Name": "name",
              "StringAttributeConstraints": {
                  "MinLength": "0",
                  "MaxLength": "2048"
              },
              "DeveloperOnlyAttribute": false,
              "Required": false,
              "AttributeDataType": "String",
              "Mutable": true
          },
          {
              "Name": "given_name",
              "StringAttributeConstraints": {
                  "MinLength": "0",
                  "MaxLength": "2048"
```

```
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "family_name",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "middle_name",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "nickname",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "preferred_username",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
```

```
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "profile",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "picture",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "website",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "email",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
```

```
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "AttributeDataType": "Boolean",
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "Name": "email_verified",
            "Mutable": true
        },
        {
            "Name": "gender",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "birthdate",
            "StringAttributeConstraints": {
                "MinLength": "10",
                "MaxLength": "10"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "zoneinfo",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
```

```
            "Mutable": true
        },
        {
            "Name": "locale",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "phone_number",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "AttributeDataType": "Boolean",
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "Name": "phone_number_verified",
            "Mutable": true
        },
        {
            "Name": "address",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "updated_at",
```

```
                "NumberAttributeConstraints": {
                    "MinValue": "0"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "Number",
                "Mutable": true
            }
        ],
        "MfaConfiguration": "OFF",
        "Name": "MyUserPool",
        "LastModifiedDate": 1547837788.189,
        "AdminCreateUserConfig": {
            "UnusedAccountValidityDays": 7,
            "AllowAdminCreateUserOnly": false
        },
        "EmailConfiguration": {
            "ReplyToEmailAddress": "jane@example.com",
            "SourceArn": "arn:aws:ses:us-east-1:111111111111:identity/
jane@example.com"
        },
        "Policies": {
            "PasswordPolicy": {
                "RequireLowercase": true,
                "RequireSymbols": true,
                "RequireNumbers": true,
                "MinimumLength": 8,
                "RequireUppercase": true
            }
        },
        "UsernameAttributes": [
            "email"
        ],
        "CreationDate": 1547837788.189,
        "EstimatedNumberOfUsers": 0,
        "Id": "us-west-2_aaaaaaaaa",
        "LambdaConfig": {}
    }
}
```

- API 세부 정보는 AWS CLI 명령 참조의 CreateUserPool을 참조하세요.

Java

SDK for Java 2.x

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

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateUserPool {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <userPoolName>\s

                Where:
                    userPoolName - The name to give your user pool when it's
 created.
                """;
```

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

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

        String id = createPool(cognitoClient, userPoolName);
        System.out.println("User pool ID: " + id);
        cognitoClient.close();
    }

    public static String createPool(CognitoIdentityProviderClient cognitoClient,
 String userPoolName) {
        try {
            CreateUserPoolRequest request = CreateUserPoolRequest.builder()
                    .poolName(userPoolName)
                    .build();

            CreateUserPoolResponse response =
 cognitoClient.createUserPool(request);
            return response.userPool().id();

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

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateUserPool 섹션을 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **CreateUserPoolClient** 함께 사용

다음 코드 예제는 CreateUserPoolClient의 사용 방법을 보여 줍니다.

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

- 사용자 풀 시작하기

CLI

AWS CLI

사용자 풀 클라이언트 생성

다음 create-user-pool-client 예제에서는 클라이언트 보안 암호, 명시적 읽기 및 쓰기 속성, 사용자 이름 암호 및 SRP 흐름으로 로그인, IdP 3개로 로그인, OAuth 범위의 하위 집합에 대한 액세스, PinPoint 분석 및 확장된 인증 세션 유효성이 있는 새 사용자 풀 클라이언트를 생성합니다.

```
aws cognito-idp create-user-pool-client \
    --user-pool-id us-west-2_EXAMPLE \
    --client-name MyTestClient \
    --generate-secret \
    --refresh-token-validity 10 \
    --access-token-validity 60 \
    --id-token-validity 60 \
    --token-validity-units AccessToken=minutes,IdToken=minutes,RefreshToken=days
 \
    --read-attributes email phone_number email_verified phone_number_verified \
    --write-attributes email phone_number \
    --explicit-auth-
flows ALLOW_USER_PASSWORD_AUTH ALLOW_USER_SRP_AUTH ALLOW_REFRESH_TOKEN_AUTH \
    --supported-identity-providers Google Facebook MyOIDC \
    --callback-urls https://www.amazon.com https://example.com http://
localhost:8001 myapp://example \
    --allowed-o-auth-flows code implicit \
    --allowed-o-auth-scopes openid profile aws.cognito.signin.user.admin solar-
system-data/asteroids.add \
    --allowed-o-auth-flows-user-pool-client \
    --analytics-configuration ApplicationArn=arn:aws:mobiletargeting:us-
west-2:767671399759:apps/thisisanexamplepinpointapplicationid,UserDataShared=TRUE
 \
```

```
    --prevent-user-existence-errors ENABLED \
    --enable-token-revocation \
    --enable-propagate-additional-user-context-data \
    --auth-session-validity 4
```

출력:


```
{
    "UserPoolClient": {
        "UserPoolId": "us-west-2_EXAMPLE",
        "ClientName": "MyTestClient",
        "ClientId": "123abc456defEXAMPLE",
        "ClientSecret": "this1234is5678my91011example1213client1415secret",
        "LastModifiedDate": 1726788459.464,
        "CreationDate": 1726788459.464,
        "RefreshTokenValidity": 10,
        "AccessTokenValidity": 60,
        "IdTokenValidity": 60,
        "TokenValidityUnits": {
            "AccessToken": "minutes",
            "IdToken": "minutes",
            "RefreshToken": "days"
        },
        "ReadAttributes": [
            "email_verified",
            "phone_number_verified",
            "phone_number",
            "email"
        ],
        "WriteAttributes": [
            "phone_number",
            "email"
        ],
        "ExplicitAuthFlows": [
            "ALLOW_USER_PASSWORD_AUTH",
            "ALLOW_USER_SRP_AUTH",
            "ALLOW_REFRESH_TOKEN_AUTH"
        ],
        "SupportedIdentityProviders": [
            "Google",
            "MyOIDC",
            "Facebook"
        ],
```

```
        "CallbackURLs": [
            "https://example.com",
            "https://www.amazon.com",
            "myapp://example",
            "http://localhost:8001"
        ],
        "AllowedOAuthFlows": [
            "implicit",
            "code"
        ],
        "AllowedOAuthScopes": [
            "aws.cognito.signin.user.admin",
            "openid",
            "profile",
            "solar-system-data/asteroids.add"
        ],
        "AllowedOAuthFlowsUserPoolClient": true,
        "AnalyticsConfiguration": {
            "ApplicationArn": "arn:aws:mobiletargeting:us-
west-2:123456789012:apps/thisisanexamplepinpointapplicationid",
            "RoleArn": "arn:aws:iam::123456789012:role/aws-service-role/cognito-
idp.amazonaws.com/AWSServiceRoleForAmazonCognitoIdp",
            "UserDataShared": true
        },
        "PreventUserExistenceErrors": "ENABLED",
        "EnableTokenRevocation": true,
        "EnablePropagateAdditionalUserContextData": true,
        "AuthSessionValidity": 4
    }
}
```

자세한 내용은 Amazon Cognito 개발자 안내서의 앱 클라이언트를 사용한 애플리케이션별 설정을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조에서 CreateUserPoolClient를 참조하세요.

Java

SDK for Java 2.x

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

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

/**
 * A user pool client app is an application that authenticates with Amazon
 * Cognito user pools.
 * When you create a user pool, you can configure app clients that allow mobile
 * or web applications
 * to call API operations to authenticate users, manage user attributes and
 * profiles,
 * and implement sign-up and sign-in flows.
 *
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateUserPoolClient {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <clientName> <userPoolId>\s
```

```
                Where:
                    clientName - The name for the user pool client to create.
                    userPoolId - The ID for the user pool.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clientName = args[0];
        String userPoolId = args[1];
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createPoolClient(cognitoClient, clientName, userPoolId);
        cognitoClient.close();
    }

    public static void createPoolClient(CognitoIdentityProviderClient
 cognitoClient, String clientName,
            String userPoolId) {
        try {
            CreateUserPoolClientRequest request =
 CreateUserPoolClientRequest.builder()
                    .clientName(clientName)
                    .userPoolId(userPoolId)
                    .build();

            CreateUserPoolClientResponse response =
 cognitoClient.createUserPoolClient(request);
            System.out.println("User pool " +
 response.userPoolClient().clientName() + " created. ID: "
                    + response.userPoolClient().clientId());

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

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateUserPoolClient 섹션을 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **`DeleteUser`** 함께 사용

다음 코드 예제는 `DeleteUser`의 사용 방법을 보여 줍니다.

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

- Lambda 함수를 사용하여 알려진 사용자를 자동으로 확인
- Lambda 함수를 사용하여 알려진 사용자를 자동으로 마이그레이션
- Amazon Cognito 사용자 인증 후 Lambda 함수를 사용하여 사용자 지정 활동 데이터 작성

C++

SDK for C++

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

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

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

        Aws::CognitoIdentityProvider::Model::DeleteUserRequest request;
        request.SetAccessToken(accessToken);
```

```
        Aws::CognitoIdentityProvider::Model::DeleteUserOutcome outcome =
                client.DeleteUser(request);

        if (outcome.IsSuccess()) {
            std::cout << "The user " << userName << " was deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::DeleteUser. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 DeleteUser를 참조하세요.

CLI

AWS CLI

사용자 삭제

이 예제에서는 사용자를 삭제합니다.

명령:

```
aws cognito-idp delete-user --access-token ACCESS_TOKEN
```

- API 세부 정보는 AWS CLI 명령 참조의 DeleteUser를 참조하세요.

Go

SDK for Go V2

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

```
import (
 "context"
 "errors"
 "log"

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

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken
 string) error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 DeleteUser를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

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

```
/**
 * Delete the signed-in user. Useful for allowing a user to delete their
 * own profile.
 * @param {{ region: string, accessToken: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").DeleteUserCommandOutput | null, unknown]>}
 */
export const deleteUser = async ({ region, accessToken }) => {
  try {
    const client = new CognitoIdentityProviderClient({ region });
    const response = await client.send(
      new DeleteUserCommand({ AccessToken: accessToken }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 DeleteUser를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 **DescribeTermsByClient** 함께 사용

다음 코드 예제는 DescribeTermsByClient의 사용 방법을 보여 줍니다.

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

- 클라이언트 토큰 및 용어 검색

Python

SDK for Python (Boto3)

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

```
    def describe_terms_by_client(
        self, user_pool_id: str, client_id: str, terms_name: str
    ) -> dict[str, Any]:
        """
        Looks up a Terms document associated with an app client by client ID,
        user pool ID, and terms name.

        :param user_pool_id: The ID of the user pool that contains the Terms
documents.
        :param client_id: The ID of the app client associated with the Terms.
        :param terms_name: The name of the terms document ('terms-of-use' or
                           'privacy-policy').
        :return: A dictionary with the Terms document details.
        """
        try:
            response = self.cognito_idp_client.describe_terms_by_client(
                UserPoolId=user_pool_id,
                ClientId=client_id,
                TermsName=terms_name,
            )
            terms = response.get("Terms", dict())
            logger.info(
                "Described terms '%s' for client %s in user pool %s.",
                terms_name,
                client_id,
                user_pool_id,
            )
            return terms
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "No terms documents found for client %s, user pool %s, "
```

```
                    "and terms name '%s'.",
                    client_id,
                    user_pool_id,
                    terms_name,
                )
            raise
```

- API 세부 정보는 SDK for Python (Boto3) API 참조의 DescribeTermsByClient를 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **`DescribeUserPoolClient`** 함께 사용

다음 코드 예제는 `DescribeUserPoolClient`의 사용 방법을 보여 줍니다.

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

- 클라이언트 토큰 및 용어 검색
- 사용자 풀 시작하기

CLI

AWS CLI

사용자 풀 클라이언트를 설명하는 방법

이 예시에서는 사용자 풀 클라이언트를 설명합니다.

명령:

```
aws cognito-idp describe-user-pool-client --user-pool-id us-west-2_aaaaaaaaa --
client-id 38fjsnc484p94kpqsnet7mpld0
```

출력:

```
{
```

```
  "UserPoolClient": {
      "UserPoolId": "us-west-2_aaaaaaaaa",
      "ClientName": "MyApp",
      "ClientId": "38fjsnc484p94kpqsnet7mpld0",
      "ClientSecret": "CLIENT_SECRET",
      "LastModifiedDate": 1548108676.163,
      "CreationDate": 1548108676.163,
      "RefreshTokenValidity": 30,
      "ReadAttributes": [
          "address",
          "birthdate",
          "custom:CustomAttr1",
          "custom:CustomAttr2",
          "email",
          "email_verified",
          "family_name",
          "gender",
          "given_name",
          "locale",
          "middle_name",
          "name",
          "nickname",
          "phone_number",
          "phone_number_verified",
          "picture",
          "preferred_username",
          "profile",
          "updated_at",
          "website",
          "zoneinfo"
      ],
      "WriteAttributes": [
          "address",
          "birthdate",
          "custom:CustomAttr1",
          "custom:CustomAttr2",
          "email",
          "family_name",
          "gender",
          "given_name",
          "locale",
          "middle_name",
          "name",
          "nickname",
```

```
            "phone_number",
            "picture",
            "preferred_username",
            "profile",
            "updated_at",
            "website",
            "zoneinfo"
        ],
        "ExplicitAuthFlows": [
            "ADMIN_NO_SRP_AUTH",
            "USER_PASSWORD_AUTH"
        ],
        "AllowedOAuthFlowsUserPoolClient": false
    }
}
```

- API 세부 정보는 AWS CLI 명령 참조의 DescribeUserPoolClient 섹션을 참조하세요.

Python

SDK for Python(Boto3)

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

```
    def describe_user_pool_client(
        self, user_pool_id: str, client_id: str
    ) -> dict[str, Any]:
        """
        Describes an app client in a user pool, including the client secret.

        :param user_pool_id: The ID of the user pool that contains the app
 client.
        :param client_id: The ID of the app client to describe.
        :return: A dictionary with the app client configuration.
        """
        try:
            response = self.cognito_idp_client.describe_user_pool_client(
                UserPoolId=user_pool_id, ClientId=client_id
```

```
            )
            client_info = response.get("UserPoolClient", dict())
            logger.info(
                "Described app client %s in user pool %s.",
                client_id,
                user_pool_id,
            )
            return client_info
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "The user pool %s or client %s does not exist. "
                    "Verify the stack deployed successfully and the outputs are
correct.",
                    user_pool_id,
                    client_id,
                )
            raise
```

- API 세부 정보는 SDK for Python (Boto3) API 참조의 DescribeUserPoolClient를 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **ForgotPassword** 함께 사용

다음 코드 예제는 ForgotPassword의 사용 방법을 보여 줍니다.

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

- Lambda 함수를 사용하여 알려진 사용자를 자동으로 마이그레이션

CLI

AWS CLI

암호를 강제로 변경하는 방법

다음 `forgot-password` 예시에서는 암호를 변경하라는 메시지를 jane@example.com으로 보냅니다.

```
aws cognito-idp forgot-password --client-id 38fjsnc484p94kpqsnet7mpld0 --
username jane@example.com
```

출력:

```
{
    "CodeDeliveryDetails": {
        "Destination": "j***@e***.com",
        "DeliveryMedium": "EMAIL",
        "AttributeName": "email"
    }
}
```

- API 세부 정보는 AWS CLI 명령 참조의 ForgotPassword를 참조하세요.

Go

SDK for Go V2

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

```
import (
 "context"
 "errors"
 "log"

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

type CognitoActions struct {
```

```
 CognitoClient *cognitoidentityprovider.Client
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 ForgotPassword를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 **GetClientToken** 함께 사용

다음 코드 예제는 GetClientToken의 사용 방법을 보여 줍니다.

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

- 클라이언트 토큰 및 용어 검색

Python

SDK for Python (Boto3)

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

```
    def get_client_token(
        self,
        client_id: str,
        client_secret: str,
        scopes: Optional[list[str]] = None,
    ) -> dict[str, Any]:
        """
        Obtains an M2M access token for an app client using its client ID
        and secret, without end-user sign-in.

        :param client_id: The ID of the app client.
        :param client_secret: The client secret for the app client.
        :param scopes: Optional list of scopes in the format
                       'resource-server-identifier/scope-name'. If omitted,
                       all scopes configured for the app client are authorized.
        :return: A dictionary containing the ClientAuthenticationResult with
                 AccessToken, ExpiresIn, and TokenType.
        """
        try:
            params = dict()
            params["ClientId"] = client_id
            params["Secret"] = client_secret
            if scopes is not None:
                params["Scopes"] = scopes

            response = self.cognito_idp_client.get_client_token(**params)
            auth_result = response.get("ClientAuthenticationResult", dict())
            logger.info(
                "Obtained M2M access token for client %s. Token type: %s, expires
in: %s seconds.",
                client_id,
                auth_result.get("TokenType", "Unknown"),
```

```
                auth_result.get("ExpiresIn", 0),
            )
            return auth_result
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotAuthorizedException":
                logger.error(
                    "Not authorized for client %s. Verify that the client ID and
"
                    "secret are correct, and that the ALLOW_CLIENT_TOKEN_AUTH
flow "
                    "is enabled for the app client.",
                    client_id,
                )
            raise
```

- API 세부 정보는 SDK for Python (Boto3) API 참조의 GetClientToken을 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **InitiateAuth** 함께 사용

다음 코드 예제는 InitiateAuth의 사용 방법을 보여 줍니다.

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

- Lambda 함수를 사용하여 알려진 사용자를 자동으로 확인
- Lambda 함수를 사용하여 알려진 사용자를 자동으로 마이그레이션
- 사용자 풀 시작하기
- MFA가 필요한 사용자 풀에 사용자 가입시키기
- Amazon Cognito 사용자 인증 후 Lambda 함수를 사용하여 사용자 지정 활동 데이터 작성

.NET

SDK for .NET

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

```
    /// <summary>
    /// Initiate authorization.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The name of the user who is authenticating.</
param>
    /// <param name="password">The password for the user who is authenticating.</
param>
    /// <returns>The response from the initiate auth request.</returns>
    public async Task<InitiateAuthResponse> InitiateAuthAsync(string clientId,
 string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var authRequest = new InitiateAuthRequest

        {
            ClientId = clientId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.InitiateAuthAsync(authRequest);
        Console.WriteLine($"Result Challenge is : {response.ChallengeName}");

        return response;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 InitiateAuth를 참조하세요.

CLI

AWS CLI

사용자 로그인

다음 initiate-auth 예제에서는 기본 사용자 이름-암호 흐름으로 사용자를 로그인하며 추가 문제는 없습니다.

```
aws cognito-idp initiate-auth \
    --auth-flow USER_PASSWORD_AUTH \
    --client-id 1example23456789 \
    --analytics-metadata AnalyticsEndpointId=d70b2ba36a8c4dc5a04a0451aEXAMPLE \
    --auth-parameters USERNAME=testuser,PASSWORD=[Password] --user-context-
data EncodedData=mycontextdata --client-metadata MyTestKey=MyTestValue
```

출력:

```
{
    "AuthenticationResult": {
        "AccessToken": "eyJra456defEXAMPLE",
        "ExpiresIn": 3600,
        "TokenType": "Bearer",
        "RefreshToken": "eyJra123abcEXAMPLE",
        "IdToken": "eyJra789ghiEXAMPLE",
        "NewDeviceMetadata": {
            "DeviceKey": "us-west-2_a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "DeviceGroupKey": "-v7w9UcY6"
        }
    }
}
```

자세한 내용은 Amazon Cognito 개발자 안내서의 Authentication을 참조하시기 바랍니다.

- API 세부 정보는 AWS CLI 명령 참조의 InitiateAuth를 참조하시기 바랍니다.

Go

SDK for Go V2

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

```
import (
 "context"
 "errors"
 "log"

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

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
```

```
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 InitiateAuth를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

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

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

  const command = new InitiateAuthCommand({
    AuthFlow: AuthFlowType.USER_PASSWORD_AUTH,
    AuthParameters: {
      USERNAME: username,
      PASSWORD: password,
    },
    ClientId: clientId,
  });

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

- API 세부 정보는 AWS SDK for JavaScript API 참조의 InitiateAuth를 참조하십시오.

Python

SDK for Python (Boto3)

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

이 예시에서는 추적된 디바이스로 인증을 시작하는 방법을 보여줍니다. 로그인을 완료하려면 클라이언트가 보안 원격 암호(SRP) 문제에 올바르게 응답해야 합니다.

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

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


    def sign_in_with_tracked_device(
        self,
        user_name,
        password,
        device_key,
        device_group_key,
        device_password,
        aws_srp,
    ):
        """
```

```
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA
 code.

        Signing in with a tracked device requires that the client respond to the
 SRP
        protocol. The scenario associated with this example uses the warrant
 package
        to help with SRP calculations.

        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains
 an
                 access token for the user.
        """
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name,
                password=device_password,
                pool_id="_",
                client_id=self.client_id,
                client_secret=None,
                client=self.cognito_idp_client,
            )

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id,
                AuthFlow="USER_PASSWORD_AUTH",
                AuthParameters={
                    "USERNAME": user_name,
                    "PASSWORD": password,
                    "DEVICE_KEY": device_key,
                },
            )
            if response_init["ChallengeName"] != "DEVICE_SRP_AUTH":
```

```
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
{response_init['ChallengeName']}."
                )

            auth_params = srp_helper.get_auth_params()
            auth_params["DEVICE_KEY"] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_SRP_AUTH",
                ChallengeResponses=auth_params,
            )
            if response_auth["ChallengeName"] != "DEVICE_PASSWORD_VERIFIER":
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}."
                )

            challenge_params = response_auth["ChallengeParameters"]
            challenge_params["USER_ID_FOR_SRP"] = device_group_key + device_key
            cr = srp_helper.process_challenge(challenge_params, {"USERNAME":
user_name})
            cr["USERNAME"] = user_name
            cr["DEVICE_KEY"] = device_key
            response_verifier =
self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_PASSWORD_VERIFIER",
                ChallengeResponses=cr,
            )
            auth_tokens = response_verifier["AuthenticationResult"]
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return auth_tokens
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 InitiateAuth를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **`ListUserPools`** 함께 사용

다음 코드 예제는 `ListUserPools`의 사용 방법을 보여 줍니다.

.NET

SDK for .NET (v4)

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

```
    /// <summary>
    /// List the Amazon Cognito user pools for an account.
    /// </summary>
    /// <returns>A list of UserPoolDescriptionType objects.</returns>
    public async Task<List<UserPoolDescriptionType>> ListUserPoolsAsync()
    {
        var userPools = new List<UserPoolDescriptionType>();

        var userPoolsPaginator = _cognitoService.Paginators.ListUserPools(new
ListUserPoolsRequest());

        await foreach (var response in userPoolsPaginator.Responses)
        {
            userPools.AddRange(response.UserPools);
        }

        return userPools;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 ListUserPools를 참조하세요.

CLI

AWS CLI

사용자 풀 나열

다음 `list-user-pools` 예제에서는 현재 CLI 자격 증명의 AWS 계정에서 사용 가능한 사용자 풀 3개를 나열합니다.

```
aws cognito-idp list-user-pools \
    --max-results 3
```

출력:

```
{
    "NextToken": "[Pagination token]",
    "UserPools": [
        {
            "CreationDate": 1681502497.741,
            "Id": "us-west-2_EXAMPLE1",
            "LambdaConfig": {
                "CustomMessage": "arn:aws:lambda:us-
east-1:123456789012:function:MyFunction",
                "PreSignUp": "arn:aws:lambda:us-
east-1:123456789012:function:MyFunction",
                "PreTokenGeneration": "arn:aws:lambda:us-
east-1:123456789012:function:MyFunction",
                "PreTokenGenerationConfig": {
                    "LambdaArn": "arn:aws:lambda:us-
east-1:123456789012:function:MyFunction",
                    "LambdaVersion": "V1_0"
                }
            },
            "LastModifiedDate": 1681502497.741,
            "Name": "user pool 1"
        },
        {
            "CreationDate": 1686064178.717,
            "Id": "us-west-2_EXAMPLE2",
            "LambdaConfig": {
            },
            "LastModifiedDate": 1686064178.873,
            "Name": "user pool 2"
```

```
        },
        {
            "CreationDate": 1627681712.237,
            "Id": "us-west-2_EXAMPLE3",
            "LambdaConfig": {
                "UserMigration": "arn:aws:lambda:us-
east-1:123456789012:function:MyFunction"
            },
            "LastModifiedDate": 1678486942.479,
            "Name": "user pool 3"
        }
    ]
}
```

자세한 내용은 Amazon Cognito 개발자 안내서의 Amazon Cognito 사용자 풀을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조에서 ListUserPools 섹션을 참조하세요.

Go

SDK for Go V2

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

```
package main

import (
 "context"
 "fmt"
 "log"

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

```
// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 cognitoClient := cognitoidentityprovider.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the user pools for your account.")
 var pools []types.UserPoolDescriptionType
 paginator := cognitoidentityprovider.NewListUserPoolsPaginator(
  cognitoClient, &cognitoidentityprovider.ListUserPoolsInput{MaxResults:
 aws.Int32(10)})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get user pools. Here's why: %v\n", err)
  } else {
   pools = append(pools, output.UserPools...)
  }
 }
 if len(pools) == 0 {
  fmt.Println("You don't have any user pools!")
 } else {
  for _, pool := range pools {
   fmt.Printf("\t%v: %v\n", *pool.Name, *pool.Id)
  }
 }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 ListUserPools를 참조하세요.

Java

SDK for Java 2.x

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

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

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

    public static void listAllUserPools(CognitoIdentityProviderClient
 cognitoClient) {
```

```
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                    .maxResults(10)
                    .build();

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

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

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

Rust

SDK for Rust

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

```
async fn show_pools(client: &Client) -> Result<(), Error> {
    let response = client.list_user_pools().max_results(10).send().await?;
    let pools = response.user_pools();
    println!("User pools:");
    for pool in pools {
        println!("  ID:              {}", pool.id().unwrap_or_default());
        println!("  Name:            {}", pool.name().unwrap_or_default());
        println!("  Lambda Config:   {:?}", pool.lambda_config().unwrap());
        println!(
            "  Last modified:   {}",
```

```
            pool.last_modified_date().unwrap().to_chrono_utc()?
        );
        println!(
            "  Creation date:   {:?}",
            pool.creation_date().unwrap().to_chrono_utc()
        );
        println!();
    }
    println!("Next token: {}", response.next_token().unwrap_or_default());

    Ok(())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 ListUserPools를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **ListUsers** 함께 사용

다음 코드 예제는 ListUsers의 사용 방법을 보여 줍니다.

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

- 사용자 풀 시작하기
- MFA가 필요한 사용자 풀에 사용자 가입시키기

.NET

SDK for .NET

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

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

```
    /// Get a list of users for the Amazon Cognito user pool.
    /// </summary>
    /// <param name="userPoolId">The user pool ID.</param>
    /// <returns>A list of users.</returns>
    public async Task<List<UserType>> ListUsersAsync(string userPoolId)
    {
        var request = new ListUsersRequest
        {
            UserPoolId = userPoolId
        };

        var users = new List<UserType>();

        var usersPaginator = _cognitoService.Paginators.ListUsers(request);
        await foreach (var response in usersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 ListUsers를 참조하세요.

CLI

AWS CLI

예제 1: 서버 측 필터를 사용하여 사용자 나열

다음 `list-users` 예제에서는 이메일 주소가 `testuser`로 시작하는 요청된 사용자 풀의 사용자 3명을 나열합니다.

```
aws cognito-idp list-users \
    --user-pool-id us-west-2_EXAMPLE \
    --filter email^=\"testuser\" \
    --max-items 3
```

출력:

```
{
    "PaginationToken": "efgh5678EXAMPLE",
    "Users": [
        {
            "Attributes": [
                {
                    "Name": "sub",
                    "Value": "eaad0219-2117-439f-8d46-4db20e59268f"
                },
                {
                    "Name": "email",
                    "Value": "testuser@example.com"
                }
            ],
            "Enabled": true,
            "UserCreateDate": 1682955829.578,
            "UserLastModifiedDate": 1689030181.63,
            "UserStatus": "CONFIRMED",
            "Username": "testuser"
        },
        {
            "Attributes": [
                {
                    "Name": "sub",
                    "Value": "3b994cfd-0b07-4581-be46-3c82f9a70c90"
                },
                {
                    "Name": "email",
                    "Value": "testuser2@example.com"
                }
            ],
            "Enabled": true,
            "UserCreateDate": 1684427979.201,
            "UserLastModifiedDate": 1684427979.201,
            "UserStatus": "UNCONFIRMED",
            "Username": "testuser2"
        },
        {
            "Attributes": [
                {
                    "Name": "sub",
                    "Value": "5929e0d1-4c34-42d1-9b79-a5ecacfe66f7"
                },
```

```
                {
                    "Name": "email",
                    "Value": "testuser3@example.com"
                }
            ],
            "Enabled": true,
            "UserCreateDate": 1684427823.641,
            "UserLastModifiedDate": 1684427823.641,
            "UserStatus": "UNCONFIRMED",
            "Username": "testuser3@example.com"
        }
    ]
}
```

자세한 내용은 Amazon Cognito 개발자 안내서의 Managing and searching for users를 참조하시기 바랍니다.

예제 2: 클라이언트 측 필터를 사용하여 사용자 나열

다음 `list-users` 예제에서는 이메일 도메인 '@example.com'이 포함된 이메일 주소와 같은 속성을 가진 사용자 3명의 속성을 나열합니다. 다른 속성에 이 문자열이 포함된 경우 해당 속성도 표시됩니다. 두 번째 사용자는 쿼리와 일치하는 속성이 없으며 표시된 출력에서 제외되지만 서버 응답에서는 제외되지 않습니다.

```
aws cognito-idp list-users \
    --user-pool-id us-west-2_EXAMPLE \
    --max-items 3
    --query Users\[\*\].Attributes\[\?Value\.contains\(\@\,\'@example.com\'\)\]
```

출력:

```
[
    [
        {
            "Name": "email",
            "Value": "admin@example.com"
        }
    ],
    [],
    [
        {
            "Name": "email",
```

```
            "Value": "operator@example.com"
        }
    ]
]
```

자세한 내용은 Amazon Cognito 개발자 안내서의 Managing and searching for users를 참조하시기 바랍니다.

- API 세부 정보는 AWS CLI 명령 참조의 ListUsers를 참조하세요.

Java

SDK for Java 2.x

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

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListUsers {
    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:
                    <userPoolId>\s

                Where:
                    userPoolId - The ID given to your user pool when it's
created.
                """;

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

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

        listAllUsers(cognitoClient, userPoolId);
        listUsersFilter(cognitoClient, userPoolId);
        cognitoClient.close();
    }

    public static void listAllUsers(CognitoIdentityProviderClient cognitoClient,
String userPoolId) {
        try {
            ListUsersRequest usersRequest = ListUsersRequest.builder()
                    .userPoolId(userPoolId)
                    .build();

            ListUsersResponse response = cognitoClient.listUsers(usersRequest);
            response.users().forEach(user -> {
                System.out.println("User " + user.username() + " Status " +
user.userStatus() + " Created "
                        + user.userCreateDate());
            });

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

```
    }

    // Shows how to list users by using a filter.
    public static void listUsersFilter(CognitoIdentityProviderClient
 cognitoClient, String userPoolId) {

        try {
            String filter = "email = \"tblue@noserver.com\"";
            ListUsersRequest usersRequest = ListUsersRequest.builder()
                    .userPoolId(userPoolId)
                    .filter(filter)
                    .build();

            ListUsersResponse response = cognitoClient.listUsers(usersRequest);
            response.users().forEach(user -> {
                System.out.println("User with filter applied " + user.username()
 + " Status " + user.userStatus()
                        + " Created " + user.userCreateDate());
            });

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

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

JavaScript

SDK for JavaScript (v3)

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

```
const listUsers = ({ userPoolId }) => {
  const client = new CognitoIdentityProviderClient({});
```

```
  const command = new ListUsersCommand({
    UserPoolId: userPoolId,
  });

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

- API 세부 정보는 AWS SDK for JavaScript API 참조의 ListUsers를 참조하십시오.

Kotlin

SDK for Kotlin

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

```
suspend fun listAllUsers(userPoolId: String) {
    val request =
        ListUsersRequest {
            this.userPoolId = userPoolId
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { cognitoClient ->
        val response = cognitoClient.listUsers(request)
        response.users?.forEach { user ->
            println("The user name is ${user.username}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListUsers를 참조하십시오.

Python

SDK for Python (Boto3)

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

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

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


    def list_users(self):
        """
        Returns a list of the users in the current user pool.

        :return: The list of users.
        """
        try:
            response =
 self.cognito_idp_client.list_users(UserPoolId=self.user_pool_id)
            users = response["Users"]
        except ClientError as err:
            logger.error(
                "Couldn't list users for %s. Here's why: %s: %s",
```

```
                self.user_pool_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return users
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 ListUsers를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

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

```
TRY.
    DATA(lo_result) = lo_cgp->listusers(
      iv_userpoolid = iv_user_pool_id
    ).

    ot_users = lo_result->get_users( ).

    MESSAGE |Found { lines( ot_users ) } users in the pool.| TYPE 'I'.

  CATCH /aws1/cx_cgpresourcenotfoundex INTO DATA(lo_ex).
    MESSAGE |User pool { iv_user_pool_id } not found.| TYPE 'E'.

  CATCH /aws1/cx_cgpnotauthorizedex INTO DATA(lo_auth_ex).
    MESSAGE 'Not authorized to list users.' TYPE 'E'.
ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API의 ListUsers을 참조하세요.

Swift

SDK for Swift

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

```
        do {
            let output = try await cognitoClient.listUsers(
                input: ListUsersInput(
                    userPoolId: poolId
                )
            )

            guard let users = output.users else {
                print("No users found.")
                return
            }

            print("\(users.count) user(s) found.")
            for user in users {
                print("  \(user.username ?? "<unknown>")")
            }
        } catch _ as NotAuthorizedException {
            print("*** Please authenticate with AWS before using this command.")
            return
        } catch _ as ResourceNotFoundException {
            print("*** The specified User Pool was not found.")
            return
        } catch {
            print("*** An unexpected type of error occurred.")
            return
        }
```

- API 세부 정보는 AWS SDK for Swift API 참조의 ListUsers를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **ResendConfirmationCode** 함께 사용

다음 코드 예제는 ResendConfirmationCode의 사용 방법을 보여 줍니다.

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

- MFA가 필요한 사용자 풀에 사용자 가입시키기

.NET

SDK for .NET

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

```
    /// <summary>
    /// Send a new confirmation code to a user.
    /// </summary>
    /// <param name="clientId">The Id of the client application.</param>
    /// <param name="userName">The username of user who will receive the code.</
param>
    /// <returns>The delivery details.</returns>
    public async Task<CodeDeliveryDetailsType> ResendConfirmationCodeAsync(string
 clientId, string userName)
    {
        var codeRequest = new ResendConfirmationCodeRequest
        {
            ClientId = clientId,
            Username = userName,
        };

        var response = await
 _cognitoService.ResendConfirmationCodeAsync(codeRequest);
```

```
        Console.WriteLine($"Method of delivery is
{response.CodeDeliveryDetails.DeliveryMedium}");

        return response.CodeDeliveryDetails;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 ResendConfirmationCode를 참조하십시오.

C++

SDK for C++

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

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

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

        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeRequest
request;
        request.SetUsername(userName);
        request.SetClientId(clientID);

        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeOutcome
outcome =
                client.ResendConfirmationCode(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "CognitoIdentityProvider::ResendConfirmationCode was
successful."
                    << std::endl;
        }
```

```
        else {
            std::cerr << "Error with
 CognitoIdentityProvider::ResendConfirmationCode. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 ResendConfirmationCode를 참조하세요.

CLI

AWS CLI

확인 코드 다시 보내기

다음 `resend-confirmation-code` 예시에서는 사용자 `jane`에게 확인 코드를 보냅니다.

```
aws cognito-idp resend-confirmation-code \
    --client-id 12a3b456c7de890f11g123hijk \
    --username jane
```

출력:

```
{
    "CodeDeliveryDetails": {
        "Destination": "j***@e***.com",
        "DeliveryMedium": "EMAIL",
        "AttributeName": "email"
    }
}
```

자세한 내용은 Amazon Cognito 개발자 안내서의 사용자 계정 가입 및 확인 섹션을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조에서 ResendConfirmationCode를 참조하세요.

Java

SDK for Java 2.x

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

```
   public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
           String userName) {
       try {
           ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
                   .clientId(clientId)
                   .username(userName)
                   .build();

           ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
           System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

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

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

JavaScript

SDK for JavaScript (v3)

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

```
const resendConfirmationCode = ({ clientId, username }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ResendConfirmationCodeCommand({
    ClientId: clientId,
    Username: username,
  });

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

- API 세부 정보는 AWS SDK for JavaScript API 참조의 ResendConfirmationCode를 참조하십시오.

Kotlin

SDK for Kotlin

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

```
suspend fun resendConfirmationCode(
    clientIdVal: String?,
    userNameVal: String?,
) {
```

```
    val codeRequest =
        ResendConfirmationCodeRequest {
            clientId = clientIdVal
            username = userNameVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ResendConfirmationCode를 참조하십시오.

Python

SDK for Python (Boto3)

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

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

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

```
        self.client_id = client_id
        self.client_secret = client_secret


    def resend_confirmation(self, user_name):
        """
        Prompts Amazon Cognito to resend an email with a new confirmation code.

        :param user_name: The name of the user who will receive the email.
        :return: Delivery information about where the email is sent.
        """
        try:
            kwargs = {"ClientId": self.client_id, "Username": user_name}
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.resend_confirmation_code(**kwargs)
            delivery = response["CodeDeliveryDetails"]
        except ClientError as err:
            logger.error(
                "Couldn't resend confirmation to %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return delivery
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 ResendConfirmationCode를 참조하세요.

Swift

SDK for Swift

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

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Requests a new confirmation code be sent to the given user's contact
    /// method.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The application client ID.
    ///   - userName: The user to resend a code for.
    ///
    /// - Returns: `true` if a new code was sent successfully, otherwise
    ///   `false`.
    func resendConfirmationCode(cipClient: CognitoIdentityProviderClient,
 clientId: String,
                                userName: String) async -> Bool {
        do {
            let output = try await cipClient.resendConfirmationCode(
                input: ResendConfirmationCodeInput(
                    clientId: clientId,
                    username: userName
                )
            )

            guard let deliveryMedium = output.codeDeliveryDetails?.deliveryMedium
 else {
                print("*** Unable to get the delivery method for the resent
 code.")
                return false
            }

            print("=====> A new code has been sent by \(deliveryMedium)")
            return true
        } catch {
            print("*** Unable to resend the confirmation code to user
 \(userName).")
            return false
        }
    }
```

- API 세부 정보는 AWS SDK for Swift API 참조의 ResendConfirmationCode를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **RespondToAuthChallenge** 함께 사용

다음 코드 예제는 RespondToAuthChallenge의 사용 방법을 보여 줍니다.

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

- MFA가 필요한 사용자 풀에 사용자 가입시키기

CLI

AWS CLI

예제 1: NEW_PASSWORD_REQUIRED 과제에 응답

다음 respond-to-auth-challenge 예제에서는 초기 인증이 반환된 NEW_PASSWORD_REQUIRED 챌린지에 응답합니다. 사용자(jane@example.com)용 새 암호를 설정합니다.

```
aws cognito-idp respond-to-auth-challenge \
    --client-id 1example23456789 \
    --challenge-name NEW_PASSWORD_REQUIRED \
    --challenge-responses USERNAME=jane@example.com,NEW_PASSWORD=[Password] \
    --session AYABeEv5HklEXAMPLE
```

출력:

```
{
    "ChallengeParameters": {},
    "AuthenticationResult": {
        "AccessToken": "ACCESS_TOKEN",
        "ExpiresIn": 3600,
        "TokenType": "Bearer",
        "RefreshToken": "REFRESH_TOKEN",
        "IdToken": "ID_TOKEN",
        "NewDeviceMetadata": {
            "DeviceKey": "us-west-2_a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "DeviceGroupKey": "-wt2ha1Zd"
        }
```

```
    }
}
```

자세한 내용은 Amazon Cognito 개발자 안내서의 Authentication을 참조하시기 바랍니다.

예제 2: SELECT_MFA_TYPE 챌린지에 응답

다음 `respond-to-auth-challenge` 예제에서는 TOTP MFA를 현재 사용자의 MFA 옵션으로 선택합니다. 사용자에게 MFA 유형을 선택하라는 메시지가 표시되고 다음으로 MFA 코드를 입력하라는 메시지가 표시됩니다.

```
aws cognito-idp respond-to-auth-challenge \
    --client-id 1example23456789
    --session AYABeEv5HklEXAMPLE
    --challenge-name SELECT_MFA_TYPE
    --challenge-responses USERNAME=testuser,ANSWER=SOFTWARE_TOKEN_MFA
```

출력:

```
{
    "ChallengeName": "SOFTWARE_TOKEN_MFA",
    "Session": "AYABeEv5HklEXAMPLE",
    "ChallengeParameters": {
        "FRIENDLY_DEVICE_NAME": "transparent"
    }
}
```

자세한 내용은 Amazon Cognito 개발자 안내서의 Adding MFA를 참조하시기 바랍니다.

예제 3: SOFTWARE_TOKEN_MFA 챌린지에 응답

다음 `respond-to-auth-challenge` 예제에서는 TOTP MFA 코드를 제공하고 로그인을 완료합니다.

```
aws cognito-idp respond-to-auth-challenge \
    --client-id 1example23456789 \
    --session AYABeEv5HklEXAMPLE \
    --challenge-name SOFTWARE_TOKEN_MFA \
    --challenge-responses USERNAME=testuser,SOFTWARE_TOKEN_MFA_CODE=123456
```

출력:

```
{
    "AuthenticationResult": {
        "AccessToken": "eyJra456defEXAMPLE",
        "ExpiresIn": 3600,
        "TokenType": "Bearer",
        "RefreshToken": "eyJra123abcEXAMPLE",
        "IdToken": "eyJra789ghiEXAMPLE",
        "NewDeviceMetadata": {
            "DeviceKey": "us-west-2_a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "DeviceGroupKey": "-v7w9UcY6"
        }
    }
}
```

자세한 내용은 Amazon Cognito 개발자 안내서의 Adding MFA를 참조하시기 바랍니다.

- API 세부 정보는 AWS CLI 명령 참조에서 RespondToAuthChallenge를 참조하세요.

JavaScript

SDK for JavaScript (v3)

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

```
const respondToAuthChallenge = ({
  clientId,
  username,
  session,
  userPoolId,
  code,
}) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new RespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: code,
```

```
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

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

- API 세부 정보는 AWS SDK for JavaScript API 참조의 RespondToAuthChallenge를 참조하십시오.

Python

SDK for Python (Boto3)

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

추적되는 디바이스로 로그인하세요. 로그인을 완료하려면 클라이언트가 보안 원격 암호(SRP) 문제에 올바르게 응답해야 합니다.

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

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

```
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_in_with_tracked_device(
        self,
        user_name,
        password,
        device_key,
        device_group_key,
        device_password,
        aws_srp,
    ):
        """
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA
 code.

        Signing in with a tracked device requires that the client respond to the
 SRP
        protocol. The scenario associated with this example uses the warrant
 package
        to help with SRP calculations.

        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains
 an
                 access token for the user.
        """
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name,
                password=device_password,
                pool_id="_",
```

```
                client_id=self.client_id,
                client_secret=None,
                client=self.cognito_idp_client,
            )

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id,
                AuthFlow="USER_PASSWORD_AUTH",
                AuthParameters={
                    "USERNAME": user_name,
                    "PASSWORD": password,
                    "DEVICE_KEY": device_key,
                },
            )
            if response_init["ChallengeName"] != "DEVICE_SRP_AUTH":
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
{response_init['ChallengeName']}."
                )

            auth_params = srp_helper.get_auth_params()
            auth_params["DEVICE_KEY"] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_SRP_AUTH",
                ChallengeResponses=auth_params,
            )
            if response_auth["ChallengeName"] != "DEVICE_PASSWORD_VERIFIER":
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}."
                )

            challenge_params = response_auth["ChallengeParameters"]
            challenge_params["USER_ID_FOR_SRP"] = device_group_key + device_key
            cr = srp_helper.process_challenge(challenge_params, {"USERNAME":
user_name})
            cr["USERNAME"] = user_name
            cr["DEVICE_KEY"] = device_key
            response_verifier =
self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_PASSWORD_VERIFIER",
                ChallengeResponses=cr,
```

```
            )
            auth_tokens = response_verifier["AuthenticationResult"]
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return auth_tokens
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 RespondToAuthChallenge를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **SignUp** 함께 사용

다음 코드 예제는 SignUp의 사용 방법을 보여 줍니다.

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

- Lambda 함수를 사용하여 알려진 사용자를 자동으로 확인
- Lambda 함수를 사용하여 알려진 사용자를 자동으로 마이그레이션
- 사용자 풀 시작하기
- MFA가 필요한 사용자 풀에 사용자 가입시키기

.NET

SDK for .NET

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

```
    /// <summary>
    /// Sign up a new user.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The username to use.</param>
    /// <param name="password">The user's password.</param>
    /// <param name="email">The email address of the user.</param>
    /// <returns>A Boolean value indicating whether the user was confirmed.</
returns>
    public async Task<bool> SignUpAsync(string clientId, string userName, string
 password, string email)
    {
        var userAttrs = new AttributeType
        {
            Name = "email",
            Value = email,
        };

        var userAttrsList = new List<AttributeType>();

        userAttrsList.Add(userAttrs);

        var signUpRequest = new SignUpRequest
        {
            UserAttributes = userAttrsList,
            Username = userName,
            ClientId = clientId,
            Password = password
        };

        var response = await _cognitoService.SignUpAsync(signUpRequest);
        return response.HttpStatusCode == HttpStatusCode.OK;
```

```
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 SignUp을 참조하십시오.

C++

SDK for C++

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

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

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

       Aws::CognitoIdentityProvider::Model::SignUpRequest request;
       request.AddUserAttributes(
               Aws::CognitoIdentityProvider::Model::AttributeType().WithName(
                       "email").WithValue(email));
       request.SetUsername(userName);
       request.SetPassword(password);
       request.SetClientId(clientID);
       Aws::CognitoIdentityProvider::Model::SignUpOutcome outcome =
               client.SignUp(request);

       if (outcome.IsSuccess()) {
           std::cout << "The signup request for " << userName << " was
successful."
                     << std::endl;
       }
       else if (outcome.GetError().GetErrorType() ==

Aws::CognitoIdentityProvider::CognitoIdentityProviderErrors::USERNAME_EXISTS) {
           std::cout
```

```
                    << "The username already exists. Please enter a different
 username."
                    << std::endl;
            userExists = true;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::SignUpRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 SignUp를 참조하세요.

CLI

AWS CLI

사용자 가입

이 예시에서는 jane@example.com에 가입합니다.

명령:

```
aws cognito-idp sign-up --client-id 3n4b5urk1ft4fl3mg5e62d9ado --
username jane@example.com --password PASSWORD --user-attributes
 Name="email",Value="jane@example.com" Name="name",Value="Jane"
```

출력:

```
{
  "UserConfirmed": false,
  "UserSub": "e04d60a6-45dc-441c-a40b-e25a787d4862"
}
```

- API 세부 정보는 AWS CLI 명령 참조에서 SignUp 섹션을 참조하세요.

Go

SDK for Go V2

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

```
import (
 "context"
 "errors"
 "log"

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

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
```

```
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 SignUp을 참조하십시오.

Java

SDK for Java 2.x

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

```
   public static void signUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String userName,
           String password, String email) {
       AttributeType userAttrs = AttributeType.builder()
               .name("email")
               .value(email)
               .build();

       List<AttributeType> userAttrsList = new ArrayList<>();
       userAttrsList.add(userAttrs);
       try {
           SignUpRequest signUpRequest = SignUpRequest.builder()
                   .userAttributes(userAttrsList)
                   .username(userName)
                   .clientId(clientId)
                   .password(password)
                   .build();
```

```
            identityProviderClient.signUp(signUpRequest);
            System.out.println("User has been signed up ");

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

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

JavaScript

SDK for JavaScript (v3)

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

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

  const command = new SignUpCommand({
    ClientId: clientId,
    Username: username,
    Password: password,
    UserAttributes: [{ Name: "email", Value: email }],
  });

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

- API 세부 정보는 AWS SDK for JavaScript API 참조의 SignUp을 참조하십시오.

Kotlin

SDK for Kotlin

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

```
suspend fun signUp(
    clientIdVal: String?,
    userNameVal: String?,
    passwordVal: String?,
    emailVal: String?,
) {
    val userAttrs =
        AttributeType {
            name = "email"
            value = emailVal
        }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)
    val signUpRequest =
        SignUpRequest {
            userAttributes = userAttrsList
            username = userNameVal
            clientId = clientIdVal
            password = passwordVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 SignUp을 참조하십시오.

Python

SDK for Python (Boto3)

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

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

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


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

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

        :param user_name: The user name that identifies the new user.
```

```
        :param password: The password for the new user.
        :param user_email: The email address for the new user.
        :return: True when the user is already confirmed with Amazon Cognito.
                 Otherwise, false.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "Password": password,
                "UserAttributes": [{"Name": "email", "Value": user_email}],
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.sign_up(**kwargs)
            confirmed = response["UserConfirmed"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "UsernameExistsException":
                response = self.cognito_idp_client.admin_get_user(
                    UserPoolId=self.user_pool_id, Username=user_name
                )
                logger.warning(
                    "User %s exists and is %s.", user_name,
response["UserStatus"]
                )
                confirmed = response["UserStatus"] == "CONFIRMED"
            else:
                logger.error(
                    "Couldn't sign up %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return confirmed
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 SignUp을 참조하세요.

Swift

SDK for Swift

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

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Create a new user in a user pool.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The ID of the app client to create a user for.
    ///   - userName: The username for the new user.
    ///   - password: The new user's password.
    ///   - email: The new user's email address.
    ///
    /// - Returns: `true` if successful; otherwise `false`.
    func signUp(cipClient: CognitoIdentityProviderClient, clientId: String,
 userName: String, password: String, email: String) async -> Bool {
        let emailAttr = CognitoIdentityProviderClientTypes.AttributeType(
            name: "email",
            value: email
        )

        let userAttrsList = [emailAttr]

        do {
            _ = try await cipClient.signUp(
                input: SignUpInput(
                    clientId: clientId,
                    password: password,
                    userAttributes: userAttrsList,
                    username: userName
                )

            )
```

```
            print("=====> User \(userName) signed up.")
        } catch _ as AWSCognitoIdentityProvider.UsernameExistsException {
            print("*** The username \(userName) already exists. Please use a
different one.")
            return false
        } catch let error as AWSCognitoIdentityProvider.InvalidPasswordException
{
            print("*** Error: The specified password is invalid. Reason:
\(error.properties.message ?? "<none available>").")
            return false
        } catch _ as AWSCognitoIdentityProvider.ResourceNotFoundException {
            print("*** Error: The specified client ID (\(clientId)) doesn't
exist.")
            return false
        } catch {
            print("*** Unexpected error: \(error)")
            return false
        }

        return true
    }
```

- API 세부 정보는 AWS SDK for Swift API 참조의 SignUp을 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **UpdateUserPool** 함께 사용

다음 코드 예제는 UpdateUserPool의 사용 방법을 보여 줍니다.

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

- Lambda 함수를 사용하여 알려진 사용자를 자동으로 확인
- Lambda 함수를 사용하여 알려진 사용자를 자동으로 마이그레이션
- Amazon Cognito 사용자 인증 후 Lambda 함수를 사용하여 사용자 지정 활동 데이터 작성

CLI

AWS CLI

사용자 풀을 생성하는 방법

다음 update-user-pool 예시에서는 사용 가능한 각 구성 옵션에 대한 예시 구문으로 사용자 풀을 수정합니다. 사용자 풀을 업데이트하려면 이전에 구성된 모든 옵션을 지정해야 합니다. 그렇지 않으면 옵션이 기본값으로 재설정됩니다.

```
aws cognito-idp update-user-pool --user-pool-id us-west-2_EXAMPLE \
    --policies PasswordPolicy=
\{MinimumLength=6,RequireUppercase=true,RequireLowercase=true,RequireNumbers=true,Require
 \
    --deletion-protection ACTIVE \
    --lambda-config PreSignUp="arn:aws:lambda:us-
west-2:123456789012:function:cognito-test-presignup-
function",PreTokenGeneration="arn:aws:lambda:us-
west-2:123456789012:function:cognito-test-pretoken-function" \
    --auto-verified-attributes "phone_number" "email" \
    --verification-message-template \{\"SmsMessage\":\""Your code is
 {####}"\",\"EmailMessage\":\""Your code is {####}"\",\"EmailSubject\":\""Your
 verification code"\",\"EmailMessageByLink\":\""Click {##here##} to verify
 your email address."\",\"EmailSubjectByLink\":\""Your verification link"\",
\"DefaultEmailOption\":\"CONFIRM_WITH_LINK\"\} \
    --sms-authentication-message "Your code is {####}" \
    --user-attribute-update-settings
 AttributesRequireVerificationBeforeUpdate="email","phone_number" \
    --mfa-configuration "OPTIONAL" \
    --device-
configuration ChallengeRequiredOnNewDevice=true,DeviceOnlyRememberedOnUserPrompt=true
 \
    --email-configuration SourceArn="arn:aws:ses:us-
west-2:123456789012:identity/admin@example.com",ReplyToEmailAddress="amdin
+noreply@example.com",EmailSendingAccount=DEVELOPER,From="admin@amazon.com",Configuration
configuration-set" \
    --sms-configuration SnsCallerArn="arn:aws:iam::123456789012:role/service-
role/SNS-SMS-Role",ExternalId="12345",SnsRegion="us-west-2" \
    --admin-create-user-config
 AllowAdminCreateUserOnly=false,InviteMessageTemplate=\{SMSMessage=\""Welcome
 {username}. Your confirmation code is {####}"\",EmailMessage=\""Welcome
 {username}. Your confirmation code is {####}"\",EmailSubject=\""Welcome to
 MyMobileGame"\"\} \
```

```
    --user-pool-tags "Function"="MyMobileGame","Developers"="Berlin" \
    --admin-create-user-config
 AllowAdminCreateUserOnly=false,InviteMessageTemplate=\{SMSMessage=\""Welcome
 {username}. Your confirmation code is {####}"\",EmailMessage=\""Welcome
 {username}. Your confirmation code is {####}"\",EmailSubject=\""Welcome to
 MyMobileGame"\"\} \
    --user-pool-add-ons AdvancedSecurityMode="AUDIT" \
    --account-recovery-setting RecoveryMechanisms=
\[\{Priority=1,Name="verified_email"\},
\{Priority=2,Name="verified_phone_number"\}\]
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon Cognito 개발자 안내서의 Updating user pool configuration을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 UpdateUserPool을 참조하세요.

Go

SDK for Go V2

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

```
import (
 "context"
 "errors"
 "log"

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

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

```
// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId
 string, triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
```

```
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 UpdateUserPool을 참조하세요.

JavaScript

SDK for JavaScript (v3)

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

```
/**
 * Connect a Lambda function to the PreSignUp trigger for a Cognito user pool
 * @param {{ region: string, userPoolId: string, handlerArn: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").UpdateUserPoolCommandOutput | null, unknown]>}
 */
export const addPreSignUpHandler = async ({
  region,
  userPoolId,
  handlerArn,
}) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({
      region,
    });

    const command = new UpdateUserPoolCommand({
```

```
      UserPoolId: userPoolId,
      LambdaConfig: {
        PreSignUp: handlerArn,
      },
    });

    const response = await cognitoClient.send(command);
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 UpdateUserPool을 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **`VerifySoftwareToken`** 함께 사용

다음 코드 예제는 `VerifySoftwareToken`의 사용 방법을 보여 줍니다.

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

- MFA가 필요한 사용자 풀에 사용자 가입시키기

.NET

SDK for .NET

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

```
    /// <summary>
    /// Verify the TOTP and register for MFA.
```

```
    /// </summary>
    /// <param name="session">The name of the session.</param>
    /// <param name="code">The MFA code.</param>
    /// <returns>The status of the software token.</returns>
    public async Task<VerifySoftwareTokenResponseType>
VerifySoftwareTokenAsync(string session, string code)
    {
        var tokenRequest = new VerifySoftwareTokenRequest
        {
            UserCode = code,
            Session = session,
        };

        var verifyResponse = await
_cognitoService.VerifySoftwareTokenAsync(tokenRequest);

        return verifyResponse.Status;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 VerifySoftwareToken을 참조하십시오.

C++

SDK for C++

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

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

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

        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenRequest request;
        request.SetUserCode(userCode);
```

```
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenOutcome outcome =
                client.VerifySoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout << "Verification of the code was successful."
                      << std::endl;
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::VerifySoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 VerifySoftwareToken을 참조하세요.

CLI

AWS CLI

TOTP 인증자의 등록 확인

다음 `verify-software-token` 예제에서는 현재 사용자의 TOTP 등록을 완료합니다.

```
aws cognito-idp verify-software-token \
    --access-token eyJra456defEXAMPLE \
    --user-code 123456
```

출력:

```
{
    "Status": "SUCCESS"
}
```

자세한 내용은 Amazon Cognito 개발자 안내서의 Adding MFA to a user pool을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 VerifySoftwareToken을 참조하시기 바랍니다.

Java

SDK for Java 2.x

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

```
   // Verify the TOTP and register for MFA.
   public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
       try {
           VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
                   .userCode(code)
                   .session(session)
                   .build();

           VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
           System.out.println("The status of the token is " +
verifyResponse.statusAsString());

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

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

JavaScript

SDK for JavaScript (v3)

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

```
const verifySoftwareToken = (totp) => {
  const client = new CognitoIdentityProviderClient({});

  // The 'Session' is provided in the response to 'AssociateSoftwareToken'.
  const session = process.env.SESSION;

  if (!session) {
    throw new Error(
      "Missing a valid Session. Did you run 'admin-initiate-auth'?",
    );
  }

  const command = new VerifySoftwareTokenCommand({
    Session: session,
    UserCode: totp,
  });

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

- API 세부 정보는 AWS SDK for JavaScript API 참조의 VerifySoftwareToken을 참조하십시오.

Kotlin

SDK for Kotlin

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

```
// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(
    sessionVal: String?,
    codeVal: String?,
) {
    val tokenRequest =
        VerifySoftwareTokenRequest {
            userCode = codeVal
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 VerifySoftwareToken을 참조하십시오.

Python

SDK for Python (Boto3)

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

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

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


    def verify_mfa(self, session, user_code):
        """
        Verify a new MFA application that is associated with a user.

        :param session: Session information returned from a previous call to
 initiate
                        authentication.
        :param user_code: A code generated by the associated MFA application.
        :return: Status that indicates whether the MFA application is verified.
        """
        try:
            response = self.cognito_idp_client.verify_software_token(
                Session=session, UserCode=user_code
            )
        except ClientError as err:
            logger.error(
                "Couldn't verify MFA. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 VerifySoftwareToken을 참조하세요.

SAP ABAP

SDK for SAP ABAP API

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

```
TRY.
    DATA(lo_result) = lo_cgp->verifysoftwaretoken(
      iv_session = iv_session
      iv_usercode = iv_user_code
    ).

    ov_status = lo_result->get_status( ).

    IF ov_status = 'SUCCESS'.
      MESSAGE 'MFA token verified successfully.' TYPE 'I'.
    ELSE.
      MESSAGE |MFA verification status: { ov_status }.| TYPE 'I'.
    ENDIF.

  CATCH /aws1/cx_cgpcodemismatchex INTO DATA(lo_code_ex).
    MESSAGE 'Invalid MFA code provided.' TYPE 'E'.

  CATCH /aws1/cx_cgpenbsoftwaretokmf00 INTO DATA(lo_enabled_ex).
    MESSAGE 'Software token MFA is already enabled.' TYPE 'E'.
ENDTRY.
```

- API 세부 정보는 SDK for SAP ABAP API 참조의 VerifySoftwareToken을 참조하세요. AWS

Swift

SDK for Swift

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

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Confirm that the user's TOTP authenticator is configured correctly by
    /// sending a code to it to check that it matches successfully.
    ///
    /// - Parameters:
    ///   - cipClient: The `CongnitoIdentityProviderClient` to use.
    ///   - session: An authentication session previously returned by an
    ///     `associateSoftwareToken()` call.
    ///   - mfaCode: The 6-digit code currently displayed by the user's
    ///     authenticator, as provided by the user.
    func verifyTOTP(cipClient: CognitoIdentityProviderClient, session: String?,
 mfaCode: String?) async {
        do {
            let output = try await cipClient.verifySoftwareToken(
                input: VerifySoftwareTokenInput(
                    session: session,
                    userCode: mfaCode
                )
            )

            guard let tokenStatus = output.status else {
                print("*** Unable to get the token's status.")
                return
            }
            print("=====> The token's status is: \(tokenStatus)")
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return
        } catch _ as CodeMismatchException {
            print("*** The specified MFA code doesn't match the expected value.")
```

```
            return
        } catch _ as UserNotFoundException {
            print("*** The specified username doesn't exist.")
            return
        } catch _ as UserNotConfirmedException {
            print("*** The user has not been confirmed.")
            return
        } catch {
            print("*** Error verifying the MFA token!")
            return
        }
    }
```

- API 세부 정보는 AWS SDK for Swift API 참조의 VerifySoftwareToken을 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDKs를 사용하는 Amazon Cognito 자격 증명 공급자에 대한 시나리오

다음 코드 예제에서는 Amazon Cognito Identity Provider AWS SDKs에서 일반적인 시나리오를 구현하는 방법을 보여줍니다. 이러한 시나리오에서는 Amazon Cognito ID 제공업체 내에서 여러 함수를 호출하거나 다른 AWS 서비스와 결합하여 특정 작업을 수행하는 방법을 보여줍니다. 각 시나리오에는 전체 소스 코드에 대한 링크가 포함되어 있습니다. 여기에서 코드를 설정 및 실행하는 방법에 대한 지침을 찾을 수 있습니다.

시나리오는 컨텍스트에 맞는 서비스 작업을 이해하는 데 도움이 되도록 중급 수준의 경험을 대상으로 합니다.

예시

- AWS SDK를 사용하여 Lambda 함수로 알려진 Amazon Cognito 사용자를 자동으로 확인
- AWS SDK를 사용하여 Lambda 함수를 사용하여 알려진 Amazon Cognito 사용자를 자동으로 마이그레이션
- AWS SDK를 사용하여 M2M 클라이언트 토큰을 가져오고 Amazon Cognito에서 용어 검색
- 사용자 풀 시작하기
- AWS SDK를 사용하여 MFA가 필요한 Amazon Cognito 사용자 풀에 사용자 등록
- Amazon Cognito ID 풀 및 인증 흐름 사용

- AWS SDK를 사용하여 Amazon Cognito 사용자 인증 후 Lambda 함수로 사용자 지정 활동 데이터 쓰기

AWS SDK를 사용하여 Lambda 함수로 알려진 Amazon Cognito 사용자를 자동으로 확인

다음 코드 예제에서는 Lambda 함수를 사용하여 알려진 Amazon Cognito 사용자를 자동으로 확인하는 방법을 보여줍니다.

- `PreSignUp` 트리거에 대해 Lambda 함수를 호출하도록 사용자 풀을 구성합니다.
- Amazon Cognito를 사용하여 사용자 가입시키기
- Lambda 함수는 DynamoDB 테이블을 스캔하고 알려진 사용자를 자동으로 확인합니다.
- 새 사용자로 로그인한 다음 리소스를 정리합니다.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
import (
 "context"
 "errors"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)
```

```
// AutoConfirm separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type AutoConfirm struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewAutoConfirm constructs a new auto confirm runner.
func NewAutoConfirm(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) AutoConfirm {
 scenario := AutoConfirm{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddPreSignUpTrigger adds a Lambda handler as an invocation target for the
 PreSignUp trigger.
func (runner *AutoConfirm) AddPreSignUpTrigger(ctx context.Context, userPoolId
 string, functionArn string) {
 log.Printf("Let's add a Lambda function to handle the PreSignUp trigger from
 Cognito.\n" +
  "This trigger happens when a user signs up, and lets your function take action
 before the main Cognito\n" +
  "sign up processing occurs.\n")
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
  actions.TriggerInfo{Trigger: actions.PreSignUp, HandlerArn:
 aws.String(functionArn)})
 if err != nil {
  panic(err)
 }
 log.Printf("Lambda function %v added to user pool %v to handle the PreSignUp
 trigger.\n",
  functionArn, userPoolId)
```

```
}

// SignUpUser signs up a user from the known user table with a password you
 specify.
func (runner *AutoConfirm) SignUpUser(ctx context.Context, clientId string,
 usersTable string) (string, string) {
 log.Println("Let's sign up a user to your Cognito user pool. When the user's
 email matches an email in the\n" +
  "DynamoDB known users table, it is automatically verified and the user is
 confirmed.")

 knownUsers, err := runner.helper.GetKnownUsers(ctx, usersTable)
 if err != nil {
  panic(err)
 }
 userChoice := runner.questioner.AskChoice("Which user do you want to use?\n",
 knownUsers.UserNameList())
 user := knownUsers.Users[userChoice]

 var signedUp bool
 var userConfirmed bool
 password := runner.questioner.AskPassword("Enter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !signedUp {
  log.Printf("Signing up user '%v' with email '%v' to Cognito.\n", user.UserName,
 user.UserEmail)
  userConfirmed, err = runner.cognitoActor.SignUp(ctx, clientId, user.UserName,
 password, user.UserEmail)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("Enter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   signedUp = true
  }
 }
 log.Printf("User %v signed up, confirmed = %v.\n", user.UserName, userConfirmed)

 log.Println(strings.Repeat("-", 88))
```

```
 return user.UserName, password
}

// SignInUser signs in a user.
func (runner *AutoConfirm) SignInUser(ctx context.Context, clientId string,
 userName string, password string) string {
 runner.questioner.Ask("Press Enter when you're ready to continue.")
 log.Printf("Let's sign in as %v...\n", userName)
 authResult, err := runner.cognitoActor.SignIn(ctx, clientId, userName, password)
 if err != nil {
  panic(err)
 }
 log.Printf("Successfully signed in. Your access token starts with: %v...\n",
 (*authResult.AccessToken)[:10])
 log.Println(strings.Repeat("-", 88))
 return *authResult.AccessToken
}

// Run runs the scenario.
func (runner *AutoConfirm) Run(ctx context.Context, stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]
 runner.helper.PopulateUserTable(ctx, stackOutputs["TableName"])

 runner.AddPreSignUpTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["AutoConfirmFunctionArn"])
 runner.resources.triggers = append(runner.resources.triggers, actions.PreSignUp)
 userName, password := runner.SignUpUser(ctx, stackOutputs["UserPoolClientId"],
 stackOutputs["TableName"])
```

```
 runner.helper.ListRecentLogEvents(ctx, stackOutputs["AutoConfirmFunction"])
 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
  runner.SignInUser(ctx, stackOutputs["UserPoolClientId"], userName, password))

 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Lambda 함수를 사용하여 PreSignUp 트리거를 처리합니다.

```
import (
 "context"
 "log"
 "os"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 dynamodbtypes "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string `dynamodbav:"UserName"`
 UserEmail string `dynamodbav:"UserEmail"`
}

// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
 userEmail, err := attributevalue.Marshal(user.UserEmail)
 if err != nil {
```

```
  panic(err)
 }
 return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the PreSignUp event by looking up a user in an Amazon
 DynamoDB table and
// specifying whether they should be confirmed and verified.
func (h *handler) HandleRequest(ctx context.Context, event
 events.CognitoEventUserPoolsPreSignup) (events.CognitoEventUserPoolsPreSignup,
 error) {
 log.Printf("Received presignup from %v for user '%v'", event.TriggerSource,
 event.UserName)
 if event.TriggerSource != "PreSignUp_SignUp" {
  // Other trigger sources, such as PreSignUp_AdminInitiateAuth, ignore the
 response from this handler.
  return event, nil
 }
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserEmail: event.Request.UserAttributes["email"],
 }
 log.Printf("Looking up email %v in table %v.\n", user.UserEmail, tableName)
 output, err := h.dynamoClient.GetItem(ctx, &dynamodb.GetItemInput{
  Key:       user.GetKey(),
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Error looking up email %v.\n", user.UserEmail)
  return event, err
 }
 if output.Item == nil {
  log.Printf("Email %v not found. Email verification is required.\n",
 user.UserEmail)
  return event, err
 }

 err = attributevalue.UnmarshalMap(output.Item, &user)
 if err != nil {
  log.Printf("Couldn't unmarshal DynamoDB item. Here's why: %v\n", err)
```

```
  return event, err
 }

 if user.UserName != event.UserName {
  log.Printf("UserEmail %v found, but stored UserName '%v' does not match
 supplied UserName '%v'. Verification is required.\n",
   user.UserEmail, user.UserName, event.UserName)
 } else {
  log.Printf("UserEmail %v found with matching UserName %v. User is confirmed.
\n", user.UserEmail, user.UserName)
  event.Response.AutoConfirmUser = true
  event.Response.AutoVerifyEmail = true
 }

 return event, err
}

func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

일반적인 작업을 수행하는 구조체를 생성합니다.

```
import (
 "context"
 "log"
 "strings"
 "time"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
```

```
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
```

```
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName
 string) (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
```

```
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context,
 functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Amazon Cognito 작업을 래핑하는 구조체를 생성합니다.

```
import (
 "context"
 "errors"
 "log"

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

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



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId
 string, triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
```

```
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}
```

```
// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}
```

```
// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken
 string) error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
```

```
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId
 string, userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



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

```
 }
 return err
}
```

DynamoDB 작업을 래핑하는 구조체를 생성합니다.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
```

```
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
```

```
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user
 User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

CloudWatch Logs 작업을 래핑하는 구조체를 생성합니다.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
```

```
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx,
 &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
```

```
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

CloudFormation 작업을 래핑하는 구조체를 생성합니다.

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName
 string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
```

```
 }
 return stackOutputs
}
```

리소스를 정리합니다.

```
import (
 "context"
 "log"
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 \n" +
```

```
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(ctx, accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
 HandlerArn: nil}
  }
  err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
   panic(err)
  }
  log.Println("Removed Cognito triggers from user pool.")
 } else {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
 }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 다음 주제를 참조하세요.
  - DeleteUser
  - InitiateAuth
  - SignUp
  - UpdateUserPool

JavaScript

SDK for JavaScript (v3)

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

대화형 '시나리오' 실행을 구성합니다. JavaScript(v3) 예제에서는 시나리오 실행기를 공유하여 복잡한 예제를 간소화합니다. 전체 소스 코드는 GitHub에 있습니다.

```
import { AutoConfirm } from "./scenario-auto-confirm.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {
  errors: [],
  users: [
    {
      UserName: "test_user_1",
      UserEmail: "test_email_1@example.com",
    },
    {
      UserName: "test_user_2",
      UserEmail: "test_email_2@example.com",
    },
    {
      UserName: "test_user_3",
      UserEmail: "test_email_3@example.com",
    },
  ],
};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
```

```
export const scenarios = {
  // Demonstrate automatically confirming known users in a database.
  "auto-confirm": AutoConfirm(context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { parseScenarioArgs } from "@aws-doc-sdk-examples/lib/scenario/index.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Cognito user pools and triggers",
    description:
      "Demonstrate how to use the AWS SDKs to customize Amazon Cognito
 authentication behavior.",
  });
}
```

이 시나리오에서는 알려진 사용자를 자동으로 확인하는 방법을 보여줍니다. 여기에서는 예제 단계를 오케스트레이션합니다.

```
import { wait } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";

import {
  getStackOutputs,
  logCleanUpReminder,
  promptForStackName,
  promptForStackRegion,
  skipWhenErrors,
} from "./steps-common.js";
import { populateTable } from "./actions/dynamodb-actions.js";
import {
  addPreSignUpHandler,
  deleteUser,
  getUser,
  signIn,
```

```
  signUpUser,
} from "./actions/cognito-actions.js";
import {
  getLatestLogStreamForLambda,
  getLogEvents,
} from "./actions/cloudwatch-logs-actions.js";

/**
 * @typedef {{
 *   errors: Error[],
 *   password: string,
 *   users: { UserName: string, UserEmail: string }[],
 *   selectedUser?: string,
 *   stackName?: string,
 *   stackRegion?: string,
 *   token?: string,
 *   confirmDeleteSignedInUser?: boolean,
 *   TableName?: string,
 *   UserPoolClientId?: string,
 *   UserPoolId?: string,
 *   UserPoolArn?: string,
 *   AutoConfirmHandlerArn?: string,
 *   AutoConfirmHandlerName?: string
 * }} State
 */

const greeting = new ScenarioOutput(
  "greeting",
  (/** @type {State} */ state) => `This demo will populate some users into the \
database created as part of the "${state.stackName}" stack. \
Then the AutoConfirmHandler will be linked to the PreSignUp \
trigger from Cognito. Finally, you will choose a user to sign up.`,
  { skipWhen: skipWhenErrors },
);

const logPopulatingUsers = new ScenarioOutput(
  "logPopulatingUsers",
  "Populating the DynamoDB table with some users.",
  { skipWhenErrors: skipWhenErrors },
);

const logPopulatingUsersComplete = new ScenarioOutput(
  "logPopulatingUsersComplete",
  "Done populating users.",
```

```
  { skipWhen: skipWhenErrors },
);

const populateUsers = new ScenarioAction(
  "populateUsers",
  async (/** @type {State} */ state) => {
    const [_, err] = await populateTable({
      region: state.stackRegion,
      tableName: state.TableName,
      items: state.users,
    });
    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const logSetupSignUpTrigger = new ScenarioOutput(
  "logSetupSignUpTrigger",
  "Setting up the PreSignUp trigger for the Cognito User Pool.",
  { skipWhen: skipWhenErrors },
);

const setupSignUpTrigger = new ScenarioAction(
  "setupSignUpTrigger",
  async (/** @type {State} */ state) => {
    const [_, err] = await addPreSignUpHandler({
      region: state.stackRegion,
      userPoolId: state.UserPoolId,
      handlerArn: state.AutoConfirmHandlerArn,
    });
    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const logSetupSignUpTriggerComplete = new ScenarioOutput(
```

```
  "logSetupSignUpTriggerComplete",
  (
    /** @type {State} */ state,
  ) => `The lambda function "${state.AutoConfirmHandlerName}" \
has been configured as the PreSignUp trigger handler for the user pool
 "${state.UserPoolId}".`,
  { skipWhen: skipWhenErrors },
);

const selectUser = new ScenarioInput(
  "selectedUser",
  "Select a user to sign up.",
  {
    type: "select",
    choices: (/** @type {State} */ state) => state.users.map((u) => u.UserName),
    skipWhen: skipWhenErrors,
    default: (/** @type {State} */ state) => state.users[0].UserName,
  },
);

const checkIfUserAlreadyExists = new ScenarioAction(
  "checkIfUserAlreadyExists",
  async (/** @type {State} */ state) => {
    const [user, err] = await getUser({
      region: state.stackRegion,
      userPoolId: state.UserPoolId,
      username: state.selectedUser,
    });

    if (err?.name === "UserNotFoundException") {
      // Do nothing. We're not expecting the user to exist before
      // sign up is complete.
      return;
    }

    if (err) {
      state.errors.push(err);
      return;
    }

    if (user) {
      state.errors.push(
        new Error(
```

```
          `The user "${state.selectedUser}" already exists in the user pool
 "${state.UserPoolId}".`,
        ),
      );
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const createPassword = new ScenarioInput(
  "password",
  "Enter a password that has at least eight characters, uppercase, lowercase,
 numbers and symbols.",
  { type: "password", skipWhen: skipWhenErrors, default: "Abcd1234!" },
);

const logSignUpExistingUser = new ScenarioOutput(
  "logSignUpExistingUser",
  (/** @type {State} */ state) => `Signing up user "${state.selectedUser}".`,
  { skipWhen: skipWhenErrors },
);

const signUpExistingUser = new ScenarioAction(
  "signUpExistingUser",
  async (/** @type {State} */ state) => {
    const signUp = (password) =>
      signUpUser({
        region: state.stackRegion,
        userPoolClientId: state.UserPoolClientId,
        username: state.selectedUser,
        email: state.users.find((u) => u.UserName === state.selectedUser)
          .UserEmail,
        password,
      });

    let [_, err] = await signUp(state.password);

    while (err?.name === "InvalidPasswordException") {
      console.warn("The password you entered was invalid.");
      await createPassword.handle(state);
      [_, err] = await signUp(state.password);
    }
```

```
    if (err) {
      state.errors.push(err);
    }
  },
  { skipWhen: skipWhenErrors },
);

const logSignUpExistingUserComplete = new ScenarioOutput(
  "logSignUpExistingUserComplete",
  (/** @type {State} */ state) =>
    `"${state.selectedUser} was signed up successfully.`,
  { skipWhen: skipWhenErrors },
);

const logLambdaLogs = new ScenarioAction(
  "logLambdaLogs",
  async (/** @type {State} */ state) => {
    console.log(
      "Waiting a few seconds to let Lambda write to CloudWatch Logs...\n",
    );
    await wait(10);

    const [logStream, logStreamErr] = await getLatestLogStreamForLambda({
      functionName: state.AutoConfirmHandlerName,
      region: state.stackRegion,
    });
    if (logStreamErr) {
      state.errors.push(logStreamErr);
      return;
    }

    console.log(
      `Getting some recent events from log stream "${logStream.logStreamName}"`,
    );
    const [logEvents, logEventsErr] = await getLogEvents({
      functionName: state.AutoConfirmHandlerName,
      region: state.stackRegion,
      eventCount: 10,
      logStreamName: logStream.logStreamName,
    });
    if (logEventsErr) {
      state.errors.push(logEventsErr);
      return;
```

```
    }

    console.log(logEvents.map((ev) => `\t${ev.message}`).join(""));
  },
  { skipWhen: skipWhenErrors },
);

const logSignInUser = new ScenarioOutput(
  "logSignInUser",
  (/** @type {State} */ state) => `Let's sign in as ${state.selectedUser}`,
  { skipWhen: skipWhenErrors },
);

const signInUser = new ScenarioAction(
  "signInUser",
  async (/** @type {State} */ state) => {
    const [response, err] = await signIn({
      region: state.stackRegion,
      clientId: state.UserPoolClientId,
      username: state.selectedUser,
      password: state.password,
    });

    if (err?.name === "PasswordResetRequiredException") {
      state.errors.push(new Error("Please reset your password."));
      return;
    }

    if (err) {
      state.errors.push(err);
      return;
    }

    state.token = response?.AuthenticationResult?.AccessToken;
  },
  { skipWhen: skipWhenErrors },
);

const logSignInUserComplete = new ScenarioOutput(
  "logSignInUserComplete",
  (/** @type {State} */ state) =>
    `Successfully signed in. Your access token starts with:
 ${state.token.slice(0, 11)}`,
  { skipWhen: skipWhenErrors },
```

```
);

const confirmDeleteSignedInUser = new ScenarioInput(
  "confirmDeleteSignedInUser",
  "Do you want to delete the currently signed in user?",
  { type: "confirm", skipWhen: skipWhenErrors },
);

const deleteSignedInUser = new ScenarioAction(
  "deleteSignedInUser",
  async (/** @type {State} */ state) => {
    const [_, err] = await deleteUser({
      region: state.stackRegion,
      accessToken: state.token,
    });

    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: (/** @type {State} */ state) =>
      skipWhenErrors(state) || !state.confirmDeleteSignedInUser,
  },
);

const logErrors = new ScenarioOutput(
  "logErrors",
  (/** @type {State}*/ state) => {
    const errorList = state.errors
      .map((err) => ` - ${err.name}: ${err.message}`)
      .join("\n");
    return `Scenario errors found:\n${errorList}`;
  },
  {
    // Don't log errors when there aren't any!
    skipWhen: (/** @type {State} */ state) => state.errors.length === 0,
  },
);

export const AutoConfirm = (context) =>
  new Scenario(
    "AutoConfirm",
    [
```

```
      promptForStackName,
      promptForStackRegion,
      getStackOutputs,
      greeting,
      logPopulatingUsers,
      populateUsers,
      logPopulatingUsersComplete,
      logSetupSignUpTrigger,
      setupSignUpTrigger,
      logSetupSignUpTriggerComplete,
      selectUser,
      checkIfUserAlreadyExists,
      createPassword,
      logSignUpExistingUser,
      signUpExistingUser,
      logSignUpExistingUserComplete,
      logLambdaLogs,
      logSignInUser,
      signInUser,
      logSignInUserComplete,
      confirmDeleteSignedInUser,
      deleteSignedInUser,
      logCleanUpReminder,
      logErrors,
    ],
    context,
  );
```

다음은 다른 시나리오와 공유되는 단계입니다.

```
import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { getCfnOutputs } from "@aws-doc-sdk-examples/lib/sdk/cfn-outputs.js";

export const skipWhenErrors = (state) => state.errors.length > 0;

export const getStackOutputs = new ScenarioAction(
  "getStackOutputs",
  async (state) => {
```

```
    if (!state.stackName || !state.stackRegion) {
      state.errors.push(
        new Error(
          "No stack name or region provided. The stack name and \
region are required to fetch CFN outputs relevant to this example.",
        ),
      );
      return;
    }

    const outputs = await getCfnOutputs(state.stackName, state.stackRegion);
    Object.assign(state, outputs);
  },
);

export const promptForStackName = new ScenarioInput(
  "stackName",
  "Enter the name of the stack you deployed earlier.",
  { type: "input", default: "PoolsAndTriggersStack" },
);

export const promptForStackRegion = new ScenarioInput(
  "stackRegion",
  "Enter the region of the stack you deployed earlier.",
  { type: "input", default: "us-east-1" },
);

export const logCleanUpReminder = new ScenarioOutput(
  "logCleanUpReminder",
  "All done. Remember to run 'cdk destroy' to teardown the stack.",
  { skipWhen: skipWhenErrors },
);
```

Lambda 함수에서 PreSignUp 트리거에 대한 핸들러.

```
import type { PreSignUpTriggerEvent, Handler } from "aws-lambda";
import type { UserRepository } from "./user-repository";
import { DynamoDBUserRepository } from "./user-repository";

export class PreSignUpHandler {
  private userRepository: UserRepository;
```

```
  constructor(userRepository: UserRepository) {
    this.userRepository = userRepository;
  }

  private isPreSignUpTriggerSource(event: PreSignUpTriggerEvent): boolean {
    return event.triggerSource === "PreSignUp_SignUp";
  }

  private getEventUserEmail(event: PreSignUpTriggerEvent): string {
    return event.request.userAttributes.email;
  }

  async handlePreSignUpTriggerEvent(
    event: PreSignUpTriggerEvent,
  ): Promise<PreSignUpTriggerEvent> {
    console.log(
      `Received presignup from ${event.triggerSource} for user
'${event.userName}'`,
    );

    if (!this.isPreSignUpTriggerSource(event)) {
      return event;
    }

    const eventEmail = this.getEventUserEmail(event);
    console.log(`Looking up email ${eventEmail}.`);
    const storedUserInfo =
      await this.userRepository.getUserInfoByEmail(eventEmail);

    if (!storedUserInfo) {
      console.log(
        `Email ${eventEmail} not found. Email verification is required.`,
      );
      return event;
    }

    if (storedUserInfo.UserName !== event.userName) {
      console.log(
        `UserEmail ${eventEmail} found, but stored UserName
'${storedUserInfo.UserName}' does not match supplied UserName
'${event.userName}'. Verification is required.`,
      );
    } else {
      console.log(
```

```
        `UserEmail ${eventEmail} found with matching UserName
 ${storedUserInfo.UserName}. User is confirmed.`,
      );
      event.response.autoConfirmUser = true;
      event.response.autoVerifyEmail = true;
    }
    return event;
  }
}

const createPreSignUpHandler = (): PreSignUpHandler => {
  const tableName = process.env.TABLE_NAME;
  if (!tableName) {
    throw new Error("TABLE_NAME environment variable is not set");
  }

  const userRepository = new DynamoDBUserRepository(tableName);
  return new PreSignUpHandler(userRepository);
};

export const handler: Handler = async (event: PreSignUpTriggerEvent) => {
  const preSignUpHandler = createPreSignUpHandler();
  return preSignUpHandler.handlePreSignUpTriggerEvent(event);
};
```

CloudWatch Logs 작업의 모듈.

```
import {
  CloudWatchLogsClient,
  GetLogEventsCommand,
  OrderBy,
  paginateDescribeLogStreams,
} from "@aws-sdk/client-cloudwatch-logs";

/**
 * Get the latest log stream for a Lambda function.
 * @param {{ functionName: string, region: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cloudwatch-logs").LogStream | null,
 unknown]>}
 */
export const getLatestLogStreamForLambda = async ({ functionName, region }) => {
```

```
  try {
    const logGroupName = `/aws/lambda/${functionName}`;
    const cwlClient = new CloudWatchLogsClient({ region });
    const paginator = paginateDescribeLogStreams(
      { client: cwlClient },
      {
        descending: true,
        limit: 1,
        orderBy: OrderBy.LastEventTime,
        logGroupName,
      },
    );

    for await (const page of paginator) {
      return [page.logStreams[0], null];
    }
  } catch (err) {
    return [null, err];
  }
};

/**
 * Get the log events for a Lambda function's log stream.
 * @param {{
 *   functionName: string,
 *   logStreamName: string,
 *   eventCount: number,
 *   region: string
 * }} config
 * @returns {Promise<[import("@aws-sdk/client-cloudwatch-logs").OutputLogEvent[]
 | null, unknown]>}
 */
export const getLogEvents = async ({
  functionName,
  logStreamName,
  eventCount,
  region,
}) => {
  try {
    const cwlClient = new CloudWatchLogsClient({ region });
    const logGroupName = `/aws/lambda/${functionName}`;
    const response = await cwlClient.send(
      new GetLogEventsCommand({
        logStreamName: logStreamName,
```

```
        limit: eventCount,
        logGroupName: logGroupName,
      }),
    );

    return [response.events, null];
  } catch (err) {
    return [null, err];
  }
};
```

Amazon Cognito 작업 모듈.

```
import {
  AdminGetUserCommand,
  CognitoIdentityProviderClient,
  DeleteUserCommand,
  InitiateAuthCommand,
  SignUpCommand,
  UpdateUserPoolCommand,
} from "@aws-sdk/client-cognito-identity-provider";

/**
 * Connect a Lambda function to the PreSignUp trigger for a Cognito user pool
 * @param {{ region: string, userPoolId: string, handlerArn: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").UpdateUserPoolCommandOutput | null, unknown]>}
 */
export const addPreSignUpHandler = async ({
  region,
  userPoolId,
  handlerArn,
}) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({
      region,
    });

    const command = new UpdateUserPoolCommand({
      UserPoolId: userPoolId,
      LambdaConfig: {
```

```
        PreSignUp: handlerArn,
      },
    });

    const response = await cognitoClient.send(command);
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Attempt to register a user to a user pool with a given username and password.
 * @param {{
 *   region: string,
 *   userPoolClientId: string,
 *   username: string,
 *   email: string,
 *   password: string
 * }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").SignUpCommandOutput | null, unknown]>}
 */
export const signUpUser = async ({
  region,
  userPoolClientId,
  username,
  email,
  password,
}) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({
      region,
    });

    const response = await cognitoClient.send(
      new SignUpCommand({
        ClientId: userPoolClientId,
        Username: username,
        Password: password,
        UserAttributes: [{ Name: "email", Value: email }],
      }),
    );
    return [response, null];
```

```
  } catch (err) {
    return [null, err];
  }
};

/**
 * Sign in a user to Amazon Cognito using a username and password authentication
 flow.
 * @param {{ region: string, clientId: string, username: string, password:
 string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").InitiateAuthCommandOutput | null, unknown]>}
 */
export const signIn = async ({ region, clientId, username, password }) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({ region });
    const response = await cognitoClient.send(
      new InitiateAuthCommand({
        AuthFlow: "USER_PASSWORD_AUTH",
        ClientId: clientId,
        AuthParameters: { USERNAME: username, PASSWORD: password },
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Retrieve an existing user from a user pool.
 * @param {{ region: string, userPoolId: string, username: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").AdminGetUserCommandOutput | null, unknown]>}
 */
export const getUser = async ({ region, userPoolId, username }) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({ region });
    const response = await cognitoClient.send(
      new AdminGetUserCommand({
        UserPoolId: userPoolId,
        Username: username,
      }),
    );
```

```
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Delete the signed-in user. Useful for allowing a user to delete their
 * own profile.
 * @param {{ region: string, accessToken: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").DeleteUserCommandOutput | null, unknown]>}
 */
export const deleteUser = async ({ region, accessToken }) => {
  try {
    const client = new CognitoIdentityProviderClient({ region });
    const response = await client.send(
      new DeleteUserCommand({ AccessToken: accessToken }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

DynamoDB 작업의 모듈.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  BatchWriteCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";

/**
 * Populate a DynamoDB table with provide items.
 * @param {{ region: string, tableName: string, items: Record<string,
 unknown>[] }} config
 * @returns {Promise<[import("@aws-sdk/lib-dynamodb").BatchWriteCommandOutput |
 null, unknown]>}
 */
export const populateTable = async ({ region, tableName, items }) => {
```

```
  try {
    const ddbClient = new DynamoDBClient({ region });
    const docClient = DynamoDBDocumentClient.from(ddbClient);
    const response = await docClient.send(
      new BatchWriteCommand({
        RequestItems: {
          [tableName]: items.map((item) => ({
            PutRequest: {
              Item: item,
            },
          })),
        },
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 다음 주제를 참조하세요.
  - DeleteUser
  - InitiateAuth
  - SignUp
  - UpdateUserPool

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 Lambda 함수를 사용하여 알려진 Amazon Cognito 사용자를 자동으로 마이그레이션

다음 코드 예제는 Lambda 함수를 사용하여 알려진 Amazon Cognito 사용자를 자동으로 마이그레이션하는 방법을 보여줍니다.

- `MigrateUser` 트리거에 대해 Lambda 함수를 호출하도록 사용자 풀을 구성합니다.
- 사용자 풀에 없는 사용자 이름과 이메일을 사용하여 Amazon Cognito에 로그인합니다.

- Lambda 함수는 DynamoDB 테이블을 스캔하고 알려진 사용자를 사용자 풀로 자동으로 마이그레이션합니다.
- 암호 찾기 흐름을 수행하여 마이그레이션된 사용자의 암호를 재설정합니다.
- 새 사용자로 로그인한 다음 리소스를 정리합니다.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
import (
 "context"
 "errors"
 "fmt"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// MigrateUser separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type MigrateUser struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}
```

```
// NewMigrateUser constructs a new migrate user runner.
func NewMigrateUser(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) MigrateUser {
 scenario := MigrateUser{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddMigrateUserTrigger adds a Lambda handler as an invocation target for the
 MigrateUser trigger.
func (runner *MigrateUser) AddMigrateUserTrigger(ctx context.Context, userPoolId
 string, functionArn string) {
 log.Printf("Let's add a Lambda function to handle the MigrateUser trigger from
 Cognito.\n" +
  "This trigger happens when an unknown user signs in, and lets your function
 take action before Cognito\n" +
  "rejects the user.\n\n")
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
  actions.TriggerInfo{Trigger: actions.UserMigration, HandlerArn:
 aws.String(functionArn)})
 if err != nil {
  panic(err)
 }
 log.Printf("Lambda function %v added to user pool %v to handle the MigrateUser
 trigger.\n",
  functionArn, userPoolId)

 log.Println(strings.Repeat("-", 88))
}

// SignInUser adds a new user to the known users table and signs that user in to
 Amazon Cognito.
func (runner *MigrateUser) SignInUser(ctx context.Context, usersTable string,
 clientId string) (bool, actions.User) {
 log.Println("Let's sign in a user to your Cognito user pool. When the username
 and email matches an entry in the\n" +
```

```
 "DynamoDB known users table, the email is automatically verified and the user
is migrated to the Cognito user pool.")

user := actions.User{}
user.UserName = runner.questioner.Ask("\nEnter a username:")
user.UserEmail = runner.questioner.Ask("\nEnter an email that you own. This
email will be used to confirm user migration\n" +
 "during this example:")

runner.helper.AddKnownUser(ctx, usersTable, user)

var err error
var resetRequired *types.PasswordResetRequiredException
var authResult *types.AuthenticationResultType
signedIn := false
for !signedIn && resetRequired == nil {
 log.Printf("Signing in to Cognito as user '%v'. The expected result is a
PasswordResetRequiredException.\n\n", user.UserName)
 authResult, err = runner.cognitoActor.SignIn(ctx, clientId, user.UserName, "_")
 if err != nil {
  if errors.As(err, &resetRequired) {
   log.Printf("\nUser '%v' is not in the Cognito user pool but was found in the
DynamoDB known users table.\n"+
    "User migration is started and a password reset is required.",
user.UserName)
  } else {
   panic(err)
  }
 } else {
  log.Printf("User '%v' successfully signed in. This is unexpected and probably
means you have not\n"+
   "cleaned up a previous run of this scenario, so the user exist in the Cognito
user pool.\n"+
   "You can continue this example and select to clean up resources, or manually
remove\n"+
   "the user from your user pool and try again.", user.UserName)
  runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
*authResult.AccessToken)
  signedIn = true
 }
}

log.Println(strings.Repeat("-", 88))
return resetRequired != nil, user
```

```
}

// ResetPassword starts a password recovery flow.
func (runner *MigrateUser) ResetPassword(ctx context.Context, clientId string,
 user actions.User) {
 wantCode := runner.questioner.AskBool(fmt.Sprintf("In order to migrate the user
 to Cognito, you must be able to receive a confirmation\n"+
  "code by email at %v. Do you want to send a code (y/n)?", user.UserEmail), "y")
 if !wantCode {
  log.Println("To complete this example and successfully migrate a user to
 Cognito, you must enter an email\n" +
   "you own that can receive a confirmation code.")
  return
 }
 codeDelivery, err := runner.cognitoActor.ForgotPassword(ctx, clientId,
 user.UserName)
 if err != nil {
  panic(err)
 }
 log.Printf("\nA confirmation code has been sent to %v.",
 *codeDelivery.Destination)
 code := runner.questioner.Ask("Check your email and enter it here:")

 confirmed := false
 password := runner.questioner.AskPassword("\nEnter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !confirmed {
  log.Printf("\nConfirming password reset for user '%v'.\n", user.UserName)
  err = runner.cognitoActor.ConfirmForgotPassword(ctx, clientId, code,
 user.UserName, password)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("\nEnter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   confirmed = true
  }
 }
 log.Printf("User '%v' successfully confirmed and migrated.\n", user.UserName)
 log.Println("Signing in with your username and password...")
```

```
 authResult, err := runner.cognitoActor.SignIn(ctx, clientId, user.UserName,
 password)
 if err != nil {
  panic(err)
 }
 log.Printf("Successfully signed in. Your access token starts with: %v...\n",
 (*authResult.AccessToken)[:10])
 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)

 log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
func (runner *MigrateUser) Run(ctx context.Context, stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]

 runner.AddMigrateUserTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["MigrateUserFunctionArn"])
 runner.resources.triggers = append(runner.resources.triggers,
 actions.UserMigration)
 resetNeeded, user := runner.SignInUser(ctx, stackOutputs["TableName"],
 stackOutputs["UserPoolClientId"])
 if resetNeeded {
  runner.helper.ListRecentLogEvents(ctx, stackOutputs["MigrateUserFunction"])
  runner.ResetPassword(ctx, stackOutputs["UserPoolClientId"], user)
 }
```

```
 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Lambda 함수를 사용하여 MigrateUser 트리거를 처리합니다.

```
import (
 "context"
 "log"
 "os"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
)

const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string `dynamodbav:"UserName"`
 UserEmail string `dynamodbav:"UserEmail"`
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the MigrateUser event by looking up a user in an Amazon
 DynamoDB table and
// specifying whether they should be migrated to the user pool.
```

```
func (h *handler) HandleRequest(ctx context.Context, event
 events.CognitoEventUserPoolsMigrateUser)
 (events.CognitoEventUserPoolsMigrateUser, error) {
 log.Printf("Received migrate trigger from %v for user '%v'",
 event.TriggerSource, event.UserName)
 if event.TriggerSource != "UserMigration_Authentication" {
  return event, nil
 }
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserName: event.UserName,
 }
 log.Printf("Looking up user '%v' in table %v.\n", user.UserName, tableName)
 filterEx := expression.Name("UserName").Equal(expression.Value(user.UserName))
 expr, err := expression.NewBuilder().WithFilter(filterEx).Build()
 if err != nil {
  log.Printf("Error building expression to query for user '%v'.\n",
 user.UserName)
  return event, err
 }
 output, err := h.dynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName:                 aws.String(tableName),
  FilterExpression:          expr.Filter(),
  ExpressionAttributeNames:  expr.Names(),
  ExpressionAttributeValues: expr.Values(),
 })
 if err != nil {
  log.Printf("Error looking up user '%v'.\n", user.UserName)
  return event, err
 }
 if len(output.Items) == 0 {
  log.Printf("User '%v' not found, not migrating user.\n", user.UserName)
  return event, err
 }

 var users []UserInfo
 err = attributevalue.UnmarshalListOfMaps(output.Items, &users)
 if err != nil {
  log.Printf("Couldn't unmarshal DynamoDB items. Here's why: %v\n", err)
  return event, err
 }

 user = users[0]
```

```
 log.Printf("UserName '%v' found with email %v. User is migrated and must reset
 password.\n", user.UserName, user.UserEmail)
 event.CognitoEventUserPoolsMigrateUserResponse.UserAttributes =
 map[string]string{
  "email":          user.UserEmail,
  "email_verified": "true", // email_verified is required for the forgot password
 flow.
 }
 event.CognitoEventUserPoolsMigrateUserResponse.FinalUserStatus =
 "RESET_REQUIRED"
 event.CognitoEventUserPoolsMigrateUserResponse.MessageAction = "SUPPRESS"

 return event, err
}

func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

일반적인 작업을 수행하는 구조체를 생성합니다.

```
import (
 "context"
 "log"
 "strings"
 "time"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
```

```
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
```

```
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName
 string) (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
 }
}
```

```
// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context,
 functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Amazon Cognito 작업을 래핑하는 구조체를 생성합니다.

```
import (
 "context"
 "errors"
 "log"

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

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

```
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId
 string, triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
```

```
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
```

```
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}
```

```
// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken
 string) error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId
 string, userName string, userEmail string) error {
```

```
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



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

```
}
```

DynamoDB 작업을 래핑하는 구조체를 생성합니다.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}
```

```
// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
```

```
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user
 User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

CloudWatch Logs 작업을 래핑하는 구조체를 생성합니다.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
```

```
)

type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx,
 &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
```

```
  events = output.Events
 }
 return events, err
}
```

CloudFormation 작업을 래핑하는 구조체를 생성합니다.

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName
 string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

리소스를 정리합니다.

```
import (
 "context"
 "log"
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 \n" +
    "that were created for this scenario.")
  }
 }()
```

```
 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(ctx, accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
 HandlerArn: nil}
  }
  err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
   panic(err)
  }
  log.Println("Removed Cognito triggers from user pool.")
 } else {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
 }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 다음 주제를 참조하세요.
  - ConfirmForgotPassword
  - DeleteUser
  - ForgotPassword
  - InitiateAuth
  - SignUp
  - UpdateUserPool

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 M2M 클라이언트 토큰을 가져오고 Amazon Cognito에서 용어 검색

다음 코드 예제에서는 Amazon Cognito machine-to-machine) 권한 부여 및 용어 검색을 사용하는 방법을 보여줍니다.

- Cognito 리소스(사용자 풀, 리소스 서버, 앱 클라이언트, 용어 문서)를 사용하여 CloudFormation 스택을 배포합니다.
- DescribeUserPoolClient를 사용하여 앱 클라이언트 보안 암호를 검색합니다.
- GetClientToken을 사용하여 M2M 액세스 토큰을 가져옵니다.
- DescribeTermsByClient를 사용하여 용어 문서를 검색합니다.
- CloudFormation 스택을 삭제하여 정리합니다.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
class ClientTokenAndTermsScenario:
    """
    Runs an interactive scenario that demonstrates the Amazon Cognito
    GetClientToken and DescribeTermsByClient APIs.
    """

    def __init__(
        self,
        cognito_wrapper: CognitoIdentityProviderWrapper,
        cfn_client: BaseClient,
    ):
```

```
        """
        :param cognito_wrapper: An instance of the CognitoIdentityProviderWrapper
class.
        :param cfn_client: A Boto3 CloudFormation client.
        """
        self.cognito_wrapper = cognito_wrapper
        self.cfn_client = cfn_client
        self.stack_name = None
        self.user_pool_id = None
        self.client_id = None
        self.client_secret = None

    def run(self) -> None:
        """Runs the scenario from start to finish."""
        print(SEPARATOR)
        print("Welcome to the Amazon Cognito Client Token and Terms Scenario.")
        print(SEPARATOR)
        print("This example creates Cognito resources in a CloudFormation stack,
then")
        print("demonstrates the new GetClientToken and DescribeTermsByClient
APIs.")
        print()

        try:
            self.setup()
            self.get_client_secret()
            self.demonstrate_get_client_token()
            self.demonstrate_describe_terms()
        except Exception:
            logger.exception("An error occurred during the scenario.")
        finally:
            self.cleanup()

        print(SEPARATOR)
        print("Amazon Cognito Client Token and Terms scenario completed.")

    # ----------------------------------------------------------------------
    # Phase 1: Setup - deploy CloudFormation stack
    # ----------------------------------------------------------------------
    def setup(self) -> None:
        """Deploys the CloudFormation prerequisite stack and retrieves
outputs."""
        stack_name = q.ask(
```

```
            f"\nEnter a name for the CloudFormation stack [{DEFAULT_STACK_NAME}]:
"
        )
        if not stack_name:
            stack_name = DEFAULT_STACK_NAME

        # Check whether the stack name already exists.
        while self._stack_exists(stack_name):
            print(f"A stack named '{stack_name}' already exists.")
            stack_name = q.ask("Enter a different stack name: ")

        self.stack_name = stack_name

        template_body = self._read_template()
        print(f"Deploying CloudFormation stack: {self.stack_name}")
        self.cfn_client.create_stack(
            StackName=self.stack_name,
            TemplateBody=template_body,
            Capabilities=["CAPABILITY_IAM"],
        )
        print(f"CloudFormation stack creation started: {self.stack_name}")
        print("Waiting for CloudFormation stack creation to complete...")

        waiter = self.cfn_client.get_waiter("stack_create_complete")
        try:
            waiter.wait(
                StackName=self.stack_name,
                WaiterConfig={"Delay": 10, "MaxAttempts": 60},
            )
        except WaiterError:
            logger.error("Stack creation failed or timed out.")
            self._print_stack_events()
            raise

        print("CloudFormation stack creation complete.")
        self._retrieve_stack_outputs()
        print(SEPARATOR)

    def _stack_exists(self, stack_name: str) -> bool:
        """Returns True if a stack with the given name exists and is not
deleted."""
        if not stack_name:
            return False
        try:
```

```
            response = self.cfn_client.describe_stacks(StackName=stack_name)
            stacks = response.get("Stacks", list())
            for stack in stacks:
                status = stack.get("StackStatus", "")
                if "DELETE_COMPLETE" not in status:
                    return True
            return False
        except ClientError as err:
            if "does not exist" in str(err):
                return False
            raise

    def _read_template(self) -> str:
        """Reads the CloudFormation template from the local file system."""
        template_path = os.path.join(
            os.path.dirname(os.path.abspath(__file__)), CFN_TEMPLATE_PATH
        )
        if not os.path.exists(template_path):
            raise FileNotFoundError(
                f"CloudFormation template not found at {template_path}. "
                "Ensure the resources/cfn_template.yaml file exists."
            )
        with open(template_path, "r") as f:
            return f.read()

    def _retrieve_stack_outputs(self) -> None:
        """Retrieves and displays outputs from the CloudFormation stack."""
        response = self.cfn_client.describe_stacks(StackName=self.stack_name)
        stacks = response.get("Stacks", list())
        if not stacks:
            raise RuntimeError(f"Stack '{self.stack_name}' not found.")

        outputs = stacks[0].get("Outputs", list())
        for output in outputs:
            key = output.get("OutputKey", "")
            value = output.get("OutputValue", "")
            print(f"Stack output {key}: {value}")
            if key == "UserPoolId":
                self.user_pool_id = value
            elif key == "ClientId":
                self.client_id = value

        if not self.user_pool_id or not self.client_id:
            raise RuntimeError(
```

```
                "Failed to retrieve UserPoolId or ClientId from stack outputs."
            )

    def _print_stack_events(self) -> None:
        """Prints recent stack events to help diagnose failures."""
        try:
            response =
self.cfn_client.describe_stack_events(StackName=self.stack_name)
            events = response.get("StackEvents", list())
            for event in events[:10]:
                status = event.get("ResourceStatus", "")
                reason = event.get("ResourceStatusReason", "")
                resource = event.get("LogicalResourceId", "")
                if "FAILED" in status:
                    print(f"  FAILED: {resource} - {reason}")
        except ClientError:
            logger.warning("Could not retrieve stack events.")

    # ------------------------------------------------------------------
    # Phase 2: Retrieve the app client secret
    # ------------------------------------------------------------------
    def get_client_secret(self) -> None:
        """Calls DescribeUserPoolClient to retrieve the client secret."""
        print(SEPARATOR)
        print("Retrieving the app client secret...")
        client_info = self.cognito_wrapper.describe_user_pool_client(
            self.user_pool_id, self.client_id
        )
        self.client_secret = client_info.get("ClientSecret", "")
        if not self.client_secret:
            raise RuntimeError(
                "Client secret not found. Ensure the app client was created "
                "with GenerateSecret: true."
            )
        print("Client secret retrieved successfully.")
        print(SEPARATOR)

    # ------------------------------------------------------------------
    # Phase 3: Demonstrate GetClientToken
    # ------------------------------------------------------------------
    def demonstrate_get_client_token(self) -> None:
        """Demonstrates obtaining M2M access tokens with GetClientToken."""
        print(SEPARATOR)
        print("Obtaining an M2M access token with GetClientToken...")
```

```
        print()

        # 3a: Get token with all scopes (no scopes param = all configured scopes)
        print("Step 1: Request a token with all configured scopes.")
        auth_result = self.cognito_wrapper.get_client_token(
            client_id=self.client_id,
            client_secret=self.client_secret,
        )
        self._display_token_result(auth_result, "all scopes")

        q.ask("\nPress Enter to continue...")

        # 3b: Get token with specific scopes
        print()
        print("Step 2: Request a token with a specific scope (my-m2m-api/read).")
        scoped_result = self.cognito_wrapper.get_client_token(
            client_id=self.client_id,
            client_secret=self.client_secret,
            scopes=["my-m2m-api/read"],
        )
        self._display_token_result(scoped_result, "my-m2m-api/read")

        print()
        print("Token was successfully scoped to the requested permissions.")
        print(SEPARATOR)

    @staticmethod
    def _display_token_result(auth_result: dict, scope_description: str) -> None:
        """Displays token metadata."""
        token_type = auth_result.get("TokenType", "Unknown")
        expires_in = auth_result.get("ExpiresIn", 0)
        access_token = auth_result.get("AccessToken", "")
        preview = access_token[:20] + "..." if len(access_token) > 20 else
access_token
        print(f"  Scope:      {scope_description}")
        print(f"  Token type: {token_type}")
        print(f"  Expires in: {expires_in} seconds")
        print(f"  Token preview: {preview}")

    # ------------------------------------------------------------------
    # Phase 4: Demonstrate DescribeTermsByClient
    # ------------------------------------------------------------------
    def demonstrate_describe_terms(self) -> None:
```

```
        """Demonstrates discovering Terms documents with
DescribeTermsByClient."""
        print(SEPARATOR)
        print("Discovering Terms documents with DescribeTermsByClient...")
        print()

        # 4a: Look up terms-of-use
        print("Looking up terms-of-use for the app client...")
        terms_of_use = self.cognito_wrapper.describe_terms_by_client(
            user_pool_id=self.user_pool_id,
            client_id=self.client_id,
            terms_name="terms-of-use",
        )
        self._display_terms(terms_of_use)

        q.ask("\nPress Enter to continue...")
        print()

        # 4b: Look up privacy-policy
        print("Looking up privacy-policy for the app client...")
        privacy_policy = self.cognito_wrapper.describe_terms_by_client(
            user_pool_id=self.user_pool_id,
            client_id=self.client_id,
            terms_name="privacy-policy",
        )
        self._display_terms(privacy_policy)

        print()
        print(
            "DescribeTermsByClient lets you discover which Terms documents are "
            "associated with a client without knowing the Terms resource ID up
front."
        )
        print(SEPARATOR)

    @staticmethod
    def _display_terms(terms: dict) -> None:
        """Displays a Terms document's details."""
        print(f"  Terms ID:          {terms.get('TermsId', 'N/A')}")
        print(f"  Terms name:        {terms.get('TermsName', 'N/A')}")
        print(f"  Client ID:         {terms.get('ClientId', 'N/A')}")
        print(f"  User pool ID:      {terms.get('UserPoolId', 'N/A')}")
        print(f"  Enforcement:       {terms.get('Enforcement', 'N/A')}")
        print(f"  Terms source:      {terms.get('TermsSource', 'N/A')}")
```

```
        links = terms.get("Links", dict())
        if links:
            print("  Links:")
            for lang_tag, url in links.items():
                print(f"    {lang_tag}: {url}")
        # Cognito returns CreationDate / LastModifiedDate as datetime objects
        # (boto3 deserializes the timestamps), so display them directly.
        creation_date = terms.get("CreationDate", None)
        if creation_date is not None:
            print(f"  Created:           {creation_date}")
        modified_date = terms.get("LastModifiedDate", None)
        if modified_date is not None:
            print(f"  Last modified:     {modified_date}")

    # ------------------------------------------------------------------
    # Phase 5: Cleanup - delete CloudFormation stack
    # ------------------------------------------------------------------
    def cleanup(self) -> None:
        """Prompts the user and deletes the CloudFormation stack."""
        if self.stack_name is None:
            return

        print(SEPARATOR)
        answer = q.ask(
            "Do you want to delete the CloudFormation stack and all resources?
(y/n) "
        )
        if answer.lower() != "y":
            print(
                f"Resources will remain. You can delete the stack later "
                f"through the AWS Console or CLI."
            )
            print(f"Stack name: {self.stack_name}")
            return

        print(
            f"CloudFormation stack '{self.stack_name}' is being deleted. "
            f"This may take a few minutes."
        )
        try:
            self.cfn_client.delete_stack(StackName=self.stack_name)
            waiter = self.cfn_client.get_waiter("stack_delete_complete")
            print("Waiting for CloudFormation stack deletion to complete...")
            waiter.wait(
```

```
                StackName=self.stack_name,
                WaiterConfig={"Delay": 10, "MaxAttempts": 60},
            )
            print(f"CloudFormation stack '{self.stack_name}' has been deleted.")
            print("All Cognito resources have been removed.")
        except (ClientError, WaiterError):
            logger.warning(
                "Stack deletion encountered an error. Attempting force delete."
            )
            try:
                self.cfn_client.delete_stack(
                    StackName=self.stack_name,
                    DeletionMode="FORCE_DELETE_STACK",
                )
                print("Force delete initiated. Please verify in the console.")
            except ClientError:
                logger.error(
                    "Force delete also failed. Clean up manually via the
 console."
                )
```

Amazon Cognito 자격 증명 공급자 작업을 래핑하는 클래스를 생성합니다.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito Identity Provider actions."""

    def __init__(self, cognito_idp_client: BaseClient):
        """
        Initializes the CognitoIdentityProviderWrapper with an Amazon Cognito
        Identity Provider client.

        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        """
        self.cognito_idp_client = cognito_idp_client
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 다음 주제를 참조하세요.
  - DescribeTermsByClient

- DescribeUserPoolClient
- GetClientToken

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

사용자 풀 시작하기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 사용자 풀 생성
- 앱 클라이언트 생성
- 사용자 풀에 대한 도메인 설정
- 관리자로 사용자 생성
- 자체 등록 활성화
- 사용자 풀의 사용자 나열
- 리소스 정리

Bash

AWS CLI Bash 스크립트 사용

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 샘플 개발자 튜토리얼 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 배워보세요.

```
#!/bin/bash

# Amazon Cognito User Pools Getting Started Script
# This script creates and configures an Amazon Cognito user pool with an app
 client

set -euo pipefail

# Security: Set restrictive umask
```

```
umask 0077

# Set up logging with secure permissions
LOG_FILE="cognito-user-pool-setup.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Cognito User Pool setup script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
  local output=$1
  local cmd=$2

  if echo "$output" | grep -qi "error\|failed"; then
    echo "ERROR: Command failed: $cmd" >&2
    echo "Output: $output" >&2
    cleanup_on_error
    exit 1
  fi
}

# Function to check AWS CLI return code
check_aws_error() {
  local exit_code=$?
  local cmd=$1

  if [ $exit_code -ne 0 ]; then
    echo "ERROR: AWS CLI command failed with exit code $exit_code: $cmd" >&2
    cleanup_on_error
    exit "$exit_code"
  fi
}

# Function to clean up resources on error
cleanup_on_error() {
  echo "Error encountered. Attempting to clean up resources..." >&2

  if [ -n "${DOMAIN_NAME:-}" ] && [ -n "${USER_POOL_ID:-}" ]; then
    echo "Deleting user pool domain: $DOMAIN_NAME" >&2
    aws cognito-idp delete-user-pool-domain \
      --user-pool-id "$USER_POOL_ID" \
```

```
      --domain "$DOMAIN_NAME" 2>/dev/null || true
  fi

  if [ -n "${USER_POOL_ID:-}" ]; then
    echo "Deleting user pool: $USER_POOL_ID" >&2
    aws cognito-idp delete-user-pool \
      --user-pool-id "$USER_POOL_ID" 2>/dev/null || true
  fi
}

# Set trap for cleanup on exit
trap cleanup_on_error EXIT ERR

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
  echo "ERROR: AWS CLI is not installed" >&2
  exit 1
fi

# Validate jq is installed
if ! command -v jq &> /dev/null; then
  echo "ERROR: jq is not installed" >&2
  exit 1
fi

# Validate openssl is installed
if ! command -v openssl &> /dev/null; then
  echo "ERROR: openssl is not installed" >&2
  exit 1
fi

if ! aws sts get-caller-identity &> /dev/null; then
  echo "ERROR: AWS CLI is not configured or credentials are invalid" >&2
  exit 1
fi

# Get the current AWS region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
  AWS_REGION="us-east-1" # Default region if not configured
fi
echo "Using AWS Region: $AWS_REGION"

# Validate region format
```

```
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
  echo "ERROR: Invalid AWS region format: $AWS_REGION" >&2
  exit 1
fi

# Generate random identifier for resource names using secure method
RANDOM_ID=$(openssl rand -hex 6)
if [ -z "$RANDOM_ID" ]; then
  echo "ERROR: Failed to generate random identifier" >&2
  exit 1
fi

USER_POOL_NAME="MyUserPool-${RANDOM_ID}"
APP_CLIENT_NAME="MyAppClient-${RANDOM_ID}"
DOMAIN_NAME="my-auth-domain-${RANDOM_ID}"

# Validate resource names don't exceed limits
if [ ${#USER_POOL_NAME} -gt 128 ]; then
  echo "ERROR: User pool name exceeds maximum length of 128 characters" >&2
  exit 1
fi

if [ ${#APP_CLIENT_NAME} -gt 128 ]; then
  echo "ERROR: App client name exceeds maximum length of 128 characters" >&2
  exit 1
fi

echo "Using random identifier: $RANDOM_ID"
echo "User pool name: $USER_POOL_NAME"
echo "App client name: $APP_CLIENT_NAME"
echo "Domain name: $DOMAIN_NAME"

# Step 1: Create a User Pool
echo "Creating user pool..."
USER_POOL_OUTPUT=$(aws cognito-idp create-user-pool \
  --pool-name "$USER_POOL_NAME" \
  --auto-verified-attributes email \
  --username-attributes email \
  --policies '{"PasswordPolicy":
{"MinimumLength":12,"RequireUppercase":true,"RequireLowercase":true,"RequireNumbers":true
 \
  --schema '[{"Name":"email","Required":true,"Mutable":true}]' \
  --mfa-configuration OFF \
```

```
  --user-attribute-update-settings '{"AttributesRequireVerificationBeforeUpdate":
["email"]}' \
  --account-recovery-setting
 'RecoveryMechanisms=[{Name=verified_email,Priority=1}]' \
  --deletion-protection INACTIVE \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "create-user-pool"

# Extract the User Pool ID using jq for safety
USER_POOL_ID=$(echo "$USER_POOL_OUTPUT" | jq -r '.UserPool.Id // empty')
if [ -z "$USER_POOL_ID" ]; then
  echo "ERROR: Failed to extract User Pool ID" >&2
  exit 1
fi

# Validate User Pool ID format
if ! [[ "$USER_POOL_ID" =~ ^[a-z]{2}-[a-z]+-[0-9]+_[a-zA-Z0-9]+$ ]]; then
  echo "ERROR: Invalid User Pool ID format: $USER_POOL_ID" >&2
  exit 1
fi

echo "User Pool created with ID: $USER_POOL_ID"

USER_POOL_ARN=$(echo "$USER_POOL_OUTPUT" | jq -r '.UserPool.Arn // empty')
aws cognito-idp tag-resource \
  --resource-arn "$USER_POOL_ARN" \
  --tags project=doc-smith,tutorial=amazon-cognito-gs

# Wait for user pool to be ready
echo "Waiting for user pool to be ready..."
sleep 5

# Step 2: Create an App Client with enhanced security
echo "Creating app client..."
APP_CLIENT_OUTPUT=$(aws cognito-idp create-user-pool-client \
  --user-pool-id "$USER_POOL_ID" \
  --client-name "$APP_CLIENT_NAME" \
  --no-generate-secret \
  --explicit-auth-flows ALLOW_REFRESH_TOKEN_AUTH ALLOW_USER_PASSWORD_AUTH \
  --callback-urls '["https://localhost:3000/callback"]' \
  --allowed-o-auth-flows 'code' \
  --allowed-o-auth-scopes 'openid' 'email' 'profile' \
  --allowed-o-auth-flows-user-pool-client \
```

```
  --prevent-user-existence-errors ENABLED \
  --enable-token-revocation \
  --access-token-validity 1 \
  --id-token-validity 1 \
  --refresh-token-validity 30 \
  --token-validity-units 'AccessToken=hours,IdToken=hours,RefreshToken=days' \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "create-user-pool-client"

# Extract the Client ID using jq for safety
CLIENT_ID=$(echo "$APP_CLIENT_OUTPUT" | jq -r '.UserPoolClient.ClientId //
 empty')
if [ -z "$CLIENT_ID" ]; then
  echo "ERROR: Failed to extract Client ID" >&2
  cleanup_on_error
  exit 1
fi

# Validate Client ID format
if ! [[ "$CLIENT_ID" =~ ^[a-z0-9]{26}$ ]]; then
  echo "ERROR: Invalid Client ID format: $CLIENT_ID" >&2
  cleanup_on_error
  exit 1
fi

echo "App Client created with ID: $CLIENT_ID"

# Step 3: Set Up a Domain for Your User Pool
echo "Setting up user pool domain..."
DOMAIN_OUTPUT=$(aws cognito-idp create-user-pool-domain \
  --user-pool-id "$USER_POOL_ID" \
  --domain "$DOMAIN_NAME" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "create-user-pool-domain"
echo "Domain created: $DOMAIN_NAME.auth.$AWS_REGION.amazoncognito.com"

# Step 4: View User Pool Details
echo "Retrieving user pool details..."
USER_POOL_DETAILS=$(aws cognito-idp describe-user-pool \
  --user-pool-id "$USER_POOL_ID" \
  --region "$AWS_REGION" \
  2>&1)
```

```
check_aws_error "describe-user-pool"
echo "User Pool details retrieved successfully"

# Step 5: View App Client Details
echo "Retrieving app client details..."
APP_CLIENT_DETAILS=$(aws cognito-idp describe-user-pool-client \
  --user-pool-id "$USER_POOL_ID" \
  --client-id "$CLIENT_ID" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "describe-user-pool-client"
echo "App Client details retrieved successfully"

# Step 6: Create a User (Admin)
echo "Creating admin user..."
ADMIN_USER_EMAIL="admin@example.com"
TEMP_PASSWORD="$(openssl rand -base64 12 | tr -d '\n')!@#"
if [ -z "$TEMP_PASSWORD" ]; then
  echo "ERROR: Failed to generate temporary password" >&2
  cleanup_on_error
  exit 1
fi

ADMIN_USER_OUTPUT=$(aws cognito-idp admin-create-user \
  --user-pool-id "$USER_POOL_ID" \
  --username "$ADMIN_USER_EMAIL" \
  --user-attributes Name=email,Value="$ADMIN_USER_EMAIL"
 Name=email_verified,Value=true \
  --temporary-password "$TEMP_PASSWORD" \
  --message-action SUPPRESS \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "admin-create-user"
echo "Admin user created: $ADMIN_USER_EMAIL"

# Securely clear temporary password from memory
unset TEMP_PASSWORD

# Step 7: Self-Registration
echo "Demonstrating self-registration..."
USER_EMAIL="user@example.com"
USER_PASSWORD="SecurePassword123!"
SIGNUP_OUTPUT=$(aws cognito-idp sign-up \
  --client-id "$CLIENT_ID" \
```

```
  --username "$USER_EMAIL" \
  --password "$USER_PASSWORD" \
  --user-attributes Name=email,Value="$USER_EMAIL" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "sign-up"
echo "User signed up: $USER_EMAIL"
echo "A confirmation code would be sent to the user's email in a real scenario"

# Securely clear password from memory
unset USER_PASSWORD

echo ""
echo "================================================="
echo "IMPORTANT: In a real scenario, the user would receive"
echo "a confirmation code via email. For this demo, we'll"
echo "use admin-confirm-sign-up instead."
echo "================================================="
echo ""

# Step 8: Confirm User Registration (using admin privileges for demo)
echo "Confirming user registration (admin method)..."
CONFIRM_OUTPUT=$(aws cognito-idp admin-confirm-sign-up \
  --user-pool-id "$USER_POOL_ID" \
  --username "$USER_EMAIL" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "admin-confirm-sign-up"
echo "User confirmed: $USER_EMAIL"

# Step 9: Set permanent password for user
echo "Setting permanent password for user..."
SET_PASSWORD="SecureUserPassword123!"
SET_PASS_OUTPUT=$(aws cognito-idp admin-set-user-password \
  --user-pool-id "$USER_POOL_ID" \
  --username "$USER_EMAIL" \
  --password "$SET_PASSWORD" \
  --permanent \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "admin-set-user-password"
echo "Permanent password set for user"

unset SET_PASSWORD
```

```
# Step 10: Authenticate a User
echo "Authenticating user..."
AUTH_PASSWORD="SecureUserPassword123!"
AUTH_OUTPUT=$(aws cognito-idp initiate-auth \
  --client-id "$CLIENT_ID" \
  --auth-flow USER_PASSWORD_AUTH \
  --auth-parameters "USERNAME=$USER_EMAIL,PASSWORD=$AUTH_PASSWORD" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "initiate-auth"
echo "User authenticated successfully"

unset AUTH_PASSWORD

# Extract auth tokens securely
ID_TOKEN=$(echo "$AUTH_OUTPUT" | jq -r '.AuthenticationResult.IdToken // empty')
ACCESS_TOKEN=$(echo "$AUTH_OUTPUT" | jq -r '.AuthenticationResult.AccessToken //
 empty')

# Validate tokens exist
if [ -z "$ID_TOKEN" ] || [ -z "$ACCESS_TOKEN" ]; then
  echo "WARNING: Failed to extract authentication tokens" >&2
else
  echo "Authentication tokens obtained successfully"
fi

# Securely clear tokens from memory
unset ID_TOKEN
unset ACCESS_TOKEN

# Step 11: List Users in the User Pool
echo "Listing users in the user pool..."
USERS_OUTPUT=$(aws cognito-idp list-users \
  --user-pool-id "$USER_POOL_ID" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "list-users"
echo "Users listed successfully"

# Display summary of created resources
echo ""
echo "================================================="
echo "RESOURCE SUMMARY"
```

```
echo "================================================="
echo "User Pool ID: $USER_POOL_ID"
echo "User Pool Name: $USER_POOL_NAME"
echo "App Client ID: $CLIENT_ID"
echo "App Client Name: $APP_CLIENT_NAME"
echo "Domain: $DOMAIN_NAME.auth.$AWS_REGION.amazoncognito.com"
echo "Admin User: $ADMIN_USER_EMAIL"
echo "Regular User: $USER_EMAIL"
echo "================================================="
echo ""

# Auto-confirm cleanup
echo ""
echo "========================================="
echo "CLEANUP"
echo "========================================="
echo "Starting cleanup process..."

# Step 12: Clean Up Resources
echo "Deleting user pool domain..."
DELETE_DOMAIN_OUTPUT=$(aws cognito-idp delete-user-pool-domain \
  --user-pool-id "$USER_POOL_ID" \
  --domain "$DOMAIN_NAME" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "delete-user-pool-domain"
echo "Domain deleted successfully"

# Wait for domain deletion to complete
echo "Waiting for domain deletion to complete..."
sleep 5

echo "Deleting user pool (this will also delete the app client)..."
DELETE_POOL_OUTPUT=$(aws cognito-idp delete-user-pool \
  --user-pool-id "$USER_POOL_ID" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "delete-user-pool"
echo "User pool deleted successfully"

echo "All resources have been cleaned up"

echo "Script completed at $(date)"
```

```
# Remove trap to prevent cleanup on successful exit
trap - EXIT ERR
```

- API 세부 정보는 AWS CLI 명령 참조의 다음 토픽을 참조하세요.
  - AdminConfirmSignUp
  - AdminCreateUser
  - CreateUserPool
  - CreateUserPoolClient
  - CreateUserPoolDomain
  - DeleteUserPool
  - DeleteUserPoolDomain
  - DescribeUserPool
  - DescribeUserPoolClient
  - InitiateAuth
  - ListUsers
  - SignUp

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 MFA가 필요한 Amazon Cognito 사용자 풀에 사용자 등록

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 사용자 이름, 암호 및 이메일 주소로 사용자를 가입시키고 확인합니다.
- MFA 애플리케이션을 사용자와 연결하여 다중 인증을 설정합니다.
- 암호와 MFA 코드를 사용하여 로그인합니다.

.NET

SDK for .NET

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

```
namespace CognitoBasics;

public class CognitoBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon Cognito.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonCognitoIdentityProvider>()
            .AddTransient<CognitoWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<CognitoBasics>();

        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();
```

```
        var cognitoWrapper = host.Services.GetRequiredService<CognitoWrapper>();

        Console.WriteLine(new string('-', 80));
        UiMethods.DisplayOverview();
        Console.WriteLine(new string('-', 80));

        // clientId - The app client Id value that you get from the AWS CDK
script.
        var clientId = configuration["ClientId"]; // "*** REPLACE WITH CLIENT ID
VALUE FROM CDK SCRIPT";

        // poolId - The pool Id that you get from the AWS CDK script.
        var poolId = configuration["PoolId"]!; // "*** REPLACE WITH POOL ID VALUE
FROM CDK SCRIPT";
        var userName = configuration["UserName"];
        var password = configuration["Password"];
        var email = configuration["Email"];

        // If the username wasn't set in the configuration file,
        // get it from the user now.
        if (userName is null)
        {
            do
            {
                Console.Write("Username: ");
                userName = Console.ReadLine();
            }
            while (string.IsNullOrEmpty(userName));
        }
        Console.WriteLine($"\nUsername: {userName}");

        // If the password wasn't set in the configuration file,
        // get it from the user now.
        if (password is null)
        {
            do
            {
                Console.Write("Password: ");
                password = Console.ReadLine();
            }
            while (string.IsNullOrEmpty(password));
        }

        // If the email address wasn't set in the configuration file,
```

```
        // get it from the user now.
        if (email is null)
        {
            do
            {
                Console.Write("Email: ");
                email = Console.ReadLine();
            } while (string.IsNullOrEmpty(email));
        }

        // Now sign up the user.
        Console.WriteLine($"\nSigning up {userName} with email address:
{email}");
        await cognitoWrapper.SignUpAsync(clientId, userName, password, email);

        // Add the user to the user pool.
        Console.WriteLine($"Adding {userName} to the user pool");
        await cognitoWrapper.GetAdminUserAsync(userName, poolId);

        UiMethods.DisplayTitle("Get confirmation code");
        Console.WriteLine($"Conformation code sent to {userName}.");
        Console.Write("Would you like to send a new code? (Y/N) ");
        var answer = Console.ReadLine();

        if (answer!.ToLower() == "y")
        {
            await cognitoWrapper.ResendConfirmationCodeAsync(clientId, userName);
            Console.WriteLine("Sending a new confirmation code");
        }

        Console.Write("Enter confirmation code (from Email): ");
        var code = Console.ReadLine();

        await cognitoWrapper.ConfirmSignupAsync(clientId, code, userName);

        UiMethods.DisplayTitle("Checking status");
        Console.WriteLine($"Rechecking the status of {userName} in the user
pool");
        await cognitoWrapper.GetAdminUserAsync(userName, poolId);

        Console.WriteLine($"Setting up authenticator for {userName} in the user
pool");
        var setupResponse = await cognitoWrapper.InitiateAuthAsync(clientId,
userName, password);
```

```
        var setupSession = await
 cognitoWrapper.AssociateSoftwareTokenAsync(setupResponse.Session);
        Console.Write("Enter the 6-digit code displayed in Google Authenticator:
 ");
        var setupCode = Console.ReadLine();

        var setupResult = await
 cognitoWrapper.VerifySoftwareTokenAsync(setupSession, setupCode);
        Console.WriteLine($"Setup status: {setupResult}");

        Console.WriteLine($"Now logging in {userName} in the user pool");
        var authSession = await cognitoWrapper.AdminInitiateAuthAsync(clientId,
 poolId, userName, password);

        Console.Write("Enter a new 6-digit code displayed in Google
 Authenticator: ");
        var authCode = Console.ReadLine();

        var authResult = await
 cognitoWrapper.AdminRespondToAuthChallengeAsync(userName, clientId, authCode,
 authSession, poolId);
        Console.WriteLine($"Authenticated and received access token:
 {authResult.AccessToken}");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Cognito scenario is complete.");
        Console.WriteLine(new string('-', 80));
    }
}


using System.Net;

namespace CognitoActions;

/// <summary>
/// Methods to perform Amazon Cognito Identity Provider actions.
/// </summary>
public class CognitoWrapper
{
    private readonly IAmazonCognitoIdentityProvider _cognitoService;

    /// <summary>
```

```
    /// Constructor for the wrapper class containing Amazon Cognito actions.
    /// </summary>
    /// <param name="cognitoService">The Amazon Cognito client object.</param>
    public CognitoWrapper(IAmazonCognitoIdentityProvider cognitoService)
    {
        _cognitoService = cognitoService;
    }

    /// <summary>
    /// List the Amazon Cognito user pools for an account.
    /// </summary>
    /// <returns>A list of UserPoolDescriptionType objects.</returns>
    public async Task<List<UserPoolDescriptionType>> ListUserPoolsAsync()
    {
        var userPools = new List<UserPoolDescriptionType>();

        var userPoolsPaginator = _cognitoService.Paginators.ListUserPools(new
ListUserPoolsRequest());

        await foreach (var response in userPoolsPaginator.Responses)
        {
            userPools.AddRange(response.UserPools);
        }

        return userPools;
    }


    /// <summary>
    /// Get a list of users for the Amazon Cognito user pool.
    /// </summary>
    /// <param name="userPoolId">The user pool ID.</param>
    /// <returns>A list of users.</returns>
    public async Task<List<UserType>> ListUsersAsync(string userPoolId)
    {
        var request = new ListUsersRequest
        {
            UserPoolId = userPoolId
        };

        var users = new List<UserType>();

        var usersPaginator = _cognitoService.Paginators.ListUsers(request);
        await foreach (var response in usersPaginator.Responses)
```

```
        {
            users.AddRange(response.Users);
        }

        return users;
    }


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

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

        var respondToAuthChallengeRequest = new
AdminRespondToAuthChallengeRequest
        {
            ChallengeName = ChallengeNameType.SOFTWARE_TOKEN_MFA,
            ClientId = clientId,
            ChallengeResponses = challengeResponses,
            Session = session,
            UserPoolId = userPoolId,
        };

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

```
        return response.AuthenticationResult;
    }


    /// <summary>
    /// Verify the TOTP and register for MFA.
    /// </summary>
    /// <param name="session">The name of the session.</param>
    /// <param name="code">The MFA code.</param>
    /// <returns>The status of the software token.</returns>
    public async Task<VerifySoftwareTokenResponseType>
VerifySoftwareTokenAsync(string session, string code)
    {
        var tokenRequest = new VerifySoftwareTokenRequest
        {
            UserCode = code,
            Session = session,
        };

        var verifyResponse = await
_cognitoService.VerifySoftwareTokenAsync(tokenRequest);

        return verifyResponse.Status;
    }


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

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

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

```
        return tokenResponse.Session;
    }


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

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

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

    /// <summary>
    /// Initiate authorization.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The name of the user who is authenticating.</
param>
    /// <param name="password">The password for the user who is authenticating.</
param>
    /// <returns>The response from the initiate auth request.</returns>
    public async Task<InitiateAuthResponse> InitiateAuthAsync(string clientId,
 string userName, string password)
    {
```

```
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var authRequest = new InitiateAuthRequest

        {
            ClientId = clientId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.InitiateAuthAsync(authRequest);
        Console.WriteLine($"Result Challenge is : {response.ChallengeName}");

        return response;
    }

    /// <summary>
    /// Confirm that the user has signed up.
    /// </summary>
    /// <param name="clientId">The Id of this application.</param>
    /// <param name="code">The confirmation code sent to the user.</param>
    /// <param name="userName">The username.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ConfirmSignupAsync(string clientId, string code,
string userName)
    {
        var signUpRequest = new ConfirmSignUpRequest
        {
            ClientId = clientId,
            ConfirmationCode = code,
            Username = userName,
        };

        var response = await _cognitoService.ConfirmSignUpAsync(signUpRequest);
        if (response.HttpStatusCode == HttpStatusCode.OK)
        {
            Console.WriteLine($"{userName} was confirmed");
            return true;
        }
        return false;
    }
```

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

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


    /// <summary>
    /// Send a new confirmation code to a user.
    /// </summary>
    /// <param name="clientId">The Id of the client application.</param>
    /// <param name="userName">The username of user who will receive the code.</
param>
    /// <returns>The delivery details.</returns>
    public async Task<CodeDeliveryDetailsType> ResendConfirmationCodeAsync(string
 clientId, string userName)
    {
        var codeRequest = new ResendConfirmationCodeRequest
        {
            ClientId = clientId,
            Username = userName,
        };

        var response = await
 _cognitoService.ResendConfirmationCodeAsync(codeRequest);
```

```
        Console.WriteLine($"Method of delivery is
 {response.CodeDeliveryDetails.DeliveryMedium}");

        return response.CodeDeliveryDetails;
    }


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

        var response = await _cognitoService.AdminGetUserAsync(userRequest);

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


    /// <summary>
    /// Sign up a new user.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The username to use.</param>
    /// <param name="password">The user's password.</param>
    /// <param name="email">The email address of the user.</param>
    /// <returns>A Boolean value indicating whether the user was confirmed.</
returns>
    public async Task<bool> SignUpAsync(string clientId, string userName, string
 password, string email)
    {
        var userAttrs = new AttributeType
        {
```

```
            Name = "email",
            Value = email,
        };

        var userAttrsList = new List<AttributeType>();

        userAttrsList.Add(userAttrs);

        var signUpRequest = new SignUpRequest
        {
            UserAttributes = userAttrsList,
            Username = userName,
            ClientId = clientId,
            Password = password
        };

        var response = await _cognitoService.SignUpAsync(signUpRequest);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 다음 주제를 참조하세요.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

C++

SDK for C++

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

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

//! Scenario that adds a user to an Amazon Cognito user pool.
/*!
  \sa gettingStartedWithUserPools()
  \param clientID: Client ID associated with an Amazon Cognito user pool.
  \param userPoolID: An Amazon Cognito user pool ID.
  \param clientConfig: Aws client configuration.
  \return bool: Successful completion.
 */
bool AwsDoc::Cognito::gettingStartedWithUserPools(const Aws::String &clientID,
                                                  const Aws::String &userPoolID,
                                                  const
 Aws::Client::ClientConfiguration &clientConfig) {
    printAsterisksLine();
    std::cout
            << "Welcome to the Amazon Cognito example scenario."
            << std::endl;
    printAsterisksLine();

    std::cout
            << "This scenario will add a user to an Amazon Cognito user pool."
            << std::endl;
    const Aws::String userName = askQuestion("Enter a new username: ");
    const Aws::String password = askQuestion("Enter a new password: ");
    const Aws::String email = askQuestion("Enter a valid email for the user: ");

    std::cout << "Signing up " << userName << std::endl;
```

```
   Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);
   bool userExists = false;
   do {
       // 1. Add a user with a username, password, and email address.
       Aws::CognitoIdentityProvider::Model::SignUpRequest request;
       request.AddUserAttributes(
               Aws::CognitoIdentityProvider::Model::AttributeType().WithName(
                       "email").WithValue(email));
       request.SetUsername(userName);
       request.SetPassword(password);
       request.SetClientId(clientID);
       Aws::CognitoIdentityProvider::Model::SignUpOutcome outcome =
               client.SignUp(request);

       if (outcome.IsSuccess()) {
           std::cout << "The signup request for " << userName << " was
successful."
                     << std::endl;
       }
       else if (outcome.GetError().GetErrorType() ==

Aws::CognitoIdentityProvider::CognitoIdentityProviderErrors::USERNAME_EXISTS) {
           std::cout
                   << "The username already exists. Please enter a different
username."
                   << std::endl;
           userExists = true;
       }
       else {
           std::cerr << "Error with CognitoIdentityProvider::SignUpRequest. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
           return false;
       }
   } while (userExists);

   printAsterisksLine();
   std::cout << "Retrieving status of " << userName << " in the user pool."
             << std::endl;
   // 2. Confirm that the user was added to the user pool.
   if (!checkAdminUserStatus(userName, userPoolID, client)) {
       return false;
   }
```

```
   std::cout << "A confirmation code was sent to " << email << "." << std::endl;

   bool resend = askYesNoQuestion("Would you like to send a new code? (y/n) ");
   if (resend) {
       // Request a resend of the confirmation code to the email address.
(ResendConfirmationCode)
       Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeRequest
request;
       request.SetUsername(userName);
       request.SetClientId(clientID);

       Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeOutcome
outcome =
               client.ResendConfirmationCode(request);

       if (outcome.IsSuccess()) {
           std::cout
                   << "CognitoIdentityProvider::ResendConfirmationCode was
successful."
                   << std::endl;
       }
       else {
           std::cerr << "Error with
CognitoIdentityProvider::ResendConfirmationCode. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
           return false;
       }
   }

   printAsterisksLine();

   {
       // 4. Send the confirmation code that's received in the email.
(ConfirmSignUp)
       const Aws::String confirmationCode = askQuestion(
               "Enter the confirmation code that was emailed: ");
       Aws::CognitoIdentityProvider::Model::ConfirmSignUpRequest request;
       request.SetClientId(clientID);
       request.SetConfirmationCode(confirmationCode);
       request.SetUsername(userName);

       Aws::CognitoIdentityProvider::Model::ConfirmSignUpOutcome outcome =
```

```
                client.ConfirmSignUp(request);

        if (outcome.IsSuccess()) {
            std::cout << "ConfirmSignup was Successful."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::ConfirmSignUp. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    std::cout << "Rechecking the status of " << userName << " in the user pool."
              << std::endl;
    if (!checkAdminUserStatus(userName, userPoolID, client)) {
        return false;
    }

    printAsterisksLine();

    std::cout << "Initiating authorization using the username and password."
              << std::endl;

    Aws::String session;
    // 5. Initiate authorization with username and password. (AdminInitiateAuth)
    if (!adminInitiateAuthorization(clientID, userPoolID,  userName, password,
 session, client)) {
        return false;
    }

    printAsterisksLine();

    std::cout
            << "Starting setup of time-based one-time password (TOTP) multi-
factor authentication (MFA)."
            << std::endl;

    {
        // 6. Request a setup key for one-time password (TOTP)
        //    multi-factor authentication (MFA). (AssociateSoftwareToken)
        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenRequest
 request;
```

```
        request.SetSession(session);

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

        if (outcome.IsSuccess()) {
            std::cout
                    << "Enter this setup key into an authenticator app, for
 example Google Authenticator."
                    << std::endl;
            std::cout << "Setup key: " << outcome.GetResult().GetSecretCode()
                      << std::endl;
#ifdef USING_QR
            printAsterisksLine();
            std::cout << "\nOr scan the QR code in the file '" << QR_CODE_PATH <<
 "."
                      << std::endl;

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

    printAsterisksLine();

    {
        Aws::String userCode = askQuestion(
                "Enter the 6 digit code displayed in the authenticator app: ");

        //  7. Send the MFA code copied from an authenticator app.
 (VerifySoftwareToken)
        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenRequest request;
        request.SetUserCode(userCode);
```

```
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenOutcome outcome =
                client.VerifySoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout << "Verification of the code was successful."
                      << std::endl;
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::VerifySoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();
    std::cout << "You have completed the MFA authentication setup." << std::endl;
    std::cout << "Now, sign in." << std::endl;

    // 8. Initiate authorization again with username and password.
(AdminInitiateAuth)
    if (!adminInitiateAuthorization(clientID, userPoolID, userName, password,
session, client)) {
        return false;
    }

    Aws::String accessToken;
    {
        Aws::String mfaCode = askQuestion(
                "Re-enter the 6 digit code displayed in the authenticator app:
");

        // 9. Send a new MFA code copied from an authenticator app.
(AdminRespondToAuthChallenge)
        Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeRequest
request;
        request.AddChallengeResponses("USERNAME", userName);
        request.AddChallengeResponses("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
        request.SetChallengeName(
```

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

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

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

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

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

       std::cout << "You have successfully added a user to Amazon Cognito."
                 << std::endl;
   }

   if (askYesNoQuestion("Would you like to delete the user that you just added?
(y/n) ")) {
       // 10. Delete the user that you just added. (DeleteUser)
       Aws::CognitoIdentityProvider::Model::DeleteUserRequest request;
       request.SetAccessToken(accessToken);

       Aws::CognitoIdentityProvider::Model::DeleteUserOutcome outcome =
               client.DeleteUser(request);

       if (outcome.IsSuccess()) {
           std::cout << "The user " << userName << " was deleted."
                     << std::endl;
       }
```

```
        else {
            std::cerr << "Error with CognitoIdentityProvider::DeleteUser. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    return true;
}

//! Routine which checks the user status in an Amazon Cognito user pool.
/*!
 \sa checkAdminUserStatus()
 \param userName: A username.
 \param userPoolID: An Amazon Cognito user pool ID.
 \return bool: Successful completion.
 */
bool AwsDoc::Cognito::checkAdminUserStatus(const Aws::String &userName,
                                           const Aws::String &userPoolID,
                                           const
 Aws::CognitoIdentityProvider::CognitoIdentityProviderClient &client) {
    Aws::CognitoIdentityProvider::Model::AdminGetUserRequest request;
    request.SetUsername(userName);
    request.SetUserPoolId(userPoolID);

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

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

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

    return outcome.IsSuccess();
}
```

```
//! Routine which starts authorization of an Amazon Cognito user.
//! This routine requires administrator credentials.
/*!
 \sa adminInitiateAuthorization()
 \param clientID: Client ID of tracked device.
 \param userPoolID: An Amazon Cognito user pool ID.
 \param userName: A username.
 \param password: A password.
 \param sessionResult: String to receive a session token.
 \return bool: Successful completion.
 */
bool AwsDoc::Cognito::adminInitiateAuthorization(const Aws::String &clientID,
                                                 const Aws::String &userPoolID,
                                                 const Aws::String &userName,
                                                 const Aws::String &password,
                                                 Aws::String &sessionResult,
                                                 const
 Aws::CognitoIdentityProvider::CognitoIdentityProviderClient &client) {
    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthRequest request;
    request.SetClientId(clientID);
    request.SetUserPoolId(userPoolID);
    request.AddAuthParameters("USERNAME", userName);
    request.AddAuthParameters("PASSWORD", password);
    request.SetAuthFlow(

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


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

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

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 다음 주제를 참조하세요.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

Java

SDK for Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthRequest;
```

```
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChalleng
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChalleng
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenRequ
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenResp
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AttributeType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AuthFlowType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ChallengeNameType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ConfirmSignUpRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeRequ
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeResp
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.SignUpRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenRequest
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenRespons
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * TIP: To set up the required user pool, run the AWS Cloud Development Kit (AWS
 * CDK) script provided in this GitHub repo at
 * resources/cdk/cognito_scenario_user_pool_with_mfa.
 *
 * This code example performs the following operations:
 *
 * 1. Invokes the signUp method to sign up a user.
 * 2. Invokes the adminGetUser method to get the user's confirmation status.
 * 3. Invokes the ResendConfirmationCode method if the user requested another
 * code.
 * 4. Invokes the confirmSignUp method.
 * 5. Invokes the AdminInitiateAuth to sign in. This results in being prompted
 * to set up TOTP (time-based one-time password). (The response is
 * “ChallengeName”: “MFA_SETUP”).
 * 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private
 * key. This can be used with Google Authenticator.
 * 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for
 * MFA.
 * 8. Invokes the AdminInitiateAuth to sign in again. This results in being
 * prompted to submit a TOTP (Response: “ChallengeName”: “SOFTWARE_TOKEN_MFA”).
 * 9. Invokes the AdminRespondToAuthChallenge to get back a token.
 */

public class CognitoMVP {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws NoSuchAlgorithmException,
 InvalidKeyException {
        final String usage = """

                Usage:
                    <clientId> <poolId>

                Where:
                    clientId - The app client Id value that you can get from the
 AWS CDK script.
                    poolId - The pool Id that you can get from the AWS CDK
 script.\s
                """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clientId = args[0];
        String poolId = args[1];
        CognitoIdentityProviderClient identityProviderClient =
CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Cognito example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter your user name");
        Scanner in = new Scanner(System.in);
        String userName = in.nextLine();

        System.out.println("*** Enter your password");
        String password = in.nextLine();

        System.out.println("*** Enter your email");
        String email = in.nextLine();

        System.out.println("1. Signing up " + userName);
        signUp(identityProviderClient, clientId, userName, password, email);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Getting " + userName + " in the user pool");
        getAdminUser(identityProviderClient, userName, poolId);

        System.out
                .println("*** Conformation code sent to " + userName + ". Would
you like to send a new code? (Yes/No)");
        System.out.println(DASHES);

        System.out.println(DASHES);
        String ans = in.nextLine();

        if (ans.compareTo("Yes") == 0) {
```

```
            resendConfirmationCode(identityProviderClient, clientId, userName);
            System.out.println("3. Sending a new confirmation code");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Enter confirmation code that was emailed");
        String code = in.nextLine();
        confirmSignUp(identityProviderClient, clientId, code, userName);
        System.out.println("Rechecking the status of " + userName + " in the user
pool");
        getAdminUser(identityProviderClient, userName, poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Invokes the initiateAuth to sign in");
        AdminInitiateAuthResponse authResponse =
initiateAuth(identityProviderClient, clientId, userName, password,
                poolId);
        String mySession = authResponse.session();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Invokes the AssociateSoftwareToken method to
generate a TOTP key");
        String newSession = getSecretForAppMFA(identityProviderClient,
mySession);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter the 6-digit code displayed in Google
Authenticator");
        String myCode = in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Verify the TOTP and register for MFA");
        verifyTOTP(identityProviderClient, newSession, myCode);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Re-enter a 6-digit code displayed in Google
Authenticator");
        String mfaCode = in.nextLine();
```

```
        AdminInitiateAuthResponse authResponse1 =
initiateAuth(identityProviderClient, clientId, userName, password,
                poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9.  Invokes the AdminRespondToAuthChallenge");
        String session2 = authResponse1.session();
        adminRespondToAuthChallenge(identityProviderClient, userName, clientId,
mfaCode, session2);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("All Amazon Cognito operations were successfully
performed");
        System.out.println(DASHES);
    }

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

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

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

        AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
                .adminRespondToAuthChallenge(respondToAuthChallengeRequest);

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

```
   // Verify the TOTP and register for MFA.
   public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
       try {
           VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
                   .userCode(code)
                   .session(session)
                   .build();

           VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
           System.out.println("The status of the token is " +
verifyResponse.statusAsString());

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

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

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

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

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

        return null;
    }

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

        AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
                .associateSoftwareToken(softwareTokenRequest);
        String secretCode = tokenResponse.secretCode();
        System.out.println("Enter this token into Google Authenticator");
        System.out.println(secretCode);
        return tokenResponse.session();
    }

    public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
            String userName) {
        try {
            ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
                    .clientId(clientId)
                    .confirmationCode(code)
                    .username(userName)
                    .build();

            identityProviderClient.confirmSignUp(signUpRequest);
            System.out.println(userName + " was confirmed");

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

    public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
```

```
            String userName) {
        try {
            ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
                    .clientId(clientId)
                    .username(userName)
                    .build();

            ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
            System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

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

    public static void signUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String userName,
            String password, String email) {
        AttributeType userAttrs = AttributeType.builder()
                .name("email")
                .value(email)
                .build();

        List<AttributeType> userAttrsList = new ArrayList<>();
        userAttrsList.add(userAttrs);
        try {
            SignUpRequest signUpRequest = SignUpRequest.builder()
                    .userAttributes(userAttrsList)
                    .username(userName)
                    .clientId(clientId)
                    .password(password)
                    .build();

            identityProviderClient.signUp(signUpRequest);
            System.out.println("User has been signed up ");

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

```
    }

    public static void getAdminUser(CognitoIdentityProviderClient
 identityProviderClient, String userName,
            String poolId) {
        try {
            AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
                    .username(userName)
                    .userPoolId(poolId)
                    .build();

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

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

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

JavaScript

SDK for JavaScript (v3)

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

최상의 경험을 위해 GitHub 리포지토리를 복제하고 이 예제를 실행하세요. 다음 코드는 전체 예제 애플리케이션의 샘플을 나타냅니다.

```
import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { signUp } from "../../../actions/sign-up.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getSecondValuesFromEntries } from "@aws-doc-sdk-examples/lib/utils/util-
csv.js";

const validateClient = (clientId) => {
  if (!clientId) {
    throw new Error(
      `App client id is missing. Did you run 'create-user-pool'?`,
    );
  }
};

const validateUser = (username, password, email) => {
  if (!(username && password && email)) {
    throw new Error(
      `Username, password, and email must be provided as arguments to the 'sign-
up' command.`,
    );
  }
};

const signUpHandler = async (commands) => {
  const [_, username, password, email] = commands;

  try {
    validateUser(username, password, email);
    /**
```

```
     * @type {string[]}
     */
    const values = getSecondValuesFromEntries(FILE_USER_POOLS);
    const clientId = values[0];
    validateClient(clientId);
    logger.log("Signing up.");
    await signUp({ clientId, username, password, email });
    logger.log(`Signed up. A confirmation email has been sent to: ${email}.`);
    logger.log(
      `Run 'confirm-sign-up ${username} <code>' to confirm your account.`,
    );
  } catch (err) {
    logger.error(err);
  }
};

export { signUpHandler };

const signUp = ({ clientId, username, password, email }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new SignUpCommand({
    ClientId: clientId,
    Username: username,
    Password: password,
    UserAttributes: [{ Name: "email", Value: email }],
  });

  return client.send(command);
};

import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { confirmSignUp } from "../../../actions/confirm-sign-up.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getSecondValuesFromEntries } from "@aws-doc-sdk-examples/lib/utils/util-
csv.js";

const validateClient = (clientId) => {
  if (!clientId) {
    throw new Error(
      `App client id is missing. Did you run 'create-user-pool'?`,
    );
  }
};
```

```
const validateUser = (username) => {
  if (!username) {
    throw new Error(
      `Username name is missing. It must be provided as an argument to the
 'confirm-sign-up' command.`,
    );
  }
};

const validateCode = (code) => {
  if (!code) {
    throw new Error(
      `Verification code is missing. It must be provided as an argument to the
 'confirm-sign-up' command.`,
    );
  }
};

const confirmSignUpHandler = async (commands) => {
  const [_, username, code] = commands;

  try {
    validateUser(username);
    validateCode(code);
    /**
     * @type {string[]}
     */
    const values = getSecondValuesFromEntries(FILE_USER_POOLS);
    const clientId = values[0];
    validateClient(clientId);
    logger.log("Confirming user.");
    await confirmSignUp({ clientId, username, code });
    logger.log(
      `User confirmed. Run 'admin-initiate-auth ${username} <password>' to sign
 in.`,
    );
  } catch (err) {
    logger.error(err);
  }
};

export { confirmSignUpHandler };
```

```
const confirmSignUp = ({ clientId, username, code }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ConfirmSignUpCommand({
    ClientId: clientId,
    Username: username,
    ConfirmationCode: code,
  });

  return client.send(command);
};

import qrcode from "qrcode-terminal";
import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { adminInitiateAuth } from "../../../actions/admin-initiate-auth.js";
import { associateSoftwareToken } from "../../../actions/associate-software-
token.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getFirstEntry } from "@aws-doc-sdk-examples/lib/utils/util-csv.js";

const handleMfaSetup = async (session, username) => {
  const { SecretCode, Session } = await associateSoftwareToken(session);

  // Store the Session for use with 'VerifySoftwareToken'.
  process.env.SESSION = Session;

  console.log(
    "Scan this code in your preferred authenticator app, then run 'verify-
software-token' to finish the setup.",
  );
  qrcode.generate(
    `otpauth://totp/${username}?secret=${SecretCode}`,
    { small: true },
    console.log,
  );
};

const handleSoftwareTokenMfa = (session) => {
  // Store the Session for use with 'AdminRespondToAuthChallenge'.
  process.env.SESSION = session;
};

const validateClient = (id) => {
  if (!id) {
```

```
    throw new Error(
      `User pool client id is missing. Did you run 'create-user-pool'?`,
    );
  }
};

const validateId = (id) => {
  if (!id) {
    throw new Error(`User pool id is missing. Did you run 'create-user-pool'?`);
  }
};

const validateUser = (username, password) => {
  if (!(username && password)) {
    throw new Error(
      `Username and password must be provided as arguments to the 'admin-
initiate-auth' command.`,
    );
  }
};

const adminInitiateAuthHandler = async (commands) => {
  const [_, username, password] = commands;

  try {
    validateUser(username, password);

    const [userPoolId, clientId] = getFirstEntry(FILE_USER_POOLS);
    validateId(userPoolId);
    validateClient(clientId);

    logger.log("Signing in.");
    const { ChallengeName, Session } = await adminInitiateAuth({
      clientId,
      userPoolId,
      username,
      password,
    });

    if (ChallengeName === "MFA_SETUP") {
      logger.log("MFA setup is required.");
      return handleMfaSetup(Session, username);
    }
```

```
    if (ChallengeName === "SOFTWARE_TOKEN_MFA") {
      handleSoftwareTokenMfa(Session);
      logger.log(`Run 'admin-respond-to-auth-challenge ${username} <totp>'`);
    }
  } catch (err) {
    logger.error(err);
  }
};

export { adminInitiateAuthHandler };

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

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

  return client.send(command);
};

import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { adminRespondToAuthChallenge } from "../../../actions/admin-respond-to-
auth-challenge.js";
import { getFirstEntry } from "@aws-doc-sdk-examples/lib/utils/util-csv.js";
import { FILE_USER_POOLS } from "./constants.js";

const verifyUsername = (username) => {
  if (!username) {
    throw new Error(
      `Username is missing. It must be provided as an argument to the 'admin-
respond-to-auth-challenge' command.`,
    );
  }
};

const verifyTotp = (totp) => {
  if (!totp) {
    throw new Error(
      `Time-based one-time password (TOTP) is missing. It must be provided as an
 argument to the 'admin-respond-to-auth-challenge' command.`,
```

```
    );
  }
};

const storeAccessToken = (token) => {
  process.env.AccessToken = token;
};

const adminRespondToAuthChallengeHandler = async (commands) => {
  const [_, username, totp] = commands;

  try {
    verifyUsername(username);
    verifyTotp(totp);

    const [userPoolId, clientId] = getFirstEntry(FILE_USER_POOLS);
    const session = process.env.SESSION;

    const { AuthenticationResult } = await adminRespondToAuthChallenge({
      clientId,
      userPoolId,
      username,
      totp,
      session,
    });

    storeAccessToken(AuthenticationResult.AccessToken);

    logger.log("Successfully authenticated.");
  } catch (err) {
    logger.error(err);
  }
};

export { adminRespondToAuthChallengeHandler };

const respondToAuthChallenge = ({
  clientId,
  username,
  session,
  userPoolId,
  code,
}) => {
  const client = new CognitoIdentityProviderClient({});
```

```
  const command = new RespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: code,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
};

import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { verifySoftwareToken } from "../../../actions/verify-software-token.js";

const validateTotp = (totp) => {
  if (!totp) {
    throw new Error(
      `Time-based one-time password (TOTP) must be provided to the 'validate-
software-token' command.`,
    );
  }
};
const verifySoftwareTokenHandler = async (commands) => {
  const [_, totp] = commands;

  try {
    validateTotp(totp);

    logger.log("Verifying TOTP.");
    await verifySoftwareToken(totp);
    logger.log("TOTP Verified. Run 'admin-initiate-auth' again to sign-in.");
  } catch (err) {
    logger.error(err);
  }
};

export { verifySoftwareTokenHandler };

const verifySoftwareToken = (totp) => {
  const client = new CognitoIdentityProviderClient({});
```

```
  // The 'Session' is provided in the response to 'AssociateSoftwareToken'.
  const session = process.env.SESSION;

  if (!session) {
    throw new Error(
      "Missing a valid Session. Did you run 'admin-initiate-auth'?",
    );
  }

  const command = new VerifySoftwareTokenCommand({
    Session: session,
    UserCode: totp,
  });

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

- API 세부 정보는 AWS SDK for JavaScript API 참조의 다음 주제를 참조하세요.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

Kotlin

SDK for Kotlin

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

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 TIP: To set up the required user pool, run the AWS Cloud Development
 Kit (AWS CDK) script provided in this GitHub repo at resources/cdk/
cognito_scenario_user_pool_with_mfa.

 This code example performs the following operations:

 1. Invokes the signUp method to sign up a user.
 2. Invokes the adminGetUser method to get the user's confirmation status.
 3. Invokes the ResendConfirmationCode method if the user requested another code.
 4. Invokes the confirmSignUp method.
 5. Invokes the initiateAuth to sign in. This results in being prompted to
 set up TOTP (time-based one-time password). (The response is “ChallengeName”:
 “MFA_SETUP”).
 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private key.
 This can be used with Google Authenticator.
 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for
 MFA.
 8. Invokes the AdminInitiateAuth to sign in again. This results in being
 prompted to submit a TOTP (Response: “ChallengeName”: “SOFTWARE_TOKEN_MFA”).
 9. Invokes the AdminRespondToAuthChallenge to get back a token.
 */

suspend fun main(args: Array<String>) {
    val usage = """
```

```
        Usage:
            <clientId> <poolId>
        Where:
            clientId - The app client Id value that you can get from the AWS CDK
script.
            poolId - The pool Id that you can get from the AWS CDK script.
    """

    if (args.size != 2) {
        println(usage)
        exitProcess(1)
    }

    val clientId = args[0]
    val poolId = args[1]

    // Use the console to get data from the user.
    println("*** Enter your use name")
    val inOb = Scanner(System.`in`)
    val userName = inOb.nextLine()
    println(userName)

    println("*** Enter your password")
    val password: String = inOb.nextLine()

    println("*** Enter your email")
    val email = inOb.nextLine()

    println("*** Signing up $userName")
    signUp(clientId, userName, password, email)

    println("*** Getting $userName in the user pool")
    getAdminUser(userName, poolId)

    println("*** Conformation code sent to $userName. Would you like to send a
new code? (Yes/No)")
    val ans = inOb.nextLine()

    if (ans.compareTo("Yes") == 0) {
        println("*** Sending a new confirmation code")
        resendConfirmationCode(clientId, userName)
    }
    println("*** Enter the confirmation code that was emailed")
    val code = inOb.nextLine()
```

```
    confirmSignUp(clientId, code, userName)

    println("*** Rechecking the status of $userName in the user pool")
    getAdminUser(userName, poolId)

    val authResponse = checkAuthMethod(clientId, userName, password, poolId)
    val mySession = authResponse.session
    val newSession = getSecretForAppMFA(mySession)
    println("*** Enter the 6-digit code displayed in Google Authenticator")
    val myCode = inOb.nextLine()

    // Verify the TOTP and register for MFA.
    verifyTOTP(newSession, myCode)
    println("*** Re-enter a 6-digit code displayed in Google Authenticator")
    val mfaCode: String = inOb.nextLine()
    val authResponse1 = checkAuthMethod(clientId, userName, password, poolId)
    val session2 = authResponse1.session
    adminRespondToAuthChallenge(userName, clientId, mfaCode, session2)
}

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

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

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

```
}

suspend fun resendConfirmationCode(
    clientIdVal: String?,
    userNameVal: String?,
) {
    val codeRequest =
        ResendConfirmationCodeRequest {
            clientId = clientIdVal
            username = userNameVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
    }
}

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

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

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

```
        println("respondToAuthChallengeResult.getAuthenticationResult()
 ${respondToAuthChallengeResult.authenticationResult}")
    }
}

// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(
    sessionVal: String?,
    codeVal: String?,
) {
    val tokenRequest =
        VerifySoftwareTokenRequest {
            userCode = codeVal
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}

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

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

suspend fun confirmSignUp(
    clientIdVal: String?,
    codeVal: String?,
```

```
    userNameVal: String?,
) {
    val signUpRequest =
        ConfirmSignUpRequest {
            clientId = clientIdVal
            confirmationCode = codeVal
            username = userNameVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
    }
}

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

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

suspend fun signUp(
    clientIdVal: String?,
    userNameVal: String?,
    passwordVal: String?,
    emailVal: String?,
) {
    val userAttrs =
        AttributeType {
            name = "email"
            value = emailVal
        }
```

```
    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)
    val signUpRequest =
        SignUpRequest {
            userAttributes = userAttrsList
            username = userNameVal
            clientId = clientIdVal
            password = passwordVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 다음 주제를 참조하세요.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

Python

SDK for Python (Boto3)

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

시나리오에 사용된 Amazon Cognito 함수를 래핑하는 클래스를 만듭니다.

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

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


    def _secret_hash(self, user_name):
        """
        Calculates a secret hash from a user name and a client secret.

        :param user_name: The user name to use when calculating the hash.
        :return: The secret hash.
        """
        key = self.client_secret.encode()
        msg = bytes(user_name + self.client_id, "utf-8")
        secret_hash = base64.b64encode(
            hmac.new(key, msg, digestmod=hashlib.sha256).digest()
        ).decode()
```

```
        logger.info("Made secret hash for %s: %s.", user_name, secret_hash)
        return secret_hash

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

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

        :param user_name: The user name that identifies the new user.
        :param password: The password for the new user.
        :param user_email: The email address for the new user.
        :return: True when the user is already confirmed with Amazon Cognito.
                 Otherwise, false.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "Password": password,
                "UserAttributes": [{"Name": "email", "Value": user_email}],
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.sign_up(**kwargs)
            confirmed = response["UserConfirmed"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "UsernameExistsException":
                response = self.cognito_idp_client.admin_get_user(
                    UserPoolId=self.user_pool_id, Username=user_name
                )
                logger.warning(
                    "User %s exists and is %s.", user_name,
response["UserStatus"]
                )
                confirmed = response["UserStatus"] == "CONFIRMED"
            else:
                logger.error(
```

```
                    "Couldn't sign up %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return confirmed


    def resend_confirmation(self, user_name):
        """
        Prompts Amazon Cognito to resend an email with a new confirmation code.

        :param user_name: The name of the user who will receive the email.
        :return: Delivery information about where the email is sent.
        """
        try:
            kwargs = {"ClientId": self.client_id, "Username": user_name}
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.resend_confirmation_code(**kwargs)
            delivery = response["CodeDeliveryDetails"]
        except ClientError as err:
            logger.error(
                "Couldn't resend confirmation to %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return delivery


    def confirm_user_sign_up(self, user_name, confirmation_code):
        """
        Confirms a previously created user. A user must be confirmed before they
        can sign in to Amazon Cognito.

        :param user_name: The name of the user to confirm.
        :param confirmation_code: The confirmation code sent to the user's
registered
                                  email address.
        :return: True when the confirmation succeeds.
```

```
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "ConfirmationCode": confirmation_code,
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            self.cognito_idp_client.confirm_sign_up(**kwargs)
        except ClientError as err:
            logger.error(
                "Couldn't confirm sign up for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return True


    def list_users(self):
        """
        Returns a list of the users in the current user pool.

        :return: The list of users.
        """
        try:
            response =
self.cognito_idp_client.list_users(UserPoolId=self.user_pool_id)
            users = response["Users"]
        except ClientError as err:
            logger.error(
                "Couldn't list users for %s. Here's why: %s: %s",
                self.user_pool_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return users
```

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

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

        :param user_name: The name of the user to sign in.
        :param password: The user's password.
        :return: The result of the sign-in attempt. When sign-in is successful,
 this
                 returns an access token that can be used to get AWS credentials.
 Otherwise,
                 Amazon Cognito returns a challenge to set up an MFA application,
                 or a challenge to enter an MFA code from a registered MFA
 application.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "AuthFlow": "ADMIN_USER_PASSWORD_AUTH",
                "AuthParameters": {"USERNAME": user_name, "PASSWORD": password},
            }
            if self.client_secret is not None:
                kwargs["AuthParameters"]["SECRET_HASH"] =
 self._secret_hash(user_name)
            response = self.cognito_idp_client.admin_initiate_auth(**kwargs)
            challenge_name = response.get("ChallengeName", None)
            if challenge_name == "MFA_SETUP":
                if (
                    "SOFTWARE_TOKEN_MFA"
                    in response["ChallengeParameters"]["MFAS_CAN_SETUP"]
                ):
                    response.update(self.get_mfa_secret(response["Session"]))
                else:
                    raise RuntimeError(
                        "The user pool requires MFA setup, but the user pool is
 not "
```

```
                        "configured for TOTP MFA. This example requires TOTP
MFA."
                    )
        except ClientError as err:
            logger.error(
                "Couldn't start sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response


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

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


    def verify_mfa(self, session, user_code):
        """
        Verify a new MFA application that is associated with a user.
```

```
        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :param user_code: A code generated by the associated MFA application.
        :return: Status that indicates whether the MFA application is verified.
        """
        try:
            response = self.cognito_idp_client.verify_software_token(
                Session=session, UserCode=user_code
            )
        except ClientError as err:
            logger.error(
                "Couldn't verify MFA. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response


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

        :param user_name: The name of the user who is signing in.
        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :param mfa_code: A code generated by the associated MFA application.
        :return: The result of the authentication. When successful, this contains
an
                 access token for the user.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
```

```
                "ChallengeName": "SOFTWARE_TOKEN_MFA",
                "Session": session,
                "ChallengeResponses": {
                    "USERNAME": user_name,
                    "SOFTWARE_TOKEN_MFA_CODE": mfa_code,
                },
            }
            if self.client_secret is not None:
                kwargs["ChallengeResponses"]["SECRET_HASH"] = self._secret_hash(
                    user_name
                )
            response =
self.cognito_idp_client.admin_respond_to_auth_challenge(**kwargs)
            auth_result = response["AuthenticationResult"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ExpiredCodeException":
                logger.warning(
                    "Your MFA code has expired or has been used already. You
might have "
                    "to wait a few seconds until your app shows you a new code."
                )
            else:
                logger.error(
                    "Couldn't respond to mfa challenge for %s. Here's why: %s:
%s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return auth_result


    def confirm_mfa_device(
        self,
        user_name,
        device_key,
        device_group_key,
        device_password,
        access_token,
        aws_srp,
    ):
        """
```

```
        Confirms an MFA device to be tracked by Amazon Cognito. When a device is
        tracked, its key and password can be used to sign in without requiring a
new
        MFA code from the MFA application.

        :param user_name: The user that is associated with the device.
        :param device_key: The key of the device, returned by Amazon Cognito.
        :param device_group_key: The group key of the device, returned by Amazon
Cognito.
        :param device_password: The password that is associated with the device.
        :param access_token: The user's access token.
        :param aws_srp: A class that helps with Secure Remote Password (SRP)
                        calculations. The scenario associated with this example
uses
                        the warrant package.
        :return: True when the user must confirm the device. Otherwise, False.
When
                 False, the device is automatically confirmed and tracked.
        """
        srp_helper = aws_srp.AWSSRP(
            username=user_name,
            password=device_password,
            pool_id="_",
            client_id=self.client_id,
            client_secret=None,
            client=self.cognito_idp_client,
        )
        device_and_pw = f"{device_group_key}{device_key}:{device_password}"
        device_and_pw_hash = aws_srp.hash_sha256(device_and_pw.encode("utf-8"))
        salt = aws_srp.pad_hex(aws_srp.get_random(16))
        x_value = aws_srp.hex_to_long(aws_srp.hex_hash(salt +
device_and_pw_hash))
        verifier = aws_srp.pad_hex(pow(srp_helper.val_g, x_value,
srp_helper.big_n))
        device_secret_verifier_config = {
            "PasswordVerifier": base64.standard_b64encode(
                bytearray.fromhex(verifier)
            ).decode("utf-8"),
            "Salt":
base64.standard_b64encode(bytearray.fromhex(salt)).decode("utf-8"),
        }
        try:
            response = self.cognito_idp_client.confirm_device(
                AccessToken=access_token,
```

```
                DeviceKey=device_key,
                DeviceSecretVerifierConfig=device_secret_verifier_config,
            )
            user_confirm = response["UserConfirmationNecessary"]
        except ClientError as err:
            logger.error(
                "Couldn't confirm mfa device %s. Here's why: %s: %s",
                device_key,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return user_confirm


    def sign_in_with_tracked_device(
        self,
        user_name,
        password,
        device_key,
        device_group_key,
        device_password,
        aws_srp,
    ):
        """
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA
 code.

        Signing in with a tracked device requires that the client respond to the
 SRP
        protocol. The scenario associated with this example uses the warrant
 package
        to help with SRP calculations.

        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
```

```
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains
an
                 access token for the user.
        """
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name,
                password=device_password,
                pool_id="_",
                client_id=self.client_id,
                client_secret=None,
                client=self.cognito_idp_client,
            )

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id,
                AuthFlow="USER_PASSWORD_AUTH",
                AuthParameters={
                    "USERNAME": user_name,
                    "PASSWORD": password,
                    "DEVICE_KEY": device_key,
                },
            )
            if response_init["ChallengeName"] != "DEVICE_SRP_AUTH":
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
{response_init['ChallengeName']}."
                )

            auth_params = srp_helper.get_auth_params()
            auth_params["DEVICE_KEY"] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_SRP_AUTH",
                ChallengeResponses=auth_params,
            )
            if response_auth["ChallengeName"] != "DEVICE_PASSWORD_VERIFIER":
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}."
                )
```

```
            challenge_params = response_auth["ChallengeParameters"]
            challenge_params["USER_ID_FOR_SRP"] = device_group_key + device_key
            cr = srp_helper.process_challenge(challenge_params, {"USERNAME":
 user_name})
            cr["USERNAME"] = user_name
            cr["DEVICE_KEY"] = device_key
            response_verifier =
 self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_PASSWORD_VERIFIER",
                ChallengeResponses=cr,
            )
            auth_tokens = response_verifier["AuthenticationResult"]
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return auth_tokens
```

시나리오를 실행하는 클래스를 생성합니다. 이 예시에서는 Amazon Cognito에서 추적할 MFA 디바이스를 등록하고 추적된 디바이스의 암호 및 정보를 사용하여 로그인하는 방법도 보여줍니다. 이렇게 하면 새 MFA 코드를 입력할 필요가 없습니다.

```
def run_scenario(cognito_idp_client, user_pool_id, client_id):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon Cognito user signup with MFA demo.")
    print("-" * 88)

    cog_wrapper = CognitoIdentityProviderWrapper(
        cognito_idp_client, user_pool_id, client_id
    )
```

```
    user_name = q.ask("Let's sign up a new user. Enter a user name: ",
q.non_empty)
    password = q.ask("Enter a password for the user: ", q.non_empty)
    email = q.ask("Enter a valid email address that you own: ", q.non_empty)
    confirmed = cog_wrapper.sign_up_user(user_name, password, email)
    while not confirmed:
        print(
            f"User {user_name} requires confirmation. Check {email} for "
            f"a verification code."
        )
        confirmation_code = q.ask("Enter the confirmation code from the email: ")
        if not confirmation_code:
            if q.ask("Do you need another confirmation code (y/n)? ",
q.is_yesno):
                delivery = cog_wrapper.resend_confirmation(user_name)
                print(
                    f"Confirmation code sent by {delivery['DeliveryMedium']} "
                    f"to {delivery['Destination']}."
                )
        else:
            confirmed = cog_wrapper.confirm_user_sign_up(user_name,
confirmation_code)
    print(f"User {user_name} is confirmed and ready to use.")
    print("-" * 88)

    print("Let's get a list of users in the user pool.")
    q.ask("Press Enter when you're ready.")
    users = cog_wrapper.list_users()
    if users:
        print(f"Found {len(users)} users:")
        pp(users)
    else:
        print("No users found.")
    print("-" * 88)

    print("Let's sign in and get an access token.")
    auth_tokens = None
    challenge = "ADMIN_USER_PASSWORD_AUTH"
    response = {}
    while challenge is not None:
        if challenge == "ADMIN_USER_PASSWORD_AUTH":
            response = cog_wrapper.start_sign_in(user_name, password)
            # start_sign_in may return an AuthenticationResult with no challenge
            # (for example, when the user pool does not require MFA). Read the
```

```
            # challenge defensively and stop if sign-in already succeeded.
            challenge = response.get("ChallengeName")
            if challenge is None:
                auth_tokens = response.get("AuthenticationResult")
                if auth_tokens is not None:
                    print(f"You're signed in as {user_name}.")
                    print("Here's your access token:")
                    pp(auth_tokens["AccessToken"])
                break
        elif response["ChallengeName"] == "MFA_SETUP":
            print("First, we need to set up an MFA application.")
            qr_img = qrcode.make(
                f"otpauth://totp/{user_name}?secret={response['SecretCode']}"
            )
            qr_img.save("qr.png")
            q.ask(
                "Press Enter to see a QR code on your screen. Scan it into an MFA
"
                "application, such as Google Authenticator."
            )
            webbrowser.open("qr.png")
            mfa_code = q.ask(
                "Enter the verification code from your MFA application: ",
q.non_empty
            )
            response = cog_wrapper.verify_mfa(response["Session"], mfa_code)
            print(f"MFA device setup {response['Status']}")
            print("Now that an MFA application is set up, let's sign in again.")
            print(
                "You might have to wait a few seconds for a new MFA code to
appear in "
                "your MFA application."
            )
            challenge = "ADMIN_USER_PASSWORD_AUTH"
        elif response["ChallengeName"] == "SOFTWARE_TOKEN_MFA":
            auth_tokens = None
            while auth_tokens is None:
                mfa_code = q.ask(
                    "Enter a verification code from your MFA application: ",
q.non_empty
                )
                auth_tokens = cog_wrapper.respond_to_mfa_challenge(
                    user_name, response["Session"], mfa_code
                )
```

```
            print(f"You're signed in as {user_name}.")
            print("Here's your access token:")
            pp(auth_tokens["AccessToken"])
            if "NewDeviceMetadata" in auth_tokens:
                print("And your device information:")
                pp(auth_tokens["NewDeviceMetadata"])
            challenge = None
        else:
            raise Exception(f"Got unexpected challenge
 {response['ChallengeName']}")
    print("-" * 88)

    # If sign-in returned neither a challenge nor an AuthenticationResult, there
    # are no tokens to work with; nothing more to do.
    if auth_tokens is None:
        return user_name

    # respond_to_mfa_challenge returns NewDeviceMetadata only when the user pool
    # has device tracking configured. When it's absent, skip the device-
confirmation
    # and tracked-device sign-in steps instead of failing on a missing key.
    device_metadata = auth_tokens.get("NewDeviceMetadata")
    if device_metadata is None:
        print(
            "This user pool doesn't track devices, so there's no device to
 confirm. "
            "To try the tracked-device sign-in flow, enable device tracking on
 the "
            "user pool and run this example again."
        )
        print("-" * 88)
    else:
        device_group_key = device_metadata["DeviceGroupKey"]
        device_key = device_metadata["DeviceKey"]
        device_password =
 base64.standard_b64encode(os.urandom(40)).decode("utf-8")

        print(
            "Let's confirm your MFA device so you don't have to re-enter MFA
 tokens for it."
        )
        q.ask("Press Enter when you're ready.")
        cog_wrapper.confirm_mfa_device(
            user_name,
```

```
            device_key,
            device_group_key,
            device_password,
            auth_tokens["AccessToken"],
            aws_srp,
        )
        print(f"Your device {device_key} is confirmed.")
        print("-" * 88)

        print(
            f"Now let's sign in as {user_name} from your confirmed device
 {device_key}.\n"
            f"Because this device is tracked by Amazon Cognito, you won't have to
 re-enter an MFA code."
        )
        q.ask("Press Enter when ready.")
        auth_tokens = cog_wrapper.sign_in_with_tracked_device(
            user_name, password, device_key, device_group_key, device_password,
 aws_srp
        )
        print("You're signed in. Your access token is:")
        pp(auth_tokens["AccessToken"])
        print("-" * 88)

    print("Don't forget to delete your user pool when you're done with this
 example.")
    print("\nThanks for watching!")
    print("-" * 88)
    return user_name


def main():
    parser = argparse.ArgumentParser(
        description="Shows how to sign up a new user with Amazon Cognito and
 associate "
        "the user with an MFA application for multi-factor authentication."
    )
    parser.add_argument(
        "user_pool_id", help="The ID of the user pool to use for the example."
    )
    parser.add_argument(
        "client_id", help="The ID of the client application to use for the
 example."
    )
```

```
    args = parser.parse_args()
    client = boto3.client("cognito-idp")
    created_user = None
    try:
        created_user = run_scenario(client, args.user_pool_id, args.client_id)
    except Exception:
        logging.exception("Something went wrong with the demo.")
    finally:
        # Clean up the user this scenario created, on success or failure, so a
        # run never leaves an orphaned user in the pool. Cleanup errors are
        # logged but do not mask the original outcome.
        if created_user is not None:
            try:
                client.admin_delete_user(
                    UserPoolId=args.user_pool_id, Username=created_user
                )
                print(f"Deleted user {created_user}.")
            except Exception:
                logging.exception(
                    "Couldn't delete user %s; delete it manually.", created_user
                )


if __name__ == "__main__":
    main()
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 다음 주제를 참조하세요.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp

- VerifySoftwareToken

Swift

SDK for Swift

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

`Package.swift` 파일.

```
// swift-tools-version: 5.9
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "cognito-scenario",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
```

```
        // from dependencies.
        .executableTarget(
            name: "cognito-scenario",
            dependencies: [
                .product(name: "AWSCognitoIdentityProvider", package: "aws-sdk-
swift"),
                .product(name: "ArgumentParser", package: "swift-argument-
parser")
            ],
            path: "Sources")

    ]
)
```

Swift 코드 파일입니다.

```
// An example demonstrating various features of Amazon Cognito. Before running
// this Swift code example, set up your development environment, including
// your credentials.
//
// For more information, see the following documentation:
// https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
//
// TIP: To set up the required user pool, run the AWS Cloud Development Kit
// (AWS CDK) script provided in this GitHub repo at
// resources/cdk/cognito_scenario_user_pool_with_mfa.
//
// This example performs the following functions:
//
// 1. Invokes the signUp method to sign up a user.
// 2. Invokes the adminGetUser method to get the user's confirmation status.
// 3. Invokes the ResendConfirmationCode method if the user requested another
//    code.
// 4. Invokes the confirmSignUp method.
// 5. Invokes the initiateAuth to sign in. This results in being prompted to
//    set up TOTP (time-based one-time password). (The response is
//    “ChallengeName”: “MFA_SETUP”).
// 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private
//    key. This can be used with Google Authenticator.
// 7. Invokes the VerifySoftwareToken method to verify the TOTP and register
//    for MFA.
// 8. Invokes the AdminInitiateAuth to sign in again. This results in being
```

```
//    prompted to submit a TOTP (Response: “ChallengeName”:
//    “SOFTWARE_TOKEN_MFA”).
// 9. Invokes the AdminRespondToAuthChallenge to get back a token.

import ArgumentParser
import Foundation

import AWSClientRuntime
import AWSCognitoIdentityProvider

struct ExampleCommand: ParsableCommand {
    @Argument(help: "The application clientId.")
    var clientId: String
    @Argument(help: "The user pool ID to use.")
    var poolId: String
    @Option(help: "Name of the Amazon Region to use")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "cognito-scenario",
        abstract: """
        Demonstrates various features of Amazon Cognito.
        """,
        discussion: """
        """
    )

    /// Prompt for an input string of at least a minimum length.
    ///
    /// - Parameters:
    ///   - prompt: The prompt string to display.
    ///   - minLength: The minimum number of characters to allow in the
    ///     response. Default value is 0.
    ///
    /// - Returns: The entered string.
    func stringRequest(_ prompt: String, minLength: Int = 1) -> String {
        while true {
            print(prompt, terminator: "")
            let str = readLine()

            guard let str else {
                continue
            }
            if str.count >= minLength {
```

```
                return str
            } else {
                print("*** Response must be at least \(minLength) character(s)
long.")
            }
        }
    }

    /// Ask a yes/no question.
    ///
    /// - Parameter prompt: A prompt string to print.
    ///
    /// - Returns: `true` if the user answered "Y", otherwise `false`.
    func yesNoRequest(_ prompt: String) -> Bool {
        while true {
            let answer = stringRequest(prompt).lowercased()
            if answer == "y" || answer == "n" {
                return answer == "y"
            }
        }
    }

    /// Get information about a specific user in a user pool.
    ///
    /// - Parameters:
    ///   - cipClient: The Amazon Cognito Identity Provider client to use.
    ///   - userName: The user to retrieve information about.
    ///   - userPoolId: The user pool to search for the specified user.
    ///
    /// - Returns: `true` if the user's information was successfully
    ///   retrieved. Otherwise returns `false`.
    func adminGetUser(cipClient: CognitoIdentityProviderClient, userName: String,
                      userPoolId: String) async -> Bool {
        do {
            let output = try await cipClient.adminGetUser(
                input: AdminGetUserInput(
                    userPoolId: userPoolId,
                    username: userName
                )
            )

            guard let userStatus = output.userStatus else {
                print("*** Unable to get the user's status.")
                return false
```

```
            }

            print("User status: \(userStatus)")
            return true
        } catch {
            return false
        }
    }

    /// Create a new user in a user pool.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The ID of the app client to create a user for.
    ///   - userName: The username for the new user.
    ///   - password: The new user's password.
    ///   - email: The new user's email address.
    ///
    /// - Returns: `true` if successful; otherwise `false`.
    func signUp(cipClient: CognitoIdentityProviderClient, clientId: String,
userName: String, password: String, email: String) async -> Bool {
        let emailAttr = CognitoIdentityProviderClientTypes.AttributeType(
            name: "email",
            value: email
        )

        let userAttrsList = [emailAttr]

        do {
            _ = try await cipClient.signUp(
                input: SignUpInput(
                    clientId: clientId,
                    password: password,
                    userAttributes: userAttrsList,
                    username: userName
                )

            )

            print("=====> User \(userName) signed up.")
        } catch _ as AWSCognitoIdentityProvider.UsernameExistsException {
            print("*** The username \(userName) already exists. Please use a
different one.")
            return false
```

```
        } catch let error as AWSCognitoIdentityProvider.InvalidPasswordException
{
            print("*** Error: The specified password is invalid. Reason:
\(error.properties.message ?? "<none available>").")
            return false
        } catch _ as AWSCognitoIdentityProvider.ResourceNotFoundException {
            print("*** Error: The specified client ID (\(clientId)) doesn't
exist.")
            return false
        } catch {
            print("*** Unexpected error: \(error)")
            return false
        }

        return true
    }

    /// Requests a new confirmation code be sent to the given user's contact
    /// method.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The application client ID.
    ///   - userName: The user to resend a code for.
    ///
    /// - Returns: `true` if a new code was sent successfully, otherwise
    ///   `false`.
    func resendConfirmationCode(cipClient: CognitoIdentityProviderClient,
clientId: String,
                                userName: String) async -> Bool {
        do {
            let output = try await cipClient.resendConfirmationCode(
                input: ResendConfirmationCodeInput(
                    clientId: clientId,
                    username: userName
                )
            )

            guard let deliveryMedium = output.codeDeliveryDetails?.deliveryMedium
else {
                print("*** Unable to get the delivery method for the resent
code.")
                return false
            }
```

```
            print("=====> A new code has been sent by \(deliveryMedium)")
            return true
        } catch {
            print("*** Unable to resend the confirmation code to user
\(userName).")
            return false
        }
    }

    /// Submit a confirmation code for the specified user. This is the code as
    /// entered by the user after they've received it by email or text
    /// message.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The app client ID the user is signing up for.
    ///   - userName: The username of the user whose code is being sent.
    ///   - code: The user's confirmation code.
    ///
    /// - Returns: `true` if the code was successfully confirmed; otherwise
`false`.
    func confirmSignUp(cipClient: CognitoIdentityProviderClient, clientId:
String,
                       userName: String, code: String) async -> Bool {
        do {
            _ = try await cipClient.confirmSignUp(
                input: ConfirmSignUpInput(
                    clientId: clientId,
                    confirmationCode: code,
                    username: userName
                )
            )

            print("=====> \(userName) has been confirmed.")
            return true
        } catch {
            print("=====> \(userName)'s code was entered incorrectly.")
            return false
        }
    }

    /// Begin an authentication session.
    ///
```

```
    /// - Parameters:
    ///   - cipClient: The `CongitoIdentityProviderClient` to use.
    ///   - clientId: The app client ID to use.
    ///   - userName: The username to check.
    ///   - password: The user's password.
    ///   - userPoolId: The user pool to use.
    ///
    /// - Returns: The session token associated with this authentication
    ///   session.
    func initiateAuth(cipClient: CognitoIdentityProviderClient, clientId: String,
                      userName: String, password: String,
                      userPoolId: String) async -> String? {
        var authParams: [String: String] = [:]

        authParams["USERNAME"] = userName
        authParams["PASSWORD"] = password

        do {
            let output = try await cipClient.adminInitiateAuth(
                input: AdminInitiateAuthInput(
                    authFlow:
CognitoIdentityProviderClientTypes.AuthFlowType.adminUserPasswordAuth,
                    authParameters: authParams,
                    clientId: clientId,
                    userPoolId: userPoolId
                )
            )

            guard let challengeName = output.challengeName else {
                print("*** Invalid response from the auth service.")
                return nil
            }

            print("=====> Response challenge is \(challengeName)")

            return output.session
        } catch _ as UserNotFoundException {
            print("*** The specified username, \(userName), doesn't exist.")
            return nil
        } catch _ as UserNotConfirmedException {
            print("*** The user \(userName) has not been confirmed.")
            return nil
        } catch {
            print("*** An unexpected error occurred.")
```

```
            return nil
        }
    }

    /// Request and display an MFA secret token that the user should enter
    /// into their authenticator to set it up for the user account.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - authSession: The authentication session to request an MFA secret
    ///     for.
    ///
    /// - Returns: A string containing the MFA secret token that should be
    ///   entered into the authenticator software.
    func getSecretForAppMFA(cipClient: CognitoIdentityProviderClient,
authSession: String?) async -> String? {
        do {
            let output = try await cipClient.associateSoftwareToken(
                input: AssociateSoftwareTokenInput(
                    session: authSession
                )
            )

            guard let secretCode = output.secretCode else {
                print("*** Unable to get the secret code")
                return nil
            }

            print("=====> Enter this token into Google Authenticator:
\(secretCode)")
            return output.session
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return nil
        } catch {
            print("*** An unexpected error occurred getting the secret for the
app's MFA.")
            return nil
        }
    }

    /// Confirm that the user's TOTP authenticator is configured correctly by
    /// sending a code to it to check that it matches successfully.
    ///
```

```
    /// - Parameters:
    ///   - cipClient: The `CongnitoIdentityProviderClient` to use.
    ///   - session: An authentication session previously returned by an
    ///     `associateSoftwareToken()` call.
    ///   - mfaCode: The 6-digit code currently displayed by the user's
    ///     authenticator, as provided by the user.
    func verifyTOTP(cipClient: CognitoIdentityProviderClient, session: String?,
mfaCode: String?) async {
        do {
            let output = try await cipClient.verifySoftwareToken(
                input: VerifySoftwareTokenInput(
                    session: session,
                    userCode: mfaCode
                )
            )

            guard let tokenStatus = output.status else {
                print("*** Unable to get the token's status.")
                return
            }
            print("=====> The token's status is: \(tokenStatus)")
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return
        } catch _ as CodeMismatchException {
            print("*** The specified MFA code doesn't match the expected value.")
            return
        } catch _ as UserNotFoundException {
            print("*** The specified username doesn't exist.")
            return
        } catch _ as UserNotConfirmedException {
            print("*** The user has not been confirmed.")
            return
        } catch {
            print("*** Error verifying the MFA token!")
            return
        }
    }

    /// Respond to the authentication challenge received from Cognito after
    /// initiating an authentication session. This involves sending a current
    /// MFA code to the service.
    ///
    /// - Parameters:
```

```
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - userName: The user's username.
    ///   - clientId: The app client ID.
    ///   - userPoolId: The user pool to sign into.
    ///   - mfaCode: The 6-digit MFA code currently displayed by the user's
    ///     authenticator.
    ///   - session: The authentication session to continue processing.
    func adminRespondToAuthChallenge(cipClient: CognitoIdentityProviderClient,
userName: String,
                                     clientId: String, userPoolId: String,
mfaCode: String,
                                     session: String) async {
        print("=====> SOFTWARE_TOKEN_MFA challenge is generated...")

        var challengeResponsesOb: [String: String] = [:]
        challengeResponsesOb["USERNAME"] = userName
        challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

        do {
            let output = try await cipClient.adminRespondToAuthChallenge(
                input: AdminRespondToAuthChallengeInput(
                    challengeName:
CognitoIdentityProviderClientTypes.ChallengeNameType.softwareTokenMfa,
                    challengeResponses: challengeResponsesOb,
                    clientId: clientId,
                    session: session,
                    userPoolId: userPoolId
                )
            )

            guard let authenticationResult = output.authenticationResult else {
                print("*** Unable to get authentication result.")
                return
            }

            print("=====> Authentication result (JWTs are redacted):")
            print(authenticationResult)
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return
        } catch _ as CodeMismatchException {
            print("*** The specified MFA code doesn't match the expected value.")
            return
        } catch _ as UserNotFoundException {
```

```
            print("*** The specified username, \(userName), doesn't exist.")
            return
        } catch _ as UserNotConfirmedException {
            print("*** The user \(userName) has not been confirmed.")
            return
        } catch let error as NotAuthorizedException {
            print("*** Unauthorized access. Reason: \(error.properties.message ??
"<unknown>")")
        } catch {
            print("*** Error responding to the MFA challenge.")
            return
        }
    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let config = try await
CognitoIdentityProviderClient.CognitoIdentityProviderClientConfig(region:
region)
        let cipClient = CognitoIdentityProviderClient(config: config)

        print("""
              This example collects information about a user, then creates that
user in the
              specified user pool. Then, it enables Multi-Factor Authentication
(MFA) for that
              user by associating an authenticator application (such as Google
Authenticator
              or a password manager that supports TOTP). Then, the user uses a
code from their
              authenticator application to sign in.

              """)

        let userName = stringRequest("Please enter a new username: ")
        let password = stringRequest("Enter a password: ")
        let email = stringRequest("Enter your email address: ", minLength: 5)

        // Submit the sign-up request to AWS.

        print("==> Signing up user \(userName)...")
        if await signUp(cipClient: cipClient, clientId: clientId,
                        userName: userName, password: password,
                        email: email) == false {
```

```
            return
        }

        // Check the user's status. This time, it should come back "unconfirmed".

        print("==> Getting the status of user \(userName) from the user pool
(should be 'unconfirmed')...")
        if await adminGetUser(cipClient: cipClient, userName: userName,
userPoolId: poolId) == false {
            return
        }

        // Ask the user if they want a replacement code sent, such as if the
        // code hasn't arrived yet. If the user responds with a "yes," send a
        // new code.

        if yesNoRequest("==> A confirmation code was sent to \(userName). Would
you like to send a new code (Y/N)? ") {
            print("==> Sending a new confirmation code...")
            if await resendConfirmationCode(cipClient: cipClient, clientId:
clientId, userName: userName) == false {
                return
            }
        }

        // Ask the user to enter the confirmation code, then send it to Amazon
        // Cognito to verify it.

        let code = stringRequest("==> Enter the confirmation code sent to
\(userName): ")
        if await confirmSignUp(cipClient: cipClient, clientId: clientId,
userName: userName, code: code) == false {
            // The code didn't match. Your application may wish to offer to
            // re-send the confirmation code here and try again.
            return
        }

        // Check the user's status again. This time it should come back
        // "confirmed".

        print("==> Rechecking status of user \(userName) in the user pool (should
be 'confirmed')...")
        if await adminGetUser(cipClient: cipClient, userName: userName,
userPoolId: poolId) == false {
```

```
            return
        }
        // Check the challenge mode. Here, it should be "mfaSetup", indicating
        // that the user needs to add MFA before using it. This returns a
        // session that can be used to register MFA, or nil if an error occurs.

        let authSession = await initiateAuth(cipClient: cipClient, clientId:
clientId,
                                             userName: userName, password:
password,
                                             userPoolId: poolId)
        if authSession == nil {
            return
        }

        // Ask Cognito for an MFA secret token that the user should enter into
        // their authenticator software (such as Google Authenticator) or
        // password manager to configure it for this user account. This
        // returns a new session that should be used for the new stage of the
        // authentication process.

        let newSession = await getSecretForAppMFA(cipClient: cipClient,
authSession: authSession)
        if newSession == nil {
            return
        }

        // Ask the user to enter the current 6-digit code displayed by their
        // authenticator. Then verify that it matches the value expected for
        // the session.

        let mfaCode1 = stringRequest("==> Enter the 6-digit code displayed in
your authenticator: ",
                                     minLength: 6)
        await verifyTOTP(cipClient: cipClient, session: newSession, mfaCode:
mfaCode1)

        // Ask the user to authenticate now that the authenticator has been
        // configured. This creates a new session using the user's username
        // and password as already entered.

        print("\nNow starting the sign-in process for user \(userName)...\n")
```

```
        let session2 = await initiateAuth(cipClient: cipClient, clientId:
 clientId,
                                          userName: userName, password: password,
 userPoolId: poolId)
        guard let session2 else {
            return
        }

        // Now that we have a new auth session, `session2`, ask the user for a
        // new 6-digit code from their authenticator, and send it to the auth
        // session.

        let mfaCode2 = stringRequest("==> Wait for your authenticator to show a
 new 6-digit code, then enter it: ",
                                     minLength: 6)
        await adminRespondToAuthChallenge(cipClient: cipClient, userName:
 userName,
                                               clientId: clientId, userPoolId: poolId,
                                               mfaCode: mfaCode2, session: session2)
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- API 세부 정보는 AWS SDK for Swift API 참조의 다음 주제를 참조하세요.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge

- AssociateSoftwareToken
- ConfirmDevice
- ConfirmSignUp
- InitiateAuth
- ListUsers
- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

Amazon Cognito ID 풀 및 인증 흐름 사용

다음 코드 예제에서는 ID 풀 인증 흐름을 보여주는 웹 기반 데모 애플리케이션을 생성하는 방법을 보여줍니다.

Python

SDK for Python(Boto3)

Amazon Cognito ID 풀 인증 흐름을 보여주는 웹 기반 데모 애플리케이션을 보여 사용자가 다양한 ID 제공업체를 사용하여 향상된 인증 흐름과 기본 인증 흐름을 대화형으로 탐색할 수 있습니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Cognito 자격 증명 공급자

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 Amazon Cognito 사용자 인증 후 Lambda 함수로 사용자 지정 활동 데이터 쓰기

다음 코드 예제는 Amazon Cognito 사용자 인증 후 Lambda 함수를 사용하여 사용자 지정 작업 데이터를 쓰는 방법을 보여줍니다.

- 관리자 함수를 사용하여 사용자 풀에 사용자를 추가합니다.
- `PostAuthentication` 트리거에 대해 Lambda 함수를 호출하도록 사용자 풀을 구성합니다.
- 새로운 사용자를 Amazon Cognito에 로그인시킵니다.
- Lambda 함수는 CloudWatch Logs와 DynamoDB 테이블에 사용자 지정 정보를 작성합니다.
- DynamoDB 테이블에서 사용자 지정 데이터를 가져오기 및 표시한 다음 리소스를 정리합니다.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
import (
 "context"
 "errors"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)
```

```
// ActivityLog separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type ActivityLog struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewActivityLog constructs a new activity log runner.
func NewActivityLog(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) ActivityLog {
 scenario := ActivityLog{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddUserToPool selects a user from the known users table and uses administrator
 credentials to add the user to the user pool.
func (runner *ActivityLog) AddUserToPool(ctx context.Context, userPoolId string,
 tableName string) (string, string) {
 log.Println("To facilitate this example, let's add a user to the user pool using
 administrator privileges.")
 users, err := runner.helper.GetKnownUsers(ctx, tableName)
 if err != nil {
  panic(err)
 }
 user := users.Users[0]
 log.Printf("Adding known user %v to the user pool.\n", user.UserName)
 err = runner.cognitoActor.AdminCreateUser(ctx, userPoolId, user.UserName,
 user.UserEmail)
 if err != nil {
  panic(err)
 }
 pwSet := false
 password := runner.questioner.AskPassword("\nEnter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
```

```
  "(the password will not display as you type):", 8)
 for !pwSet {
  log.Printf("\nSetting password for user '%v'.\n", user.UserName)
  err = runner.cognitoActor.AdminSetUserPassword(ctx, userPoolId, user.UserName,
 password)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("\nEnter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   pwSet = true
  }
 }

 log.Println(strings.Repeat("-", 88))

 return user.UserName, password
}

// AddActivityLogTrigger adds a Lambda handler as an invocation target for the
 PostAuthentication trigger.
func (runner *ActivityLog) AddActivityLogTrigger(ctx context.Context, userPoolId
 string, activityLogArn string) {
 log.Println("Let's add a Lambda function to handle the PostAuthentication
 trigger from Cognito.\n" +
  "This trigger happens after a user is authenticated, and lets your function
 take action, such as logging\n" +
  "the outcome.")
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
  actions.TriggerInfo{Trigger: actions.PostAuthentication, HandlerArn:
 aws.String(activityLogArn)})
 if err != nil {
  panic(err)
 }
 runner.resources.triggers = append(runner.resources.triggers,
 actions.PostAuthentication)
 log.Printf("Lambda function %v added to user pool %v to handle
 PostAuthentication Cognito trigger.\n",
  activityLogArn, userPoolId)
```

```
 log.Println(strings.Repeat("-", 88))
}

// SignInUser signs in as the specified user.
func (runner *ActivityLog) SignInUser(ctx context.Context, clientId string,
 userName string, password string) {
 log.Printf("Now we'll sign in user %v and check the results in the logs and the
 DynamoDB table.", userName)
 runner.questioner.Ask("Press Enter when you're ready.")
 authResult, err := runner.cognitoActor.SignIn(ctx, clientId, userName, password)
 if err != nil {
  panic(err)
 }
 log.Println("Sign in successful.",
  "The PostAuthentication Lambda handler writes custom information to CloudWatch
 Logs.")

 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)
}

// GetKnownUserLastLogin gets the login info for a user from the Amazon DynamoDB
 table and displays it.
func (runner *ActivityLog) GetKnownUserLastLogin(ctx context.Context, tableName
 string, userName string) {
 log.Println("The PostAuthentication handler also writes login data to the
 DynamoDB table.")
 runner.questioner.Ask("Press Enter when you're ready to continue.")
 users, err := runner.helper.GetKnownUsers(ctx, tableName)
 if err != nil {
  panic(err)
 }
 for _, user := range users.Users {
  if user.UserName == userName {
   log.Println("The last login info for the user in the known users table is:")
   log.Printf("\t%+v", *user.LastLogin)
  }
 }
 log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
func (runner *ActivityLog) Run(ctx context.Context, stackName string) {
 defer func() {
```

```
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]
 runner.helper.PopulateUserTable(ctx, stackOutputs["TableName"])
 userName, password := runner.AddUserToPool(ctx, stackOutputs["UserPoolId"],
 stackOutputs["TableName"])

 runner.AddActivityLogTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["ActivityLogFunctionArn"])
 runner.SignInUser(ctx, stackOutputs["UserPoolClientId"], userName, password)
 runner.helper.ListRecentLogEvents(ctx, stackOutputs["ActivityLogFunction"])
 runner.GetKnownUserLastLogin(ctx, stackOutputs["TableName"], userName)

 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Lambda 함수를 사용하여 `PostAuthentication` 트리거를 처리합니다.

```
import (
 "context"
 "fmt"
 "log"
 "os"
```

```
 "time"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 dynamodbtypes "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

const TABLE_NAME = "TABLE_NAME"

// LoginInfo defines structured login data that can be marshalled to a DynamoDB
 format.
type LoginInfo struct {
 UserPoolId string `dynamodbav:"UserPoolId"`
 ClientId   string `dynamodbav:"ClientId"`
 Time       string `dynamodbav:"Time"`
}

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string    `dynamodbav:"UserName"`
 UserEmail string    `dynamodbav:"UserEmail"`
 LastLogin LoginInfo `dynamodbav:"LastLogin"`
}

// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
 userEmail, err := attributevalue.Marshal(user.UserEmail)
 if err != nil {
  panic(err)
 }
 return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the PostAuthentication event by writing custom data to
 the logs and
```

```
// to an Amazon DynamoDB table.
func (h *handler) HandleRequest(ctx context.Context,
 event events.CognitoEventUserPoolsPostAuthentication)
 (events.CognitoEventUserPoolsPostAuthentication, error) {
 log.Printf("Received post authentication trigger from %v for user '%v'",
 event.TriggerSource, event.UserName)
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserName:  event.UserName,
  UserEmail: event.Request.UserAttributes["email"],
  LastLogin: LoginInfo{
   UserPoolId: event.UserPoolID,
   ClientId:   event.CallerContext.ClientID,
   Time:       time.Now().Format(time.UnixDate),
  },
 }
 // Write to CloudWatch Logs.
 fmt.Printf("%#v", user)

 // Also write to an external system. This examples uses DynamoDB to demonstrate.
 userMap, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshal to DynamoDB map. Here's why: %v\n", err)
 } else if len(userMap) == 0 {
  log.Printf("User info marshaled to an empty map.")
 } else {
  _, err := h.dynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
   Item:      userMap,
   TableName: aws.String(tableName),
  })
  if err != nil {
   log.Printf("Couldn't write to DynamoDB. Here's why: %v\n", err)
  } else {
   log.Printf("Wrote user info to DynamoDB table %v.\n", tableName)
  }
 }

 return event, nil
}

func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
```

```
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

일반적인 작업을 수행하는 구조체를 생성합니다.

```
import (
 "context"
 "log"
 "strings"
 "time"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
```

```
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName
 string) (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
```

```
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context,
 functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
```

```
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Amazon Cognito 작업을 래핑하는 구조체를 생성합니다.

```
import (
 "context"
 "errors"
 "log"

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

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}
```

```
// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId
 string, triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
```

```
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
```

```
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}
```

```
// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken
 string) error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId
 string, userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}
```

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

DynamoDB 작업을 래핑하는 구조체를 생성합니다.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
```

```
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
```

```
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user
 User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
```

```
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

CloudWatch Logs 작업을 래핑하는 구조체를 생성합니다.

```
import (
 "context"
 "fmt"
 "log"

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

type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
```

```
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx,
 &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

CloudFormation 작업을 래핑하는 구조체를 생성합니다.

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string
```

```
type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName
 string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

리소스를 정리합니다.

```
import (
 "context"
 "log"
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger
```

```
 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 \n" +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(ctx, accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
 HandlerArn: nil}
  }
  err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
 triggerList...)
  if err != nil {
```

```
    log.Println("Couldn't update Cognito triggers during cleanup.")
    panic(err)
   }
   log.Println("Removed Cognito triggers from user pool.")
  } else {
   log.Println("Be sure to remove resources when you're done with them to avoid
  unexpected charges!")
  }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 다음 주제를 참조하세요.
  - AdminCreateUser
  - AdminSetUserPassword
  - DeleteUser
  - InitiateAuth
  - UpdateUserPool

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDKs를 사용한 Amazon Cognito Sync의 코드 예제

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

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

Amazon Cognito Sync

- AWS SDKs 사용한 Amazon Cognito Sync의 기본 예제
  - AWS SDKs를 사용한 Amazon Cognito Sync 작업
    - AWS SDK와 ListIdentityPoolUsage 함께 사용

AWS SDKs 사용한 Amazon Cognito Sync의 기본 예제

다음 코드 예제는 Amazon Cognito Sync의 기본 사항을 AWS SDK와 함께 사용하는 방법을 보여줍니다.

예제

- AWS SDKs를 사용한 Amazon Cognito Sync 작업
  - AWS SDK와 ListIdentityPoolUsage 함께 사용

AWS SDKs를 사용한 Amazon Cognito Sync 작업

다음 코드 예제에서는 AWS SDKs를 사용하여 개별 Amazon Cognito Sync 작업을 수행하는 방법을 보여줍니다. 각 예시에는 GitHub에 대한 링크가 포함되어 있습니다. 여기에서 코드 설정 및 실행에 대한 지침을 찾을 수 있습니다.

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

예제

- AWS SDK와 ListIdentityPoolUsage 함께 사용

AWS SDK와 **`ListIdentityPoolUsage`** 함께 사용

다음 코드 예시는 `ListIdentityPoolUsage`의 사용 방법을 보여 줍니다.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn show_pools(client: &Client) -> Result<(), Error> {
    let response = client
        .list_identity_pool_usage()
```

```
        .max_results(10)
        .send()
        .await?;

    let pools = response.identity_pool_usages();
    println!("Identity pools:");

    for pool in pools {
        println!(
            "  Identity pool ID:    {}",
            pool.identity_pool_id().unwrap_or_default()
        );
        println!(
            "  Data storage:        {}",
            pool.data_storage().unwrap_or_default()
        );
        println!(
            "  Sync sessions count: {}",
            pool.sync_sessions_count().unwrap_or_default()
        );
        println!(
            "  Last modified:       {}",
            pool.last_modified_date().unwrap().to_chrono_utc()?
        );
        println!();
    }

    println!("Next token: {}", response.next_token().unwrap_or_default());

    Ok(())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 ListIdentityPoolUsage를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

기타 기능

Amazon Cognito는 사용자 풀을 사용하여 애플리케이션에 로그인하는 것 외에도 특수한 요구 사항에 맞는 추가 기능을 제공합니다. Amazon Cognito 자격 증명 풀은 사용자의 자격 증명을 권한이 제한된 임시 AWS 자격 증명과 교환하여 애플리케이션이 AWS 서비스를 직접 호출할 수 있도록 합니다.

주제

- Amazon Cognito 자격 증명 풀
- Amazon Cognito ID 풀 시작

Amazon Cognito 자격 증명 풀

Amazon Cognito 자격 증명 풀은 AWS 보안 인증으로 교환할 수 있는 페더레이션 자격 증명의 디렉터리입니다. 자격 증명 풀은 앱 사용자가 로그인했는지 또는 아직 식별하지 않았는지에 관계없이 앱 사용자를 위한 임시 AWS 자격 증명을 생성합니다. AWS Identity and Access Management (IAM) 역할 및 정책을 사용하면 사용자에게 부여할 권한 수준을 선택할 수 있습니다. 사용자는 게스트로 시작하고 AWS 서비스에 보관하는 자산을 검색할 수 있습니다. 그런 다음 사용자는 서드 파티 ID 제공업체를 통해 로그인하여 등록된 회원에게 제공되는 자산에 대한 액세스 권한을 잠금 해제할 수 있습니다. 서드 파티 ID 제공업체는 Apple이나 Google 같은 소비자(소셜) OAuth 2.0 공급자, 사용자 지정 SAML 또는 OIDC ID 제공업체 또는 자체 설계한 사용자 지정 인증 체계(개발자 공급자라고도 함)일 수 있습니다.

Amazon Cognito 자격 증명 풀의 기능

에 대한 요청 서명 AWS 서비스

Amazon Simple Storage Service(Amazon S3) 및 Amazon DynamoDB와 AWS 서비스 같은에 대한 API 요청에 서명합니다. Amazon Pinpoint와 Amazon CloudWatch 같은 서비스를 사용하여 사용자 활동을 분석합니다.

리소스 기반 정책으로 요청 필터링

리소스에 대한 사용자 액세스에 대한 세분화된 제어를 실행합니다. 사용자 클레임을 IAM 세션 태그로 변환하고 개별 사용자 하위 집합에 리소스 액세스 권한을 부여하는 IAM 정책을 구축합니다.

게스트 액세스 할당

아직 로그인하지 않은 사용자의 경우 액세스 범위가 좁은 AWS 보안 인증을 생성하도록 자격 증명 풀을 구성합니다. SSO(Single Sign-On) 공급자를 통해 사용자를 인증하여 액세스 권한을 높입니다.

사용자 특성에 따라 IAM 역할 할당

인증된 모든 사용자에게 단일 IAM 역할을 할당하거나 각 사용자의 클레임을 기반으로 역할을 선택합니다.

다양한 ID 제공업체 수락

자격 AWS 증명을 위해 ID 또는 액세스 토큰, 사용자 풀 토큰, SAML 어설션 또는 소셜 공급자 OAuth 토큰을 교환합니다.

자체 자격 증명 검증

자체 사용자 검증을 수행하고 개발자 AWS 자격 증명을 사용하여 사용자에게 자격 증명을 발급합니다.

인증 및 권한 부여 서비스를 앱에 제공하는 Amazon Cognito 사용자 풀이 이미 있을 수도 있습니다. 사용자 풀을 자격 증명 풀에 대한 ID 제공업체(idP)로 설정할 수 있습니다. 이 경우 사용자는 사용자 풀 IdPs 통해 인증하고, 클레임을 공통 OIDC 자격 증명 토큰으로 통합하고, 해당 토큰을 AWS 자격 증명으로 교환할 수 있습니다. 그러면 사용자는 서명된 요청의 보안 인증을 AWS 서비스에 제시할 수 있습니다.

또한 어떤 ID 제공업체의 인증된 클레임도 자격 증명 풀에 직접 제시할 수 있습니다. Amazon Cognito는 SAML, OAuth, OIDC 공급자의 사용자 클레임을 단기 보안 인증을 위한 AssumeRoleWithWebIdentity API 요청으로 사용자 지정합니다.

Amazon Cognito 사용자 풀은 SSO 지원 앱에 대한 OIDC ID 제공업체와 같습니다. 자격 증명 풀은 IAM 권한 부여에서 가장 잘 작동하는 리소스 종속성이 있는 앱에 AWS ID 제공업체 역할을 합니다.

Amazon Cognito 자격 증명 풀은 다음 자격 증명 공급자를 지원합니다.

- 퍼블릭 공급자: Amazon을 사용하여 ID 풀 IdP로 로그인 설정, Facebook을 ID 풀 IdP로 설정, Google을 ID 풀 IdP로 설정, Sign in with Apple을 ID 풀 IdP로 설정, Twitter.
- Amazon Cognito 사용자 풀
- OIDC 제공업체를 ID 풀 IdP로 설정
- SAML 제공업체를 ID 풀 IdP로 설정
- 개발자 인증 자격 증명

Amazon Cognito 자격 증명 풀 리전 가용성에 대한 자세한 내용은 AWS 서비스 리전 가용성을 참조하세요.

Amazon Cognito 자격 증명 풀에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- ID 풀 콘솔 개요
- 자격 증명 풀 인증 흐름
- IAM 역할
- Amazon Cognito ID 풀을 위한 보안 모범 사례
- 액세스 제어에 속성 사용
- 역할 기반 액세스 제어 사용
- 자격 증명 얻기
- 임시 자격 증명 AWS 서비스 으로 액세스
- ID 풀 타사 ID 제공업체
- 개발자 인증 자격 증명
- 인증되지 않은 사용자를 인증된 사용자로 전환

ID 풀 콘솔 개요

Amazon Cognito 자격 증명 풀은 게스트(인증되지 않음)인 사용자와 인증되어 토큰을 받은 사용자에게 임시 AWS 자격 증명을 제공합니다. ID 풀은 외부 ID 제공업체에 연결된 사용자 식별자의 저장소입니다.

ID 풀의 기능과 옵션을 이해하는 한 가지 방법은 Amazon Cognito 콘솔에서 생성하는 것입니다. 다양한 설정이 인증 흐름, 역할 기반 및 속성 기반 액세스 제어, 게스트 액세스에 미치는 영향을 탐색할 수 있습니다. 여기에서 이 가이드의 후반부 장으로 이동하여 애플리케이션에 적절한 구성 요소를 추가하여 ID 풀 인증을 구현할 수 있습니다.

주제

- 자격 증명 풀 생성
- 사용자 IAM 역할
- 인증된 자격 증명 및 인증되지 않은 자격 증명
- 게스트 액세스 활성화 또는 비활성화
- 자격 증명 유형과 연관된 역할 변경
- ID 제공업체 편집
- 자격 증명 풀 삭제

- 자격 증명 풀에서 자격 증명 삭제

자격 증명 풀 생성

콘솔에서 새 자격 증명 풀을 만들려면

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

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

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

    b. 에 사용하려는 역할이 이미 있는 경우 기존 IAM 역할 사용을 선택할 수 있습니다. AWS 계정 `cognito-identity.amazonaws.com`을 포함하도록 IAM 역할 신뢰 정책을 구성해야 합니다. 특정 자격 증명 풀의 인증된 사용자로부터 요청이 시작되었다는 증거가 제시되는 경우에만 Amazon Cognito가 역할을 맡을 수 있도록 역할 신뢰 정책을 구성합니다. 자세한 내용은 역할 트러스트 및 권한 섹션을 참조하세요.
5. ID 제공업체 연결에서 자격 증명 풀 신뢰 구성에서 선택한 ID 제공업체(idP)의 세부 정보를 입력합니다. OAuth 앱 클라이언트 정보를 제공하거나, Amazon Cognito 사용자 풀을 선택하거나, IAM IdP를 선택하거나, 개발자 공급자의 사용자 지정 식별자를 입력하라는 메시지가 표시될 수 있습니다.

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

i. 규칙으로 역할 선택을 선택한 경우 사용자 인증의 소스 클레임, 클레임을 비교할 연산자, 이 역할 선택과 일치하는 값을 입력하고 역할 할당이 일치할 때 할당할 역할을 입력합니다. 다른 조건에 따라 추가 규칙을 생성하려면 다른 항목 추가를 선택합니다.

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

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

i. 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.

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

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

6. 속성 구성에서 자격 증명 풀 이름 아래에 이름을 입력합니다.
7. 기본(Classic) 인증에서 기본 흐름 활성화 여부를 선택합니다. 기본 흐름이 활성화되면 IdP에 대한 역할 선택을 우회하고 AssumeRoleWithWebIdentity를 직접 호출할 수 있습니다. 자세한 내용은 자격 증명 풀 인증 흐름 섹션을 참조하세요.
8. 자격 증명 풀에 태그를 적용하려면 태그에서 태그 추가를 선택합니다.
9. 검토 및 생성에서 새 자격 증명 풀에 대한 선택 사항을 확인합니다. 편집을 선택하여 마법사로 돌아가서 설정을 변경합니다. 완료하면 자격 증명 풀 생성을 선택합니다.

사용자 IAM 역할

IAM 역할은 사용자가와 같은 AWS 리소스에 액세스할 수 있는 권한을 정의합니다Amazon Cognito Sync. 애플리케이션의 사용자가 생성한 역할을 수임합니다. 인증된 사용자와 인증되지 않은 사용자에게 다른 역할을 지정할 수 있습니다. IAM 역할에 대한 자세한 내용은 IAM 역할 섹션을 참조하세요.

인증된 자격 증명 및 인증되지 않은 자격 증명

Amazon Cognito 자격 증명 풀은 인증된 자격 증명과 인증되지 않은 자격 증명을 모두 지원합니다. 인증된 자격 증명은 지원되는 자격 증명 공급자가 인증한 사용자를 위한 것이고, 인증되지 않은 자격 증명은 대개 게스트 사용자를 위한 것입니다.

- 퍼블릭 로그인 공급자를 통해 인증된 자격 증명을 구성하려면 ID 풀 타사 ID 제공업체 섹션을 참조하세요.

- 자체 백엔드 인증 프로세스를 구성하려면 섹션을 참조하세요개발자 인증 자격 증명.

게스트 액세스 활성화 또는 비활성화

Amazon Cognito 자격 증명 풀 게스트 액세스(인증되지 않은 자격 증명)는 자격 증명 공급자로 인증하지 않는 사용자에게 고유한 식별자 및 AWS 자격 증명을 제공합니다. 애플리케이션에서 로그인하지 않은 사용자를 허용하는 경우 인증되지 않은 자격 증명에 대한 액세스를 활성화할 수 있습니다. 자세한 내용은 Amazon Cognito ID 풀 시작 섹션을 참조하세요.

자격 증명 풀에서 게스트 액세스를 업데이트하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. 게스트 액세스를 찾습니다. 현재 게스트 액세스를 지원하지 않는 자격 증명 풀에서는 상태가 비활성입니다.

   a. 게스트 액세스가 활성이고 이를 비활성화하려면 비활성화를 선택합니다.
   b. 게스트 액세스가 비활성이고 이를 활성화하려면 편집을 선택합니다.

      - 자격 증명 풀의 게스트 사용자의 기본 IAM 역할을 선택합니다.

        A. Amazon Cognito가 기본 권한 및 자격 증명 풀과의 신뢰 관계가 있는 새 역할을 생성하도록 하려면 새 IAM 역할 생성을 선택합니다. 새 역할을 식별하는 IAM 역할 이름을 입력합니다(예: `myidentitypool_authenticatedrole`). 정책 문서 보기를 선택하여 Amazon Cognito가 새 IAM 역할에 할당하는 권한을 검토합니다.
        B. 에 사용하려는 역할이 이미 있는 경우 기존 IAM 역할 사용을 선택할 수 있습니다. AWS 계정 `cognito-identity.amazonaws.com`을 포함하도록 IAM 역할 신뢰 정책을 구성해야 합니다. 특정 자격 증명 풀의 인증된 사용자로부터 요청이 시작되었다는 증거가 제시되는 경우에만 Amazon Cognito가 역할을 맡을 수 있도록 역할 신뢰 정책을 구성합니다. 자세한 내용은 역할 트러스트 및 권한 섹션을 참조하세요.
        C. 변경 사항 저장을 선택합니다.
        D. 게스트 액세스를 활성화하려면 사용자 액세스 탭에서 활성화를 선택합니다.

자격 증명 유형과 연관된 역할 변경

자격 증명 풀의 모든 자격 증명은 인증 또는 미인증입니다. 인증된 자격 증명은 퍼블릭 로그인 공급자(Amazon Cognito 사용자 풀, Login with Amazon, Sign in with Apple, Facebook, Google, SAML 또는

OpenID Connect Providers) 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 권한을 지정하는 정책 AWS 서비스 이 연결되어 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 어떤 AWS 서비스 자격 증명 유형에 액세스할 수 있는지 제어할 수 있습니다. 자격 증명 풀에서 역할과 연관된 정책을 보거나 수정하려면 AWS IAM 콘솔을 참조하세요.

자격 증명 풀 기본 인증 역할 또는 인증되지 않은 역할을 변경하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. 게스트 액세스 또는 인증된 액세스를 찾습니다. 현재 해당 액세스 유형에 대해 구성되지 않은 자격 증명 풀에서는 상태가 비활성입니다. 편집을 선택합니다.
4. 자격 증명 풀의 게스트 또는 인증된 사용자의 기본 IAM 역할을 선택합니다.

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

    b. 에 사용하려는 역할이 이미 있는 경우 기존 IAM 역할 사용을 선택할 수 있습니다. AWS 계정 `cognito-identity.amazonaws.com`을 포함하도록 IAM 역할 신뢰 정책을 구성해야 합니다. 특정 자격 증명 풀의 인증된 사용자로부터 요청이 시작되었다는 증거가 제시되는 경우에만 Amazon Cognito가 역할을 맡을 수 있도록 역할 신뢰 정책을 구성합니다. 자세한 내용은 역할 트러스트 및 권한 섹션을 참조하세요.
5. 변경 사항 저장을 선택합니다.

ID 제공업체 편집

사용자가 소비자 ID 제공업체(예: Amazon Cognito 사용자 풀, Login with Amazon, Sign in with Apple, Facebook, Google)를 사용하여 인증하는 것을 허용하는 경우 Amazon Cognito 자격 증명 풀(페더레이션 자격 증명) 콘솔에서 애플리케이션 식별자를 지정할 수 있습니다. 이 퍼블릭 자격 증명 공급자는 애플리케이션 ID(퍼블릭 로그인 공급자가 제공함)와 자격 증명 풀을 연결합니다.

이 페이지에서 각 공급자에 대한 인증 규칙을 구성할 수도 있습니다. 각 공급자에 대해 최대 25개의 규칙이 허용됩니다. 규칙은 저장한 순서대로 각 공급자에 적용됩니다. 자세한 내용은 역할 기반 액세스 제어 사용 섹션을 참조하세요.

> **Warning**
>
> 자격 증명 풀에서 연결된 IdP 애플리케이션 ID를 변경하면 기존 사용자가 해당 자격 증명 풀을 사용하여 인증할 수 없게 됩니다. 자세한 내용은 ID 풀 타사 ID 제공업체 섹션을 참조하세요.

자격 증명 풀 ID 제공업체(idP)를 업데이트하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. ID 제공업체를 찾습니다. 편집할 ID 제공업체를 선택합니다. 새 IdP를 추가하려면 ID 제공업체 추가를 선택합니다.
   - ID 제공업체 추가를 선택한 경우 추가하려는 자격 증명 유형 중 하나를 선택합니다.
4. 애플리케이션 ID를 변경하려면 ID 제공업체 정보에서 편집을 선택합니다.
5. Amazon Cognito가 이 공급자를 통해 인증한 사용자에게 보안 인증을 발급할 때 요청하는 역할을 변경하려면 역할 설정에서 편집을 선택합니다.
   - 인증된 역할을 구성할 때 설정한 기본 역할을 해당 IdP의 사용자에게 할당하거나 규칙을 사용하여 역할 선택을 선택할 수 있습니다. Amazon Cognito 사용자 풀 IdP를 사용하면 토큰에서 preferred_role을 포함한 역할을 선택할 수도 있습니다. `cognito:preferred_role` 클레임에 대한 자세한 내용은 그룹에 우선 순위 값 할당을 참조하세요.
     i. 규칙으로 역할 선택을 선택한 경우 사용자 인증의 소스 클레임, 클레임을 비교할 연산자, 이 역할 선택과 일치하는 값을 입력하고 역할 할당이 일치할 때 할당할 역할을 입력합니다. 다른 조건에 따라 추가 규칙을 생성하려면 다른 항목 추가를 선택합니다.
     ii. 역할 해결을 선택합니다. 사용자의 클레임이 규칙과 일치하지 않는 경우 보안 인증을 거부하거나 인증된 역할의 보안 인증을 발급할 수 있습니다.
6. Amazon Cognito가 이 공급자를 통해 인증한 사용자에게 보안 인증을 발급할 때 할당하는 보안 주체 태그를 변경하려면 액세스 제어를 위한 속성에서 편집을 선택합니다.
   a. 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.
   b. `sub` 및 `aud` 클레임 기반 보안 주체 태그를 적용하려면 기본 매핑 사용을 선택합니다.

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

7. 변경 사항 저장을 선택합니다.

자격 증명 풀 삭제

자격 증명 풀 삭제는 실행 취소할 수 없습니다. 자격 증명 풀을 삭제하면 해당 자격 증명 풀을 사용하는 모든 앱과 사용자의 작동이 중지됩니다.

자격 증명 풀을 삭제하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 삭제할 자격 증명 풀 옆의 라디오 버튼을 선택합니다.
2. 삭제를 선택합니다.
3. 자격 증명 풀의 이름을 입력하거나 붙여넣고 삭제를 선택합니다.

> **Warning**
>
> 삭제(Delete) 버튼을 선택하면 자격 증명 풀과 여기에 포함된 모든 사용자 데이터를 영구적으로 삭제합니다. 자격 증명 풀을 삭제하면 자격 증명 풀을 사용하는 애플리케이션 및 기타 서비스가 작동을 중지합니다.

자격 증명 풀에서 자격 증명 삭제

자격 증명 풀에서 자격 증명을 삭제하면 Amazon Cognito가 해당 페더레이션 사용자에 대해 저장한 식별 정보가 제거됩니다. 사용자가 보안 인증을 다시 요청하면 자격 증명 풀이 여전히 사용자의 ID 제공 업체를 신뢰하는 경우 새 자격 증명 ID를 받게 됩니다. 이 작업은 실행 취소할 수 없습니다.

자격 증명을 삭제하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 자격 증명 브라우저 탭을 선택합니다.
3. 삭제하려는 자격 증명 옆의 확인란을 선택하고 삭제를 선택합니다. 자격 증명을 정말로 삭제할지 다시 묻는 메시지가 나오면 확인 후 삭제를 선택합니다.

자격 증명 풀 인증 흐름

Amazon Cognito를 사용하면 디바이스와 플랫폼에서 일관성이 유지되는 최종 사용자의 고유한 식별자를 생성할 수 있습니다. 또한 Amazon Cognito는 AWS 리소스에 액세스하기 위해 애플리케이션에 제한된 권한의 임시 자격 증명을 제공합니다. 이 페이지에서는 Amazon Cognito의 기본 인증 방법을 다루며, 자격 증명 풀 내에서 자격 증명의 수명 주기에 대해 설명합니다.

외부 공급자 인증 흐름

Amazon Cognito로 인증하는 사용자는 자격 증명을 부트스트랩하기 위해 다단계 프로세스를 진행합니다. Amazon Cognito는 퍼블릭 공급자로부터 인증을 받기 위한, 향상된 흐름과 기본 흐름의 두 가지 흐름을 제공합니다.

이러한 흐름 중 하나를 완료하면 역할의 액세스 정책에 정의된 AWS 서비스 대로 다른에 액세스할 수 있습니다. 기본적으로 Amazon Cognito 콘솔은 Amazon Cognito Sync 스토어 및 Amazon Mobile Analytics에 액세스할 수 있는 역할을 생성합니다. 추가 액세스 권한을 부여하는 방법에 대한 자세한 내용은 IAM 역할을 참조하세요.

ID 풀은 공급자의 다음 아티팩트를 수락합니다.

제공업체	인증 아티팩트
Amazon Cognito 사용자 풀	ID 토큰
OIDC(OpenID Connect)	ID 토큰
SAML 2.0	SAML 어설션
소셜 공급자	액세스 토큰

향상된(단순화된) 인증 흐름

향상된 인증 흐름을 사용하면 앱은 먼저 GetId 요청에서 인증된 Amazon Cognito 사용자 풀 또는 타사 ID 제공업체의 인증 증명을 제공합니다.

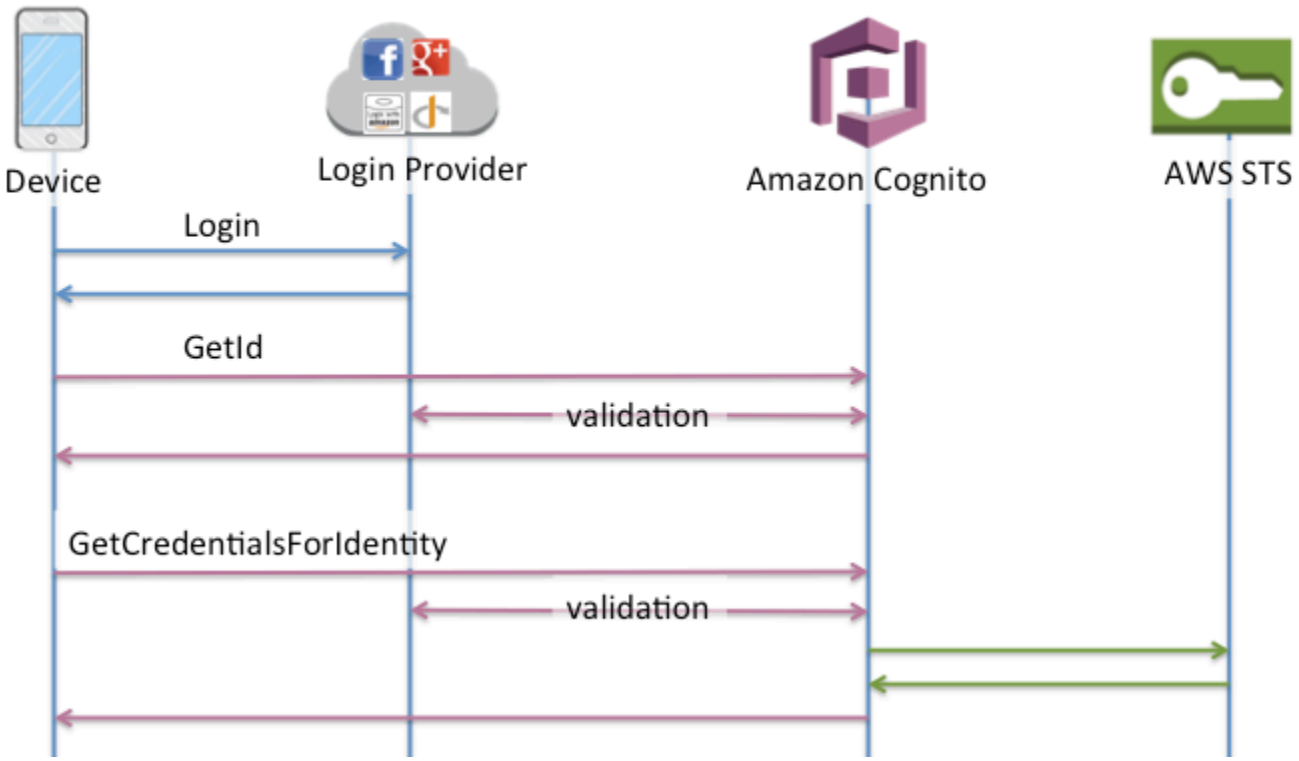
1. 애플리케이션은 GetID 요청에서 승인된 Amazon Cognito 사용자 풀 또는 타사 ID 제공업체의 JSON 웹 토큰 또는 SAML 어설션 인증 증명을 제공합니다.
2. 사용자의 ID 풀은 자격 증명 ID를 반환합니다.

3. 애플리케이션은 GetCredentialsForIdentity 요청에서 자격 증명 ID를 동일한 인증 증명과 결합합니다.
4. 자격 증명 풀은 AWS 자격 증명을 반환합니다.
5. 애플리케이션은 임시 자격 증명으로 AWS API 요청에 서명합니다.

향상된 인증은 ID 풀 구성에서 IAM 역할 선택 및 자격 증명 검색의 로직을 관리합니다. 기본 역할을 선택하고, 속성 기반 액세스 제어(ABAC) 또는 역할 기반 액세스 제어(RBAC) 원칙을 역할 선택에 적용하도록 ID 풀을 구성할 수 있습니다. 향상된 인증의 자격 AWS 증명은 1시간 동안 유효합니다.

향상된 인증의 작업 순서

1. `GetId`
2. `GetCredentialsForIdentity`



기본(클래식) 인증 흐름

기본 인증 흐름을 구현하면 애플리케이션에서 사용자가 수임할 IAM 역할을 선택합니다.

1. 애플리케이션은 GetID 요청에서 승인된 Amazon Cognito 사용자 풀 또는 타사 ID 제공업체의 JSON 웹 토큰 또는 SAML 어설션 인증 증명을 제공합니다.
2. 사용자의 ID 풀은 자격 증명 ID를 반환합니다.
3. 애플리케이션은 GetOpenIdToken 요청에서 자격 증명 ID를 동일한 인증 증명과 결합합니다.
4. `GetOpenIdToken`은 ID 풀에서 발급한 새 OAuth 2.0 토큰을 반환합니다.
5. 애플리케이션에서 AssumeRoleWithWebIdentity 요청에 새 토큰을 표시합니다.
6. AWS Security Token Service (AWS STS)는 AWS 자격 증명을 반환합니다.
7. 애플리케이션은 임시 자격 증명으로 AWS API 요청에 서명합니다.

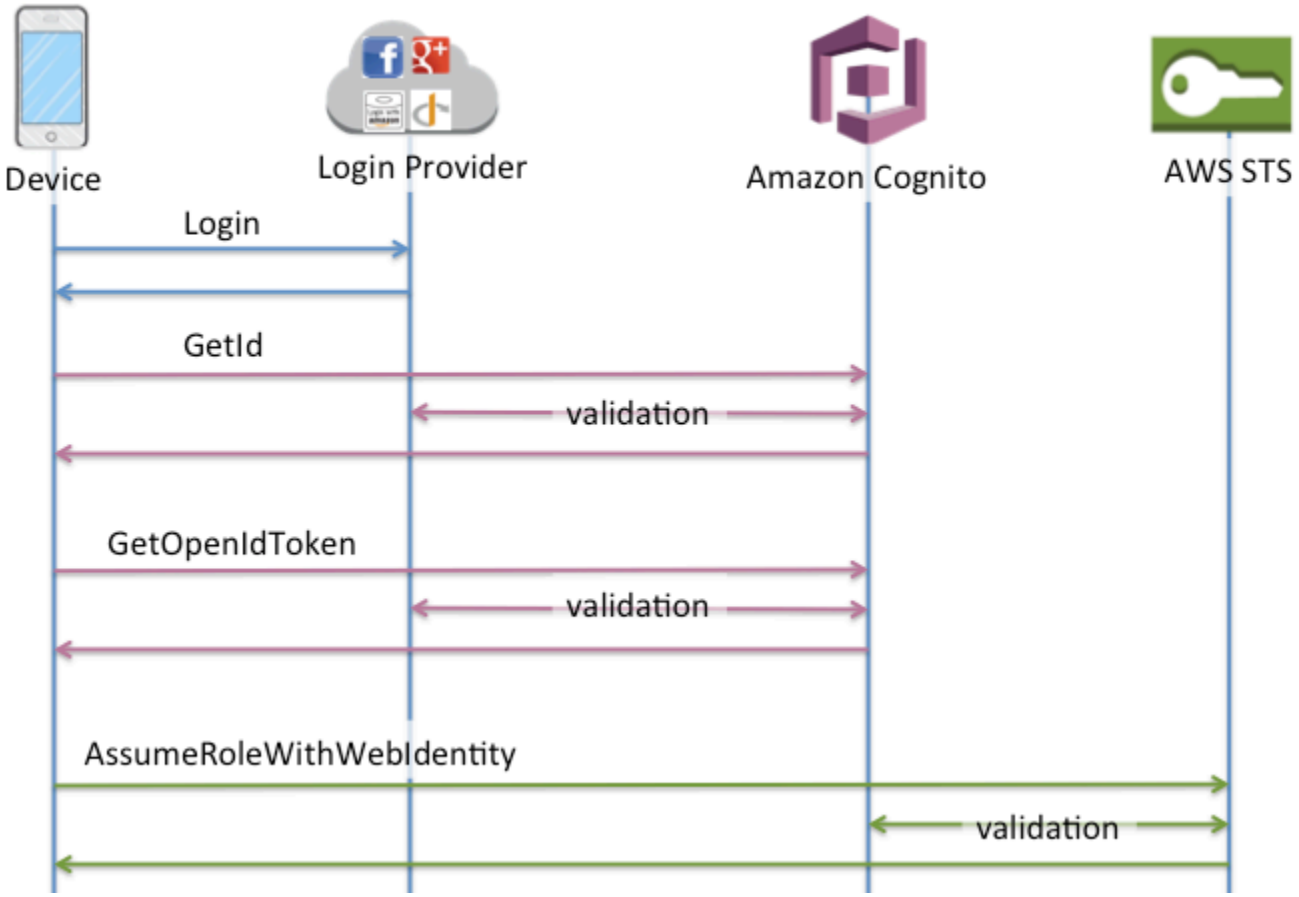
기본 워크플로를 통해 사용자에게 배포하는 자격 증명을 보다 세밀하게 제어할 수 있습니다. 향상된 인증 흐름의 `GetCredentialsForIdentity` 요청은 액세스 토큰의 내용을 기반으로 역할을 요청합니다. 클래식 워크플로의 `AssumeRoleWithWebIdentity` 요청은 앱에 충분한 신뢰 정책으로 구성한 AWS Identity and Access Management 역할에 대한 자격 증명을 요청할 수 있는 더 많은 기능을 부여합니다. 사용자 지정 역할 세션 기간을 요청할 수도 있습니다.

역할 매핑이 없는 사용자 풀에서 기본 인증 흐름으로 로그인할 수 있습니다. 이 유형의 ID 풀에는 기본 인증 또는 인증되지 않은 역할이 없으며 역할 기반 또는 속성 기반 액세스 제어가 구성되어 있지 않습니다. 역할 매핑이 있는 ID 풀에서 `GetOpenIdToken`를 시도하면 다음과 같은 오류가 발생합니다.

```
Basic (classic) flow is not supported with RoleMappings, please use enhanced flow.
```

기본 인증의 작업 순서

1. `GetId`
2. `GetOpenIdToken`
3. `AssumeRoleWithWebIdentity`

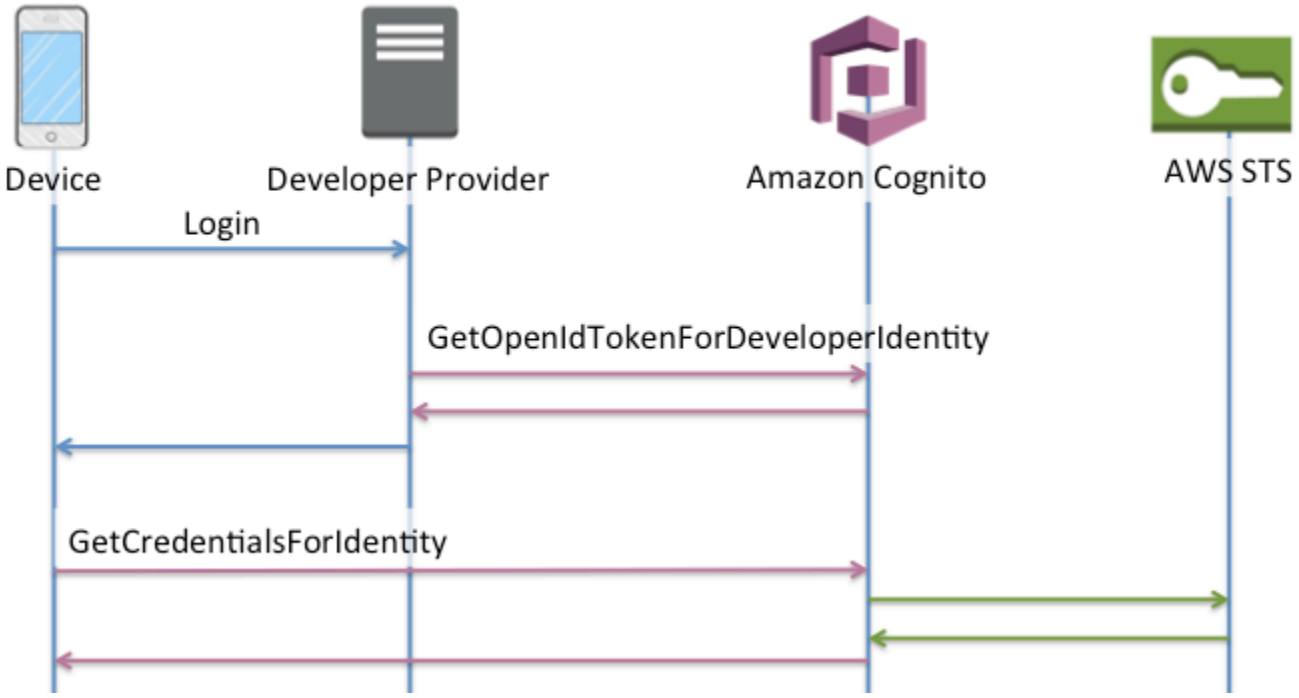


개발자가 인증한 인증 흐름

개발자 인증 자격 증명을 사용할 때 클라이언트는 Amazon Cognito에 속하지 않는 코드가 포함된 다른 인증 흐름을 사용하여 자체 인증 시스템에서 사용자를 검증합니다. ID 풀의 관점에서 자격 증명에 대한 요청에 제시되는 클레임은 임의 식별자이며 인증은 애플리케이션에서 인코딩하는 IAM 자격 증명에 의해 승인됩니다.

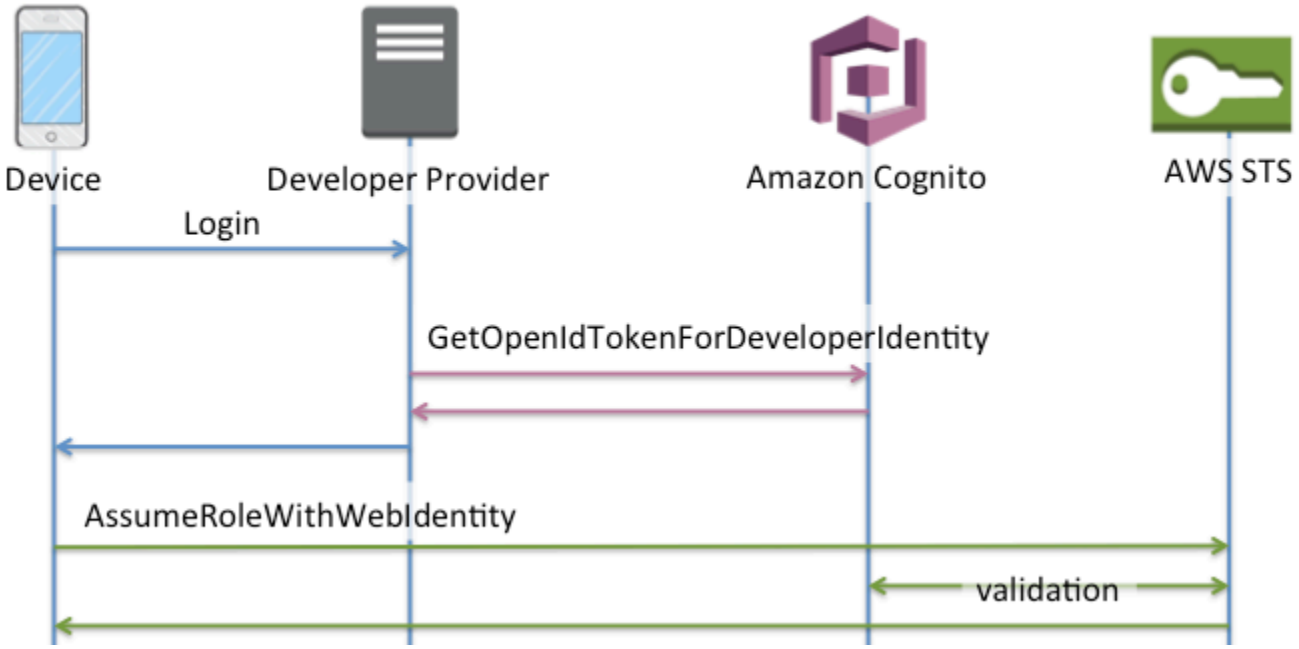
개발자 공급자를 통한 향상된 인증의 작업 순서

1. 개발자 공급자를 통해 로그인(Amazon Cognito에 속하지 않는 코드)
2. 사용자 로그인 검증(Amazon Cognito에 속하지 않는 코드)
3. GetOpenIdTokenForDeveloperIdentity
4. GetCredentialsForIdentity



개발자 공급자를 사용한 기본 인증의 작업 순서

1. ID 풀 외부에서 로직을 구현하여 로그인하고 개발자-공급자 식별자를 생성합니다.
2. 저장된 서버 측 AWS 자격 증명을 검색합니다.
3. 승인된 AWS 자격 증명으로 서명된 GetOpenIdTokenForDeveloperIdentity API 요청에서 개발자 공급자 식별자를 전송합니다.
4. AssumeRoleWithWebIdentity를 사용하여 애플리케이션 자격 증명을 요청합니다.



어떤 인증 흐름을 구현해야 하나요?

향상된 흐름은 개발자의 노력이 가장 적게 들면서 가장 안전한 선택입니다.

- 향상된 흐름은 API 요청의 복잡성, 크기 및 속도를 줄입니다.
- 애플리케이션은 AWS STS에 추가 API 요청을 할 필요가 없습니다.
- ID 풀은 사용자가 받아야 하는 IAM 역할 자격 증명에 대해 사용자를 평가합니다. 클라이언트에서 역할 선택을 위해 로직을 포함할 필요는 없습니다.

> **Important**
>
> 새 ID 풀을 생성할 때 모범 사례로 기본(클래식) 인증을 기본적으로 활성화하지 마세요. 기본 인증을 구현하려면 먼저 웹 ID에 대한 IAM 역할의 신뢰 관계를 평가합니다. 그런 다음 클라이언트에 역할 선택을 위한 로직을 빌드하고 사용자가 수정하지 못하도록 클라이언트를 보호하세요.

기본 인증 흐름은 IAM 역할 선택의 로직을 애플리케이션에 위임합니다. 이 흐름에서 Amazon Cognito는 사용자의 인증 또는 미인증 세션을 검증하고 자격 증명으로 교환할 수 있는 토큰을 발급합니다 AWS STS. 사용자는 기본 인증의 토큰을 ID 풀 및 `amr` 또는 인증됨/인증되지 않음 상태를 신뢰하는 모든 IAM 역할에 대해 교환할 수 있습니다.

마찬가지로 개발자 인증은 ID 제공업체 인증 검증의 지름길입니다. Amazon Cognito는 요청 내용에 대한 추가 검증 없이 GetOpenIdTokenForDeveloperIdentity 요청을 승인하는 AWS 자격 증명을 신뢰합니다. 사용자의 액세스로부터 개발자 인증을 승인하는 보안 암호를 보호합니다.

인증 흐름 API 작업 개요

GetId

GetId API 호출은 Amazon Cognito에서 새 ID를 설정하는 데 필요한 첫 번째 호출입니다.

인증되지 않은 액세스

Amazon Cognito는 애플리케이션에서 인증되지 않은 게스트 액세스를 허용할 수 있습니다. 자격 증명 풀에서 이 기능이 활성화되면 사용자는 언제든지 `GetId` API를 통해 새 자격 증명 ID를 요청할 수 있습니다. 애플리케이션에서는 다음에 Amazon Cognito를 호출하기 위해 이 자격 증명 ID를 캐시해야 합니다. 브라우저의 AWS Mobile SDKs 및 AWS SDK for JavaScript에는이 캐싱을 처리하는 자격 증명 공급자가 있습니다.

인증된 액세스

퍼블릭 로그인 공급자(Facebook, Google+, Login with Amazon 또는 Sign in with Apple)의 지원을 통해 애플리케이션을 구성한 경우 사용자는 해당 공급자에서 자신을 식별하는 토큰(OAuth 또는 OpenID Connect)을 제공할 수도 있습니다. `GetId` 호출에 사용된 경우 Amazon Cognito는 새 인증 자격 증명을 생성하거나 이미 특정 로그인에 연결되어 있는 자격 증명을 반환합니다. Amazon Cognito는 공급자로 토큰을 검증하고 다음을 확인하여 이를 수행합니다.

- 토큰이 유효하며 구성된 공급자로부터 받음
- 토큰이 만료되지 않음
- 토큰이 해당 공급자를 통해 생성된 애플리케이션 식별자와 일치함(예: Facebook 앱 ID)
- 토큰이 사용자 식별자와 일치함

GetCredentialsForIdentity

자격 증명 ID를 설정한 후 GetCredentialsForIdentity API를 호출할 수 있습니다. 이 작업은 GetOpenIdToken과 AssumeRoleWithWebIdentity를 차례대로 호출하는 것과 기능적으로 동일합니다.

사용자 대신 Amazon Cognito에서 `AssumeRoleWithWebIdentity`를 호출하기 위해 자격 증명 풀에 IAM 역할이 연결되어 있어야 합니다. Amazon Cognito 콘솔을 통하거나 SetIdentityPoolRoles 작업을 통해 수동으로 이를 수행할 수 있습니다.

GetOpenIdToken

자격 증명 ID를 설정한 후 GetOpenIdToken API 요청을 수행합니다. 첫 번째 요청 후 자격 증명 ID를 캐시하고 GetOpenIdToken을 사용하여 해당 자격 증명에 대한 후속 기본(클래식) 세션을 시작합니다.

GetOpenIdToken API 요청에 대한 응답은 Amazon Cognito가 생성하는 토큰입니다. 이 토큰을 AssumeRoleWithWebIdentity 요청의 WebIdentityToken 파라미터로 제출할 수 있습니다.

OpenID 토큰을 제출하기 전에 앱에서 토큰을 확인하세요. SDK의 OIDC 라이브러리 또는 aws-jwt-verify와 같은 라이브러리를 사용하여 Amazon Cognito가 토큰을 발급했는지 확인할 수 있습니다. OpenID 토큰의 서명 키 ID, 즉 kid는 Amazon Cognito 자격 증명 jwks_uri 문서†에 나열된 것 중 하나입니다. 이러한 키는 변경될 수 있습니다. Amazon Cognito 자격 증명 토큰을 확인하는 함수는 jwks_uri 문서에서 키 목록을 정기적으로 업데이트해야 합니다. Amazon Cognito는 jwks_uri 캐시 제어 응답 헤더에 새로 고침 기간을 설정하며, 현재 max-age인 30일로 설정되어 있습니다.

인증되지 않은 액세스

미인증 자격 증명에 대한 토큰을 얻으려면 자격 증명 ID 자체만 필요합니다. 인증된 자격 증명이나 비활성화된 자격 증명에 대한 미인증 토큰은 가져올 수 없습니다.

인증된 액세스

인증 자격 증명이 있는 경우 이미 해당 자격 증명과 연관된 로그인에 대해 유효한 토큰을 하나 이상 전달해야 합니다. GetOpenIdToken을 호출하는 동안 전달된 모든 토큰은 이전에 언급한 동일한 검증을 통과해야 합니다. 토큰 중 하나가 실패한 경우 전체 호출이 실패합니다. GetOpenIdToken 호출의 응답에도 자격 증명 ID가 포함됩니다. 이는 전달된 자격 증명 ID가 반환된 자격 증명 ID가 아닐 수도 있기 때문입니다.

로그인 연결

아직 자격 증명과 연관되지 않은 로그인에 대한 토큰을 제출하면 해당 로그인이 연관된 자격 증명에 '연결된' 것으로 간주됩니다. 퍼블릭 공급자당 로그인 하나만 연결할 수 있습니다. 퍼블릭 공급자에 로그인을 둘 이상 연결하려고 시도하면 ResourceConflictException 오류 응답이 발생합니다. 로그인이 기존 자격 증명에 연결만 되어 있는 경우 GetOpenIdToken에서 반환된 자격 증명 ID는 전달된 자격 증명 ID와 동일합니다.

자격 증명 병합

현재 지정된 자격 증명과 연결되지 않았지만 다른 자격 증명에 연결된 로그인에 대한 토큰을 전달하면 자격 증명 두 개가 병합됩니다. 병합되면 자격 증명 하나가 연관된 모든 로그인의 상위/소유자가 되며 다른 자격 증명이 비활성화됩니다. 이 경우 상위/소유자의 자격 증명 ID가 반환

됩니다. 이 값이 다른 경우 로컬 캐시를 업데이트해야 합니다. 브라우저의 AWS Mobile SDKs AWS SDK for JavaScript에 있는 공급자가이 작업을 자동으로 수행합니다.

GetOpenIdTokenForDeveloperIdentity

개발자 인증 ID를 사용할 경우 GetOpenIdTokenForDeveloperIdentity 작업은 디바이스에서 GetId 및 GetOpenIdToken의 사용을 대체합니다. 애플리케이션이 AWS 자격 증명으로 이 API 작업에 대한 요청에 서명하기 때문에 Amazon Cognito는 요청에 제공된 사용자 식별자가 유효하다고 신뢰합니다. 개발자 인증은 Amazon Cognito가 외부 공급자를 통해 수행하는 토큰 검증을 대체합니다.

이 API의 페이로드에는 `logins` 맵이 포함됩니다. 이 맵에는 개발자 공급자의 키와 시스템의 사용자에 대한 식별자 값이 포함되어야 합니다. 사용자 식별자가 기존 자격 증명에 아직 연결되지 않은 경우 Amazon Cognito는 새 자격 증명을 생성하고 새 자격 증명 ID와 해당 자격 증명에 대한 OpenID Connect 토큰을 반환합니다. 사용자 식별자가 이미 연결된 경우 Amazon Cognito는 기존 자격 증명 ID와 OpenID Connect 토큰을 반환합니다. 첫 번째 요청 후 개발자 자격 증명 ID를 캐시하고 `GetOpenIdTokenForDeveloperIdentity`을 사용하여 해당 자격 증명에 대한 후속 기본(클래식) 세션을 시작합니다.

`GetOpenIdTokenForDeveloperIdentity` API 요청에 대한 응답은 Amazon Cognito가 생성하는 토큰입니다. 이 토큰을 `AssumeRoleWithWebIdentity` 요청의 `WebIdentityToken` 파라미터로 제출할 수 있습니다.

OpenID Connect 토큰을 제출하기 전에 앱에서 토큰을 확인하세요. SDK의 OIDC 라이브러리 또는 aws-jwt-verify와 같은 라이브러리를 사용하여 Amazon Cognito가 토큰을 발급했는지 확인할 수 있습니다. OpenID 토큰의 서명 키 ID, 즉 `kid`는 Amazon Cognito 자격 증명 jwks_uri 문서†에 나열된 것 중 하나입니다. 이러한 키는 변경될 수 있습니다. Amazon Cognito 자격 증명 토큰을 확인하는 함수는 jwks_uri 문서에서 키 목록을 정기적으로 업데이트해야 합니다. Amazon Cognito는 jwks_uri `cache-control` 응답 헤더에 새로 고침 기간을 설정하며, 현재 `max-age`인 30일로 설정되어 있습니다.

로그인 연결

외부 공급자와 마찬가지로 아직 자격 증명에 연결되지 않은 추가 로그인을 제공하면 이러한 로그인과 해당 자격 증명이 암시적으로 연결됩니다. 외부 공급자 로그인을 자격 증명에 연결하면 사용자는 해당 공급자와 함께 외부 공급자 인증 흐름을 사용할 수 있습니다. 하지만 `GetId` 또는 `GetOpenIdToken`을 호출할 때는 로그인 맵에서 개발자 공급자 이름을 사용할 수 없습니다.

자격 증명 병합

개발자 인증 자격 증명을 사용하면 Amazon Cognito는 MergeDeveloperIdentities API를 통해 암시적 병합과 명시적 병합을 모두 지원합니다. 이러한 명시적 병합을 통해 시스템에서 사용자 식

별자가 있는 두 개의 자격 증명을 단일 자격 증명으로 표시할 수 있습니다. 소스 및 대상 사용자 식별자를 제공하기만 하면 Amazon Cognito에서는 이를 병합합니다. 다음에 이 두 사용자 식별자에 대한 OpenId Connect 토큰을 요청하면 동일한 자격 증명 ID가 반환됩니다.

AssumeRoleWithWebIdentity

OpenID Connect 토큰이 있으면 AssumeRoleWithWebIdentity API 요청을 통해 이를 임시 AWS 자격 증명으로 교환할 수 있습니다 AWS Security Token Service (AWS STS).

생성할 수 있는 자격 증명의 수에 대한 제한이 없으므로 사용자에게 부여되는 권한을 이해하는 것이 중요합니다. 애플리케이션에 대해 서로 다른 IAM 역할, 즉 인증되지 않은 사용자에 대한 역할과 인증된 사용자에 대한 역할을 설정합니다. Amazon Cognito 콘솔은 ID 풀을 처음 설정할 때 기본 역할을 생성할 수 있습니다. 이러한 역할에는 사실상 권한이 부여되지 않습니다. 필요에 맞게 수정합니다.

역할 트러스트 및 권한에 대해 자세히 알아보세요.

† 기본 Amazon Cognito 자격 증명 jwks_uri 문서에는 대부분의 AWS 리전에 있는 자격 증명 풀의 토큰을 서명하는 키에 대한 정보가 포함되어 있습니다. 다음 리전에는 서로 다른 jwks_uri 문서가 있습니다.

Amazon Cognito Identity JSON web key URIs in other AWS 리전

AWS 리전	jwks_uri 문서의 경로
AWS GovCloud (US-West)	`https://cognito-identity.us-gov-west-1.amazonaws.com/.well-known/jwks_uri`
China (Beijing)	`https://cognito-identity.cn-north-1.amazonaws.com.cn/.well-known/jwks_uri`
Opt-in Regions like Europe (Milan) and Africa (Cape Town)	`https://cognito-identity.  #  #.amazonaws.com/.well-known/jwks_uri`

발급자(`iss`)가 제공한 jwks_uri를 외삽하거나 Amazon Cognito로부터 OpenID 토큰에서 jwks_uri를 외삽할 수도 있습니다. OIDC 표준 검색 엔드포인트 `<issuer>/.well-known/openid-configuration`는 토큰의 jwks_uri 경로를 나열합니다.

IAM 역할

자격 증명 풀을 생성하는 동안 사용자가 맡을 IAM 역할을 업데이트하라는 메시지가 표시됩니다. 사용자가 앱에 로그인할 때 Amazon Cognito는 사용자에 대해 임시 AWS 자격 증명을 생성합니다. IAM 역할은 이런 식으로 작동합니다. 이러한 임시 자격 증명은 특정 IAM 역할에 연결됩니다. IAM 역할을 사용하면 AWS 리소스에 액세스할 수 있는 일련의 권한을 정의할 수 있습니다.

인증 및 미인증 사용자에 대해 기본 IAM 역할을 지정할 수 있습니다. 또한 사용자의 ID 토큰에 있는 클레임에 따라 각 사용자에 대한 역할을 선택하는 규칙을 정의할 수 있습니다. 자세한 내용은 역할 기반 액세스 제어 사용 섹션을 참조하세요.

기본적으로 Amazon Cognito 콘솔은 Amazon Mobile Analytics 및 Amazon Cognito Sync에 대한 액세스 권한을 제공하는 IAM 역할을 생성합니다. 또는 기존 IAM 역할을 사용하도록 선택할 수 있습니다.

다른 서비스에 대한 액세스를 허용하거나 제한하도록 IAM 역할을 수정합니다. 이렇게 하려면 IAM 콘솔에 로그인합니다. 그런 다음 역할(Roles)을 선택하고 역할을 선택합니다. 선택한 역할에 연결된 정책은 권한(Permissions) 탭에 나열됩니다. 해당하는 정책 관리(Manage Policy) 링크를 선택하여 액세스 정책을 사용자 지정할 수 있습니다. 정책 사용 및 정의에 대한 자세한 내용은 IAM 정책 개요를 참조하세요.

> **Note**
>
> 가장 좋은 방법은 최소 권한 부여 원칙을 따르는 정책을 정의하는 것입니다. 다시 말해, 해당 정책은 사용자가 작업을 수행하는 데 필요한 권한만을 포함합니다. 자세한 내용은 IAM 사용 설명서에서 최소 권한 부여를 참조하세요.
>
> 인증되지 않은 자격 증명은 앱에 로그인하지 않은 사용자가 수임합니다. 일반적으로, 인증되지 않은 자격 증명에 할당한 권한은 인증된 자격 증명의 권한보다 더 제한적이어야 합니다.

주제

- 트러스트 정책 설정
- 액세스 정책
- 역할 트러스트 및 권한

트러스트 정책 설정

Amazon Cognito는 IAM 역할을 사용하여 애플리케이션의 사용자에 대한 임시 자격 증명을 생성합니다. 권한에 대한 액세스는 역할의 신뢰 관계에 의해 제어됩니다. 역할 트러스트 및 권한 섹션에 대해 자세히 알아봅니다. Amazon Cognito는 AWS STS 및 자격 증명 풀 IdPs.

에 표시되는 토큰 AWS STS 은 사용자 풀, 소셜 또는 OIDC 공급자 토큰 또는 SAML 어설션을 자체 토큰으로 변환하는 자격 증명 풀에 의해 생성됩니다. 자격 증명 풀 토큰에는 자격 증명 풀 ID인 aud 클레임이 포함되어 있습니다.

IAM 역할 신뢰 정책의 `Principal`가 `cognito-identity.amazonaws.com`과 같은 ID 풀 서비스 보안 주체인 경우 모든 ID 풀이 역할을 수임하도록 허용하는 역할 신뢰 정책을 생성하거나 수정할 수 없습니다. ID 풀 보안 주체의 경우 `Action` 요소에 `cognito-identity.amazonaws.com:aud`과 같은 조건 키로 지정된 대로 ID 풀에서만 `AssumeRoleWithWebIndentity`를 수행해야 하는 `Condition`이 있어야 합니다. 다른 조건 키를 사용할 수 있지만 `aud`는 필수입니다. 이 유형의 조건 없이 역할 신뢰 정책을 저장하려고 하면 IAM에서 오류를 반환합니다.

OIDC(웹 자격 증명) 페더레이션 키에 대한 자세한 내용은 AWS OIDC 페더레이션에 사용 가능한 키를 참조하세요.

다음은 Amazon Cognito에 사용할 수 있는 OIDC 페더레이션 조건 키입니다.

`cognito-identity.amazonaws.com:aud`

역할을 하나 이상의 ID 풀에서 작업으로 제한합니다. Amazon Cognito는 ID 풀 토큰의 aud 클레임에 있는 소스 ID 풀을 나타냅니다.

`cognito-identity.amazonaws.com:amr`

역할을 `authenticated` 또는 `unauthenticated`(게스트) 사용자로 제한합니다. Amazon Cognito는 ID 풀 토큰의 `amr` 클레임에 인증 상태를 나타냅니다.

`cognito-identity.amazonaws.com:sub`

UUID를 통해 역할을 한 명 이상의 사용자로 제한합니다. 이 UUID는 ID 풀에 있는 사용자의 자격 증명 ID입니다. 이 값은 사용자의 원래 ID 제공업체의 `sub` 값이 아닙니다. Amazon Cognito는 이 UUID를 ID 풀 토큰의 `sub` 클레임에 나타냅니다.

다음 예제 역할 신뢰 정책은 페더레이션 서비스 보안 주체가 AWS STS API를 호출`cognito-identity.amazonaws.com`하도록 허용합니다`AssumeRoleWithWebIdentity`. API 요청의 자격 증명 풀 토큰에 다음과 같은 클레임이 있는 경우에만 요청이 성공합니다.

1. 자격 증명 풀 ID us-west-2:abcdefg-1234-5678-910a-0e8443553f95에 대한 aud 클레임.
2. 사용자가 로그인했지만 게스트 사용자가 아닌 경우에 추가되는 authenticated에 대한 amr 클레임.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "cognito-identity.amazonaws.com"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "cognito-identity.amazonaws.com:aud": "us-
west-2:abcdefg-1234-5678-910a-0e8443553f95"
                },
                "ForAnyValue:StringLike": {
                    "cognito-identity.amazonaws.com:amr": "authenticated"
                }
            }
        }
    ]
}
```

기본(클래식) 인증의 IAM 역할에 대한 신뢰 정책

요약

ID 풀은 대상 역할의 신뢰 정책에 aud 조건이 포함된 경우에만 기본 인증 흐름에서 사용자를 대신하여 역할을 맡을 수 있습니다.

기본 인증은 향상된 인증과 마찬가지로 안전하지 않은 역할 신뢰 정책에 대한 제한이 있습니다. 지원되는 ID 풀을 aud 조건으로 제한하지 않는 역할 신뢰 정책은 저장할 수 없습니다. 이 제한은 서비스가 시작될 때 적용되지 않았습니다. 이 요구 사항을 적용하기 전에 추가 보안 조건이 없는 역할 신뢰 정책을

생성할 수 있습니다. 이 요구 사항을 적용한 후는 조건으로 보호되지 않는 역할을 수임하도록 웹 자격 증명을 AWS STS 허용하지만 이러한 조건을 도입하지 않으면 해당 역할을 수정할 수 없습니다.

향상된 흐름 인증을 사용하려면 IAM 역할이 ID 풀과 동일한 AWS 계정 에 있어야 합니다. 그러나 애플리케이션이 `AssumeRoleWithWebIdentity` 요청을 구성하는 기본 인증에서는 애플리케이션이 다른 계정에서 역할을 맡도록 요청할 수 있습니다. 대상 역할에 aud 조건을 적용하지 않는 레거시 신뢰 정책이 있는 경우 교차 계정 수임 역할 요청이 실패합니다.

자격 증명 풀이 자격 증명에 대해 발급하는 토큰에는 자격 증명 풀 AWS 계정 의 출처에 대한 정보가 포함됩니다. AssumeRoleWithWebIdentity API 요청에 자격 증명 풀 토큰을 제시하면는 발신 자격 증명 풀이 AWS 계정 IAM 역할과 동일한에 있는지 AWS STS 확인합니다. 에서 요청이 교차 계정이라고 AWS STS 판단하면 역할 신뢰 정책에 aud 조건이 있는지 확인합니다. 역할 신뢰 정책에 해당 조건이 없으면 assume-role 호출은 실패합니다. 요청이 교차 계정이 아닌 경우이 제한을 적용 AWS STS 하지 않습니다. 가장 좋은 방법은 항상 이 유형의 조건을 ID 풀 역할의 신뢰 정책에 적용하는 것입니다.

다음은 여러 ID 풀을 사용한 기본 인증을 위한 IAM 역할의 최소 요구 사항을 충족하는 신뢰 정책의 예입니다. 가장 좋은 방법은 `"cognito-identity.amazonaws.com:amr": "authenticated"` 조건이 있는 인증된 ID만 허용하는 것입니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "cognito-identity.amazonaws.com"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "cognito-identity.amazonaws.com:aud": [
                        "us-west-2:abcdefg-1234-5678-910a-0e8443553f95",
                        "us-west-2:hijklmo-5678-9101-112b-0e4221776g96",
                        "us-west-2:pqrstuv-9101-1121-314c-0e2110887h97"
                    ]
                }
            }
        }
    ]
```

```
}
```

추가 신뢰 정책 조건

다음 신뢰 정책 조건을 사용하여 IAM 역할을 맡을 수 있는 소스 ID 풀, ID 및 제공업체를 정의할 수 있습니다.

> **Note**
>
> 향상된 인증 흐름에서 ID 풀이 수임하는 IAM 역할에 대한 신뢰 정책에서 aws:SourceIp 조건 키를 구현하지 마세요. 향상된 흐름은 애플리케이션을 대신하여 AssumeRoleWithWebIdentity 요청을 생성하기 때문에 요청의 소스 IP는 애플리케이션 클라이언트의 IP가 아니며 조건이 충족되지 않습니다. 네트워크 기반 조건 키는 향상된 흐름의 서비스 측 기능이 부족한 기본 흐름을 통해서만 ID 풀이 맡는 역할에 유효합니다.

자격 증명 풀에서 역할 재사용

여러 자격 증명 풀에서 역할을 재사용하려면 역할이 공통 권한 집합을 공유하므로 다음과 같이 여러 자격 증명 풀을 포함할 수 있습니다.

```
"StringEquals": {
    "cognito-identity.amazonaws.com:aud": [
        "us-east-1:12345678-abcd-abcd-abcd-123456790ab",
        "us-east-1:98765432-dcba-dcba-dcba-123456790ab"
    ]
}
```

특정 자격 증명으로 액세스 제한

특정 앱 사용자 집합에 제한된 정책을 생성하려면 `cognito-identity.amazonaws.com:sub` 값을 선택하세요.

```
"StringEquals": {
    "cognito-identity.amazonaws.com:aud": "us-east-1:12345678-abcd-abcd-
abcd-123456790ab",
    "cognito-identity.amazonaws.com:sub": [
        "us-east-1:12345678-1234-1234-1234-123456790ab",
        "us-east-1:98765432-1234-1234-1243-123456790ab"
```

```
    ]
}
```

특정 공급자로 액세스 제한

특정 공급자(자체 로그인 공급자)로 로그인한 사용자로 제한된 정책을 생성하려면 값을 확인하세요`cognito-identity.amazonaws.com:amr`.

```
"ForAnyValue:StringLike": {
    "cognito-identity.amazonaws.com:amr": "login.myprovider.myapp"
}
```

예를 들어, Facebook만 신뢰하는 앱에는 다음 amr 절이 있습니다.

```
"ForAnyValue:StringLike": {
    "cognito-identity.amazonaws.com:amr": "graph.facebook.com"
}
```

액세스 정책

역할에 연결하는 권한은 해당 역할을 맡은 모든 사용자에게 적용됩니다. 사용자의 액세스를 분할하려면 정책 조건 및 변수를 사용해야 합니다. 자세한 내용은 IAM 정책 요소: 변수 및 태그 단원을 참조하세요. sub 조건을 사용하여 액세스 정책에서 Amazon Cognito ID에 대한 작업을 제한할 수 있습니다. 이 옵션은 주의해서 사용해야 하며, 일관된 사용자 ID가 부족한 미인증 ID의 경우에는 더욱 주의해야 합니다. Amazon Cognito와의 웹 페더레이션을 위한 IAM 정책 변수에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 IAM 및 AWS STS 조건 컨텍스트 키를 참조하세요.

보안 강화를 위해, Amazon Cognito는 `GetCredentialsForIdentity`를 사용하여 향상된 흐름에서 미인증 사용자에 할당된 보안 인증 정보에 범위 축소 정책을 적용합니다. 범위 축소 정책은 미인증 역할에 적용하는 IAM 정책에 인라인 세션 정책 및 AWS 관리형 세션 정책을 추가합니다. 역할에 대한 IAM 정책과 세션 정책 모두에서 액세스 권한을 부여해야 하므로, 범위 축소 정책은 다음 목록에 있는 서비스 이외의 서비스에 대한 사용자의 액세스를 제한합니다.

> **Note**
>
> 기본(클래식) 흐름에서는 자체 AssumeRoleWithWebIdentity API 요청을 하고, 이러한 제한을 요청에 적용할 수 있습니다. 보안 모범 사례는 미인증 사용자에게 이 범위 제한 정책을 초과하는 권한을 할당하지 않는 것입니다.

또한 Amazon Cognito는 인증된 사용자와 인증되지 않은 사용자가 Amazon Cognito 자격 증명 풀 및 Amazon Cognito Sync에 API를 요청하는 것을 방지합니다. 다른 에서는 웹 자격 증명의 서비스 액세스를 제한할 AWS 서비스 수 있습니다.

향상된 흐름으로 요청을 하는 데 성공하면, Amazon Cognito가 백그라운드에서 `AssumeRoleWithWebIdentity` API 요청을 합니다. 이 요청의 파라미터 중에서 Amazon Cognito에는 다음이 포함됩니다.

1. 사용자의 자격 증명 ID입니다.
2. 사용자가 수임할 IAM 역할의 ARN입니다.
3. 인라인 세션 정책을 추가하는`policy` 파라미터입니다.
4. Amazon CloudWatch에서 추가 권한을 부여하는AWS 관리형 정책이 값인 `PolicyArns.member.N` 파라미터입니다.

미인증 사용자가 액세스할 수 있는 서비스

향상된 흐름을 사용하는 경우, Amazon Cognito가 사용자 세션에 적용하는 범위 축소 정책에 따라 사용자는 다음 표에 나열된 서비스 이외의 서비스를 사용할 수 없습니다. 일부 서비스의 경우, 특정 작업만 허용됩니다.

카테고리	서비스
분석	Amazon Data Firehose Amazon Managed Service for Apache Flink
애플리케이션 통합	Amazon Simple Queue Service
AR 및 VR	Amazon Sumerian[1]
비즈니스 애플리케이션	Amazon Mobile Analytics Amazon Simple Email Service
컴퓨팅	AWS Lambda
암호화 및 PKI	AWS Key Management Service 1
Database	Amazon DynamoDB

카테고리	서비스
	Amazon SimpleDB
프런트 엔드 웹 및 모바일	AWS AppSync Amazon Location Service Amazon Simple Notification Service Amazon Pinpoint Amazon Location Service
게임 개발	Amazon GameLift Servers
사물 인터넷(IoT)	AWS IoT
기계 학습	Amazon CodeWhisperer Amazon Comprehend Amazon Lex Amazon Machine Learning Amazon Personalize Amazon Polly Amazon Rekognition Amazon SageMaker AI[1] Amazon Textract[1] Amazon Transcribe Amazon Translate
관리 및 거버넌스	Amazon CloudWatch Amazon CloudWatch Logs

카테고리	서비스
네트워킹 및 콘텐츠 전송	Amazon API Gateway
보안, 자격 증명 및 규정 준수	Amazon Cognito 사용자 풀
스토리지	Amazon Simple Storage Service

1 다음 표의 AWS 서비스 의 경우 인라인 정책은 작업의 하위 집합을 부여합니다. 표에는 각 항목에서 사용할 수 있는 작업이 표시됩니다.

AWS 서비스	미인증 향상된 흐름 사용자의 최대 권한

AWS Key Management Service	`Encrypt`
	`Decrypt`
	`ReEncryptTo`
	`ReEncryptFrom`
	`GenerateDataKey`
	`GenerateDataKeyPair`
	`GenerateDataKeyPair`
	`GenerateDataKeyPairWithoutPlaintext`
	`GenerateDataKeyWithoutPlaintext`

Amazon SageMaker AI	`InvokeEndpoint`
Amazon Textract	`DetectDocumentText` `AnalyzeDocument`
Amazon Sumerian	`View*`
Amazon Location Service	`SearchPlaceIndex*`

AWS 서비스	미인증 향상된 흐름 사용자의 최대 권한

	`GetPlace`
	
	`CalculateRoute*`
	
	`*Geofence`
	
	`*Geofences`
	
	`*DevicePosition*`

이 목록을 AWS 서비스 넘어에 대한 액세스 권한을 부여하려면 자격 증명 풀에서 기본(클래식) 인증 흐름을 활성화합니다. 인증되지 않은 사용자의 IAM 역할에 할당된 정책에서 AWS 서비스 허용하는의 `NotAuthorizedException` 오류가 사용자에게 표시되는 경우 사용 사례에서 해당 서비스를 제거할 수 있는지 평가합니다. 제거할 수 없다면 기본 흐름으로 전환하세요.

게스트 사용자를 위한 인라인 세션 정책

Amazon Cognito는 먼저 IAM 자격 증명 요청에 인라인 정책을 적용합니다. 인라인 세션 정책은 사용자의 유효 권한에 다음 목록에 나오지 않는 AWS 서비스 에 대한 액세스를 포함하지 못하게 합니다. 또한 사용자의 IAM 역할에 적용하는 정책 AWS 서비스 에서 이러한에 권한을 부여해야 합니다. 수임하는 역할 세션에 대한 사용자의 유효 권한은 해당 역할에 할당된 정책과 관련 세션 정책의 교차점입니다. 자세한 정보는 AWS Identity and Access Management 사용 설명서의 세션 정책을 참조하세요.

Amazon Cognito는 기본적으로 활성화된 AWS 리전 의 사용자 세션에 다음과 같은 인라인 정책을 추가합니다. 인라인 정책 및 기타 세션 정책의 순 효과에 대한 개요는 미인증 사용자가 액세스할 수 있는 서비스 섹션을 참조하세요.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:*",
                "logs:*",
                "dynamodb:*",
```

```
                "kinesis:*",
                "mobileanalytics:*",
                "s3:*",
                "ses:*",
                "sns:*",
                "sqs:*",
                "lambda:*",
                "machinelearning:*",
                "execute-api:*",
                "iot:*",
                "gamelift:*",
                "cognito-identity:*",
                "cognito-idp:*",
                "lex:*",
                "polly:*",
                "comprehend:*",
                "translate:*",
                "transcribe:*",
                "rekognition:*",
                "mobiletargeting:*",
                "firehose:*",
                "appsync:*",
                "personalize:*",
                "sagemaker:InvokeEndpoint",
                "cognito-sync:*",
                "codewhisperer:*",
                "textract:DetectDocumentText",
                "textract:AnalyzeDocument",
                "sdb:*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

다른 모든 리전의 경우, 인라인 범위 축소 정책에는 다음 `Action` 문을 제외하고 기본 리전에 나열된 모든 항목이 포함됩니다.

```
            "cognito-sync:*",
            "sumerian:View*",
```

```
                "codewhisperer:*",
                "textract:DetectDocumentText",
                "textract:AnalyzeDocument",
                "sdb:*"
```

게스트에 대한 AWS 관리형 세션 정책

또한 Amazon Cognito는 AWS 관리형 정책을 인증되지 않은 게스트의 향상된 흐름 세션에 세션 정책으로 적용합니다. 이 정책은 AmazonCognitoUnAuthedIdentitiesSessionPolicy 정책으로 인증되지 않은 사용자의 권한 범위를 제한합니다.

인증되지 않은 IAM 역할에 연결하는 정책에서도 이 권한을 부여해야 합니다. 수임하는 역할 세션에 대한 사용자의 유효 권한은 해당 역할에 할당된 IAM 정책과 관련 세션 정책의 교차점입니다. 자세한 정보는 AWS Identity and Access Management 사용 설명서의 세션 정책을 참조하세요.

이 AWS 관리형 정책 및 기타 세션 정책의 순 효과에 대한 개요는 섹션을 참조하세요미인증 사용자가 액세스할 수 있는 서비스.

AmazonCognitoUnAuthedIdentitiesSessionPolicy 관리형 정책에는 다음 권한이 있습니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "rum:PutRumEvents",
            "polly:*",
            "comprehend:*",
            "translate:*",
            "transcribe:*",
            "rekognition:*",
            "mobiletargeting:*",
            "firehose:*",
            "personalize:*",
            "sagemaker:InvokeEndpoint",
            "geo:GetMap*",
            "geo:SearchPlaceIndex*",
            "geo:GetPlace",
            "geo:CalculateRoute*",
```

```
            "geo:*Geofence",
            "geo:*Geofences",
            "geo:*DevicePosition*",
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncryptTo",
            "kms:ReEncryptFrom",
            "kms:GenerateDataKey",
            "kms:GenerateDataKeyPair",
            "kms:GenerateDataKeyPairWithoutPlaintext",
            "kms:GenerateDataKeyWithoutPlaintext"
        ],
        "Resource": "*"
    }]
}
```

액세스 정책 예제

이 섹션에서는 특정 작업을 완료하는 데 필요한 최소 권한만 사용자에게 부여하는 Amazon Cognito 액세스 정책의 예시를 찾을 수 있습니다. 가능한 경우 정책 변수를 사용하여 지정된 자격 증명 ID에 대한 권한을 한층 더 제한할 수 있습니다. 예를 들면, ${cognito-identity.amazonaws.com:sub}를 사용합니다. 자세한 내용은 AWS 모바일 블로그에서 Amazon Cognito 인증 이해 3부: 역할 및 정책을 참조하세요.

> **Note**
>
> 보안 모범 사례로, 정책은 사용자가 작업을 수행하는 데 필요한 권한만을 포함해야 합니다. 다시 말해서 가능한 경우 항상 액세스 범위를 객체에 대한 개별 자격 증명으로 조정해야 합니다.

Amazon S3에서 단일 객체에 대한 읽기 권한을 자격 증명에 부여

다음 액세스 정책은 지정된 S3 버킷에서 단일 객체를 검색할 수 있는 읽기 권한을 자격 증명에게 부여합니다.

JSON

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

```
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket/assets/my_picture.jpg"]
    }
  ]
}
```

Amazon S3에서 자격 증명 관련 경로에 대한 읽기 및 쓰기 권한을 모두 자격 증명에 부여

다음 액세스 정책은 접두사를 `${cognito-identity.amazonaws.com:sub}` 변수에 매핑하여 S3 버킷에서 특정 접두사 "folder"에 액세스할 수 있는 읽기 및 쓰기 권한을 부여합니다.

이 정책을 사용하면 `${cognito-identity.amazonaws.com:sub}`를 통해 삽입된 `us-east-1:12345678-1234-1234-1234-123456790ab`와 같은 자격 증명은 객체를 `arn:aws:s3:::amzn-s3-demo-bucket/us-east-1:12345678-1234-1234-1234-123456790ab`로 가져오고 넣고 나열할 수 있습니다. 하지만 `arn:aws:s3:::amzn-s3-demo-bucket`의 다른 객체에 대한 액세스는 이 자격 증명에게 부여되지 않습니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket"],
      "Condition": {"StringLike": {"s3:prefix": ["${cognito-
identity.amazonaws.com:sub}/*"]}}
    },
    {
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket/${cognito-
identity.amazonaws.com:sub}/*"]
```

```
    }
  ]
}
```

Amazon S3 Access Grants를 사용하면 유사한 액세스 모델을 얻을 수 있습니다.

자격 증명에 Amazon DynamoDB에 대한 세분화된 액세스 권한 할당

다음 액세스 정책은 Amazon Cognito 환경 변수를 사용하여 Amazon DynamoDB 리소스에 대한 세분화된 액세스 제어를 제공합니다. 이러한 변수는 자격 증명 ID를 기준으로 DynamoDB의 항목에 대한 액세스 권한을 부여합니다. 자세한 내용은 Amazon DynamoDB 개발자 가이드에서 IAM 정책 조건을 사용하여 세부적인 액세스 제어 구현을 참조하세요.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:BatchGetItem",
        "dynamodb:Query",
        "dynamodb:PutItem",
        "dynamodb:UpdateItem",
        "dynamodb:DeleteItem",
        "dynamodb:BatchWriteItem"
      ],
      "Resource": [
        "arn:aws:dynamodb:us-west-2:123456789012:table/MyTable"
      ],
      "Condition": {
        "ForAllValues:StringEquals": {
          "dynamodb:LeadingKeys":  ["${cognito-identity.amazonaws.com:sub}"]
        }
      }
    }
  ]
}
```

Lambda 함수를 호출할 수 있는 권한을 자격 증명에 부여

다음 액세스 정책은 Lambda 함수를 호출할 수 있는 권한을 자격 증명에 부여합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
         {
             "Effect": "Allow",
             "Action": "lambda:InvokeFunction",
             "Resource": [
                 "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
             ]
         }
     ]
 }
```

레코드를 Amazon Kinesis Data Streams에 게시할 수 있는 권한을 자격 증명에 부여

다음 액세스 정책은 자격 증명이 Kinesis Data Streams 중 하나에 대해 `PutRecord` 작업을 사용하도록 허용합니다. 계정의 모든 스트림에 데이터 레코드를 추가해야 하는 사용자에게 이 정책을 적용할 수 있습니다. 자세한 내용은 Amazon Kinesis Data Streams 개발자 가이드에서 IAM을 사용하여 Amazon Kinesis Data Streams 리소스에 대한 액세스 제어를 참조하세요.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "kinesis:PutRecord",
            "Resource": [
                "arn:aws:kinesis:us-east-1:111122223333:stream/stream1"
            ]
        }
    ]
}
```

Amazon Cognito Sync 스토어에 있는 자신의 데이터에 액세스할 수 있는 권한을 자격 증명에 부여

다음 액세스 정책은 Amazon Cognito Sync 스토어에 있는 자신의 데이터에만 액세스할 수 있는 권한을 자격 증명에 부여합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[{
      "Effect":"Allow",
      "Action":"cognito-sync:*",
      "Resource":["arn:aws:cognito-sync:us-east-1:123456789012:identitypool/
${cognito-identity.amazonaws.com:aud}/identity/${cognito-
identity.amazonaws.com:sub}/*"]
      }]
  }
```

역할 트러스트 및 권한

이러한 역할이 서로 다른 방식은 신뢰 관계에 있습니다. 다음은 인증되지 않은 역할에 대한 신뢰 정책의 예입니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Federated": "cognito-identity.amazonaws.com"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "cognito-identity.amazonaws.com:aud": "us-east-1:12345678-corner-
cafe-123456790ab"
        },
```

```
        "ForAnyValue:StringLike": {
          "cognito-identity.amazonaws.com:amr": "unauthenticated"
        }
      }
    }
  ]
}
```

이 정책은 cognito-identity.amazonaws.com의 페더레이션 사용자(OpenID Connect 토큰의 발급자)에게 이 역할을 맡을 수 있는 권한을 부여합니다. 또한 이 정책은 토큰의 aud(이 경우 자격 증명 풀 ID)가 자격 증명 풀과 일치하도록 제한합니다. 마지막으로, 정책은 unauthenticated 값을 지닌 Amazon Cognito GetOpenIdToken API 작업에서 발행한 토큰의 다중 값 amr 클레임의 배열 멤버 중 하나를 지정합니다

Amazon Cognito에서 토큰을 생성하면 해당 토큰의 amr을 unauthenticated 또는 authenticated로 설정합니다. amr이 authenticated이면 토큰에 인증 중에 사용되는 모든 공급자가 포함됩니다. 즉, 다음과 같이 amr 조건을 변경하여 Facebook을 통해 로그인한 사용자만 신뢰하는 역할을 생성할 수 있습니다.

```
"ForAnyValue:StringLike": {
  "cognito-identity.amazonaws.com:amr": "graph.facebook.com"
}
```

역할에 대한 신뢰 관계를 변경할 때 또는 자격 증명 풀에 대해 역할을 사용하려고 시도할 때 주의를 기울이세요. 역할이 자격 증명 풀을 신뢰하도록 올바르게 구성되지 않은 경우 다음과 같은 STS 예외가 발생합니다.

```
AccessDenied -- Not authorized to perform sts:AssumeRoleWithWebIdentity
```

이러한 메시지가 표시되면 자격 증명 풀과 인증 유형에서 적절한 역할을 사용하는지 확인합니다.

Amazon Cognito ID 풀을 위한 보안 모범 사례

Amazon Cognito 자격 증명 풀은 애플리케이션에 대한 임시 AWS 자격 증명을 제공합니다. AWS 계정에는 종종 애플리케이션 사용자에게 필요한 리소스와 프라이빗 백엔드 리소스가 모두 포함되어 있습니다. AWS 자격 증명을 구성하는 IAM 역할 및 정책은 이러한 리소스에 대한 액세스 권한을 부여할 수 있습니다.

ID 풀 구성의 주요 모범 사례는 애플리케이션이 과도하거나 의도하지 않은 권한 없이 작업을 완료할 수 있도록 하는 것입니다. 보안 구성 오류를 방지하려면 프로덕션에 릴리스하려는 각 애플리케이션을 시작하기 전에 이러한 권장 사항을 검토하세요.

주제

- IAM 구성 모범 사례
- ID 풀 구성 모범 사례

IAM 구성 모범 사례

게스트 또는 인증된 사용자가 애플리케이션에서 ID 풀 자격 증명이 필요한 세션을 시작하면 애플리케이션이 IAM 역할에 대한 임시 AWS 자격 증명을 검색합니다. 자격 증명은 기본 역할, ID 풀 구성의 규칙에 의해 선택된 역할 또는 앱에서 선택된 사용자 지정 역할에 대한 것일 수 있습니다. 각 역할에 할당된 권한을 통해 사용자는 AWS 리소스에 액세스할 수 있습니다.

일반적인 IAM 모범 사례에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 IAM 모범 사례를 참조하세요.

IAM 역할에서 신뢰 정책 조건 사용

IAM을 사용하려면 ID 풀의 역할에 하나 이상의 신뢰 정책 조건이 있어야 합니다. 예를 들어이 조건은 역할의 범위를 인증된 사용자로만 설정할 수 있습니다. AWS STS 또한 교차 계정 기본 인증 요청에는 `cognito-identity.amazonaws.com:aud` 및 라는 두 가지 특정 조건이 있어야 합니다`cognito-identity.amazonaws.com:amr`. 가장 좋은 방법은 ID 풀 서비스 보안 주체 `cognito-identity.amazonaws.com`을 신뢰하는 모든 IAM 역할에 이러한 두 조건을 모두 적용하는 것입니다.

- `cognito-identity.amazonaws.com:aud`: ID 풀 토큰의 aud 클레임은 신뢰할 수 있는 ID 풀 ID와 일치해야 합니다.
- `cognito-identity.amazonaws.com:amr`: ID 풀 토큰의 amr 클레임은 인증되거나 인증되지 않아야 합니다. 이 조건에서는 인증되지 않은 게스트 또는 인증된 사용자에 대해서만 역할에 대한 액세스를 예약할 수 있습니다. 이 조건의 값을 추가로 구체화하여 `graph.facebook.com`과 같은 특정 공급자의 사용자로 역할을 제한할 수 있습니다.

다음 예제 역할 신뢰 정책은 다음 조건에서 역할에 대한 액세스 권한을 부여합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Federated": "cognito-identity.amazonaws.com"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "cognito-identity.amazonaws.com:aud": "us-east-1:a1b2c3d4-5678-90ab-
cdef-EXAMPLE11111"
        },
        "ForAnyValue:StringLike": {
          "cognito-identity.amazonaws.com:amr": "authenticated"
        }
      }
    }
  ]
}
```

ID 풀과 관련된 요소

- `"Federated": "cognito-identity.amazonaws.com"`: 사용자는 ID 풀에서 와야 합니다.
- `"cognito-identity.amazonaws.com:aud": "us-east-1:a1b2c3d4-5678-90ab-cdef-example11111"`: 사용자는 특정 ID 풀 `us-east-1:a1b2c3d4-5678-90ab-cdef-example11111`에서 와야 합니다.
- `"cognito-identity.amazonaws.com:amr": "authenticated"`: 사용자를 인증해야 합니다. 게스트 사용자는 역할을 수임할 수 없습니다.

최소 권한 적용

인증된 액세스 또는 게스트 액세스에 대해 IAM 정책을 사용하여 권한을 설정할 때 특정 작업을 수행하는 데 필요한 특정 권한 또는 최소 권한만 부여합니다. 다음 예제 IAM 정책은 역할에 적용될 때 Amazon S3 버킷의 단일 이미지 파일에 대한 읽기 전용 액세스 권한을 부여합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::mybucket/assets/my_picture.jpg"]
    }
  ]
}
```

ID 풀 구성 모범 사례

자격 증명 풀에는 AWS 자격 증명 생성을 위한 유연한 옵션이 있습니다. 애플리케이션이 가장 안전한 방법으로 작동할 수 있는 경우 디자인 바로 가기를 사용하지 마세요.

게스트 액세스의 영향 이해

인증되지 않은 게스트 액세스를 사용하면 사용자가 로그인하기 전에 t사용자의 AWS 계정 에서 데이터를 검색할 수 있습니다. ID 풀 ID를 알고 있는 사람은 누구나 인증되지 않은 자격 증명을 요청할 수 있습니다. ID 풀 ID는 기밀 정보가 아닙니다. 게스트 액세스를 활성화하면 인증되지 않은 세션에 부여하는 AWS 권한을 모든 사용자가 사용할 수 있습니다.

모범 사례는 게스트 액세스를 비활성화한 상태로 두고 사용자가 인증한 후에만 필요한 리소스를 가져오는 것입니다. 애플리케이션에 로그인하기 전에 리소스에 대한 액세스 권한이 필요한 경우 다음 예방 조치를 취하세요.

- 인증되지 않은 역할에 적용되는 자동 제한 사항을 숙지합니다.
- 애플리케이션의 특정 요구 사항에 맞게 인증되지 않은 IAM 역할의 권한을 모니터링하고 조정합니다.
- 특정 리소스에 대한 액세스 권한을 부여합니다.
- 인증되지 않은 기본 IAM 역할의 신뢰 정책을 보호합니다.
- 인터넷에 있는 모든 사람에게 IAM 역할의 권한을 부여할 수 있다고 확신하는 경우에만 게스트 액세스를 활성화합니다.

기본적으로 향상된 인증 사용

기본(클래식) 인증을 통해 Amazon Cognito는 IAM 역할 선택을 앱에 위임합니다. 반면 향상된 흐름은 ID 풀의 중앙 집중식 로직을 사용하여 IAM 역할을 결정합니다. 또한 IAM 권한에 대한 상한을 설정하는 범위 축소 정책을 통해 인증되지 않은 ID에 대한 추가 보안을 제공합니다. 향상된 흐름은 개발자의 노력이 가장 적게 들면서 가장 안전한 선택입니다. 이러한 옵션에 대해 자세히 알아보려면 자격 증명 풀 인증 흐름 섹션을 참조하세요.

기본 흐름은 자격 증명에 대한 AWS STS API 요청의 역할 선택 및 조립에 들어가는 클라이언트 측 로직을 노출할 수 있습니다. 향상된 흐름은 ID 풀 자동화 뒤에 있는 로직 및 스태프 역할 요청을 모두 숨깁니다.

기본 인증을 구성할 때 IAM 역할 및 권한에 IAM 모범 사례를 적용합니다.

개발자 공급자를 안전하게 사용

개발자 인증 ID는 서버 측 애플리케이션을 위한 ID 풀의 기능입니다. 자격 증명 풀이 개발자 인증에 필요한 인증의 유일한 증거는 자격 증명 풀 개발자의 AWS 자격 증명입니다. ID 풀은 이 인증 흐름에 표시되는 개발자-제공자 식별자의 유효성에 대한 제한을 적용하지 않습니다.

개발자 공급자는 다음 조건에서만 구현하는 것이 가장 좋습니다.

- 개발자 인증 자격 증명 사용에 대한 책임을 생성하려면 개발자 공급자 이름과 식별자를 설계하여 인증 소스를 표시합니다. 예를 들어 `"Logins" : {"MyCorp provider" : "[provider application ID]"}`입니다.
- 수명이 긴 사용자 자격 증명을 사용하지 마세요. EC2 인스턴스 프로파일 및 Lambda 실행 역할과 같은 서비스 연결 역할이 있는 ID를 요청하도록 서버 측 클라이언트를 구성합니다.
- 동일한 ID 풀에서 내부 및 외부 신뢰 소스를 혼합하지 마세요. 개발자 공급자와 Single Sign-On(SSO) 공급자를 별도의 ID 풀에 추가합니다.

액세스 제어에 속성 사용

액세스 제어를 위한 속성은 ABAC(속성 기반 액세스 제어)의 Amazon Cognito 자격 증명 풀 구현입니다. IAM 정책을 사용하여 사용자 속성을 기반으로 Amazon Cognito 자격 증명 풀을 통한 AWS 리소스 액세스를 제어할 수 있습니다. 이러한 속성은 소셜 및 기업 자격 증명 공급자에서 가져올 수 있습니다. 공급자의 액세스 및 ID 토큰 또는 SAML 어설션 내의 속성을 IAM 권한 정책에서 참조되는 태그에 매핑할 수 있습니다.

Amazon Cognito 자격 증명 풀에서 기본 매핑을 선택하거나 사용자 지정 매핑을 생성할 수 있습니다. 기본 매핑을 사용하면 고정된 사용자 속성 세트를 기반으로 IAM 정책을 작성할 수 있습니다. 사용자 지정 매핑을 사용하면 IAM 권한 정책에서 참조되는 사용자 속성 세트를 선택할 수 있습니다. Amazon Cognito 콘솔의 속성 이름이 IAM 권한 정책에서 참조되는 태그인 보안 주체의 태그 키에 매핑됩니다.

예를 들어 무료 멤버십과 유료 멤버십이 있는 미디어 스트리밍 서비스를 보유하고 있다고 가정해 보겠습니다. 미디어 파일을 Amazon S3에 저장하고 무료 또는 프리미엄 태그로 태깅합니다. 액세스 제어를 위한 속성을 사용하여 사용자 프로필의 일부인 사용자 멤버십 수준에 따라 무료 및 유료 콘텐츠에 대한 액세스를 허용할 수 있습니다. IAM 권한 정책에 전달될 보안 주체의 태그 키에 멤버십 속성을 매핑할 수 있습니다. 이렇게 하면 단일 권한 정책을 생성하고 콘텐츠 파일의 멤버십 수준 및 태그 값에 따라 프리미엄 콘텐츠에 대한 액세스를 조건부로 허용할 수 있습니다.

주제

- Amazon Cognito 자격 증명 풀에서 액세스 제어에 속성 사용
- 액세스 제어에 속성 사용 정책 예
- 액세스 제어를 위한 속성 끄기(콘솔)
- 기본 공급자 매핑

속성을 사용하여 액세스를 제어하면 여러 가지 장점이 있습니다.

- 액세스 제어를 위한 속성을 사용하면 더 효율적으로 권한을 관리할 수 있습니다. 직무별로 여러 개의 정책을 만드는 대신 사용자 속성을 사용하는 기본 권한 정책을 만들 수 있습니다.
- 애플리케이션에 리소스 또는 사용자를 추가하거나 제거할 때마다 정책을 업데이트할 필요가 없습니다. 권한 정책은 일치하는 사용자 속성을 가진 사용자에게만 액세스 권한을 부여합니다. 예를 들어 사용자의 직책에 따라 특정 S3 버킷에 대한 액세스를 제어해야 할 수 있습니다. 이 경우 정의된 작책 내의 사용자만 이러한 파일에 액세스하도록 허용하는 권한 정책을 생성할 수 있습니다. 자세한 내용은 IAM 자습서: ABAC에 SAML 세션 태그 사용을 참조하세요.
- 해당 속성의 값에 따라 권한을 허용하거나 거부하는 정책에 속성을 보안 주체 태그로 전달할 수 있습니다.

Amazon Cognito 자격 증명 풀에서 액세스 제어에 속성 사용

액세스 제어에 속성을 사용하려면 먼저 다음 사전 요구 사항을 충족해야 합니다.

- AWS 계정
- 사용자 풀

- 자격 증명 풀
- SDK 설정
- 통합된 자격 증명 공급자
- 보안 인증

액세스 제어를 위한 속성을 사용하기 위해서는 데이터 세트 소스로 설정한 클레임이 선택한 태그 키의 값을 설정합니다. Amazon Cognito는 태그 키와 값을 사용자 세션에 적용합니다. IAM 정책은 `${aws:PrincipalTag/tagkey}` 조건을 통해 사용자의 액세스를 평가할 수 있습니다. IAM은 정책과 비교하여 사용자 태그의 값을 평가합니다.

보안 인증 정보를 사용자에게 전달할 IAM 역할을 준비해야 합니다. 이러한 역할의 신뢰 정책에서 Amazon Cognito가 사용자에 대한 역할을 맡을 수 있도록 허용해야 합니다. 액세스 제어 속성의 경우 Amazon Cognito가 사용자 임시 세션에 보안 주체 태그를 적용할 수 있도록 허용해야 합니다. AssumeRoleWithWebIdentity 작업을 통해 이러한 역할을 맡을 수 있는 권한을 부여합니다. 권한 전용 작업 `sts:TagSession`을 사용하여 사용자 세션에 태그를 지정할 수 있는 권한을 부여합니다. 자세한 내용은 AWS Identity and Access Management 사용 설명서의 AWS Security Token Service에서 세션 태그 전달을 참조하세요. Amazon Cognito 서비스 보안 주체 `cognito-identity.amazonaws.com`에 `sts:AssumeRoleWithWebIdentity` 및 `sts:TagSession` 권한을 부여하는 신뢰 정책의 예는 액세스 제어에 속성 사용 정책 예 섹션을 참조하세요.

> **Important**
>
> Amazon Cognito 사용자 풀의 사용자 지정 속성을 보안 주체 태그에 매핑하는 경우 최종 사용자가 해당 속성을 수정할 수 없는지 확인합니다. 사용자가 변경할 수 있는 속성은 IAM 정책에서 보안 경계 역할을 해서는 `UpdateUserAttributes` 안 됩니다. 사용자 수정을 방지하려면 다음 중 하나를 수행합니다.
>
> - 자격 증명 풀에 구성한 앱 클라이언트뿐만 아니라 사용자 풀에 있는 모든 앱 클라이언트의 WriteAttributes에서 속성을 제거합니다.
> - 속성을 사용자 풀 스키마에 추가할 `Mutable: false` 때 로 정의합니다.
>
> 사용자 속성은 사용자 풀의 모든 앱 클라이언트에서 공유됩니다. 사용자가 앱 클라이언트를 통해 속성을 변경하면 모든 앱 클라이언트의 해당 사용자의 ID 토큰에 새 값이 나타납니다. 앱 클라이언트`WriteAttributes`에를 지정하지 않으면 앱 클라이언트는 모든 표준 및 사용자 지정 속성에 대한 쓰기 액세스 권한을 갖습니다. 나중에 추가하는 앱 클라이언트를 포함하여 생성하는 각 앱 클라이언트`WriteAttributes`에 대해를 지정합니다.

사용자는 사용자 프로필이 생성된 후에는 변경할 수 없는 속성을 변경할 수 없습니다. 그러나 가입할 때 값을 설정할 수 있습니다. 사용자 풀이 셀프 서비스 가입을 허용하는 경우 모든 앱 클라이언트`WriteAttributes`의에서 속성을 제거합니다. 그런 다음, 다음 중 하나를 수행하세요.

- 를 사용하여 사용자를 생성할 때 값을 설정합니다`AdminCreateUser`.
- 예상치 못한 값이 포함된 가입 요청을 거부합니다. 자세한 내용은 the section called “사전 가입” 섹션을 참조하세요.

또한 보안 주체 태그에 매핑된 속성 값에 (별표) 또는 * (?물음표)와 같이 IAM 정책 평가에서 특별한 의미를 갖는 문자가 포함되어 있지 않은지 확인합니다. IAM 정책 `Resource` 요소 및 `StringLike` 조건 연산자에서 이러한 문자는 와일드카드로 작동합니다. 속성 값을 * 로 설정하고 보안 주체 태그에 매핑하면 `${aws:PrincipalTag/`*`tagkey`*`}` 평가에서 의도한 것보다 많은 리소스가 일치할 수 있습니다. 이렇게 하면 다른 테넌트 또는 프로젝트에 속한 리소스에 대한 액세스 권한을 부여할 수 있습니다. 애플리케이션 로직에서 속성 값을 검증합니다. 또는 애플리케이션이 보안 주체 태그로 전달하기 전에 사전 토큰 생성 Lambda 트리거를 사용하여 이러한 문자가 포함된 값을 거부하거나 삭제합니다.
애플리케이션에서 다른 목적으로 속성을 사용자가 작성할 수 있도록 유지해야 하는 경우 ID 토큰에 포함되기 전에 사전 토큰 생성 Lambda 트리거를 사용하여 클레임을 검증하거나 재정의합니다.

콘솔에서 액세스 제어를 위한 속성을 구성하려면

1. Amazon Cognito 콘솔에 로그인하고 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. ID 제공업체를 찾습니다. 편집할 ID 제공업체를 선택합니다. 새 IdP를 추가하려면 ID 제공업체 추가를 선택합니다.
4. Amazon Cognito가 이 공급자를 통해 인증한 사용자에게 보안 인증을 발급할 때 할당하는 보안 주체 태그를 변경하려면 액세스 제어를 위한 속성에서 편집을 선택합니다.
   a. 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.
   b. `sub` 및 `aud` 클레임 기반 보안 주체 태그를 적용하려면 기본 매핑 사용을 선택합니다.
   c. 보안 주체 태그에 대한 속성의 자체 사용자 지정 스키마를 생성하려면 사용자 지정 매핑 사용을 선택합니다. 그런 다음 태그에 표시하려는 각 클레임에서 소싱하려는 태그 키를 입력합니다.

5. 변경 사항 저장을 선택합니다.

액세스 제어에 속성 사용 정책 예

회사 법무 부서의 직원이 해당 부서에 속하고 보안 수준으로 분류된 버킷의 모든 파일을 나열해야 하는 시나리오를 생각해 보세요. 이 직원이 자격 증명 공급자로부터 가져온 토큰에 다음 클레임이 포함되어 있다고 가정합니다.

클레임

```
{
            "...": "other claims",
            "sub" : "57e7b692-4f66-480d-98b8-45a6729b4c88",
            "department" : "legal",
            "clearance" : "confidential",
            "...": "other claims"
        }
```

이러한 속성은 태그에 매핑될 수 있으며 IAM 권한 정책에서 보안 주체 태그로 참조할 수 있습니다. 이제 자격 증명 제공자의 끝에서 사용자 프로파일을 변경하여 액세스를 관리할 수 있습니다. 또는 정책 자체를 변경하지 않고 이름이나 태그를 사용하여 리소스 측에서 속성을 변경할 수 있습니다.

다음 권한 정책은 두 가지 기능을 합니다.

- 사용자의 부서 이름과 일치하는 접두사로 끝나는 모든 S3 버킷에 대한 목록 액세스를 허용합니다.
- 파일의 승인 태그가 사용자의 승인 속성과 일치하는 한 이러한 버킷의 파일에 대한 읽기 액세스를 허용합니다.

권한 정책

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
     {
         "Effect": "Allow",
         "Action": "s3:List*",
```

```
            "Resource": "arn:aws:s3:::*-${aws:PrincipalTag/department}"
        },
        {
            "Effect": "Allow",
            "Action": "s3:GetObject*",
            "Resource": "arn:aws:s3:::*-${aws:PrincipalTag/department}/*",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/clearance": "${aws:PrincipalTag/
clearance}"
                }
            }
        }
    ]
}
```

신뢰 정책은 이 역할을 수임할 수 있는 사용자를 결정합니다. 신뢰 관계 정책은 `sts:AssumeRoleWithWebIdentity` 및 `sts:TagSession`을 사용하여 액세스하도록 허용합니다. 생성한 자격 증명 풀로 정책을 제한하는 조건을 추가하고 인증된 역할에만 정책이 적용되도록 합니다.

신뢰 정책

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "cognito-identity.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRoleWithWebIdentity",
        "sts:TagSession"
      ],
      "Condition": {
        "StringEquals": {
          "cognito-identity.amazonaws.com:aud": "IDENTITY-POOL-ID"
```

```
        },
        "ForAnyValue:StringLike": {
          "cognito-identity.amazonaws.com:amr": "authenticated"
        }
      }
    }
  ]
}
```

액세스 제어를 위한 속성 끄기(콘솔)

액세스 제어를 위한 속성을 비활성화하려면 다음 절차를 따릅니다.

액세스 제어를 위한 속성을 비활성화하려면

1. Amazon Cognito 콘솔에 로그인하고 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. ID 제공업체를 찾습니다. 편집할 ID 제공업체를 선택합니다.
4. 액세스 제어를 위한 속성에서 편집을 선택합니다.
5. 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.
6. 변경 사항 저장을 선택합니다.

기본 공급자 매핑

다음 표에는 Amazon Cognito가 지원하는 인증 공급자에 대한 기본 매핑 정보가 나와 있습니다.

제공업체	토큰 유형	보안 주체 태그 값	예제
Amazon Cognito 사용자 풀	ID 토큰	aud(클라이언트 ID) 및 sub(사용자 ID)	"6jk8ltokc7ac9es6jrtg9q572f", "57e7b692-4f66-480d-98b8-45a6729b4c88"
Facebook	액세스 토큰	aud(app_id), sub(user_id)	"492844718097981", "112177216992379"
Google	ID 토큰	aud(클라이언트 ID) 및 sub(사용자 ID)	"620493171733-eebk7c0hcp5lj3e1tlqp1g

제공업체	토큰 유형	보안 주체 태그 값	예제
			ntt3k0rncv.apps.googleusercontent.com", "109220063452404746097"
SAML	어설션	"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier" , "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name"	"auth0|5e28d196f8f55a0eaaa95de3", "user123@gmail.com"
Apple	ID 토큰	aud(클라이언트 ID) 및 sub(사용자 ID)	"com.amazonaws.ec2-54-80-172-243.compute-1.client", "001968.a6ca34e9c1e742458a26cf8005854be9.0733"
Amazon	액세스 토큰	aud (Client ID on Amzn Dev Ac), user_id(user ID)	"amzn1.application-oa2-client.9d70d9382d3446108aaee3dd763a0fa6", "amzn1.account.AGHNIFJQMFSBG3G6XCPVB35ORQAA"
표준 OIDC 공급자	ID 및 액세스 토큰	aud (as client_id), sub (as user ID)	"620493171733-eebk7c0hcp5lj3e1tlqp1gntt3k0rncv.apps.googleusercontent.com", "109220063452404746097"

제공업체	토큰 유형	보안 주체 태그 값	예제
Twitter	액세스 토큰	aud (app ID; app Secret), sub (user ID)	"DfwifTtKEX1FiIBRnOTIR0CFK; Xgj5xb8xIrIVCPjXgLIdkW7fXmwcJJrFvnoK9gwZkLexo1y5z1", "1269003884292222976"
DevAuth	맵	해당 사항 없음	"tag1", "tag2"

> **Note**
>
> [보안 주체의 태그 키(Tag Key for Principal)] 및 [속성 이름(Attribute names)]에 기본 속성 매핑 옵션이 자동으로 채워집니다. 기본 매핑은 변경할 수 없습니다.

역할 기반 액세스 제어 사용

Amazon Cognito 자격 증명 풀은 인증된 사용자에게 AWS 리소스에 액세스할 수 있는 권한이 제한된 임시 자격 증명 세트를 할당합니다. 각 사용자의 권한은 생성하는 IAM 역할을 통해 제어됩니다. 사용자의 ID 토큰에 있는 클레임에 따라 각 사용자의 역할을 선택하는 규칙을 정의할 수 있습니다. 인증된 사용자의 기본 역할을 정의할 수 있습니다. 인증되지 않은 게스트 사용자의 권한이 제한된 개별 IAM 역할을 정의할 수도 있습니다.

역할 매핑에 사용할 역할 생성

각 역할에 적합한 신뢰 정책을 추가하여 자격 증명 풀의 인증된 사용자에게만 Amazon Cognito에서 역할을 위임할 수 있게 해야 합니다. 다음은 신뢰 정책의 예제입니다.

JSON

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

```
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Federated": "cognito-identity.amazonaws.com"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "cognito-identity.amazonaws.com:aud": "us-east-1:12345678-corner-
cafe-123456790ab"
        },
        "ForAnyValue:StringLike": {
          "cognito-identity.amazonaws.com:amr": "authenticated"
        }
      }
    }
  ]
}
```

이 정책에 의해 `cognito-identity.amazonaws.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, 정책은 `authenticated` 값을 지닌 Amazon Cognito `GetOpenIdToken` API 작업에서 발행한 토큰의 다중 값 `amr` 클레임의 배열 멤버 중 하나를 지정합니다

역할 전달 권한 부여

사용자가 자격 증명 풀에 대한 사용자의 기존 권한을 초과하는 권한으로 역할을 설정하도록 허용하려면, `set-identity-pool-roles` API에 역할을 전달할 수 있는 `iam:PassRole` 권한을 해당 사용자에게 부여하세요. 예를 들어, 사용자가 Amazon S3에 쓸 수 없지만 자격 증명 풀에 설정한 IAM 역할이 Amazon S3에 대한 쓰기 권한을 부여하면 역할에 대해 `iam:PassRole` 권한이 부여된 경우에만 사용자가 이 역할을 설정할 수 있습니다. 다음 예제 정책은 `iam:PassRole` 권한을 허용하는 방법을 보여줍니다.

JSON

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

```
            "Sid": "Stmt1",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/myS3WriteAccessRole"
            ]
        }
    ]
}
```

이 정책 예제에서는 `iam:PassRole` 역할에 대해 `myS3WriteAccessRole` 권한이 부여됩니다. 역할은 역할의 Amazon 리소스 이름(ARN)을 사용하여 지정됩니다. 또한 사용자에게 이 정책을 연결해야 합니다. 자세한 내용은 관리형 정책 작업을 참조하세요.

> **Note**
>
> Lambda 함수는 리소스 기반 정책을 사용합니다. 이때 정책은 Lambda 함수 자체에 직접 연결됩니다. Lambda 함수를 호출하는 규칙을 생성하는 경우 역할은 전달하지 않습니다. 따라서 규칙을 생성하는 사용자에게 `iam:PassRole` 권한이 필요하지 않습니다. Lambda 함수 권한 부여에 대한 자세한 내용은 권한 관리: Lambda 함수 정책 사용을 참조하세요.

토큰을 사용하여 사용자에게 역할 할당

Amazon Cognito 사용자 풀을 통해 로그인하는 사용자의 경우 사용자 풀에서 할당한 ID 토큰에서 역할을 전달할 수 있습니다. ID 토큰의 다음 클레임에 역할이 표시됩니다.

- `cognito:preferred_role` 클레임은 역할 ARN입니다.
- `cognito:roles` 클레임은 허용된 역할 ARN 집합을 포함하는 쉼표로 구분된 문자열입니다.

클레임은 다음과 같이 설정됩니다.

- `cognito:preferred_role` 클레임은 최상위(최저) `Precedence` 값을 가진 그룹의 역할로 설정됩니다. 허용된 역할이 하나뿐인 경우 `cognito:preferred_role`이 해당 역할로 설정됩니다. 여러 역할이 있고 우선 순위가 가장 높은 역할 하나가 없는 경우 이 클레임이 설정되지 않습니다.
- 역할이 하나라도 있는 경우 `cognito:roles` 클레임이 설정됩니다.

토큰을 사용하여 역할을 할당할 때 사용자에게 할당할 수 있는 역할이 여러 개인 경우 Amazon Cognito 자격 증명 풀(페더레이션 자격 증명)은 다음과 같이 역할을 선택합니다.

- GetCredentialsForIdentity `CustomRoleArn` 파라미터가 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 않았고 `cognito:roles` 클레임이 설정되었으며 `GetCredentialsForIdentity`에 대한 호출에 `CustomRoleArn`이 지정되지 않은 경우, 할당할 역할을 결정하는 데 콘솔의 역할 확인 설정 또는 `AmbiguousRoleResolution` 필드(SetIdentityPoolRoles API의 `RoleMappings` 파라미터)가 사용됩니다.

규칙 기반 매핑을 사용하여 사용자에게 역할 할당

규칙을 사용하여 자격 증명 공급자 토큰의 클레임을 IAM 역할에 매핑할 수 있습니다.

각 규칙은 토큰 클레임(예: Amazon Cognito 사용자 풀의 ID 토큰에 있는 사용자 속성), 일치 유형, 값 및 IAM 역할을 지정합니다. 이때 일치 유형은 `Equals`, `NotEqual`, `StartsWith` 또는 `Contains`가 될 수 있습니다. 사용자가 클레임에 대해 일치하는 값을 가진 경우 자격 증명을 얻으면 해당 역할을 수임할 수 있습니다. 예를 들어, `Sales`의 `custom:dept` 사용자 지정 속성 값으로 사용자의 특정한 IAM 역할을 할당하는 규칙을 만들 수 있습니다.

> **Note**
>
> 규칙 설정에서 표준 속성을 구별하기 위해 사용자 지정 속성에 `custom:` 접두사가 필요합니다.

규칙은 순서대로 평가되며 순서를 무시하도록 `CustomRoleArn`이 지정되지 않으면 첫 번째 일치 규칙의 IAM 역할이 사용됩니다. Amazon Cognito 사용자 풀의 사용자 속성에 대한 자세한 내용은 사용자 속성 작업 섹션을 참조하세요.

자격 증명 풀(연동 자격 증명) 콘솔에서 인증 공급자에 대한 여러 규칙을 설정할 수 있습니다. 규칙은 순서대로 적용됩니다. 규칙을 드래그하여 순서를 변경할 수 있습니다. 첫 번째 일치 규칙이 우선합니다. 일치 유형이 `NotEqual`이고 클레임이 없으면 규칙이 평가되지 않습니다. 일치하는 규칙이 없으면 역할 해결 설정이 기본 인증된 역할 사용 또는 거부에 적용됩니다.

API 및 CLI에서, RoleMapping 유형의 AmbiguousRoleResolution 필드에 일치하는 규칙이 없을 때 할당할 역할을 지정할 수 있습니다. 이 유형은 SetIdentityPoolRoles API의 RoleMappings 파라미터에 지정됩니다.

Amazon Cognito 콘솔에서 ID 제공업체에 규칙 기반 매핑을 추가하려면 IdP를 추가하거나 업데이트하고 역할 선택에서 규칙이 있는 역할 선택을 선택합니다. 여기에서 공급자 클레임을 IAM 역할에 매핑하는 규칙을 추가할 수 있습니다.

RoleMapping 유형의 RulesConfiguration 필드를 사용하여 AWS CLI 또는 API에서 자격 증명 공급자에 대한 규칙 기반 매핑을 설정할 수 있습니다. SetIdentityPoolRoles API의 RoleMappings 파라미터에서 이 필드를 지정할 수 있습니다.

예를 들어 다음 AWS CLI 명령은 OIDC IdP에 의해 인증된 Sacramento 위치의 arn:aws:iam::123456789012:role/Sacramento_team_S3_admin 사용자에게 역할을 할당하는 규칙을 추가합니다. arn:aws:iam::123456789012:oidc-provider/myOIDCIdP

```
aws cognito-identity set-identity-pool-roles --region us-east-1 --cli-input-json
 file://role-mapping.json
```

role-mapping.json의 콘텐츠:

```
{
    "IdentityPoolId": "us-east-1:12345678-corner-cafe-123456790ab",
    "Roles": {
        "authenticated": "arn:aws:iam::123456789012:role/myS3WriteAccessRole",
        "unauthenticated": "arn:aws:iam::123456789012:role/myS3ReadAccessRole"
    },
    "RoleMappings": {
        "arn:aws:iam::123456789012:oidc-provider/myOIDCIdP": {
            "Type": "Rules",
            "AmbiguousRoleResolution": "AuthenticatedRole",
            "RulesConfiguration": {
                "Rules": [
                    {
                        "Claim": "locale",
                        "MatchType": "Equals",
                        "Value": "Sacramento",
                        "RoleARN": "arn:aws:iam::123456789012:role/
Sacramento_team_S3_admin"
                    }
                ]
            }
```

```
        }
    }
}
```

자격 증명 풀에 구성된 각 사용자 풀 또는 기타 인증 공급자에 대해 규칙을 25개까지 생성할 수 있습니다. 이 제한은 조정할 수 없습니다. 자세한 내용은 Amazon Cognito의 할당량을 참조하세요.

규칙 기반 매핑에 사용할 토큰 클레임

Amazon Cognito

Amazon Cognito ID 토큰은 JSON 웹 토큰(JWT)으로 표시됩니다. `name`, `family_name` 및 `phone_number`와 같은 인증된 사용자의 자격 증명에 대한 클레임이 이 토큰에 포함됩니다. 표준 클레임에 대한 자세한 내용은 OpenID Connect 사양을 참조하세요. 다음은 표준 클레임 외에 Amazon Cognito와 관련된 추가 클레임입니다.

- `cognito:groups`
- `cognito:roles`
- `cognito:preferred_role`

Amazon

다음 클레임은 클레임의 가능한 값과 함께 Login with Amazon에서 사용할 수 있습니다.

- `iss`: www.amazon.com
- `aud`: 앱 ID
- `sub`: sub Login with Amazon 토큰

Facebook

다음 클레임은 클레임의 가능한 값과 함께 Facebook에서 사용할 수 있습니다.

- `iss`: graph.facebook.com
- `aud`: 앱 ID
- `sub`: sub Facebook 토큰

Google

Google 토큰에는 OpenID Connect 사양의 표준 클레임이 포함됩니다. OpenID 토큰의 모든 클레임은 규칙 기반 매핑에 사용할 수 있습니다. Google 토큰에서 사용할 수 있는 클레임에 대한 자세한 내용은 OpenID Connect를 참조하세요.

Apple

Apple 토큰에는 OpenID Connect 사양의 표준 클레임이 포함됩니다. Apple 토큰에서 사용할 수 있는 클레임에 대한 자세한 내용은 Apple 설명서에서 Sign in with Apple로 사용자 인증을 참조하세요. Apple의 토큰에 항상 `email`이 포함되어 있지는 않습니다.

OpenID

Open ID 토큰의 모든 클레임은 규칙 기반 매핑에 사용할 수 있습니다. 표준 클레임에 대한 자세한 내용은 OpenID Connect 사양을 참조하세요. 사용 가능한 추가 클레임에 대한 내용은 OpenID 제공업체 설명서를 참조하세요.

SAML

수신된 SAML 어설션에서 클레임을 구문 분석합니다. SAML 어설션에서 사용할 수 있는 모든 클레임은 규칙 기반 매핑에서 사용할 수 있습니다.

역할 기반 액세스 제어의 모범 사례

Important

역할에 매핑하는 클레임을 최종 사용자가 수정할 수 있는 경우 최종 사용자가 역할을 맡고 그에 따라 정책을 설정할 수 있습니다. 최종 사용자가 직접 설정할 수 없는 클레임은 승격된 권한을 가진 역할에만 매핑하세요. Amazon Cognito 사용자 풀에서 각 사용자 속성에 대해 앱당 읽기 및 쓰기 권한을 설정할 수 있습니다.

Important

Amazon Cognito 사용자 풀에서 그룹의 역할을 설정하면 이 역할이 사용자의 ID 토큰을 통해 전달됩니다. 이 역할을 사용하려면 자격 증명 풀의 인증된 역할 선택에 대해 토큰으로부터 역할 선택을 설정해야 합니다.
콘솔의 [역할 확인(Role resolution)] 설정과 SetIdentityPoolRoles API의 `RoleMappings` 파라미터를 사용하여 토큰에서 올바른 역할을 확인할 수 없는 경우 실행할 기본 동작을 지정할 수 있습니다.

자격 증명 얻기

Amazon Cognito를 사용하여 애플리케이션에 제한된 권한의 임시 자격 증명을 제공하여 사용자가 AWS 리소스에 액세스할 수 있도록 할 수 있습니다. 이 섹션에서는 자격 증명을 가져오는 방법과 자격 증명 풀에서 Amazon Cognito 자격 증명을 가져오는 방법을 설명합니다.

Amazon Cognito는 인증된 자격 증명과 인증되지 않은 자격 증명을 모두 지원합니다. 미인증 사용자는 자격 증명이 인증되지 않았으므로 앱 혹은 자격 증명의 인증이 중요하지 않은 경우 게스트 사용자 역할에 적합합니다. 인증받은 사용자는 타사 자격 증명 공급자(IdP)를 통해, 또는 자격 증명을 인증받은 사용자 풀을 통해 애플리케이션에 로그인합니다. 미인증 사용자의 액세스 권한을 허용하지 않도록 리소스 권한 범위를 충분히 정했는지 확인하세요.

Amazon Cognito 자격 증명(identity)은 자격 증명(credential)이 아닙니다. ()에서 웹 자격 증명 연동 지원을 사용하여 자격 증명으로 교환됩니다 AWS Security Token Service AWS STS. 앱 사용자의 AWS 자격 증명을 얻으려면 AWS SDK의 Cognito 자격 증명 풀 자격 증명 공급자를 사용합니다. 예를 들어 AWS SDK for JavaScript(v3)에서 `@aws-sdk/credential-providers` 패키지의 `fromCognitoIdentityPool` 함수를 사용합니다. 그런 다음 자격 증명은를 사용하여 자격 증명으로 교환됩니다 AWS STS.

웹 자격 증명 보안 인증 공급자는 AWS SDK의 기본 보안 인증 공급자 체인의 일부입니다. SDK 또는의 로컬 `config` 파일에서 자격 증명 풀 토큰을 AWS 설정하려면 `web_identity_token_file` 프로필 항목을 AWS CLI추가합니다. SDK 및 도구 참조 안내서의 역할 자격 증명 공급자 수임을 참조하세요. AWS SDKs

SDK에 웹 자격 증명 보안 인증을 채우는 방법에 대한 자세한 내용은 SDK 개발자 안내서를 참조하세요. 최상의 결과를 얻으려면에 내장된 자격 증명 풀 통합으로 프로젝트를 시작합니다 AWS Amplify.

AWS 자격 증명 풀을 사용하여 자격 증명을 가져오고 설정하기 위한 SDK 리소스

- Amplify 개발자 센터의 자격 증명 풀 페더레이션(Android)
- Amplify 개발자 센터의 자격 증명 풀 페더레이션(iOS)
- AWS SDK for JavaScript 개발자 안내서의 Amazon Cognito 자격 증명을 사용하여 사용자 인증
- AWS SDK for .NET 개발자 안내서의 Amazon Cognito 자격 증명 공급자
- AWS SDK for Go 개발자 안내서의 프로그래밍 방식으로 자격 증명 지정
- AWS SDK for Java 2.x 개발자 안내서의 코드에 임시 자격 증명 제공
- AWS SDK for PHP 개발자 안내서의 assumeRoleWithWebIdentityCredentialProvider 공급자
- AWS SDK for Python (Boto3) 설명서의 웹 ID 제공업체를 사용하여 역할 수임
- AWS SDK for Rust 개발자 안내서의 자격 증명 및 기본 리전 지정

다음 섹션에서는 일부 AWS SDKs의 예제 코드를 제공합니다.

Android

Amazon Cognito를 사용하여 애플리케이션에 제한된 권한의 임시 자격 증명을 제공하여 사용자가 AWS 리소스에 액세스할 수 있도록 할 수 있습니다. Amazon Cognito는 인증된 자격 증명과 인증되지 않은 자격 증명을 모두 지원합니다. 앱에 AWS 자격 증명을 제공하려면 다음 단계를 따르세요.

Android 앱에서 Amazon Cognito 자격 증명 풀을 사용하려면를 설정합니다 AWS Amplify. 자세한 내용은 Amplify 개발자 센터의 인증을 참조하세요.

Amazon Cognito 자격 증명 검색

인증되지 않은 사용자를 허용하는 경우 최종 사용자의 고유한 Amazon Cognito 식별자(자격 증명 ID)를 즉시 검색할 수 있습니다. 사용자를 인증하는 경우 자격 증명 공급자에서 로그인 토큰을 설정한 후 자격 증명 ID를 검색할 수 있습니다.

```
String identityId = credentialsProvider.getIdentityId();
Log.d("LogTag", "my ID is " + identityId);
```

> **Note**
>
> 애플리케이션의 기본 스레드에서 `getIdentityId()`, `refresh()` 또는 `getCredentials()`를 호출하지 마세요. Android 3.0(API 레벨 11)을 기준으로 기본 애플리케이션 스레드에서 네트워크 I/O를 수행하는 경우 앱이 자동으로 실패하고 NetworkOnMainThreadException이 발생합니다. `AsyncTask`를 사용하여 코드를 백그라운드 스레드로 이동해야 합니다. 자세한 내용은 Android 설명서를 참조하세요. 이미 로컬에 캐시된 경우에만 `getCachedIdentityId()`를 호출하여 ID를 검색할 수 있습니다. 그렇지 않으면 메서드가 null을 반환합니다.

iOS - Objective-C

Amazon Cognito를 사용하여 애플리케이션에 제한된 권한의 임시 자격 증명을 제공하여 사용자가 AWS 리소스에 액세스할 수 있도록 할 수 있습니다. Amazon Cognito 자격 증명 풀은 인증된 자격 증명과 인증되지 않은 자격 증명을 모두 지원합니다. 앱에 AWS 자격 증명을 제공하려면 다음 단계를 완료하세요.

iOS 앱에서 Amazon Cognito 자격 증명 풀을 사용하려면를 설정합니다 AWS Amplify. 자세한 내용은 Amplify 개발자 센터의 Swift 인증 및 Flutter 인증을 참조하세요.

Amazon Cognito 자격 증명 검색

인증되지 않은 사용자를 허용하는 경우 또는 사용자를 인증하는 경우 자격 증명 공급자에서 로그인 토큰을 설정한 후에 최종 사용자의 Amazon Cognito 식별자(자격 증명 ID)를 즉시 검색할 수 있습니다.

```
// Retrieve your Amazon Cognito ID
[[credentialsProvider getIdentityId] continueWithBlock:^id(AWSTask *task) {
    if (task.error) {
        NSLog(@"Error: %@", task.error);
    }
    else {
        // the task result will contain the identity id
        NSString *cognitoId = task.result;
    }
    return nil;
}];
```

Note

`getIdentityId`는 비동기식 호출입니다. 자격 증명 ID가 이미 공급자에 설정된 경우 `credentialsProvider.identityId`를 호출하여 해당 자격 증명을 검색할 수 있으며 이 자격 증명은 로컬에 캐시됩니다. 그러나 공급자에 자격 증명 ID가 설정되지 않은 경우를 호출하면가 `credentialsProvider.identityId` 반환됩니다`nil`. 자세한 내용은 Amplify iOS SDK 참조를 참조하세요.

iOS - Swift

사용자가 AWS 리소스에 액세스할 수 있도록 Amazon Cognito를 사용하여 애플리케이션에 제한된 권한의 임시 자격 증명을 제공할 수 있습니다. Amazon Cognito는 인증된 자격 증명과 인증되지 않은 자격 증명을 모두 지원합니다. 앱에 AWS 자격 증명을 제공하려면 다음 단계를 따르세요.

iOS 앱에서 Amazon Cognito 자격 증명 풀을 사용하려면를 설정합니다 AWS Amplify. 자세한 내용은 Amplify 개발자 센터의 Swift 인증을 참조하세요.

Amazon Cognito 자격 증명 검색

인증되지 않은 사용자를 허용하는 경우 또는 사용자를 인증하는 경우 자격 증명 공급자에서 로그인 토큰을 설정한 후에 최종 사용자의 Amazon Cognito 식별자(자격 증명 ID)를 즉시 검색할 수 있습니다.

```
// Retrieve your Amazon Cognito ID
```

```
credentialsProvider.getIdentityId().continueWith(block: { (task) -> AnyObject? in
    if (task.error != nil) {
        print("Error: " + task.error!.localizedDescription)
    }
    else {
        // the task result will contain the identity id
        let cognitoId = task.result!
        print("Cognito id: \(cognitoId)")
    }
    return task;
})
```

> **Note**
>
> `getIdentityId`는 비동기식 호출입니다. 자격 증명 ID가 이미 공급자에 설정된 경우 `credentialsProvider.identityId`를 호출하여 해당 자격 증명을 검색할 수 있으며 이 자격 증명은 로컬에 캐시됩니다. 그러나 공급자에 자격 증명 ID가 설정되지 않은 경우를 호출하면가 `credentialsProvider.identityId` 반환됩니다`nil`. 자세한 내용은 Amplify iOS SDK 참조를 참조하세요.

JavaScript

자격 증명 풀을 아직 만들지 않았다면 `fromCognitoIdentityPool`를 사용하기 전에 Amazon Cognito 콘솔에서 자격 증명 풀을 만듭니다.

자격 증명 공급자로 자격 증명 풀을 구성한 후 `@aws-sdk/credential-providers` 패키지의 `fromCognitoIdentityPool` 함수를 사용하여 사용자를 인증할 수 있습니다. 글로벌를 설정하는 대신 `credentials` 속성을 통해 자격 증명 공급자를 각 서비스 클라이언트에 직접 `AWS.config.credentials`연결합니다. v3 SDK는 클라이언트가 처음 요청할 때 공급자를 느리게 확인하므로 자격 증명을 얻기 위한 별도의 호출이 없습니다. 다음 예제에서는 공급자를에 연결합니다. `S3Client`

```
import { S3Client } from "@aws-sdk/client-s3";
import { fromCognitoIdentityPool } from "@aws-sdk/credential-providers";

// Set the region where your identity pool exists (for example, us-east-1 or eu-west-1)
const region = "REGION";

// Attach the credentials provider to your service client
```

```
const client = new S3Client({
  region,
  credentials: fromCognitoIdentityPool({
    identityPoolId: "IDENTITY_POOL_ID",
    logins: { // optional tokens, used for authenticated login
      "graph.facebook.com": "FBTOKEN",
      "www.amazon.com": "AMAZONTOKEN",
      "accounts.google.com": "GOOGLETOKEN",
      "appleid.apple.com": "APPLETOKEN"
    },
    clientConfig: { region },
  }),
});

// Credentials are resolved lazily when the client first makes a request.
// To inspect them directly, call the provider and await the result
// (from inside an async function):
const credentials = await client.config.credentials();
const { accessKeyId, secretAccessKey, sessionToken } = credentials;
```

선택 사항인 `logins` 속성은 공급자의 자격 증명 토큰에 대한 자격 증명 공급자 이름의 맵입니다. 자격 증명 공급자에게서 토큰을 받는 방법은 어떤 공급자를 사용하느냐에 따라 다릅니다. 예를 들어 Facebook이 자격 증명 공급자 중 하나인 경우 Facebook SDK의 `FB.login` 함수를 사용하여 자격 증명 공급자 토큰을 얻을 수 있습니다.

```
FB.login(function (response) {
    if (response.authResponse) { // logged in
        const region = "us-east-1";
        const client = new S3Client({
          region,
          credentials: fromCognitoIdentityPool({
            identityPoolId: "us-east-1:1699ebc0-7900-4099-b910-2df94f52a030",
            logins: {
              "graph.facebook.com": response.authResponse.accessToken
            },
            clientConfig: { region },
          }),
        });

        console.log('You are now logged in.');
    } else {
        console.log('There was a problem logging you in.');
    }
```

```
});
```

Amazon Cognito 자격 증명 검색

최종 사용자의 고유한 Amazon Cognito 식별자(자격 증명 ID)를 검색할 수 있습니다. 인증되지 않은 사용자를 허용하는 경우 즉시 검색할 수 있습니다. 사용자를 인증하는 경우 로그인 토큰을 설정한 후 검색합니다. `fromCognitoIdentityPool` 공급자는 확인된 자격 증명`identityId`에를 노출하므로 공급자를 사용하는 클라이언트의 확인된 자격 증명에서 읽을 수 있습니다.

```
// Inside an async function:
const { identityId } = await client.config.credentials();
```

Unity

Amazon Cognito를 사용하여 애플리케이션에 제한된 권한의 임시 자격 증명을 제공하여 사용자가 AWS 리소스에 액세스할 수 있도록 할 수 있습니다. Amazon Cognito는 인증된 자격 증명과 인증되지 않은 자격 증명을 모두 지원합니다. 앱에 AWS 자격 증명을 제공하려면 다음 단계를 따르세요.

AWS SDK for Unity는 이제 SDK for .NET의 일부입니다. 에서 Amazon Cognito를 시작하려면 AWS SDK for .NET 개발자 안내서의 Amazon Cognito 자격 증명 공급자를 SDK for .NET참조하세요. 또는를 사용하여 앱을 빌드하는 옵션은 Amplify Dev Center를 참조하세요 AWS Amplify.

Amazon Cognito 자격 증명 검색

인증되지 않은 사용자를 허용하는 경우 또는 사용자를 인증하는 경우 자격 증명 공급자에서 로그인 토큰을 설정한 후에 최종 사용자의 Amazon Cognito 식별자(자격 증명 ID)를 즉시 검색할 수 있습니다.

```
string identityId = await credentials.GetIdentityIdAsync();
```

임시 자격 증명 AWS 서비스 으로 액세스

ID 풀을 사용한 인증이 성공적으로 완료되면 AWS 자격 증명 세트가 생성됩니다. 이러한 자격 증명을 사용하면 애플리케이션이 IAM 인증으로 보호되는 AWS 리소스를 요청할 수 있습니다. 자격 증명 풀 API 작업에 액세스하기 위해 애플리케이션에 추가할 수 있는 다양한 AWS SDKs를 사용하면 임시 자격 증명을 생성하는 인증되지 않은 API 요청을 할 수 있습니다. 그런 다음 클라이언트에 다른 SDKs AWS 서비스 를 추가하고 해당 임시 자격 증명으로 요청에 서명할 수 있습니다. 임시 자격 증명 역할에 부여된 IAM 권한은 다른 서비스에서 요청하는 작업을 허용해야 합니다.

Amazon Cognito 자격 증명 공급자를 구성하고 AWS 자격 증명을 검색한 후 AWS 서비스 클라이언트를 생성합니다. 다음은 AWS SDK 설명서의 몇 가지 예입니다.

AWS 클라이언트 생성을 위한 SDK 리소스

- AWS SDK for C++ 개발자 안내서의 AWS 클라이언트 구성
- AWS SDK for Go 개발자 안내서의에서 AWS SDK for Go V2 사용 AWS 서비스
- AWS SDK for Java 2.x 개발자 안내서의 HTTP 클라이언트 구성
- AWS SDK for JavaScript 개발자 안내서의 서비스 객체 생성 및 호출
- AWS SDK for Python (Boto3) 설명서에서 클라이언트 생성
- AWS SDK for Rust 개발자 안내서의 서비스 클라이언트 생성
- AWS SDK for Swift 개발자 안내서의 클라이언트 사용

다음 조각은 Amazon DynamoDB 클라이언트를 초기화합니다.

Android

Android 앱에서 Amazon Cognito 자격 증명 풀을 사용하려면를 설정합니다 AWS Amplify. 자세한 내용은 Amplify 개발자 센터의 인증을 참조하세요.

```
// Create a service client with the provider
AmazonDynamoDB client = new AmazonDynamoDBClient(credentialsProvider);
```

자격 증명 공급자는 Amazon Cognito와 통신하여 인증된 사용자와 인증되지 않은 사용자의 고유 식별자와 AWS Mobile SDK에 대한 임시의 제한된 권한 AWS 자격 증명을 모두 검색합니다. 검색된 자격 증명은 한시간 동안 유효하며 만료된 경우 공급자는 자격 증명을 새로 고칩니다.

iOS - Objective-C

iOS 앱에서 Amazon Cognito 자격 증명 풀을 사용하려면를 설정합니다 AWS Amplify. 자세한 내용은 Amplify 개발자 센터의 Swift 인증 및 Flutter 인증을 참조하세요.

```
// create a configuration that uses the provider
AWSServiceConfiguration *configuration = [AWSServiceConfiguration
 configurationWithRegion:AWSRegionUSEast1 provider:credentialsProvider];
// get a client with the default service configuration
AWSDynamoDB *dynamoDB = [AWSDynamoDB defaultDynamoDB];
```

자격 증명 공급자는 Amazon Cognito와 통신하여 인증된 사용자와 인증되지 않은 사용자의 고유 식별자와 AWS Mobile SDK에 대한 임시의 제한된 권한 AWS 자격 증명을 모두 검색합니다. 검색된 자격 증명은 한시간 동안 유효하며 만료된 경우 공급자는 자격 증명을 새로 고칩니다.

iOS - Swift

iOS 앱에서 Amazon Cognito 자격 증명 풀을 사용하려면를 설정합니다 AWS Amplify. 자세한 내용은 Amplify 개발자 센터의 Swift 인증을 참조하세요.

```
// get a client with the default service configuration
let dynamoDB = AWSDynamoDB.default()

// get a client with a custom configuration
AWSDynamoDB.register(with: configuration!, forKey: "USWest2DynamoDB");
let dynamoDBCustom = AWSDynamoDB(forKey: "USWest2DynamoDB")
```

자격 증명 공급자는 Amazon Cognito와 통신하여 인증된 사용자와 인증되지 않은 사용자의 고유 식별자와 AWS Mobile SDK에 대한 임시의 제한된 권한 AWS 자격 증명을 모두 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 식별자와 AWS Mobile SDK의 임시 제한 권한 AWS 자격 증명을 모두 검색합니다. 검색된 자격 증명은 한시간 동안 유효하며 만료된 경우 공급자는 자격 증명을 새로 고칩니다.

Unity

AWS SDK for Unity는 이제 SDK for .NET의 일부입니다. 에서 Amazon Cognito를 시작하려면 AWS SDK for .NET 개발자 안내서의 Amazon Cognito 자격 증명 공급자를 SDK for .NET참조하세요. 또는를 사용하여 앱을 빌드하는 옵션은 Amplify Dev Center를 참조하세요 AWS Amplify.

```
// create a service client that uses credentials provided by Cognito
AmazonDynamoDBClient client = new AmazonDynamoDBClient(credentials, REGION);
```

자격 증명 공급자는 Amazon Cognito와 통신하여 인증된 사용자와 인증되지 않은 사용자의 고유 식별자와 AWS Mobile SDK의 임시 제한 권한 AWS 자격 증명을 모두 검색합니다. 검색된 자격 증명은 한시간 동안 유효하며 만료된 경우 공급자는 자격 증명을 새로 고칩니다.

ID 풀 타사 ID 제공업체

Amazon Cognito 자격 증명 풀을 사용하면 다양한 외부 자격 증명 공급자(IdPs)와 통합하여 애플리케이션의 페더레이션 인증을 통해 임시 AWS 자격 증명을 제공할 수 있습니다. 이러한 외부 IdPs와 함께 작동하도록 자격 증명 풀을 구성하면 Amazon Cognito 사용자 풀, 소셜 공급자, OIDC 공급자 또는 SAML 공급자의 인증을 통해 사용자의 백엔드 AWS 리소스에 대한 액세스를 승인할 수 있습니다. 이 섹션에서는 IdP를 설정하고 Amazon Cognito ID 풀과 통합하는 단계를 다룹니다.

`logins` 속성을 사용하여 자격 증명 공급자(IdP)에게서 받은 자격 증명을 설정할 수 있습니다. ID 풀을 여러 IdP와 연결할 수도 있습니다. 예를 들어 고유한 Amazon Cognito 자격 증명을 두 IdP 로그인 모두와 연결하도록 `logins` 속성에 Facebook 토큰과 Google 토큰을 둘 다 설정할 수 있습니다. 사용자는 어느 계정으로도 인증할 수 있지만 Amazon Cognito는 동일한 사용자 식별자를 반환합니다.

다음 지침은 Amazon Cognito 자격 증명 풀이 지원하는 IdP를 통한 인증을 안내합니다.

주제

- Facebook을 ID 풀 IdP로 설정
- Amazon을 사용하여 ID 풀 IdP로 로그인 설정
- Google을 ID 풀 IdP로 설정
- Sign in with Apple을 ID 풀 IdP로 설정
- OIDC 제공업체를 ID 풀 IdP로 설정
- SAML 제공업체를 ID 풀 IdP로 설정

Facebook을 ID 풀 IdP로 설정

Amazon Cognito ID 풀은 Facebook과 함께 작동하여 애플리케이션 사용자에게 페더레이션 인증을 제공합니다. 이 섹션에서는 Facebook을 IdP로 사용하여 애플리케이션을 등록하고 설정하는 방법을 설명합니다.

Facebook 설정

Facebook 사용자 인증 및 Facebook API와의 상호 작용 이전에 Facebook을 통해 애플리케이션을 등록합니다.

Facebook 개발자 포털을 통해 애플리케이이션을 설정할 수 있습니다. Amazon Cognito 자격 증명 풀에 Facebook을 통합하기 전에 이 절차를 수행합니다.

> **Note**
>
> Amazon Cognito ID 풀 페더레이션은 Facebook Limited Login과 호환되지 않습니다. Limited Login에 설정된 권한을 초과하지 않고 Facebook Login for iOS를 설정하는 방법에 대한 자세한 내용은 Meta for Developers의 Facebook Login for iOS - 빠른 시작을 참조하세요.

Facebook 설정

1. Facebook 개발자 포털에서 Facebook 자격 증명으로 로그인합니다.
2. 앱 메뉴에서 새 앱 추가를 선택합니다.
3. 플랫폼을 선택하고 빠른 시작 프로세스를 완료합니다.

Android

Android 앱을 Facebook 로그인과 통합하는 방법에 대한 자세한 내용은 Facebook Getting Started Guide(Facebook 시작 안내서)를 참조하세요.

iOS - Objective-C

iOS Objective-C 앱을 Facebook 로그인과 통합하는 방법에 대한 자세한 내용은 Facebook Getting Started Guide(Facebook 시작 안내서)를 참조하세요.

iOS - Swift

iOS Swift 앱을 Facebook 로그인과 통합하는 방법에 대한 자세한 내용은 Facebook Getting Started Guide(Facebook 시작 안내서)를 참조하세요.

JavaScript

JavaScript 웹 앱을 Facebook Login과 통합하는 방법에 대한 자세한 내용은 웹의 Facebook 로그인 흐름을 참조하세요.

Amazon Cognito 자격 증명 풀 콘솔에서 ID 제공업체 구성

다음 절차를 사용하여 ID 제공업체를 구성합니다.

Facebook ID 제공업체(idP)를 추가하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.

3. ID 제공업체 추가를 선택합니다.
4. Facebook을 선택합니다.
5. 개발자용 Meta에서 생성한 OAuth 프로젝트의 앱 ID를 입력합니다. 자세한 내용은 개발자용 Meta 문서의 Facebook 로그인을 참조하세요.
6. Amazon Cognito가 이 공급자를 통해 인증된 사용자에게 보안 인증을 발급할 때 요청하는 역할을 설정하려면 역할 설정을 구성하세요.

   - 인증된 역할을 구성할 때 설정한 기본 역할을 이 IdP의 사용자에게 할당하거나 규칙을 사용하여 역할 선택을 선택할 수 있습니다.

     i. 규칙으로 역할 선택을 선택한 경우 사용자 인증의 소스 클레임, 클레임을 비교할 연산자, 이 역할 선택과 일치하는 값을 입력하고 역할 할당이 일치할 때 할당할 역할을 입력합니다. 다른 조건에 따라 추가 규칙을 생성하려면 다른 항목 추가를 선택합니다.

     ii. 역할 해결을 선택합니다. 사용자의 클레임이 규칙과 일치하지 않는 경우 보안 인증을 거부하거나 인증된 역할의 보안 인증을 발급할 수 있습니다.
7. Amazon Cognito가 이 공급자를 통해 인증한 사용자에게 보안 인증을 발급할 때 할당하는 보안 주체 태그를 변경하려면 액세스 제어를 위한 속성을 구성합니다.

   a. 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.

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

   c. 보안 주체 태그에 대한 속성의 자체 사용자 지정 스키마를 생성하려면 사용자 지정 매핑 사용을 선택합니다. 그런 다음 태그에 표시하려는 각 클레임에서 소싱하려는 태그 키를 입력합니다.
8. 변경 사항 저장을 선택합니다.

Facebook 사용

Android

Facebook 인증을 추가하려면 먼저 Facebook 안내서를 따라 Facebook SDK를 애플리케이션에 통합합니다. 그런 다음 Facebook으로 로그인(Login with Facebook) 버튼을 Android 사용자 인터페이스에 추가합니다. Facebook SDK는 세션 객체를 사용하여 상태를 추적합니다. Amazon Cognito는 이 세션 객체의 액세스 토큰을 사용하여 사용자를 인증하고, 고유 식별자를 생성하고, 필요한 경우 사용자에게 다른 AWS 리소스에 대한 액세스 권한을 부여합니다.

Facebook SDK로 사용자를 인증한 후에는 Amazon Cognito 자격 증명 공급자에 세션 토큰을 추가합니다.

Facebook SDK 4.0 이상:

```
Map<String, String> logins = new HashMap<String, String>();
logins.put("graph.facebook.com", AccessToken.getCurrentAccessToken().getToken());
credentialsProvider.setLogins(logins);
```

Facebook SDK 4.0 이전:

```
Map<String, String> logins = new HashMap<String, String>();
logins.put("graph.facebook.com", Session.getActiveSession().getAccessToken());
credentialsProvider.setLogins(logins);
```

Facebook 로그인 프로세스에서 해당 SDK의 싱글톤 세션을 초기화합니다. Facebook 세션 객체에는 Amazon Cognito가 인증된 최종 사용자의 AWS 자격 증명을 생성하는 데 사용하는 OAuth 토큰이 포함되어 있습니다. 또한 Amazon Cognito는 사용자 데이터베이스에 이 특정 Facebook 자격 증명과 일치하는 사용자가 있는지 확인하는 데에도 이 토큰을 사용합니다. 사용자가 이미 있으면 API가 기존의 식별자를 반환합니다. 그렇지 않으면 API에서 새 식별자를 반환합니다. 클라이언트 SDK가 로컬 디바이스에서 식별자를 자동으로 캐시합니다.

> **Note**
>
> 로그인 맵을 설정한 후 `refresh` 또는를 호출`get`하여 AWS 자격 증명을 검색합니다.

iOS - Objective-C

Facebook 인증을 추가하려면 먼저 Facebook 안내서를 따라 Facebook SDK를 애플리케이션에 통합합니다. 그런 다음 Facebook으로 로그인 버튼을 귀하의 사용자 인터페이스에 추가합니다. Facebook SDK는 세션 객체를 사용하여 상태를 추적합니다. Amazon Cognito는 이 세션 객체의 액세스 토큰을 사용하여 사용자를 인증하고 고유한 Amazon Cognito 자격 증명 풀(페더레이션 자격 증명)에 바인딩합니다.

Amazon Cognito에 Facebook 액세스 토큰을 제공하려면 AWSIdentityProviderManager 프로토콜을 구현합니다.

`logins` 메서드를 구현할 때 `AWSIdentityProviderFacebook`이 포함된 사전을 반환합니다. 다음 코드 예와 같이 이 사전은 키 역할을 하고 인증된 Facebook 사용자에게서 받은 현재 액세스 토큰은 값 역할을 합니다.

```
- (AWSTask<NSDictionary<NSString *, NSString *> *> *)logins {
```

```
    FBSDKAccessToken* fbToken = [FBSDKAccessToken currentAccessToken];
    if(fbToken){
        NSString *token = fbToken.tokenString;
        return [AWSTask taskWithResult: @{ AWSIdentityProviderFacebook : token }];
    }else{
            return [AWSTask taskWithError:[NSError errorWithDomain:@"Facebook Login"
                                                              code:-1
                                                          userInfo:@{@"error":@"No current
 Facebook access token"}]];
    }
}
```

`AWSCognitoCredentialsProvider`를 인스턴스화할 때 생성자에서 `AWSIdentityProviderManager`의 값으로 `identityProviderManager`를 구현하는 클래스를 전달하십시오. 자세한 내용을 보려면 AWSCognitoCredentialsProvider 참조 페이지에서 initWithRegionType:identityPoolId:identityProviderManager를 선택하세요.

iOS - Swift

Facebook 인증을 추가하려면 먼저 Facebook 안내서를 따라 Facebook SDK를 애플리케이션에 통합합니다. 그런 다음 Facebook으로 로그인 버튼을 귀하의 사용자 인터페이스에 추가합니다. Facebook SDK는 세션 객체를 사용하여 상태를 추적합니다. Amazon Cognito는 이 세션 객체의 액세스 토큰을 사용하여 사용자를 인증하고 고유한 Amazon Cognito 자격 증명 풀(페더레이션 자격 증명)에 바인딩합니다.

> **Note**
>
> Amazon Cognito ID 풀 페더레이션은 Facebook Limited Login과 호환되지 않습니다. Limited Login에 설정된 권한을 초과하지 않고 Facebook Login for iOS를 설정하는 방법에 대한 자세한 내용은 Meta for Developers의 Facebook Login for iOS - 빠른 시작을 참조하세요.

Amazon Cognito에 Facebook 액세스 토큰을 제공하려면 AWSIdentityProviderManager 프로토콜을 구현합니다.

`logins` 메서드를 구현할 때 `AWSIdentityProviderFacebook`이 포함된 사전을 반환합니다. 다음 코드 예와 같이 이 사전은 키 역할을 하고 인증된 Facebook 사용자에게서 받은 현재 액세스 토큰은 값 역할을 합니다.

```
class FacebookProvider: NSObject, AWSIdentityProviderManager {
```

```
    func logins() -> AWSTask<NSDictionary> {
        if let token = AccessToken.current?.authenticationToken {
            return AWSTask(result: [AWSIdentityProviderFacebook:token])
        }
        return AWSTask(error:NSError(domain: "Facebook Login", code: -1 , userInfo:
 ["Facebook" : "No current Facebook access token"]))
    }
}
```

`AWSCognitoCredentialsProvider`를 인스턴스화할 때 생성자에서 `AWSIdentityProviderManager`의 값으로 `identityProviderManager`를 구현하는 클래스를 전달하십시오. 자세한 내용을 보려면 AWSCognitoCredentialsProvider 참조 페이지로 이동하고 initWithRegionType:identityPoolId:identityProviderManager를 선택합니다.

JavaScript

Facebook 인증을 추가하려면 Facebook Login for the Web(웹용 Facebook 로그인)에 따라 웹 사이트에 Facebook으로 로그인(Login with Facebook) 버튼을 추가합니다. Facebook SDK는 세션 객체를 사용하여 상태를 추적합니다. Amazon Cognito는이 세션 객체의 액세스 토큰을 사용하여 사용자를 인증하고, 고유 식별자를 생성하고, 필요한 경우 사용자에게 다른 AWS 리소스에 대한 액세스 권한을 부여합니다.

Facebook SDK로 사용자를 인증한 후에는 Amazon Cognito 자격 증명 공급자에 세션 토큰을 추가합니다.

```
import { fromCognitoIdentityPool } from "@aws-sdk/credential-providers";
import { S3Client } from "@aws-sdk/client-s3";

FB.login(function (response) {

  // Check if the user logged in successfully.
  if (response.authResponse) {

    console.log('You are now logged in.');

    // Attach a Cognito identity pool credentials provider to your service client,
    // passing the Facebook access token in the logins map.
    const client = new S3Client({
      region: 'REGION',
      credentials: fromCognitoIdentityPool({
        identityPoolId: 'IDENTITY_POOL_ID',
        logins: {
```

```
          'graph.facebook.com': response.authResponse.accessToken
        },
        clientConfig: { region: 'REGION' }
      })
    });

    // The provider resolves and caches AWS credentials the first time the
    // client makes a request. Access AWS resources with client here.

  } else {
    console.log('There was a problem logging you in.');
  }

});
```

Facebook SDK는 Amazon Cognito가 인증된 최종 사용자의 AWS 자격 증명을 생성하는 데 사용하는 OAuth 토큰을 가져옵니다. 또한 Amazon Cognito는 사용자 데이터베이스에 이 Facebook 자격 증명과 일치하는 사용자가 있는지 확인하는 데에도 이 토큰을 사용합니다. 사용자가 이미 있으면 API가 기존의 식별자를 반환합니다. 그렇지 않으면 새 식별자가 반환됩니다. 식별자는 로컬 디바이스에서 클라이언트 SDK에 의해 자동으로 캐시됩니다.

> **Note**
>
> 자격 증명 공급자는 서비스 클라이언트가 처음 요청할 때 자격 AWS 증명을 조기에 확인합니다. AWS SDK for JavaScript에서 Amazon Cognito 자격 증명 풀을 사용하는 방법에 대한 자세한 내용은 AWS SDK for JavaScript v3 개발자 안내서를 참조하세요.

Unity

Facebook 인증을 추가하려면 먼저 Facebook 안내서를 따라 Facebook SDK를 애플리케이션에 통합합니다. Amazon Cognito는 FB 객체의 Facebook 액세스 토큰을 사용하여 Amazon Cognito 자격 증명에 연결된 고유한 사용자 식별자를 생성합니다.

Facebook SDK로 사용자를 인증한 후에는 Amazon Cognito 자격 증명 공급자에 세션 토큰을 추가합니다.

```
void Start()
{
    FB.Init(delegate() {
        if (FB.IsLoggedIn) { //User already logged in from a previous session
```

```
            AddFacebookTokenToCognito();
        } else {
            FB.Login ("email", FacebookLoginCallback);
        }
    });
}

void FacebookLoginCallback(FBResult result)
{
    if (FB.IsLoggedIn)
    {
        AddFacebookTokenToCognito();
    }
    else
    {
        Debug.Log("FB Login error");
    }
}

void AddFacebookTokenToCognito()
{
    credentials.AddLogin ("graph.facebook.com",
 AccessToken.CurrentAccessToken.TokenString);
}
```

`FB.AccessToken`을 사용하기 전에 `FB.Login()`을 호출하고 `FB.IsLoggedIn`이 true인지 확인합니다.

Amazon을 사용하여 ID 풀 IdP로 로그인 설정

Amazon Cognito ID 풀은 Login with Amazon과 함께 작동하여 모바일 및 웹 앱 사용자에게 페더레이션 인증을 제공합니다. 이 섹션에서는 Login with Amazon을 자격 증명 공급자(IdP)로 사용하여 애플리케이션을 등록하고 설정하는 방법을 설명합니다.

개발자 포털에서 Login with Amazon이 Amazon Cognito와 연동하도록 설정합니다. 자세한 내용은 Login with Amazon FAQ에서 Setting Up Login with Amazon(Login with Amazon 설정)을 참조하세요.

> **Note**
>
> Unity 플랫폼에서는 Login with Amazon을 기본적으로 통합할 수 없습니다. 대신 웹 보기를 사용하여 브라우저 로그인 흐름을 수행합니다.

Login with Amazon 설정

Login with Amazon 구현

Amazon 개발자 포털에서 OAuth 애플리케이션을 자격 증명 풀과 통합하도록 설정하고, Login with Amazon 설명서를 찾고, SDK를 다운로드할 수 있습니다. 개발자 콘솔(Developer console)을 선택한 다음 개발자 포털에서 Login with Amazon을 선택합니다. 애플리케이션에 대한 보안 프로필을 생성한 다음 앱에 대한 Login with Amazon 인증 메커니즘을 빌드할 수 있습니다. Login with Amazon 인증을 앱과 통합하는 방법에 대한 자세한 내용은 자격 증명 얻기 섹션을 참조하세요.

Amazon에서는 새 보안 프로필에 대한 OAuth 2.0 클라이언트 ID(client ID)를 발급합니다. 보안 프로필 웹 설정(Web Settings) 탭에서 클라이언트 ID(client ID)를 찾을 수 있습니다. 자격 증명 풀에서 Login with Amazon IdP의 App ID 필드에 보안 프로필 ID를 입력합니다.

> **Note**
>
> 자격 증명 풀에서 Login with Amazon IdP의 App ID 필드에 보안 프로필 ID를 입력합니다. 이는 클라이언트 ID를 사용하는 사용자 풀과 다릅니다.

Amazon Cognito 콘솔에서 외부 공급자 구성

Login with Amazon ID 제공업체(idP)를 추가하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. ID 제공업체 추가를 선택합니다.
4. Login with Amazon을 선택합니다.
5. Login with Amazon에서 생성한 OAuth 프로젝트의 앱 ID를 입력합니다. 자세한 내용은 Login with Amazon 설명서를 참조하세요.
6. Amazon Cognito가 이 공급자를 통해 인증된 사용자에게 보안 인증을 발급할 때 요청하는 역할을 설정하려면 역할 설정을 구성하세요.

    - 인증된 역할을 구성할 때 설정한 기본 역할을 이 IdP의 사용자에게 할당하거나 규칙을 사용하여 역할 선택을 선택할 수 있습니다.

        i. 규칙으로 역할 선택을 선택한 경우 사용자 인증의 소스 클레임, 클레임을 비교할 연산자, 이 역할 선택과 일치하는 값을 입력하고 역할 할당이 일치할 때 할당할 역할을 입력합니다. 다른 조건에 따라 추가 규칙을 생성하려면 다른 항목 추가를 선택합니다.

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

7. Amazon Cognito가 이 공급자를 통해 인증한 사용자에게 보안 인증을 발급할 때 할당하는 보안 주체 태그를 변경하려면 액세스 제어를 위한 속성을 구성합니다.

   a. 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.

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

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

8. 변경 사항 저장을 선택합니다.

Login with Amazon 사용: Android

Amazon 로그인을 승인한 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 승인한 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 승인한 후 `AMZNAccessTokenDelegate`의 `requestDidSucceed` 메서드에서 Amazon Cognito 자격 증명 공급자에 토큰을 전달할 수 있습니다.

```
func requestDidSucceed(apiResult: APIResult!) {
    if apiResult.api == API.AuthorizeUser {
        AIMobileLib.getAccessTokenForScopes(["profile"], withOverrideParams: nil,
 delegate: self)
    } else if apiResult.api == API.GetAccessToken {
        credentialsProvider.logins =
 [AWSCognitoLoginProviderKey.LoginWithAmazon.rawValue: apiResult.result]
    }
}
```

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 전달하세요.

```
import { fromCognitoIdentityPool } from "@aws-sdk/credential-providers";
import { S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({
   region: 'REGION',
   credentials: fromCognitoIdentityPool({
       identityPoolId: 'IDENTITY_POOL_ID',
       logins: {
           'www.amazon.com': 'Amazon Access Token'
       },
       clientConfig: { region: 'REGION' }
   })
});
```

Google을 ID 풀 IdP로 설정

Amazon Cognito ID 풀은 Google과 함께 작동하여 모바일 애플리케이션 사용자에게 페더레이션 인증을 제공합니다. 이 섹션에서는 Google을 IdP로 사용하여 애플리케이션을 등록하고 설정하는 방법을 설명합니다.

Android

> **Note**
>
> 앱이 Google을 사용하고 여러 모바일 플랫폼에서 사용 가능한 경우 앱을 OpenID Connect 공급자로 구성해야 합니다. 더 나은 통합을 위해 생성된 모든 클라이언트 ID를 추가 대상 값으로 추가합니다. Google 클라이언트 간 자격 증명 모델에 대한 자세한 내용은 클라이언트 간 ID를 참조하세요.

Google 설정

Android용 Google 로그인을 활성화하려면 애플리케이션을 위한 Google 개발자 콘솔 프로젝트를 생성합니다.

1. Google 개발자 콘솔로 이동하여 새 프로젝트를 만듭니다.
2. API 및 서비스(APIs & Services)를 선택한 다음 OAuth 동의 화면(OAuth consent screen)을 선택합니다. Google에서 프로필 데이터를 앱과 공유하는 데 동의를 요청할 때 Google이 사용자에게 표시하는 정보를 사용자 지정합니다.
3. 자격 증명(Credentials)을 선택한 다음 자격 증명 생성(Create credentials)을 선택합니다. OAuth 클라이언트 ID(OAuth client ID)를 선택합니다. Android를 애플리케이션 유형(Application type)으로 선택합니다. 앱을 개발하는 각 플랫폼에 대해 별도의 클라이언트 ID를 생성합니다.
4. 자격 증명(Credentials)에서 서비스 계정 관리(Manage service accounts)를 선택합니다. 서비스 계정 생성(Create service account)을 선택합니다. 서비스 계정 세부 정보를 입력한 다음 생성 후 계속(Create and continue)을 선택합니다.
5. 서비스 계정에 프로젝트에 대한 액세스 권한을 부여합니다. 앱에 필요한 대로 사용자에게 서비스 계정에 대한 액세스 권한을 부여합니다.
6. 새 서비스 계정을 선택하고 키(Keys) 탭을 선택하고 키 추가(Add key)를 선택합니다. 새 JSON 키를 생성하고 다운로드합니다.

Google 개발자 콘솔을 사용하는 방법에 대한 자세한 내용은 Google 클라우드 설명서에서 프로젝트 만들기 및 관리를 참조하세요.

Google을 Android 앱에 통합하는 방법에 대한 자세한 내용은 Google ID 설명서에서 Sign in with Google을 사용하여 사용자 인증을 참조하세요.

Google ID 제공업체(idP)를 추가하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. ID 제공업체 추가를 선택합니다.
4. Google을 선택합니다.
5. Google Cloud Platform에서 생성한 OAuth 프로젝트의 클라이언트 ID를 입력합니다. 자세한 내용은 Google Cloud Platform Console 도움말의 OAuth 2.0 설정을 참조하세요.
6. Amazon Cognito가 이 공급자를 통해 인증된 사용자에게 보안 인증을 발급할 때 요청하는 역할을 설정하려면 역할 설정을 구성하세요.
   - 인증된 역할을 구성할 때 설정한 기본 역할을 이 IdP의 사용자에게 할당하거나 규칙을 사용하여 역할 선택을 선택할 수 있습니다.
     i. 규칙으로 역할 선택을 선택한 경우 사용자 인증의 소스 클레임, 클레임을 비교할 연산자, 이 역할 선택과 일치하는 값을 입력하고 역할 할당이 일치할 때 할당할 역할을 입력합니다. 다른 조건에 따라 추가 규칙을 생성하려면 다른 항목 추가를 선택합니다.
     ii. 역할 해결을 선택합니다. 사용자의 클레임이 규칙과 일치하지 않는 경우 보안 인증을 거부하거나 인증된 역할의 보안 인증을 발급할 수 있습니다.
7. Amazon Cognito가 이 공급자를 통해 인증한 사용자에게 보안 인증을 발급할 때 할당하는 보안 주체 태그를 변경하려면 액세스 제어를 위한 속성을 구성합니다.
   a. 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.
   b. sub 및 aud 클레임 기반 보안 주체 태그를 적용하려면 기본 매핑 사용을 선택합니다.
   c. 보안 주체 태그에 대한 속성의 자체 사용자 지정 스키마를 생성하려면 사용자 지정 매핑 사용을 선택합니다. 그런 다음 태그에 표시하려는 각 클레임에서 소싱하려는 태그 키를 입력합니다.
8. 변경 사항 저장을 선택합니다.

Google 사용

애플리케이션에서 Google로 로그인하도록 허용하려면 Android용 Google 설명서의 지침을 따르세요. 사용자가 로그인하면 Google에서 OpenID Connect 인증 토큰을 요청합니다. 그런 다음 Amazon Cognito에서 토큰을 사용하여 사용자를 인증하고 고유한 식별자를 생성합니다.

다음 코드 예는 Google Play 서비스에서 인증 토큰을 가져오는 방법을 보여줍니다.

```
GooglePlayServicesUtil.isGooglePlayServicesAvailable(getApplicationContext());
AccountManager am = AccountManager.get(this);
Account[] accounts = am.getAccountsByType(GoogleAuthUtil.GOOGLE_ACCOUNT_TYPE);
String token = GoogleAuthUtil.getToken(getApplicationContext(), accounts[0].name,
        "audience:server:client_id:YOUR_GOOGLE_CLIENT_ID");
Map<String, String> logins = new HashMap<String, String>();
logins.put("accounts.google.com", token);
credentialsProvider.setLogins(logins);
```

iOS - Objective-C

> **Note**
>
> 앱이 Google을 사용하고 여러 모바일 플랫폼에서 사용 가능한 경우 Google을 OpenID Connect 공급자로 구성합니다. 더 나은 통합을 위해 생성된 모든 클라이언트 ID를 추가 대상 값으로 추가합니다. Google 클라이언트 간 자격 증명 모델에 대한 자세한 내용은 클라이언트 간 ID를 참조하세요.

Google 설정

iOS용 Google 로그인을 사용하려면 애플리케이션을 위한 Google 개발자 콘솔 프로젝트를 생성합니다.

1. Google 개발자 콘솔로 이동하여 새 프로젝트를 만듭니다.
2. API 및 서비스(APIs & Services)를 선택한 다음 OAuth 동의 화면(OAuth consent screen)을 선택합니다. Google에서 프로필 데이터를 앱과 공유하는 데 동의를 요청할 때 Google이 사용자에게 표시하는 정보를 사용자 지정합니다.
3. 자격 증명(Credentials)을 선택한 다음 자격 증명 생성(Create credentials)을 선택합니다. OAuth 클라이언트 ID(OAuth client ID)를 선택합니다. iOS를 애플리케이션 유형(Application type)으로 선택합니다. 앱을 개발하는 각 플랫폼에 대해 별도의 클라이언트 ID를 생성합니다.
4. 자격 증명(Credentials)에서 서비스 계정 관리(Manage service accounts)를 선택합니다. 서비스 계정 생성(Create service account)을 선택합니다. 서비스 계정 세부 정보를 입력한 다음 생성 후 계속(Create and continue)을 선택합니다.
5. 서비스 계정에 프로젝트에 대한 액세스 권한을 부여합니다. 앱에 필요한 대로 사용자에게 서비스 계정에 대한 액세스 권한을 부여합니다.
6. 새 서비스 계정을 선택합니다. 키(Keys) 탭을 선택한 다음 키 추가(Add key)를 선택합니다. 새 JSON 키를 생성하고 다운로드합니다.

Google 개발자 콘솔을 사용하는 방법에 대한 자세한 내용은 Google 클라우드 설명서에서 프로젝트 만들기 및 관리를 참조하세요.

Google을 iOS 앱에 통합하는 방법에 대한 자세한 내용은 Google 자격 증명 설명서에서 Google Sign-In for iOS를 참조하세요.

Google ID 제공업체(idP)를 추가하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. ID 제공업체 추가를 선택합니다.
4. Google을 선택합니다.
5. Google Cloud Platform에서 생성한 OAuth 프로젝트의 클라이언트 ID를 입력합니다. 자세한 내용은 Google Cloud Platform Console 도움말의 OAuth 2.0 설정을 참조하세요.
6. Amazon Cognito가 이 공급자를 통해 인증된 사용자에게 보안 인증을 발급할 때 요청하는 역할을 설정하려면 역할 설정을 구성하세요.
   - 인증된 역할을 구성할 때 설정한 기본 역할을 이 IdP의 사용자에게 할당하거나 규칙을 사용하여 역할 선택을 선택할 수 있습니다.
     i. 규칙으로 역할 선택을 선택한 경우 사용자 인증의 소스 클레임, 클레임을 비교할 연산자, 이 역할 선택과 일치하는 값을 입력하고 역할 할당이 일치할 때 할당할 역할을 입력합니다. 다른 조건에 따라 추가 규칙을 생성하려면 다른 항목 추가를 선택합니다.
     ii. 역할 해결을 선택합니다. 사용자의 클레임이 규칙과 일치하지 않는 경우 보안 인증을 거부하거나 인증된 역할의 보안 인증을 발급할 수 있습니다.
7. Amazon Cognito가 이 공급자를 통해 인증한 사용자에게 보안 인증을 발급할 때 할당하는 보안 주체 태그를 변경하려면 액세스 제어를 위한 속성을 구성합니다.
   a. 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.
   b. sub 및 aud 클레임 기반 보안 주체 태그를 적용하려면 기본 매핑 사용을 선택합니다.
   c. 보안 주체 태그에 대한 속성의 자체 사용자 지정 스키마를 생성하려면 사용자 지정 매핑 사용을 선택합니다. 그런 다음 태그에 표시하려는 각 클레임에서 소싱하려는 태그 키를 입력합니다.
8. 변경 사항 저장을 선택합니다.

Google 사용

애플리케이션에서 Google로 로그인하도록 허용하려면 iOS용 Google 설명서를 따르세요. 인증에 성공하면 OpenID Connect 인증 토큰을 가져오고 Amazon Cognito에서 이를 사용하여 사용자를 인증하고 고유 식별자를 생성합니다.

인증에 성공하면 `id_token`이 포함된 `GTMOAuth2Authentication` 객체를 가져오고 Amazon Cognito에서 이를 사용하여 사용자를 인증하고 고유 식별자를 생성합니다.

```
- (void)finishedWithAuth: (GTMOAuth2Authentication *)auth error: (NSError *) error {
        NSString *idToken = [auth.parameters objectForKey:@"id_token"];
        credentialsProvider.logins = @{ @(AWSCognitoLoginProviderKeyGoogle): idToken };
    }
```

iOS - Swift

> **Note**
>
> 앱이 Google을 사용하고 여러 모바일 플랫폼에서 사용 가능한 경우 Google을 OpenID Connect 공급자로 구성합니다. 더 나은 통합을 위해 생성된 모든 클라이언트 ID를 추가 대상 값으로 추가합니다. Google 클라이언트 간 자격 증명 모델에 대한 자세한 내용은 클라이언트 간 ID를 참조하세요.

Google 설정

iOS용 Google 로그인을 사용하려면 애플리케이션을 위한 Google 개발자 콘솔 프로젝트를 생성합니다.

1. Google 개발자 콘솔로 이동하여 새 프로젝트를 만듭니다.
2. API 및 서비스(APIs & Services)를 선택한 다음 OAuth 동의 화면(OAuth consent screen)을 선택합니다. Google에서 프로필 데이터를 앱과 공유하는 데 동의를 요청할 때 Google이 사용자에게 표시하는 정보를 사용자 지정합니다.
3. 자격 증명(Credentials)을 선택한 다음 자격 증명 생성(Create credentials)을 선택합니다. OAuth 클라이언트 ID(OAuth client ID)를 선택합니다. iOS를 애플리케이션 유형(Application type)으로 선택합니다. 앱을 개발하는 각 플랫폼에 대해 별도의 클라이언트 ID를 생성합니다.
4. 자격 증명(Credentials)에서 서비스 계정 관리(Manage service accounts)를 선택합니다. 서비스 계정 생성(Create service account)을 선택합니다. 서비스 계정 세부 정보를 입력한 다음 생성 후 계속(Create and continue)을 선택합니다.

5. 서비스 계정에 프로젝트에 대한 액세스 권한을 부여합니다. 앱에 필요한 대로 사용자에게 서비스 계정에 대한 액세스 권한을 부여합니다.
6. 새 서비스 계정을 선택하고 키(Keys) 탭을 선택하고 키 추가(Add key)를 선택합니다. 새 JSON 키를 생성하고 다운로드합니다.

Google 개발자 콘솔을 사용하는 방법에 대한 자세한 내용은 Google 클라우드 설명서에서 프로젝트 만들기 및 관리를 참조하세요.

Google을 iOS 앱에 통합하는 방법에 대한 자세한 내용은 Google 자격 증명 설명서에서 Google Sign-In for iOS를 참조하세요.

Amazon Cognito 콘솔 홈 페이지에서 자격 증명 풀 관리(Manage Identity Pools)를 선택합니다.

Amazon Cognito 콘솔에서 외부 공급자 구성

1. Google을 외부 공급자로 사용할 자격 증명 풀의 이름을 선택합니다. 자격 증명 풀에 대한 대시보드 페이지가 표시됩니다.
2. 대시보드 페이지의 우측 상단 모서리에서 자격 증명 풀 편집을 선택합니다. 자격 증명 풀 편집(Edit identity pool) 페이지가 표시됩니다.
3. 아래로 스크롤하고 인증 공급자(Authentication providers)를 선택하여 섹션을 확장합니다.
4. Google 탭을 선택합니다.
5. 잠금 해제를 선택합니다.
6. Google에서 가져온 Google 클라이언트 ID를 입력하고 변경 사항 저장(Save Changes)을 선택합니다.

Google 사용

애플리케이션에서 Google로 로그인하도록 허용하려면 iOS용 Google 설명서를 따르세요. 인증에 성공하면 OpenID Connect 인증 토큰이 생성됩니다. Amazon Cognito에서는 이 토큰을 사용하여 사용자를 인증하고 고유한 식별자를 생성합니다.

인증에 성공하면 `GTMOAuth2Authentication`를 포함하는 `id_token` 객체가 생성됩니다. Amazon Cognito에서는 이 토큰을 사용하여 사용자를 인증하고 고유한 식별자를 생성합니다.

```
func finishedWithAuth(auth: GTMOAuth2Authentication!, error: NSError!) {
    if error != nil {
      print(error.localizedDescription)
```

```
    }
    else {
      let idToken = auth.parameters.objectForKey("id_token")
      credentialsProvider.logins = [AWSCognitoLoginProviderKey.Google.rawValue:
 idToken!]
    }
}
```

JavaScript

> **Note**
>
> 앱이 Google을 사용하고 여러 모바일 플랫폼에서 사용 가능한 경우 Google을 OpenID Connect 공급자로 구성해야 합니다. 더 나은 통합을 위해 생성된 모든 클라이언트 ID를 추가 대상 값으로 추가합니다. Google 클라이언트 간 자격 증명 모델에 대한 자세한 내용은 클라이언트 간 ID를 참조하세요.

Google 설정

JavaScript 웹 앱용 Google 로그인을 사용하려면 애플리케이션을 위한 Google 개발자 콘솔 프로젝트를 생성합니다.

1. Google 개발자 콘솔로 이동하여 새 프로젝트를 만듭니다.
2. API 및 서비스(APIs & Services)를 선택한 다음 OAuth 동의 화면(OAuth consent screen)을 선택합니다. Google에서 프로필 데이터를 앱과 공유하는 데 동의를 요청할 때 Google이 사용자에게 표시하는 정보를 사용자 지정합니다.
3. 자격 증명(Credentials)을 선택한 다음 자격 증명 생성(Create credentials)을 선택합니다. OAuth 클라이언트 ID(OAuth client ID)를 선택합니다. 웹 애플리케이션(Web application)을 애플리케이션 유형(Application type)으로 선택합니다. 앱을 개발하는 각 플랫폼에 대해 별도의 클라이언트 ID를 생성합니다.
4. 자격 증명(Credentials)에서 서비스 계정 관리(Manage service accounts)를 선택합니다. 서비스 계정 생성(Create service account)을 선택합니다. 서비스 계정 세부 정보를 입력한 다음 생성 후 계속(Create and continue)을 선택합니다.
5. 서비스 계정에 프로젝트에 대한 액세스 권한을 부여합니다. 앱에 필요한 대로 사용자에게 서비스 계정에 대한 액세스 권한을 부여합니다.
6. 새 서비스 계정을 선택하고 키(Keys) 탭을 선택하고 키 추가(Add key)를 선택합니다. 새 JSON 키를 생성하고 다운로드합니다.

Google 개발자 콘솔을 사용하는 방법에 대한 자세한 내용은 Google 클라우드 설명서에서 프로젝트 만들기 및 관리를 참조하세요.

Google을 웹 앱에 통합하는 방법에 대한 자세한 내용은 Google 자격 증명 설명서에서 Sign in With Google을 참조하세요.

Amazon Cognito 콘솔에서 외부 공급자 구성

Google ID 제공업체(idP)를 추가하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. ID 제공업체 추가를 선택합니다.
4. Google을 선택합니다.
5. Google Cloud Platform에서 생성한 OAuth 프로젝트의 클라이언트 ID를 입력합니다. 자세한 내용은 Google Cloud Platform Console 도움말의 OAuth 2.0 설정을 참조하세요.
6. Amazon Cognito가 이 공급자를 통해 인증된 사용자에게 보안 인증을 발급할 때 요청하는 역할을 설정하려면 역할 설정을 구성하세요.
   - 인증된 역할을 구성할 때 설정한 기본 역할을 이 IdP의 사용자에게 할당하거나 규칙을 사용하여 역할 선택을 선택할 수 있습니다.
     i. 규칙으로 역할 선택을 선택한 경우 사용자 인증의 소스 클레임, 클레임을 비교할 연산자, 이 역할 선택과 일치하는 값을 입력하고 역할 할당이 일치할 때 할당할 역할을 입력합니다. 다른 조건에 따라 추가 규칙을 생성하려면 다른 항목 추가를 선택합니다.
     ii. 역할 해결을 선택합니다. 사용자의 클레임이 규칙과 일치하지 않는 경우 보안 인증을 거부하거나 인증된 역할의 보안 인증을 발급할 수 있습니다.
7. Amazon Cognito가 이 공급자를 통해 인증한 사용자에게 보안 인증을 발급할 때 할당하는 보안 주체 태그를 변경하려면 액세스 제어를 위한 속성을 구성합니다.
   a. 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.
   b. sub 및 aud 클레임 기반 보안 주체 태그를 적용하려면 기본 매핑 사용을 선택합니다.
   c. 보안 주체 태그에 대한 속성의 자체 사용자 지정 스키마를 생성하려면 사용자 지정 매핑 사용을 선택합니다. 그런 다음 태그에 표시하려는 각 클레임에서 소싱하려는 태그 키를 입력합니다.
8. 변경 사항 저장을 선택합니다.

Google 사용

애플리케이션에서 Google로 로그인하도록 허용하려면 웹용 Google 설명서를 따르세요.

인증에 성공하면 id_token이 포함된 응답 객체가 생성됩니다 Amazon Cognito에서는 이 객체를 사용하여 사용자를 인증하고 고유한 식별자를 생성합니다.

```
import { fromCognitoIdentityPool } from "@aws-sdk/credential-providers";
import { S3Client } from "@aws-sdk/client-s3";

function signinCallback(authResult) {
  if (authResult['status']['signed_in']) {

     // Attach a Cognito identity pool credentials provider to your service client,
     // passing the Google ID token in the logins map.
     const client = new S3Client({
        region: 'REGION',
        credentials: fromCognitoIdentityPool({
           identityPoolId: 'IDENTITY_POOL_ID',
           logins: {
              'accounts.google.com': authResult['id_token']
           },
           clientConfig: { region: 'REGION' }
        })
     });

     // The provider resolves and caches AWS credentials the first time the
     // client makes a request. Access AWS resources with client here.
  }
}
```

Sign in with Apple을 ID 풀 IdP로 설정

Amazon Cognito ID 풀은 Sign in with Apple과 함께 작동하여 모바일 애플리케이션 및 웹 애플리케이션 사용자에게 페더레이션 인증을 제공합니다. 이 섹션에서는 Apple로 로그인을 자격 증명 공급자(IdP)로 사용하여 애플리케이션을 등록하고 설정하는 방법을 설명합니다.

Apple로 로그인을 인증 공급자로 자격 증명 풀에 추가하려면 두 단계를 진행해야 합니다. 먼저 애플리케이션에서 Apple로 로그인을 통합한 후 자격 증명 풀에서 Sign in with Apple을 구성합니다. Apple로 로그인 설정에 대한 최신 정보는 Apple 개발자 설명서의 Apple로 로그인용 환경 구성을 참조하세요.

Sign in with Apple 설정

Apple로 로그인을 IdP로 구성하려면 애플리케이션을 Apple에 등록하여 클라이언트 ID를 받습니다.

1. Apple에서 개발자 계정을 생성합니다.
2. Apple 자격 증명으로 로그인합니다.
3. 왼쪽 탐색 창에서 Certificates, IDs & Profiles(인증서, ID 및 프로필)를 선택합니다.
4. 왼쪽 탐색 창에서 Identifiers(식별자)를 선택합니다.
5. 식별자(Identifiers) 페이지에서 +아이콘을 선택합니다.
6. 새 식별자 등록(Register a New Identifier) 페이지에서 앱 ID(App IDs)를 선택한 다음 계속(Continue)을 선택합니다.
7. 앱 ID 등록(Register an App ID) 페이지에서 다음을 수행합니다.
    a. 설명에 설명을 입력합니다.
    b. Bundle ID(번들 ID)에서 식별자를 입력합니다. 이 번들 ID(Bundle ID)를 적어 둡니다. Apple을 자격 증명 풀의 공급자로 구성하려면 이 값이 필요합니다.
    c. 기능(Capabilities)에서 Apple로 로그인(Sign In with Apple)을 선택한 다음 편집(Edit)을 선택합니다.
    d. Sign in with Apple: 앱 ID 구성 페이지에서 앱에 적절한 설정을 선택합니다. 그런 다음 저장을 선택합니다.
    e. 계속을 선택합니다.
8. 앱 ID 확인(Confirm your App ID) 페이지에서 등록(Register)을 선택합니다.
9. Sign In with Apple을 네이티브 iOS 애플리케이션과 통합하려면 10단계로 이동합니다. 11단계는 Sign in with Apple JS와 통합하려는 애플리케이션을 위한 것입니다.
10. 식별자(Identifiers) 페이지에서 앱 ID(App IDs) 메뉴를 선택한 다음 서비스 ID(Services IDs)를 선택합니다. + 아이콘을 선택합니다.
11. 새 식별자 등록(Register a New Identifier) 페이지에서 서비스 ID(Services IDs)를 선택한 다음 계속(Continue)을 선택합니다.
12. 서비스 ID 등록(Register a Services ID) 페이지에서 다음을 수행합니다.
    a. 설명에 설명을 입력합니다.
    b. Identifier(식별자)에 식별자를 입력합니다. 서비스 ID를 적어 둡니다. Apple을 자격 증명 풀의 공급자로 구성하려면 이 값이 필요합니다.
    c. Apple로 로그인을 선택한 다음 구성을 선택합니다.

d. 웹 인증 구성(Web Authentication Configuration) 페이지에서 기본 앱 ID(Primary App ID)를 선택합니다. 웹 사이트 URL(Website URLs)에서 + 아이콘을 선택합니다. Domains and Subdomains(도메인 및 하위 도메인)에 앱의 도메인 이름을 입력합니다. 반환 URL(Return URLs)에 Apple로 로그인을 통해 사용자를 인증한 후 권한 부여가 사용자를 리디렉션하는 콜백 URL을 입력합니다.

e. 다음(Next)을 선택합니다.

f. 계속(Continue), 등록(Register)을 차례로 선택합니다.

13. 왼쪽 탐색 창에서 키를 선택합니다.
14. 키(Keys) 페이지에 + 아이콘을 선택합니다.
15. 새 키 등록(Register a New Key) 페이지에서 다음을 수행합니다.

a. 키 이름에서 키 이름을 입력합니다.

b. Apple로 로그인을 선택한 다음 구성을 선택합니다.

c. Configure Key(키 구성) 페이지에서 Primary App ID(기본 앱 ID)를 선택하고 저장을 선택합니다.

d. 계속(Continue), 등록(Register)을 차례로 선택합니다.

> **Note**
>
> Apple로 로그인을 네이티브 iOS 애플리케이션과 통합하려면 Implementing User Authentication with Sign in with Apple(Apple로 로그인으로 사용자 인증 구현)을 참조하세요. Apple로 로그인을 네이티브 iOS 이외의 플랫폼에서 통합하려면 Sign in with Apple JS(Apple JS로 로그인)를 참조하세요.

Amazon Cognito 페더레이션 자격 증명 콘솔에서 외부 공급자 구성

다음 절차를 사용하여 외부 공급자를 구성합니다.

Sign in with Apple ID 제공업체(idP)를 추가하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. ID 제공업체 추가를 선택합니다.
4. Sign in with Apple을 선택합니다.

5. Apple 개발자와 함께 생성한 OAuth 프로젝트의 서비스 ID를 입력합니다. 자세한 내용은 Sign in with Apple 설명서의 Sign in with Apple로 사용자 인증을 참조하세요.
6. Amazon Cognito가 이 공급자를 통해 인증된 사용자에게 보안 인증을 발급할 때 요청하는 역할을 설정하려면 역할 설정을 구성하세요.
   - 인증된 역할을 구성할 때 설정한 기본 역할을 이 IdP의 사용자에게 할당하거나 규칙을 사용하여 역할 선택을 선택할 수 있습니다.
     i. 규칙으로 역할 선택을 선택한 경우 사용자 인증의 소스 클레임, 클레임을 비교할 연산자, 이 역할 선택과 일치하는 값을 입력하고 역할 할당이 일치할 때 할당할 역할을 입력합니다. 다른 조건에 따라 추가 규칙을 생성하려면 다른 항목 추가를 선택합니다.
     ii. 역할 해결을 선택합니다. 사용자의 클레임이 규칙과 일치하지 않는 경우 보안 인증을 거부하거나 인증된 역할의 보안 인증을 발급할 수 있습니다.
7. Amazon Cognito가 이 공급자를 통해 인증한 사용자에게 보안 인증을 발급할 때 할당하는 보안 주체 태그를 변경하려면 액세스 제어를 위한 속성을 구성합니다.
   a. 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.
   b. sub 및 aud 클레임 기반 보안 주체 태그를 적용하려면 기본 매핑 사용을 선택합니다.
   c. 보안 주체 태그에 대한 속성의 자체 사용자 지정 스키마를 생성하려면 사용자 지정 매핑 사용을 선택합니다. 그런 다음 태그에 표시하려는 각 클레임에서 소싱하려는 태그 키를 입력합니다.
8. 변경 사항 저장을 선택합니다.

Amazon Cognito 페더레이션 자격 증명 CLI에서 Apple을 공급자로 사용하여 로그인 예제

이 예에서는 Apple로 로그인을 IdP로 사용하여 `MyIdentityPool`이라는 자격 증명 풀을 생성합니다.

```
aws cognito-identity create-identity-pool --identity-
pool-name MyIdentityPool --supported-login-providers
appleid.apple.com="sameple.apple.clientid"
```

자세한 내용은 자격 증명 풀 생성 섹션을 참조하세요.

Amazon Cognito 자격 증명 ID 생성

이 예에서는 Amazon Cognito ID를 생성(또는 검색)합니다. 이는 퍼블릭 API이므로 이 API를 호출하는 데 자격 증명이 필요하지 않습니다.

```
aws cognito-identity get-id --identity-pool-id SampleIdentityPoolId --
logins appleid.apple.com="SignInWithAppleIdToken"
```

자세한 내용은 get-id 섹션을 참조하세요.

Amazon Cognito 자격 증명 ID의 자격 증명 가져오기

이 예제에서는 제공된 자격 증명 ID 및 Sign in with Apple 로그인의 자격 증명을 반환합니다. 이는 퍼블릭 API이므로 이 API를 호출하는 데 자격 증명이 필요하지 않습니다.

```
aws cognito-identity get-credentials-for-identity --identity-id
SampleIdentityId --logins appleid.apple.com="SignInWithAppleIdToken"
```

자세한 내용은 get-credentials-for-identity 섹션을 참조하세요.

Sign in with Apple 사용: Android

Apple은 Android용 Sign in with Apple을 지원하는 SDK를 제공하지 않습니다. 대신 웹 보기에서 웹 흐름을 사용할 수 있습니다.

- 애플리케이션에서 Sign in with Apple을 구성하려면 Apple 설명서의 Sign in with Apple 웹 페이지 구성을 참조하세요.
- Sign in with Apple 버튼을 Android 사용자 인터페이스에 추가하려면 Apple 설명서의 웹에 Sign in with Apple 버튼 표시를 수행합니다.
- Apple로 로그인을 사용하여 사용자를 안전하게 인증하려면 Apple 설명서에서 Authenticating Users with Sign in with Apple(Apple로 로그인으로 사용자 인증)을 따르세요.

세션 객체를 사용하여 Apple로 로그인하여 상태를 추적합니다. Amazon Cognito는이 세션 객체의 ID 토큰을 사용하여 사용자를 인증하고, 고유 식별자를 생성하고, 필요한 경우 사용자에게 다른 AWS 리소스에 대한 액세스 권한을 부여합니다.

```
@Override
public void onSuccess(Bundle response) {
    String token = response.getString("id_token");
    Map<String, String> logins = new HashMap<String, String>();
    logins.put("appleid.apple.com", token);
    credentialsProvider.setLogins(logins);
}
```

Sign in with Apple 사용: iOS - Objective-C

Apple은 네이티브 iOS 애플리케이션에서 Sign in with Apple에 대한 SDK 지원을 제공했습니다. 네이티브 iOS 디바이스에서 Apple로 로그인으로 사용자 인증을 구현하려면 Apple 설명서에서 Implementing User Authentication with Sign in with Apple(Apple로 로그인으로 사용자 인증 구현)을 따르세요.

Amazon Cognito는 ID 토큰을 사용하여 사용자를 인증하고, 고유 식별자를 생성하고, 필요한 경우 사용자에게 다른 AWS 리소스에 대한 액세스 권한을 부여합니다.

```
(void)finishedWithAuth: (ASAuthorizationAppleIDCredential *)auth error: (NSError *)
 error {
        NSString *idToken = [ASAuthorizationAppleIDCredential
 objectForKey:@"identityToken"];
        credentialsProvider.logins = @{ "appleid.apple.com": idToken };
    }
```

Sign in with Apple 사용: iOS - Swift

Apple은 네이티브 iOS 애플리케이션에서 Sign in with Apple에 대한 SDK 지원을 제공했습니다. 네이티브 iOS 디바이스에서 Apple로 로그인으로 사용자 인증을 구현하려면 Apple 설명서에서 Implementing User Authentication with Sign in with Apple(Apple로 로그인으로 사용자 인증 구현)을 따르세요.

Amazon Cognito는 ID 토큰을 사용하여 사용자를 인증하고, 고유 식별자를 생성하고, 필요한 경우 사용자에게 다른 AWS 리소스에 대한 액세스 권한을 부여합니다.

iOS에서 Apple로 로그인 설정에 대한 자세한 내용은 Set up Sign in with Apple(Apple로 로그인 설정)을 참조하세요.

```
func finishedWithAuth(auth: ASAuthorizationAppleIDCredential!, error: NSError!) {
    if error != nil {
      print(error.localizedDescription)
    }
    else {
      let idToken = auth.identityToken,
      credentialsProvider.logins = ["appleid.apple.com": idToken!]
    }
}
```

Sign in with Apple 사용: JavaScript

Apple은 JavaScript용 Sign in with Apple을 지원하는 SDK를 제공하지 않습니다. 대신 웹 보기에서 웹 흐름을 사용할 수 있습니다.

- 애플리케이션에서 Sign in with Apple을 구성하려면 Apple 설명서의 Sign in with Apple 웹 페이지 구성을 참조하세요.
- Sign in with Apple 버튼을 JavaScript 사용자 인터페이스에 추가하려면 Apple 설명서의 웹에 Sign in with Apple 버튼 표시를 수행합니다.
- Apple로 로그인을 사용하여 사용자를 안전하게 인증하려면 Apple 설명서에서 Authenticating Users with Sign in with Apple(Apple로 로그인으로 사용자 인증)을 따르세요.

세션 객체를 사용하여 Apple로 로그인하여 상태를 추적합니다. Amazon Cognito는이 세션 객체의 ID 토큰을 사용하여 사용자를 인증하고, 고유 식별자를 생성하고, 필요한 경우 사용자에게 다른 AWS 리소스에 대한 액세스 권한을 부여합니다.

```
import { fromCognitoIdentityPool } from "@aws-sdk/credential-providers";
import { S3Client } from "@aws-sdk/client-s3";

function signinCallback(authResult) {
     // Attach a Cognito identity pool credentials provider to your service client,
     // passing the Apple ID token in the logins map.
     const client = new S3Client({
        region: 'REGION',
        credentials: fromCognitoIdentityPool({
           identityPoolId: 'IDENTITY_POOL_ID',
           logins: {
              'appleid.apple.com': authResult['id_token']
           },
           clientConfig: { region: 'REGION' }
        })
     });

     // The provider resolves and caches AWS credentials the first time the
     // client makes a request. Access AWS resources with client here.
}
```

OIDC 제공업체를 ID 풀 IdP로 설정

OpenID Connect는 여러 로그인 공급자가 지원하는 인증에 대한 개방형 표준입니다. Amazon Cognito를 사용하면 AWS Identity and Access Management를 통해 구성하는 OpenID Connect 제공업체와 자격 증명을 연결할 수 있습니다.

OpenID Connect 공급자 추가

OpenID Connect 공급자를 생성하는 방법에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 OIDC(OpenID Connect) ID 제공업체 생성을 참조하세요.

Amazon Cognito와 공급자 연결

OIDC ID 제공업체(idP)를 생성하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. ID 제공업체 추가를 선택합니다.
4. OpenId Connect(OIDC)를 선택합니다.
5. AWS 계정의 IAM IdP에서 OIDC ID 제공업체를 선택합니다. 새 SAML 공급자를 추가하려면 새 공급자 생성을 선택하여 IAM 콘솔로 이동합니다.
6. Amazon Cognito가 이 공급자를 통해 인증된 사용자에게 보안 인증을 발급할 때 요청하는 역할을 설정하려면 역할 설정을 구성하세요.
   - 인증된 역할을 구성할 때 설정한 기본 역할을 이 IdP의 사용자에게 할당하거나 규칙을 사용하여 역할 선택을 선택할 수 있습니다.
     i. 규칙으로 역할 선택을 선택한 경우 사용자 인증의 소스 클레임, 클레임을 비교할 연산자, 이 역할 선택과 일치하는 값을 입력하고 역할 할당이 일치할 때 할당할 역할을 입력합니다. 다른 조건에 따라 추가 규칙을 생성하려면 다른 항목 추가를 선택합니다.
     ii. 역할 해결을 선택합니다. 사용자의 클레임이 규칙과 일치하지 않는 경우 보안 인증을 거부하거나 인증된 역할의 보안 인증을 발급할 수 있습니다.
7. Amazon Cognito가 이 공급자를 통해 인증한 사용자에게 보안 인증을 발급할 때 할당하는 보안 주체 태그를 변경하려면 액세스 제어를 위한 속성을 구성합니다.
   a. 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.
   b. sub 및 aud 클레임 기반 보안 주체 태그를 적용하려면 기본 매핑 사용을 선택합니다.
   c. 보안 주체 태그에 대한 속성의 자체 사용자 지정 스키마를 생성하려면 사용자 지정 매핑 사용을 선택합니다. 그런 다음 태그에 표시하려는 각 클레임에서 소싱하려는 태그 키를 입력합니다.
8. 변경 사항 저장을 선택합니다.

여러 OpenID Connect 공급자를 단일 자격 증명 풀에 연결할 수 있습니다.

OpenID Connect 사용

로그인하고 ID 토큰을 받는 방법은 공급자 설명서를 참조하세요.

토큰을 받은 후에는 토큰을 로그인 맵에 추가합니다. 공급자의 URI를 키로 사용합니다.

OpenID Connect 토큰 검증

Amazon Cognito와 처음 통합할 때 `InvalidToken` 예외가 발생할 수 있습니다. Amazon Cognito가 OpenID Connect(OIDC) 토큰을 검증하는 방법을 이해해야 합니다.

> **Note**
>
> 여기(https://tools.ietf.org/html/rfc7523)에 명시된 바와 같이 Amazon Cognito는 시스템 간의 클럭 스큐를 처리하는 5분의 유예 기간을 제공합니다.

1. `iss` 파라미터가 로그인 맵에 사용된 키(예: login.provider.com)와 일치해야 합니다.
2. 서명이 유효해야 합니다. RSA 퍼블릭 키를 통해 서명을 확인할 수 있어야 합니다.

   > **Note**
   >
   > ID 풀은 짧은 기간 동안 OIDC IdP 서명 키의 캐시를 유지합니다. 제공업체가 서명 키를 변경하면 Amazon Cognito는 이 캐시가 새로 고칠 때까지 `NoKeyFound` 오류를 반환할 수 있습니다. 이 오류가 발생하면 ID 풀이 서명 키를 새로 고칠 때까지 약 10분 정도 기다립니다.

3. 인증서 퍼블릭 키의 지문은 OIDC 공급자를 생성할 때 IAM에서 설정한 지문과 일치합니다.
4. `azp` 파라미터가 있으면 이 값을 OIDC 공급자에서 나열된 클라이언트 ID와 비교하여 확인합니다.
5. `azp` 파라미터가 없으면 `aud` 파라미터를 OpenId Connect 공급자에서 나열된 클라이언트 ID와 비교하여 확인합니다.

jwt.io 웹 사이트는 토큰을 디코딩하고 이러한 값을 확인하는 데 사용할 수 있는 중요한 리소스입니다.

Android

```
Map<String, String> logins = new HashMap<String, String>();
logins.put("login.provider.com", token);
credentialsProvider.setLogins(logins);
```

iOS - Objective-C

```
credentialsProvider.logins = @{ "login.provider.com": token }
```

JavaScript

```
import { fromCognitoIdentityPool } from "@aws-sdk/credential-providers";
import { S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({
 region: 'REGION',
 credentials: fromCognitoIdentityPool({
    identityPoolId: 'IDENTITY_POOL_ID',
    logins: {
       'login.provider.com': token
    },
    clientConfig: { region: 'REGION' }
 })
});
```

SAML 제공업체를 ID 풀 IdP로 설정

Amazon Cognito ID 풀을 사용하면 SAML 2.0을 통해 ID 제공업체(IdPs)로 사용자를 인증할 수 있습니다. Amazon Cognito에서 SAML을 지원하는 IdP를 사용하여 사용자를 위한 간단한 온보딩 흐름을 제공할 수 있습니다. SAML을 지원하는 IdP는 사용자가 수임할 수 있는 IAM 역할을 지정합니다. 이렇게 하면 사용자마다 서로 다른 권한 집합을 받을 수 있습니다.

SAML IdP에 맞게 자격 증명 풀 구성

다음 단계에서는 자격 증명 풀에서 SAML 기반 IdP를 사용하도록 구성하는 방법을 설명합니다.

> **Note**
>
> 자격 증명 풀에서 SAML 공급자를 지원하도록 구성하기 전에 먼저 IAM 콘솔에서 SAML IdP를 구성합니다. 자세한 내용은 IAM 사용 설명서에서 AWS와 서드 파티 SAML 솔루션 공급자 통합을 참조하세요.

SAML ID 제공업체(idP)를 추가하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.

2. 사용자 액세스 탭을 선택합니다.
3. ID 제공업체 추가를 선택합니다.
4. SAML을 선택합니다.
5. AWS 계정의 IAM IdP에서 SAML ID 제공업체를 선택합니다. 새 SAML 공급자를 추가하려면 새 공급자 생성을 선택하여 IAM 콘솔로 이동합니다.
6. Amazon Cognito가 이 공급자를 통해 인증된 사용자에게 보안 인증을 발급할 때 요청하는 역할을 설정하려면 역할 설정을 구성하세요.

   - 인증된 역할을 구성할 때 설정한 기본 역할을 이 IdP의 사용자에게 할당하거나 규칙을 사용하여 역할 선택을 선택할 수 있습니다.

     i. 규칙으로 역할 선택을 선택한 경우 사용자 인증의 소스 클레임, 클레임을 비교할 연산자, 이 역할 선택과 일치하는 값을 입력하고 역할 할당이 일치할 때 할당할 역할을 입력합니다. 다른 조건에 따라 추가 규칙을 생성하려면 다른 항목 추가를 선택합니다.

     ii. 역할 해결을 선택합니다. 사용자의 클레임이 규칙과 일치하지 않는 경우 보안 인증을 거부하거나 인증된 역할의 보안 인증을 발급할 수 있습니다.
7. Amazon Cognito가 이 공급자를 통해 인증한 사용자에게 보안 인증을 발급할 때 할당하는 보안 주체 태그를 변경하려면 액세스 제어를 위한 속성을 구성합니다.

   a. 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.

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

   c. 보안 주체 태그에 대한 속성의 자체 사용자 지정 스키마를 생성하려면 사용자 지정 매핑 사용을 선택합니다. 그런 다음 태그에 표시하려는 각 클레임에서 소싱하려는 태그 키를 입력합니다.
8. 변경 사항 저장을 선택합니다.

SAML IdP 구성

SAML 공급자를 생성한 후에 SAML IdP에서 IdP와 AWS간에 신뢰 당사자 신뢰를 추가하도록 구성합니다. 많은 IdP에서 IdP가 XML 문서에서 신뢰 당사자 정보 및 인증서를 읽는 데 사용할 수 있는 URL을 지정할 수 있습니다. 의 AWS경우 https://signin.aws.amazon.com/static/saml-metadata.xml 사용할 수 있습니다. 다음 단계는 IdP의 SAML 어설션 응답을 구성하여에 AWS 필요한 클레임을 채우는 것입니다. 클레임 구성에 대한 자세한 내용은 인증 응답을 위한 SAML 어설션 구성을 참조하세요.

SAML IdP가 SAML 메타데이터에 서명 인증서를 두 개 이상 포함하는 경우, 로그인 시 ID 풀은 SAML 어설션이 SAML 메타데이터에 있는 인증서와 일치할 경우 유효하다고 판단합니다.

SAML로 사용자 역할 사용자 정의

Amazon Cognito Identity와 함께 SAML을 사용하면 최종 사용자에 맞게 역할을 사용자 지정할 수 있습니다. Amazon Cognito는 향상된 흐름을 SAML 기반 IdP를 통해서만 지원합니다. 자격 증명 풀에서 SAML 기반 IdP를 사용할 수 있도록 인증된 역할 또는 인증되지 않은 역할을 지정할 필요는 없습니다. `https://aws.amazon.com/SAML/Attributes/Role` 클레임 속성은 쉼표로 구분된 하나 이상의 역할 및 공급자 ARN 쌍을 지정합니다. 이는 사용자가 수임할 수 있는 역할입니다. SAML IdP가 IdP에서 얻을 수 있는 사용자 속성 정보를 기반으로 역할 속성을 채우도록 구성할 수 있습니다. SAML 어설션에서 여러 역할을 받으면 `getCredentialsForIdentity`를 호출하는 동안 선택적 `customRoleArn` 파라미터를 채웁니다. 역할이 SAML 어설션의 클레임에 있는 역할과 일치하는 경우 사용자가 이 `customRoleArn`을 수임합니다.

SAML IdP를 사용하여 사용자 인증

SAML 기반 IdP와 페더레이션하려면 사용자가 login. AWS federation에서 IdP 시작 로그인을 사용하는 URL을 결정합니다. AD FS 2.0에서 URL 형식은 `https://<fqdn>/adfs/ls/IdpInitiatedSignOn.aspx?loginToRp=urn:amazon:webservices`입니다.

Amazon Cognito에서 SAML IdP에 대한 지원을 추가하려면 먼저 iOS나 Android 애플리케이션에서 SAML 자격 증명 공급자로 사용자를 인증합니다. SAML IdP와 통합하고 인증하는 데 사용하는 코드는 SAML 공급자에 따라 다릅니다. 사용자가 인증된 후에 Amazon Cognito API를 사용하여 Amazon Cognito Identity에 결과 SAML 어설션을 제공할 수 있습니다.

자격 증명 풀 API 요청의`Logins` 맵에서 SAML 어설션을 반복하거나 재생할 수 없습니다. 재생된 SAML 어설션에는 이전 API 요청의 ID와 중복되는 어설션 ID가 있습니다. `Logins` 맵에서 SAML 어설션을 수락할 수 있는 API 작업에는 GetID, GetCredentialsForIdentity, GetOpenIdToken, 및 GetOpenIDTokenForDeveloperIdentity이 포함됩니다. 자격 증명 풀 인증 흐름에서 API 요청당 SAML 어설션 ID를 한 번 재생할 수 있습니다. 예를 들어 `GetId` 요청과 후속 `GetCredentialsForIdentity` 요청에는 동일한 SAML 어설션을 제공할 수 있지만 두 번째 `GetId` 요청에서는 제공할 수 없습니다.

개발자 인증 자격 증명

Amazon Cognito는 Facebook을 ID 풀 IdP로 설정, Google을 ID 풀 IdP로 설정, Amazon을 사용하여 ID 풀 IdP로 로그인 설정, Sign in with Apple을 ID 풀 IdP로 설정을 통한 웹 아이덴티티 페더레이션 외에 개발자 인증 자격 증명을 지원합니다. 개발자 인증 자격 증명을 사용하면 Amazon Cognito를 사용하여 사용자 데이터를 동기화하고 AWS 리소스에 액세스하는 동시에 기존 인증 프로세스를 통해 사용자를 등록하고 인증할 수 있습니다. 개발자 인증 자격 증명을 사용하려면 최종 사용자 디바이스, 인증을 위

한 백엔드 및 Amazon Cognito 간의 상호 작용이 필요합니다. 자세한 내용은 AWS 블로그의 Amazon Cognito 인증 파트 2: 개발자 인증 자격 증명 이해를 참조하세요.

인증 흐름 이해

GetOpenIdTokenForDeveloperIdentity API 작업은 향상된 인증과 기본 인증 모두에 대한 개발자 인증을 시작할 수 있습니다. 이 API는 관리 자격 증명으로 요청을 인증합니다. `Logins` 맵은 사용자 지정 식별자와 쌍을 이루는 `login.mydevprovider`와 같은 ID 풀 개발자 공급자 이름입니다.

예제:

```
"Logins": {
        "login.mydevprovider": "my developer identifier"
    }
```

향상된 인증

키가에서 반환한 토큰의 발급자이고 `GetOpenIdTokenForDeveloperIdentity`값이 토큰 자체인 `Logins` 맵을 사용하여 GetCredentialsForIdentity API 작업을 호출합니다.

발급자(및 필요한 로그인 키)는 자격 증명 풀의 리전에 따라 다릅니다. 리전 및 토큰 발급자의 전체 목록은 리전별 토큰 발급자 섹션을 참조하세요.

글로벌 발급자를 사용하는 리전의 자격 증명 풀 예제:

```
"Logins": {
        "cognito-identity.amazonaws.com": "eyJra12345EXAMPLE"
    }
```

최신 리전(예: eu-south-2)의 자격 증명 풀 예제:

```
"Logins": {
        "cognito-identity.eu-south-2.amazonaws.com": "eyJra12345EXAMPLE"
    }
```

> **Important**
>
> 로그인 맵 키는 토큰의 `iss` (발급자) 클레임과 정확히 일치해야 합니다. 일치하지 않으면 가 다음을 `GetCredentialsForIdentity` 반환합니다. `NotAuthorizedException: Invalid login token. Issuer doesn't match providerName`

> **Tip**
>
> 프로그래밍 방식으로 올바른 로그인 키를 확인하려면 토큰(JWT)을 디코딩하고 `iss` 클레임 값을 읽습니다. 해당 값을 로그인 맵 키로 사용합니다.

개발자 인증 ID가 있는 `GetCredentialsForIdentity`는 ID 풀의 기본 인증 역할에 대한 임시 자격 증명을 반환합니다.

기본 인증

AssumeRoleWithWebIdentity API 작업을 호출하고 적절한 신뢰 관계가 정의된 IAM 역할의 `RoleArn`을 요청합니다. `WebIdentityToken`의 값을 `GetOpenIdTokenForDeveloperIdentity`에서 가져온 토큰으로 설정합니다.

개발자 인증 ID 인증 흐름 및 외부 공급자 ID와 어떻게 다른지에 대한 자세한 내용은 자격 증명 풀 인증 흐름 섹션을 참조하세요.

개발자 공급자 이름 정의 및 자격 증명 풀에 연결

개발자 인증 자격 증명을 사용하려면 개발자 공급자에 연결된 자격 증명 풀이 필요합니다. 이렇게 하려면 다음 단계를 따릅니다.

사용자 지정 개발자 공급자를 추가하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 사용자 액세스 탭을 선택합니다.
3. ID 제공업체 추가를 선택합니다.
4. 사용자 지정 개발자 공급자를 선택합니다.
5. 개발자 공급자 이름을 입력합니다. 개발자 공급자는 추가 후에는 변경 또는 제거할 수 없습니다.
6. 변경 사항 저장을 선택합니다.

참고: 공급자 이름을 설정하면 해당 이름을 변경할 수 없습니다.

자격 증명 공급자 구현

Android

개발자 인증 자격 증명을 사용하려면 AWSAbstractCognitoIdentityProvider를 확장하는 고유한 ID 제공업체 클래스를 구현합니다. 자격 증명 공급자 클래스는 토큰이 속성으로 포함된 응답 객체를 반환해야 합니다.

다음은 ID 제공업체의 기본 예입니다.

```
public class DeveloperAuthenticationProvider extends
 AWSAbstractCognitoDeveloperIdentityProvider {

  private static final String developerProvider = "<Developer_provider_name>";

  public DeveloperAuthenticationProvider(String accountId, String identityPoolId,
 Regions region) {
    super(accountId, identityPoolId, region);
    // Initialize any other objects needed here.
  }

  // Return the developer provider name which you choose while setting up the
  // identity pool in the &COG; Console

  @Override
  public String getProviderName() {
    return developerProvider;
  }

  // Use the refresh method to communicate with your backend to get an
  // identityId and token.

  @Override
  public String refresh() {

    // Override the existing token
    setToken(null);

    // Get the identityId and token by making a call to your backend
    // (Call to your backend)

    // Call the update method with updated identityId and token to make sure
    // these are ready to be used from Credentials Provider.
```

```
    update(identityId, token);
    return token;

  }

  // If the app has a valid identityId return it, otherwise get a valid
  // identityId from your backend.

  @Override
  public String getIdentityId() {

    // Load the identityId from the cache
    identityId = cachedIdentityId;

    if (identityId == null) {
       // Call to your backend
    } else {
       return identityId;
    }

  }
}
```

이 자격 증명 공급자를 사용하려면 해당 공급자를 CognitoCachingCredentialsProvider에 전달해야 합니다. 다음은 그 예입니다.

```
DeveloperAuthenticationProvider developerProvider = new
 DeveloperAuthenticationProvider( null, "IDENTITYPOOLID", Regions.USEAST1);
CognitoCachingCredentialsProvider credentialsProvider = new
 CognitoCachingCredentialsProvider( context, developerProvider, Regions.USEAST1);
```

> **Note**
>
> 이전 버전의 AWS SDK for Android(2.x 이하)는 cognito-identity.amazonaws.com를 개발자 인증 자격 증명의 내부 공급자 이름으로 사용합니다. 자격 증명 풀이 리전화된 발급자를 사용하는 최신 리전에 있는 경우 현재 SDK 버전으로 업그레이드합니다. 자세한 내용은 리전 토큰 발급자 형식 섹션을 참조하세요.

iOS - Objective-C

개발자 인증 자격 증명을 사용하려면 AWSCognitoCredentialsProviderHelper를 확장하는 자체 ID 제공업체 클래스를 구현합니다. 자격 증명 공급자 클래스는 토큰이 속성으로 포함된 응답 객체를 반환해야 합니다.

```
@implementation DeveloperAuthenticatedIdentityProvider
/*
 * Use the token method to communicate with your backend to get an
 * identityId and token.
 */

- (AWSTask <NSString*> *) token {
    //Write code to call your backend:
    //Pass username/password to backend or some sort of token to authenticate user
    //If successful, from backend call getOpenIdTokenForDeveloperIdentity with logins
 map
    //containing "your.provider.name":"enduser.username"
    //Return the identity id and token to client
    //You can use AWSTaskCompletionSource to do this asynchronously

    // Set the identity id and return the token
    self.identityId = response.identityId;
    return [AWSTask taskWithResult:response.token];
}

@end
```

이 자격 증명 공급자를 사용하려면 다음 예에 표시된 대로 해당 공급자를 `AWSCognitoCredentialsProvider`에 전달합니다.

```
DeveloperAuthenticatedIdentityProvider * devAuth =
 [[DeveloperAuthenticatedIdentityProvider alloc]
 initWithRegionType:AWSRegionYOUR_IDENTITY_POOL_REGION
                                         identityPoolId:@"YOUR_IDENTITY_POOL_ID"
                                        useEnhancedFlow:YES
                                identityProviderManager:nil];
AWSCognitoCredentialsProvider *credentialsProvider = [[AWSCognitoCredentialsProvider
 alloc]

 initWithRegionType:AWSRegionYOUR_IDENTITY_POOL_REGION
                                                        identityProvider:devAuth];
```

미인증 자격 증명과 개발자 인증 자격 증명을 모두 지원하려면 `AWSCognitoCredentialsProviderHelper` 구현에서 `logins` 메서드를 무시합니다.

```
- (AWSTask<NSDictionary<NSString *, NSString *> *> *)logins {
    if(/*logic to determine if user is unauthenticated*/) {
        return [AWSTask taskWithResult:nil];
    }else{
        return [super logins];
    }
}
```

개발자 인증 자격 증명 및 소셜 공급자를 지원하려면 `AWSCognitoCredentialsProviderHelper`의 `logins` 구현에서 현재 공급자가 누구인지 관리해야 합니다.

```
- (AWSTask<NSDictionary<NSString *, NSString *> *> *)logins {
    if(/*logic to determine if user is unauthenticated*/) {
        return [AWSTask taskWithResult:nil];
    }else if (/*logic to determine if user is Facebook*/){
        return [AWSTask taskWithResult: @{ AWSIdentityProviderFacebook :
 [FBSDKAccessToken currentAccessToken] }];
    }else {
        return [super logins];
    }
}
```

> **Note**
>
> 이전 버전의 iOS용 AWS SDK(2.x 이하)는 `cognito-identity.amazonaws.com`를 개발자 인증 자격 증명의 내부 공급자 이름으로 사용합니다. 자격 증명 풀이 리전화된 발급자를 사용하는 최신 리전에 있는 경우 현재 SDK 버전으로 업그레이드합니다. 자세한 내용은 리전 토큰 발급자 형식 섹션을 참조하세요.

iOS - Swift

개발자 인증 자격 증명을 사용하려면 AWSCognitoCredentialsProviderHelper를 확장하는 자체 ID 제공업체 클래스를 구현합니다. 자격 증명 공급자 클래스는 토큰이 속성으로 포함된 응답 객체를 반환해야 합니다.

```
import AWSCore
/*
 * Use the token method to communicate with your backend to get an
 * identityId and token.
 */
class DeveloperAuthenticatedIdentityProvider : AWSCognitoCredentialsProviderHelper {
    override func token() -> AWSTask<NSString> {
    //Write code to call your backend:
    //pass username/password to backend or some sort of token to authenticate user, if
 successful,
    //from backend call getOpenIdTokenForDeveloperIdentity with logins map containing
 "your.provider.name":"enduser.username"
    //return the identity id and token to client
    //You can use AWSTaskCompletionSource to do this asynchronously

    // Set the identity id and return the token
    self.identityId = resultFromAbove.identityId
    return AWSTask(result: resultFromAbove.token)
}
```

이 자격 증명 공급자를 사용하려면 다음 예에 표시된 대로 해당 공급자를 AWSCognitoCredentialsProvider에 전달합니다.

```
let devAuth =
 DeveloperAuthenticatedIdentityProvider(regionType: .YOUR_IDENTITY_POOL_REGION,
 identityPoolId: "YOUR_IDENTITY_POOL_ID", useEnhancedFlow: true,
 identityProviderManager:nil)
let credentialsProvider =
 AWSCognitoCredentialsProvider(regionType: .YOUR_IDENTITY_POOL_REGION,
 identityProvider:devAuth)
let configuration = AWSServiceConfiguration(region: .YOUR_IDENTITY_POOL_REGION,
 credentialsProvider:credentialsProvider)
AWSServiceManager.default().defaultServiceConfiguration = configuration
```

미인증 자격 증명과 개발자 인증 자격 증명을 모두 지원하려면 AWSCognitoCredentialsProviderHelper 구현에서 logins 메서드를 무시합니다.

```
override func logins () -> AWSTask<NSDictionary> {
    if(/*logic to determine if user is unauthenticated*/) {
        return AWSTask(result:nil)
    }else {
        return super.logins()
```

```
    }
}
```

개발자 인증 자격 증명 및 소셜 공급자를 지원하려면 `AWSCognitoCredentialsProviderHelper`의 `logins` 구현에서 현재 공급자가 누구인지 관리해야 합니다.

```
override func logins () -> AWSTask<NSDictionary> {
    if(/*logic to determine if user is unauthenticated*/) {
        return AWSTask(result:nil)
    }else if (/*logic to determine if user is Facebook*/){
        if let token = AccessToken.current?.authenticationToken {
            return AWSTask(result: [AWSIdentityProviderFacebook:token])
        }
        return AWSTask(error:NSError(domain: "Facebook Login", code: -1 , userInfo:
 ["Facebook" : "No current Facebook access token"]))
    }else {
        return super.logins()
    }
}
```

JavaScript

백엔드에서 자격 증명 ID와 세션 토큰을 가져오면에서 `fromCognitoIdentity` 공급자에게 전달합니다`@aws-sdk/credential-providers`. 로그인 키는 토큰의 `iss` 클레임과 일치해야 합니다. 글로벌 발급자를 사용하는 리전의 자격 증명 풀의 경우를 사용합니다`cognito-identity.amazonaws.com`. 다른 리전의 자격 증명 풀의 경우를 사용합니다`cognito-identity.`*`region`*`.amazonaws.com`. 자세한 내용은 리전 토큰 발급자 형식 섹션을 참조하세요.

```
import { fromCognitoIdentity } from "@aws-sdk/credential-providers";

const token = 'TOKEN_RETURNED_FROM_YOUR_PROVIDER';
const region = 'IDENTITY_POOL_REGION';

// Extract the issuer from the token
const payload = token.split('.')[1];
const claims = JSON.parse(Buffer.from(payload, 'base64url').toString());
const providerName = claims.iss;

// Your backend already established the identity, so use fromCognitoIdentity
// (which calls GetCredentialsForIdentity directly for the identity ID you pass)
```

```
// rather than fromCognitoIdentityPool (which would call GetId and resolve a new
 identity).
const credentials = fromCognitoIdentity({
   identityId: 'IDENTITY_ID_RETURNED_FROM_YOUR_PROVIDER',
   logins: {
      [providerName]: token
   },
   clientConfig: { region }
});

// Attach the provider to each service client that needs credentials.
const client = new SomeServiceClient({ region, credentials });
```

로그인 맵 업데이트(Android 및 iOS 전용)

Android

인증 시스템을 통해 사용자를 성공적으로 인증한 후 개발자 공급자 이름 및 개발자 사용자 식별자로 로그인 맵을 업데이트합니다. 이것은 인증 시스템에서 사용자를 고유하게 식별하는 영숫자 문자열입니다. `refresh`가 변경되었을 수 있으므로 로그인 맵을 업데이트한 후 `identityId` 메서드를 호출해야 합니다.

```
HashMap<String, String> loginsMap = new HashMap<String, String>();
loginsMap.put(developerAuthenticationProvider.getProviderName(),
 developerUserIdentifier);

credentialsProvider.setLogins(loginsMap);
credentialsProvider.refresh();
```

iOS - Objective-C

iOS SDK는 자격 증명이 없거나 만료된 경우 `logins` 메서드만 호출하여 최신 로그인 맵을 가져옵니다. SDK에서 새 자격 증명을 얻도록 강제하려면(예를 들어 최종 사용자가 미인증 상태에서 인증 상태로 변경되었으며 인증된 사용자에 대한 자격 증명을 원하는 경우) `credentialsProvider`에 대해 `clearCredentials`를 호출하세요.

```
[credentialsProvider clearCredentials];
```

iOS - Swift

iOS SDK는 자격 증명이 없거나 만료된 경우 `logins` 메서드만 호출하여 최신 로그인 맵을 가져옵니다. SDK에서 새 자격 증명을 얻도록 강제하려면(예: 최종 사용자가 미인증 상태에서 인증 상

태로 변경되었으며 인증된 사용자에 대한 자격 증명을 원하는 경우) `clearCredentials`에 대해 `credentialsProvider`를 호출하세요.

```
credentialsProvider.clearCredentials()
```

토큰 가져오기(서버 측)

GetOpenIdTokenForDeveloperIdentity를 호출하여 토큰을 받습니다. 이 API는 AWS 개발자 자격 증명을 사용하여 백엔드에서 호출해야 합니다. 클라이언트 SDK에서 호출해서는 안 됩니다. API는 Cognito 자격 증명 풀 ID, ID 제공업체 이름이 키로, 식별자가 값으로 포함된 로그인 맵 및 필요할 경우 Cognito 자격 증명 ID를 받습니다(예를 들어 미인증 사용자를 인증된 사용자로 변경하는 경우). 식별자는 사용자의 사용자 이름, 이메일 주소 또는 숫자 값일 수 있습니다. API는 사용자에 대한 고유한 Cognito ID와 최종 사용자에 대한 OpenID Connect 토큰을 사용하여 호출에 응답합니다.

`GetOpenIdTokenForDeveloperIdentity`에 의해 반환된 토큰에 대해 유의해야 할 몇 가지 사항은 다음과 같습니다.

- 캐시할 수 있도록 토큰에 대한 사용자 지정 만료 시간을 지정할 수 있습니다. 사용자 지정 만료 시간을 제공하지 않으면 토큰은 15분 동안 유효합니다.
- 설정할 수 있는 최대 토큰 지속 시간은 24시간입니다.
- 토큰 지속 시간의 증가에 따른 보안 영향을 주의하세요. 공격자는이 토큰을 획득하면 토큰 기간 동안 최종 사용자의 AWS 자격 증명으로 교환할 수 있습니다.

다음 Java 조각은 Amazon Cognito 클라이언트를 초기화하고 개발자 인증 자격 증명의 토큰을 검색하는 방법을 보여 줍니다.

```
// authenticate your end user as appropriate
// ....

// if authenticated, initialize a cognito client with your AWS developer credentials
AmazonCognitoIdentity identityClient = new AmazonCognitoIdentityClient(
  new BasicAWSCredentials("access_key_id", "secret_access_key")
);

// create a new request to retrieve the token for your end user
GetOpenIdTokenForDeveloperIdentityRequest request =
  new GetOpenIdTokenForDeveloperIdentityRequest();
request.setIdentityPoolId("YOUR_COGNITO_IDENTITY_POOL_ID");
```

```
request.setIdentityId("YOUR_COGNITO_IDENTITY_ID"); //optional, set this if your client
 has an
                                                    //identity ID that you want to link
 to this
                                                    //developer account

// set up your logins map with the username of your end user
HashMap<String,String> logins = new HashMap<>();
logins.put("YOUR_IDENTITY_PROVIDER_NAME","YOUR_END_USER_IDENTIFIER");
request.setLogins(logins);

// optionally set token duration (in seconds)
request.setTokenDuration(60 * 15l);
GetOpenIdTokenForDeveloperIdentityResult response =
  identityClient.getOpenIdTokenForDeveloperIdentity(request);

// obtain identity id and token to return to your client
String identityId = response.getIdentityId();
String token = response.getToken();

//code to return identity id and token to client
//...
```

앞의 단계에 따라 앱에 개발자 인증 자격 증명을 통합할 수 있어야 합니다. 문제나 질문이 있는 경우 포럼에 게시하세요.

리전 토큰 발급자 형식

에서 반환된 OpenID Connect 토큰에는 `iss` (발급자) 클레임이 `GetOpenIdTokenForDeveloperIdentity` 포함되어 있습니다. 이 클레임의 형식은 자격 증명 풀의 리전에 따라 다릅니다 AWS .

`GetCredentialsForIdentity` (향상된 흐름)를 호출할 때 로그인 맵 키는 토큰의 `iss` 클레임과 정확히 일치해야 합니다.

리전별 토큰 발급자

토큰 발급자 값은 자격 증명 풀의 리전에 따라 다릅니다.

- 글로벌 발급자를 사용하는 리전(us-east-1, us-east-2, us-west-1, us-west-2, ap-northeast-1, ap-northeast-2, ap-northeast-3, ap-south-1, ap-southeast-1, ap-southeast-2, ca-central-1, eu-central-1, eu-north-1, eu-west-1, eu-west-2, eu-west-3, sa-east-1): `cognito-identity.amazonaws.com`

- 중국 리전(cn-north-1): `cognito-identity.cn-north-1.amazonaws.com.cn`
- 나머지 모든 상용 AWS 리전: `cognito-identity.region.amazonaws.com`

프로그래밍 방식으로 로그인 키 확인

로그인 키를 하드코딩하는 대신 토큰에서 추출할 수 있습니다.

Java

```
import java.util.Base64;
import org.json.JSONObject;

String[] parts = token.split("\\.");
String payload = new String(Base64.getUrlDecoder().decode(parts[1]));
String issuer = new JSONObject(payload).getString("iss");
// Use 'issuer' as the logins map key in GetCredentialsForIdentity
```

Python

```
import base64, json

payload = token.split('.')[1]
# Add padding if needed
payload += '=' * (4 - len(payload) % 4)
claims = json.loads(base64.urlsafe_b64decode(payload))
issuer = claims['iss']
# Use 'issuer' as the logins map key
```

JavaScript

```
const payload = token.split('.')[1];
const claims = JSON.parse(Buffer.from(payload, 'base64url').toString());
const issuer = claims.iss;
// Use 'issuer' as the logins map key
```

마이그레이션 고려 사항

자격 증명 풀을 리전화된 발급자를 사용하는 리전으로 마이그레이션하는 경우 다음 사항에 유의하세요.

1. 로그인 키cognito-identity.amazonaws.com로 하드코딩되는 클라이언트는 새 리전의 자격 증명 풀에 대해 실패합니다.
2. 리전화된 발급자(cognito-identity.*new-region*.amazonaws.com)를 사용하거나 토큰에서 iss 클레임을 동적으로 읽으려면(권장) 클라이언트 코드를 업데이트해야 합니다.
3. 이전 버전의 AWS Mobile SDKs 하드코딩합니다. 여기에는 Android 2.x 이하용 AWS SDK와 iOS 2.x 이하용 AWS SDK가 포함됩니다. 자격 증명 풀에서 리전화된 발급자를 사용하는 경우 현재 SDK 버전으로 업그레이드합니다.

기존 소셜 자격 증명에 연결

개발자 인증 자격 증명을 사용하는 경우 모든 공급자 연결은 백엔드에서 수행해야 합니다. 사용자의 소셜 자격 증명(Login with Amazon, Apple, Facebook 또는 Google로 로그인)에 사용자 지정 자격 증명을 연결하려면 GetOpenIdTokenForDeveloperIdentity를 호출할 때 로그인 맵에 자격 증명 공급자 토큰을 추가하세요. 이를 위해 클라이언트 SDK에서 백엔드를 호출하여 최종 사용자를 인증할 때 최종 사용자의 소셜 공급자 토큰을 추가로 전달합니다.

예를 들어, 사용자 지정 자격 증명을 Facebook에 연결하려고 시도할 경우 GetOpenIdTokenForDeveloperIdentity를 호출할 때 로그인 맵에 자격 증명 공급자 식별자 이외에 Facebook 토큰도 추가합니다.

```
logins.put("YOUR_IDENTITY_PROVIDER_NAME","YOUR_END_USER_IDENTIFIER");
logins.put("graph.facebook.com","END_USERS_FACEBOOK_ACCESSTOKEN");
```

공급자 간 전환 지원

Android

애플리케이션에는 개발자 인증 자격 증명과 함께 퍼블릭 공급자(Login with Amazon, Sign in with Apple, Facebook 또는 Google)를 사용한 미인증 자격 증명 또는 인증 자격 증명 지원이 필요할 수 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 시스템에 연락하는 대신 Amazon Cognito와 직접 상호 작용합니다. 따라서 모바일 애플리케이션은 앱 사용자가 선택한 사항에 따라 두 개의 고유 흐름을 지원할 수 있어야 합니다. 이를 위해 사용자 지정 ID 제공업체를 일부 변경해야 합니다.

refresh 메서드는 로그인 맵을 확인합니다. 맵이 비어 있지 않고 개발자 공급자 이름의 키가 있는 경우 백엔드를 호출합니다. 그렇지 않으면 getIdentityId 메서드를 호출하고 null을 반환합니다.

```
public String refresh() {

   setToken(null);

   // If the logins map is not empty make a call to your backend
   // to get the token and identityId
   if (getProviderName() != null &&
      !this.loginsMap.isEmpty() &&
      this.loginsMap.containsKey(getProviderName())) {

      /**
       * This is where you would call your backend
       **/

      // now set the returned identity id and token in the provider
      update(identityId, token);
      return token;

   } else {
      // Call getIdentityId method and return null
      this.getIdentityId();
      return null;
   }
}
```

마찬가지로 getIdentityId 메서드에는 로그인 맵의 내용에 따라 두 개의 흐름이 있습니다.

```
public String getIdentityId() {

   // Load the identityId from the cache
   identityId = cachedIdentityId;

   if (identityId == null) {

      // If the logins map is not empty make a call to your backend
      // to get the token and identityId

      if (getProviderName() != null && !this.loginsMap.isEmpty()
         && this.loginsMap.containsKey(getProviderName())) {

         /**
           * This is where you would call your backend
          **/
```

```
            // now set the returned identity id and token in the provider
            update(identityId, token);
            return token;

        } else {
            // Otherwise call &COG; using getIdentityId of super class
            return super.getIdentityId();
        }

    } else {
        return identityId;
    }

}
```

iOS - Objective-C

애플리케이션에는 개발자 인증 자격 증명과 함께 퍼블릭 공급자(Login with Amazon, Sign in with Apple, Facebook 또는 Google)를 사용한 미인증 자격 증명 또는 인증 자격 증명 지원이 필요할 수 있습니다. 이렇게 하려면 현재 자격 증명 공급자에 따라 정확한 로그인 맵을 반환할 수 있도록 AWSCognitoCredentialsProviderHelper `logins` 메서드를 무시하세요. 이 예는 미인증, Facebook, 개발자 인증 간 피벗할 수 있는 방법을 보여 줍니다.

```
- (AWSTask<NSDictionary<NSString *, NSString *> *> *)logins {
    if(/*logic to determine if user is unauthenticated*/) {
        return [AWSTask taskWithResult:nil];
    }else if (/*logic to determine if user is Facebook*/){
        return [AWSTask taskWithResult: @{ AWSIdentityProviderFacebook :
 [FBSDKAccessToken currentAccessToken] }];
    }else {
        return [super logins];
    }
}
```

미인증 상태에서 인증 상태로 전환하면 `[credentialsProvider clearCredentials];`를 호출하여 SDK에서 새 인증 자격 증명을 얻도록 강제해야 합니다. 두 인증 공급자 간에 전환하고 두 공급자를 연결하려고 하지 않는 경우(예를 들어 로그인 딕셔너리에서 여러 공급자에 대한 토큰을 제공하지 않는 경우) `[credentialsProvider clearKeychain];`을 호출해야 합니다. 이렇게 하면 자격 증명과 자격 증명이 모두 지워지고 SDK가 새 자격 증명을 가져오도록 강제됩니다.

iOS - Swift

애플리케이션에는 개발자 인증 자격 증명과 함께 퍼블릭 공급자(Login with Amazon, Sign in with Apple, Facebook 또는 Google)를 사용한 미인증 자격 증명 또는 인증 자격 증명 지원이 필요할 수 있습니다. 이렇게 하려면 현재 자격 증명 공급자에 따라 정확한 로그인 맵을 반환할 수 있도록 AWSCognitoCredentialsProviderHelper `logins` 메서드를 무시하세요. 이 예는 미인증, Facebook, 개발자 인증 간 피벗할 수 있는 방법을 보여 줍니다.

```
override func logins () -> AWSTask<NSDictionary> {
    if(/*logic to determine if user is unauthenticated*/) {
        return AWSTask(result:nil)
    }else if (/*logic to determine if user is Facebook*/){
        if let token = AccessToken.current?.authenticationToken {
            return AWSTask(result: [AWSIdentityProviderFacebook:token])
        }
        return AWSTask(error:NSError(domain: "Facebook Login", code: -1 , userInfo:
 ["Facebook" : "No current Facebook access token"]))
    }else {
        return super.logins()
    }
}
```

미인증 상태에서 인증 상태로 전환하면 `credentialsProvider.clearCredentials()`를 호출하여 SDK에서 새 인증 자격 증명을 얻도록 강제해야 합니다. 두 인증 공급자 간에 전환하고 두 공급자를 연결하려고 하지 않는 경우(즉, 로그인 딕셔너리에서 여러 공급자에 대한 토큰을 제공하지 않는 경우) `credentialsProvider.clearKeychain()`을 호출해야 합니다. 이렇게 하면 자격 증명과 자격 증명이 모두 지워지고 SDK가 새 자격 증명을 가져오도록 강제됩니다.

Unity

애플리케이션에는 개발자 인증 자격 증명과 함께 퍼블릭 공급자(Login with Amazon, Sign in with Apple, Facebook 또는 Google)를 사용한 미인증 자격 증명 또는 인증 자격 증명 지원이 필요할 수 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 시스템에 연락하는 대신 Amazon Cognito와 직접 상호 작용합니다. 모바일 애플리케이션은 앱 사용자가 선택한 사항에 따라 두 개의 고유 흐름을 지원할 수 있어야 합니다. 이를 위해 사용자 지정 자격 증명 공급자를 일부 변경해야 합니다.

Unity에서 이를 수행하는 권장 방법은 AbstractCognitoIdentityProvider 대신 AmazonCognitoEnhancedIdentityProvide에서 ID 제공업체를 확장하고 사용자가 자체 백

엔드로 인증되지 않은 경우 자체 대신 상위 RefreshAsync 메서드를 호출하는 것입니다. AbstractCognitoIdentityProvider 사용자가 인증 상태이면 이전에 설명한 것과 동일한 흐름을 사용할 수 있습니다.

인증되지 않은 사용자를 인증된 사용자로 전환

Amazon Cognito 자격 증명 풀은 인증된 사용자와 인증되지 않은 사용자를 모두 지원합니다. 인증되지 않은 사용자는 자격 증명 공급자(IdPs로 로그인하지 않은 경우에도 AWS 리소스에 액세스할 수 있습니다. 이러한 수준의 액세스는 사용자가 로그인하기 전에 사용자에게 콘텐츠를 표시하는 데 유용합니다. 인증되지 않은 각 사용자는 개별적으로 로그인 및 인증되지 않았더라도 자격 증명 풀에 고유한 자격 증명이 있습니다.

이 섹션에서는 사용자가 인증되지 않은 자격 증명으로 로그인하는 것에서 인증된 자격 증명을 사용하는 것으로 전환하는 경우를 설명합니다.

Android

사용자는 애플리케이션에 인증되지 않은 게스트로 로그인할 수 있습니다. 결국 지원되는 IdPs. Amazon Cognito는 이전 자격 증명이 새 자격 증명과 동일한 고유 식별자를 유지하고, 프로파일 데이터가 자동으로 병합되도록 합니다.

`IdentityChangedListener` 인터페이스의 프로파일 병합을 통해 애플리케이션에 알립니다. 인터페이스에 `identityChanged` 메서드를 구현하여 이러한 메시지를 수신할 수 있습니다.

```
@override
public void identityChanged(String oldIdentityId, String newIdentityId) {
    // handle the change
}
```

iOS - Objective-C

사용자는 애플리케이션에 인증되지 않은 게스트로 로그인할 수 있습니다. 결국 지원되는 IdPs. Amazon Cognito는 이전 자격 증명이 새 자격 증명과 동일한 고유 식별자를 유지하고, 프로파일 데이터가 자동으로 병합되도록 합니다.

`NSNotificationCenter`는 애플리케이션에 프로파일 병합을 알립니다.

```
[[NSNotificationCenter defaultCenter] addObserver:self
                                      selector:@selector(identityIdDidChange:)
                                      name:AWSCognitoIdentityIdChangedNotification
```

```
                                        object:nil];

-(void)identityDidChange:(NSNotification*)notification {
    NSDictionary *userInfo = notification.userInfo;
    NSLog(@"identity changed from %@ to %@",
        [userInfo objectForKey:AWSCognitoNotificationPreviousId],
        [userInfo objectForKey:AWSCognitoNotificationNewId]);
}
```

iOS - Swift

사용자는 애플리케이션에 인증되지 않은 게스트로 로그인할 수 있습니다. 결국 지원되는 IdPs. Amazon Cognito는 이전 자격 증명이 새 자격 증명과 동일한 고유 식별자를 유지하고, 프로파일 데이터가 자동으로 병합되도록 합니다.

`NSNotificationCenter`는 애플리케이션에 프로파일 병합을 알립니다.

```
[NSNotificationCenter.defaultCenter().addObserver(observer: self
   selector:"identityDidChange"
   name:AWSCognitoIdentityIdChangedNotification
   object:nil)

func identityDidChange(notification: NSNotification!) {
  if let userInfo = notification.userInfo as? [String: AnyObject] {
    print("identity changed from: \(userInfo[AWSCognitoNotificationPreviousId])
    to: \(userInfo[AWSCognitoNotificationNewId])")
  }
}
```

JavaScript

처음에 인증되지 않은 사용자

사용자는 일반적으로 인증되지 않은 역할로 시작합니다. 이 역할의 경우 `logins` 속성 없이 `fromCognitoIdentityPool` 자격 증명 공급자를 생성하여 서비스 클라이언트에 연결합니다. 이 경우, 기본 구성은 다음과 같을 수 있습니다.

```

import { fromCognitoIdentityPool } from "@aws-sdk/credential-providers";

// Create a credentials provider without a logins map for the unauthenticated role.
```

```
const region = "us-east-1";
const credentials = fromCognitoIdentityPool({
    identityPoolId: "us-east-1:1699ebc0-7900-4099-b910-2df94f52a030",
    clientConfig: { region }
});

// Attach the provider to each service client that needs credentials.
const client = new SomeServiceClient({ region, credentials });
```

인증된 사용자로 전환

인증되지 않은 사용자가 IdP에 로그인하고 토큰이 있는 경우 토큰이 포함된 `logins` 맵으로 새 `fromCognitoIdentityPool` 공급자를 생성하는 사용자 지정 함수를 호출하여 사용자를 인증되지 않음에서 인증됨으로 전환할 수 있습니다.

```
// Called when an identity provider has a token for a logged in user
function userLoggedIn(providerName, token) {
    // Re-create the provider with a logins map to switch to the authenticated role.
    const credentials = fromCognitoIdentityPool({
        identityPoolId: "us-east-1:1699ebc0-7900-4099-b910-2df94f52a030",
        logins: {
            [providerName]: token
        },
        clientConfig: { region }
    });

    // Re-create any service clients so they pick up the new credentials.
    // SomeServiceClient is a placeholder: import the client for the service you're
 calling
    // (for example, S3Client from @aws-sdk/client-s3).
    const client = new SomeServiceClient({ region, credentials });
}
```

`fromCognitoIdentityPool` 공급자가 각 서비스 클라이언트에 직접 연결되므로 인증되지 않은 역할에서 인증된 역할로 전환하면 `logins` 맵을 사용하여 새 공급자를 생성하고 나중에 생성하는 모든 서비스 클라이언트에 전달해야 합니다. AWS SDK for JavaScript v2와 달리 v3 SDK는 글로벌 구성 객체를 사용하지 않으므로 재설정할 공유 자격 증명이 없습니다.

`fromCognitoIdentityPool` 공급자에 대한 자세한 내용은 AWS SDK for JavaScript v3 보안 인증 공급자를 참조하세요.

Amazon Cognito ID 풀 시작

Amazon Cognito 자격 증명 풀을 통해 사용자 고유의 자격 증명을 생성하고 사용자의 권한을 할당할 수 있습니다. ID 풀은 다음 유형의 인증 서비스에서 ID를 가져올 수 있습니다.

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

사용자가 공급자로 인증하고 자격 증명 풀에 대한 권한을 제시하면 임시 AWS 자격 증명을 얻습니다. 사용자의 자격 증명에는 다른 AWS 서비스에 대한 액세스 권한을 정의한 권한이 있습니다.

주제

- Amazon Cognito에서 자격 증명 풀 생성
- SDK 설정
- 자격 증명 공급자 통합
- 자격 증명 얻기
- ID 풀에 대한 예제 애플리케이션

Amazon Cognito에서 자격 증명 풀 생성

Amazon Cognito 콘솔을 통해 자격 증명 풀을 만들거나 AWS Command Line Interface 또는 Amazon Cognito API를 사용할 수 있습니다. 다음 절차는 콘솔에서 새 ID 풀을 생성하는 일반적인 가이드입니다. 또한 콘솔로 바로 건너뛰고 안내형 환경 및 인라인 도움말 콘텐츠를 따를 수 있습니다.

콘솔에서 새 자격 증명 풀을 만들려면

1. Amazon Cognito 콘솔에 로그인하고 자격 증명 풀을 선택합니다. Amazon Cognito 리소스를 생성하고 관리할 수 있도록 IAM 보안 주체에 권한을 할당하려면 AWS Amazon Cognito에 대한 관리형 정책 섹션을 참조하세요. `AmazonCognitoPowerUser` 정책은 ID 풀을 생성하는 데 충분합니다.
2. 자격 증명 풀 생성을 선택합니다.
3. 자격 증명 풀 신뢰 구성에서 인증된 액세스, 게스트 액세스 또는 둘 다에 대해 자격 증명 풀을 설정하도록 선택합니다.

- 인증된 액세스를 선택한 경우 자격 증명 풀에서 인증된 자격 증명의 소스로 설정하려는 자격 증명 유형을 하나 이상 선택합니다. 사용자 지정 개발자 공급자를 구성하는 경우 자격 증명 풀을 생성한 후에는 이를 수정하거나 삭제할 수 없습니다.

4. 권한 구성에서 자격 증명 풀의 인증된 사용자 또는 게스트 사용자의 기본 IAM 역할을 선택합니다.

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

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

5. ID 제공업체 연결에서 자격 증명 풀 신뢰 구성에서 선택한 ID 제공업체(idP)의 세부 정보를 입력합니다. OAuth 앱 클라이언트 정보를 제공하거나, Amazon Cognito 사용자 풀을 선택하거나, IAM IdP를 선택하거나, 개발자 공급자의 사용자 지정 식별자를 입력하라는 메시지가 표시될 수 있습니다.

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

      i. 규칙으로 역할 선택을 선택한 경우 사용자 인증의 소스 클레임, 클레임을 비교할 연산자, 이 역할 선택과 일치하는 값을 입력하고 역할 할당이 일치할 때 할당할 역할을 입력합니다. 다른 조건에 따라 추가 규칙을 생성하려면 다른 항목 추가를 선택합니다.

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

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

      i. 보안 주체 태그를 적용하지 않으려면 비활성을 선택합니다.

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

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

6. 속성 구성에서 자격 증명 풀 이름 아래에 이름을 입력합니다.
7. 기본(Classic) 인증에서 기본 흐름 활성화 여부를 선택합니다. 기본 흐름이 활성화되면 IdP에 대한 역할 선택을 우회하고 AssumeRoleWithWebIdentity를 직접 호출할 수 있습니다. 자세한 내용은 자격 증명 풀 인증 흐름 섹션을 참조하세요.
8. 자격 증명 풀에 태그를 적용하려면 태그에서 태그 추가를 선택합니다.
9. 검토 및 생성에서 새 자격 증명 풀에 대한 선택 사항을 확인합니다. 편집을 선택하여 마법사로 돌아가서 설정을 변경합니다. 완료하면 자격 증명 풀 생성을 선택합니다.

SDK 설정

Amazon Cognito 자격 증명 풀을 사용하려면 AWS Amplify, AWS SDK for Java또는를 설정합니다 SDK for .NET. 자세한 내용은 다음 항목을 참조하세요.

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

자격 증명 공급자 통합

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

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

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

자체 백엔드 인증 프로세스 통합에 대한 자세한 내용은 개발자 인증 자격 증명 단원을 참조하십시오.

자격 증명 얻기

Amazon Cognito 자격 증명 풀은 게스트(인증되지 않음)인 사용자와 토큰을 인증하고 받은 사용자에게 임시 AWS 자격 증명을 제공합니다. 이러한 AWS 자격 증명을 사용하면 앱이 Amazon API Gateway를 AWS 통해 내부 AWS 또는 외부의 백엔드에 안전하게 액세스할 수 있습니다. 자격 증명 얻기 섹션을 참조하세요.

ID 풀에 대한 예제 애플리케이션

Amazon Cognito 자격 증명 풀의 가장 일반적인 사용 사례는 여러 로그인 시스템의 사용자를 페더레이션하고 액세스가 제한된 임시 AWS 자격 증명을 클라이언트에 직접 제공하는 것입니다. 이렇게 하면 AWS 리소스에 액세스할 수 있는 권한을 위해 자격 증명 브로커를 빌드할 필요가 없습니다. 예를 들어 사용자가 소셜 미디어 계정으로 로그인하고 모바일 애플리케이션을 위해 Amazon S3의 앱 자산에 액세스하도록 해야 할 수 있습니다. 또한 ID 풀은 사용자 풀로 로그인하는 사용자에게 자격 증명을 제공합니다.

이 자습서에서는 ID 풀에서 지원되는 ID 제공업체(idP)를 사용하여 향상된 기본 인증 흐름에서 임시 인증 및 게스트 자격 증명을 얻을 수 있는 웹 애플리케이션을 생성합니다. 웹 개발 경험이 있는 경우 GitHub에서 예시 앱을 다운로드합니다.

GitHub에서 예시 애플리케이션 다운로드

이 예시 애플리케이션은 Amazon Cognito ID 풀의 다음 기능을 보여줍니다.

ID 풀의 인증 흐름

- 세부 API 요청 분석으로 인증 흐름 개선
- 세부 API 요청 내역이 포함된 기본 인증 흐름

게스트(미인증) 액세스 구현

- 로그인하지 않고도 제한된 AWS 서비스 액세스 제공

지원되는 ID 제공업체와의 통합

- 소비자 액세스를 위한 소셜 IdP(Facebook, Amazon, Twitter, Apple 및 Google)
- 기업 사용자를 위한 엔터프라이즈 IdP(OpenID Connect 또는 SAML을 통해)
- Amazon Cognito 사용자 풀

AWS 자격 증명 관리

- ID 제공업체 토큰을 임시 AWS 자격 증명으로 교환

- 임시 자격 증명을 사용하여 AWS 서비스에 안전하게 액세스

개발 웹 서버에서 애플리케이션을 설정하고 브라우저에서 애플리케이션에 액세스하면 다음 옵션이 표시됩니다.



주제

- 사전 조건
- 인증 제공업체 설정
- 데모 애플리케이션 배포
- ID 풀의 인증 방법 살펴보기
- 다음 단계

사전 조건

시작하기 전에 다음 리소스를 구성해야 합니다.

- Amazon Cognito에 액세스할 수 있는 AWS 계정입니다. AWS 계정이 없는 경우의 지침을 따릅니다시작하기 AWS.
- 개발 시스템에 Python 3.11 이상이 설치되어 있습니다.
- GitHub 액세스 권한.
- AWS Amazon Cognito APIs. 이러한 자격 증명은 개발자 인증에 필요합니다.

특정 SDK에서 AWS 자격 증명 및 자격 증명 풀 페더레이션을 구현하는 방법에 대한 자세한 내용은 섹션을 참조하세요the section called “자격 증명 얻기”.

인증 제공업체 설정

이 애플리케이션에서 최상의 결과를 얻으려면 하나 이상의 서드 파티 ID 제공업체(idP) 또는 Amazon Cognito 사용자 풀을 설정하고 Amazon Cognito ID 풀과 통합합니다. 사전 조건을 완료하고 이 데모 애플리케이션을 실행하기 전에 구성할 ID 제공업체를 선택합니다. Amazon Cognito 콘솔은 ID 풀 및 제공업체를 구성하는 프로세스를 안내합니다.

Amazon Cognito 사용자 풀

- the section called “인증”
- the section called “앱 클라이언트”

소셜 ID 제공업체

- Google: the section called “Google”
- Facebook: the section called “Facebook”
- Amazon: the section called “Login with Amazon”

OpenID Connect(OIDC) 공급자

- the section called “OpenID Connect 공급자”

SAML 공급자

- the section called “SAML 자격 증명 공급자”

> **Note**
>
> 이 데모 애플리케이션의 경우 지원되는 모든 ID 제공업체를 설정할 필요가 없습니다. 사용 사례와 일치하는 것으로 시작할 수 있습니다. 각 링크는 자세한 구성 지침을 제공합니다.

데모 애플리케이션 배포

리포지토리 복제

1. 터미널 창을 엽니다.
2. `aws-doc-sdk-examples` 리포지토리를 복제하거나 리포지토리에서 이 폴더를 검색합니다.

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

3. 프로젝트 디렉터리로 이동합니다.

```
cd python/example_code/cognito/scenarios/identity_pools_example_demo/web
```

자격 증명 풀 생성

애플리케이션에 대한 Amazon Cognito ID 풀을 생성하려면 the section called “ID 풀 구성”의 지침을 따릅니다.

데모 애플리케이션의 ID 풀을 구성하려면

1. Amazon Cognito 콘솔을 엽니다.
2. 왼쪽 탐색 메뉴에서 ID 풀을 선택합니다. 기존 ID 풀을 선택하거나 새 ID 풀을 생성합니다.
3. 사용자 액세스에서 인증된 액세스 및 게스트 액세스를 활성화합니다. 새 IAM 역할 또는 기존 IAM 역할을 구성하고 각 사용자 유형에 부여할 권한을 할당합니다.
4. 사용자 액세스에서 구성하려는 ID 제공업체를 설정합니다.
5. ID 풀 속성에서 기본(클래식) 인증을 활성화합니다.
6. 브라우저를 ID 풀의 콘솔에 열어 둡니다. 애플리케이션 설정에서 ID 풀 ID 및 기타 구성 정보를 사용합니다.

애플리케이션 구성 및 실행

다음 단계는 데모 애플리케이션의 초기 설정을 안내합니다.

데모 애플리케이션을 구성하려면

1. `aws-doc-sdk-examples` 복제본에서 `python/example_code/cognito/scenarios/identity_pools_example_demo/web`의 명령줄을 엽니다.
2. 예제 환경 파일을 복사하여 `.env` 파일을 생성합니다.

```
cp .env.example .env
```

3. 텍스트 편집기에서 `.env` 파일을 엽니다. 파일의 예제 값을 사용자 고유의 구성 값으로 대체합니다.

4. 백엔드 종속성을 설치합니다.

```
pip install -r requirements.txt
```

5. 백엔드 서버를 시작합니다.

```
cd backend
python oauth_server.py
```

6. 새 터미널 창을 열고 프로젝트 디렉터리로 이동하여 프런트엔드 서버를 시작합니다.

```
cd frontend
python -m http.server 8001
```

7. http://localhost:8001에서 애플리케이션 브라우저를 엽니다. 브라우저에 자격 증명 풀 인증을 테스트할 준비가 된 데모 애플리케이션 인터페이스가 표시됩니다.

ID 풀의 인증 방법 살펴보기

이 섹션에서는 Amazon Cognito ID 풀 데모 애플리케이션을 사용하여 기본 및 향상된 인증 흐름을 안내합니다. 이 데모에서는 자격 증명 풀이 다양한 자격 증명 공급자와 협력하여 애플리케이션 사용자에게 임시 AWS 자격 증명을 제공하는 방법을 알아봅니다.

예제 애플리케이션의 대화형 데모 섹션에서는 먼저 ID 풀에서 지원하는 두 가지 유형의 액세스 중에서 선택합니다.

인증되지 않은 (게스트) 액세스

아직 인증하지 않은 사용자에게 AWS 자격 증명을 제공합니다.

인증된 액세스

사용 가능한 전체 범위의 권한으로 자격 AWS 증명에 대한 자격 증명 공급자 토큰을 교환합니다. `.env` 파일에 구성한 ID 제공업체 중에서 선택합니다.

인증되지 않은 (게스트) 액세스

이 단계에서는 자격 증명 풀의 게스트 액세스 기능을 통해 인증되지 않은(게스트) 사용자의 임시 AWS 자격 증명을 얻는 방법을 보여줍니다. 데모 애플리케이션에서는 향상된 흐름과 기본 흐름을 모두 테스트하여 사용자 로그인 없이 ID 풀이 자격 증명을 발급하는 방법을 확인합니다. 게스트 액세스는 인증된

액세스와 동일한 API 시퀀스를 사용하지만 ID 제공업체 토큰(예: Google, Facebook의 OAuth 토큰 또는 엔터프라이즈 공급자의 SAML 어설션)은 제공하지 않습니다.

인증 없이 사용자에게 제한된 AWS 액세스 권한을 제공하는 방법에 대한 정보를 찾고 있다면 계속 읽어보세요. 게스트 액세스를 구현한 후에는 익명 사용자에게 보안 AWS 인증을 안전하게 제공하고 두 인증 흐름 간의 차이점을 이해하는 방법을 알아봅니다.

> **Important**
>
> 인증되지 않은 액세스는 인터넷 액세스 권한이 있는 모든 사용자에게 자격 증명을 발급할 수 있으므로 퍼블릭 APIs 및 그래픽 자산과 같이 최소한의 보안이 필요한 AWS 리소스에 가장 적합합니다. 이 단계를 진행하기 전에 게스트 액세스가 활성화된 ID 풀을 구성했는지 확인하고 권한을 제한하는 적절한 IAM 정책이 마련되어 있는지 확인합니다.

Guest access with enhanced flow

향상된 흐름은 두 개의 API 요청이 있는 인증되지 않은 사용자의 AWS 자격 증명을 얻는 간소화된 접근 방식입니다.

향상된 흐름으로 게스트 액세스를 테스트하려면

1. 데모 애플리케이션에서 대화형 데모 섹션으로 이동합니다.
2. 게스트 액세스 탭을 선택합니다.
3. 향상된 흐름 탭을 선택합니다.
4. 게스트 액세스 테스트를 선택합니다.
5. 앱은 추가 인증 프롬프트 없이 자격 증명 풀에서 임시 AWS 자격 증명을 가져옵니다.
6. 인증에 성공하면 결과 패널이 표시되는 웹 인터페이스가 표시되고 탐색할 수 있는 다음과 같은 두 가지 옵션이 있습니다.

   a. 자격 증명만 보기 버튼: API 흐름 세부 정보 없이 생성된 임시 AWS 자격 증명을 직접 보려면이 버튼을 선택합니다.

```
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "Credentials": {
    "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "SessionToken": "IQoJb3JpZ2luX2VjEEXAMPLE......",
```

```
    "Expiration": "2025-08-07T00:58:21-07:00"
  }
}
```

b. 세부 API 흐름 보기 버튼: 단계별 API 요청을 보려면 이 버튼을 선택합니다.

- identityPoolId가 있는 GetId() API 요청. 게스트 액세스에 인증 토큰이 필요하지 않음

```
{
  "IdentityPoolId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
}
```

유효한 경우 사용자의 IdentityID를 찾거나 생성하고 반환합니다. 다음은 응답의 예입니다.

```
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
}
```

- 반환된 identityPoolId가 있는 GetCredentialsForIdentity().

```
POST GetCredentialsForIdentity
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
}
```

Cognito는 게스트 액세스를 검증하고, 인증되지 않은 역할을 내부적으로 수임하고 AWS STS, 임시 AWS 자격 증명을 반환합니다. (이 호출에는 IAM 인증이 없습니다. 역할 신뢰는 cognito-identity-amazonzaws.com을 허용해야 합니다.)

```
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "Credentials": {
    "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "SessionToken": "IQoJb3JpZ2luX2VjEEXAMPLE......",
    "Expiration": "2025-08-07T00:58:21-07:00"
  }
}
```

Guest access with basic flow

기본 흐름은 ID 검색 및 자격 증명 생성을 위한 별도의 API 요청을 통해 인증 프로세스에 대한 세분화된 제어를 제공합니다.

기본 흐름을 사용하여 게스트 액세스를 테스트하려면

1. 데모 애플리케이션에서 대화형 데모 섹션으로 이동합니다.
2. 게스트 액세스 탭을 선택합니다.
3. 기본 흐름 탭을 선택합니다.
4. 게스트 액세스 테스트를 선택합니다.
5. 앱은 추가 인증 프롬프트 없이 자격 증명 풀에서 임시 AWS 자격 증명을 가져옵니다.
6. 인증에 성공하면 결과 패널이 표시되는 웹 인터페이스가 표시되고 탐색할 수 있는 두 가지 옵션이 있습니다.

   a. 자격 증명만 보기 버튼: API 흐름 세부 정보 없이 생성된 임시 AWS 자격 증명을 직접 보려면이 버튼을 선택합니다.

   ```
   {
     "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
     "Credentials": {
       "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
       "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
       "SessionToken": "IQoJb3JpZ2luX2VjEEXAMPLE......",
       "Expiration": "2025-08-12T13:36:17-07:00"
     }
   }
   ```

   b. 세부 API 흐름 보기 버튼: 단계별 API 요청을 보려면 이 버튼을 선택합니다.

   - ID 풀 ID가 있는 `GetId()` API 요청. 게스트 액세스에 인증 토큰이 필요하지 않음.

   ```
   POST GetId
   {
     "IdentityPoolId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
   }
   ```

   유효한 경우 사용자의 `IdentityID`를 찾거나 생성하고 반환합니다. 다음은 응답의 예입니다.

```
{
   "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
}
```

- 반환된 IdentityID와 동일한 Logins 맵이 있는 GetOpenIdToken()

```
POST GetOpenIdToken
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
}
```

응답:

```
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "Token": "eyJraWQiOiJFWAMPLE......"
}
```

이 단계에서 발생하는 작업: Amazon Cognito는 이 IdentityId를 나타내는 cognito-identity.amazonaws.com에서 단기 OpenID Connect 웹 ID 토큰을 발행합니다. 토큰에는 aud(자격 증명 풀 ID) 및 amr(인증 또는 미인증)을 포함하여가 AWS STS 평가하는 OIDC 클레임이 포함됩니다. IAM 역할의 신뢰 정책에는 이러한 클레임이 필요합니다.

- AssumeRoleWithWebIdentity() - 앱이를 AWS STS 직접 호출하여 Amazon Cognito OpenID 토큰을 임시 AWS 자격 증명으로 교환합니다.

```
POST sts:AssumeRoleWithWebIdentity
{
  "RoleArn": "arn:aws:iam::111122223333:role/
Cognito_IdentityPoolUnauth_Role",
  "WebIdentityToken": "eyJraWQiOiJFWAMPLE......"
}
```

응답:

```
{
  "Credentials": {
    "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "SessionToken": "FwoGZXIvYXdzEEXAMPLE......"
```

```
  }
}
```

이 단계에서 발생하는 작업: 검증되면 임시 AWS 자격 증명을 반환합니다.

임시 자격 증명 사용

이러한 임시 자격 증명은 표준 AWS 자격 증명으로 작동하지만 자격 증명 풀의 인증되지 않은 IAM 역할에 정의된 제한된 권한을 갖습니다. AWS SDK 또는와 함께 사용할 수 있습니다 AWS CLI. 자격 증명으로 AWS SDKs 구성하는 방법에 대한 자세한 내용은SDK 및 도구 참조 안내서의 표준화된 자격 증명 공급자를 참조하세요. AWS SDKs

아래 예제는 전체 목록이 아니지만 ID 풀의 게스트 기능이 사용자 경험을 개선할 수 있는 일반적인 방법을 보여줍니다.

퍼블릭, 읽기 전용 콘텐츠

다음 예제에서는 게스트 사용자로 제한된 Amazon S3 액세스를 위해 자격 증명 제공업체를 구성합니다.

Python

```
# Example: Using credentials with boto3
import boto3

# Configure client with temporary credentials
s3_client = boto3.client(
    's3',
    aws_access_key_id='AKIAIOSFODNN7EXAMPLE',
    aws_secret_access_key='wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY',
    aws_session_token='IQoJb3JpZ2luX2VjEEXAMPLE......'
)

# Make API requests within IAM role permissions
response = s3_client.list_objects_v2(Bucket='my-public-bucket')

# Access public content
for obj in response.get('Contents', []):
    print(f"File: {obj['Key']}, Size: {obj['Size']} bytes")
```

JavaScript

```
// Example: Accessing public content
import { S3Client, GetObjectCommand } from "@aws-sdk/client-s3";

const s3Client = new S3Client({
    region: "us-east-1",
    credentials: {
        accessKeyId: 'AKIAIOSFODNN7EXAMPLE',
        secretAccessKey: 'wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY',
        sessionToken: 'IQoJb3JpZ2luX2VjEEXAMPLE......'
    }
});

// Access public images or documents
const response = await s3Client.send(new GetObjectCommand({
    Bucket: 'my-public-content',
    Key: 'product-catalog.pdf'
}));
```

"Try-before-login" 기능

다음 예제에서는 Amazon DynamoDB에 대한 읽기 전용 액세스를 게스트 사용자로 사용합니다.

Python

```
# Example: Limited app functionality for trial users
import boto3

dynamodb = boto3.client(
    'dynamodb',
    aws_access_key_id='AKIAIOSFODNN7EXAMPLE',
    aws_secret_access_key='wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY',
    aws_session_token='IQoJb3JpZ2luX2VjEEXAMPLE......'
)

# Allow guest users to view sample data (limited to 5 items)
response = dynamodb.scan(TableName='SampleProducts', Limit=5)
```

JavaScript

```
// Example: Limited app functionality for trial users
```

```
import { DynamoDBClient, ScanCommand } from "@aws-sdk/client-dynamodb";

const dynamodbClient = new DynamoDBClient({
    region: "us-east-1",
    credentials: {
        accessKeyId: 'AKIAIOSFODNN7EXAMPLE',
        secretAccessKey: 'wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY',
        sessionToken: 'IQoJb3JpZ2luX2VjEEXAMPLE......'
    }
});

// Allow guest users to view sample data (limited to 5 items)
const response = await dynamodbClient.send(new ScanCommand({
    TableName: 'SampleProducts',
    Limit: 5
}));
```

소셜 ID 제공업체 인증

이 단계에서는 Amazon Cognito ID 풀에서 소셜 ID 제공업체를 사용하는 전체 흐름을 살펴봅니다. 소셜 인증은 페더레이션 ID 관리를 통해 보안을 유지하면서 익숙한 로그인 환경을 제공합니다. Google, Facebook, Amazon과 같은 소셜 자격 증명 공급자(IdP)에서 로그인한 다음 해당 IdP 토큰을 임시 AWS 자격 증명으로 교환할 수 있습니다. Twitter와 Apple 통합은 ID 풀에서도 지원되지만 예제 애플리케이션에서는 지원되지 않습니다.

ID 풀 자체는 사용자 디렉터리가 아닙니다. 암호나 프로필 필드는 저장되지 않습니다. 대신 외부 IdP를 신뢰하여 사용자를 인증하고 이미 인증된 사용자가 IAM 역할에 대한 자격 증명을 제공하여 AWS 서비스를 직접 호출할 수 있도록 권한을 부여하는 데 중점을 둡니다.

Social identity provider with enhanced flow

이 섹션에서는 소셜 ID 제공업체를 사용하여 사용자에 로그인하고 향상된 흐름을 사용하여 Amazon Cognito ID 풀의 공급자 토큰을 AWS 리소스를 요청하는 임시 자격 증명으로 교환하는 방법을 보여줍니다.

예제 애플리케이션에서 향상된 흐름으로 소셜 로그인 사용

1. 데모 애플리케이션에서 대화형 데모 섹션으로 이동합니다.
2. 인증된 액세스 탭을 선택합니다.
3. 향상된 흐름 탭을 선택합니다.

4. Sign in with Google, Sign in with Facebook 또는 Login with Amazon과 같이 사용자가 구성한 지원되는 소셜 공급자를 선택합니다.
5. 로그인하고 애플리케이션과 사용자 데이터를 공유하는 데 동의합니다.
6. 공급자가 앱의 리디렉션 URI로 다시 리디렉션합니다.
7. 앱이 공급자 토큰을 자격 증명 풀로 보내고 임시 AWS 자격 증명을 검색합니다.
8. 앱은 웹 인터페이스에 결과 패널을 표시합니다.

   인증에 성공하면 결과 패널이 표시되는 웹 인터페이스가 표시되고 탐색할 수 있는 다음과 같은 두 가지 옵션이 있습니다.

   a. 자격 증명만 보기 버튼: API 흐름 세부 정보 없이 생성된 임시 AWS 자격 증명을 직접 보려면 이 버튼을 선택합니다.

```
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "Credentials": {
    "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "SessionToken": "IQoJb3JpZ2luX2VjEEXAMPLE......",
    "Expiration": "2025-08-12T13:36:17-07:00"
  }
}
```

   b. 세부 API 흐름 보기 버튼: 단계별 API 요청을 보려면 이 버튼을 선택합니다.

   - 앱은 소셜 IdP로 사용자를 로그인하고 공급자 토큰을 가져옵니다. ID 풀은 소셜 제공업체로부터 다음과 같은 아티팩트를 허용합니다.

ID 공급자	Cognito 공급자 키	용도
Google	`accounts.google.com`	Google Sign-In의 OAuth 2.0 토큰
Facebook	`graph.facebook.com`	Facebook Login의 액세스 토큰
Amazon	`www.amazon.com`	Login with Amazon의 OAuth 토큰

소셜 제공업체와의 인증에 성공하면 앱은 액세스 토큰 및 기타 인증 세부 정보가 포함된 OAuth 응답을 수신합니다.

```
{
   "access_token": "ya29.A0AS3H6NEXAMPLE......",
   "expires_in": 3599,
   "scope": "openid https://www.examplesocial....",
   "token_type": "Bearer",
   "id_token": "eyJhbGciOiJSUzI1NiIsEXAMPLE......"
}
```

- ID 풀 ID 및 소셜 공급자 토큰이 포함된 Logins 맵이 포함된 GetId() API 요청

```
POST GetId
{
  "IdentityPoolId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "Logins": {
    "accounts.google.com": "eyJhbGciOiJSUzI1NiIsEXAMPLE......"
  }
}
```

응답:

```
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
}
```

- 반환된 IdentityID와 동일한 Logins 맵이 있는 GetCredentialsForIdentity()

```
POST GetCredentialsForIdentity
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "Logins": {
    "accounts.google.com": "eyJhbGciOiJSUzI1NiIsEXAMPLE......"
  }
}
```

응답:

```
{
```

```
  "Credentials": {
    "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "SessionToken": "IQoJb3JpZ2luX2VjEEXAMPLE......",
    "Expiration": "2025-08-07T00:58:21-07:00"
  },
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
}
```

발생한 일: Amazon Cognito는 구성된 공급자에 대해 토큰을 검증하고, 공급자 구성을 기반으로 IAM 역할을 선택하고, 사용자를 대신하여 AWS STS 를 호출했습니다. 그러면 ID 풀이 임시 자격 증명을 반환했습니다.

Social identity provider with basic flow

이 섹션에서는 소셜 자격 증명 공급자를 사용하여 사용자에 로그인하고 기본 흐름을 사용하여 Amazon Cognito 자격 증명 풀의 공급자 토큰을 AWS 서비스를 호출하기 위한 임시 자격 증명으로 교환하는 방법을 보여줍니다.

예제 애플리케이션의 기본 흐름과 함께 소셜 로그인 사용

1. 데모 애플리케이션에서 대화형 데모 섹션으로 이동합니다.
2. 인증된 액세스 탭을 선택합니다.
3. 기본 흐름 탭을 선택합니다.
4. Sign in with Google, Sign in with Facebook 또는 Login with Amazon과 같이 사용자가 구성한 지원되는 소셜 공급자를 선택합니다.
5. 로그인하고 애플리케이션과 사용자 데이터를 공유하는 데 동의합니다.
6. 공급자가 앱의 리디렉션 URI로 다시 리디렉션합니다.
7. 앱이 공급자 토큰을 자격 증명 풀로 보내고 임시 AWS 자격 증명을 검색합니다.
8. 앱은 웹 인터페이스에 결과 패널을 표시합니다.

   인증에 성공하면 결과 패널이 표시되는 웹 인터페이스가 표시되고 탐색할 수 있는 다음과 같은 두 가지 옵션이 있습니다.

   a. 자격 증명만 보기 버튼: API 흐름 세부 정보 없이 생성된 임시 AWS 자격 증명을 직접 보려면이 버튼을 선택합니다.

```
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
```

```
  "Credentials": {
    "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "SessionToken": "IQoJb3JpZ2luX2VjEEXAMPLE......",
    "Expiration": "2025-08-12T13:36:17-07:00"
  }
}
```

b. 세부 API 흐름 보기 버튼: 단계별 API 요청을 보려면 이 버튼을 선택합니다.

- 앱은 소셜 IdP로 사용자를 로그인하고 공급자 토큰을 가져옵니다. ID 풀은 소셜 제공업체로부터 다음과 같은 아티팩트를 허용합니다.

ID 공급자	Cognito 공급자 키	용도
Google	`accounts.google.com`	Google Sign-In의 OAuth 2.0 토큰
Facebook	`graph.facebook.com`	Facebook Login의 액세스 토큰
Amazon	`www.amazon.com`	Login with Amazon의 OAuth 토큰

소셜 제공업체와의 인증에 성공하면 앱은 액세스 토큰 및 기타 인증 세부 정보가 포함된 OAuth 응답을 수신합니다.

```
{
   "access_token": "ya29.A0AS3H6NEXAMPLE......",
   "expires_in": 3599,
   "scope": "openid https://www.examplesocial....",
   "token_type": "Bearer",
   "id_token": "eyJhbGciOiJSUzI1NiIsEXAMPLE......"
}
```

- ID 풀 ID 및 소셜 공급자 토큰이 포함된 `Logins` 맵이 포함된 `GetId()` API 요청

```
POST GetId
{
  "IdentityPoolId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "Logins": {
```

```
    "accounts.google.com": "token..."
  }
}
```

응답:

```
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
}
```

- 반환된 IdentityID와 동일한 Logins 맵이 있는 GetOpenIdToken()

```
POST GetOpenIdToken
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "Logins": {
    "accounts.google.com": "token..."
  }
}
```

응답:

```
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "Token": "eyJraWQiOiJFWAMPLE......"
}
```

- OpenID 토큰이 있는 AssumeRoleWithWebIdentity()

```
POST AssumeRoleWithWebIdentity
{
  "RoleArn": "arn:aws:iam::111122223333:role/Cognito_IdentityPoolAuth_Role",
  "WebIdentityToken": "eyJraWQiOiJFWAMPLE......"
}
```

응답:

```
{
  "Credentials": {
    "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
```

```
    "SessionToken": "IQoJb3JpZ2luX2VjEEXAMPLE......",
    "Expiration": "2025-08-12T14:36:17-07:00"
  }
}
```

발생한 작업: Amazon Cognito는 구성된 공급자에 대해 토큰을 검증하고 OpenID 토큰을 발급했습니다. 애플리케이션을 AWS STS 직접 호출하여 IAM 역할을 수임하고 임시 자격 증명을 받습니다.

소셜 액세스 이해

- 소셜 사용자는 소셜 공급자를 통해 인증한 후 Amazon Cognito 자격 증명 풀을 통해 임시 AWS 자격 증명을 받습니다.
- 인증된 각 사용자는 세션 간에 지속되는 고유한 자격 증명 ID를 가져옵니다.
- 이러한 자격 증명은 인증된 액세스를 위해 특별히 설계된 IAM 역할에 연결되어 게스트 액세스보다 광범위한 권한을 제공합니다.
- 소셜 공급자 토큰은 자격 AWS 증명으로 교환되어 사용자 자격 증명과 권한을 유지합니다.

Amazon Cognito 사용자 풀 인증

이 단계에서는 사용자 풀 관리형 로그인 통합을 사용한 Amazon Cognito 인증을 살펴봅니다. 사용자 풀을 IdP로 ID 풀에 연결하면 사용자 풀 토큰이 ID 풀에 임시 자격 증명을 발급할 수 있는 권한을 부여합니다.

User pool authentication with enhanced flow

향상된 흐름은 단일 API 요청으로 Amazon Cognito ID 풀을 통해 AWS 자격 증명을 얻는 간소화된 접근 방식을 제공합니다.

ID 풀 확장 흐름과 함께 Amazon Cognito 사용자 풀 인증 사용

1. 데모 애플리케이션에서 대화형 데모 섹션으로 이동합니다.
2. 인증된 액세스 탭을 선택합니다.
3. 향상된 흐름 탭을 선택합니다.
4. Amazon Cognito 사용자 풀로 로그인 선택
5. 관리형 로그인에서 사용자 이름과 암호로 로그인을 완료합니다.
6. 사용자 풀은 권한 부여 코드를 사용하여 애플리케이션 리디렉션 URI로 다시 리디렉션됩니다.

7. 애플리케이션은 권한 부여 코드를 사용자 풀과 JSON 웹 토큰으로 교환합니다.
8. 애플리케이션은 ID 토큰을 자격 증명 풀과 임시 AWS 자격 증명으로 교환합니다.
9. 앱은 웹 인터페이스에 결과 패널을 표시합니다.

   인증에 성공하면 결과 패널이 표시되는 웹 인터페이스가 표시되고 탐색할 수 있는 다음과 같은 두 가지 옵션이 있습니다.

   a. 자격 증명만 보기 버튼: API 흐름 세부 정보 없이 생성된 임시 AWS 자격 증명을 직접 보려면 이 버튼을 선택합니다.

```
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "Credentials": {
    "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "SessionToken": "IQoJb3JpZ2luX2VjEEXAMPLE......",
    "Expiration": "2025-08-12T13:36:17-07:00"
  }
}
```

   b. 세부 API 흐름 보기 버튼: 단계별 API 요청을 보려면 이 버튼을 선택합니다.

   - 앱은 Amazon Cognito를 사용하여 사용자를 로그인합니다. 사용자 풀을 사용한 인증에 성공하면 앱은 ID 토큰(JWT)이 포함된 OAuth 2.0 응답을 수신합니다. ID 풀은 다음 공급자 키 형식을 사용하여 사용자 풀에서 JWT ID 토큰을 수락합니다.

ID 공급자	Cognito 공급자 키	용도
Amazon Cognito 사용자 풀	`cognito-idp.{region}.amazonaws.com/{user-pool-id}`	Amazon Cognito 사용자 풀의 JWT ID 토큰

   사용자 풀을 사용한 인증에 성공하면 앱은 ID 토큰(JWT)이 포함된 다음과 같은 OAuth 2.0 응답을 수신합니다.

```
{
   "id_token": "eyJraWQiOiJFWAMPLE......",
   "token_type": "Bearer",
   "expires_in": 3600
```

```
}
```

- id_token에 매핑된 사용자 풀 공급자 키가 포함된 identityPoolId 및 Logins 맵을 사용한 GetId( ) API 요청입니다. Amazon Cognito는 사용자 풀 ID 토큰의 서명, 발급자, 만료 및 대상 그룹(aud)이 ID 풀의 이 사용자 풀 IdP에 대해 등록한 앱 클라이언트 ID 중 하나와 일치하는지 확인했습니다.

```
POST GetId
{
  "AccountId": "111122223333",
  "IdentityPoolId": "us-east-1:1ac4a76d-1fef-48aa-83af-4224799c0b5c",
  "Logins": {
    "cognito-idp.us-east-1.amazonaws.com/us-east-1_EXAMPLE123":
 "eyJraWQiOiJFWAMPLE......"
  }
}
```

유효한 경우 사용자의 IdentityID를 찾거나 생성하고 반환합니다. 다음은 응답의 예입니다.

```
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
}
```

- 반환된 identityPoolId가 있는 GetCredentialsForIdentity()와 id_token이 있는 동일한 Logins 맵. Amazon Cognito는 사용자 풀 ID 토큰의 서명, 발급자, 만료 및 대상 그룹(aud)이 ID 풀의 이 사용자 풀 IdP에 대해 등록한 앱 클라이언트 ID 중 하나와 일치하는지 재검증합니다.

```
POST GetCredentialsForIdentity
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "Logins": {
    "cognito-idp.us-east-1.amazonaws.com/us-east-1_EXAMPLE123":
 "eyJraWQiOiJFWAMPLE......"
  }
}
```

유효한 경우 IAM 역할(토큰 역할, 규칙 또는 기본값)을 선택하고 사용자를 대신하여 AWS STS 를 호출한 다음 임시 AWS 자격 증명을 반환합니다. 다음은 응답의 예입니다.

```
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "Credentials": {
    "AccessKeyId": "ASIAW7TIP7EJEXAMPLE",
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "SessionToken": "IQoJb3JpZ2luX2VjEEXAMPLE......",
    "Expiration": "2025-08-12T14:36:17-07:00"
  }
}
```

User pool authentication with basic flow

기본 흐름은 ID 검색 및 자격 증명 생성을 위한 별도의 API 요청을 통해 인증 프로세스에 대한 세분화된 제어를 제공합니다.

ID 풀 기본 흐름과 함께 Amazon Cognito 사용자 풀 인증 사용

1. 데모 애플리케이션에서 대화형 데모 섹션으로 이동합니다.
2. 인증된 액세스 탭을 선택합니다.
3. 기본 흐름 탭을 선택합니다.
4. Amazon Cognito 사용자 풀로 로그인 선택
5. 관리형 로그인에서 사용자 이름과 암호로 로그인을 완료합니다.
6. 사용자 풀은 권한 부여 코드를 사용하여 애플리케이션 리디렉션 URI로 다시 리디렉션됩니다.
7. 애플리케이션은 권한 부여 코드를 사용자 풀과 JSON 웹 토큰으로 교환합니다.
8. 애플리케이션은 ID 토큰을 자격 증명 풀과 임시 AWS 자격 증명으로 교환합니다.
9. 앱은 웹 인터페이스에 결과 패널을 표시합니다.

   인증에 성공하면 결과 패널이 표시되는 웹 인터페이스가 표시되고 탐색할 수 있는 다음과 같은 두 가지 옵션이 있습니다.

   a. 자격 증명만 보기 버튼: API 흐름 세부 정보 없이 생성된 임시 AWS 자격 증명을 직접 보려면이 버튼을 선택합니다.

```
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "Credentials": {
    "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
```

```
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "SessionToken": "IQoJb3JpZ2luX2VjEEXAMPLE......",
    "Expiration": "2025-08-12T13:36:17-07:00"
  }
}
```

b. 세부 API 흐름 보기 버튼: 단계별 API 요청을 보려면 이 버튼을 선택합니다.

- 앱은 Amazon Cognito 사용자 풀을 사용하여 사용자에 로그인하고 ID 토큰(JWT)을 아티팩트로 가져옵니다. 사용자 풀을 사용한 인증에 성공하면 앱은 ID 토큰(JWT)이 포함된 OAuth 응답을 수신합니다. ID 풀은 인증에 이 토큰을 사용합니다.

```
{
   "id_token": "eyJraWQiOiJFWAMPLE......",
   "token_type": "Bearer",
   "expires_in": 3600
}
```

- ID 풀 ID와 사용자 풀 공급자 키 및 ID 토큰을 값으로 포함하는 Logins 맵이 포함된 GetId( ) API 요청입니다. Amazon Cognito는 사용자 풀 ID 토큰의 서명, 만료 및 대상 그룹(aud)이 ID 풀의 이 사용자 풀 IdP에 대해 등록한 앱 클라이언트 ID 중 하나와 일치하는지 확인했습니다.

```
POST GetId
{
  "AccountId": "111122223333",
  "IdentityPoolId": "us-east-1:1ac4a76d-1fef-48aa-83af-4224799c0b5c",
  "Logins": {
    "cognito-idp.us-east-1.amazonaws.com/us-east-1_EXAMPLE123":
 "eyJraWQiOiJFWAMPLE......"
  }
}
```

유효한 경우 사용자의 IdentityID를 찾거나 생성하고 반환합니다. 다음은 응답의 예입니다.

```
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
}
```

- 반환된 IdentityID와 동일한 Logins 맵이 있는 GetOpenIdToken( )

```
POST GetOpenIdToken
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "Logins": {
    "cognito-idp.us-east-1.amazonaws.com/us-east-1_EXAMPLE123":
 "eyJraWQiOiJFWAMPLE......"
  }
}
```

응답:

```
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "Token": "eyJraWQiOiJFWAMPLE......"
}
```

이 단계에서 발생하는 작업: Amazon Cognito는 이 `IdentityId`를 나타내는 cognito-identity.amazonaws.com에서 단기 OpenID Connect 웹 ID 토큰을 발행합니다. 토큰에는 aud(자격 증명 풀 ID) 및 amr(인증 또는 미인증)을 포함하여가 AWS STS 평가하는 OIDC 클레임이 포함됩니다. IAM 역할의 신뢰 정책에는 이러한 클레임이 필요합니다.

- `AssumeRoleWithWebIdentity()` - 앱이를 AWS STS 직접 호출하여 Amazon Cognito OpenID 토큰을 임시 AWS 자격 증명으로 교환합니다.

```
POST sts:AssumeRoleWithWebIdentity
{
  "RoleArn": "arn:aws:iam::111122223333:role/Cognito_IdentityPoolAuth_Role",
  "WebIdentityToken": "eyJraWQiOiJFWAMPLE......",
  "RoleSessionName": "CognitoIdentityCredentials"
}
```

응답:

```
{
  "Credentials": {
    "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "SessionToken": "FwoGZXIvYXdzEEXAMPLE......",
    "Expiration": "2025-08-12T14:36:17-07:00"
  },
```

```
  "AssumedRoleUser": {
    "AssumedRoleId": "AROAW7TIP7EJYEXAMPLE:CognitoIdentityCredentials",
    "Arn": "arn:aws:sts::111122223333:assumed-role/
Cognito_IdentityPoolAuth_Role/CognitoIdentityCredentials"
  }
}
```

데모 애플리케이션이 수행한 작업: 앱이 `GetOpenIdToken()`에서 AWS STS로 OpenID 토큰을 전송하여 임시 자격 증명을 요청했습니다. AWS STS 는 검증 검사를 수행하고 자격 증명을 발급했습니다.

사용자 풀 액세스 이해

- 사용자 풀 사용자는 Amazon Cognito 자격 증명 풀을 통해 임시 AWS 자격 증명을 받습니다.
- 이러한 자격 증명은 ID 풀 구성에 지정된 IAM 역할에 연결됩니다.
- 사용자 풀 ID 토큰은 자격 증명 풀을 통해 AWS 자격 증명으로 교환됩니다.

SAML 인증

이 단계에서는 SAML 인증을 살펴봅니다. 사용자는 AWS 서비스에 액세스하기 위해 SAML을 지원하는 엔터프라이즈 자격 증명 공급자로 로그인할 수 있습니다. 예제 애플리케이션에서는 SAML을 사용한 기본 흐름이 지원되지 않습니다.

SAML authentication with enhanced flow

이 섹션에서는 SAML 자격 증명 공급자를 사용하여 사용자에 로그인하고 향상된 흐름을 사용하여 Amazon Cognito 자격 증명 풀의 SAML 어설션을 서비스를 호출 AWS 하기 위한 임시 AWS 자격 증명으로 교환하는 방법을 보여줍니다.

ID 풀 확장 흐름과 함께 SAML 인증 사용

1. 데모 애플리케이션에서 대화형 데모 섹션으로 이동합니다.
2. 인증된 액세스 탭을 선택합니다.
3. 향상된 흐름 탭을 선택합니다.
4. SAML 공급자로 로그인을 선택합니다.
5. 엔터프라이즈 자격 증명으로 로그인을 완료합니다.
6. 사용자 풀은 SAML 어설션을 사용하여 애플리케이션 리디렉션 URI로 다시 리디렉션됩니다.

7. 애플리케이션은 권한 부여 코드를 사용자 풀과 JSON 웹 토큰으로 교환합니다.
8. 애플리케이션은 SAML 응답을 자격 증명 풀과 임시 AWS 자격 증명으로 교환합니다.
9. 앱은 웹 인터페이스에 결과 패널을 표시합니다.

   인증에 성공하면 결과 패널이 표시되는 웹 인터페이스가 표시되고 탐색할 수 있는 다음과 같은 두 가지 옵션이 있습니다.

   a. 자격 증명만 보기 버튼: API 흐름 세부 정보 없이 생성된 임시 AWS 자격 증명을 직접 보려면 이 버튼을 선택합니다.

```
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "Credentials": {
    "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "SessionToken": "IQoJb3JpZ2luX2VjEEXAMPLE......",
    "Expiration": "2025-08-12T13:36:17-07:00"
  }
}
```

   b. 세부 API 흐름 보기 버튼: 단계별 API 요청을 보려면 이 버튼을 선택합니다.

   - 앱은 SAML IdP로 사용자를 로그인하고 SAML 응답을 가져옵니다. ID 풀은 SAML 공급자 ARN을 키로 사용하여 엔터프라이즈 공급자의 SAML 어설션을 수락합니다.

ID 공급자	Cognito 공급자 키	용도
SAML 공급자	`arn:aws:iam::111122223333:saml-provider/EXAMPLE`	엔터프라이즈 IdP의 SAML 어설션

   SAML 공급자와의 인증에 성공하면 앱은 HTTP POST를 통해 콜백 URL에 대한 SAML 응답을 수신합니다.

```
{
  "saml_response": "PD94bWwgdmVyc2lvbj0iMS4wIiBFWAMPLE...",
  "provider_arn": "arn:aws:iam::111122223333:saml-provider/EXAMPLE",
  "status": "Authentication successful"
}
```

- ID 풀 ID와 SAML 공급자 ARN 및 어설션이 포함된 Logins 맵이 포함된 GetId() API 요청

```
POST GetId
{
  "AccountId": "111122223333",
  "IdentityPoolId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "Logins": {
    "arn:aws:iam::111122223333:saml-provider/EXAMPLE":
 "PD94bWwgdmVyc2lvbj0iMS4wIiBFWAMPLE..."
  }
}
```

응답:

```
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
}
```

- 반환된 IdentityID와 동일한 Logins 맵이 있는 GetCredentialsForIdentity()

```
POST GetCredentialsForIdentity
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "Logins": {
    "arn:aws:iam::111122223333:saml-provider/EXAMPLE":
 "PD94bWwgdmVyc2lvbj0iMS4wIiBFWAMPLE..."
  }
}
```

응답:

```
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "Credentials": {
    "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "SessionToken": "IQoJb3JpZ2luX2VjEEXAMPLE......"
  }
}
```

발생한 일: Amazon Cognito는 구성된 공급자에 대해 SAML 어설션을 검증하고, SAML 속성 또는 규칙을 기반으로 IAM 역할을 선택하고, 사용자를 대신하여 AWS STS 를 호출했습니다.

SAML 액세스 이해

- 엔터프라이즈 사용자는 SAML 공급자를 통해 인증한 후 Amazon Cognito 자격 증명 풀에서 임시 AWS 자격 증명을 받습니다.
- 인증된 각 사용자는 세션 간에 지속되는 고유한 자격 증명 ID를 가져옵니다.
- 이러한 자격 증명은 인증된 액세스를 위해 특별히 설계된 IAM 역할에 연결되어 게스트 액세스보다 광범위한 권한을 제공합니다.
- SAML 어설션은 자격 AWS 증명과 교환되어 사용자 자격 증명과 엔터프라이즈 속성을 유지합니다.

OpenID Connect(OIDC) 인증

이 단계에서는 엔터프라이즈 ID 제공업체를 사용한 OIDC 인증을 살펴봅니다. 사용자는 조직의 엔터프라이즈 자격 증명 공급자(예: Azure AD, Okta 또는 Google Workspace)를 통해 로그인하여 AWS 서비스에 액세스할 수 있습니다. 표준 기반 인증을 AWS 리소스와 통합하는 방법에 대한 정보를 찾고 있다면 계속 읽어보세요. OIDC 인증을 구현한 후에는 세분화된 액세스 제어를 위해 OIDC 클레임을 활용하는 방법을 알아봅니다.

OIDC authentication with enhanced flow

이 섹션에서는 OIDC 자격 증명 공급자를 사용하여 사용자에 로그인하고 향상된 흐름을 사용하여 Amazon Cognito 자격 증명 풀의 OIDC 토큰을 서비스를 호출 AWS 하기 위한 임시 AWS 자격 증명으로 교환하는 방법을 보여줍니다.

ID 풀 확장 흐름과 함께 OIDC 로그인 사용

1. 데모 애플리케이션에서 대화형 데모 섹션으로 이동합니다.
2. 인증된 액세스 탭을 선택합니다.
3. 향상된 흐름 탭을 선택합니다.
4. OIDC 공급자로 로그인을 선택합니다.
5. 엔터프라이즈 자격 증명으로 로그인을 완료합니다.
6. OIDC 공급자가 권한 부여 코드를 사용하여 앱으로 다시 리디렉션합니다.
7. 애플리케이션은 권한 부여 코드를 사용자 풀과 JSON 웹 토큰으로 교환합니다.
8. 애플리케이션은 OIDC 토큰을 자격 증명 풀로 보내고 임시 AWS 자격 증명을 검색합니다.

9. 앱은 웹 인터페이스에 결과 패널을 표시합니다.

   인증에 성공하면 결과 패널이 표시되는 웹 인터페이스가 표시되고 탐색할 수 있는 다음과 같은 두 가지 옵션이 있습니다.

   a. 자격 증명만 보기 버튼: API 흐름 세부 정보 없이 생성된 임시 AWS 자격 증명을 직접 보려면 이 버튼을 선택합니다.

   ```
   {
     "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
     "Credentials": {
       "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
       "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
       "SessionToken": "IQoJb3JpZ2luX2VjEEXAMPLE......",
       "Expiration": "2025-08-12T13:36:17-07:00"
     }
   }
   ```

   b. 세부 API 흐름 보기 버튼: 단계별 API 요청을 보려면 이 버튼을 선택합니다.

   - 앱은 OIDC IdP로 사용자를 로그인하고 ID 토큰을 가져옵니다. ID 풀은 엔터프라이즈 공급자의 OIDC 토큰을 수락합니다.

ID 공급자	Cognito 공급자 키	용도
OIDC 공급자	`example-provider.com/oauth2/default`	엔터프라이즈 IdP의 OIDC ID 토큰

   OIDC 공급자와의 인증에 성공하면 앱은 토큰이 포함된 OAuth 2.0 응답을 받습니다.

   ```
   {
     "token_type": "Bearer",
     "expires_in": 3600,
     "access_token": "eyJraWQiOiJFWAMPLE......",
     "scope": "email openid profile",
     "id_token": "eyJraWQiOiJFWAMPLE......"
   }
   ```

   - ID 풀 ID 및 OIDC 공급자 토큰이 포함된 `Logins` 맵이 포함된 `GetId()` API 요청

   ```
   POST GetId
   ```

```
{
  "AccountId": "111122223333",
  "IdentityPoolId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "Logins": {
    "example-provider.com/oauth2/default": "eyJraWQiOiJFWAMPLE......"
  }
}
```

응답:

```
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
}
```

- 반환된 IdentityID와 동일한 Logins 맵이 있는 GetCredentialsForIdentity()

```
POST GetCredentialsForIdentity
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "Logins": {
    "example-provider.com/oauth2/default": "eyJraWQiOiJFWAMPLE......"
  }
}
```

응답:

```
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "Credentials": {
    "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "SessionToken": "IQoJb3JpZ2luX2VjEEXAMPLE......"
  }
}
```

발생한 일: Amazon Cognito는 구성된 공급자에 대해 OIDC 토큰을 검증하고, IAM 역할(기본값, 클레임 기반 또는 규칙 매핑)을 선택하고, 사용자를 AWS STS 대신하여를 호출했습니다.

OIDC authentication with basic flow

이 섹션에서는 OIDC 자격 증명 공급자를 사용하여 사용자에 로그인하고 기본 흐름을 사용하여 Amazon Cognito 자격 증명 풀의 OIDC 토큰을 서비스를 호출 AWS 하기 위한 임시 AWS 자격 증명으로 교환하는 방법을 보여줍니다.

ID 풀 기본 흐름과 함께 OIDC 로그인 사용

1. 데모 애플리케이션에서 대화형 데모 섹션으로 이동합니다.
2. 인증된 액세스 탭을 선택합니다.
3. 기본 흐름 탭을 선택합니다.
4. OIDC 공급자로 로그인을 선택합니다.
5. 엔터프라이즈 자격 증명으로 로그인을 완료합니다.
6. OIDC 공급자가 권한 부여 코드를 사용하여 앱으로 다시 리디렉션합니다.
7. 애플리케이션은 권한 부여 코드를 사용자 풀과 JSON 웹 토큰으로 교환합니다.
8. 애플리케이션은 OIDC 토큰을 자격 증명 풀로 보내고 임시 AWS 자격 증명을 검색합니다.
9. 앱은 웹 인터페이스에 결과 패널을 표시합니다.

   인증에 성공하면 결과 패널이 표시되는 웹 인터페이스가 표시되고 탐색할 수 있는 다음과 같은 두 가지 옵션이 있습니다.

   a. 자격 증명만 보기 버튼: API 흐름 세부 정보 없이 생성된 임시 AWS 자격 증명을 직접 보려면 이 버튼을 선택합니다.

   ```
   {
     "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
     "Credentials": {
       "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
       "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
       "SessionToken": "IQoJb3JpZ2luX2VjEEXAMPLE......",
       "Expiration": "2025-08-12T13:36:17-07:00"
     }
   }
   ```

   b. 세부 API 흐름 보기 버튼: 단계별 API 요청을 보려면 이 버튼을 선택합니다.

   - 앱은 OIDC IdP로 사용자를 로그인하고 ID 토큰을 가져옵니다. ID 풀은 엔터프라이즈 공급자의 OIDC 토큰을 수락합니다.

ID 공급자	Cognito 공급자 키	용도
OIDC 공급자	`example-provider.com/oauth2/default`	엔터프라이즈 IdP의 OIDC ID 토큰

OIDC 공급자와의 인증에 성공하면 앱은 토큰이 포함된 OAuth 2.0 응답을 받습니다.

```
{
  "token_type": "Bearer",
  "expires_in": 3600,
  "access_token": "eyJraWQiOiJFWAMPLE......",
  "scope": "openid email profile",
  "id_token": "eyJraWQiOiJFWAMPLE......"
}
```

- ID 풀 ID 및 OIDC 공급자 토큰이 포함된 `Logins`맵이 포함된 `GetId()` API 요청

```
POST GetId
{
  "IdentityPoolId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "Logins": {
    "example-provider.com/oauth2/default": "eyJraWQiOiJFWAMPLE......"
  }
}
```

응답:

```
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
}
```

- 반환된 IdentityID와 동일한 `Logins` 맵이 있는 `GetOpenIdToken()`

```
POST GetOpenIdToken
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "Logins": {
    "example-provider.com/oauth2/default": "eyJraWQiOiJFWAMPLE......"
  }
```

```
}
```

응답:

```
{
  "IdentityId": "us-east-1:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "Token": "eyJraWQiOiJFWAMPLE......"
}
```

- OpenID 토큰이 있는 `AssumeRoleWithWebIdentity()`

```
POST AssumeRoleWithWebIdentity
{
  "RoleArn": "arn:aws:iam::111122223333:role/Cognito_IdentityPoolAuth_Role",
  "WebIdentityToken": "eyJraWQiOiJFWAMPLE......"
}
```

응답:

```
{
  "Credentials": {
    "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "SessionToken": "FwoGZXIvYXdzEEXAMPLE......",
    "Expiration": "2025-08-12T14:36:17-07:00"
  }
}
```

발생한 작업: Amazon Cognito는 구성된 공급자에 대해 OIDC 토큰을 검증하고 OpenID 토큰을 반환했습니다. 애플리케이션은를 AWS STS 직접 호출하여 적절한 IAM 역할을 수임하고 수명이 짧은 자격 증명을 받았습니다.

OIDC 인증 이해

- 표준 기반: OIDC는 OAuth 2.0을 기반으로 하며 표준화된 ID 정보를 제공합니다.
- 토큰 검증: ID 토큰의 신뢰성을 검증할 수 있습니다.
- 클레임 기반 액세스: OIDC 클레임을 역할 매핑 및 액세스 제어에 사용할 수 있습니다.
- 엔터프라이즈 통합: 널리 사용되는 엔터프라이즈 ID 제공업체와 함께 작동합니다.

다음 단계

이제 데모 애플리케이션을 설정하고 살펴보았으므로 다음을 수행할 수 있습니다.

- 아직 테스트하지 않은 추가 ID 제공업체 구성
- 향상된 인증과 기본 인증을 모두 사용하여 차이점을 이해합니다.
- 자체 사용 사례에 맞게 데모 사용자 지정
- Amazon Cognito ID 풀을 자체 애플리케이션에 통합합니다.

Amazon Cognito의 보안

의 클라우드 보안 AWS 이 최우선 순위입니다. AWS 고객은 보안에 가장 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 이점을 누릴 수 있습니다.

보안은 AWS 와 사용자 간의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS 는 AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다. AWS 또한는 안전하게 사용할 수 있는 서비스를 제공합니다. 타사 감사자는 AWS 규정 준수 프로그램 일환으로 보안의 효과를 정기적으로 테스트하고 확인합니다. Amazon Cognito에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램 AWS 제공 범위 내 서비스규정 준수 프로그램 .
- 클라우드의 보안 - 사용자의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 여러분은 데이터의 민감도, 회사 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 문서는 Amazon Cognito를 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 보안 및 규정 준수 목표에 맞게 Amazon Cognito를 구성하는 방법을 보여줍니다. 또한 Amazon Cognito 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법을 알아봅니다.

악의적인 가입, 로그인 및 계정 탈취, 즉 MFA, 적응형 인증 및 위협 방지에 대한 사용자 풀별 보호는 섹션을 참조하세요위협 방지를 통한 사용자 풀 보호.

내용

- Amazon Cognito의 데이터 보호
- Amazon Cognito의 Identity and Access Management
- Amazon Cognito의 로깅 및 모니터링
- 인터페이스 엔드포인트를 사용하여 Amazon Cognito에 액세스(AWS PrivateLink)
- Amazon Cognito에 대한 규정 준수 검증
- Amazon Cognito의 복원성
- Amazon Cognito의 인프라 보안
- Amazon Cognito 사용자 풀의 구성 및 취약성 분석
- AWS Amazon Cognito에 대한 관리형 정책

Amazon Cognito의 데이터 보호

AWS 공동 책임 모델 Amazon Cognito(Amazon Cognito)의 데이터 보호에 적용됩니다. 이 모델에 설명된 대로 AWS 는 모든 AWS 클라우드를 실행하는 글로벌 인프라를 보호할 책임이 있습니다. 사용자는 이 인프라에 호스팅되는 콘텐츠에 대한 통제 권한을 유지할 책임이 있습니다. 이 콘텐츠에는 사용하는 AWS 서비스에 대한 보안 구성 및 관리 작업이 포함됩니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요.

데이터 보호를 위해 AWS 계정 자격 증명을 보호하고 AWS Identity and Access Management (IAM)를 사용하여 개별 사용자 계정을 설정하는 것이 좋습니다. 이러한 방식에서는 각 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용합니다.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다 AWS CloudTrail.
- 서비스 내의 AWS 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용합니다.
- Amazon S3에 저장된 개인 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용합니다.

이름 필드와 같은 자유 형식 필드에 고객 계정 번호와 같은 중요 식별 정보를 절대 입력하지 마세요. 여기에는 Amazon Cognito 또는 기타 AWS 서비스에서 콘솔 AWS CLI, API 또는 AWS SDKs를 사용하여 작업하는 경우가 포함됩니다. Amazon Cognito 또는 기타 서비스에 입력하는 모든 데이터를 진단 로그에 포함할 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명 정보를 URL에 포함시키지 마세요.

데이터 암호화

데이터 암호화는 일반적으로 저장 데이터 암호화와 전송 중 데이터 암호화의 두 가지 범주로 나뉩니다.

저장 데이터 암호화

Amazon Cognito 사용자 풀 및 자격 증명 풀 내의 데이터는 업계 표준에 따라 저장 시 암호화됩니다.

- Amazon Cognito는 AWS 소유 키를 사용하여 자격 증명 풀의 고객 데이터를 암호화합니다. 이 동작은 변경할 수 없습니다.
- 기본적으로 Amazon Cognito는 AWS 소유 키를 사용하여 사용자 풀의 고객 데이터도 암호화합니다. 대신 고객 관리형 키로 고객 정보를 암호화하도록 사용자 풀을 구성할 수도 있습니다.

AWS 소유 키

Amazon Cognito는 AWS 소유 KMS 키를 사용하여 사용자 풀 또는 자격 증명 풀의 데이터를 암호화합니다. 이 유형의 키는에 표시되지 않습니다 AWS KMS.

고객 관리형 키

Amazon Cognito는 고객 관리형 키를 사용하여 사용자 풀의 데이터를 암호화합니다. 고객 관리형 키 정책, 교체 및 예약된 삭제 관리는 사용자가 소유합니다.

> **Note**
>
> 고객 관리형 키를 사용한 암호화는 일부 사용자 풀에서 사용하지 못할 수 있습니다. 새로 생성된 사용자 풀에는 항상 이러한 형식의 암호화를 사용할 수 있습니다.

고객 관리형 키를 사용한 사용자 풀 암호화에 대해 알아야 할 사항

1. 암호화 설정을 구성하는 작업을 수행하지 않더라도 사용자 풀의 모든 고객 데이터는 저장 시 암호화됩니다.
2. Amazon Cognito는 유휴 사용자 풀 암호화를 위해 사용자 풀 AWS 리전 과 동일한에서 대칭 KMS 키만 지원합니다. 비대칭 키를 사용하여 유휴 시 사용자 풀 암호화를 구성할 수 없습니다. 단일 리전 키와 사용자 풀과 동일한 리전에 있는 다중 리전 키를 사용하여 저장 시 암호화를 구성할 수 있습니다.
3. 사용자 풀 암호화는 별칭이 아닌 KMS 키 ARN으로만 구성할 수 있습니다.

PII 암호화

Amazon Cognito는 검색 가능한 암호화를 사용하여 사용자 속성 검색에서 개인 식별 정보(PII)의 기밀성, 무결성 및 가용성을 지원합니다. 사용자 풀 데이터 세트에 최적화된 이러한 해시 기반 메시지 인증 코드(HMAC) 함수는 사용자 속성의 일반 텍스트 값과 암호화된 값 간에 매핑됩니다. Amazon Cognito는 사용자 풀을 암호화하는 KMS 키를 사용하여 HMAC 값을 계산합니다. 이 보호는 다음 속성에 적용됩니다.

- `sub`
- `email`
- `phone_number`
- `given_name`
- `family_name`

- name
- username
- preferred_username
- cognito:user_status

다음 절차에서는 사용자 풀에서 저장 데이터 암호화를 구성합니다. Amazon Cognito와 AWS 서비스 같이 액세스를 위임하는 KMS 키 정책에 대한 자세한 내용은 키 정책의 Amazon Cognito에 대한 권한을 참조하세요.

Set customer managed key policy

고객 관리형 키를 사용하려면 키가 Amazon Cognito 서비스 보안 주체를 신뢰하여 키에 대한 암호화 및 복호화 작업을 수행해야 합니다. 다음 예제와 같이 KMS 키의 키 정책을 구성합니다. 이 정책을 작성하는 IAM 보안 주체는 kms:PutKeyPolicy 권한이 있는 KMS 키에 대한 쓰기 액세스 권한이 있어야 합니다.

```
{
    "Id": "cognito-cmk-policy",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow Amazon Cognito service access",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "cognito-idp.amazonaws.com",
                    "identitystore.amazonaws.com"
                ]
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKeyWithoutPlainText"
            ],
            "Resource": "*",
            "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": [
                        "arn:aws:cognito-idp:us-east-1:111122223333:userpool/us-
east-1_EXAMPLE"
```

```
                    ]
                },
                "StringEquals": {
                    "aws:SourceAccount": [
                        "111122223333"
                    ]
                },
                "StringLike": {
                    "kms:EncryptionContext:aws:cognito-idp:userpool-arn":
 "arn:aws:cognito-idp:us-east-1:111122223333:userpool/us-east-1_EXAMPLE"
                }
            }
        },
        {
            "Sid": "Allow Amazon Cognito service DescribeKey access",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "cognito-idp.amazonaws.com",
                    "identitystore.amazonaws.com"
                ]
            },
            "Action": [
                "kms:DescribeKey"
            ],
            "Resource": "*",
            "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": [
                        "arn:aws:cognito-idp:us-east-1:111122223333:userpool/us-
east-1_EXAMPLE"
                    ]
                },
                "StringEquals": {
                    "aws:SourceAccount": [
                        "111122223333"
                    ]
                }
            }
        },
        {
            "Sid": "Allow access through Amazon Cognito for all principals in
 account that are authorized to use Amazon Cognito",
            "Effect": "Allow",
```

```
            "Principal": {
                "AWS": "*"
            },
            "Action": [
                "kms:DescribeKey"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": [
                        "cognito-idp.us-east-1.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "Allow access through Amazon Cognito for all principals in
account that are authorized to use Amazon Cognito",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKeyWithoutPlainText"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": [
                        "cognito-idp.us-east-1.amazonaws.com"
                    ]
                },
                "StringLike": {
                    "kms:EncryptionContext:aws:cognito-idp:userpool-arn":
"arn:aws:cognito-idp:us-east-1:111122223333:userpool/us-east-1_EXAMPLE"
                }
            }
        },
        {
            "Sid": "Allow administrators to manage the key",
            "Effect": "Allow",
```

```
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        }
    ]
}
```

Set customer managed key policy with permissions for encryption of exported logs

로그 내보내기가 구성된 경우 KMS 키에이 정책을 적용합니다. 이 정책은 Amazon Cognito가 로그를 내보낼 때 로그를 암호화하고 중간 서비스는 로그를 해독한 후 CloudWatch Logs 로그 그룹에 쓸 수 있도록 허용합니다. 다음 예제와 같이 KMS 키의 키 정책을 구성합니다. 이 정책을 작성하는 IAM 보안 주체는 `kms:PutKeyPolicy` 권한이 있는 KMS 키에 대한 쓰기 액세스 권한이 있어야 합니다.

```
{
    "Id": "cognito-cmk-policy",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow Amazon Cognito service access",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "cognito-idp.amazonaws.com",
                    "identitystore.amazonaws.com"
                ]
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKeyWithoutPlainText"
            ],
            "Resource": "*",
            "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": [
                        "arn:aws:cognito-idp:us-east-1:111122223333:userpool/us-
east-1_EXAMPLE"
                    ]
                },
```

```
                "StringEquals": {
                    "aws:SourceAccount": [
                        "111122223333"
                    ]
                },
                "StringLike": {
                    "kms:EncryptionContext:aws:cognito-idp:userpool-arn":
 "arn:aws:cognito-idp:us-east-1:111122223333:userpool/us-east-1_EXAMPLE"
                }
            }
        },
        {
            "Sid": "Allow Amazon Cognito service DescribeKey access",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "cognito-idp.amazonaws.com",
                    "identitystore.amazonaws.com"
                ]
            },
            "Action": [
                "kms:DescribeKey"
            ],
            "Resource": "*",
            "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": [
                        "arn:aws:cognito-idp:us-east-1:111122223333:userpool/us-
east-1_EXAMPLE"
                    ]
                },
                "StringEquals": {
                    "aws:SourceAccount": [
                        "111122223333"
                    ]
                }
            }
        },
        {
            "Sid": "Allow access through Amazon Cognito for all principals in
 account that are authorized to use Amazon Cognito",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
```

```
            },
            "Action": [
                "kms:DescribeKey"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": [
                        "cognito-idp.us-east-1.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "Allow access through Amazon Cognito for all principals in
account that are authorized to use Amazon Cognito",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKeyWithoutPlainText"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": [
                        "cognito-idp.us-east-1.amazonaws.com"
                    ]
                },
                "StringLike": {
                    "kms:EncryptionContext:aws:cognito-idp:userpool-arn":
"arn:aws:cognito-idp:us-east-1:111122223333:userpool/us-east-1_EXAMPLE"
                }
            }
        },
        {
            "Sid": "Allow Amazon Cognito service log delivery access",
            "Effect": "Allow",
            "Principal": {
                "Service": [
```

```
                    "cognito-idp.amazonaws.com"
                ]
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Encrypt"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "kms:EncryptionContext:SourceArn": "arn:aws:logs:us-
east-1:111122223333:*"
                }
            }
        },
        {
            "Sid": "Allow IngestionHub service log delivery access",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "delivery.logs.amazonaws.com"
                ]
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "kms:EncryptionContext:SourceArn": "arn:aws:logs:us-
east-1:111122223333:*"
                }
            }
        },
        {
            "Sid": "Allow administrators to manage the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        }
```

```
    ]
}
```

Configure encryption at rest in the console

사용자 풀에서 저장 시 암호화를 구성하려면

1. Amazon Cognito 콘솔로 이동합니다. AWS 자격 증명을 입력하라는 메시지가 나타날 수 있습니다.
2. [사용자 풀(User Pools)]을 선택합니다.
3. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
4. 설정 메뉴를 선택하고 사용자 풀 보안 탭으로 이동합니다. 저장 시 암호화를 찾아 편집을 선택합니다.
5. 키 유형에서 AWS 소유 키 또는 고객 관리형 키를 선택합니다.

   a. AWS 소유 키를 선택한 경우 추가 구성이 필요하지 않습니다.

   b. 고객 관리형 키를 선택한 경우 고객 관리형 키 ARN에 KMS 키의 ARN을 입력합니다. AWS KMS 콘솔에서 AWS KMS 키 생성을 선택하고 새 창을 열어 새 KMS 키를 생성할 수도 있습니다.
6. 변경 사항 저장을 선택합니다.

Configure encryption at rest with the API

CreateUserPool 또는 UpdateUserPool API 요청에서 키 구성을 설정합니다. 다음 부분 예제 요청 본문은 제공된 고객 관리형 키를 사용하도록 사용자 풀을 설정합니다. 전체 예제 요청은 예제 섹션을 참조하세요.

```
"KeyConfiguration": {
   "KeyType": "CUSTOMER_MANAGED_KEY",
   "KmsKeyArn": "arn:aws:kms:us-east-1:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE22222"
},
```

다음 부분 예제 요청 본문은 AWS 소유 키를 사용하도록 사용자 풀을 설정합니다.

```
"KeyConfiguration": {
   "KeyType": "AWS_OWNED_KEY"
},
```

DescribeUserPool 응답에 KeyConfiguration 파라미터가 포함되지 않은 경우 사용자 풀은 AWS 소유 키로 저장 데이터를 암호화하도록 구성됩니다.

전송 중 암호화

관리형 서비스인 Amazon Cognito는 AWS 글로벌 네트워크 보안으로 보호됩니다. AWS 보안 서비스 및가 인프라를 AWS 보호하는 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하세요. 인프라 보안 모범 사례를 사용하여 환경을 설계하려면 보안 원칙 AWS Well-Architected Framework의 인프라 보호를 참조하세요 AWS .

AWS 에서 게시한 API 호출을 사용하여 네트워크를 통해 Amazon Cognito에 액세스합니다. 클라이언트는 다음을 지원해야 합니다.

- Transport Layer Security(TLS). TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

Amazon Cognito 사용자 풀과 자격 증명 풀에는 IAM 인증, 비인증 및 토큰 승인 API 작업이 있습니다. 일부는 사용자 풀 도메인 구성과 같은 관리 작업을 위한 컨트롤 플레인 유형 작업이고, 다른 하나는 인증을 위한 데이터 플레인 유형 작업입니다. 자세한 내용은 권한 부여 모델별로 그룹화된 API 작업 목록 섹션을 참조하세요. Amazon Cognito API 작업의 모든 클래스는 사용자 풀,cognito-idp 자격 증명 풀 및 서비스 엔드포인트cognito-identity의 네임스페이스를 공유합니다. AWS 서비스 엔드포인트에는 최소 TLS 버전 1.2로 전송 중 암호화가 필요합니다.

Amazon Cognito 사용자 풀은 서비스 소유 Amazon CloudFront 배포에서 제공되는 웹 도메인에서 관리형 로그인 및 클래식 호스팅 UI를 호스팅합니다. Amazon Cognito는 해당 배포에서 전송 중 암호화 설정을 관리합니다. 관리형 로그인의 암호화 설정에 대한 자세한 내용은 관리형 로그인 장의 TLS 버전을 참조하세요.

Amazon Cognito의 Identity and Access Management

AWS Identity and Access Management (IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어하는 데 도움이 AWS 서비스 되는 입니다. IAM 관리자는 어떤 사용자가 Amazon Cognito 리소스를 사용할 수 있는 인증(로그인) 및 권한(권한 있음)을 받을 수 있는지 제어합니다. IAM은 추가 비용 없이 사용할 수 AWS 서비스 있는 입니다.

주제

- 대상
- ID를 통한 인증
- 정책을 사용하여 액세스 관리
- Amazon Cognito에서 IAM을 사용하는 방법
- Amazon Cognito의 자격 증명 기반 정책 예제
- Amazon Cognito 자격 증명 및 액세스 문제 해결
- Amazon Cognito에 서비스 연결 역할 사용

대상

AWS Identity and Access Management (IAM)를 사용하는 방법은 역할에 따라 다릅니다.

- 서비스 사용자 - 기능에 액세스할 수 없는 경우 관리자에게 권한 요청(참조Amazon Cognito 자격 증명 및 액세스 문제 해결)
- 서비스 관리자 - 사용자 액세스 결정 및 권한 요청 제출(Amazon Cognito에서 IAM을 사용하는 방법 참조)
- IAM 관리자 - 액세스를 관리하기 위한 정책 작성(Amazon Cognito의 자격 증명 기반 정책 예제 참조)

ID를 통한 인증

인증은 자격 증명 자격 증명을 AWS 사용하여에 로그인하는 방법입니다. AWS 계정 루트 사용자, IAM 사용자 또는 IAM 역할을 수임하여 인증해야 합니다.

AWS IAM Identity Center (IAM Identity Center), Single Sign-On 인증 또는 Google/Facebook 자격 증명과 같은 자격 증명 소스의 자격 증명을 사용하여 페더레이션 자격 증명으로 로그인할 수 있습니다. 로그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의 AWS 계정에 로그인하는 방법 섹션을 참조하세요.

프로그래밍 방식 액세스를 위해는 요청에 암호화 방식으로 서명할 수 있는 SDK 및 CLI를 AWS 제공합니다. 자세한 내용은 IAM 사용 설명서의 API 요청용AWS Signature Version 4 섹션을 참조하세요.

AWS 계정 루트 사용자

를 생성할 때 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 AWS 계정 theroot 사용자라는 하나의 로그인 자격 증명으로 AWS 계정시작합니다. 일상적인 태스크에 루트 사용자를 사용하

지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명이 필요한 작업은 IAM 사용 설명서의 루트 사용자 자격 증명이 필요한 작업 섹션을 참조하세요.

페더레이션 ID

가장 좋은 방법은 인간 사용자에게 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 AWS 서비스 사용하여에 액세스하도록 요구하는 것입니다.

페더레이션 자격 증명은 엔터프라이즈 디렉터리, 웹 자격 증명 공급자 또는 자격 증명 소스의 자격 증명을 AWS 서비스 사용하여 Directory Service 에 액세스하는 사용자입니다. 페더레이션 ID는 임시 자격 증명을 제공하는 역할을 수임합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center를 추천합니다. 자세한 정보는 AWS IAM Identity Center 사용 설명서의 What is IAM Identity Center?를 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가진 ID입니다. 장기 자격 증명이 있는 IAM 사용자 대신 임시 자격 증명을 사용하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 자격 증명 공급자와의 페더레이션을 사용하여 IAM 사용 설명서의 임시 자격 증명을 AWS 사용하여에 액세스하도록 인간 사용자에게 요구하기를 참조하세요.

IAM 그룹은 IAM 사용자 모음을 지정하고 대규모 사용자 집합에 대한 관리 권한을 더 쉽게 만듭니다. 자세한 내용은 IAM 사용 설명서의 IAM 사용자 사용 사례 섹션을 참조하세요.

IAM 역할

IAM 역할은 임시 자격 증명을 제공하는 특정 권한이 있는 자격 증명입니다. 사용자에서 IAM 역할(콘솔)로 전환하거나 또는 API 작업을 호출하여 역할을 수임할 수 있습니다. AWS CLI AWS 자세한 내용은 IAM 사용 설명서의 역할 수임 방법을 참조하세요.

IAM 역할은 페더레이션 사용자 액세스, 임시 IAM 사용자 권한, 교차 계정 액세스, 교차 서비스 액세스 및 Amazon EC2에서 실행되는 애플리케이션에 유용합니다. 자세한 내용은 IAM 사용 설명서의 교차 계정 리소스 액세스를 참조하세요.

정책을 사용하여 액세스 관리

정책을 AWS 생성하고 자격 증명 또는 리소스에 연결하여 AWS 에서 액세스를 제어합니다. 정책은 자격 증명 또는 리소스와 연결될 때 권한을 정의합니다.는 보안 주체가 요청할 때 이러한 정책을 AWS 평가합니다. 대부분의 정책은에 JSON 문서 AWS 로 저장됩니다. JSON 정책 문서에 대한 자세한 내용은 IAM 사용 설명서의 JSON 정책 개요 섹션을 참조하세요.

정책을 사용하여 관리자는 어떤 보안 주체가 어떤 리소스에 대해 어떤 조건에서 작업을 수행할 수 있는지 정의하여 누가 무엇을 액세스할 수 있는지 지정합니다.

기본적으로 사용자 및 역할에는 어떠한 권한도 없습니다. IAM 관리자는 IAM 정책을 생성하고 사용자가 수임할 수 있는 역할에 추가합니다. IAM 정책은 작업을 수행하기 위해 사용하는 방법과 관계없이 작업에 대한 권한을 정의합니다.

ID 기반 정책

ID 기반 정책은 ID(사용자, 사용자 그룹 또는 역할)에 연결하는 JSON 권한 정책 문서입니다. 이러한 정책은 자격 증명이 수행할 수 있는 작업, 대상 리소스 및 이에 관한 조건을 제어합니다. ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

ID 기반 정책은 인라인 정책(단일 ID에 직접 포함) 또는 관리형 정책(여러 ID에 연결된 독립 실행형 정책)일 수 있습니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책 및 인라인 정책 중에서 선택 섹션을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 예를 들어 IAM 역할 신뢰 정책 및 Amazon S3 버킷 정책이 있습니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS 관리형 정책을 사용할 수 없습니다.

기타 정책 유형

AWS 는 보다 일반적인 정책 유형에서 부여한 최대 권한을 설정할 수 있는 추가 정책 유형을 지원합니다.

- 권한 경계 - ID 기반 정책에서 IAM 엔터티에 부여할 수 있는 최대 권한을 설정합니다. 자세한 정보는 IAM 사용 설명서의 IAM 엔터티의 권한 범위를 참조하세요.
- 서비스 제어 정책(SCP) - AWS Organizations내 조직 또는 조직 단위에 대한 최대 권한을 지정합니다. 자세한 내용은AWS Organizations 사용 설명서의 서비스 제어 정책을 참조하세요.
- 리소스 제어 정책(RCP) – 계정의 리소스에 사용할 수 있는 최대 권한을 설정합니다. 자세한 내용은 AWS Organizations 사용 설명서의 리소스 제어 정책(RCP)을 참조하세요.

- 세션 정책 – 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 자세한 내용은 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 에서 여러 정책 유형이 관련될 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

Amazon Cognito에서 IAM을 사용하는 방법

IAM을 사용하여 Amazon Cognito에 대한 액세스를 관리하기 전에 Amazon Cognito에서 사용할 수 있는 IAM 기능을 알아보세요.

Amazon Cognito에서 사용할 수 있는 IAM 기능

IAM 특성	Amazon Cognito 지원
자격 증명 기반 정책	예
리소스 기반 정책	아니요
정책 작업	예
정책 리소스	예
정책 조건 키	예
ACL	아니요
ABAC(정책 내 태그)	부분적
임시 자격 증명	예
위탁자 권한	아니요
서비스 역할	예
서비스 연결 역할	예

Amazon Cognito 및 기타 AWS 서비스에서 대부분의 IAM 기능을 사용하는 방법을 전체적으로 알아보려면 IAM 사용 설명서의 AWS IAM으로 작업하는 서비스를 참조하세요.

Amazon Cognito의 자격 증명 기반 정책

ID 기반 정책 지원: 예

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자 및 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

IAM ID 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. JSON 정책에서 사용할 수 있는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

Amazon Cognito의 자격 증명 기반 정책 예제

Amazon Cognito 자격 증명 기반 정책 예제를 보려면 Amazon Cognito의 자격 증명 기반 정책 예제 섹션을 참조하세요.

Amazon Cognito 내의 리소스 기반 정책

리소스 기반 정책 지원: 아니요

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예제는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는가 포함될 수 있습니다 AWS 서비스.

교차 계정 액세스를 활성화하려는 경우, 전체 계정이나 다른 계정의 IAM 개체를 리소스 기반 정책의 보안 주체로 지정할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM에서 교차 계정 리소스 액세스를 참조하세요.

Amazon Cognito의 정책 작업

정책 작업 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 `Action`요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명합니다. 연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하세요.

Amazon Cognito 작업 목록을 보려면 서비스 권한 부여 참조에서 Amazon Cognito에서 정의한 작업을 참조하세요.

Amazon Cognito의 정책 작업은 작업 앞에 다음 접두사를 사용합니다.

```
cognito-identity
```

단일 문에서 여러 작업을 지정하려면 쉼표로 구분합니다.

```
"Action": [
      "cognito-identity:action1",
      "cognito-identity:action2"
         ]
```

서명된 API vs 서명되지 않은 API

AWS 자격 증명으로 Amazon Cognito API 요청에 서명하면 AWS Identity and Access Management (IAM) 정책에서 요청을 제한할 수 있습니다. AWS 보안 인증 정보로 서명해야 하는 API 요청에는 `AdminInitiateAuth`를 사용한 서버 측 로그인과 `UpdateUserPool`과 같은 Amazon Cognito 리소스를 생성, 확인 또는 수정하는 작업이 포함됩니다. 서명된 API 요청에 대한 자세한 내용은 AWS API 요청 서명을 참조하세요.

Amazon Cognito 일반 사용자에게 제공하려는 앱을 위한 소비자 ID 제품이므로 다음과 같은 서명되지 않은 API에 액세스할 수 있습니다. 앱은 사용자와 잠재 사용자를 위해 이러한 API 요청을 수행합니다. 일부 API는 새 인증 세션을 시작하기 위해 `InitiateAuth`와 같은 사전 승인이 필요하지 않습니다. 일부 API는 인증된 기존 세션이 있는 사용자에 대한 MFA 설정을 완료하기 위해 `VerifySoftwareToken`과 같이 권한 부여를 위한 액세스 토큰 또는 세션 키를 사용합니다. 서명되지 않고 승인된 Amazon Cognito 사용자 풀 API는 Amazon Cognito API 참조에 표시된 요청 구문에서 `Session` 또는 `AccessToken` 파라미터를 지원합니다. 서명되지 않은 Amazon Cognito Identity API는 Amazon Cognito Federated Identities API 참조에 표시된 대로 `IdentityId` 파라미터를 지원합니다.

Amazon Cognito 사용자 풀 API 작업의 인증 모델과 역할에 대한 자세한 내용은 권한 부여 모델별로 그룹화된 API 작업 목록을 참조하세요.

Amazon Cognito 자격 증명 풀 API 작업

- GetId
- GetOpenIdToken
- GetCredentialsForIdentity
- UnlinkIdentity

Amazon Cognito 사용자 풀 API 작업

- AssociateSoftwareToken
- ChangePassword
- ConfirmDevice
- ConfirmForgotPassword
- ConfirmSignUp
- DeleteUser
- DeleteUserAttributes
- ForgetDevice
- ForgotPassword
- GetDevice
- GetUser
- GetUserAttributeVerificationCode
- GlobalSignOut
- InitiateAuth
- ListDevices
- ResendConfirmationCode
- RespondToAuthChallenge
- RevokeToken
- SetUserMFAPreference
- SetUserSettings
- SignUp
- UpdateAuthEventFeedback
- UpdateDeviceStatus

- `UpdateUserAttributes`
- `VerifySoftwareToken`
- `VerifyUserAttribute`

Amazon Cognito 자격 증명 기반 정책 예제를 보려면 Amazon Cognito의 자격 증명 기반 정책 예제 섹션을 참조하세요.

Amazon Cognito의 정책 리소스

정책 리소스 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Resource` JSON 정책 요소는 작업이 적용되는 하나 이상의 객체를 지정합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

Amazon 리소스 이름(ARN)

Amazon Cognito 페더레이션 자격 증명의 ARN

Amazon Cognito 자격 증명 풀(페더레이션 자격 증명)에서는 다음 예와 같이 Amazon 리소스 이름(ARN) 형식을 사용하여 특정 자격 증명 풀에 대한 IAM 사용자의 액세스를 제한할 수 있습니다. ARN에 대한 자세한 내용은 IAM 식별자를 참조하세요.

```
arn:aws:cognito-identity:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID
```

Amazon Cognito Sync의 ARN

Amazon Cognito Sync에서도 고객은 자격 증명 풀 ID, 자격 증명 ID 및 데이터 집합 이름으로 액세스를 제한할 수 있습니다.

자격 증명 풀에 대해 작동하는 API의 경우, 자격 증명 풀 ARN 형식은 Amazon Cognito 페더레이션 자격 증명과 동일하지만 서비스 이름이 `cognito-identity`가 아니라 `cognito-sync`입니다

```
arn:aws:cognito-sync:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID
```

RegisterDevice와 같이 단일 자격 증명에 대해 작동하는 API의 경우, 다음 ARN 형식으로 개별 자격 증명을 참조할 수 있습니다.

```
arn:aws:cognito-sync:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID/
identity/IDENTITY_ID
```

UpdateRecords 및 ListRecords와 같이 데이터 집합에 대해 작동하는 API의 경우, 다음 ARN 형식을 사용하여 개별 데이터 집합을 참조할 수 있습니다.

```
arn:aws:cognito-sync:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID/
identity/IDENTITY_ID/dataset/DATASET_NAME
```

Amazon Cognito 사용자 풀의 ARN

Amazon Cognito 사용자 풀의 경우 다음 ARN 형식을 사용하여 특정 사용자 풀에 대한 사용자의 액세스를 제한할 수 있습니다.

```
arn:aws:cognito-idp:REGION:ACCOUNT_ID:userpool/USER_POOL_ID
```

Amazon Cognito 리소스 유형 및 해당 ARN의 목록을 보려면 서비스 권한 부여 참조에서 Amazon Cognito에서 정의한 리소스를 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 Amazon Cognito에서 정의한 작업을 참조하세요.

Amazon Cognito 자격 증명 기반 정책 예제를 보려면 Amazon Cognito의 자격 증명 기반 정책 예제 섹션을 참조하세요.

Amazon Cognito에 사용되는 정책 조건 키

서비스별 정책 조건 키 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

Condition 요소는 정의된 기준에 따라 문이 실행되는 시기를 지정합니다. 같음(equals) 또는 미만(less than)과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

Amazon Cognito 조건 키 목록을 보려면 서비스 권한 부여 참조에서 Amazon Cognito의 조건 키를 참조하세요. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 Amazon Cognito에서 정의한 작업을 참조하세요.

Amazon Cognito 자격 증명 기반 정책 예제를 보려면 Amazon Cognito의 자격 증명 기반 정책 예제 섹션을 참조하세요.

Amazon Cognito의 액세스 제어 목록(ACL)

ACL 지원: 아니요

액세스 제어 목록(ACL)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Amazon Cognito에서 속성 기반 액세스 제어(ABAC)

ABAC 지원(정책의 태그): 부분적

속성 기반 액세스 제어(ABAC)는 태그라고 불리는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. IAM 엔터티 및 AWS 리소스에 태그를 연결한 다음 보안 주체의 태그가 리소스의 태그와 일치할 때 작업을 허용하는 ABAC 정책을 설계할 수 있습니다.

태그에 근거하여 액세스를 제어하려면 `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`또는 `aws:TagKeys`조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다.

서비스가 모든 리소스 유형에 대해 세 가지 조건 키를 모두 지원하는 경우, 값은 서비스에 대해 예입니다. 서비스가 일부 리소스 유형에 대해서만 세 가지 조건 키를 모두 지원하는 경우, 값은 부분적입니다.

ABAC에 대한 자세한 내용은 IAM 사용 설명서의 ABAC 권한 부여를 통한 권한 정의를 참조하세요. ABAC 설정 단계가 포함된 자습서를 보려면 IAM 사용 설명서의 속성 기반 액세스 제어(ABAC) 사용을 참조하세요.

Amazon Cognito에서 임시 보안 인증 정보 사용

임시 자격 증명 지원: 예

임시 자격 증명은 AWS 리소스에 대한 단기 액세스를 제공하며 페더레이션을 사용하거나 역할을 전환할 때 자동으로 생성됩니다. 장기 액세스 키를 사용하는 대신 임시 자격 증명을 동적으로 생성하는 것이 AWS 좋습니다. 자세한 내용은 IAM 사용 설명서의 IAM의 임시 보안 자격 증명 및 IAM으로 작업하는AWS 서비스 섹션을 참조하세요.

Amazon Cognito의 서비스 간 보안 주체 권한

전달 액세스 세션(FAS) 지원: 아니요

전달 액세스 세션(FAS)은를 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대한 요청과 AWS 서비스함께 사용합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

Amazon Cognito의 서비스 역할

서비스 역할 지원: 예

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 가정하는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS 서비스 AWS에 권한을 위임할 역할 생성을 참조하세요.

Amazon Cognito 서비스 역할에 대한 자세한 내용은 푸시 동기화 활성화 및 푸시 동기화 구현 섹션을 참조하세요.

> **Warning**
>
> 서비스 역할에 대한 권한을 변경하면 Amazon Cognito 기능이 중단될 수 있습니다. Amazon Cognito가 관련 지침을 제공하는 경우에만 서비스 역할을 편집하세요.

Amazon Cognito의 서비스 연결 역할

서비스 연결 역할 지원: 예

서비스 연결 역할은에 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 태스크를 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은에 표시 AWS 계정 되며 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

Amazon Cognito 서비스 연결 역할 생성 또는 관리에 대한 자세한 내용은 Amazon Cognito에 서비스 연결 역할 사용 섹션을 참조하세요.

Amazon Cognito의 자격 증명 기반 정책 예제

기본적으로 사용자 및 역할은 Amazon Cognito 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성(콘솔)을 참조하세요.

각 리소스 유형에 대한 ARN 형식을 비롯하여 Amazon Cognito에서 정의되는 작업 및 리소스 유형에 대한 자세한 내용은 서비스 승인 참조의 Amazon Cognito에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

주제

- 정책 모범 사례
- Amazon Cognito 콘솔 사용
- 사용자가 자신의 고유한 권한을 볼 수 있도록 허용
- 특정 자격 증명 풀로 콘솔 액세스 제한
- 풀의 모든 자격 증명에 특정 데이터 집합에 대한 액세스 허용

정책 모범 사례

ID 기반 정책에 따라 계정에서 사용자가 Amazon Cognito 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따르세요.

- AWS 관리형 정책을 시작하고 최소 권한으로 전환 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반적인 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. 에서 사용할 수 있습니다 AWS 계정. 사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 추가로 줄이는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책 또는 AWS 직무에 대한 관리형 정책을 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우, 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 IAM의 정책 및 권한을 참조하세요.
- IAM 정책의 조건을 사용하여 액세스 추가 제한 – 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어, SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. AWS 서비스와 같은 특정를 통해 사용되는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다 CloudFormation. 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 확인하여 안전하고 기능적인 권한 보장 - IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 새로운 및 기존 정책을

확인합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer에서 정책 검증을 참조하세요.

- 다중 인증(MFA) 필요 -에서 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를 AWS 계정켭니다. API 작업을 직접적으로 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 내용은 IAM 사용 설명서의 MFA를 통한 보안 API 액세스를 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

> **Note**
>
> Amazon Cognito 콘솔의 원래 버전과 새 버전은 Amazon Cognito 리소스를 보고 수정할 때 기본 동작이 다릅니다. `aws:ViaAWSService` 조건이 true인 경우에만 `cognito-idp` 서비스 접두사에서 작업에 대한 권한을 부여한 경우, 영향을 받는 IAM 보안 주체가 원래 콘솔에서 Amazon Cognito 리소스에 유효할 수 있겠지만 새 콘솔에서는 그렇지 않습니다. Amazon Cognito 콘솔에서 작업하려면 IAM 정책에서 Amazon Cognito 권한에 대한 `aws:ViaAWSService` 조건을 설정하지 마세요.

Amazon Cognito 콘솔 사용

Amazon Cognito 콘솔에 액세스하려면 최소한의 권한 세트가 있어야 합니다. 이러한 권한은에서 Amazon Cognito 리소스에 대한 세부 정보를 나열하고 볼 수 있도록 허용해야 합니다 AWS 계정. 최소 필수 권한보다 더 제한적인 ID 기반 정책을 생성하는 경우, 콘솔이 해당 정책에 연결된 엔티티(사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

AWS CLI 또는 AWS API만 호출하는 사용자에게는 최소 콘솔 권한을 허용할 필요가 없습니다. 대신, 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.

사용자와 역할이 여전히 Amazon Cognito 콘솔을 사용할 수 있도록 하려면 Amazon Cognito `ConsoleAccess` 또는 `ReadOnly` AWS 관리형 정책을 엔터티에 연결합니다. 자세한 내용은 IAM 사용 설명서의 사용자에게 권한 추가를 참조하세요.

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예제는 IAM 사용자가 자신의 사용자 ID에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 AWS CLI 또는 AWS API를 사용하여 프로그래밍 방식으로이 작업을 완료할 수 있는 권한이 포함됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

특정 자격 증명 풀로 콘솔 액세스 제한

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "cognito-identity:ListIdentityPools"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cognito-identity:*"
            ],
            "Resource": "arn:aws:cognito-identity:us-
east-1:111122223333:identitypool/us-east-1:1a1a1a1a-ffff-1111-9999-12345678"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cognito-sync:*"
            ],
            "Resource": "arn:aws:cognito-sync:us-
east-1:111122223333:identitypool/us-east-1:1a1a1a1a-ffff-1111-9999-12345678"
        }
    ]
}
```

풀의 모든 자격 증명에 특정 데이터 집합에 대한 액세스 허용

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cognito-sync:ListRecords",
                "cognito-sync:UpdateRecords"
            ],
            "Resource": "arn:aws:cognito-sync:us-
east-1:111122223333:identitypool/us-east-1:1a1a1a1a-ffff-1111-9999-12345678/
identity/*/dataset/UserProfile"
        }
```

```
    ]
}
```

Amazon Cognito 자격 증명 및 액세스 문제 해결

다음 정보를 사용하여 Amazon Cognito와 IAM에서 작업할 때 발생할 수 있는 공통적인 문제를 진단하고 수정할 수 있습니다.

주제

- Amazon Cognito에서 작업을 수행할 권한이 없음
- iam:PassRole을 수행하도록 인증되지 않음
- 관리자인데, 다른 사용자가 Amazon Cognito에 액세스하도록 허용하기를 원함
- 내 AWS 계정 외부의 사람이 내 Amazon Cognito 리소스에 액세스하도록 허용하고 싶습니다.

Amazon Cognito에서 작업을 수행할 권한이 없음

작업을 수행할 권한이 없다는 오류가 표시되면 작업을 수행할 수 있도록 정책을 업데이트해야 합니다.

다음의 예제 오류는 `mateojackson` IAM 사용자가 콘솔을 사용하여 가상 *`my-example-widget`* 리소스에 대한 세부 정보를 보려고 하지만 가상 `cognito-identity:`*`GetWidget`* 권한이 없을 때 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 cognito-identity:GetWidget on resource: my-example-widget
```

이 경우, `cognito-identity:`*`GetWidget`* 작업을 사용하여 *`my-example-widget`* 리소스에 액세스할 수 있도록 `mateojackson` 사용자 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

iam:PassRole을 수행하도록 인증되지 않음

`iam:PassRole` 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 Amazon Cognito에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스 에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 역할을 서비스에 전달할 권한이 있어야 합니다.

다음 예제 오류는 marymajor라는 IAM 사용자가 콘솔을 사용하여 Amazon Cognito에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 권한이 없습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우, Mary가 iam:PassRole작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

관리자인데, 다른 사용자가 Amazon Cognito에 액세스하도록 허용하기를 원함

다른 사용자가 Amazon Cognito에 액세스하도록 허용하려면 액세스가 필요한 사람 또는 애플리케이션에 권한을 부여해야 합니다. AWS IAM Identity Center 를 사용하여 사용자 및 애플리케이션을 관리하는 경우 사용자 또는 그룹에 권한 세트를 할당하여 액세스 수준을 정의합니다. 권한 세트는 IAM 정책을 자동으로 생성하고 사용자 또는 애플리케이션과 연결된 IAM 역할에 할당합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 권한 세트를 참조하세요.

IAM Identity Center를 사용하지 않는 경우 액세스가 필요한 사용자 또는 애플리케이션에 대한 IAM 엔터티(사용자 또는 역할)를 생성해야 합니다. 그런 다음 Amazon Cognito에서 올바른 권한을 부여하는 정책을 엔터티에 연결해야 합니다. 권한이 부여되면 사용자 또는 애플리케이션 개발자에게 자격 증명을 제공합니다. 이들은 이 자격 증명을 사용하여 AWS에 액세스합니다. IAM 사용자, 그룹, 정책 및 권한 생성에 대해 자세히 알아보려면 IAM 사용자 설명서의 IAM 자격 증명과 IAM의 권한 및 정책을 참조하세요.

내 AWS 계정 외부의 사람이 내 Amazon Cognito 리소스에 액세스하도록 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우, 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세한 내용은 다음을 참조하세요.

- Amazon Cognito에서 이러한 기능을 지원하는지 여부를 알아보려면 Amazon Cognito에서 IAM을 사용하는 방법 섹션을 참조하세요.
- 소유 AWS 계정 한의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 소유한 다른의 IAM 사용자에게 액세스 권한 제공을 참조 AWS 계정 하세요.

- 타사에 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사가 AWS 계정 소유한에 대한 액세스 권한 제공을 AWS 계정참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하세요.
- 크로스 계정 액세스에 대한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 크로스 계정 리소스 액세스를 참조하세요.

Amazon Cognito에 서비스 연결 역할 사용

Amazon Cognito는 AWS Identity and Access Management (IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은가 역할을 수임 AWS 서비스 하도록 허용하는 신뢰 정책이 있는 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 Amazon Cognito에서 사전 정의하며 서비스가 사용자를 대신하여 다른 AWS 서비스를 호출하는 데 필요한 모든 권한을 포함합니다.

필요한 권한을 수동으로 추가할 필요가 없으므로 서비스 연결 역할은 Amazon Cognito를 더 쉽게 설정할 수 있습니다. Amazon Cognito에서 서비스 연결 역할의 권한을 정의하므로 다르게 정의되지 않은 한, Amazon Cognito만 해당 역할을 수임할 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며 이 권한 정책은 다른 IAM 엔터티에 연결할 수 없습니다.

먼저 관련 리소스를 삭제한 후에만 서비스 연결 역할을 삭제할 수 있습니다. 이렇게 하면 리소스에 대한 액세스 권한을 실수로 삭제할 수 없기 때문에 Amazon Cognito 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용은 IAM으로 작업하는AWS 서비스를 참조하고 서비스 연결 역할 열에 예가 있는 서비스를 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예(Yes) 링크를 선택합니다.

Amazon Cognito에 대한 서비스 연결 역할 권한

Amazon Cognito는 다음 서비스 연결 역할을 사용합니다.

- AWSServiceRoleForAmazonCognitoIdpEmailService - Amazon Cognito 사용자 풀 서비스가 이메일 전송에 Amazon SES 자격 증명을 사용하도록 허용합니다.

- AWSServiceRoleForAmazonCognitoIdp - Amazon Cognito 사용자 풀이 이벤트를 게시하고 Amazon Pinpoint 프로젝트의 엔드포인트를 구성하도록 허용합니다.

AWSServiceRoleForAmazonCognitoIdpEmailService

AWSServiceRoleForAmazonCognitoIdpEmailService 서비스 연결 역할은 역할을 수임하기 위해 다음 서비스를 신뢰합니다.

- email.cognito-idp.amazonaws.com

역할 권한 정책은 Amazon Cognito가 지정된 리소스에서 다음 작업을 완료하도록 허용합니다.

AWSServiceRoleForAmazonCognitoIdpEmailService에 허용되는 작업:

- 작업: ses:SendEmail 및 ses:SendRawEmail
- 리소스: *

이 정책은 Amazon Cognito가 지정된 리소스에서 다음 작업을 완료하는 기능을 거부합니다.

거부된 작업

- 작업: ses:List*
- 리소스: *

이러한 권한을 통해 Amazon Cognito는 Amazon SES에서 확인된 이메일 주소를 사용자에게 이메일로 보내는 용도로만 사용할 수 있습니다. Amazon Cognito는 사용자가 클라이언트 앱에서 사용자 풀에 대한 특정 작업(예: 가입 또는 암호 재설정)을 수행할 때 사용자에게 이메일을 보냅니다.

IAM 엔터티(사용자, 그룹, 역할 등)가 서비스 연결 역할을 생성하고 편집하거나 삭제할 수 있도록 권한을 구성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 권한을 참조하세요.

AWSServiceRoleForAmazonCognitoIdp

AWSServiceRoleForAmazonCognitoIdp 서비스 연결 역할은 역할을 수임하기 위해 다음 서비스를 신뢰합니다.

- email.cognito-idp.amazonaws.com

역할 권한 정책은 Amazon Cognito가 지정된 리소스에서 다음 작업을 완료하도록 허용합니다.

AWSServiceRoleForAmazonCognitoIdp에 허용되는 작업

- 작업: cognito-idp:Describe

- 리소스: *

이 권한을 통해 Amazon Cognito는 사용자를 위해 `Describe` Amazon Cognito API 작업을 호출할 수 있습니다.

> **Note**
>
> `createUserPoolClient` 및를 사용하여 Amazon Cognito를 Amazon Pinpoint와 통합하면 `updateUserPoolClient`리소스 권한이 SLR에 인라인 정책으로 추가됩니다. 인라인 정책은 `mobiletargeting:UpdateEndpoint` 및 `mobiletargeting:PutEvents` 권한을 제공합니다. 이러한 권한을 통해 Amazon Cognito는 Cognito와 통합하는 Pinpoint 프로젝트의 이벤트를 게시하고 엔드포인트를 구성할 수 있습니다.

Amazon Cognito에 대한 서비스 연결 역할 생성

서비스 연결 역할은 수동으로 생성할 필요가 없습니다. Amazon SES 구성을 사용하여 AWS CLI, 또는 Amazon Cognito API에서 이메일 전송을 처리하도록 사용자 풀 AWS Management Console을 구성하면 Amazon Cognito가 서비스 연결 역할을 생성합니다.

이 서비스 연결 역할을 삭제했다가 다시 생성해야 하는 경우 동일한 프로세스를 사용하여 계정에서 역할을 다시 생성할 수 있습니다. Amazon SES 구성을 사용하여 이메일 전송을 처리하도록 사용자 풀을 구성할 경우 Amazon Cognito가 서비스 연결 역할을 다시 생성합니다.

Amazon Cognito가 이 역할을 생성하려면 먼저 사용자 풀을 설정하는 데 사용하는 IAM 권한에 `iam:CreateServiceLinkedRole` 작업을 포함해야 합니다. IAM에서의 권한 업데이트에 대한 자세한 내용은 IAM 사용 설명서에서 IAM 사용자의 권한 변경을 참조하세요.

Amazon Cognito에 대한 서비스 연결 역할 편집

에서 AmazonCognitoIdp 또는 AmazonCognitoIdpEmailService 서비스 연결 역할을 편집할 수 없습니다 AWS Identity and Access Management. 서비스 연결 역할을 생성한 후에는 다양한 엔터티가 역할을 참조할 수 있기 때문에 역할 이름을 변경할 수 없습니다. 하지만 IAM을 사용하여 역할의 설명을 편집할 수 있습니다. 자세한 내용은 IAM 사용자 설명서의 서비스 연결 역할 편집을 참조하세요.

Amazon Cognito에 대한 서비스 연결 역할 삭제

서비스 연결 역할이 필요한 특성 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제하는 것이 좋습니다. 이러한 역할을 삭제하면 Amazon Cognito가 적극적으로 모니터링하거나 유지하는 엔터

티만 보유합니다. AmazonCognitoIdp 또는 AmazonCognitoIdpEmailService 서비스 연결 역할을 삭제하려면 먼저 해당 역할을 사용하는 각 사용자 풀에 대해 다음 중 하나를 수행해야 합니다.

- 사용자 풀을 삭제합니다.
- 기본 이메일 기능을 사용하도록 사용자 풀의 이메일 설정을 업데이트합니다. 기본 설정에서는 서비스 연결 역할을 사용하지 않습니다.

역할을 사용하는 사용자 풀을 AWS 리전 사용하여 각에서 작업을 수행해야 합니다.

> **Note**
>
> 리소스를 삭제하려고 할 때 Amazon Cognito 서비스가 역할을 사용 중이면 삭제에 실패할 수 있습니다. 이 문제가 발생하면 몇 분 기다렸다가 작업을 다시 시도하세요.

Amazon Cognito 사용자 풀을 삭제하려면

1. 에 로그인 AWS Management Console 하고에서 Amazon Cognito 콘솔을 엽니다https://console.aws.amazon.com/cognito.
2. 사용자 풀 관리를 선택합니다.
3. [사용자 풀(Your User Pools)] 페이지에서 삭제할 사용자 풀을 선택합니다.
4. 풀 삭제를 선택합니다.
5. [사용자 풀 삭제(Delete user pool)] 창에서 **delete**를 입력하고 [풀 삭제(Delete pool)]를 선택합니다.

기본 이메일 기능을 사용하도록 Amazon Cognito 사용자 풀을 업데이트하려면

1. 에 로그인 AWS Management Console 하고에서 Amazon Cognito 콘솔을 엽니다https://console.aws.amazon.com/cognito.
2. 사용자 풀 관리를 선택합니다.
3. [사용자 풀(Your User Pools)] 페이지에서 업데이트할 사용자 풀을 선택합니다.
4. 왼쪽의 탐색 메뉴에서 메시지 사용자 지정을 선택합니다.
5. [Amazon SES 구성을 통해 이메일을 보내시겠습니까?(Do you want to send emails through your Amazon SES Configuration?)] 아래에서 [아니요 - Cognito(기본값)을 사용합니다(No - Use Cognito (Default))]를 선택합니다.

6. 이메일 계정 옵션 설정을 마치면 변경 사항 저장을 선택합니다.

IAM을 사용하여 수동으로 서비스 연결 역할을 삭제하려면 다음을 수행하세요.

IAM 콘솔 AWS CLI, 또는 AWS API를 사용하여 AmazonCognitoIdp 또는 AmazonCognitoIdpEmailService 서비스 연결 역할을 삭제합니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 삭제를 참조하세요.

Amazon Cognito 서비스 연결 역할이 지원되는 리전

Amazon Cognito는 서비스를 사용할 수 AWS 리전 있는 모든에서 서비스 연결 역할을 지원합니다. 자세한 내용은 AWS 리전 및 엔드포인트를 참조하세요.

Amazon Cognito의 로깅 및 모니터링

모니터링은 Amazon Cognito 및 기타 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 데 중요한 부분입니다. Amazon Cognito는 현재 다음 AWS 서비스 를 지원하므로, 조직 및 조직 내에서 발생하는 작업을 모니터링할 수 있습니다.

- AWS CloudTrail - CloudTrail을 사용하면 Amazon Cognito 콘솔에서 API 호출을 캡처하고 Amazon Cognito API 작업에 대한 코드 호출에서 API 호출을 캡처할 수 있습니다. 예를 들어 사용자가 인증할 때 CloudTrail은 요청의 IP 주소, 요청한 사람, 요청한 시기와 같은 세부 정보를 기록할 수 있습니다.
- Amazon CloudWatch Logs - CloudWatch Logs를 사용하면 사용자 활동에 대한 세밀한 로그를 로그 그룹에 보낼 수 있습니다. 예를 들어, 자세한 사용자 활동 로그를 검토하여 사용자에게 이메일과 SMS 메시지를 전송할 때의 문제를 해결할 수 있습니다.
- Amazon CloudWatch 지표 – CloudWatch 지표를 사용하면 이벤트 발생 시 거의 실시간으로 모니터링 및 보고하고 자동 조치를 취할 수 있습니다. 예를 들어 제공된 지표에 대한 CloudWatch 대시보드를 만들어 Amazon Cognito 사용자 풀을 모니터링하거나 제공된 지표에 대한 CloudWatch 경보를 생성하여, 설정 임계값 위반이 발생할 경우 알림을 받을 수 있습니다.
- Amazon CloudWatch Logs Insights – CloudWatch Logs Insights를 사용하면 Amazon Cognito CloudTrail 로그 파일을 모니터링하기 위해 CloudWatch로 이벤트를 전송하도록 CloudTrail을 구성할 수 있습니다.

주제

- 비용 모니터링 및 관리
- Amazon Cognito 사용자 풀에서 로그 내보내기

- CloudWatch 및 Service Quotas에서 할당량 및 사용량 추적
- Amazon Cognito 로그인 AWS CloudTrail

비용 모니터링 및 관리

다른와 마찬가지로 Amazon Cognito 구성 및 사용량이 AWS 청구서에 미치는 영향을 이해하는 AWS 서비스것이 중요합니다. 프로덕션에 사용자 풀을 배포하기 위한 준비의 일환으로 활동 및 리소스 소비에 대한 모니터링 및 보호 조치를 설정합니다. 어디에서 볼 수 있고 어떤 작업이 추가 비용을 발생시키는지 알면 청구서의 놀라움에 대한 예방 조치를 설정할 수 있습니다.

Amazon Cognito는 다음 사용량 차원에 대해 요금을 부과합니다.

- 사용자 풀 월간 활성 사용자(MAU) 비율은 기능 플랜에 따라 다릅니다.
- OIDC 또는 SAML 페더레이션으로 로그인한 사용자 풀 MAU
- 토큰 엔드포인트의 `client_credentials` 권한 부여와 `GetClientToken` API 작업 모두에서 기계 간(M2M) 권한 부여를 위한 요청 볼륨
- 사용자 풀 API의 일부 카테고리에 대해 기본 할당량을 초과하여 구매한 사용량

또한 이메일 메시지, SMS 메시지 및 Lambda 트리거와 같은 사용자 풀의 기능에 따라 종속 서비스 비용이 발생할 수 있습니다. 전체 개요는 Amazon Cognito 요금 섹션을 참조하세요.

비용 보기 및 예측

제품 출시 및 새 사용자 기반 열기와 같은 대량 이벤트는 MAU 수를 늘리고 비용에 영향을 미칠 수 있습니다. 새 사용자 수를 미리 예측하고 발생하는 활동을 관찰합니다. 추가 할당량을 구매하여 볼륨을 수용하거나 추가 보안 조치를 사용하여 볼륨을 제어하려는 경우가 있습니다.

AWS 결제 및 비용 관리 콘솔에서 AWS 비용을 보고 확인할 수 있습니다. 청구 및 결제 섹션에서 Amazon Cognito에 대한 최신 요금을 확인할 수 있습니다. 청구서, 서비스별 요금에서 `Cognito`를 필터링하여 사용량을 확인합니다. 자세한 내용은 AWS Billing 사용 설명서에서 결제 보기를 참조하세요.

API 요청 속도를 모니터링하려면 Service Quotas 콘솔에서 사용률 지표를 검토합니다. 예를 들어 토큰 엔드포인트에 대한 `client_credentials` 권한 부여 요청과 GetClientToken API 요청은 모두 요청 할당량을 공유하므로 ClientAuthentication 요청 속도로 표시됩니다. 청구서에서 이러한 요청은 요청을 생성한 앱 클라이언트와 연결됩니다. 이 정보를 사용하면 다중 테넌트 아키텍처의 테넌트에게 비용을 공평하게 할당할 수 있습니다.

일정 기간 동안의 토큰 엔드포인트 M2M 요청 수를 가져오려면 AWS CloudTrail CloudWatch Logs로 이벤트를 전송하여 분석할 수도 있습니다. 클라이언트 자격 증명이 부여된 Token_POST 이벤트에 대해 CloudTrail 이벤트를 쿼리하세요. 다음 CloudWatch 인사이트 쿼리는 이 개수를 반환합니다. GetClientToken API 작업으로 수행한 M2M 요청을 포함하려면이 섹션 앞부분에서 설명한 ClientAuthentication 요청 사용률 지표를 사용합니다.

```
filter eventName = "Token_POST" and @message like '"grant_type":["client_credentials"]'
 | stats count(*)
```

비용 관리

Amazon Cognito는 사용자 수, 기능 사용량 및 요청 볼륨을 기준으로 요금을 청구합니다. 다음은 Amazon Cognito에서 비용을 관리하기 위한 몇 가지 팁입니다.

비활성 사용자 활성화하지 않기

사용자를 활성화하는 일반적인 작업은 로그인, 가입 및 암호 재설정입니다. 보다 자세한 목록은 월별 활성 사용자) 섹션을 참조하세요. Amazon Cognito는 비활성 사용자를 청구서에 계산하지 않습니다. 사용자를 활성 상태로 설정하는 모든 작업을 피하세요. AdminGetUser API 작업 대신 ListUsers 작업을 사용하여 사용자를 쿼리합니다. 비활성 사용자를 대상으로 사용자 풀 작업에 대한 대량 관리 테스트를 수행하지 마세요.

페더레이션 사용자 연결

SAML 2.0 또는 OpenID Connect(OIDC) ID 제공업체로 로그인하는 사용자는 로컬 사용자보다 비용이 더 높습니다. 이러한 사용자를 로컬 사용자 프로필에 연결할 수 있습니다. 연결된 사용자는 페더레이션 사용자와 함께 제공되는 속성 및 액세스 권한을 사용하여 로컬 사용자로 로그인할 수 있습니다. 한 달 동안 연결된 로컬 계정으로만 로그인하는 SAML 또는 OIDC IdP의 사용자는 로컬 사용자로 요금이 청구됩니다.

요청 속도 관리

사용자 풀이 할당량 상한에 도달하는 경우 볼륨을 처리하기 위한 추가 용량 구매를 고려할 수 있습니다. 애플리케이션의 요청 볼륨을 줄일 수도 있습니다. 자세한 내용은 할당량 한도에 대한 요청 속도 최적화 섹션을 참조하세요.

필요한 경우에만 새 토큰 요청

클라이언트 자격 증명 부여를 사용하여 기계 간(M2M)에 권한 부여하면 대량의 토큰 요청이 발생할 수 있습니다. 각 새 토큰 요청은 요청 속도 할당량과 청구서 크기에 영향을 미칩니다. 비용을 최적화하려면 애플리케이션 설계에 토큰 만료 설정 및 토큰 처리를 포함하세요.

- 애플리케이션이 새 토큰을 요청할 때 이전에 발급된 토큰의 캐시된 버전을 수신하도록 액세스 토큰을 캐시합니다. 이 방법을 구현하면 캐싱 프록시가 이전에 획득한 토큰의 만료를 인식하지 못하고 액세스 토큰을 요청하는 애플리케이션에 대한 보호 역할을 합니다. 캐싱 토큰은 Lambda 함수 및 Docker 컨테이너와 같은 수명이 짧은 마이크로서비스에 적합합니다.
- 토큰 만료를 설명하는 애플리케이션에 토큰 처리 메커니즘을 구현합니다. 이전 토큰이 만료될 때까지 새 토큰을 요청하지 마세요. 가장 좋은 방법은 토큰 수명 중 약 75%에서 토큰을 새로 고치는 것입니다. 이 방법은 애플리케이션의 사용자 연속성을 보장하면서 토큰 기간을 극대화합니다.

  각 애플리케이션의 기밀성 및 가용성 요구 사항을 평가하고 적절한 유효 기간이 있는 액세스 토큰을 발급하도록 사용자 풀 앱 클라이언트를 구성합니다. 사용자 지정 토큰 기간은 자격 증명 요청 빈도를 지속적으로 관리할 수 있는 수명이 긴 API 및 서버에 가장 적합합니다.

AdminGetUser가 아닌 ListUsers

사용자 풀에 있는 사용자의 속성을 쿼리하려면 가능한 경우 ListUsers API 작업 및 관련 SDK 메서드를 사용합니다. AdminGetUser는 사용자를 해당 월에 활성으로 표시하고 사용자 풀에 대한 청구서를 계산하는 데 사용되는 월별 활성 사용자(MAU)에 기여합니다.

기능 플랜 관리

사용자 풀에서 기능 플랜을 선택하면 사용자 풀의 모든 MAU에 청구 요금이 적용됩니다. 상위 수준 기능 플랜과 함께 제공되는 기능이 필요하지 않은 사용자가 있는 경우 다른 사용자 풀로 분리합니다.

Amazon Cognito 사용자 풀에서 로그 내보내기

CloudWatch 로그 그룹과 AWS 서비스같은 일부 추가 활동에 대한 세부 로그를 다른에 보내도록 사용자 풀을 구성할 수 있습니다. 이러한 로그는의 로그보다 세분화되어 AWS CloudTrail있으며 사용자 풀 문제를 해결하고 위협 방지를 통해 사용자 로그인 활동을 분석하는 데 유용할 수 있습니다. SMS 및 이메일 알림 오류의 로그를 스트리밍하려는 경우 사용자 풀은 CloudWatch 로그 그룹에 `ERROR` 수준 로그를 보냅니다. 사용자 로그인 활동의 로그를 스트리밍하려는 경우 사용자 풀은 로그 그룹, Amazon Data Firehose 스트림 또는 Amazon S3 버킷에 `INFO` 수준 로그를 전송합니다. 사용자 풀에서 두 옵션을 결합할 수 있습니다.

주제

- 로그 내보내기에 대해 알아야 할 사항
- 이메일 및 SMS 메시지 전송 오류 내보내기
- 위협 방지 사용자 활동 로그 내보내기

로그 내보내기에 대해 알아야 할 사항

비용 영향

Amazon Data Firehose, Amazon S3 및 CloudWatch Logs에는 데이터 수집 및 검색 비용이 발생합니다. 로깅 구성이 AWS 청구서에 영향을 미칠 수 있습니다. 자세한 내용은 다음을 참조하세요.

- Amazon CloudWatch 요금의 벤딩된 로그.
- Amazon Data Firehose 요금
- Amazon S3 요금

사용자 활동 로그 내보내기에는 보안 평가가 포함되며 사용자 풀 위협 방지의 함수입니다. Amazon Cognito는 위협 방지가 감사 전용 또는 전체 함수 모드이고 사용자 풀이 Plus 기능 플랜에 있는 경우에만 이러한 로그를 생성합니다.

사용자 활동 로그는 **`INFO`** 레벨입니다.

내보낸 사용자 활동 로그는 `INFO` 오류 레벨에만 있으며 인증 활동의 통계 및 보안 분석을 위한 정보를 제공합니다. 제한 오류와 같은 `WARNING` 및 `ERROR` 오류 레벨의 메시지는 내보낸 로그에 포함되지 않습니다.

최선의 작업 제공

Amazon Cognito에서 로그를 전송하는 것은 최선의 작업을 기반으로 합니다. 사용자 풀이 제공하는 로그의 양과 CloudWatch Logs, Amazon S3 및 Firehose에 대한 서비스 할당량은 로그 제공에 영향을 미칠 수 있습니다.

기존 외부 로그는 영향을 받지 않습니다.

이러한 로깅 옵션은 사용자 풀의 다음 로그 기능을 대체하거나 변경하지 않습니다.

1. 가입 및 로그인과 같은 일상적인 사용자 활동의 CloudTrail 로그.
2. CloudWatch 지표를 사용한 대규모 사용자 활동 분석.

이와 별도로 CloudWatch 로그에서 CloudWatch 콘솔에서 사용자 풀 가져오기 결과 보기 및 Lambda 트리거를 사용하여 사용자 풀 워크플로 사용자 정의의 로그도 찾을 수 있습니다. Amazon Cognito와 Lambda는 사용자 로그에 지정한 로그 그룹과는 다른 로그 그룹에 이러한 로그를 저장합니다.

사용자 풀에만 적용

ID 풀에는 로그 내보내기 기능이 없습니다.

사용자 권한 및 서비스 연결 역할이 필요합니다.

로그 내보내기를 설정하는 AWS 보안 주체는 다음 주제에 설명된 대로 대상 리소스를 수정할 수 있는 권한이 있어야 합니다. Amazon Cognito는 사용자를 대신하여 서비스 연결 역할을 생성하고 대상 리소스에 로그를 전달하는 역할을 담당합니다.

Amazon Cognito에서 로그를 보내기 위한 권한 부여 모델에 대한 자세한 내용은 Amazon CloudWatch Logs 사용 설명서의 AWS 서비스에서 로깅 활성화를 참조하세요.

로그 수준은 로그 유형에만 해당됩니다.

메시지 전송 로그는 `userNotification` 유형 및 `ERROR` 오류 수준의 로그입니다. 고급 보안 사용자 활동 로그는 `userAuthEvents` 유형 및 `INFO` 오류 수준입니다. `LogConfigurations`의 두 멤버, 즉 `userNotification`에서 CloudWatch Logs로, `userAuthEvents`에서 Firehose, Amazon S3 또는 CloudWatch Logs로 각각 다른 멤버를 결합할 수 있습니다.

여러 대상으로 사용자 활동 로그를 보낼 수 없습니다. CloudWatch Logs 이외의 대상에는 사용자 알림 로그를 보낼 수 없습니다.

다양한 구성 옵션

Amazon Cognito 사용자 풀 API 또는 AWS SDK를 통해서만 사용자 알림 로그를 구성할 수 있습니다. API를 사용하거나 Amazon Cognito 콘솔에서 고급 보안 사용자 활동 로그를 구성할 수 있습니다. 둘 다 설정하려면 SetLogDeliveryConfiguration의 예제 요청에 설명된 대로 API를 사용합니다.

대규모 리소스 기반 정책에 필요한 추가 구성

리소스 정책 크기가 5,120자보다 큰 로그 그룹에 로그를 전송하려면 경로가 `/aws/vendedlogs`로 시작하는 로그 그룹을 구성합니다. 자세한 내용은 특정 AWS 서비스에서 로깅 활성화를 참조하세요.

Amazon S3에서 폴더 자동 생성

Amazon S3 버킷으로 위협 방지 로그 내보내기를 구성하면 Amazon Cognito가 버킷에 `AWSLogs` 폴더를 생성할 수 있습니다. 이 폴더는 모든 경우에 생성되지 않으며 구성은 생성하지 않고 성공할 수 있습니다.

이메일 및 SMS 메시지 전송 오류 내보내기

이메일 및 SMS 메시지 전송 오류의 경우 사용자 풀에서 오류 수준 사용자 알림 로그를 전달할 수 있습니다. 이 기능을 활성화하면 Amazon Cognito에서 로그를 전송할 로그 그룹을 선택할 수 있습니다. 사

용자 알림 로깅은 사용자 풀이 Amazon SNS 및 Amazon SES를 통해 전송하는 이메일 및 SMS 메시지의 상태를 확인하려는 경우에 유용합니다. 이 로그 내보내기 옵션은 사용자 활동 내보내기와 달리 Plus 기능 플랜이 필요하지 않습니다.

SetLogDeliveryConfiguration API 요청에서 Amazon Cognito 사용자 풀 API를 사용하여 세부 알림 로그를 구성할 수 있습니다. GetLogDeliveryConfiguration API 요청에서 사용자 풀의 로깅 구성을 볼 수 있습니다. 다음은 요청 본문의 예입니다.

```
{
   "LogConfigurations": [
      {
         "CloudWatchLogsConfiguration": {
            "LogGroupArn": "arn:aws:logs:us-west-2:123456789012:log-group:example-user-
pool-exported"
         },
         "EventSource": "userNotification",
         "LogLevel": "ERROR"
      }
   ],
   "UserPoolId": "us-west-2_EXAMPLE"
}
```

다음 권한이 있는 AWS 자격 증명으로 이러한 요청을 승인해야 합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
            {
            "Sid": "ManageUserPoolLogs",
            "Action": [
                "cognito-idp:SetLogDeliveryConfiguration",
                "cognito-idp:GetLogDeliveryConfiguration"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        },
        {
```

```
            "Sid": "CognitoLog",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        },
        {
            "Sid": "CognitoLoggingCWL",
            "Action": [
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

다음은 사용자 풀의 이벤트 예시입니다. 이 로그 스키마는 변경될 수 있습니다. 일부 필드는 null 값으로 로깅될 수 있습니다.

```
{
    "eventTimestamp": "1687297330677",
    "eventSource": "USER_NOTIFICATION",
    "logLevel": "ERROR",
    "message": {
        "details": "String"
    },
    "logSourceId": {
        "userPoolId": "String"
    }
}
```

위협 방지 사용자 활동 로그 내보내기

Plus 기능 플랜과 위협 방지 기능을 갖춘 사용자 풀은 사용자 활동 이벤트(사용자 로그인, 로그아웃 및 사용자 풀의 기타 인증 작업에 대한 세부 정보와 보안 평가)를 기록합니다. 자체 로그 관리 시스템에서 사용자 활동 로그를 검토하거나 아카이브를 생성할 수 있습니다. 이 데이터를 Amazon CloudWatch Logs 로그 그룹, Amazon Data Firehose 스트림 또는 Amazon Simple Storage Service(Amazon S3) 버킷으로 내보낼 수 있습니다. 여기에서 이 데이터를 운영 프로세스에 맞는 방식으로 데이터를 분석, 정규화 또는 처리하는 다른 시스템에 수집할 수 있습니다. 이 유형의 데이터를 내보내려면 사용자 풀이 Plus 기능 플랜에 있어야 하며 사용자 풀에서 위협 방지가 활성화되어 있어야 합니다.

이러한 사용자 활동 로그의 정보를 사용하여 사용자 로그인 및 계정 관리 활동의 프로파일을 볼 수 있습니다. 기본적으로 Amazon Cognito는 사용자 풀을 기반으로 하는 스토리지에 이러한 이벤트를 캡처합니다. 다음 예제는 로그인하고 위험 요소가 없는 것으로 평가된 사용자를 위한 예제 이벤트입니다. `AdminListUserAuthEvents` API 작업을 사용하여 이 정보를 검색할 수 있습니다. 다음은 출력의 예제입니다.


```
{
    "AuthEvents": [
        {
            "EventId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
            "EventType": "SignIn",
            "CreationDate": "2024-06-27T10:49:59.139000-07:00",
            "EventResponse": "Pass",
            "EventRisk": {
                "RiskDecision": "NoRisk",
                "CompromisedCredentialsDetected": false
            },
            "ChallengeResponses": [
                {
                    "ChallengeName": "Password",
                    "ChallengeResponse": "Success"
                }
            ],
            "EventContextData": {
                "IpAddress": "192.0.2.1",
                "DeviceName": "Chrome 126, Windows 10",
                "Timezone": "-07:00",
                "City": "null",
                "Country": "United States"
            }
        }
    ],
```

```
    "NextToken": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222#2024-06-27T17:49:59.139Z"
}
```

Amazon Cognito 콘솔 또는 SetLogDeliveryConfiguration API 작업에서 사용자 활동에 대한 로그 내보내기를 활성화할 수 있습니다.

AWS Management Console

1. 사용하려는 버킷이 아직 없는 경우 S3 버킷, Firehose 스트림 또는 CloudWatch 로그 그룹을 생성합니다.
2. Amazon Cognito 콘솔에 로그인합니다.
3. [사용자 풀(User Pools)]을 선택합니다.
4. 목록에서 기존 사용자 풀을 선택하거나 사용자 풀을 생성합니다.
5. 고급 보안 탭을 선택합니다. 사용자 활동 로그 내보내기를 찾아 편집을 선택합니다.
6. 로깅 상태에서 사용자 활동 로그 내보내기 활성화 옆의 확인란을 선택합니다.
7. 로깅 대상에서 로그를 처리할 CloudWatch 로그 그룹, Amazon Data Firehose 스트림 또는 S3 버킷 AWS 서비스 을 선택합니다.
8. 선택하면 리소스 선택기가 해당 리소스 유형으로 채워집니다. 목록에서 로그 그룹, 스트림 또는 버킷을 선택합니다. 생성 버튼을 선택하여 선택한 서비스의 AWS Management Console 로 이동하여 새 리소스를 생성할 수도 있습니다.
9. 변경 사항 저장을 선택합니다.

API

사용자 활동 로그의 대상 유형 하나를 선택합니다.

다음은 Firehose 스트림을 로그 대상으로 설정하는 `SetLogDeliveryConfiguration` 요청 본문의 예입니다.

```
{
   "LogConfigurations": [
      {
         "EventSource": "userAuthEvents",
         "FirehoseConfiguration": {
            "StreamArn": "arn:aws:firehose:us-west-2:123456789012:deliverystream/
example-user-pool-activity-exported"
         },
         "LogLevel": "INFO"
```

```
      }
   ],
   "UserPoolId": "us-west-2_EXAMPLE"
}
```

다음은 Amazon S3 버킷을 로그 대상으로 설정하는 SetLogDeliveryConfiguration 요청 본문의 예입니다.

```
{
   "LogConfigurations": [
      {
         "EventSource": "userAuthEvents",
         "S3Configuration": {
            "BucketArn": "arn:aws:s3:::amzn-s3-demo-logging-bucket"
         },
         "LogLevel": "INFO"
      }
   ],
   "UserPoolId": "us-west-2_EXAMPLE"
}
```

다음은 CloudWatch 로그 그룹을 로그 대상으로 설정하는 SetLogDeliveryConfiguration 요청 본문의 예입니다.

```
{
   "LogConfigurations": [
      {
         "EventSource": "userAuthEvents",
         "CloudWatchLogsConfiguration": {
            "LogGroupArn": "arn:aws:logs:us-west-2:123456789012:log-group:DOC-
EXAMPLE-LOG-GROUP"
         },
         "LogLevel": "INFO"
      }
   ],
   "UserPoolId": "us-west-2_EXAMPLE"
}
```

로그 전송을 구성하는 사용자는 사용자 풀 관리자여야 하며 다음과 같은 추가 권한이 있어야 합니다.

Amazon S3

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ManageUserPoolLogs",
            "Action": [
                "cognito-idp:SetLogDeliveryConfiguration",
                "cognito-idp:GetLogDeliveryConfiguration"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        },
        {
            "Sid": "ManageLogsS3",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "s3:PutBucketPolicy",
                "s3:GetBucketPolicy"
            ],
            "Resource": "*"
        }
    ]
}
```

CloudWatch Logs

JSON

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

```
                "cognito-idp:SetLogDeliveryConfiguration",
                "cognito-idp:GetLogDeliveryConfiguration"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        },
        {
            "Sid": "ManageLogsCWL",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries",
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Amazon Data Firehose

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ManageUserPoolLogs",
            "Action": [
                "cognito-idp:SetLogDeliveryConfiguration",
                "cognito-idp:GetLogDeliveryConfiguration"
            ],
            "Resource": [
```

```
                "*"
            ],
            "Effect": "Allow"
        },
        {
            "Sid": "ManageUserPoolLogsFirehose",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "iam:CreateServiceLinkedRole",
                "firehose:TagDeliveryStream"
            ],
            "Resource": "*"
        }
    ]
}
```

다음은 사용자 풀의 이벤트 예시입니다. 이 로그 스키마는 변경될 수 있습니다. 일부 필드는 null 값으로 로깅될 수 있습니다.

```
{
    "eventTimestamp": "1687297330677",
    "eventSource": "USER_ACTIVITY",
    "logLevel": "INFO",
    "message": {
        "version": "1",
        "eventId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "eventType": "SignUp",
        "userSub": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
        "userName": "test-user",
        "userPoolId": "us-west-2_EXAMPLE",
        "clientId": "1example23456789",
        "creationDate": "Wed Jul 17 17:25:55 UTC 2024",
        "eventResponse": "InProgress",
        "riskLevel": "",
        "riskDecision": "PASS",
        "challenges": [],
        "deviceName": "Other, Other",
        "ipAddress": "192.0.2.1",
        "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
        "idpName": "",
        "compromisedCredentialDetected": "false",
```

```
        "city": "Seattle",
        "country": "United States",
        "eventFeedbackValue": "",
        "eventFeedbackDate": "",
        "eventFeedbackProvider": "",
        "hasContextData": "true"
    },
    "logSourceId": {
        "userPoolId": "us-west-2_EXAMPLE"
    }
}
```

CloudWatch 및 Service Quotas에서 할당량 및 사용량 추적

Amazon CloudWatch를 사용하거나 Service Quotas를 사용하여 Amazon Cognito 사용자 풀을 모니터링할 수 있습니다. Service Quotas에서 자격 증명 풀 사용을 모니터링할 수도 있습니다. CloudWatch는 원시 데이터를 수집하여 읽기 쉽고 실시간에 가까운 지표로 처리합니다. CloudWatch에서 특정 임계값을 주시하다가 해당 임계값이 충족될 때 알림을 전송하거나 조치를 취하도록 경보를 설정할 수 있습니다. 서비스 할당량에 대한 CloudWatch 경보를 생성하려면 CloudWatch 경보 생성을 참조하세요. Amazon Cognito 지표는 5분 간격으로 제공됩니다. CloudWatch의 보존 기간에 대한 자세한 내용은 Amazon CloudWatch FAQ 페이지를 참조하세요.

Service Quotas를 사용하여 Amazon Cognito 사용자 풀 및 자격 증명 풀 할당량 사용량을 보고 관리할 수 있습니다. Service Quotas 콘솔에는 서비스 할당량 보기, 서비스 할당량 증가 요청, 현재 사용률 보기 등의 세 가지 기능이 있습니다. 첫 번째 기능을 사용하여 할당량을 보고 할당량 조정이 가능한지 확인할 수 있습니다. 두 번째 기능을 사용하여 Service Quotas 증가를 요청할 수 있습니다. 마지막 기능을 사용하여 할당량 사용률을 볼 수 있습니다. 이 기능은 계정이 일정 기간 동안 활성화되어 있었던 경우에만 사용할 수 있습니다. Service Quotas 콘솔에서 할당량을 확인하는 방법에 대한 자세한 내용은 Service Quotas 보기를 참조하세요.

> **Note**
>
> Amazon Cognito 지표는 5분 간격으로 제공됩니다. CloudWatch의 보존 기간에 대한 자세한 내용은 Amazon CloudWatch FAQ 페이지를 참조하세요.

CloudWatch 교차 계정 관찰성에서 모니터링 계정으로 설정된에 로그인 AWS 계정 한 경우 해당 모니터링 계정을 사용하여 서비스 할당량을 시각화하고 해당 모니터링 계정에 연결된 소스 계정의 지표에 대한 경보를 설정할 수 있습니다. 자세한 내용은 CloudWatch 크로스 계정 관찰성을 참조하세요.

주제

- CloudWatch의 사용자 풀 지표
- Service Quotas의 지표

CloudWatch의 사용자 풀 지표

사용자 풀은 사용자 활동 통계를 CloudWatch에 지표로 보고합니다. CloudWatch에서 사용자 풀의 인증 활동 및 할당량 사용량을 분석할 수 있습니다. 이러한 지표의 정보를 사용하여 주목할 만한 이벤트에 대한 경보를 설정하고 필요에 따라 사용자 풀 구성을 조정할 수 있습니다. 사용자 활동 로깅에 사용자 풀의 사용자 활동에 대한 세부 기록이 있는 경우 CloudWatch 지표에는 집계된 통계 및 성능 지표가 있습니다.

다음 표에는 Amazon Cognito 사용자 풀에 대해 제공되는 지표가 나열되어 있습니다. Amazon Cognito는 네임스페이스 `AWS/Cognito` 및`AWS/Usage`에 지표를 게시합니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 Namespaces를 참조하세요.

할당량 및 사용량 추적에 대한 자세한 내용은 할당량 사용량 추적 및 월별 활성 사용자(MAU) 추적 섹션을 참조하세요.

> **Note**
>
> 지난 2주 동안 새로운 데이터 포인트가 없는 지표는 콘솔에 나타나지 않습니다. 콘솔의 [모든 지표(All metrics)] 탭에서 검색 상자에 지표 이름이나 차원 이름을 입력할 때도 표시되지 않습니다. 또한 list-metrics 명령의 결과에서 반환되지 않습니다. 이러한 지표를 검색하는 가장 좋은 방법은 AWS CLI의 `get-metric-data` 또는 `get-metric-statistics` 명령을 사용하는 것입니다.

지표	설명	네임스페이스
`SignUpSuccesses`	Amazon Cognito 사용자 풀에 완료된 사용자 등록 요청의 총 개수를 제공합니다. 성공적인 사용자 등록 요청은 1의 값을 생성하지만 실패한 요청은 0의 값을 생성합니다. 제한된 요청	`AWS/Cognito`

지표	설명	네임스페이스
	도 실패한 요청으로 간주되므로 제한된 요청도 0이 됩니다. 성공한 사용자 등록 요청의 백분율을 찾으려면 이 메트릭에 대한 Average 통계를 사용합니다. 총 사용자 등록 요청 수를 계산하려면 이 메트릭에 대한 Sample Count 통계를 사용합니다. 성공한 사용자 등록 요청의 총 수를 계산하려면 이 메트릭에 대한 Sum 통계를 사용합니다. 실패한 사용자 등록 요청의 총 수를 계산하려면 CloudWatch Math 표현식을 사용하고 Sample Count 통계에서 Sum 통계를 뺍니다. 이 지표는 각 사용자 풀 클라이언트마다 사용자 풀별로 게시됩니다. 관리자가 사용자 등록을 수행하는 경우 지표는 사용자 풀 클라이언트가 Admin인 상태로 게시됩니다. 사용자 가져오기 및 사용자 마이그레이션의 경우 이 지표를 내보내지 않습니다. 메트릭 차원: UserPool, UserPoolClient 단위: 개	

지표	설명	네임스페이스
SignUpThrottles	Amazon Cognito 사용자 풀에 대한 제한된 사용자 등록 요청의 총 수를 제공합니다. 사용자 등록 요청이 제한될 때마다 1개가 게시됩니다. 제한된 사용자 등록 요청의 총 수를 계산하려면 이 메트릭에 대한 Sum 통계를 사용합니다. 이 지표는 각 클라이언트마다 사용자 풀별로 게시됩니다. 관리자가 제한된 요청을 하는 경우 지표는 사용자 풀 클라이언트가 Admin인 상태로 게시됩니다. 메트릭 차원: UserPool, UserPoolClient 단위: 개	AWS/Cognito

지표	설명	네임스페이스
SignInSuccesses	Amazon Cognito 사용자 풀에 성공한 사용자 인증 요청의 총 수를 제공합니다. 인증 토큰이 사용자에게 발급되면 사용자 인증이 성공한 것으로 간주됩니다. 성공한 인증은 1의 값을 생성하지만 실패한 요청은 0의 값을 생성합니다. 제한된 요청도 실패한 요청으로 간주되므로 제한된 요청도 0이 됩니다. 성공한 사용자 인증 요청의 백분율을 찾으려면 이 메트릭에 대한 Average 통계를 사용합니다. 총 사용자 인증 요청 수를 계산하려면 이 메트릭에 대한 Sample Count 통계를 사용합니다. 성공한 사용자 인증 요청의 총 수를 계산하려면 이 메트릭에 대한 Sum 통계를 사용합니다. 실패한 사용자 인증 요청의 총 수를 계산하려면 CloudWatch Math 표현식을 사용하고 Sample Count 통계에서 Sum 통계를 뺍니다. 이 지표는 각 클라이언트마다 사용자 풀별로 게시됩니다. 잘못된 사용자 풀 클라이언트에 요청이 제공되는 경우 지표의 해당 사용자 풀 클라이언트 값에는 요청에 전송된 실제 유효하지 않은 값 대신 고정 값 Invalid가 포함됩니다.	AWS/Cognito

지표	설명	네임스페이스
	Amazon Cognito 토큰 새로 고침 요청은 이 지표에 포함되지 않습니다. `Refresh` 토큰 통계를 제공하는 별도의 지표가 있습니다. 메트릭 차원: `UserPool`, `UserPoolClient` 단위: 개	

지표	설명	네임스페이스
SignInThrottles	Amazon Cognito 사용자 풀에서 이루어진 제한된 사용자 인증 요청의 총 수를 제공합니다. 인증 요청이 제한될 때마다 1개가 게시됩니다. 제한된 사용자 인증 요청의 총 수를 계산하려면 이 메트릭에 대한 Sum 통계를 사용합니다. 이 지표는 각 클라이언트마다 사용자 풀별로 게시됩니다. 잘못된 사용자 풀 클라이언트에 요청이 제공되는 경우 지표의 해당 사용자 풀 클라이언트 값에는 요청에 전송된 실제 유효하지 않은 값 대신 고정 값 Invalid가 포함됩니다. Amazon Cognito 토큰 새로 고침 요청은 이 지표에 포함되지 않습니다. Refresh 토큰 통계를 제공하는 별도의 지표가 있습니다. 메트릭 차원: UserPool, UserPoolClient 단위: 개	AWS/Cognito

지표	설명	네임스페이스
TokenRefreshSuccesses	Amazon Cognito 사용자 풀에 대해 만들어진 Amazon Cognito 토큰을 새로 고치려면 성공한 총 요청 수를 제공합니다. 새로 고침에 성공한 Amazon Cognito 토큰 요청은 값 1을 생성하지만 실패한 요청은 값 0을 생성합니다. 제한된 요청도 실패한 요청으로 간주되므로 제한된 요청도 0이 됩니다. Amazon Cognito 토큰을 새로 고치기 위해 성공한 요청의 백분율을 구하려면 이 지표의 Average 통계를 사용합니다. Amazon Cognito 토큰을 새로 고치기 위한 총 요청 수를 계산하려면 이 지표의 Sample Count 통계를 사용합니다. Amazon Cognito 토큰을 새로 고치기 위해 성공한 총 요청 수를 계산하려면 이 지표의 Sum 통계를 사용합니다. Amazon Cognito 토큰을 새로 고치기 위해 실패한 총 요청 수를 계산하려면 CloudWatch Math 표현식을 사용하고 Sample Count 통계에서 Sum 통계를 뺍니다. 이 지표는 각 사용자 풀 클라이언트별로 게시됩니다. 요청에 잘못된 사용자 풀 클라이언트가 있는 경우 사용자 풀 클라이	AWS/Cognito

지표	설명	네임스페이스
	언트 값에 Invalid라는 고정 값이 포함됩니다. 메트릭 차원: UserPool, UserPoolClient 단위: 개	
TokenRefreshThrottles	Amazon Cognito 사용자 풀에 대해 만들어진 Amazon Cognito 토큰을 새로 고치려면 제한된 총 요청 수를 제공합니다. 새로 고침 Amazon Cognito 토큰 요청이 제한될 때마다 1개가 게시됩니다. Amazon Cognito 토큰을 새로 고치기 위한 제한된 총 요청 수를 계산하려면 이 지표의 Sum 통계를 사용합니다. 이 지표는 각 클라이언트마다 사용자 풀별로 게시됩니다. 잘못된 사용자 풀 클라이언트에 요청이 제공되는 경우 지표의 해당 사용자 풀 클라이언트 값에는 요청에 전송된 실제 유효하지 않은 값 대신 고정 값 Invalid가 포함됩니다. 메트릭 차원: UserPool, UserPoolClient 단위: 개	AWS/Cognito

지표	설명	네임스페이스
FederationSuccesses	Amazon Cognito 사용자 풀에 성공한 ID 페더레이션 요청의 총 수를 제공합니다. Amazon Cognito가 사용자에게 인증 토큰을 발급하면 아이덴티티 페더레이션이 성공한 것으로 간주됩니다. 자격 증명 연동 요청이 성공하면 값 1이 생성되고 요청이 실패하면 값이 0입니다. 인증 코드는 생성하지만 토큰은 생성하지 않는 요청과 제한된 요청은 값이 0입니다. 성공한 자격 증명 연동 요청의 비율을 찾으려면 이 메트릭의 Average 통계를 사용합니다. 총 자격 증명 연동 요청 수를 계산하려면 이 메트릭의 Sample Count 통계를 사용합니다. 성공한 자격 증명 연동 요청의 총 수를 계산하려면 이 메트릭의 Sum 통계를 사용합니다. 실패한 ID 페더레이션 요청의 총 수를 계산하려면 CloudWatch Math 표현식을 사용하고 Sample Count 통계에서 Sum 통계를 뺍니다. 메트릭 차원: UserPool, UserPoolClient , IdentityProvider 단위: 개	AWS/Cognito

지표	설명	네임스페이스
FederationThrottles	Amazon Cognito 사용자 풀에 대한 제한된 ID 페더레이션 요청의 총 수를 제공합니다. ID 페더레이션 요청이 제한될 때마다 1개가 게시됩니다. 제한된 자격 증명 연동 요청의 총 수를 계산하려면 이 메트릭에 대한 Sum 통계를 사용합니다. 메트릭 차원: UserPool, UserPoolClient , IdentityProvider 단위: 개	AWS/Cognito
CallCount	범주와 관련된 고객의 총 호출 수를 제공합니다. 이 지표에는 제한된 호출, 실패한 호출, 성공한 호출 등 모든 호출이 포함됩니다. 범주 할당량은 AWS 계정 및 리전의 모든 사용자 풀에 걸쳐 각 계정에 적용됩니다. 이 지표의 Sum 통계를 사용하여 범주의 총 호출 수를 계산할 수 있습니다. 지표 차원: 서비스, 유형, 리소스, 클래스 단위: 개	AWS/Usage

지표	설명	네임스페이스
ThrottleCount	범주와 관련된 총 제한된 호출 수를 제공합니다. 이 지표는 계정 수준에서 게시됩니다. 이 지표의 Sum 통계를 사용하여 범주의 총 호출 수를 계산할 수 있습니다. 지표 차원: 서비스, 유형, 리소스, 클래스 단위: 개	AWS/Usage

위협 방지 지표 보기

사용자 풀이 게시하는 지표에는 위협 방지 설정이 사용자 인증 활동에 미치는 영향에 대한 통계 정보가 있습니다. 손상된 자격 증명으로 로그인을 시도하는 사용자가 몇 명인지 알고 싶을 수 있습니다. 또한 어느 정도의 위험이 있는 것으로 평가된 로그인 활동의 비율을 확인할 수 있습니다. Amazon Cognito는 Amazon CloudWatch의 계정에 위협 방지 기능의 측정치를 게시합니다. Amazon Cognito는 위협 방지 지표를 위험 수준별 및 요청 수준별로 함께 그룹화합니다.

위험 분석에 컨텍스트를 추가하려면 사용자 풀 또는 내보낸 데이터 소스에서 개별 사용자 로그인 시도에 대한 정보를 볼 수 있습니다.

CloudWatch 콘솔에서 지표 보기

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표를 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 제목 행에서 확인란을 선택합니다.
- 리소스로 필터링하려면 리소스 ID를 선택한 후 검색에 추가를 선택합니다.
- 지표로 필터링하려면 지표 이름을 선택한 후 검색에 추가를 선택합니다.

지표	설명	측정치 차원	네임스페이스
CompromisedCredentialRisk	Amazon Cognito가 손상된 자격 증명을 탐지한 요청.	작업: 작업 유형 `PasswordChange` , `SignIn` 또는 `SignUp`이 유일한 차원입니다. UserPoolId: 사용자 풀의 식별자 RiskLevel: 높음(기본), 중간 또는 낮음	`AWS/Cognito`
AccountTakeoverRisk	Amazon Cognito가 계정 탈취 위험을 탐지한 요청.	작업: 작업 유형 `PasswordChange` , `SignIn` 또는 `SignUp`이 유일한 차원입니다. UserPoolId: 사용자 풀의 식별자 RiskLevel: 높음, 중간 또는 낮음	`AWS/Cognito`
OverrideBlock	Amazon Cognito가 개발자가 제공한 구성의 결과로 차단한 요청.	작업: 작업 유형 `PasswordChange` , `SignIn` 또는 `SignUp`이 유일한 차원입니다.	`AWS/Cognito`

지표	설명	측정치 차원	네임스페이스
		UserPoolId: 사용자 풀의 식별자 RiskLevel: 높음, 중간 또는 낮음	
Risk	Amazon Cognito가 위험한 것으로 표시한 요청.	작업: `PasswordChange` , `SignIn` 또는 `SignUp` 등의 작업 유형 UserPoolId: 사용자 풀의 식별자	`AWS/Cognito`
NoRisk	Amazon Cognito가 어떤 위험도 식별하지 않은 요청.	작업: `PasswordChange` , `SignIn` 또는 `SignUp` 등의 작업 유형 UserPoolId: 사용자 풀의 식별자	`AWS/Cognito`

Amazon Cognito는 CloudWatch에서 분석할 수 있도록 사전 정의된 두 개의 측정치 그룹을 제공합니다. 위험 분류별(By Risk Classification)은 Amazon Cognito가 위험하다고 식별한 요청에 대한 위험의 세부 수준을 식별합니다. 요청 분류별(By Request Classification)은 요청 수준별로 집계된 지표를 반영합니다.

집계된 측정치 그룹	설명
위험 분류별	Amazon Cognito가 위험한 것으로 식별한 요청입니다.
요청 분류별	요청별로 집계된 측정치

Amazon Cognito 사용자 풀에 대한 차원

다음 차원은 Amazon Cognito에 의해 게시되는 사용량 지표를 구체화하는 데 사용됩니다. 차원은 `CallCount` 및 `ThrottleCount` 지표에만 적용됩니다.

차원	설명
Service	리소스가 포함된 AWS 서비스의 이름입니다. Amazon Cognito 사용량 지표의 경우 이 차원 값은 `Cognito user pool`입니다.
유형	보고되는 엔터티의 유형입니다. Amazon Cognito 사용량 지표에 대한 유일한 유효 값은 API입니다.
Resource	실행 중인 리소스의 유형입니다. 유일한 유효 값은 범주 이름입니다.
Class	추적 중인 리소스의 클래스입니다. Amazon Cognito는 클래스 차원을 사용하지 않습니다.

CloudWatch 콘솔을 사용하여 지표 추적

CloudWatch를 사용하여 Amazon Cognito 사용자 풀 지표를 추적하고 수집할 수 있습니다. CloudWatch 대시보드에는 사용하는 모든 AWS 서비스에 대한 지표가 표시됩니다. CloudWatch를 사용하여 지표 경보를 생성할 수 있습니다. 알림을 보내거나 모니터링 중인 특정 리소스를 변경하도록 경보를 설정할 수 있습니다. CloudWatch에서 서비스 할당량 지표를 보려면 다음 단계를 수행합니다.

1. CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표를 선택합니다.
3. [모든 지표(All metrics)]에서 지표와 차원을 선택합니다.
4. 지표 옆의 확인란을 선택합니다. 지표는 그래프에 표시됩니다.

> **Note**
>
> 지난 2주 동안 새로운 데이터 포인트가 없는 지표는 콘솔에 나타나지 않습니다. 콘솔의 모든 지표 탭에 있는 검색 상자에 지표 이름이나 차원 이름을 입력할 때도 나타나지 않으며 list-metrics

명령의 결과에도 반환되지 않습니다. 이러한 지표를 검색하는 가장 좋은 방법은 AWS CLI에서 `get-metric-data` 또는 `get-metric-statistics` 명령을 사용하는 것입니다.

할당량에 대한 CloudWatch 경보 생성

Amazon Cognito는 `CallCount` 및 `ThrottleCount` API에 대한 AWS 서비스 할당량에 해당하는 CloudWatch 사용량 지표를 제공합니다. APIs CloudWatch에서 사용량을 추적하는 데 대한 자세한 내용은 할당량 사용량 추적 섹션을 참조하세요.

Service Quotas 콘솔에서 사용량이 서비스 할당량에 가까워지면 알리는 경보를 생성할 수 있습니다. Service Quotas 콘솔을 사용하여 CloudWatch 경보를 설정하는 방법에 대해 자세히 알아보려면 Service Quotas 및 CloudWatch 경보를 참조하세요.

Service Quotas의 지표

Service Quotas로 중앙 위치에서 Amazon Cognito 사용자 풀 및 자격 증명 풀 할당량을 보고 관리할 수 있습니다. Service Quotas 콘솔을 사용하여 특정 할당량에 대한 세부 정보를 확인하고, 할당량 사용률을 모니터링하고, 할당량 증가를 요청할 수 있습니다. 일부 할당량 유형의 경우 CloudWatch 경보를 생성하여 할당량 사용률을 추적할 수 있습니다. 추적할 수 있는 Amazon Cognito 지표에 대한 자세한 내용은 할당량 사용량 추적을 참조하세요.

Amazon Cognito 사용자 풀 및 자격 증명 풀 서비스 할당량 사용률을 보려면 다음 단계를 수행합니다.

1. Service Quotas 콘솔을 엽니다.
2. 탐색 창에서 AWS 서비스를 선택합니다.
3. AWS 서비스 목록에서 Amazon Cognito 사용자 풀 또는 Amazon Cognito 페더레이션 ID를 검색하고 선택합니다. 서비스 할당량 페이지가 표시됩니다.
4. CloudWatch 모니터링을 지원하는 할당량을 선택합니다. 예를 들어, Amazon Cognito 사용자 풀에서 `Rate of UserAuthentication requests`를 선택합니다.
5. 아래로 스크롤하여 [모니터링(Monitoring)]으로 이동합니다. 이 섹션은 CloudWatch 모니터링을 지원하는 할당량에 대해서만 표시됩니다.
6. [모니터링(Monitoring)에서 현재 서비스 할당량 사용률을 그래프로 볼 수 있습니다.
7. [모니터링(Monitoring)]에서 1시간, 3시간, 12시간, 1일, 3일 또는 1주를 선택합니다.
8. 선택하여 서비스 할당량 사용률을 보려면 그래프 내의 아무 영역이나 선택합니다. 여기에서 대시보드에 그래프를 추가하거나 작업 메뉴를 사용하여 지표에서 보기를 선택할 수 있습니다. 그러면 CloudWatch 콘솔의 관련 지표로 이동합니다.

Amazon Cognito 로그인 AWS CloudTrail

Amazon Cognito는 Amazon Cognito에서 사용자 AWS CloudTrail, 역할 또는 서비스가 수행한 작업에 대한 레코드를 제공하는 AWS 서비스와 통합됩니다. CloudTrail은 Amazon Cognito 콘솔의 호출 및 Amazon Cognito API 작업에 대한 코드 호출을 포함하여 Amazon Cognito에 대한 API 호출의 하위 집합을 이벤트로 캡처합니다. 추적을 생성하면 Amazon Cognito에 대한 이벤트를 포함하는 CloudTrail 이벤트를 Amazon S3 버킷에 전송하도록 선택할 수 있습니다. 추적을 구성하지 않은 경우에도 이벤트 기록에서 CloudTrail 콘솔의 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 Amazon Cognito에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

구성 및 활성화 방법을 비롯한 CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

특정 CloudTrail 이벤트에 대해 Amazon CloudWatch 경보를 만들 수도 있습니다. 예를 들어 자격 증명 풀 구성이 변경되면 경보를 트리거하도록 CloudWatch를 설정할 수 있습니다. 자세한 내용은 CloudTrail 이벤트에 대한 CloudWatch 경보 생성: 예제를 참조하세요.

주제

- Amazon Cognito가 CloudTrail로 전송하는 정보
- Amazon CloudWatch Logs Insights를 사용하여 Amazon Cognito CloudTrail 이벤트 분석
- Amazon Cognito 이벤트 예시

Amazon Cognito가 CloudTrail로 전송하는 정보

를 생성할 때 CloudTrail이 켜집니다 AWS 계정. Amazon Cognito에서 지원되는 이벤트 활동이 발생하면 해당 활동은 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. AWS 계정에서 최근 이벤트를 보고 검색하고 다운로드할 수 있습니다. 자세한 내용은 CloudTrail 이벤트 기록을 사용하여 이벤트 보기를 참조하세요.

Amazon Cognito에 대한 이벤트를 포함하여 AWS 계정에 이벤트를 지속적으로 기록하려면 추적을 생성합니다. CloudTrail 추적은 Amazon S3 버킷으로 로그 파일을 전송합니다. 콘솔에서 추적을 생성하면 기본적으로 모든 Region에 추적이 적용됩니다. 추적은 AWS 파티션의 모든 리전에서 이벤트를 로깅하고 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 또한 CloudTrail 로그에서 수집된 이벤트 데이터를 추가로 분석하고 조치를 취하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음을 참조하세요.

- 추적 생성 개요

- CloudTrail 지원 서비스 및 통합
- CloudTrail에 대한 Amazon SNS 알림 구성
- 여러 리전에서 CloudTrail 로그 파일 받기 및 여러 계정에서 CloudTrail 로그 파일 받기

모든 이벤트 또는 로그 항목에는 요청을 생성했던 사용자에 관한 정보가 포함됩니다. ID 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 아니면 IAM 사용자 자격 증명 정보로 했는지 여부.
- 역할 또는 페더레이션 사용자의 임시 자격 증명을 사용하여 요청이 생성되었는지 여부.
- 요청이 다른 AWS 서비스에서 이루어졌는지 여부입니다.

자세한 내용은 CloudTrail userIdentity 요소를 참조하세요.

의 기밀 데이터 AWS CloudTrail

사용자 풀과 자격 증명 풀은 사용자 데이터를 처리하기 때문에 Amazon Cognito는 `HIDDEN_DUE_TO_SECURITY_REASONS` 값을 사용하여 CloudTrail 이벤트의 일부 비공개 필드를 가립니다. Amazon Cognito가 이벤트에 채우지 않는 필드의 예는 Amazon Cognito 이벤트 예시 섹션을 참조하세요. Amazon Cognito는 암호 및 토큰과 같이 일반적으로 사용자 정보를 포함하는 일부 필드만 가립니다. Amazon Cognito는 API 요청의 비공개 필드에 채우는 개인 식별 정보의 자동 감지 또는 마스킹을 수행하지 않습니다.

사용자 풀 이벤트

Amazon Cognito는 사용자 풀 작업(User pool actions) 페이지에 나열된 모든 작업을 CloudTrail 로그 파일에 이벤트로 로그하도록 지원합니다. Amazon Cognito는 사용자 풀 이벤트를 관리 이벤트로 CloudTrail에 로깅합니다.

Amazon Cognito 사용자 풀 CloudTrail 항목의 `eventType` 필드는 앱이 Amazon Cognito 사용자 풀 API에 요청을 했는지 아니면 OpenID Connect, SAML 2.0 또는 관리형 로그인 페이지용 리소스를 제공하는 엔드포인트에 요청을 했는지 알려줍니다. API 요청의 `eventType`은 `AwsApiCall`이고 엔드포인트 요청의 `eventType`은 `AwsServiceEvent`입니다.

Amazon Cognito는 관리형 로그인 서비스에 다음 요청을 CloudTrail의 이벤트로 기록합니다.

Hosted UI (classic) events

CloudTrail의 호스팅 UI(클래식) 이벤트

연산	설명
`Login_GET` , `CognitoAuthentication`	사용자가 보안 인증을 보거나 Login 엔드포인트에 제출합니다.
`OAuth2_Authorize_GET` , `Beta_Authorize_GET`	사용자가 권한 부여 엔드포인트를 봅니다.
`OAuth2Response_GET` , `OAuth2Response_POST`	사용자가 `/oauth2/idpresponse` 엔드포인트에 IdP 토큰을 제출합니다.
`SAML2Response_POST` , `Beta_SAML2Response_POST`	사용자가 `/saml2/idpresponse` 엔드포인트에 IdP SAML 어설션을 제출합니다.
`Login_OIDC_SAML_POST`	사용자가 Login 엔드포인트에 사용자 이름을 입력하고 IdP 식별자와 일치합니다.
`Token_POST` , `Beta_Token_POST`	사용자가 Token 엔드포인트에 인증 코드를 제출합니다.
`Signup_GET` , `Signup_POST`	사용자가 `/signup` 엔드포인트에 가입 정보를 제출합니다.
`Confirm_GET` , `Confirm_POST`	사용자가 호스팅 UI에서 확인 코드를 제출합니다.
`ResendCode_POST`	사용자가 호스팅 UI에서 확인 코드를 다시 보내라는 요청을 제출합니다.
`ForgotPassword_GET` , `ForgotPassword_POST`	사용자가 `/forgotPassword` 엔드포인트에 암호를 재설정하라는 요청을 제출합니다.
`ConfirmForgotPassword_GET` , `ConfirmForgotPassword_POST`	사용자가 `/confirmForgotPassword` 엔드포인트에 `ForgotPassword` 요청을 확인하는 코드를 제출합니다.

연산	설명
ResetPassword_GET , ResetPassword_POST	사용자가 호스팅 UI에서 새 암호를 제출합니다.
Mfa_GET, Mfa_POST	사용자가 호스팅 UI에서 다중 인증(MFA) 코드를 제출합니다.
MfaOption_GET , MfaOption_POST	사용자가 호스팅 UI에서 MFA에 대해 선호하는 방법을 선택합니다.
MfaRegister_GET , MfaRegister_POST	MFA 등록 시 사용자가 호스팅된 UI에서 다중 인증(MFA) 코드를 제출합니다.
Logout	사용자가 /logout 엔드포인트에서 로그아웃합니다.
SAML2Logout_POST	사용자가 /saml2/logout 엔드포인트에서 로그아웃합니다.
Error_GET	사용자가 호스팅 UI에서 오류 페이지를 봅니다.
UserInfo_GET , UserInfo_POST	사용자 또는 IdP가 userInfo 엔드포인트와 정보를 교환합니다.
Confirm_With_Link_GET	사용자가 Amazon Cognito가 이메일 메시지로 보낸 링크를 기반으로 확인을 제출합니다.
Event_Feedback_GET	사용자가 Amazon Cognito에 위협 방지 이벤트에 대한 피드백을 제출합니다.

Managed login events

CloudTrail의 관리형 로그인 이벤트

연산	설명
`login_POST`	사용자가 자격 증명을 Login 엔드포인트에 제출합니다.
`login_continue_POST`	이미 한 번 로그인한 사용자는 다시 로그인하기로 선택합니다.
`forgotPassword_POST`	사용자가 암호를 재설정합니다.
`selectChallenge_POST`	사용자는 사용자 이름 또는 자격 증명을 제출한 후 인증 문제에 응답합니다.
`confirmUser_GET`	사용자가 확인 또는 검증 이메일 메시지에서 링크를 엽니다.
`mfa_back_POST`	사용자는 MFA 프롬프트 다음에 뒤로 버튼을 선택합니다.
`mfa_options_POST`	사용자가 MFA 옵션을 선택합니다.
`mfa_phone_register_POST`	사용자가 전화번호를 제출하여 MFA 팩터로 등록합니다. 이 작업을 수행하면 Amazon Cognito가 전화번호로 MFA 코드를 전송합니다.
`mfa_phone_verify_POST`	사용자가 전화번호로 전송된 MFA 코드를 제출합니다.
`mfa_phone_resendCode_POST`	사용자가 MFA 코드를 전화번호로 재전송하라는 요청을 제출합니다.
`mfa_totp_POST`	사용자가 TOTP MFA 코드를 제출합니다.
`signup_POST`	사용자가 `/signup` 관리형 로그인 페이지에 정보를 제출합니다.

연산	설명
signup_confirm_POST	사용자가 이메일 또는 SMS 메시지에서 확인 코드를 제출합니다.
verifyCode_POST	사용자가 암호 없는 인증을 위해 일회용 암호(OTP)를 제출합니다.
passkeys_add_POST	사용자가 새 패스키 자격 증명 등록 요청을 제출합니다.
passkeys_add_GET	사용자는 패스키를 등록할 수 있는 페이지로 이동합니다.
login_passkey_POST	사용자가 패스키로 로그인합니다.

> **Note**
>
> Amazon Cognito는 사용자별 요청에 대한 CloudTrail 로그에 UserSub는 기록하지만 UserName은 기록하지 않습니다. ListUsers API를 호출하고 하위 집합을 표시하는 필터를 사용하여 특정 UserSub의 사용자를 찾을 수 있습니다.

ID 풀 이벤트

데이터 이벤트

Amazon Cognito는 다음과 같은 Amazon Cognito 자격 증명 이벤트를 데이터 이벤트로 로깅합니다. 데이터 이벤트는 CloudTrail이 기본적으로 로깅하지 않는 대용량 데이터 영역 API 작업입니다. 데이터 이벤트에는 추가 요금이 적용됩니다.

- GetCredentialsForIdentity
- GetId
- GetOpenIdToken
- GetOpenIdTokenForDeveloperIdentity
- UnlinkIdentity

이러한 API 작업에 대해 CloudTrail 로그를 생성하려면 추적에서 데이터 이벤트를 활성화하고 Cognito identity pools(Cognito 자격 증명 풀)에 대한 이벤트 선택기를 선택해야 합니다. 자세한 내용은 AWS CloudTrail 사용 설명서의 추적을 위해 데이터 이벤트 로깅을 참조하십시오.

다음 CLI 명령을 사용해서도 추적에 자격 증명 풀 이벤트 선택기를 추가할 수 있습니다.

```
aws cloudtrail put-event-selectors --trail-name <trail name> --advanced-event-selectors
 \
"{\
   \"Name\": \"Cognito Selector\",\
   \"FieldSelectors\": [\
      {\
         \"Field\": \"eventCategory\",\
         \"Equals\": [\
            \"Data\"\
         ]\
      },\
      {\
         \"Field\": \"resources.type\",\
         \"Equals\": [\
            \"AWS::Cognito::IdentityPool\"\
         ]\
      }\
   ]\
}"
```

관리 이벤트

Amazon Cognito는 나머지 Amazon Cognito 자격 증명 풀 API 작업을 관리 이벤트로 로깅합니다. CloudTrail은 기본적으로 관리 이벤트 API 작업을 로깅합니다.

Amazon Cognito가 CloudTrail에 로깅하는 Amazon Cognito 자격 증명 풀 API 작업 목록은 Amazon Cognito 자격 증명 풀 API 참조에서 확인하세요.

Amazon Cognito Sync

Amazon Cognito는 모든 Amazon Cognito Sync API 작업을 관리 이벤트로 로깅합니다. Amazon Cognito가 CloudTrail에 로깅하는 Amazon Cognito Sync API 작업 목록은 Amazon Cognito Sync API Reference(Amazon Cognito Sync API 참조)를 참조하세요.

Amazon CloudWatch Logs Insights를 사용하여 Amazon Cognito CloudTrail 이벤트 분석

Amazon CloudWatch Logs Insights를 사용하여 Amazon Cognito CloudTrail 이벤트를 검색 및 분석할 수 있습니다. CloudWatch Logs에 이벤트를 전송하도록 추적을 구성하면 CloudTrail은 추적 설정과 일치하는 이벤트만 전송합니다.

Amazon Cognito CloudTrail 이벤트를 쿼리하거나 조사하려면 CloudTrail 콘솔에서 추적 설정에서 관리 이벤트 옵션을 선택해야 AWS 리소스에 대해 수행된 관리 작업을 모니터링할 수 있습니다. 오류, 비정상적인 활동 또는 비정상적인 사용자 행동을 식별하려는 경우 추적 설정에서 Insights 이벤트 옵션을 선택할 수도 있습니다.

Amazon Cognito 쿼리 샘플

Amazon CloudWatch 콘솔에서 다음 쿼리를 사용할 수 있습니다.

일반 쿼리

최근에 추가된 로그 이벤트 25개를 찾습니다.

```
fields @timestamp, @message | sort @timestamp desc | limit 25
| filter eventSource = "cognito-idp.amazonaws.com"
```

예외를 포함하여 최근에 추가된 로그 이벤트 25개의 목록을 표시합니다.

```
fields @timestamp, @message | sort @timestamp desc | limit 25
| filter eventSource = "cognito-idp.amazonaws.com" and @message like /Exception/
```

예외 및 오류 쿼리

Amazon Cognito 사용자 풀 sub를 사용하여 오류 코드 NotAuthorizedException과 함께 최근에 추가된 로그 이벤트 25개를 찾습니다.

```
fields @timestamp, additionalEventData.sub as user | sort @timestamp desc | limit 25
| filter eventSource = "cognito-idp.amazonaws.com" and errorCode=
 "NotAuthorizedException"
```

sourceIPAddress 및 해당 eventName이 있는 레코드의 수를 찾습니다.

```
filter eventSource = "cognito-idp.amazonaws.com"
```

```
| stats count(*) by sourceIPAddress, eventName
```

NotAuthorizedException 오류를 트리거한 상위 25개 IP 주소를 찾습니다.

```
filter eventSource = "cognito-idp.amazonaws.com" and errorCode=
 "NotAuthorizedException"
| stats count(*) as count by sourceIPAddress, eventName
| sort count desc | limit 25
```

ForgotPassword API를 호출한 상위 25개 IP 주소를 찾습니다.

```
filter eventSource = "cognito-idp.amazonaws.com" and eventName = 'ForgotPassword'
| stats count(*) as count by sourceIPAddress
| sort count desc | limit 25
```

Amazon Cognito 이벤트 예시

Amazon Cognito는 사용자 인증 활동 및 관리 활동에 AWS CloudTrail 대한 정보를에 기록합니다. 이는 사용자 풀과 ID 풀 모두에 적용됩니다. 예를 들어, 동일한 추적 또는 UpdateAuthEventFeedback 및 SetRiskConfiguration 이벤트에서 GetId 및 UpdateIdentityPool 이벤트를 볼 수 있습니다. 또한 사용자 풀 API의 작업에 해당하지 않는 호스팅 UI 활동에 대한 사용자 풀 로그가 표시됩니다. 이 섹션에는 사용자가 볼 수 있는 로그의 몇 가지 예가 나와 있습니다. 모든 작업에 대한 CloudTrail 이벤트 스키마를 이해하려면 해당 작업에 대한 요청을 생성하고 추적에서 생성한 이벤트를 검토합니다.

추적은 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 전송할 수 있습니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스의 단일 요청을 나타내며 요청된 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보를 포함하고 있습니다. CloudTrail 로그 파일은 퍼블릭 API 직접 호출에 대한 순서 지정된 스택 추적이 아니기 때문에 특정 순서로 표시되지 않습니다.

주제

- 호스팅 UI 가입에 대한 CloudTrail 이벤트 예제
- SAML 요청에 대한 CloudTrail 이벤트 예제
- 토큰 엔드포인트로의 요청에 대한 CloudTrail 이벤트 예제
- CreateIdentityPool에 대한 CloudTrail 이벤트 예제
- GetCredentialsForIdentity에 대한 CloudTrail 이벤트 예시
- GetId에 대한 CloudTrail 이벤트 예시
- GetOpenIdToken에 대한 CloudTrail 이벤트 예시

- GetOpenIdTokenForDeveloperIdentity에 대한 CloudTrail 이벤트 예시
- UnlinkIdentity에 대한 CloudTrail 이벤트 예시

호스팅 UI 가입에 대한 CloudTrail 이벤트 예제

다음 CloudTrail 이벤트 예제는 사용자가 호스팅 UI를 통해 가입할 때 Amazon Cognito가 로그하는 정보를 보여 줍니다.

Amazon Cognito는 신규 사용자가 앱의 로그인 페이지로 이동할 때 다음 이벤트를 로그합니다.


```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-04-06T05:38:12Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Login_GET",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "errorCode": "",
    "errorMessage": "",
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200.0
        },
        "requestParameters":
        {
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "response_type":
            [
                "token"
            ],
            "client_id":
            [
```

```
                "1example23456789"
            ]
        }
    },
    "eventID": "382ae09a-151d-4116-8f2b-6ac0a804a38c",
    "readOnly": true,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

Amazon Cognito는 신규 사용자가 앱의 로그인 페이지에서 가입을 선택할 때 다음 이벤트를 로그합니다.

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-05T23:21:43Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Signup_GET",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
            "response_type":
            [
```

```
                "code"
            ],
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "client_id":
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.auth.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_EXAMPLE"
    },
    "requestID": "7a63e7c2-b057-4f3d-a171-9d9113264fff",
    "eventID": "5e7b27a0-6870-4226-adb4-f86cd51ac5d8",
    "readOnly": true,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

Amazon Cognito는 신규 사용자가 사용자 이름을 선택하고 이메일 주소를 입력한 다음 앱의 로그인 페이지에서 암호를 선택할 때 다음 이벤트를 로깅합니다. Amazon Cognito는 사용자의 자격 증명에 대한 식별 정보를 CloudTrail에 로그하지 않습니다.

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-05T23:22:05Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Signup_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
```

```
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 302
        },
        "requestParameters":
        {
            "password":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "requiredAttributes[email]":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "response_type":
            [
                "code"
            ],
            "_csrf":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "client_id":
            [
                "1example23456789"
            ],
            "username":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ]
        },
        "userPoolDomain": "mydomain.auth.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_EXAMPLE"
    },
    "requestID": "9ad58dd8-3517-4aa8-96a5-d17a01df9eb4",
    "eventID": "c75eb7a5-eb8c-43d1-8331-f4412e756e69",
```

```
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

Amazon Cognito는 신규 사용자가 가입한 후 호스팅 UI의 사용자 확인 페이지에 액세스할 때 다음 이벤트를 로그합니다.

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-05T23:22:06Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Confirm_GET",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
            "response_type":
            [
                "code"
            ],
            "redirect_uri":
            [
                "https://www.amazon.com"
```

```
            ],
            "client_id":
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.auth.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_EXAMPLE"
    },
    "requestID": "58a5b170-3127-45bb-88cc-3e652d779e0b",
    "eventID": "7f87291a-6d50-409a-822f-e3a5ec7e60da",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

Amazon Cognito는 호스팅 UI의 사용자 확인 페이지에서 사용자가 Amazon Cognito가 이메일 메시지로 보낸 코드를 입력할 때 다음 이벤트를 로그합니다.

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-05T23:23:32Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Confirm_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
```

```
            "status": 302
        },
        "requestParameters":
        {
            "confirm":
            [
                ""
            ],
            "deliveryMedium":
            [
                "EMAIL"
            ],
            "sub":
            [
                "704b1e47-34fe-40e9-8c41-504997494531"
            ],
            "code":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "destination":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "response_type":
            [
                "code"
            ],
            "_csrf":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "cognitoAsfData":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "client_id":
            [
                "1example23456789"
            ],
```

```
            "username":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ]
        },
        "userPoolDomain": "mydomain.auth.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_EXAMPLE"
    },
    "requestID": "9764300a-ed35-4f87-8a0f-b18b3fe2b11e",
    "eventID": "e24ac6e5-2f70-4c6e-ad4e-2f08a547bb36",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

SAML 요청에 대한 CloudTrail 이벤트 예제

Amazon Cognito는 SAML IdP로 인증된 사용자가 `/saml2/idpresponse` 엔드포인트에 SAML 어설션을 제출할 때 다음 이벤트를 로그합니다.

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-06T00:50:57Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "SAML2Response_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
```

```
        {
            "status": 302
        },
        "requestParameters":
        {
            "RelayState":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "SAMLResponse":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ]
        },
        "userPoolDomain": "mydomain.auth.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_EXAMPLE"
    },
    "requestID": "4f6f15d1-c370-4a57-87f0-aac4817803f7",
    "eventID": "9824b50f-d9d1-4fb8-a2c1-6aa78ca5902a",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "625647942648",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

토큰 엔드포인트로의 요청에 대한 CloudTrail 이벤트 예제

다음은 Token 엔드포인트에 대한 요청의 이벤트 예제입니다.

Amazon Cognito는 인증되고 인증 코드를 받은 사용자가 `/oauth2/token` 엔드포인트에 코드를 제출할 때 다음 이벤트를 로그합니다.

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
```

```
    "eventTime": "2022-05-12T22:12:30Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Token_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
            "code":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "grant_type":
            [
                "authorization_code"
            ],
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "client_id":
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.auth.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_EXAMPLE"
    },
    "requestID": "f257f752-cc14-4c52-ad5b-152a46915238",
    "eventID": "0bd1586d-cd3e-4d7a-abaf-fd8bfc3912fd",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
```

```
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

Amazon Cognito는 백엔드 시스템에서 /oauth2/token 엔드포인트에 액세스 토큰에 대한 client_credentials 요청을 제출할 때 다음 이벤트를 로그합니다.

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-12T21:07:05Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Token_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
            "grant_type":
            [
                "client_credentials"
            ],
            "client_id":
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.auth.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_EXAMPLE"
    },
    "requestID": "4f871256-6825-488a-871b-c2d9f55caff2",
```

```
    "eventID": "473e5cbc-a5b3-4578-9ad6-3dfdcb8a6d34",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

Amazon Cognito는 앱이 /oauth2/token 엔드포인트와 새 ID에 대한 새로 고침 토큰 및 액세스 토큰을 교환할 때 다음 이벤트를 로그합니다.

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-12T22:16:40Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Token_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
            "refresh_token":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "grant_type":
            [
```

```
                "refresh_token"
            ],
            "client_id":
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.auth.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_EXAMPLE"
    },
    "requestID": "2829f0c6-a3a9-4584-b046-11756dfe8a81",
    "eventID": "12bd3464-59c7-44fa-b8ff-67e1cf092018",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

CreateIdentityPool에 대한 CloudTrail 이벤트 예제

다음 예제는 `CreateIdentityPool` 작업의 요청에 대한 로그 항목입니다. Alice라는 IAM 사용자가 한 요청입니다.

```
{
    "eventVersion": "1.03",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/Alice",
        "accountId": "123456789012",
        "accessKeyId": "['EXAMPLE_KEY_ID']",
        "userName": "Alice"
    },
    "eventTime": "2016-01-07T02:04:30Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "CreateIdentityPool",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
```

```
    "userAgent": "USER_AGENT",
    "requestParameters": {
        "identityPoolName": "TestPool",
        "allowUnauthenticatedIdentities": true,
        "supportedLoginProviders": {
            "graph.facebook.com": "000000000000000"
        }
    },
    "responseElements": {
        "identityPoolName": "TestPool",
        "identityPoolId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE",
        "allowUnauthenticatedIdentities": true,
        "supportedLoginProviders": {
            "graph.facebook.com": "000000000000000"
        }
    },
    "requestID": "15cc73a1-0780-460c-91e8-e12ef034e116",
    "eventID": "f1d47f93-c708-495b-bff1-cb935a6064b2",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```


GetCredentialsForIdentity에 대한 CloudTrail 이벤트 예시

다음 예제는 `GetCredentialsForIdentity` 작업의 요청에 대한 로그 항목입니다.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown"
    },
    "eventTime": "2023-01-19T16:55:08Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "GetCredentialsForIdentity",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.4",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off
 command/cognito-identity.get-credentials-for-identity",
    "requestParameters": {
        "logins": {
            "cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa":
 "HIDDEN_DUE_TO_SECURITY_REASONS"
        },
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
```

```
    },
    "responseElements": {
        "credentials": {
            "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
            "sessionToken": "aAaAaAaAaAaAab1111111111EXAMPLE",
            "expiration": "Jan 19, 2023 5:55:08 PM"
        },
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
    },
    "requestID": "659dfc23-7c4e-4e7c-858a-1abce884d645",
    "eventID": "6ad1c766-5a41-4b28-b5ca-e223ccb00f0d",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
}
```

GetId에 대한 CloudTrail 이벤트 예시

다음 예제는 `GetId` 작업의 요청에 대한 로그 항목입니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown"
    },
    "eventTime": "2023-01-19T16:55:05Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "GetId",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.4",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off
 command/cognito-identity.get-id",
    "requestParameters": {
        "identityPoolId": "us-east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE",
        "logins": {
```

```
            "cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa":
 "HIDDEN_DUE_TO_SECURITY_REASONS"
        }
    },
    "responseElements": {
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
    },
    "requestID": "dc28def9-07c8-460a-a8f3-3816229e6664",
    "eventID": "c5c459d9-40ec-41fd-8f6b-57865d5a9975",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
}
```

GetOpenIdToken에 대한 CloudTrail 이벤트 예시

다음 예제는 GetOpenIdToken 작업의 요청에 대한 로그 항목입니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown"
    },
    "eventTime": "2023-01-19T16:55:08Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "GetOpenIdToken",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.4",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off
 command/cognito-identity.get-open-id-token",
    "requestParameters": {
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE",
        "logins": {
            "cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa":
 "HIDDEN_DUE_TO_SECURITY_REASONS"
        }
```

```
    },
    "responseElements": {
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
    },
    "requestID": "a506ba18-10d7-4fdb-9548-a8187b2e38bb",
    "eventID": "19ffc1a6-6ed8-4580-a4e1-3062c5ce6457",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
}
```

GetOpenIdTokenForDeveloperIdentity에 대한 CloudTrail 이벤트 예시

다음 예제는 GetOpenIdTokenForDeveloperIdentity 작업의 요청에 대한 로그 항목입니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA1EXAMPLE:johns-AssumedRoleSession",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/johns-AssumedRoleSession",
        "accountId": "111122223333",
        "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROA1EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "attributes": {
                "creationDate": "2023-01-19T16:53:14Z",
                "mfaAuthenticated": "false"
            }
        }
```

```
    },
    "eventTime": "2023-01-19T16:55:08Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "GetOpenIdTokenForDeveloperIdentity",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "27.0.3.154",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off
 command/cognito-identity.get-open-id-token-for-developer-identity",
    "requestParameters": {
        "tokenDuration": 900,
        "identityPoolId": "us-east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE",
        "logins": {
            "JohnsDeveloperProvider": "HIDDEN_DUE_TO_SECURITY_REASONS"
        }
    },
    "responseElements": {
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
    },
    "requestID": "b807df87-57e7-4dd6-b90c-b06f46a61c21",
    "eventID": "f26fed91-3340-4d70-91ae-cdf555547b76",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
}
```

UnlinkIdentity에 대한 CloudTrail 이벤트 예시

다음 예제는 `UnlinkIdentity` 작업의 요청에 대한 로그 항목입니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown"
    },
    "eventTime": "2023-01-19T16:55:08Z",
    "eventSource": "cognito-identity.amazonaws.com",
```

```
    "eventName": "UnlinkIdentity",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.4",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off
 command/cognito-identity.unlink-identity",
    "requestParameters": {
        "logins": {
            "cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa":
 "HIDDEN_DUE_TO_SECURITY_REASONS"
        },
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE",
        "loginsToRemove": ["cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa"]
    },
    "responseElements": null,
    "requestID": "99c2c8e2-9c29-416f-bb17-b650a5cbada9",
    "eventID": "d8e26126-202a-43c2-b458-3f225efaedc7",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
}
```


인터페이스 엔드포인트를 사용하여 Amazon Cognito에 액세스 (AWS PrivateLink)

AWS PrivateLink 를 사용하여 VPC와 Amazon Cognito 간에 프라이빗 연결을 생성할 수 있습니다. 인터넷 게이트웨이, NAT 디바이스, VPN 연결 또는 Direct Connect 연결을 사용하지 않고 VPC에 있는 것처럼 Amazon Cognito에 액세스할 수 있습니다. VPC의 인스턴스는 Amazon Cognito에 액세스하는 데 퍼블릭 IP 주소가 필요하지 않습니다.

AWS PrivateLink에서 제공되는 인터페이스 엔드포인트를 생성하여 이 프라이빗 연결을 설정합니다. 인터페이스 엔드포인트에 대해 사용 설정하는 각 서브넷에서 엔드포인트 네트워크 인터페이스를 생성합니다. 이는 Amazon Cognito로 향하는 트래픽의 진입점 역할을 하는 요청자 관리형 네트워크 인터페이스입니다.

자세한 내용은 AWS PrivateLink 안내서의 AWS PrivateLink를 통해 AWS 서비스 에 액세스를 참조하세요.

> **Important**
>
> 다음 인증 유형은 현재를 통해 지원되지 않습니다. AWS PrivateLink
>
> 1. OAuth 2.0 클라이언트 자격 증명을 사용한 Machine to Machine(M2M) 권한 부여는 토큰 엔드포인트에서 흐릅니다. `GetClientToken` API 작업을 통한 M2M 권한 부여는 사용자 풀 도메인이 필요하지 않기 때문에 지원됩니다.
> 2. 관리형 로그인 및 클래식 호스팅 UI를 사용하여 로그인합니다.

주제

- AWS PrivateLink 통합을 위한 인증 흐름
- 의 운영 모드 AWS PrivateLink
- Amazon Cognito 고려 사항
- 리소스 제어 정책을 사용하여 액세스 제어
- Amazon Cognito용 인터페이스 엔드포인트 생성
- 엔드포인트의 엔드포인트 정책 생성
- AWS PrivateLink 작업에 대한 자격 증명 기반 정책 생성

AWS PrivateLink 통합을 위한 인증 흐름

다음 표에서는 VPCs의 클라이언트가 사용할 수 있는 인증 흐름과 이를 관리하는 데 적용할 수 있는 IAM 정책을 설명합니다. 사용자 풀에 대한 요청에서 평가할 수 있는 정책은 리소스 제어 정책(RCPs), VPC 엔드포인트 정책 및 자격 증명 기반 정책입니다.

Resource	인증 흐름	클라이언트가 VPC 엔드포인트를 전송할 때 평가되는 정책	클라이언트 오리진이 퍼블릭일 때 평가되는 정책
사용자 풀	관리형 로그인 및 클래식 호스팅 UI 로그인	없음(액세스 없음)[1]	없음[2]

Resource	인증 흐름	클라이언트가 VPC 엔드포인트를 전송할 때 평가되는 정책	클라이언트 오리진이 퍼블릭일 때 평가되는 정책
사용자 풀	토큰 엔드포인트에서 Machine-to-machine 권한 부여(`GetClientToken` API 작업을 사용한 M2M 권한 부여는 SDK 및 REST API 인증되지 않은 요청 행에서 다룹니다.)	없음(액세스 없음)[1]	없음[2]
사용자 풀	SDK 및 REST API 인증되지 않은 요청	RCPs, VPC 엔드포인트 정책[3]	RCPs
사용자 풀	SDK 및 REST API SigV4 인증 요청	RCPs, VPC 엔드포인트 정책, 자격 증명 기반 정책[3]	RCPs, 자격 증명 기반 정책
자격 증명 풀	SDK 및 REST API 인증되지 않은 요청(기본 및 향상된 흐름)	RCPs, VPC 엔드포인트 정책	RCPs
자격 증명 풀	SDK 및 REST API SigV4 인증 요청(개발자 인증 흐름)	RCPs, 자격 증명 기반 정책	RCPs, 자격 증명 기반 정책

[1] VPC 엔드포인트는 사용자 풀 도메인에 대한 요청을 수락하지 않습니다. 클라이언트에 인터넷에 대한 경로가 있는 경우 NAT가 적용되어 오리진이 공개됩니다.

[2] 사용자 풀 도메인이 존재하면 VPC 엔드포인트를 전송하는 사용자 풀 요청이 완료되지 않습니다. 모든 클라이언트는 사용자 풀 도메인 및 API 서비스 엔드포인트로만 퍼블릭 전송 경로를 가져올 수 있으므로 사용자 풀에 VPC 엔드포인트를 사용할 수 없습니다. 도메인이 할당된 사용자 풀은와 호환되지 않습니다 AWS PrivateLink.

[3] 사용자 풀에는 도메인이 할당되어 있지 않아야 합니다.

의 운영 모드 AWS PrivateLink

다음 예제 구현 모델은 AWS PrivateLink 및 Amazon Cognito에서 지원됩니다.

Resource	구현	작업
사용자 풀	완전 프라이빗 SDK 또는 REST API 애플리케이션	1. 도메인 삭제 2. VPC 엔드포인트 생성 3. VPC를 제외한 Deny 모든 cognito-idp 작업으로 RCP 구성
사용자 풀	프라이빗 및 퍼블릭	1. 도메인 삭제 2. VPC 엔드포인트 생성
사용자 풀	프라이빗 또는 퍼블릭 OAuth 2.0 권한 부여 서버	1. VPC에서 사용할 수 없음
자격 증명 풀	완전 프라이빗	1. VPC 엔드포인트 생성 2. VPC를 제외한 Deny 모든 자격 증명 작업에 대해 RCP 구성
자격 증명 풀	프라이빗 및 퍼블릭	1. VPC 엔드포인트 생성

Amazon Cognito 고려 사항

Amazon Cognito에 대한 인터페이스 엔드포인트를 설정하기 전에 AWS PrivateLink 가이드의 고려 사항을 검토하세요. Amazon Cognito는 인터페이스 엔드포인트를 통해 모든 Amazon Cognito API 작업을 호출할 수 있도록 지원합니다. 이러한 작업에 대한 자세한 내용은 Amazon Cognito 사용자 풀 API 참조 및 Amazon Cognito 연동 자격 증명 API 참조를 참조하세요.

주제

- 사용자 풀 및 AWS PrivateLink
- 자격 증명 풀 및 AWS PrivateLink

사용자 풀 및 AWS PrivateLink

인터페이스 엔드포인트를 통해 모든 사용자 풀 API 작업에 요청할 수 있지만, 클라이언트 자격 증명 부여 및 관리형 로그인과 같은 사용자 풀 OAuth 2.0 권한 부여 서버에서 애플리케이션이 요청하는 작업은 요청할 수 없습니다.

cognito-idp 사용자 풀 API에 인증되지 않은 인증 및 토큰 권한 부여 API 작업이 있습니다. VPC 엔드포인트 및 리소스 제어 정책에서 인증된 작업에 대한 권한을 부여할 수 있습니다. 자격 증명 기반 정책과 달리 인증되지 않은 작업 및 토큰 인증 작업에 대한 권한을 부여할 수도 있습니다. VPC 엔드포인트 및 리소스 제어 정책 유형은 다른 퍼블릭 작업에 대한 요청을 평가 및 거부하거나 허용할 수 있습니다.

도메인 엔드포인트에 대한 요청도 퍼블릭이지만 정책에서는 평가할 수 없습니다. VPC 프라이빗 DNS는 사용자 풀 도메인에 대한 요청을 VPC 엔드포인트로 라우팅하지 않습니다. 퍼블릭-인터넷 경로를 통해서만 도메인 서비스를 요청할 수 있습니다. 자세한 내용은 정책이 사용자 풀 작업에 미치는 영향 섹션을 참조하세요.

주제

- 지원되는 연산자
- 정책이 사용자 풀 작업에 미치는 영향

지원되는 연산자

VPC의 시스템은 사용자 풀 API 작업에는 요청을 보낼 수 있지만 사용자 풀 도메인 엔드포인트에는 보낼 수 없습니다. 머신 투 머신(M2M), 페더레이션 로그인 및 권한 부여 코드 부여와 같은 도메인 엔드포인트를 사용하는 OpenID Connect(OIDC) 및 OAuth 2.0 워크플로는 VPC 엔드포인트를 통해 액세스할 수 없습니다. machine-to-machine VPC 엔드포인트 정책은 이러한 HTTP 워크플로에 영향을 주지 않으며 처리할 수 없습니다. VPC 내에서 도메인 엔드포인트에 대한 요청은 항상 인터페이스 엔드포인트에서 실패하지만 사용자 풀에 대해 VPC 엔드포인트를 설정할 때 퍼블릭 DNS 및 라우팅을 통해 계속 사용할 수 있습니다.

VPC의 시스템에서 도메인이 할당되지 않도록 Amazon Cognito는 인터페이스 엔드포인트에서 CreateUserPoolDomain 요청을 차단합니다. 이렇게 하면 VPC에 있는 시스템에서 사용자 풀에 도메인이 추가되지 않습니다. 모든 시스템에서 도메인이 추가되지 않도록 하려면 다음 예제와 같은 리소스 제어 정책(RCP)을에 적용합니다 AWS 계정. 이 정책은 지정된 사용자 풀에 대한 CreateUserPoolDomain 작업을 차단합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Principal": "*",
            "Effect": "Deny",
            "Action": [
                "cognito-idp:CreateUserPoolDomain"
```

```
            ],
            "Resource": "arn:aws:cognito-idp:us-east-1:123456789012:userpool/us-
east-1_EXAMPLE"
        }
    ]
}
```

사용자 풀에 도메인이 있을 수 있으며, 모든 경우에 해당 도메인을 사용할 수 없습니다 AWS PrivateLink. `cognito-idp` 서비스 엔드포인트에 대한 모든 SDK 기반 사용자 풀 API 요청은를 AWS PrivateLink제외하고를 통한 요청을 수락합니다`CreateUserPoolDomain`. 사용자 풀 API 서비스 엔드포인트 및 도메인 엔드포인트는 퍼블릭-인터넷 경로를 통해 항상 액세스할 수 있습니다. 퍼블릭 소스의 액세스를 해결하려면 AWS WAF 웹 ACLs.

정책이 사용자 풀 작업에 미치는 영향

일반적으로 퍼블릭 및 미인증 사용자 풀 API 작업을 비롯한 모든 사용자 풀 API 작업은 VPC 엔드포인트 정책 및 리소스 제어 정책(RCPs. VPC 조건 키를 사용하여 자격 증명 기반 정책에서 사용자 풀 액세스에 제한을 적용할 수도 있습니다. SigV4 형식의 인증 정보가 포함된 요청만 자격 증명 기반 정책에서 제어할 수 있습니다. 관리형 로그인 및 클래식 호스팅 UI 작업은 별도의 범주이며 VPC 전송 또는 작업에 대한 모든 유형의 정책 적용에 사용할 수 없습니다.

인증되지 않은 작업

클라이언트 측 애플리케이션에 대한 Amazon Cognito 작업은 SigV4로 인증되지 않습니다. 예제 작업은의 예제 정책에 나와 있습니다엔드포인트의 엔드포인트 정책 생성. 인증되지 않은 작업의 추가 예는 `GetUser` 및 입니다`AssociateSoftwareToken`. 이러한 작업을 자격 증명 기반 정책에 추가해도 아무런 효과가 없습니다. 그러나 VPC 엔드포인트 정책 및 RCPs.

인증되지 않은 작업은 IAM 보안 주체와 연결되지 않습니다. VPC 엔드포인트 정책 또는 RCP는 이러한 작업에 대한 모든 보안 주체를 허용해야 합니다.

인증된 작업

사용자 풀 관리 및 서버 측 인증을 위한 API 작업은 SigV4로 인증됩니다. 인증된 작업의 경우 VPC 엔드포인트에 적용하는 엔드포인트 정책, 조직의 리소스 제어 정책 및 보안 주체에 적용하는 자격 증명 기반 정책으로 보안 주체를 제한할 수 있습니다. 자격 증명 기반 및 리소스 제어 정책은 `aws:SourceVpc` 및와 같은 네트워크 기반 조건 키가 있는 VPC 인식 정책입니다`aws:SourceVpce`.

사용자 풀에 대한 API 작업의 서버 측, 클라이언트 측 및 관리 클래스에 대한 자세한 내용은 섹션을 참조하세요API 및 SDK 인증을 위한 권한 부여 모델.

자격 증명 풀 및 AWS PrivateLink

Amazon Cognito 자격 증명 풀은를 통한 모든 API 작업을 지원합니다 AWS PrivateLink.

주제

- 지원되는 연산자
- AWS STS 통합을 통한 네트워크 컨텍스트 제한
- 서비스별 컨텍스트 키

지원되는 연산자

모든 자격 증명 풀 API 작업은 인터페이스 엔드포인트를 통해 지원됩니다. 자격 증명 풀에는 도메인 엔드포인트가 없으며 동일한 제한이 적용되지 않습니다. 그러나 ID 풀에는 와의 통합으로 인해 네트워크 기반 액세스 제어에 대한 특정 고려 사항이 있습니다 AWS STS.

AWS STS 통합을 통한 네트워크 컨텍스트 제한

자격 증명 풀은 작업을 사용하여 AWS STS `AssumeRoleWithWebIdentity` 임시 AWS 자격 증명을 제공합니다. 자격 증명 풀이 향상된 인증 흐름 AWS PrivateLink 에서를 AWS STS 호출할 때 , `aws:SourceIp` `aws:SourceVpc`및와 같은 네트워크 컨텍스트 키`aws:SourceVpce`에는 애플리케이션의 네트워크 컨텍스트가 아닌 자격 증명 풀 서비스 인프라의 값이 포함됩니다.

IAM 역할 신뢰 정책 또는 리소스 제어 정책(RCPs 네트워크 기반 조건 키를 사용하여 액세스를 제한하는 경우 자격 증명 풀 작업이 예기치 않게 거부될 수 있습니다. 이 제한을 해결하기 위해 다음 방법 중 하나를 사용할 수 있습니다.

서비스 식별을 위한 보안 주체 태그

자격 증명 풀에 사용되는 IAM 역할에 태그를 지정하고 보안 주체에 적절한 태그가 있는 경우 작업을 허용하도록 정책을 수정합니다. 먼저 자격 증명 풀 역할에 태그를 추가합니다.

```
aws iam tag-role \
    --role-name MyIdentityPoolRole \
    --tags Key=CognitoServiceCall,Value=true
```

그런 다음 태그가 지정된 보안 주체를 허용하도록 네트워크 기반 정책을 수정합니다. 예를 들어 RCP에서는 다음과 같습니다.

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

```
    "Statement": [
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Resource": "*",
            "Condition": {
                "NotIpAddress": {
                    "aws:SourceIp": ["allowed-ip-ranges"]
                },
                "StringNotEqualsIfExists": {
                    "aws:ResourceTag/CognitoServiceCall": "true"
                }
            }
        }
    ]
}
```

서비스별 컨텍스트 키

자격 증명 풀은 VPC 엔드포인트 정책 및 RCPs에서 리소스 수준 권한 부여를 위한 서비스별 컨텍스트 키를 제공합니다. 이러한 컨텍스트 키를 사용하면 세분화된 액세스 제어를 활성화하고 정책에서 인증된 사용자와 인증되지 않은 사용자를 구분할 수 있습니다.

GetId, GetCredentialsForIdentity, GetOpenIdTokenGetOpenIdToken, UnlinkIdentity와 같은 non-SigV4 작업에 사용할 수 있는 서비스별 컨텍스트 키

- `cognito-identity-unauth:IdentityPoolArn` - 인증되지 않은 사용자의 자격 증명 풀 ARN을 기준으로 액세스를 필터링합니다.
- `cognito-identity-unauth:AccountId` - 인증되지 않은 사용자의 AWS 계정 ID를 기준으로 액세스를 필터링합니다.
- `cognito-identity-auth:IdentityPoolArn` - 인증된 사용자의 자격 증명 풀 ARN을 기준으로 액세스를 필터링합니다.
- `cognito-identity-auth:AccountId` - 인증된 사용자의 AWS 계정 ID를 기준으로 액세스를 필터링합니다.

DeleteIdentities 및 DescribeIdentity와 같은 SigV4 작업에 사용할 수 있는 서비스별 컨텍스트 키

- `cognito-identity:IdentityPoolArn` - 자격 증명 풀 ARN을 기준으로 액세스를 필터링합니다.

다음 예제와 같이 VPC 엔드포인트 정책에서 이러한 컨텍스트 키를 사용하여 인증 상태에 따라 액세스를 제한할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "cognito-identity:GetId",
                "cognito-identity:GetCredentialsForIdentity"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cognito-identity-unauth:IdentityPoolArn": "arn:aws:cognito-
identity:us-east-1:123456789012:identitypool/us-east-1:12345678-ffff-ffff-ffff-123456"
                }
            }
        }
    ]
}
```

리소스 제어 정책을 사용하여 액세스 제어

Amazon Cognito는 리소스 제어 정책(RCP)을 사용하여 리소스에 대한 액세스 제어를 지원합니다.RCPs 네트워크 기반 조건 키를 사용하여 RCPs는 사용자 풀 및 자격 증명 풀에 AWS PrivateLink 액세스할 수 있는 네트워크와 작업을 정의할 수 있습니다. RCPs의 `Action` 문은 인증된 사용자 풀 API 작업과 인증되지 않은 사용자 풀 API 작업 모두에 대한 액세스를 제어할 수 있습니다.

예를 들어, 다음 예제 정책은 특정 VPC의 모든 사용자 풀에 대한 액세스를 금지합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyCognitoAccessOutsideVPC",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "cognito-idp:*",
      "Resource": "*",
```

```
      "Condition": {
        "StringNotEqualsIfExists": {
          "aws:SourceVpc": "vpc-02d6770f46ef1653b"
        }
      }
    }
  ]
}
```

Amazon Cognito용 인터페이스 엔드포인트 생성

Amazon VPC 콘솔 또는 ()를 사용하여 Amazon Cognito에 대한 인터페이스 엔드포인트를 생성할 수 있습니다AWS CLI. AWS Command Line Interface 자세한 내용은 AWS PrivateLink 안내서의 인터페이스 엔드포인트 생성을 참조하세요.

다음 서비스 이름을 사용하여 Amazon Cognito 사용자 풀에 대한 인터페이스 엔드포인트를 생성합니다.

```
com.amazonaws.region.cognito-idp
```

다음 서비스 이름을 사용하여 Amazon Cognito 자격 증명 풀에 대한 인터페이스 엔드포인트를 생성합니다.

```
com.amazonaws.region.cognito-identity
```

인터페이스 엔드포인트에 대해 프라이빗 DNS를 활성화하면 기본 리전 DNS 이름을 사용하여 Amazon Cognito에 API 요청을 할 수 있습니다. 예를 들어 사용자 풀의 `cognito-idp.us-east-1.amazonaws.com` 경우 , 자격 증명 풀의 `cognito-identity.us-east-1.amazonaws.com` 경우 입니다.

엔드포인트의 엔드포인트 정책 생성

엔드포인트 정책은 인터페이스 엔드포인트에 연결할 수 있는 IAM 리소스입니다. 기본 엔드포인트 정책은 인터페이스 엔드포인트를 통해 Amazon Cognito에 대한 전체 액세스를 허용합니다. VPC에서 Amazon Cognito에 허용되는 액세스를 제어하려면 인터페이스 엔드포인트에 사용자 지정 엔드포인트 정책을 연결합니다.

엔드포인트 정책은 다음 정보를 지정합니다.

- 작업을 수행할 수 있는 위탁자(AWS 계정, IAM 사용자, IAM 역할)

- 수행할 수 있는 작업
- 작업을 수행할 수 있는 리소스.
- 요청이 허용되거나 거부되기 전에 충족되어야 하는 조건입니다.

자세한 내용은 AWS PrivateLink 안내서의 엔드포인트 정책을 사용하여 서비스에 대한 액세스 제어를 참조하세요.

예: 사용자 풀 작업에 대한 VPC 엔드포인트 정책

다음은 사용자 풀에 대한 사용자 지정 엔드포인트 정책의 예입니다. 이 정책을 인터페이스 엔드포인트에 연결하면 모든 리소스의 모든 보안 주체에 대해 나열된 사용자 풀 작업에 대한 액세스 권한이 부여됩니다.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Principal": {
            "AWS": "arn:aws:iam::123456789012:assumed-role/MyWebAppRole/
MyWebAppSession"
         },
         "Effect": "Allow",
         "Action": [
            "cognito-idp:AdminInitiateAuth",
            "cognito-idp:AdminRespondToAuthChallenge",
            "cognito-idp:AdminSetUserPassword"
         ],
         "Resource":"arn:aws:cognito-idp:us-east-1:123456789012:userpool/us-
east-1_EXAMPLE"
      },
      {
         "Effect": "Allow",
         "Action": [
            "cognito-idp:InitiateAuth",
            "cognito-idp:RespondToAuthChallenge",
            "cognito-idp:ForgotPassword",
            "cognito-idp:ConfirmForgotPassword"
         ],
         "Resource":"arn:aws:cognito-idp:us-east-1:123456789012:userpool/us-
east-1_EXAMPLE"
      }
```

```
    ]
}
```

예: 자격 증명 풀 작업에 대한 VPC 엔드포인트 정책

다음은 자격 증명 풀에 대한 사용자 지정 엔드포인트 정책의 예입니다. 이 정책은 서비스별 컨텍스트 키를 사용하여 특정 자격 증명 풀의 인증된 사용자에 대한 액세스를 제한합니다.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": "*",
         "Action": [
            "cognito-identity:GetId",
            "cognito-identity:GetCredentialsForIdentity",
            "cognito-identity:GetOpenIdToken"
         ],
         "Resource": "*",
         "Condition": {
            "StringEquals": {
               "cognito-identity-auth:IdentityPoolArn": "arn:aws:cognito-identity:us-
east-1:123456789012:identitypool/us-east-1:12345678-ffff-ffff-ffff-123456"
            }
         }
      }
   ]
}
```

AWS PrivateLink 작업에 대한 자격 증명 기반 정책 생성

자격 증명 기반 정책은 AWS 보안 주체에 연결할 수 있는 IAM 리소스입니다. IAM 인증 작업에 대한 자격 증명 기반 정책을 사용하여 VPC 엔드포인트를 통해 Amazon Cognito에 대한 액세스를 제어할 수 있습니다. 엔드포인트 정책과 달리 자격 증명 기반 정책에서는 인증되지 않은 작업에 대한 권한을 구성할 수 없습니다. 인증된 또는 관리 작업에는 서명 버전 4 인증이 필요합니다. 사용자 풀의 경우 인증된 작업에는 AdminInitiateAuth와 같은 서버 측 인증 요청과 UpdateUserPool과 같은 관리 요청이 포함됩니다. 자격 증명 풀의 경우 인증된 작업에는 DeleteIdentities 및 DescribeIdentity와 같은 관리 요청이 포함됩니다.

자격 증명 기반 정책은 다음 정보를 지정합니다.

- 수행할 수 있는 작업
- 작업을 수행할 수 있는 리소스.
- 요청이 허용되거나 거부되기 전에 충족되어야 하는 조건입니다.

예: 사용자 풀 서버 측 인증을 위한 자격 증명 기반 정책

다음 예제 정책은 지정된 엔드포인트에서 지정된 사용자 풀의 나열된 사용자 풀 작업에 대한 액세스 권한을 부여합니다. 웹 애플리케이션의 수임된 IAM 역할에이 정책을 적용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cognito-idp:AdminInitiateAuth",
                "cognito-idp:AdminRespondToAuthChallenge",
                "cognito-idp:AdminSetUserPassword"
            ],
            "Resource": "arn:aws:cognito-idp:us-east-1:123456789012:userpool/us-
east-1_EXAMPLE",
            "Condition": {
                "StringEquals": {
                    "aws:SourceVpce": "vpce-1a2b3c4d"
                }
            }
        }
    ]
}
```

예: 자격 증명 풀 관리 작업에 대한 자격 증명 기반 정책

다음 예제 정책은 지정된 VPC 엔드포인트에서 자격 증명 풀 관리 작업에 대한 액세스 권한을 부여합니다. 자격 증명 풀 관리를 수행해야 하는 IAM 보안 주체에이 정책을 적용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "cognito-identity:DeleteIdentities",
                "cognito-identity:DescribeIdentity"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceVpce": "vpce-1a2b3c4d",
                    "cognito-identity:IdentityPoolArn": "arn:aws:cognito-identity:us-
east-1:123456789012:identitypool/us-east-1:12345678-ffff-ffff-ffff-123456"
                }
            }
        }
    ]
}
```

Amazon Cognito에 대한 규정 준수 검증

타사 감사자는 여러 규정 준수 프로그램의 일환으로 Amazon Cognito의 보안 및 AWS 규정 준수를 평가합니다. 여기에는 SOC, PCI, FedRAMP, HIPAA 등이 포함됩니다.

특정 규정 준수 프로그램 범위에 속하는 AWS 서비스 목록은 규정 준수 AWS 프로그램 범위에 속하는 서비스규정 준수 . 일반적인 내용은 AWS 규정 준수 프로그램을 참조하세요.

를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다 AWS Artifact. 자세한 내용은 에서 보고서 다운로드 AWS Artifact에서 .

Amazon Cognito 사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 결정됩니다. AWS 에서는 규정 준수를 지원할 다음과 같은 리소스를 제공합니다.

- 보안 및 규정 준수 빠른 시작 안내서 – 이 배포 안내서에서는 아키텍처 고려 사항에 관해 설명하고 AWS에서 보안 및 규정 준수에 중점을 둔 기본 환경을 배포하기 위한 단계를 제공합니다.
- HIPAA 보안 및 규정 준수 백서 설계 -이 백서에서는 기업이를 사용하여 HIPAA 준수 애플리케이션을 AWS 생성하는 방법을 설명합니다.
- AWS 규정 준수 리소스 -이 워크북 및 가이드 모음은 산업 및 위치에 적용될 수 있습니다.
- AWS Config 개발자 안내서의 규칙을 사용하여 리소스 평가 - 리소스 구성이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 AWS Config평가합니다.
- AWS Security Hub CSPM -이 AWS 서비스는 보안 업계 표준 및 모범 사례 준수를 확인하는 데 도움이 AWS 되는 내 보안 상태에 대한 포괄적인 보기를 제공합니다.

Amazon Cognito의 복원성

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다. 리전은 물리적으로 분리되고 격리된 다수의 가용 영역을 제공하며, 이러한 영역은 짧은 지연 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워크를 통해 연결되어 있습니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

주제

- 리전 데이터 고려 사항

리전 데이터 고려 사항

Amazon Cognito 사용자 풀은 각각 한 AWS 리전에서 생성되며 해당 리전에만 사용자 프로필 데이터를 저장합니다. 사용자 풀은 선택적 기능이 구성된 방식에 따라 사용자 데이터를 다른 AWS 리전으로 전송할 수 있습니다.

- Amazon Cognito 사용자 풀을 사용한 이메일 주소 라우팅 확인에 기본 `no-reply@verificationemail.com` 이메일 주소 설정을 사용할 경우, 이메일은 연결된 사용자 풀과 동일한 리전을 통해 라우팅됩니다.
- 다른 이메일 주소를 사용하여 Amazon Cognito 사용자 풀로 Amazon Simple Email Service(Amazon SES)를 구성하는 경우 해당 이메일 주소는 Amazon SES의 이메일 주소와 연결된 AWS 리전을 통해 라우팅됩니다.
- Amazon Cognito 사용자 풀에서 전송된 SMS 메시지는 이메일 또는 전화 확인 구성에 다르게 명시되어 있지 않는 한 동일한 리전 Amazon SNS를 통해 라우팅됩니다.
- Amazon Cognito 사용자 풀과 함께 Amazon Pinpoint 분석이 사용되는 경우 이벤트 데이터는 미국 동부(버지니아 북부) 리전으로 라우팅됩니다.

> **Note**
>
> Amazon Pinpoint는 북미, 유럽, 아시아 및 오세아니아의 여러 AWS 리전에서 사용할 수 있습니다. Amazon Pinpoint 리전에는 Amazon Pinpoint API가 포함되어 있습니다. Amazon Cognito에서 Amazon Pinpoint 리전을 지원하는 경우 Amazon Cognito는 동일한 Amazon Pinpoint 리전 내의 Amazon Pinpoint 프로젝트로 이벤트를 전송합니다. Amazon Cognito Amazon

Pinpoint에서 리전을 지원하지 않는 경우 Amazon Cognito는 us-east-1에서 이벤트 전송만 지원합니다. 자세한 Amazon Pinpoint 리전 정보는 Amazon Pinpoint 엔드포인트 및 할당량과 Amazon Cognito 사용자 풀로 Amazon Pinpoint 분석 사용을 참조하세요.

Amazon Cognito의 인프라 보안

관리형 서비스인 Amazon Cognito는 AWS 글로벌 네트워크 보안으로 보호됩니다. AWS 보안 서비스 및가 인프라를 AWS 보호하는 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하세요. 인프라 보안 모범 사례를 사용하여 환경을 설계하려면 보안 원칙 AWS Well-Architected Framework의 인프라 보호를 참조하세요 AWS .

AWS 에서 게시한 API 호출을 사용하여 네트워크를 통해 Amazon Cognito에 액세스합니다. 클라이언트는 다음을 지원해야 합니다.

- Transport Layer Security(TLS). TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

Amazon Cognito 사용자 풀의 구성 및 취약성 분석

AWS 는 게스트 운영 체제(OS) 및 데이터베이스 패치, 방화벽 구성, 재해 복구와 같은 기본 보안 작업을 처리합니다. 적합한 제3자가 이 절차를 검토하고 인증하였습니다. 자세한 내용은 다음 리소스를 참조하세요.

- Amazon Cognito에 대한 규정 준수 검증
- 공동 책임 모델

AWS Amazon Cognito에 대한 관리형 정책

사용자, 그룹 및 역할에 권한을 추가하려면 직접 정책을 작성하는 것보다 AWS 관리형 정책을 사용하는 것이 더 쉽습니다. 팀에 필요한 권한만 제공하는 IAM 고객 관리형 정책을 생성하기 위해서는 시간과 전문 지식이 필요합니다. 빠르게 시작하려면 AWS 관리형 정책을 사용할 수 있습니다. 이러한 정책은

일반적인 사용 사례를 다루며 AWS 계정에서 사용할 수 있습니다. AWS 관리형 정책에 대한 자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책을 참조하세요.

AWS 서비스는 AWS 관리형 정책을 유지 관리하고 업데이트합니다. AWS 관리형 정책에서는 권한을 변경할 수 없습니다. 서비스에서 때때로 추가 권한을 AWS 관리형 정책에 추가하여 새로운 기능을 지원합니다. 이 유형의 업데이트는 정책이 연결된 모든 ID(사용자, 그룹 및 역할)에 적용됩니다. 서비스는 새로운 기능이 시작되거나 새 작업을 사용할 수 있을 때 AWS 관리형 정책에 업데이트됩니다. 서비스는 AWS 관리형 정책에서 권한을 제거하지 않으므로 정책 업데이트로 인해 기존 권한이 손상되지 않습니다.

또한는 여러 서비스에 걸쳐 있는 직무에 대한 관리형 정책을 AWS 지원합니다. 예를 들어 ReadOnlyAccess AWS 관리형 정책은 모든 AWS 서비스 및 리소스에 대한 읽기 전용 액세스를 제공합니다. 서비스가 새 기능을 시작하면는 새 작업 및 리소스에 대한 읽기 전용 권한을 AWS 추가합니다. 직무 정책의 목록과 설명은 IAM 사용 설명서의 직무에 관한AWS 관리형 정책을 참조하세요.

AWS Amazon Cognito에 대한 액세스 권한을 부여하는 관리형 IAM 정책

- `AmazonCognitoPowerUser` - 자격 증명 풀 및 사용자 풀의 모든 측면을 액세스하고 관리할 수 있는 권한입니다. 이 정책의 권한을 보려면 AmazonCognitoPowerUser를 참조하세요.
- `AmazonCognitoReadOnly` - 자격 증명 풀 및 사용자 풀에 대한 읽기 전용 액세스 권한입니다. 이 정책의 권한을 보려면 AmazonCognitoReadOnly를 참조하세요.
- `AmazonCognitoDeveloperAuthenticatedIdentities` - 인증 시스템을 Amazon Cognito와 통합할 권한입니다. 이 정책의 권한을 보려면 AmazonCognitoDeveloperAuthenticatedIdentities를 참조하세요.

이러한 정책은 Amazon Cognito 팀에서 관리하므로 새 API가 추가되더라도 IAM 사용자는 계속해서 동일한 수준의 액세스 권한을 갖게 됩니다.

> **Note**
>
> 새 자격 증명 풀을 생성할 때 인증된 사용자 및 게스트 사용자 액세스를 위한 새 역할을 자동으로 생성할 수 있습니다. 새 IAM 역할로 자격 증명 풀을 생성하는 관리자에게는 역할을 생성할 수 있는 IAM 권한도 있어야 합니다.

인증되지 않은 게스트 액세스 권한이 있는 자격 증명 풀은 인증되지 않은 사용자에게 세션 정책으로 추가 AWS 관리형 정책을 적용합니다. https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html#policies_session 이 AWS 관리형 정책은 관리 용도로 사용되지 않습니다. 대신,

자격 증명 풀 향상된 인증 흐름에서 게스트 사용자에게 적용할 수 있는 권한의 범위를 제한합니다. 자세한 내용은 IAM 역할 섹션을 참조하세요.

AWS Amazon Cognito가 게스트 사용자에게 부여하는 관리형 IAM 정책

- `AmazonCognitoUnAuthedIdentitiesSessionPolicy` - 인라인 세션 정책과 함께 IAM 관리자가 ID 풀 게스트 사용자에게 부여할 수 있는 권한을 제한합니다. Amazon Cognito는 이 정책을 게스트 세션에 자동으로 적용합니다. 자세한 내용은 게스트에 대한 AWS 관리형 세션 정책 섹션을 참조하세요.

AWS 관리형 정책에 대한 Amazon Cognito 업데이트

이 서비스가 이러한 변경 사항을 추적하기 시작한 이후부터 Amazon Cognito의 AWS 관리형 정책 업데이트에 대한 세부 정보를 봅니다. 이 페이지의 변경 사항에 대한 자동 알림을 받아보려면 Amazon Cognito Document history(문서 기록) 페이지에서 RSS 피드를 구독하세요.

변경	설명	Date
`AmazonCognitoPowerUser` - 변경 사항	Amazon Cognito는 Amazon Cognito 사용자 풀 관리 고급 사용자를 위한 AWS 최종 사용자 메시징 SMS API 작업 DescribeAccountAttributes 사용을 허용하는 새 작업을 추가했습니다.	2025년 2월 27일
`AmazonCognitoUnAuthedIdentitiesSessionPolicy` - 변경 사항	Amazon Cognito는 자격 증명 풀에서 인증 AWS Key Management Service 되지 않은(게스트) 사용자의 사용을 허용하는 새로운 작업을 추가했습니다.	2024년 10월 30일

변경	설명	Date
AmazonCognitoUnAuthedIdentitiesSessionPolicy - 변경 사항	Amazon Cognito는 ID 풀의 인증되지 않은(게스트) 사용자를 위해 Amazon Location Service를 사용할 수 있도록 허용하는 새 작업을 추가했습니다.	2024년 8월 9일
AmazonCognitoUnAuthedIdentitiesSessionPolicy -새 정책	자격 증명 풀에 있는 게스트 사용자의 권한 범위 축소에 대한 AWS 관리형 정책이 추가되었습니다.	2023년 7월 14일
AmazonCognitoPowerUser 및 AmazonCognitoReadOnly -변경	고급 사용자가 Amazon Cognito 사용자 풀에 대한 AWS WAF 웹 ACLs 연결을 보고 관리할 수 있는 새 권한이 추가되었습니다. 읽기 전용 사용자가 Amazon Cognito 사용자 풀에 대한 AWS WAF 웹 ACLs 연결을 볼 수 있도록 허용하는 새 권한이 추가되었습니다.	2022년 7월 19일

변경	설명	Date
`AmazonCognitoPowerUser` - 변경 사항	Amazon Cognito가 Amazon Simple Notification Service `PutIdentityPolicy` 및 `ListConfigurationSets` 작업을 호출할 수 있도록 허용하는 새 권한을 추가했습니다. 이 변경으로 인해 Amazon Cognito 사용자 풀은 Amazon SES 전송 권한 부여 정책을 업데이트하고 사용자 풀에서 이메일 전송을 구성할 때 Amazon SES 구성 집합을 적용할 수 있습니다.	2021년 11월 17일
`AmazonCognitoPowerUser` - 변경 사항	Amazon Cognito가 Amazon Simple Notification Service의 `GetSMSSandboxAccountStatus` 작업을 호출하도록 허용하는 새 권한이 추가되었습니다. 이 변경으로 인해 Amazon Cognito 사용자 풀은 사용자 풀을 통해 모든 최종 사용자에게 메시지를 전송하기 위해 Amazon Simple Notification Service 샌드박스를 종료해야 할지 여부를 결정할 수 있습니다.	2021년 6월 1일
Amazon Cognito가 변경 사항 추적 시작	Amazon Cognito는 AWS 관리형 정책에 대한 변경 사항 추적을 시작했습니다.	2021년 3월 1일

레거시

> **Important**
>
> Amazon Cognito Sync는 신규 고객이 이용할 수 없습니다. 새 애플리케이션을 시작하는 경우 및 DynamoDBAWS AppSync와 같은 현재 대안을 대신 사용합니다. 이 섹션의 내용은 기존 Amazon Cognito Sync 구현이 있는 고객을 위해 유지됩니다.

이 섹션에서는 이전 세대 Amazon Cognito 기능을 수집합니다. Amazon Cognito Sync는 위에서 언급한 대로 신규 고객이 이용할 수 없습니다. 클래식 호스팅 UI는 앱 클라이언트당 선택할 수 있는 이전 브랜딩 버전이지만 관리형 로그인은 새 작업을 위한 현재 브랜딩 버전입니다. 클래식 호스팅 UI에서 관리형 로그인으로 앱 클라이언트를 이동하려면 섹션을 참조하세요클래식 호스팅 UI에서 관리형 로그인으로 마이그레이션.

주제

- Amazon Cognito Sync
- Amazon Cognito Sync를 자격 증명 풀과 함께 사용
- 호스팅 UI(클래식) 브랜딩 사용자 지정

Amazon Cognito Sync

> **Note**
>
> Amazon Cognito Sync는 더 이상 신규 고객에게 공개되지 않습니다. Amazon Cognito Sync의 대안은 AWS AppSync 및 DynamoDB를 탐색하세요. 자세히 알아보기

> **Note**
>
> Amazon Cognito Sync를 처음 사용하는 경우 AWS AppSync를 사용하세요. Amazon Cognito Sync와 마찬가지로 AWS AppSync 는 디바이스 간에 애플리케이션 데이터를 동기화하기 위한 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 디바이스 간에 애플리케이션 관련 사용자 데이터를 동기화할 수 있는 AWS 서비스 및 클라이언트 라이브러리입니다. Amazon Cognito Sync는 자체 백엔드를 사용하지 않고도 모바일 디바이스와 웹 간에 사용자 프로필 데이터를 동기화할 수 있습니다. 클라이언트 라이브러리는 디바이스 연결 상태와 관계없이 앱에서 데이터를 읽고 쓸 수 있도록 로컬로 데이터를 캐싱합니다. 디바이스가 온라인 상태일 때 데이터를 동기화할 수 있습니다. 푸시 동기화를 설정한 경우 업데이트가 있음을 다른 디바이스에 즉시 알릴 수 있습니다.

Amazon Cognito 자격 증명 리전 가용성에 대한 자세한 내용은 AWS 서비스 리전 가용성을 참조하세요.

Amazon Cognito Sync에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- Amazon Cognito Sync 가용성 변경
- Amazon Cognito Sync 시작하기
- 클라이언트 간 데이터 동기화
- 이벤트 콜백 처리
- 푸시 동기화 구현
- Amazon Cognito Sync 스트림 구현
- Amazon Cognito Events를 사용하여 워크플로 사용자 지정

Amazon Cognito Sync 가용성 변경

Amazon Cognito Sync는 더 이상 신규 고객에게 공개되지 않습니다. Amazon Cognito Sync의 대안은 AWS AppSync 및 DynamoDB를 살펴보세요.

이 페이지에서는 Amazon Cognito Sync 변경 사항에 대한 정보를 제공하고 Amazon Cognito Sync 고객을 위한 대안을 제공합니다.

AWS AppSync는 개발자가 GraphQL APIs. DynamoDB는 프로필, 기본 설정 및 애플리케이션 설정과 같은 사용자별 데이터를 위한 키-값 저장소입니다. Amazon Cognito Sync에서 마이그레이션하려면 새 서비스에 의한 다양한 APIs 및 쿼리 패턴으로 애플리케이션에 대한 데이터 전송 및 업데이트가 필요합니다.

추가 질문이 있는 경우 AWS Support에 문의하십시오.

자주 묻는 질문(FAQ)

서비스에 어떤 의미가 있습니까?

Amazon Cognito Sync는 더 이상 신규 고객을 받지 않습니다. 서비스는 기존 고객을 위해 계속 운영되지만 새로운 기능 개발은 없습니다.

기존 고객은 어떤 영향을 받나요?

기존 고객의 워크로드에는 어떠한 차질도 발생하지 않을 것입니다. Amazon Cognito Sync를 정상적으로 계속 사용할 수 있습니다.

Amazon Cognito Sync의 권장 대안은 무엇인가요?

- AWS AppSync - GraphQL APIs 수행합니다.
- DynamoDB - 간단한 키-값 사용자 데이터 스토리지(프로파일, 기본 설정, 설정).

데이터를 마이그레이션하려면 어떻게 해야 하나요?

마이그레이션하려면 데이터 스키마를 업데이트하고, Amazon Cognito Sync 스토어에서 선택한 대안으로 데이터를 마이그레이션하고, 새 서비스를 사용하도록 클라이언트 애플리케이션을 업데이트해야 합니다.

문제가 있는 경우 어떻게 도움을 받을 수 있나요?

이에 대한 질문이 있는 경우 AWS Support에 문의하십시오.

Amazon Cognito Sync 시작하기

> **Note**
>
> Amazon Cognito Sync를 처음 사용하는 경우 AWS AppSync를 사용하세요. Amazon Cognito Sync와 마찬가지로 AWS AppSync 는 디바이스 간에 애플리케이션 데이터를 동기화하기 위한 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.

> **Note**
>
> 이 섹션의 코드 예제에서는 아카이브된 AWS Mobile SDK(Android, iOS, Unity 및 Xamarin/용)의 Amazon Cognito Sync 클라이언트를 사용합니다.NET). 해당 Mobile SDK는 더 이상 유지

관리되지 않으며 현재 패키지 리포지토리에서 해당 패키지를 사용할 수 없습니다. 이 예제는 해당 SDK와 여전히 통합되는 기존 고객을 위해 남아 있습니다. 새 개발의 경우를 사용합니다AWS AppSync.

Amazon Cognito Sync는 애플리케이션 관련 사용자 데이터의 디바이스 간 동기화를 지원하는 AWS 서비스 및 클라이언트 라이브러리입니다. 를 사용하여 모바일 디바이스와 웹 애플리케이션에서 사용자 프로파일 데이터를 동기화할 수 있습니다. 클라이언트 라이브러리는 디바이스 연결 상태와 관계없이 앱에서 데이터를 읽고 쓸 수 있도록 로컬로 데이터를 캐싱합니다. 디바이스가 온라인 상태인 경우 데이터를 동기화할 수 있으며, 푸시 동기화를 설정한 경우 업데이트가 가능함을 다른 디바이스에 즉시 알립니다.

Amazon Cognito에서 자격 증명 풀 설정

Amazon Cognito Sync에는 사용자 자격 증명을 제공하기 위해 Amazon Cognito 자격 증명 풀이 필요합니다. 따라서 먼저 자격 증명 풀을 설정해야 Amazon Cognito Sync를 사용할 수 있습니다. 자격 증명 풀을 생성하고 SDK를 설치하려면 Amazon Cognito ID 풀 시작를 참조하세요.

데이터 저장 및 동기화

자격 증명 풀을 설정하고 SDK를 설치하면 디바이스 간에 데이터를 저장하고 동기화하는 작업을 시작할 수 있습니다. 자세한 내용은 클라이언트 간 데이터 동기화 섹션을 참조하세요.

클라이언트 간 데이터 동기화

> **Note**
>
> Amazon Cognito Sync를 처음 사용하는 경우 AWS AppSync를 사용하세요. Amazon Cognito Sync와 마찬가지로 AWS AppSync 는 디바이스 간에 애플리케이션 데이터를 동기화하기 위한 서비스입니다.
> 앱 기본 설정이나 게임 상태 같은 사용자 데이터를 동기화할 수 있습니다. 또한 이러한 기능을 더욱 확장해, 복수의 사용자가 공유 데이터를 실시간으로 동기화하고 협업할 수 있게 합니다.

Amazon Cognito를 사용하면 키-값 쌍이 포함된 데이터 세트에 사용자 데이터를 저장할 수 있습니다. Amazon Cognito는 이 데이터를 자격 증명 풀의 자격 증명과 연결하여 앱이 로그인 및 디바이스 전반에서 액세스할 수 있도록 합니다. Amazon Cognito 서비스와 최종 사용자의 디바이스 간에 이 데이터를 동기화하려면 동기화 메서드를 호출합니다. 각 데이터 세트의 최대 크기는 1MB일 수 있습니다. 자격 증명에 데이터 세트를 최대 20개까지 연결할 수 있습니다.

Amazon Cognito Sync 클라이언트는 자격 증명 데이터에 대한 로컬 캐시를 생성합니다. 앱에서 키를 읽고 쓸 때 이 로컬 캐시와 통신합니다. 이렇게 통신하면 디바이스에서 수행한 모든 변경 사항을 오프라인 상태에서도 디바이스에서 즉시 사용할 수 있습니다. 동기화 메서드가 호출되면 서비스에서 변경 사항을 디바이스로 가져오고 모든 로컬 변경 사항이 서비스로 푸시됩니다. 이때 변경 사항은 다른 디바이스에서 동기화하는 데 사용 가능합니다.

Amazon Cognito Sync 클라이언트 초기화

Amazon Cognito Sync 클라이언트를 초기화하려면 먼저 자격 증명 공급자를 생성해야 합니다. 자격 증명 공급자는 앱이 AWS 리소스에 액세스할 수 있도록 임시 AWS 자격 증명을 획득합니다. 필요한 헤더 파일도 가져와야 합니다. 다음 단계를 사용하여 Amazon Cognito Sync 클라이언트를 초기화합니다.

Android

1. 자격 증명 얻기의 지침에 따라 자격 증명 공급자를 생성합니다.
2. 다음과 같이 Amazon Cognito 패키지를 가져옵니다. `import com.amazonaws.mobileconnectors.cognito.*;`
3. Amazon Cognito Sync를 초기화합니다. 다음과 같이 Android 앱 컨텍스트, 자격 증명 풀 ID, AWS 리전 및 초기화된 Amazon Cognito 자격 증명 공급자를 전달합니다.

```
CognitoSyncManager client = new CognitoSyncManager(
    getApplicationContext(),
    Regions.YOUR_REGION,
    credentialsProvider);
```

iOS - Objective-C

1. 자격 증명 얻기의 지침에 따라 자격 증명 공급자를 생성합니다.
2. 다음과 같이 `AWSCore` 및 `Cognito`를 가져오고 `AWSCognito`를 초기화합니다.

```
#import <AWSiOSSDKv2/AWSCore.h>
#import <AWSCognitoSync/Cognito.h>

AWSCognito *syncClient = [AWSCognito defaultCognito];
```

3. CocoaPods를 사용하는 경우 `<AWSiOSSDKv2/AWSCore.h>`를 `AWSCore.h`로 바꿉니다. Amazon Cognito 가져오기에 대해 동일한 구문을 따릅니다.

iOS - Swift

1. 자격 증명 얻기의 지침에 따라 자격 증명 공급자를 생성합니다.
2. 다음과 같이 AWSCognito를 가져오고 초기화합니다.

```
import AWSCognito
let syncClient = AWSCognito.default()!
```

JavaScript

1. Amazon Cognito Sync Manager for JavaScript를 다운로드합니다.
2. 프로젝트에 Sync Manager 라이브러리를 포함합니다.
3. 자격 증명 얻기의 지침에 따라 자격 증명 공급자를 생성합니다.
4. 다음과 같이 Sync 관리자를 초기화합니다.

```
var syncManager = new AWS.CognitoSyncManager();
```

Unity

1. 자격 증명 얻기의 지침에 따라 CognitoAWSCredentials의 인스턴스를 생성합니다.
2. CognitoSyncManager의 인스턴스를 만듭니다. 다음과 같이 CognitoAwsCredentials 객체 및 AmazonCognitoSyncConfig를 전달하고 최소한 리전 세트를 포함합니다.

```
AmazonCognitoSyncConfig clientConfig = new AmazonCognitoSyncConfig { RegionEndpoint =
 REGION };
CognitoSyncManager syncManager = new CognitoSyncManager(credentials, clientConfig);
```

Xamarin

1. 자격 증명 얻기의 지침에 따라 CognitoAWSCredentials의 인스턴스를 생성합니다.
2. CognitoSyncManager의 인스턴스를 만듭니다. 다음과 같이 CognitoAwsCredentials 객체 및 AmazonCognitoSyncConfig를 전달하고 최소한 리전 세트를 포함합니다.

```
AmazonCognitoSyncConfig clientConfig = new AmazonCognitoSyncConfig { RegionEndpoint =
 REGION };
```

```
CognitoSyncManager syncManager = new CognitoSyncManager(credentials, clientConfig);
```

데이터 집합 이해

Amazon Cognito는 사용자 프로필 데이터를 데이터 세트로 구성합니다. 각 데이터 세트에는 최대 1MB의 데이터가 키 값 쌍의 양식으로 포함될 수 있습니다. 데이터 세트는 동기화할 수 있는 가장 세분화된 엔터티입니다. 데이터 세트에서 수행된 읽기 및 쓰기 작업은 동기화 메서드가 호출될 때까지 로컬 스토어에만 영향을 줍니다. Amazon Cognito는 고유한 문자열로 데이터 세트를 식별합니다. 다음과 같이 새 데이터 세트를 생성하거나 기존 데이터 세트를 열 수 있습니다.

Android

```
Dataset dataset = client.openOrCreateDataset("datasetname");
```

데이터 세트를 삭제하려면 다음과 같이 먼저 메서드를 호출하여 로컬 스토리지에서 데이터 세트를 제거한 다음 `synchronize` 메서드를 호출하여 Amazon Cognito에서 데이터 세트를 삭제합니다.

```
dataset.delete();
dataset.synchronize(syncCallback);
```

iOS - Objective-C

```
AWSCognitoDataset *dataset = [syncClient openOrCreateDataset:@"myDataSet"];
```

데이터 세트를 삭제하려면 다음과 같이 먼저 메서드를 호출하여 로컬 스토리지에서 데이터 세트를 제거한 다음 `synchronize` 메서드를 호출하여 Amazon Cognito에서 데이터 세트를 삭제합니다.

```
[dataset clear];
[dataset synchronize];
```

iOS - Swift

```
let dataset = syncClient.openOrCreateDataset("myDataSet")!
```

데이터 세트를 삭제하려면 다음과 같이 먼저 메서드를 호출하여 로컬 스토리지에서 데이터 세트를 제거한 다음 `synchronize` 메서드를 호출하여 Amazon Cognito에서 데이터 세트를 삭제합니다.

```
dataset.clear()
dataset.synchronize()
```

JavaScript

```
syncManager.openOrCreateDataset('myDatasetName', function(err, dataset) {
   // ...
});
```

Unity

```
string myValue = dataset.Get("myKey");
dataset.Put("myKey", "newValue");
```

데이터 세트에서 키를 삭제하려면 다음과 같이 Remove를 사용합니다.

```
dataset.Remove("myKey");
```

Xamarin

```
Dataset dataset = syncManager.OpenOrCreateDataset("myDatasetName");
```

데이터 세트를 삭제하려면 다음과 같이 먼저 메서드를 호출하여 로컬 스토리지에서 데이터 세트를 제거한 다음 synchronize 메서드를 호출하여 Amazon Cognito에서 데이터 세트를 삭제합니다.

```
dataset.Delete();
dataset.SynchronizeAsync();
```

데이터 집합의 데이터 읽기 및 쓰기

Amazon Cognito 데이터 세트는 키를 통해 값에 액세스할 수 있는 사전으로 작동합니다. 다음 예와 같이 데이터 세트가 사전인 것처럼 데이터 세트의 키와 값을 읽거나 추가하거나 수정할 수 있습니다.

데이터 세트에 작성된 값은 동기화 메서드를 호출할 때까지 로컬로 캐시된 데이터의 사본에만 영향을 줍니다.

Android

```
String value = dataset.get("myKey");
```

```
dataset.put("myKey", "my value");
```

iOS - Objective-C

```
[dataset setString:@"my value" forKey:@"myKey"];
NSString *value = [dataset stringForKey:@"myKey"];
```

iOS - Swift

```
dataset.setString("my value", forKey:"myKey")
let value = dataset.stringForKey("myKey")
```

JavaScript

```
dataset.get('myKey', function(err, value) {
  console.log('myRecord: ' + value);
});

dataset.put('newKey', 'newValue', function(err, record) {
  console.log(record);
});

dataset.remove('oldKey', function(err, record) {
  console.log(success);
});
```

Unity

```
string myValue = dataset.Get("myKey");
dataset.Put("myKey", "newValue");
```

Xamarin

```
//obtain a value
string myValue = dataset.Get("myKey");

// Create a record in a dataset and synchronize with the server
dataset.OnSyncSuccess += SyncSuccessCallback;
```

```
dataset.Put("myKey", "myValue");
dataset.SynchronizeAsync();

void SyncSuccessCallback(object sender, SyncSuccessEventArgs e) {
  // Your handler code here
}
```

Android

데이터 세트에서 키를 제거하려면 다음과 같이 `remove` 메서드를 사용합니다.

```
dataset.remove("myKey");
```

iOS - Objective-C

데이터 세트에서 키를 삭제하려면 다음과 같이 `removeObjectForKey`를 사용합니다.

```
[dataset removeObjectForKey:@"myKey"];
```

iOS - Swift

데이터 세트에서 키를 삭제하려면 다음과 같이 `removeObjectForKey`를 사용합니다.

```
dataset.removeObjectForKey("myKey")
```

Unity

데이터 세트에서 키를 삭제하려면 다음과 같이 `Remove`를 사용합니다.

```
dataset.Remove("myKey");
```

Xamarin

`Remove`를 사용하여 데이터 세트에서 키를 삭제할 수 있습니다.

```
dataset.Remove("myKey");
```

로컬 데이터를 동기화 스토어와 동기화

Android

synchronize 메서드는 로컬로 캐시된 데이터를 Amazon Cognito Sync 스토어에 저장된 데이터와 비교합니다. Amazon Cognito Sync 스토어에서 원격 변경 사항을 가져오고, 충돌이 발생할 경우 충돌 해결책이 호출되며, 디바이스의 업데이트된 값이 서비스로 푸시됩니다. 데이터 세트를 동기화하려면 해당 synchronize 메서드를 호출합니다.

```
dataset.synchronize(syncCallback);
```

아래 설명과 같이 synchronize 메서드는 SyncCallback 인터페이스의 구현을 받습니다.

synchronizeOnConnectivity() 메서드는 연결을 사용할 수 있는 경우 동기화하려고 시도합니다. 연결이 즉시 사용 가능하게 되면 synchronizeOnConnectivity()는 synchronize()처럼 동작합니다. 그렇지 않으면 연결 변경 사항을 모니터링하고 연결이 사용 가능하게 되면 동기화를 수행합니다. synchronizeOnConnectivity()가 여러 번 호출되면 마지막 동기화 요청만 유지되고 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 이벤트 콜백 처리 섹션을 참조하세요.

iOS - Objective-C

synchronize 메서드는 로컬로 캐시된 데이터를 Amazon Cognito Sync 스토어에 저장된 데이터와 비교합니다. Amazon Cognito Sync 스토어에서 원격 변경 사항을 가져오고, 충돌이 발생할 경우 충돌 해결책이 호출되며, 디바이스의 업데이트된 값이 서비스로 푸시됩니다. 데이터 세트를 동기화하려면 해당 synchronize 메서드를 호출합니다.

synchronize 메서드는 비동기식이며 AWSTask 객체를 반환하여 응답을 처리합니다.

```
[[dataset synchronize] continueWithBlock:^id(AWSTask *task) {
    if (task.isCancelled) {
        // Task cancelled.
    } else if (task.error) {
        // Error while executing task.
    } else {
        // Task succeeded. The data was saved in the sync store.
    }
    return nil;
}];
```

synchronizeOnConnectivity 메서드는 디바이스가 연결된 경우 동기화하려고 시도합니다. 먼저 synchronizeOnConnectivity는 연결을 확인하며, 디바이스가 온라인 상태인 경우 즉시 synchronize를 호출하고 시도와 연관된 AWSTask 객체를 반환합니다.

디바이스가 오프라인인 경우 synchronizeOnConnectivity는 1) 다음에 디바이스가 온라인 상태가 될 때 동기화를 예약하며 2) nil 결과와 함께 AWSTask를 반환합니다. 예약된 동기화는 데이터 세트 객체의 수명 주기 동안 유효합니다. 다시 연결되기 전에 앱이 종료된 경우 데이터가 동기화되지 않습니다. 예약된 동기화 중 이벤트가 발생할 때 알림을 받으려면 AWSCognito에 있는 알림의 관찰자를 추가해야 합니다.

데이터 세트 동기화 및 다른 콜백에 대한 자세한 내용은 이벤트 콜백 처리 섹션을 참조하세요.

iOS - Swift

synchronize 메서드는 로컬로 캐시된 데이터를 Amazon Cognito Sync 스토어에 저장된 데이터와 비교합니다. Amazon Cognito Sync 스토어에서 원격 변경 사항을 가져오고, 충돌이 발생할 경우 충돌 해결책이 호출되며, 디바이스의 업데이트된 값이 서비스로 푸시됩니다. 데이터 세트를 동기화하려면 해당 synchronize 메서드를 호출합니다.

synchronize 메서드는 비동기식이며 AWSTask 객체를 반환하여 응답을 처리합니다.

```
dataset.synchronize().continueWith(block: { (task) -> AnyObject? in

            if task.isCancelled {
                // Task cancelled.
            } else if task.error != nil {
                // Error while executing task
            } else {
                // Task succeeded. The data was saved in the sync store.
            }
            return task
})
```

synchronizeOnConnectivity 메서드는 디바이스가 연결된 경우 동기화하려고 시도합니다. 먼저 synchronizeOnConnectivity는 연결을 확인하며, 디바이스가 온라인 상태인 경우 즉시 synchronize를 호출하고 시도와 연관된 AWSTask 객체를 반환합니다.

디바이스가 오프라인인 경우 synchronizeOnConnectivity는 1) 다음에 디바이스가 온라인 상태가 될 때 동기화를 예약하며 2) nil 결과와 함께 AWSTask 객체를 반환합니다. 예약된 동기화는 데이터 세트 객체의 수명 주기 동안 유효합니다. 다시 연결되기 전에 앱이 종료된 경우 데이터가 동기화되지

않습니다. 예약된 동기화 중 이벤트가 발생할 때 알림을 받으려면 AWSCognito에 있는 알림의 관찰자를 추가해야 합니다.

데이터 세트 동기화 및 다른 콜백에 대한 자세한 내용은 이벤트 콜백 처리 섹션을 참조하세요.

JavaScript

synchronize 메서드는 로컬로 캐시된 데이터를 Amazon Cognito Sync 스토어에 저장된 데이터와 비교합니다. Amazon Cognito Sync 스토어에서 원격 변경 사항을 가져오고, 충돌이 발생할 경우 충돌 해결책이 호출되며, 디바이스의 업데이트된 값이 서비스로 푸시됩니다. 데이터 세트를 동기화하려면 해당 synchronize 메서드를 호출합니다.

```
dataset.synchronize();
```

데이터 세트 동기화 및 다른 콜백에 대한 자세한 내용은 이벤트 콜백 처리 섹션을 참조하세요.

Unity

synchronize 메서드는 로컬로 캐시된 데이터를 Amazon Cognito Sync 스토어에 저장된 데이터와 비교합니다. Amazon Cognito Sync 스토어에서 원격 변경 사항을 가져오고, 충돌이 발생할 경우 충돌 해결책이 호출되며, 디바이스의 업데이트된 값이 서비스로 푸시됩니다. 데이터 세트를 동기화하려면 해당 synchronize 메서드를 호출합니다.

```
dataset.Synchronize();
```

동기화는 비동기적으로 실행되며 데이터 세트에서 지정할 수 있는 여러 콜백 중 하나를 호출합니다.

데이터 세트 동기화 및 다른 콜백에 대한 자세한 내용은 이벤트 콜백 처리 섹션을 참조하세요.

Xamarin

synchronize 메서드는 로컬로 캐시된 데이터를 Amazon Cognito Sync 스토어에 저장된 데이터와 비교합니다. Amazon Cognito Sync 스토어에서 원격 변경 사항을 가져오고, 충돌이 발생할 경우 충돌 해결책이 호출되며, 디바이스의 업데이트된 값이 서비스로 푸시됩니다. 데이터 세트를 동기화하려면 해당 synchronize 메서드를 호출합니다.

```
dataset.SynchronizeAsync();
```

데이터 세트 동기화 및 다른 콜백에 대한 자세한 내용은 이벤트 콜백 처리 섹션을 참조하세요.

이벤트 콜백 처리

> **Note**
>
> Amazon Cognito Sync를 처음 사용하는 경우 AWS AppSync를 사용하세요. Amazon Cognito Sync와 마찬가지로 AWS AppSync 는 디바이스 간에 애플리케이션 데이터를 동기화하기 위한 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 Sync 개발자는 다양한 콜백을 구현하여 다양한 동기화 이벤트 및 시나리오를 처리할 수 있습니다. Android SDK의 `SyncCallback` 인터페이스는 데이터세트가 성공적으로 다운로드되면 `onSuccess()`, 예외가 발생하면 `onFailure()`, 로컬 데이터와 원격 데이터 간의 충돌을 해결하기 위한 `onConflict()` 등 데이터세트 동기화에 대한 알림을 구성합니다.

iOS SDK에서 `AWSCognitoDidStartSynchronizeNotification`과 같은 유사한 알림을 등록하고 충돌 해결을 위해 `AWSCognitoRecordConflictHandler`와 같은 처리기를 설정할 수 있습니다. JavaScript, Unity 및 Xamarin 플랫폼에는 유사한 콜백 메커니즘이 있습니다. 이러한 콜백을 구현하면 애플리케이션이 Amazon Cognito Sync를 사용할 때 발생할 수 있는 다양한 동기화 이벤트 및 시나리오를 정상적으로 처리할 수 있습니다.

Android

SyncCallback 인터페이스

`SyncCallback` 인터페이스를 구현하여 앱에서 데이터 세트 동기화에 대한 알림을 받을 수 있습니다. 그러면 앱에서는 로컬 데이터 삭제, 미인증 및 인증 프로파일 병합 및 동기화 충돌 해결에 대해 결정할 수 있습니다. 인터페이스에 필요한 다음 메서드를 구현해야 합니다.

- `onSuccess()`
- `onFailure()`
- `onConflict()`
- `onDatasetDeleted()`
- `onDatasetsMerged()`

모든 콜백을 지정하지 않으려면 `DefaultSyncCallback` 클래스를 사용할 수도 있습니다. 이 클래스는 모든 콜백에 대해 기본적으로 빈 구현을 제공합니다.

onSuccess

onSuccess() 콜백은 동기화 스토어에서 데이터 세트를 성공적으로 다운로드한 경우 트리거됩니다.

```
@Override
public void onSuccess(Dataset dataset, List<Record> newRecords) {
}
```

onFailure

동기화 중 예외가 발생할 경우 onFailure()가 호출됩니다.

```
@Override
public void onFailure(DataStorageException dse) {
}
```

onConflict

로컬 스토어와 동기화 스토어에서 동일한 키가 수정된 경우 충돌이 발생할 수 있습니다. onConflict() 메서드는 충돌 해결을 처리합니다. 이 메서드를 구현하지 않으면 Amazon Cognito Sync 클라이언트는 기본적으로 가장 최근 변경 사항을 사용합니다.

```
@Override
public boolean onConflict(Dataset dataset, final List<SyncConflict> conflicts) {
    List<Record> resolvedRecords = new ArrayList<Record>();
    for (SyncConflict conflict : conflicts) {
        /* resolved by taking remote records */
        resolvedRecords.add(conflict.resolveWithRemoteRecord());

        /* alternately take the local records */
        // resolvedRecords.add(conflict.resolveWithLocalRecord());

        /* or customer logic, say concatenate strings */
        // String newValue = conflict.getRemoteRecord().getValue()
        //     + conflict.getLocalRecord().getValue();
        // resolvedRecords.add(conflict.resolveWithValue(newValue);
    }
    dataset.resolve(resolvedRecords);

    // return true so that synchronize() is retried after conflicts are resolved
    return true;
```

```
}
```

onDatasetDeleted

데이터 세트가 삭제되면 Amazon Cognito 클라이언트는 `SyncCallback` 인터페이스를 사용하여 로컬로 캐싱된 데이터 세트의 사본도 삭제해야 하는지 여부를 확인합니다. `onDatasetDeleted()` 메서드를 구현하여 로컬 데이터를 사용하여 수행할 사항을 클라이언트 SDK에게 알립니다.

```
@Override
public boolean onDatasetDeleted(Dataset dataset, String datasetName) {
    // return true to delete the local copy of the dataset
    return true;
}
```

onDatasetMerged

이전에 연결되지 않은 두 자격 증명이 서로 연결되면 해당 데이터 세트가 모두 병합됩니다. 애플리케이션은 `onDatasetsMerged()` 메서드를 통해 병합에 대한 알림을 받습니다.

```
@Override
public boolean onDatasetsMerged(Dataset dataset, List<String> datasetNames) {
    // return false to handle Dataset merge outside the synchronization callback
    return false;
}
```

iOS - Objective-C

동기화 알림

Amazon Cognito 클라이언트는 동기화 호출 중에 여러 `NSNotification` 이벤트를 내보냅니다. 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. 에는 동기화되는 데이터 세트의 이름인 키 데이터 세트가 `userInfo` 포함되어 있습니다.

AWSCognitoDidEndSynchronizeNotification

동기화 작업이 완료(성공 등)될 때 호출됩니다. 에는 동기화되는 데이터 세트의 이름인 키 데이터 세트가 `userInfo` 포함되어 있습니다.

AWSCognitoDidFailToSynchronizeNotification

동기화 작업이 실패할 때 호출됩니다. `userInfo` 에는 동기화되는 데이터 세트의 이름인 키 데이터 세트와 실패를 유발한 오류가 포함된 키 오류가 포함됩니다.

AWSCognitoDidChangeRemoteValueNotification

로컬 변경 사항이 Amazon Cognito에 성공적으로 푸시된 경우 호출됩니다. `userInfo` 에는 동기화되는 데이터 세트의 이름인 키 데이터 세트와 푸시된 레코드 키의 NSArray가 포함된 키 키 키가 포함됩니다.

AWSCognitoDidChangeLocalValueFromRemoteNotification

동기화 작업으로 인해 로컬 값이 변경될 때 호출됩니다. `userInfo` 에는 동기화되는 데이터 세트의 이름인 키 데이터 세트와 변경된 레코드 키의 NSArray가 포함된 키 키 키가 포함됩니다.

충돌 해결 핸들러

동기화 작업 중 로컬 스토어와 동기화 스토어에서 동일한 키가 수정된 경우 충돌이 발생할 수 있습니다. 충돌 해결 핸들러를 설정하지 않은 경우 Amazon Cognito에서는 기본적으로 가장 최근 업데이트를 선택합니다.

AWSCognitoRecordConflictHandler를 구현하고 할당하면 기본 충돌 해결을 변경할 수 있습니다. AWSCognitoConflict 입력 파라미터 충돌에는 로컬로 캐시된 데이터와 동기화 스토어의 충돌 레코드 둘 모두에 대한 AWSCognitoRecord 객체가 포함됩니다. AWSCognitoConflict를 사용하면 로컬 레코드 [conflict resolveWithLocalRecord]와 원격 레코드 [conflict resolveWithRemoteRecord] 또는 새로운 값인 [conflict resolveWithValue:value]를 사용하여 충돌을 해결할 수 있습니다. 이 방법에서 nil을 반환하면 동기화가 계속되지 않으며 다음에 동기화 프로세스가 시작될 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 `AWSCognitoDatasetDeletedHandler`를 사용하여 로컬로 캐싱된 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 구현하세요.

클라이언트 수준에서 데이터 세트 삭제 핸들러를 설정할 수 있습니다.

```
client.datasetDeletedHandler = ^BOOL (NSString *datasetName) {
    // make a backup of the data if you choose
    ...
    // delete the local data (default behavior)
    return YES;
};
```

또는 데이터 세트 수준에서 해당 핸들러를 설정할 수 있습니다.

```
dataset.datasetDeletedHandler = ^BOOL (NSString *datasetName) {
    // override default and keep the local data
    return NO;
};
```

데이터 세트 병합 핸들러

이전에 연결되지 않은 두 자격 증명이 서로 연결되면 해당 데이터 세트가 모두 병합됩니다. 애플리케이션은 DatasetMergeHandler를 통해 병합에 대한 알림을 받습니다. 핸들러는 루트 데이터 세트의 이름과 루트 데이터 세트의 병합으로 표시된 데이터 세트 이름의 배열을 수신합니다.

DatasetMergeHandler가 구현되지 않은 경우 이러한 데이터 세트는 무시되지만 자격 증명의 최대 총 데이터 세트 20개에서 공간을 계속 사용합니다.

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 동기화 호출 중에 여러 NSNotification 이벤트를 내보냅니다. 표준 NSNotificationCenter를 통해 이러한 알림을 모니터링하도록 등록할 수 있습니다.

```
NSNotificationCenter.defaultCenter().addObserver(observer: self,
```

```
    selector: "myNotificationHandler",
    name:NOTIFICATION_TYPE,
    object:nil)
```

Amazon Cognito는 아래에 나열된 대로 5가지 알림 유형을 지원합니다.

AWSCognitoDidStartSynchronizeNotification

동기화 작업이 시작될 때 호출됩니다. 에는 동기화되는 데이터 세트의 이름인 키 데이터 세트가 `userInfo` 포함되어 있습니다.

AWSCognitoDidEndSynchronizeNotification

동기화 작업이 완료(성공 등)될 때 호출됩니다. 에는 동기화되는 데이터 세트의 이름인 키 데이터 세트가 `userInfo` 포함되어 있습니다.

AWSCognitoDidFailToSynchronizeNotification

동기화 작업이 실패할 때 호출됩니다. `userInfo` 에는 동기화되는 데이터 세트의 이름인 키 데이터 세트와 실패를 유발한 오류가 포함된 키 오류가 포함됩니다.

AWSCognitoDidChangeRemoteValueNotification

로컬 변경 사항이 Amazon Cognito에 성공적으로 푸시된 경우 호출됩니다. `userInfo` 에는 동기화되는 데이터 세트의 이름인 키 데이터 세트와 푸시된 레코드 키의 NSArray가 포함된 키 키 키가 포함됩니다.

AWSCognitoDidChangeLocalValueFromRemoteNotification

동기화 작업으로 인해 로컬 값이 변경될 때 호출됩니다. `userInfo` 에는 동기화되는 데이터 세트의 이름인 키 데이터 세트와 변경된 레코드 키의 NSArray가 포함된 키 키 키가 포함됩니다.

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] 또

는 새로운 값인 [conflict resolveWithValue:value]를 사용하여 충돌을 해결할 수 있습니다. 이 방법에서 nil을 반환하면 동기화가 계속되지 않으며 다음에 동기화 프로세스가 시작될 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 `AWSCognitoDatasetDeletedHandler`를 사용하여 로컬로 캐싱된 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 구현하세요.

클라이언트 수준에서 데이터 세트 삭제 핸들러를 설정할 수 있습니다.

```
client.datasetDeletedHandler = {
      (datasetName: String!) -> Bool in
      // make a backup of the data if you choose
      ...
      // delete the local data (default behaviour)
      return true
}
```

또는 데이터 세트 수준에서 해당 핸들러를 설정할 수 있습니다.

```
dataset.datasetDeletedHandler = {
```

```
      (datasetName: String!) -> Bool in
      // make a backup of the data if you choose
      ...
      // delete the local data (default behaviour)
      return true
}
```

데이터 집합 병합 핸들러

이전에 연결되지 않은 두 자격 증명이 서로 연결되면 해당 데이터 세트가 모두 병합됩니다. 애플리케이션은 DatasetMergeHandler를 통해 병합에 대한 알림을 받습니다. 핸들러는 루트 데이터 세트의 이름과 루트 데이터 세트의 병합으로 표시된 데이터 세트 이름의 배열을 수신합니다.

DatasetMergeHandler가 구현되지 않은 경우 이러한 데이터 세트는 무시되지만 자격 증명의 최대 총 데이터 세트 20개에서 공간을 계속 사용합니다.

클라이언트 수준에서 데이터 세트 병합 핸들러를 설정할 수 있습니다.

```
client.datasetMergedHandler = {
    (datasetName: String!, datasets: [AnyObject]!) -> Void in
    for nameObject in datasets {
        if let name = nameObject as? String {
            let merged = AWSCognito.defaultCognito().openOrCreateDataset(name)
            merged.clear()
            merged.synchronize()
        }
    }
}
```

또는 데이터 세트 수준에서 해당 핸들러를 설정할 수 있습니다.

```
dataset.datasetMergedHandler = {
    (datasetName: String!, datasets: [AnyObject]!) -> Void in
    for nameObject in datasets {
        if let name = nameObject as? String {
            let merged = AWSCognito.defaultCognito().openOrCreateDataset(name)
            // do something with the data if it differs from existing dataset
              ...
              // now delete it
            merged.clear()
            merged.synchronize()
        }
```

```
    }
}
```

JavaScript

동기화 콜백

데이터 세트에 대해 synchronize()를 수행하면 선택적으로 콜백을 지정하여 다음의 각 상태를 처리할 수 있습니다.

```
dataset.synchronize({

   onSuccess: function(dataset, newRecords) {
      //...
   },

   onFailure: function(err) {
      //...
   },

   onConflict: function(dataset, conflicts, callback) {
      //...
   },

   onDatasetDeleted: function(dataset, datasetName, callback) {
      //...
   },

  onDatasetMerged: function(dataset, datasetNames, callback) {
      //...
  }

});
```

onSuccess()

onSuccess() 콜백은 동기화 스토어에서 데이터 세트를 성공적으로 업데이트한 경우 트리거됩니다. 콜백을 정의하지 않으면 동기화가 자동으로 수행됩니다.

```
onSuccess: function(dataset, newRecords) {
   console.log('Successfully synchronized ' + newRecords.length + ' new records.');
}
```

onFailure()

동기화 중 예외가 발생할 경우 `onFailure()`가 호출됩니다. 콜백을 정의하지 않으면 동기화가 자동으로 실패합니다.

```
onFailure: function(err) {
   console.log('Synchronization failed.');
   console.log(err);
}
```

onConflict()

로컬 스토어와 동기화 스토어에서 동일한 키가 수정된 경우 충돌이 발생할 수 있습니다.
`onConflict()` 메서드는 충돌 해결을 처리합니다. 이 메서드를 구현하지 않으면 충돌이 있는 경우 동기화가 중단됩니다.

```
onConflict: function(dataset, conflicts, callback) {

   var resolved = [];

   for (var i=0; i<conflicts.length; i++) {

      // Take remote version.
      resolved.push(conflicts[i].resolveWithRemoteRecord());

      // Or... take local version.
      // resolved.push(conflicts[i].resolveWithLocalRecord());

      // Or... use custom logic.
      // var newValue = conflicts[i].getRemoteRecord().getValue() +
 conflicts[i].getLocalRecord().getValue();
      // resolved.push(conflicts[i].resovleWithValue(newValue);

   }

   dataset.resolve(resolved, function() {
      return callback(true);
   });

   // Or... callback false to stop the synchronization process.
   // return callback(false);
```

```
}
```

onDatasetDeleted()

데이터 세트가 삭제되면 Amazon Cognito 클라이언트는 onDatasetDeleted() 콜백을 사용하여 로컬로 캐시된 데이터 세트의 사본도 삭제해야 하는지 여부를 결정합니다. 기본적으로 데이터 세트는 삭제되지 않습니다.

```
onDatasetDeleted: function(dataset, datasetName, callback) {

   // Return true to delete the local copy of the dataset.
   // Return false to handle deleted datasets outside the synchronization callback.

   return callback(true);

}
```

onDatasetMerged()

이전에 연결되지 않은 두 자격 증명이 서로 연결되면 해당 데이터 세트가 모두 병합됩니다. 애플리케이션은 onDatasetsMerged() 콜백을 통해 병합에 대한 알림을 받습니다.

```
onDatasetMerged: function(dataset, datasetNames, callback) {

   // Return true to continue the synchronization process.
   // Return false to handle dataset merges outside the synchronization callback.

   return callback(false);

}
```

Unity

데이터 세트를 열거나 생성한 후 동기화 메서드를 사용할 때 트리거되는 다른 콜백을 설정할 수 있습니다. 데이터 세트에 콜백을 등록하는 방법은 다음과 같습니다.

```
dataset.OnSyncSuccess += this.HandleSyncSuccess;
dataset.OnSyncFailure += this.HandleSyncFailure;
dataset.OnSyncConflict = this.HandleSyncConflict;
dataset.OnDatasetMerged = this.HandleDatasetMerged;
dataset.OnDatasetDeleted = this.HandleDatasetDeleted;
```

SyncSuccess 및 SyncFailure는 = 대신 +=를 사용하므로 이에 대해 둘 이상의 콜백을 구독할 수 있습니다.

OnSyncSuccess

OnSyncSuccess 콜백은 클라우드에서 데이터 세트를 성공적으로 업데이트한 경우 트리거됩니다. 콜백을 정의하지 않으면 동기화가 자동으로 수행됩니다.

```
private void HandleSyncSuccess(object sender, SyncSuccessEvent e)
{
    // Continue with your game flow, display the loaded data, etc.
}
```

OnSyncFailure

동기화 중 예외가 발생할 경우 OnSyncFailure가 호출됩니다. 콜백을 정의하지 않으면 동기화가 자동으로 실패합니다.

```
private void HandleSyncFailure(object sender, SyncFailureEvent e)
{
    Dataset dataset = sender as Dataset;
    if (dataset.Metadata != null) {
        Debug.Log("Sync failed for dataset : " + dataset.Metadata.DatasetName);
    } else {
        Debug.Log("Sync failed");
    }
    // Handle the error
    Debug.LogException(e.Exception);
}
```

OnSyncConflict

로컬 스토어와 동기화 스토어에서 동일한 키가 수정된 경우 충돌이 발생할 수 있습니다. OnSyncConflict 콜백은 충돌 해결을 처리합니다. 이 메서드를 구현하지 않으면 충돌이 있는 경우 동기화가 중단됩니다.

```
private bool HandleSyncConflict(Dataset dataset, List < SyncConflict > conflicts)
{
  if (dataset.Metadata != null) {
    Debug.LogWarning("Sync conflict " + dataset.Metadata.DatasetName);
  } else {
```

```
    Debug.LogWarning("Sync conflict");
  }
  List < Amazon.CognitoSync.SyncManager.Record > resolvedRecords = new List <
 Amazon.CognitoSync.SyncManager.Record > ();
  foreach(SyncConflict conflictRecord in conflicts) {
    // SyncManager provides the following default conflict resolution methods:
    //      ResolveWithRemoteRecord - overwrites the local with remote records
    //      ResolveWithLocalRecord - overwrites the remote with local records
    //      ResolveWithValue - to implement your own logic
    resolvedRecords.Add(conflictRecord.ResolveWithRemoteRecord());
  }
  // resolves the conflicts in local storage
  dataset.Resolve(resolvedRecords);
  // on return true the synchronize operation continues where it left,
  //      returning false cancels the synchronize operation
  return true;
}
```

OnDatasetDeleted

데이터 세트가 삭제되면 Amazon Cognito 클라이언트는 `OnDatasetDeleted` 콜백을 사용하여 로컬로 캐시된 데이터 세트의 사본도 삭제해야 하는지 여부를 결정합니다. 기본적으로 데이터 세트는 삭제되지 않습니다.

```
private bool HandleDatasetDeleted(Dataset dataset)
  {
      Debug.Log(dataset.Metadata.DatasetName + " Dataset has been deleted");
      // Do clean up if necessary
      // returning true informs the corresponding dataset can be purged in the local
 storage and return false retains the local dataset
      return true;
  }
```

OnDatasetMerged

이전에 연결되지 않은 두 자격 증명이 서로 연결되면 해당 데이터 세트가 모두 병합됩니다. 애플리케이션은 `OnDatasetsMerged` 콜백을 통해 병합에 대한 알림을 받습니다.

```
public bool HandleDatasetMerged(Dataset localDataset, List<string> mergedDatasetNames)
{
    foreach (string name in mergedDatasetNames)
    {
```

```
        Dataset mergedDataset = syncManager.OpenOrCreateDataset(name);
        //Lambda function to delete the dataset after fetching it
        EventHandler<SyncSuccessEvent> lambda;
        lambda = (object sender, SyncSuccessEvent e) => {
            ICollection<string> existingValues = localDataset.GetAll().Values;
            ICollection<string> newValues = mergedDataset.GetAll().Values;

            //Implement your merge logic here

            mergedDataset.Delete(); //Delete the dataset locally
            mergedDataset.OnSyncSuccess -= lambda; //We don't want this callback to be
 fired again
            mergedDataset.OnSyncSuccess += (object s2, SyncSuccessEvent e2) => {
                localDataset.Synchronize(); //Continue the sync operation that was
 interrupted by the merge
            };
            mergedDataset.Synchronize(); //Synchronize it as deleted, failing to do so
 will leave us in an inconsistent state
        };
        mergedDataset.OnSyncSuccess += lambda;
        mergedDataset.Synchronize(); //Asnchronously fetch the dataset
    }

    // returning true allows the Synchronize to continue and false stops it
    return false;
}
```

Xamarin

데이터 세트를 열거나 생성한 후 동기화 메서드를 사용할 때 트리거되는 다른 콜백을 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 구현

> **Note**
>
> Amazon Cognito Sync를 처음 사용하는 경우 AWS AppSync를 사용하세요. Amazon Cognito Sync와 마찬가지로 AWS AppSync 는 디바이스 간에 애플리케이션 데이터를 동기화하기 위한 서비스입니다.
> 앱 기본 설정이나 게임 상태 같은 사용자 데이터를 동기화할 수 있습니다. 또한 이러한 기능을 더욱 확장해, 복수의 사용자가 공유 데이터를 실시간으로 동기화하고 협업할 수 있게 합니다.

Amazon Cognito는 자격 증명과 디바이스 간의 연결 관계를 자동으로 추적합니다. 푸시 동기화 기능을 사용하면 자격 증명 데이터가 변경될 경우 지정한 자격 증명의 모든 인스턴스가 통지되도록 할 수 있습니다. 특정 자격 증명에 대해 동기화 스토어 데이터가 변경될 때마다 푸시 동기화는 해당 자격 증명과 연결된 모든 디바이스가 변경 사항에 대해 알리는 자동 푸시 알림을 받도록 합니다.

> **Note**
>
> 푸시 동기화는 JavaScript, Unity 또는 Xamarin에 대해 지원되지 않습니다.

먼저 푸시 동기화에 대한 계정을 설정하고 Amazon Cognito 콘솔에서 푸시 동기화를 활성화해야 푸시 동기화를 사용할 수 있습니다.

Amazon Simple Notification Service(Amazon SNS) 앱 생성

SNS 개발자 가이드의 설명에 따라, 지원되는 플랫폼에 대해 Amazon SNS 앱을 생성하고 구성합니다.

Amazon Cognito 콘솔에서 푸시 동기화 사용

Amazon Cognito 콘솔을 통해 푸시 동기화를 활성화할 수 있습니다. 콘솔 홈 페이지에서 다음을 수행합니다.

1. 푸시 동기화를 활성화할 자격 증명 풀 이름을 클릭합니다. 자격 증명 풀에 대한 대시보드 페이지가 표시됩니다.

2. 대시보드(Dashboard) 페이지의 우측 상단 모서리에서 자격 증명 풀 관리(Manage Identity Pools)를 클릭합니다. 연동 자격 증명 페이지가 나타납니다.
3. 아래로 스크롤하고 푸시 동기화를 클릭하여 확장합니다.
4. 서비스 역할 드롭다운 메뉴에서 SNS 알림을 전송할 권한을 Cognito에 부여하는 IAM 역할을 선택합니다. AWS IAM 콘솔에서 [역할 생성(Create role)]을 클릭하여 자격 증명 풀과 연결된 역할을 생성하거나 수정합니다.
5. 플랫폼 애플리케이션을 선택한 다음 변경 사항 저장을 클릭합니다.
6. 애플리케이션에 SNS 액세스 권한 부여

AWS Identity and Access Management 콘솔에서 전체 Amazon SNS 액세스 권한을 갖도록 IAM 역할을 구성하거나 전체 Amazon SNS 액세스 권한이 있는 새 역할을 생성합니다. 다음 역할 트러스트 정책 예제는 Amazon Cognito Sync에 IAM 역할을 수임할 수 있는 제한적인 권한을 부여합니다. Amazon Cognito Sync는 `aws:SourceArn` 조건의 자격 증명 풀과 `aws:SourceAccount` 조건의 계정을 둘 다 대신하는 경우에만 역할을 수임할 수 있습니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "cognito-sync.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "AWS:SourceAccount": "123456789012"
                },
                "ArnLike": {
                    "AWS:SourceArn": "arn:aws:cognito-identity:us-
east-1:123456789012:identitypool/us-east-1:177a950c-2c08-43f0-9983-28727EXAMPLE"
                }
            }
        }
    ]
}
```

IAM 역할에 대한 자세한 내용은 역할(위임 및 연동)을 참조하세요.

앱에서 푸시 동기화 사용: Android

애플리케이션에서 Google Play 서비스를 가져와야 합니다. Android SDK Manager를 통해 최신 버전의 Google Play SDK를 다운로드할 수 있습니다. Android 구현의 Android 설명서에 따라 앱을 등록하고 GCM으로부터 등록 ID를 받습니다. 등록 ID가 있는 경우 아래 조각에 표시된 대로 Amazon Cognito를 사용하여 디바이스를 등록해야 합니다.

```
String registrationId = "MY_GCM_REGISTRATION_ID";
try {
    client.registerDevice("GCM", registrationId);
} catch (RegistrationFailedException rfe) {
    Log.e(TAG, "Failed to register device for silent sync", rfe);
} catch (AmazonClientException ace) {
    Log.e(TAG, "An unknown error caused registration for silent sync to fail", ace);
}
```

이제 디바이스를 구독하여 특정 데이터 세트에서 업데이트를 받을 수 있습니다.

```
Dataset trackedDataset = client.openOrCreateDataset("myDataset");
if (client.isDeviceRegistered()) {
    try {
        trackedDataset.subscribe();
    } catch (SubscribeFailedException sfe) {
        Log.e(TAG, "Failed to subscribe to datasets", sfe);
    } catch (AmazonClientException ace) {
        Log.e(TAG, "An unknown error caused the subscription to fail", ace);
    }
}
```

데이터 세트에서 푸시 알림을 받는 것을 중지하려면 unsubscribe 메서드를 호출하면 됩니다. `CognitoSyncManager` 객체에서 모든 데이터 세트(또는 특정 하위 집합)를 구독하려면 `subscribeAll()`을 사용하세요.

```
if (client.isDeviceRegistered()) {
    try {
        client.subscribeAll();
    } catch (SubscribeFailedException sfe) {
        Log.e(TAG, "Failed to subscribe to datasets", sfe);
    } catch (AmazonClientException ace) {
        Log.e(TAG, "An unknown error caused the subscription to fail", ace);
```

```
    }
}
```

Android BroadcastReceiver 객체의 구현에서 수정된 데이터 세트의 최신 버전을 확인하고 앱에서 다시 동기화해야 하는지 여부를 결정할 수 있습니다.

```
@Override
public void onReceive(Context context, Intent intent) {

    PushSyncUpdate update = client.getPushSyncUpdate(intent);

    // The update has the source (cognito-sync here), identityId of the
    // user, identityPoolId in question, the non-local sync count of the
    // data set and the name of the dataset. All are accessible through
    // relevant getters.

    String source = update.getSource();
    String identityPoolId = update.getIdentityPoolId();
    String identityId = update.getIdentityId();
    String datasetName = update.getDatasetName();
    long syncCount = update.getSyncCount();

    Dataset dataset = client.openOrCreateDataset(datasetName);

    // need to access last sync count. If sync count is less or equal to
    // last sync count of the dataset, no sync is required.

    long lastSyncCount = dataset.getLastSyncCount();
    if (lastSyncCount < syncCount) {
        dataset.synchronize(new SyncCallback() {
            // ...
        });
    }

}
```

다음 키는 푸시 알림 페이로드에서 사용할 수 있습니다.

- `source`: cognito-sync. 이 키는 알림 간 차별화 요소의 역할을 수행할 수 있습니다.
- `identityPoolId`: 자격 증명 풀 ID입니다. 수신자의 관점에서 필수가 아닌 경우에도 확인 또는 추가 정보를 위해 이 키를 사용할 수 있습니다.
- `identityId`: 풀 내의 자격 증명 ID입니다.

- `datasetName`: 업데이트된 데이터 세트의 이름입니다. openOrCreateDataset 호출을 위해 이 키를 사용할 수 있습니다.
- `syncCount`: 원격 데이터 세트에 대한 동기화 수입니다. 로컬 데이터 세트가 최신이 아니며 수신되는 동기화가 최신임을 확인하는 방법으로 이 키를 사용할 수 있습니다.

앱에서 푸시 동기화 사용: iOS - Objective-C

앱에 대한 디바이스 토큰을 얻으려면 원격 알림 등록의 Apple 설명서를 따르세요. APN에서 NSData 객체로 디바이스 토큰을 받은 경우 아래에 표시된 대로 동기화 클라이언트의 `registerDevice:` 메서드를 사용하여 디바이스를 Amazon Cognito에 등록해야 합니다.

```
AWSCognito *syncClient = [AWSCognito defaultCognito];
    [[syncClient registerDevice: devToken] continueWithBlock:^id(AWSTask *task) {
        if(task.error){
            NSLog(@"Unable to registerDevice: %@", task.error);
        } else {
            NSLog(@"Successfully registered device with id: %@", task.result);
        }
        return nil;
      }
    ];
```

디버그 모드에서는 디바이스가 APNs샌드박스에 등록되고, 릴리스 모드에서는 APNs. 특정 데이터 세트에서 업데이트를 받으려면 `subscribe` 메서드를 사용하세요.

```
[[[syncClient openOrCreateDataset:@"MyDataset"] subscribe]
 continueWithBlock:^id(AWSTask *task) {
        if(task.error){
            NSLog(@"Unable to subscribe to dataset: %@", task.error);
        } else {
            NSLog(@"Successfully subscribed to dataset: %@", task.result);
        }
        return nil;
      }
    ];
```

데이터 세트에서 푸시 알림을 받는 것을 중지하려면 `unsubscribe` 메서드를 호출하면 됩니다.

```
[[[syncClient openOrCreateDataset:@”MyDataset”] unsubscribe]
 continueWithBlock:^id(AWSTask *task) {
```

```
        if(task.error){
            NSLog(@"Unable to unsubscribe from dataset: %@", task.error);
        } else {
            NSLog(@"Successfully unsubscribed from dataset: %@", task.result);
        }
        return nil;
      }
    ];
```

AWSCognito 객체에서 모든 데이터 세트를 구독하려면 subscribeAll을 호출하세요.

```
[[syncClient subscribeAll] continueWithBlock:^id(AWSTask *task) {
        if(task.error){
            NSLog(@"Unable to subscribe to all datasets: %@", task.error);
        } else {
            NSLog(@"Successfully subscribed to all datasets: %@", task.result);
        }
        return nil;
      }
    ];
```

subscribeAll을 호출하기 전에 각 데이터 세트에 대해 한 번 이상 동기화하여 데이터 세트가 서버에 존재하도록 해야 합니다.

푸시 알림에 대응하려면 앱 위임에 didReceiveRemoteNotification 메서드를 구현해야 합니다.

```
- (void)application:(UIApplication *)application didReceiveRemoteNotification:
(NSDictionary *)userInfo
    {
        [[NSNotificationCenter defaultCenter]
 postNotificationName:@"CognitoPushNotification" object:userInfo];
    }
```

알림 핸들러를 사용하여 알림을 게시한 경우 데이터 세트를 처리하는 애플리케이션의 다른 곳에서 알림에 대응할 수 있습니다. 다음과 같이 알림을 구독할 경우

```
[[NSNotificationCenter defaultCenter] addObserver:self
 selector:@selector(didReceivePushSync:)
    name: :@"CognitoPushNotification" object:nil];
```

알림에 대해 다음과 같이 작동합니다.

```
- (void)didReceivePushSync:(NSNotification*)notification
    {
        NSDictionary * data = [(NSDictionary *)[notification object]
 objectForKey:@"data"];
        NSString * identityId = [data objectForKey:@"identityId"];
        NSString * datasetName = [data objectForKey:@"datasetName"];
        if([self.dataset.name isEqualToString:datasetName] && [self.identityId
 isEqualToString:identityId]){
            [[self.dataset synchronize] continueWithBlock:^id(AWSTask *task) {
                if(!task.error){
                    NSLog(@"Successfully synced dataset");
                }
                return nil;
            }];
        }
    }
```

다음 키는 푸시 알림 페이로드에서 사용할 수 있습니다.

- `source`: cognito-sync. 이 키는 알림 간 차별화 요소의 역할을 수행할 수 있습니다.
- `identityPoolId`: 자격 증명 풀 ID입니다. 수신자의 관점에서 필수가 아닌 경우에도 확인 또는 추가 정보를 위해 이 키를 사용할 수 있습니다.
- `identityId`: 풀 내의 자격 증명 ID입니다.
- `datasetName`: 업데이트된 데이터 세트의 이름입니다. `openOrCreateDataset` 호출을 위해 이 키를 사용할 수 있습니다.
- `syncCount`: 원격 데이터 세트에 대한 동기화 수입니다. 로컬 데이터 세트가 최신이 아니며 수신되는 동기화가 최신임을 확인하는 방법으로 이 키를 사용할 수 있습니다.

앱에서 푸시 동기화 사용: iOS - Swift

앱에 대한 디바이스 토큰을 얻으려면 원격 알림 등록의 Apple 설명서를 따르세요. APN에서 NSData 객체로 디바이스 토큰을 받은 경우 아래에 표시된 대로 동기화 클라이언트의 registerDevice: 메서드를 사용하여 디바이스를 Amazon Cognito에 등록해야 합니다.

```
let syncClient = AWSCognito.default()
syncClient.registerDevice(devToken).continueWith(block: { (task: AWSTask!) ->
 AnyObject! in
    if (task.error != nil) {
        print("Unable to register device: " + task.error.localizedDescription)
```

```
    } else {
        print("Successfully registered device with id: \(task.result)")
    }
    return task
})
```

디버그 모드에서는 디바이스가 APNs샌드박스에 등록되고, 릴리스 모드에서는 APNs. 특정 데이터 세트에서 업데이트를 받으려면 subscribe 메서드를 사용하세요.

```
syncClient.openOrCreateDataset("MyDataset").subscribe().continueWith(block: { (task:
 AWSTask!) -> AnyObject! in
  if (task.error != nil) {
      print("Unable to subscribe to dataset: " + task.error.localizedDescription)

  } else {
      print("Successfully subscribed to dataset: \(task.result)")
  }
  return task
})
```

데이터 세트에서 푸시 알림을 받는 것을 중지하려면 unsubscribe 메서드를 호출하세요.

```
syncClient.openOrCreateDataset("MyDataset").unsubscribe().continueWith(block: { (task:
 AWSTask!) -> AnyObject! in
  if (task.error != nil) {
      print("Unable to unsubscribe to dataset: " + task.error.localizedDescription)

  } else {
      print("Successfully unsubscribed to dataset: \(task.result)")
  }
  return task
})
```

AWSCognito 객체에서 모든 데이터 세트를 구독하려면 subscribeAll을 호출하세요.

```
syncClient.openOrCreateDataset("MyDataset").subscribeAll().continueWith(block: { (task:
 AWSTask!) -> AnyObject! in
  if (task.error != nil) {
      print("Unable to subscribe to all datasets: " + task.error.localizedDescription)
```

```
  } else {
      print("Successfully subscribed to all datasets: \(task.result)")
  }
  return task
})
```

subscribeAll을 호출하기 전에 각 데이터 세트에 대해 한 번 이상 동기화하여 데이터 세트가 서버에 존재하도록 해야 합니다.

푸시 알림에 대응하려면 앱 위임에 didReceiveRemoteNotification 메서드를 구현해야 합니다.

```
func application(application: UIApplication, didReceiveRemoteNotification userInfo:
 [NSObject : AnyObject],
  fetchCompletionHandler completionHandler: (UIBackgroundFetchResult) -> Void) {

 NSNotificationCenter.defaultCenter().postNotificationName("CognitoPushNotification",
 object: userInfo)
})
```

알림 핸들러를 사용하여 알림을 게시한 경우 데이터 세트를 처리하는 애플리케이션의 다른 곳에서 알림에 대응할 수 있습니다. 다음과 같이 알림을 구독할 경우

```
NSNotificationCenter.defaultCenter().addObserver(observer:self,
   selector:"didReceivePushSync:",
   name:"CognitoPushNotification",
   object:nil)
```

알림에 대해 다음과 같이 작동합니다.

```
func didReceivePushSync(notification: NSNotification) {
    if let data = (notification.object as! [String: AnyObject])["data"] as? [String:
 AnyObject] {
        let identityId = data["identityId"] as! String
        let datasetName = data["datasetName"] as! String

        if self.dataset.name == datasetName && self.identityId == identityId {
          dataset.synchronize().continueWithBlock {(task) -> AnyObject! in
              if task.error == nil {
                print("Successfully synced dataset")
              }
              return nil
```

```
            }
        }
    }
}
```

다음 키는 푸시 알림 페이로드에서 사용할 수 있습니다.

- `source`: cognito-sync. 이 키는 알림 간 차별화 요소의 역할을 수행할 수 있습니다.
- `identityPoolId`: 자격 증명 풀 ID입니다. 수신자의 관점에서 필수가 아닌 경우에도 확인 또는 추가 정보를 위해 이 키를 사용할 수 있습니다.
- `identityId`: 풀 내의 자격 증명 ID입니다.
- `datasetName`: 업데이트된 데이터 세트의 이름입니다. `openOrCreateDataset` 호출을 위해 이 키를 사용할 수 있습니다.
- `syncCount`: 원격 데이터 세트에 대한 동기화 수입니다. 로컬 데이터 세트가 최신이 아니며 수신되는 동기화가 최신임을 확인하는 방법으로 이 키를 사용할 수 있습니다.

Amazon Cognito Sync 스트림 구현

> **Note**
>
> Amazon Cognito Sync를 처음 사용하는 경우 AWS AppSync를 사용하세요. Amazon Cognito Sync와 마찬가지로 AWS AppSync 는 디바이스 간에 애플리케이션 데이터를 동기화하기 위한 서비스입니다.
> 앱 기본 설정이나 게임 상태 같은 사용자 데이터를 동기화할 수 있습니다. 또한 이러한 기능을 더욱 확장해, 복수의 사용자가 공유 데이터를 실시간으로 동기화하고 협업할 수 있게 합니다.

Amazon Cognito 스트림은 개발자에게 Amazon Cognito에 저장된 데이터에 대한 제어와 통찰력을 제공합니다. 개발자는 이제 데이터가 업데이트 및 동기화될 때 이벤트를 수신하도록 Kinesis 스트림을 구성할 수 있습니다. Amazon Cognito는 각 데이터 집합 변경 사항을 사용자가 소유하는 Kinesis 스트림으로 실시간으로 푸시합니다.

Amazon Cognito 스트림을 사용하여 모든 동기화 데이터를 Kinesis로 이동할 수 있습니다. 그러면 추가 분석을 위한 Amazon Redshift와 같이 데이터 웨어하우스 도구로 스트리밍될 수 있습니다. Kinesis에 대한 자세한 내용은 Amazon Kinesis 사용 시작하기를 참조하세요.

스트림 구성

Amazon Cognito 스트림은 Amazon Cognito 콘솔에서 설정할 수 있습니다. Amazon Cognito 콘솔에서 Amazon Cognito 스트림을 사용하려면 게시할 Kinesis 스트림과 선택한 스트림에 이벤트를 게시할 Amazon Cognito 권한을 부여하는 IAM 역할을 선택해야 합니다.

콘솔 홈 페이지에서 다음을 수행합니다.

1. Amazon Cognito 스트림을 설정할 자격 증명 풀 이름을 클릭합니다. 자격 증명 풀에 대한 대시보드 페이지가 표시됩니다.
2. 대시보드(Dashboard) 페이지의 우측 상단 모서리에서 자격 증명 풀 관리(Manage Identity Pools)를 클릭합니다. 연동 자격 증명 관리(Manage Federated Identities) 페이지가 나타납니다.
3. 아래로 스크롤하고 Cognito 스트림을 클릭하여 확장합니다.
4. 스트림 이름 드롭다운 메뉴에서 기존 Kinesis 스트림의 이름을 선택합니다. 또는 스트림 생성을 클릭하여 스트림 하나를 생성하고 스트림 이름과 샤드 수를 입력합니다. 샤드에 대한 자세한 내용과 스트림에 필요한 샤드의 수 예상에 대한 도움말은 Kinesis 개발자 가이드를 참조하세요.
5. [게시 역할(Publish role)] 드롭다운 메뉴에서 스트림을 게시할 Amazon Cognito 권한을 부여하는 IAM 역할을 선택합니다. AWS IAM 콘솔에서 [역할 생성(Create role)]을 클릭하여 자격 증명 풀과 연결된 역할을 생성하거나 수정합니다.
6. 스트림 상태 드롭다운 메뉴에서 활성을 선택하여 스트림 업데이트를 활성화합니다. 변경 사항 저장을 클릭합니다.

Amazon Cognito 스트림을 성공적으로 구성한 후이 자격 증명 풀의 데이터 세트에 대한 모든 후속 업데이트가 스트림으로 전송됩니다.

스트림 콘텐츠

스트림으로 전송된 각 레코드는 단일 동기화를 나타냅니다. 다음은 스트림으로 전송된 레코드의 예제입니다.

```
{
    "identityPoolId": "Pool Id",
    "identityId": "Identity Id",
    "dataSetName": "Dataset Name",
    "operation": "(replace|remove)",
    "kinesisSyncRecords": [
        {
            "key": "Key",
            "value": "Value",
            "syncCount": 1,
```

```
            "lastModifiedDate": 1424801824343,
            "deviceLastModifiedDate": 1424801824343,
            "op": "(replace|remove)"
        },
        ...
    ],
    "lastModifiedDate": 1424801824343,
    "kinesisSyncRecordsURL": "S3Url",
    "payloadType": "(S3Url|Inline)",
    "syncCount": 1
}
```

Kinesis 최대 페이로드 크기인 1MB보다 큰 업데이트의 경우 Amazon Cognito는 업데이트의 전체 콘텐츠가 포함된 미리 서명된 Amazon S3 URL을 포함합니다.

Amazon Cognito 스트림을 구성한 후 Kinesis 스트림을 삭제하거나 Amazon Cognito Sync가 더 이상 역할을 맡을 수 없도록 역할 신뢰 권한을 변경하는 경우 Amazon Cognito 스트림을 해제합니다. Kinesis 스트림을 다시 생성하거나 역할을 수정한 다음 스트림을 다시 설정해야 합니다.

대량 게시

Amazon Cognito 스트림을 구성한 경우 자격 증명 풀의 기존 데이터에 대해 대량 게시 작업을 실행할 수 있습니다. 콘솔을 통해 또는 API를 통해 직접 대량 게시 작업을 시작하면 Amazon Cognito는 업데이트를 수신하는 것과 동일한 스트림에이 데이터를 게시하기 시작합니다.

Amazon Cognito는 대량 게시 작업을 사용할 때 스트림에 전송된 데이터의 고유성을 보장하지 않습니다. 업데이트 및 대량 게시의 일부로 동일한 업데이트를 받을 수 있습니다. 스트림에서 레코드를 처리할 때 이 점을 유의하시기 바랍니다.

모든 스트림을 대량으로 게시하려면 스트림 구성의 1-6단계를 따른 다음 대량 게시 시작(Start bulk publish)을 클릭합니다. 대량 게시 작업은 지정된 시간에 한 번으로 제한되며 대량 게시 요청은 24시간마다 한 번으로 제한됩니다.

Amazon Cognito Events를 사용하여 워크플로 사용자 지정

> **Note**
>
> Amazon Cognito Sync를 처음 사용하는 경우 AWS AppSync를 사용하세요. Amazon Cognito Sync와 마찬가지로 AWS AppSync 는 디바이스 간에 애플리케이션 데이터를 동기화하기 위한 서비스입니다.

앱 기본 설정이나 게임 상태 같은 사용자 데이터를 동기화할 수 있습니다. 또한 이러한 기능을 더욱 확장해, 복수의 사용자가 공유 데이터를 실시간으로 동기화하고 협업할 수 있게 합니다.

Amazon Cognito 이벤트를 사용하면 Amazon Cognito의 중요한 이벤트에 대한 응답으로 AWS Lambda 함수를 실행할 수 있습니다. Amazon Cognito는 데이터 집합이 동기화될 경우 동기화 트리거 이벤트를 유발합니다. 사용자가 데이터를 업데이트할 때 동기화 트리거 이벤트를 사용하여 작업을 수행할 수 있습니다. 이 함수는 클라우드에 저장되고 사용자의 다른 디바이스에 동기화되기 전에 데이터를 평가하고 선택적으로 조작합니다. 사용자의 다른 디바이스에 동기화되기 전에 디바이스에서 제공된 데이터를 검증하거나 플레이어가 새 레벨에 도달할 때 상 제공 등 수신 데이터에 따라 데이터 세트의 다른 값을 업데이트하는 데 유용합니다.

아래 단계는 Amazon Cognito 데이터 세트가 동기화될 때마다 실행되는 Lambda 함수를 설정하는 방법을 안내합니다.

> **Note**
>
> Amazon Cognito 이벤트를 사용할 때는 Amazon Cognito Identity에서 받은 자격 증명만 사용할 수 있습니다. 연결된 Lambda 함수가 있지만 AWS 계정 자격 증명(개발자 자격 증명)`UpdateRecords`으로를 호출하면 Lambda 함수가 호출되지 않습니다.

에서 함수 생성 AWS Lambda

Amazon Cognito와 Lambda를 통합하려면 먼저 Lambda에서 함수를 생성해야 합니다. 그렇게 하려면 다음을 수행하세요.

Amazon Cognito에서 Lambda 함수 선택

1. Lambda 콘솔을 엽니다.
2. [Lambda 함수 생성(Create a Lambda function)]을 클릭합니다.
3. 블루프린트 선택(Select blueprint) 화면에서 "cognito-sync-trigger"를 검색하고 선택합니다.
4. 이벤트 소스 구성(Configure event sources) 화면에서 이벤트 소스 유형을 "Cognito Sync 트리거"로 설정해 두고 자격 증명 풀을 선택합니다. 다음을 클릭합니다.

   > **Note**
   >
   > 콘솔 외부에서 Amazon Cognito Sync 트리거를 구성하는 경우 Amazon Cognito가 함수를 호출할 수 있도록 Lambda 리소스 기반 권한을 추가해야 합니다. Lambda 콘솔(에 대한 리소

스 기반 정책 사용 AWS Lambda 참조)에서 또는 Lambda AddPermission 작업을 사용하여 이 권한을 추가할 수 있습니다.

Lambda 리소스 기반 정책 예제

다음 AWS Lambda 리소스 기반 정책은 Amazon Cognito에 Lambda 함수를 호출할 수 있는 제한적인 권한을 부여합니다. Amazon Cognito는 `aws:SourceArn` 조건의 자격 증명 풀과 `aws:SourceAccount` 조건의 계정을 대신하는 경우에만 함수를 호출할 수 있습니다.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "default",
    "Statement": [
        {
            "Sid": "lambda-allow-cognito-my-function",
            "Effect": "Allow",
            "Principal": {
                "Service": "cognito-sync.amazonaws.com"
            },
            "Action": "lambda:InvokeFunction",
            "Resource": "arn:aws:lambda:us-
east-1:111122223333:function:MyFunction",
            "Condition": {
                "StringEquals": {
                    "AWS:SourceAccount": "111122223333"
                },
                "ArnLike": {
                    "AWS:SourceArn": "arn:aws:cognito-
identity:us-east-1:111122223333:identitypool/us-
east-1:abcdefg-1234-5678-910a-0e8443553f95"
                }
            }
        }
    ]
}
```

5. 함수 구성(Configure function) 화면에서 함수 이름과 설명을 입력합니다. 런타임(Runtime)을 "Node.js"로 설정합니다. 이 예제에서는 코드를 변경하지 않고 그대로 둡니다. 기본 예제는 동기화되는 데이터를 변경하지 않으며 Amazon Cognito Sync 트리거 이벤트가 발생했다는 사실만 기록합니다. 핸들러 이름을 "index.handler"로 설정해 둡니다. 역할의 경우 AWS Lambda에 액세스할

코드 권한을 부여하는 IAM 역할을 선택합니다. 역할을 수정하려면 IAM 콘솔을 참조하세요. 고급(Advanced) 설정을 변경하지 않고 그대로 둡니다. 다음을 클릭합니다.

6. 검토(Review) 화면에서 세부 정보를 검토하고 함수 생성(Create function)을 클릭합니다. 다음 페이지에는 새 Lambda 함수가 표시됩니다.

Lambda에 작성된 적절한 함수가 있는 경우 해당 함수를 Amazon Cognito Sync 트리거 이벤트에 대한 핸들러로 선택해야 합니다. 아래 단계는 이 프로세스를 소개합니다.

콘솔 홈 페이지에서 다음을 수행합니다.

1. Amazon Cognito 이벤트를 설정할 자격 증명 풀 이름을 클릭합니다. 자격 증명 풀에 대한 대시보드 페이지가 표시됩니다.
2. 대시보드(Dashboard) 페이지의 우측 상단 모서리에서 연동 자격 증명 관리(Manage Federated Identities)를 클릭합니다. 연동 자격 증명 관리(Manage Federated Identities) 페이지가 나타납니다.
3. 아래로 스크롤하고 Cognito 이벤트(Cognito Events)를 클릭하여 확장합니다.
4. 동기화 트리거 드롭다운 메뉴에서 동기화 이벤트가 발생할 때 트리거할 Lambda 함수를 선택합니다.
5. 변경 사항 저장을 클릭합니다.

이제 데이터 세트가 동기화될 때마다 Lambda 함수가 실행됩니다. 다음 섹션에서는 함수에서 데이터가 동기화될 때 이 데이터를 읽고 수정하는 방법에 대해 설명합니다.

동기화 트리거에 대한 Lambda 함수 작성

동기화 트리거는 서비스 공급자 인터페이스에서 사용하는 프로그래밍 패턴을 따릅니다. Amazon Cognito는 Lambda 함수에 입력을 다음 JSON 형식으로 제공합니다.

```
{
  "version": 2,
  "eventType": "SyncTrigger",
  "region": "us-east-1",
  "identityPoolId": "identityPoolId",
  "identityId": "identityId",
  "datasetName": "datasetName",
  "datasetRecords": {
    "SampleKey1": {
      "oldValue": "oldValue1",
      "newValue": "newValue1",
```

```
      "op": "replace"
    },
    "SampleKey2": {
      "oldValue": "oldValue2",
      "newValue": "newValue2",
      "op": "replace"
    },..
  }
}
```

Amazon Cognito는 입력과 동일한 형식으로 함수의 반환 값을 예상합니다.

동기화 트리거 이벤트에 대한 함수를 작성할 때 다음 사항을 준수합니다.

- Amazon Cognito가 UpdateRecords 동안 Lambda 함수를 호출하면 함수는 5초 이내에 응답해야 합니다. 그렇지 않으면 Amazon Cognito Sync 서비스에서 `LambdaSocketTimeoutException` 예외가 발생합니다. 이 제한 시간 값은 늘릴 수 없습니다.
- `LambdaThrottledException` 예외가 발생하면 동기화 작업을 다시 시도하여 레코드를 업데이트하세요.
- Amazon Cognito는 데이터 세트에 있는 모든 레코드를 함수에 대한 입력으로 제공합니다.
- 앱 사용자가 업데이트하는 레코드에 `replace`로 설정된 op 필드가 있습니다. 삭제된 레코드에 `remove`로 설정된 op 필드가 있습니다.
- 앱 사용자가 레코드를 업데이트하지 않은 경우에도 레코드를 수정할 수 있습니다.
- datasetRecords를 제외한 모든 필드는 읽기 전용이며 변경할 수 없습니다. 이러한 필드를 변경하면 레코드를 업데이트할 수 없습니다.
- 레코드 값을 수정하려면 이 값을 업데이트하고 op를 `replace`로 업데이트합니다.
- 레코드를 제거하려면 op를 `remove`로 설정하거나 이 값을 null로 설정합니다.
- 레코드를 추가하려면 datasetRecords 배열에 새 레코드를 추가합니다.
- Amazon Cognito는 Amazon이 레코드를 업데이트할 때 응답에서 생략된 레코드를 무시합니다.

샘플 Lambda 함수

다음은 데이터에 액세스하고 데이터를 수정 및 제거하는 방법을 보여주는 Lambda 함수의 예입니다.

```
console.log('Loading function');

exports.handler = function(event, context) {
    console.log(JSON.stringify(event, null, 2));
```

```
    //Check for the event type
    if (event.eventType === 'SyncTrigger') {

        //Modify value for a key
        if('SampleKey1' in event.datasetRecords){
            event.datasetRecords.SampleKey1.newValue = 'ModifyValue1';
            event.datasetRecords.SampleKey1.op = 'replace';
        }

        //Remove a key
        if('SampleKey2' in event.datasetRecords){
            event.datasetRecords.SampleKey2.op = 'remove';
        }

        //Add a key
        if(!('SampleKey3' in event.datasetRecords)){
            event.datasetRecords.SampleKey3={'newValue':'ModifyValue3', 'op' :
 'replace'};
        }

    }
    context.done(null, event);
};
```

Amazon Cognito Sync를 자격 증명 풀과 함께 사용

Amazon Cognito Sync는 디바이스 간에 애플리케이션 관련 사용자 데이터를 동기화할 수 있는 AWS 서비스 및 클라이언트 라이브러리입니다. Amazon Cognito Sync는 자체 백엔드를 사용하지 않고도 모바일 디바이스와 웹 간에 사용자 프로필 데이터를 동기화할 수 있습니다. 클라이언트 라이브러리는 디바이스 연결 상태와 관계없이 앱에서 데이터를 읽고 쓸 수 있도록 로컬로 데이터를 캐싱합니다. 디바이스가 온라인 상태일 때 데이터를 동기화할 수 있습니다. 푸시 동기화를 설정한 경우 업데이트가 있음을 다른 디바이스에 즉시 알릴 수 있습니다.

데이터 집합 관리

애플리케이션에서 Amazon Cognito Sync 기능을 구현한 경우 Amazon Cognito 자격 증명 풀 콘솔에서는 개별 자격 증명에 대한 데이터 집합과 레코드를 수동으로 생성하고 삭제할 수 있습니다. Amazon Cognito 자격 증명 풀 콘솔에서 자격 증명의 데이터 집합 또는 레코드에 대한 변경 사항은 사용자가 콘솔에서 동기화(Synchronize)를 선택할 때까지 저장되지 않습니다. 자격 증명이 동기화(Synchronize)를 호출할 때까지 최종 사용자에게 변경 사항이 표시되지 않습니다. 개별 자격 증명에 대해 다른 장치에서

동기화되고 있는 데이터는 특정 자격 증명에 대한 목록 데이터 집합 페이지를 새로 고칠 경우 표시됩니다.

자격 증명에 대한 데이터 집합 생성

Amazon Cognito Sync는 데이터 세트를 하나의 자격 증명과 연결합니다. 자격 증명이 나타내는 사용자에 대한 식별 정보로 데이터 세트를 채운 다음 해당 정보를 사용자의 모든 디바이스에 동기화할 수 있습니다.

자격 증명에 데이터 세트와 데이터 세트 레코드를 추가하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 자격 증명 브라우저 탭을 선택합니다.
3. 편집할 자격 증명을 선택합니다.
4. 데이터 세트에서 데이터 세트 생성을 선택합니다.
5. 데이터 세트 이름을 입력하고 데이터 세트 생성을 선택합니다.
6. 데이터 세트에 레코드를 추가하려면 자격 증명 세부 정보에서 데이터 세트를 선택합니다. 레코드에서 레코드 생성을 선택합니다.
7. 레코드의 키와 값을 입력합니다. 확인을 선택합니다. 레코드를 더 추가하려면 이 단계를 반복합니다.

자격 증명과 연결된 데이터 집합 삭제

자격 증명에서 데이터 세트와 해당 레코드를 삭제하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 자격 증명 브라우저 탭을 선택합니다.
3. 삭제할 데이터 세트가 포함된 자격 증명을 선택합니다.
4. 데이터 세트에서 삭제할 데이터 세트 옆에 있는 라디오 버튼을 선택합니다.
5. 삭제를 선택합니다. 선택 사항을 검토하고 삭제를 다시 선택합니다.

데이터 대량 게시

Amazon Cognito Sync 스토어에 이미 저장되어 있는 데이터를 Amazon Kinesis 스트림으로 내보내는 데 대량 게시를 사용할 수 있습니다. 모든 스트림을 대량 게시하는 방법에 대한 자세한 내용은 Amazon Cognito Sync 스트림 구현 섹션을 참조하세요.

푸시 동기화 활성화

Amazon Cognito는 자격 증명과 디바이스 간의 연결 관계를 자동으로 추적합니다. 푸시 동기화 기능을 사용하면 자격 증명 데이터가 변경될 경우 지정한 자격 증명의 모든 인스턴스가 통지되도록 할 수 있습니다. 푸시 동기화는 자격 증명의 데이터 세트가 변경될 때마다 해당 자격 증명과 연결된 모든 디바이스가 변경 사항에 대해 알리는 자동 푸시 알림을 받도록 합니다.

Amazon Cognito 콘솔에서 푸시 동기화를 활성화할 수 있습니다.

푸시 동기화를 활성화하려면

1. Amazon Cognito 콘솔에서 자격 증명 풀을 선택합니다. 자격 증명 풀을 선택합니다.
2. 자격 증명 풀 속성 탭을 선택합니다.
3. 푸시 동기화에서 편집을 선택합니다.
4. 자격 증명 풀과 푸시 동기화 활성화를 선택합니다.
5. 현재 AWS 리전에 생성한 Amazon Simple Notification Service(SNS) 플랫폼 애플리케이션 중 하나를 선택합니다. Amazon Cognito는 플랫폼 애플리케이션에 푸시 알림을 게시합니다. 플랫폼 애플리케이션 생성을 선택하여 Amazon SNS 콘솔로 이동하고 새 플랫폼 애플리케이션을 생성합니다.
6. Amazon Cognito는 플랫폼 애플리케이션에 게시하기 위해 AWS 계정의 IAM 역할을 맡습니다. Amazon Cognito가 기본 권한 및 자격 증명 풀과의 신뢰 관계가 있는 새 역할을 생성하도록 하려면 새 IAM 역할 생성을 선택합니다. 새 역할을 식별하는 IAM 역할 이름을 입력합니다(예: `myidentitypool_authenticatedrole`). 정책 문서 보기를 선택하여 Amazon Cognito가 새 IAM 역할에 할당하는 권한을 검토합니다.
7. 에 사용하려는 역할이 이미 있는 경우 기존 IAM 역할 사용을 선택할 수 있습니다. AWS 계정 `cognito-identity.amazonaws.com`을 포함하도록 IAM 역할 신뢰 정책을 구성해야 합니다. 특정 자격 증명 풀의 인증된 사용자로부터 요청이 시작되었다는 증거가 제시되는 경우에만 Amazon Cognito가 역할을 맡을 수 있도록 역할 신뢰 정책을 구성합니다. 자세한 내용은 역할 트러스트 및 권한 섹션을 참조하세요.
8. 변경 사항 저장을 선택합니다.

Amazon Cognito 스트림 설정

Amazon Cognito 스트림은 개발자에게 Amazon Cognito Sync에 저장된 데이터에 대한 제어와 인사이트를 제공합니다. 개발자는 이제 이벤트를 데이터로 수신하도록 Kinesis 스트림을 구성할 수 있습니다. Amazon Cognito는 각 데이터 집합 변경 사항을 사용자가 소유하는 Kinesis 스트림으로 실시간으

로 푸시합니다. Amazon Cognito 콘솔에서 Amazon Cognito 스트림을 설정하는 방법에 대한 지침은 Amazon Cognito Sync 스트림 구현 섹션을 참조하세요.

Amazon Cognito 이벤트 설정

Amazon Cognito 이벤트를 사용하면 Amazon Cognito Sync의 중요한 이벤트에 대한 응답으로 AWS Lambda 함수를 실행할 수 있습니다. Amazon Cognito Sync는 데이터 집합이 동기화될 경우 동기화 트리거 이벤트를 유발합니다. 사용자가 데이터를 업데이트할 때 동기화 트리거 이벤트를 사용하여 작업을 수행할 수 있습니다. 콘솔에서 Amazon Cognito 이벤트를 설정하는 방법에 대한 자세한 내용은 Amazon Cognito Events를 사용하여 워크플로 사용자 지정 섹션을 참조하세요.

에 대한 자세한 내용은 단원을 AWS Lambda참조하십시오AWS Lambda.

호스팅 UI(클래식) 브랜딩 사용자 지정

AWS Management Console또는 AWS CLI API를 사용하여 호스팅 UI에 대한 클래식 사용자 지정 설정을 지정할 수 있습니다. 앱에 표시할 사용자 지정 로고 이미지를 업로드할 수 있습니다. UI의 디자인과 느낌에 일부 계단식 스타일 시트(CSS) 옵션을 적용할 수도 있습니다.

UI 기본값을 사용자 지정하고 특정 설정으로 개별 앱 클라이언트를 재정의할 수 있습니다. Amazon Cognito는 클라이언트 수준 설정이 없는 모든 앱 클라이언트에 기본 구성을 적용합니다.

Amazon Cognito 콘솔 및 API 요청에서 UI 사용자 지정을 설정하는 요청은 크기가 135KB를 초과해서는 안 됩니다. 드물지만 요청 헤더, CSS 파일 및 로고의 합계가 135KB를 초과할 때도 있습니다. Amazon Cognito는 이미지 파일을 Base64로 인코딩합니다. 이렇게 하면 100KB 이미지의 크기가 130KB로 늘어나, 요청 헤더와 CSS에는 5KB가 유지됩니다. 요청이 너무 크면 AWS Management Console 또는 `SetUICustomization` API 요청이 `request parameters too large` 오류를 반환합니다. 로고 이미지를 100KB 이하로 조정하고 CSS 파일은 3KB 이하로 조정하세요. CSS와 로고 사용자 지정을 별도로 설정할 수는 없습니다.

> **Note**
>
> UI를 사용자 지정하려면 사용자 풀에 대한 도메인을 설정해야 합니다.

클래식 브랜딩으로 사용자 지정 로고 지정

Amazon Cognito는 사용자 지정 로고를 Login 엔드포인트의 입력 필드 위 중간에 배치합니다.

사용자 지정 호스팅 UI 로고용으로 350x178픽셀까지 확대할 수 있는 PNG, JPG 또는 JPEG 파일을 선택합니다. 로고 파일의 크기는 100KB를 초과할 수 없으며, Amazon Cognito가 Base64로 인코딩한 후에는 130KB를 초과할 수 없습니다. APIImageFileSetUICustomization에서를 설정하려면 파일을 Base64-encoded 텍스트 문자열로 변환하거나에서 파일 경로를 AWS CLI제공하고 Amazon Cognito가 자동으로 인코딩하도록 합니다.

클래식 브랜딩에서 CSS 사용자 지정 항목 지정

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

- 다음 CSS 클래스 이름 중 하나를 사용할 수 있습니다.
  - `background-customizable`
  - `banner-customizable`
  - `errorMessage-customizable`
  - `idpButton-customizable`
  - `idpButton-customizable:hover`
  - `idpDescription-customizable`
  - `inputField-customizable`
  - `inputField-customizable:focus`
  - `label-customizable`
  - `legalText-customizable`
  - `logo-customizable`
  - `passwordCheck-valid-customizable`
  - `passwordCheck-notValid-customizable`
  - `redirect-customizable`
  - `socialButton-customizable`
  - `submitButton-customizable`
  - `submitButton-customizable:hover`
  - `textDescription-customizable`
- 속성 값에는 `@import`, `@supports`, `@page` 또는 `@media` 문이나 Javascript 값을 제외한 HTML이 포함될 수 있습니다.

다음 CSS 속성을 사용자 지정할 수 있습니다.

레이블

- font-weight는 100에서 900 사이의 100의 배수입니다.
- color는 텍스트 색상입니다. 법적 CSS 색상 값이어야 합니다.

입력 필드

- width는 포함 블록의 백분율로 표시되는 너비입니다.
- height는 픽셀(px) 단위의 입력 필드 높이입니다.
- color는 텍스트 색상입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.
- background-color는 입력 필드의 배경색입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.
- border는 앱 창 경계의 너비, 투명도 및 색상을 지정하는 표준 CSS 경계 값입니다. 너비의 범위는 1~100px입니다. 투명도는 단색(solid) 또는 없음(none)입니다. 표준 색상 값이면 무엇이든 가능합니다.

텍스트 설명

- padding-top은 텍스트 설명 위의 패딩 양입니다.
- padding-bottom은 텍스트 설명 아래의 패딩 양입니다.
- display는 `block` 또는 `inline`일 수 있습니다.
- font-size는 텍스트 설명의 글꼴 크기 입니다.
- color는 텍스트 색상입니다. 법적 CSS 색상 값이어야 합니다.

제출 버튼

- font-size는 버튼 텍스트의 글꼴 크기 입니다.
- font-weight는 버튼 텍스트의 글꼴 두께로 `bold`, `italic` 또는 `normal`일 수 있습니다.
- margin은 버튼의 위쪽, 오른쪽, 아래쪽 및 왼쪽 마진 크기를 나타내는 4자 문자열입니다.
- font-size는 텍스트 설명의 글꼴 크기 입니다.
- width는 포함 블록의 비율로 표시되는 버튼 텍스트의 너비입니다.
- height는 픽셀(px) 단위의 버튼 높이입니다.
- color는 버튼 텍스트 색상입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.
- background-color는 버튼의 배경색입니다. 표준 색상 값이면 무엇이든 가능합니다.

배너

- padding은 배너의 위쪽, 오른쪽, 아래쪽 및 왼쪽 패딩 크기를 나타내는 4자 문자열입니다.
- background-color는 배너의 배경색입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.

제출 버튼 가리키기

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

자격 증명 공급자 버튼 가리키기

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

암호 확인이 유효하지 않음

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

배경

- background-color는 앱 창의 배경색입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.

오류 메시지

- margin은 위쪽, 오른쪽, 아래쪽 및 왼쪽 마진 크기를 나타내는 4자 문자열입니다.
- padding은 패딩 크기입니다.
- font-size는 글꼴 크기 입니다.
- width는 포함 블록의 비율로 표시되는 오류 메시지의 너비입니다.
- background-color는 오류 메시지의 배경색입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.
- border는 경계의 너비, 투명도 및 색상을 지정하는 3자 문자열입니다.
- color는 오류 메시지 텍스트 색상입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.
- box-sizing은 크기 속성(너비 및 높이)에 포함시켜야 할 브라우저를 나타내는 데 사용됩니다.

자격 증명 공급자 버튼

- height는 픽셀(px) 단위의 버튼 높이입니다.
- width는 포함 블록의 비율로 표시되는 버튼 텍스트의 너비입니다.
- text-align은 텍스트 정렬 설정입니다. `left`, `right` 또는 `center`일 수 있습니다.
- margin-bottom은 아래쪽 마진 설정입니다.
- color는 버튼 텍스트 색상입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.
- background-color는 버튼의 배경색입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.

- border-color는 버튼 경계의 색상입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.

자격 증명 공급자 설명

- padding-top은 설명 위의 패딩 양입니다.
- padding-bottom은 설명 아래의 패딩 양입니다.
- display는 `block` 또는 `inline`일 수 있습니다.
- font-size는 설명의 글꼴 크기 입니다.
- color는 기업 ID로 로그인과 같은 IdP 섹션 헤더의 텍스트 색상입니다. 법적 CSS 색상 값이어야 합니다.

법적 고지 텍스트

- color는 텍스트 색상입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.
- font-size는 글꼴 크기 입니다.

> **Note**
>
> [법적 고지 텍스트(Legal text)]를 사용자 정의하는 경우 로그인 페이지에서 소셜 자격 증명 공급자 아래에 표시되는 [먼저 묻지 않고 계정에 게시하지 않습니다.(We won't post to any of your accounts without asking first)]라는 메시지를 사용자 정의하는 것입니다.

로고

- max-width는 포함 블록의 비율로 표시되는 최대 너비입니다.
- max-height는 포함 블록의 비율로 표시되는 최대 높이입니다.
- background-color는 투명한 섹션이 있는 로고의 배경 색상입니다. 법적 CSS 색상 값이어야 합니다.

입력 필드 포커스

- border-color는 입력 필드의 색상입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.
- outline은 픽셀 단위의 입력 필드 경계 너비입니다.

소셜 버튼

- height는 픽셀(px) 단위의 버튼 높이입니다.
- text-align은 텍스트 정렬 설정입니다. `left`, `right` 또는 `center`일 수 있습니다.
- width는 포함 블록의 비율로 표시되는 버튼 텍스트의 너비입니다.
- margin-bottom은 아래쪽 마진 설정입니다.

암호 확인이 유효함

- color는 "Password check valid" 메시지의 텍스트 색상입니다. 표준 CSS 색상 값이면 무엇이든 가능합니다.

에서 클래식 브랜딩을 사용하여 호스팅 UI 사용자 지정 AWS Management Console

를 사용하여 앱 AWS Management Console 에 대한 UI 사용자 지정 설정을 지정할 수 있습니다.

> **Note**
>
> 사용자 풀에 대한 세부 사항을 포함시켜 URL `https://<your_domain>/login?response_type=code&client_id=<your_app_client_id>&redirect_uri=<your_callbac` 구성하고 이를 브라우저에 입력하면 사용자가 지정한 호스팅 UI를 볼 수 있습니다. 브라우저를 새로고침하고 콘솔의 변경 내용이 나타나기까지 최대 1분 정도 기다려야 할 수도 있습니다. 도메인은 [앱 통합(App integration)] 탭의 [도메인(Domain)] 아래에 표시됩니다. 앱 클라이언트 ID와 콜백 URL은 [앱 클라이언트(App clients)] 아래에 표시됩니다.

앱 UI 사용자 지정 설정 지정 방법

1. Amazon Cognito 콘솔에 로그인합니다.
2. 탐색 창에서 [사용자 풀(User Pools)]을 선택한 다음 편집할 사용자 풀을 선택합니다.
3. 도메인 탭에서 도메인을 생성하거나 기존 도메인을 업데이트합니다. 브랜딩 버전에서 호스팅 UI(클래식)를 사용하도록 도메인을 설정합니다.
4. 관리형 로그인 메뉴를 선택합니다.
5. 모든 앱 클라이언트의 UI 설정을 사용자 지정하려면 호스팅 UI 설정에서 스타일을 찾아 편집을 선택합니다.
6. 하나의 앱 클라이언트에 대한 UI 설정을 사용자 지정하려면 앱 클라이언트 메뉴로 이동하여 수정하려는 앱 클라이언트를 선택한 다음, 호스팅 UI(클래식) 스타일을 찾아 재정의를 선택합니다. 편집을 선택합니다.
7. 고유한 로고 이미지 파일을 업로드하려면 [파일 선택(Choose file)] 또는 [현재 파일 바꾸기(Replace current file)]를 선택합니다.
8. 호스팅 UI CSS를 사용자 지정하려면 CSS template.css를 다운로드한 다음, 사용자 지정하려는 값으로 템플릿을 수정합니다. 템플릿에 포함된 키만 호스트된 UI에 사용할 수 있습니다. 추가된

CSS 키는 UI에 반영되지 않습니다. CSS 파일을 사용자 정의한 후 [파일 선택(Choose file)] 또는 [현재 파일 바꾸기(Replace current file)]를 선택하여 사용자 정의 CSS 파일을 업로드합니다.

사용자 풀 API 및에서 클래식 브랜딩을 사용하여 호스팅 UI 사용자 지정 AWS CLI

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

사용자 풀의 내장 앱 UI에 대한 UI 사용자 지정 설정을 가져오려면 다음 API 작업을 사용합니다.

- AWS CLI: `aws cognito-idp get-ui-customization`
- AWS API: GetUICustomization

사용자 풀의 내장 앱 UI에 대한 UI 사용자 지정 설정을 설정하려면 다음 API 작업을 사용합니다.

- AWS CLI 이미지 파일에서: `aws cognito-idp set-ui-customization --user-pool-id <your-user-pool-id> --client-id <your-app-client-id> --image-file fileb://"<path-to-logo-image-file>" --css ".label-customizable{ color: <color>;}"`
- AWS CLI Base64 바이너리 텍스트로 인코딩된 이미지 포함: `aws cognito-idp set-ui-customization --user-pool-id <your-user-pool-id> --client-id <your-app-client-id> --image-file <base64-encoded-image-file> --css ".label-customizable{ color: <color>;}"`
- AWS API: SetUICustomization

Amazon Cognito의 문서 기록

다음 표에서는 Amazon Cognito 설명서에 대한 중요 추가 사항을 설명합니다. 또한 사용자로부터 받은 의견을 수렴하기 위해 설명서가 자주 업데이트됩니다. 피드백을 제출하려면 Amazon Cognito 설명서의 페이지 하단에서 Feedback(피드백) 링크를 찾으세요.

변경 사항	설명	날짜
재구성된 API 및 엔드포인트 참조	사용자 풀 엔드포인트 및 관리형 로그인 참조는 이제 가이드의 최상위 섹션입니다. 가이드는 더 이상 Amazon Cognito 사용자 풀, Amazon Cognito 자격 증명 풀 및 Amazon Cognito Sync API 참조에 연결하지 않습니다. 이러한 API 참조는 해당 API 참조 가이드에서 계속 사용할 수 있습니다.	2026년 10월 8일.
AWS WAF 사용자 지정 블록 응답의 상태 코드	이제 Amazon Cognito는 관리형 로그인 및 클래식 호스팅 UI의 AWS WAF 사용자 지정 블록 응답에서 `403` 상태 코드 대신 `5xx` 상태 코드를 반환합니다.	2026년 10월 1일
Amazon Cognito Sync는 더 이상 신규 고객에게 공개되지 않습니다.	Amazon Cognito Sync는 더 이상 신규 고객에게 공개되지 않습니다. Amazon Cognito Sync에 대한 대안은 AWS AppSync 및 DynamoDB를 살펴보세요. 자세히 알아보기	2026년 7월 31일
GetClientTokenMachine-to-machine 권한 부여	이제 사용자 풀 도메인 없이 Amazon Cognito `GetClientToken` API 작업을 사용하여 M2M(Machinemachine-to-	2026년 7월 29일

	machine) 권한 부여를 위한 액세스 토큰을 발급할 수 있습니다.	
암호 해시 사용자 가져오기	이제 기존 암호 해시가 있는 사용자를 가져와 암호 재설정 없이 즉시 로그인할 수 있습니다.	2026년 7월 13일
프로비저닝된 제한 관리	이제 Amazon Cognito 프로비저닝 APIs.	2026년 7월 2일
Amazon Cognito Sync는 2026년 7월 30일부터 신규 고객에게 더 이상 공개되지 않습니다.	Amazon Cognito Sync는 2026년 7월 30일부터 신규 고객에게 더 이상 공개되지 않습니다. 기존 Amazon Cognito Sync 고객은 서비스를 계속 사용할 수 있습니다. Amazon Cognito Sync에 대한 대안은 AWS AppSync 및 DynamoDB를 살펴보세요. 자세히 알아보기	2026년 6월 30일
사용자 지정 도메인에 대한 TLS 버전 선택	이제 Amazon Cognito API를 사용하여 사용자 풀 도메인을 생성하거나 업데이트할 때 사용자 지정 도메인의 최소 TLS 버전을 선택할 수 있습니다.	2026년 6월 29일
사용자 풀에 대한 다중 리전 복제.	이제 추가에서 사용자 풀의 추가 복사본을 설정할 수 있습니다 AWS 리전.	2026년 5월 28일
유휴 사용자 풀 암호화를 위한 고객 관리형 KMS 키입니다.	이제 사용자 지정 KMS 키를 사용하여 유휴 시 사용자 풀 암호화를 구성할 수 있습니다.	2026년 5월 28일

Lambda 트리거 재시도 동작 설명서가 업데이트되었습니다.	Amazon Cognito가 고정된 재시도 횟수를 지정하지 않고 제한 시간에 Lambda 함수 호출을 재시도할 수 있음을 명확히 하기 위해 설명서를 업데이트했습니다.	2026년 3월 23일
앱 클라이언트의 클라이언트 보안 암호 교체.	이제 앱 클라이언트당 최대 2개의 보안 암호를 연결하여 보안 암호를 교체할 수 있습니다. 보안 암호를 추가할 때 사용자 지정 보안 암호 값을 제공할 수도 있습니다.	2026년 1월 30일
인바운드 연동 Lambda 트리거는 Amazon Cognito 사용자 풀에서 사용할 수 있습니다.	이제 인바운드 페더레이션 Lambda 트리거를 사용하여 외부 자격 증명 공급자와의 인증 프로세스 중에 페더레이션 사용자 속성을 변환할 수 있습니다.	2026년 1월 28일
AWS PrivateLink 는 Amazon Cognito 자격 증명 풀에서 사용할 수 있습니다.	이제 VPC 엔드포인트를 통해 Amazon Cognito 자격 증명 풀 API 작업에 요청을 보낼 수 있습니다.	2025년 12월 10일
AWS PrivateLink 는 Amazon Cognito 사용자 풀에서 사용할 수 있습니다.	이제 VPC 엔드포인트를 통해 Amazon Cognito 사용자 풀 API 작업에 요청을 보낼 수 있습니다.	2025년 11월 4일
Amazon Cognito 사용자 풀이 이제 리소스 바인딩을 지원합니다.	이제 리소스 서버가 RFC 8707 리소스 바인딩을 사용하여 액세스 토큰의 대상 확인을 수행할 수 있습니다.	2025년 10월 24일

문제 해결 장이 추가되었습니다.	사용자 풀의 일반적인 문제에 대한 솔루션이 포함된 새로운 장입니다.	2025년 10월 24일
사용자 풀 JSON 웹 키 문서 요청에 대한 새 할당량을 추가했습니다.	사용자 풀 OAuth 2.0 권한 부여 서버에서 `.well-known/jwks.json` 을 가져오기 위한 새 할당량 항목입니다.	2025년 10월 13일
관리형 로그인의 이용 약관 문서.	이제 관리형 로그인 가입 페이지에서 이용 약관 및 개인 정보 보호 정책에 대한 링크를 추가할 수 있습니다.	2025년 10월 2일
ID 풀에 대한 새로운 시작 예시 애플리케이션.	이제 ID 풀의 기능을 보여주는 예시 애플리케이션을 설정할 수 있습니다.	2025년 9월 30일
ID 풀에 대한 IAM 역할 신뢰 정책의 요구 사항 업데이트.	이제 조건 키가 OIDC(웹 ID) 페더레이션 대상의 범위를 하나 이상의 ID 풀로 지정하는 경우에만 Amazon Cognito ID 풀 서비스 보안 주체에 대한 역할 신뢰 정책을 변경할 수 있습니다.	2025년 8월 1일
이제 멕시코(중부)에서 Amazon Cognito를 사용할 수 있습니다 AWS 리전.	이제 멕시코(중부) 리전에서 Amazon Cognito 리소스를 생성할 수 있습니다.	2025년 7월 24일
이제 아시아 태평양(태국)에서 Amazon Cognito를 사용할 수 있습니다 AWS 리전.	이제 아시아 태평양(태국) 리전에서 Amazon Cognito 리소스를 생성할 수 있습니다.	2025년 7월 24일
AWS WAF 관리형 로그인의 웹 ACLs.	이제 관리형 로그인 브랜딩 버전이 있는 사용자 풀 앱 클라이언트에 AWS WAF 웹 ACL 규칙을 적용할 수 있습니다.	2025년 6월 24일

Lambda 트리거 예시를 업데이트했습니다.	사용자 지정 이메일 및 SMS 발신자 Lambda 트리거의 예시 함수가 Node.js 22.x와 호환되도록 업데이트되었습니다. 이제 예시도 테스트에 더 쉽게 액세스할 수 있습니다.	2025년 5월 19일
새 `prompt` 파라미터.	이제 `prompt` 파라미터를 사용하여 기존 관리형 로그인 세션의 재인증을 더 잘 제어할 수 있습니다. 이 파라미터의 값을 타사 제공업체에게 전달할 수도 있습니다.	2025년 5월 15일
M2M 요청에 대한 클라이언트 메타데이터입니다.	이제 클라이언트 자격 증명이나 M2M(머신 간) 요청에서 클라이언트 메타데이터를 전달할 수 있습니다. Amazon Cognito는 M2M 클라이언트 메타데이터를 사전 토큰 생성 Lambda 트리거에 전달합니다.	2025년 4월 29일
새로 고침 토큰 교체.	이제 새 새로 고침 토큰을 가져오고 새로 고침 토큰 요청에서 원본을 무효화할 수 있습니다.	2025년 4월 22일
이제 아시아 태평양(말레이시아)에서 Amazon Cognito를 사용할 수 있습니다 AWS 리전.	이제 아시아 태평양(말레이시아) 리전에서 Amazon Cognito 리소스를 생성할 수 있습니다.	2025년 3월 7일
시스템 ID에 대한 액세스 토큰 사용자 지정.	사전 토큰 생성 Lambda 트리거에는 이제 M2M(머신 간) 권한 부여를 위한 클라이언트 자격 증명 권한 부여에서 액세스 토큰 클레임 및 범위를 수정하는 버전 3 이벤트가 있습니다.	2025년 3월 3일

AmazonCognitoPowerUser AWS 관리형 정책과 관련된 정보가 업데이트되었습니다.	Amazon Cognito 사용자 풀 고급 사용자를 위한 AWS 관리형 정책에 AWS 최종 사용자 메시징 SMS 작업이 추가되었습니다.	2025년 2월 27일
OpenID Connect(OIDC) 통합의 개요를 업데이트했습니다.	Amazon Cognito가 OIDC ID 제공업체를 통해 인증하는 방법을 보여주는 다이어그램을 추가했습니다.	2025년 2월 25일
MFA 로직에 대한 정보가 추가되었습니다.	Amazon Cognito가 런타임 시 사용자에게 사용자 풀 다중 인증(MFA) 설정을 적용하는 방법을 보여주는 다이어그램을 추가했습니다.	2025년 2월 25일
Amazon Cognito 사용자 풀 보안 모범 사례를 추가했습니다.	보안 암호 보안 및 사용자 풀 구성의 보안 모범 사례 준수에 대한 페이지가 추가되었습니다.	2025년 2월 25일
사용자 풀의 시작하기 리소스에 대한 업데이트.	Amazon Cognito 사용자 풀 시작하기 환경에는 새로운 콘솔 설계 및 애플리케이션 옵션이 있습니다.	2024년 11월 21일
기능 플랜이 포함된 새로운 요금 모델.	사용자 풀에 대한 청구 모델을 업데이트했습니다. 고급 보안 기능은 이제 위협 방지 기능이 되었습니다. 고급 보안 기능 라이선스의 구성 요소는 이제 Essentials 및 Plus 기능 플랜에 포함되어 있습니다.	2024년 11월 21일
새로운 관리형 로그인 기능.	호스팅 UI에 대한 업데이트인 관리형 로그인을 시작했습니다.	2024년 11월 21일

새로운 인증 방법과 새로운 인증 흐름.	이제 패스키와 일회용 암호를 사용하여 Amazon Cognito 사용자 풀에 로그인할 수 있습니다.	2024년 11월 21일
AmazonCognitoUnAuthedIdentitiesSessionPolicy 에 대한 정보가 업데이트되었습니다.	인증되지 않은 자격 증명의 범위를 줄이기 위해 AWS 관리형 정책의 AWS Key Management Service 작업을 인라인 정책에서 AWS 관리형 정책으로 이동했습니다.	2024년 11월 1일
login_hint 파라미터를 추가했습니다.	이제 호스팅 UI, OIDC IdP 및 Google IdP에 대한 권한 부여 요청에 사용자 이름 힌트를 추가할 수 있습니다.	2024년 10월 3일
이메일 MFA를 위한 새로운 고급 보안 기능.	이제 고급 보안 기능이 포함된 이메일 메시지로 다중 인증(MFA) 코드를 보낼 수 있습니다.	2024년 9월 12일
새 콘텐츠 및 페이지 변경 사항.	제목 수정, 불필요한 콘텐츠 제거, 시나리오 기반 소개 추가, 사용자 풀 OIDC 이동 및 사용자 풀 섹션에 대한 호스팅 UI 엔드포인트 참조.	2024년 9월 9일
AmazonCognitoUnAuthedIdentitiesSessionPolicy 에 대한 정보가 업데이트되었습니다.	자격 증명 풀에서 인증되지 않은 자격 증명의 범위 축소에 대한 AWS 관리형 정책은 이제 Amazon Location Service를 허용합니다.	2024년 8월 9일

Lambda 트리거를 사용한 사용자 지정 인증을 위한 새로운 위협 방지 및 향상된 위협 탐지.	이제 위협 방지를 사용하여 사용자 지정 인증 로그인을 분석하고 적응형 인증 응답을 적용할 수 있습니다. 위협 방지는 이제 로그인 트래픽에서 시도 간 불가능한 지리적 거리를 분석합니다.	2024년 8월 8일
암호 재사용 방지 및 사용자 활동 로그 내보내기를 위한 새로운 고급 보안 기능.	이제 사용자 활동 로그를 내보내고 Amazon Cognito 사용자 풀의 고급 보안 기능을 사용하여 암호 기록 정책을 설정할 수 있습니다.	2024년 8월 6일
이제 캐나다 서부(캘거리) 및 아시아 태평양(홍콩)에서 Amazon Cognito를 사용할 수 있습니다 AWS 리전.	이제 캐나다 서부(캘거리) 및 아시아 태평양(홍콩) 리전에서 Amazon Cognito 리소스를 생성할 수 있습니다.	2024년 7월 9일
고급 보안을 위한 애플리케이션 동작에 대한 설명 개선	고급 보안 적응형 인증을 위한 디바이스 컨텍스트 데이터에 대한 정보가 업데이트되었습니다.	2024년 6월 10일
사전 토큰 Lambda 트리거의 복잡한 개체에 대한 지원이 추가되었습니다.	이제 ID 및 액세스 토큰 클레임에 배열 및 JSON 객체를 추가할 수 있습니다.	2024년 5월 30일
Verified Permissions 및 Amazon Cognito 대한 정보가 업데이트되었습니다.	Amazon Verified Permissions는 이제 Amazon Cognito와 더 직접적으로 통합됩니다.	2024년 5월 15일
다중 리전 Amazon SES가 ID를 확인했습니다.	Amazon SES가 AWS 리전 없는 일부 에서는 Amazon Cognito 사용자 풀이 두 원격 리전 간에 이메일을 로드 밸런싱합니다.	2024년 5월 10일

M2M 권한 부여 및 비용 관리에 대한 정보가 추가되었습니다.	Amazon Cognito 사용자 풀에서 M2M(기계 간) 사용 사례에 클라이언트 자격 증명 권한 부여를 사용하는 방법을 알아봅니다.	2024년 5월 9일
이제 유럽(스페인) 및 아시아 태평양(하이데라바드)에서 Amazon Cognito를 사용할 수 있습니다 AWS 리전.	이제 유럽(스페인) 및 아시아 태평양(하이데라바드) 리전에서 Amazon Cognito 리소스를 생성할 수 있습니다.	2024년 4월 15일
이제 아시아 태평양(멜버른)에서 Amazon Cognito를 사용할 수 있습니다 AWS 리전.	이제 아시아 태평양(멜버른) 리전에서 Amazon Cognito 리소스를 생성할 수 있습니다.	2024년 4월 4일
Flutter for Amazon Cognito 사용자 풀에 Android 앱 예시가 추가되었습니다.	GitHub의 Flutter 애플리케이션 예시에서 Amazon Cognito용 스타터 모바일 앱을 빌드할 수 있습니다.	2024년 4월 4일
새로 시작되는 콘텐츠	시작, 일반적인 시나리오, 멀티테넌트 모범 사례 및 로그인 후 리소스 액세스를 위한 콘텐츠 확장.	2024년 4월 1일
이제 유럽(취리히)에서 Amazon Cognito를 사용할 수 있습니다 AWS 리전.	이제 유럽(취리히) 리전에서 Amazon Cognito 리소스를 생성할 수 있습니다.	2024년 3월 14일
이제 중동(UAE)에서 Amazon Cognito를 사용할 수 있습니다 AWS 리전.	이제 중동(UAE) 리전에서 Amazon Cognito 리소스를 생성할 수 있습니다.	2024년 3월 8일
새로운 SAML 기능 및 향상된 콘텐츠.	이제 SAML 요청에 서명하고, SAML 응답을 암호화하고, IdP 시작 SAML SSO를 설정할 수 있습니다.	2024년 2월 1일

할당량 증가를 사용할 수 있습니다.	이제 Amazon Cognito 요청 속도 할당량을 위해 추가 용량을 구매할 수 있습니다.	2024년 1월 25일
Amazon Cognito ID 풀은 Service Quotas의 요청 비율을 뒷받침합니다.	이제 Amazon Cognito 자격 증명 풀의 초당 요청(RPS) 할당량을 모니터링하고 Service Quotas 콘솔에서 증가를 요청할 수 있습니다.	2023년 12월 19일
액세스 토큰의 콘텐츠를 사용자 지정하는 새로운 기능이 추가되었습니다.	이제 사용자 풀 액세스 토큰에서 클레임 및 범위를 추가, 수정 및 제거할 수 있습니다.	2023년 12월 12일
앱 클라이언트 및 OAuth 범위에 대한 내용을 개선했습니다.	앱 클라이언트를 사용한 애플리케이션별 설정 및 범위, M2M 및 리소스 서버 관련 사항을 보다 명확하게 수정했습니다. 기존 콘솔 지침을 제거했습니다.	2023년 11월 14일
디바이스 및 디바이스 인증에 대한 내용을 개선했습니다.	디바이스 키 및 디바이스 SRP 인증 사용에 대한 새로운 콘텐츠를 추가했습니다.	2023년 10월 18일
AWS Management Console 지침을 업데이트했습니다.	사용자 풀 콘솔 참조를 제거하고 관련 주제 내에서 주제를 재배치하고 Amazon Cognito 콘솔의 탭 기반 구성에 지침을 추가했습니다.	2023년 8월 30일
LOGIN 엔드포인트에 대한 직접 액세스 강조 해제.	사용자 풀 Login 엔드포인트의 시각적 개요를 추가하고 권한 부여 엔드포인트를 통한 인증 시작을 강조했습니다.	2023년 8월 30일

이제 아시아 태평양(오사카) 및 이스라엘(텔아비브)에서 Amazon Cognito를 사용할 수 있습니다 AWS 리전.	이제 아시아 태평양(오사카) 및 이스라엘(텔아비브) 리전에서 Amazon Cognito 리소스를 생성할 수 있습니다.	2023년 8월 30일
Amazon Verified Permissions를 통한 Amazon Cognito의 권한 부여에 대한 정보가 소개되었습니다.	앱에서 Verified Permissions API를 호출하여 중앙 기관으로부터 액세스 결정을 내릴 수 있습니다.	2023년 8월 1일
Amazon CloudWatch Logs에 사용자 풀의 세부 사용자 활동을 로깅하는 새로운 기능이 추가되었습니다.	이제 CloudWatch 로그 그룹에 이메일 및 SMS 메시지 전송 오류를 로깅할 수 있습니다.	2023년 8월 1일
자격 증명 풀 게스트 사용자의 AWS 관리형 정책에 대한 정보가 업데이트되었습니다.	자격 증명 풀 게스트 사용자의 권한 범위에는 이제 인라인 세션 정책과 AWS 관리형 세션 정책이 모두 포함됩니다.	2023년 5월 16일
Amazon Cognito ID 풀에 대한 콘텐츠 개선 및 새로운 콘솔 지침.	새로운 콘솔 경험을 반영하여 새로운 콘솔 안내를 추가하고 자격 증명 풀에 대한 코드 통합 세부 정보를 개선했습니다.	2023년 5월 16일
서비스 홈페이지 및 사용자 풀 홈페이지가 추가 및 개선되었습니다.	Amazon Cognito 및 사용자 풀에 대한 개요 페이지가 업데이트되었습니다.	2023년 5월 16일
사용자 풀 토큰 설명서의 일반 개선 사항.	예시 토큰을 업데이트하고 토큰 검증에 대한 새로운 정보를 추가했습니다.	2023년 2월 16일
이제 Amazon Cognito ID 풀 데이터 이벤트를 AWS CloudTrail에 로깅할 수 있습니다.	CloudTrail이 데이터 이벤트를 로깅하는 추적에서 Amazon Cognito 자격 증명 풀 대용량 API 작업 중 일부를 지원합니다.	2023년 2월 15일

Lambda 트리거 예시 및 설명을 업데이트했습니다.	Lambda 트리거 예시를 JavaScript 버전 3으로 업데이트했습니다. 이제 Lambda 트리거를 API 작업과 직접 연관시킬 수 있습니다.	2023년 1월 31일
Amazon Cognito 자격 증명 풀은 인증되지 않은 세션에 AWS 관리형 정책을 적용합니다.	이제 향상된 흐름을 사용하여 인증하는 자격 증명 풀 사용자에게 세션에 추가 AWS 관리형 정책이 적용됩니다.	2023년 1월 31일
코드 예시를 추가했습니다.	이제이 안내서에는 다양한 프로그래밍 언어로 된 Amazon Cognito 앱의 예제 코드가 포함되어 있습니다.	2023년 1월 23일
Amazon Cognito 사용자 풀을 사용한 API 모델 및 인증 관련 정보가 추가되었습니다.	Amazon Cognito 사용자 풀에는 요청 권한 부여를 위한 여러 API 인터페이스와 형식이 있습니다.	2022년 12월 15일
이제 유럽(밀라노)에서 Amazon Cognito를 사용할 수 있습니다 AWS 리전.	이제 유럽(밀라노) 리전에서 Amazon Cognito 사용자 풀을 생성할 수 있습니다.	2022년 12월 6일
사용자 풀 삭제 방지 관련 정보가 추가되었습니다.	를 사용하여 새 사용자 풀을 생성하면 기본적으로 삭제로부터 보호 AWS Management Console됩니다.	2022년 10월 20일
호스팅 UI에 대한 사용 설명서와 호스팅된 UI에 TOTP MFA에 대한 정보가 추가되었습니다.	이제 사용자는 Amazon Cognito 호스팅 UI에서 TOTP MFA 디바이스를 등록할 수 있습니다. 이제 기본 호스팅 UI를 미리 볼 수 있습니다.	2022년 9월 8일

AWS WAF 및 Amazon Cognito에 대한 정보가 추가되었습니다.	이제 AWS WAF 웹 ACL을 Amazon Cognito 사용자 풀과 연결할 수 있습니다.	2022년 8월 3일
더 많은 예제 AWS CloudTrail 이벤트가 추가되었습니다.	이제 Amazon Cognito가 페더레이션 및 호스팅 UI 요청을 추적에 기록합니다.	2022년 6월 15일
2단계 속성 검증에 대한 정보가 추가되었습니다.	이제 사용자가 새 이메일 주소 또는 전화번호를 인증해야 로그인할 수 있는지 여부를 선택할 수 있습니다.	2022년 6월 9일
페더레이션 설명서가 업데이트되었습니다. 새로운 IP 주소 전파 기능이 추가되었습니다.	사용자 풀 소셜 IdP를 설정하기 위한 연습이 업데이트되었습니다. 페더레이션 사용자 프로필 및 속성 매핑에 대한 정보가 추가되었습니다. 고급 보안을 위한 디바이스 지문에 대한 새로운 정보가 추가되었습니다.	2022년 5월 31일
호스팅 UI와의 상호 작용 없이 페더레이션 사용자 로그인	Amazon Cognito가 자동으로 사용자를 페더레이션 로그인으로 안내할 수 있도록 애플리케이션을 북마크하는 방법에 대한 새 페이지를 추가했습니다.	2022년 5월 29일
Amazon Cognito 사용자 풀에 대한 리전 내 SMS 및 이메일 메시징	이제 사용자 풀과 동일한의 SMS 메시지에는 Amazon Simple Notification Service AWS 리전 를 사용하고 이메일 메시지에는 Amazon Simple Email Service를 사용할 수 있습니다.	2022년 3월 14일
할당량 페이지 업데이트	리소스 및 요청 속도 할당량을 추가하고 확인했습니다.	2022년 1월 10일

새 Amazon Cognito 사용자 풀 콘솔 환경	업데이트된 Amazon Cognito 콘솔에서 사용자 풀을 생성하고 관리하는 지침을 업데이트했습니다.	2021년 11월 18일
RevokeToken API 및 취소 엔드포인트	RevgeToken 작업을 사용하여 사용자의 새로 고침 토큰을 취소할 수 있습니다.	2021년 6월 10일
멀티 테넌트 모범 사례	멀티 테넌트 애플리케이션에 대한 모범 사례가 추가되었습니다.	2021년 3월 4일
액세스 제어용 속성	Amazon Cognito 자격 증명 풀은 고객이 사용자에게 AWS 리소스에 대한 액세스 권한을 부여하는 방법으로 액세스 제어(AFAC)에 대한 속성을 제공합니다. 사용자가 Amazon Cognito와 페더레이션하는 데 사용하는 자격 증명 공급자의 사용자 속성을 기반으로 권한 부여를 수행할 수 있습니다.	2021년 1월 15일
사용자 지정 SMS 발신자 Lambda 트리거 및 사용자 지정 이메일 발신자 Lambda 트리거	사용자 지정 SMS 발신자 Lambda 트리거와 사용자 지정 이메일 발신자 Lambda 트리거를 사용하면 서드 파티 공급자가 Lambda 함수 코드 내에서 사용자에게 이메일 및 SMS 알림을 보낼 수 있습니다.	2020년 11월 30일
Amazon Cognito 토큰 업데이트	업데이트된 만료 정보가 액세스, ID 및 새로 고침 토큰에 추가되었습니다.	2020년 10월 29일

Amazon Cognito Service Quotas	Amazon Cognito 범주 할당량에 Service Quotas를 사용할 수 있습니다. Service Quotas 콘솔을 사용하면 할당량 사용량을 보고 할당량 증가를 요청하고 CloudWatch 경보를 생성하여 할당량 사용량을 모니터링할 수 있습니다. 이 변경 사항의 일환으로, Amazon Cognito 사용자 풀의 사용 가능한 CloudWatch 지표 섹션이 새 정보를 반영하도록 업데이트되었습니다. 새 섹션의 제목은 'CloudWatch 및 Service Quotas에서 할당량 및 사용량 추적'입니다.	2020년 10월 29일
Amazon Cognito 할당량 분류	할당량 범주를 사용하여 할당량 사용량을 모니터링하고 할당량 증가를 요청할 수 있습니다. 할당량은 일반적인 사용 사례에 따라 범주로 분류됩니다.	2020년 8월 17일
이제 Amazon Cognito가 US AWS GovCloud에서 지원됩니다.	이제 AWS GovCloud(미국) 리전에서 Amazon Cognito가 지원됩니다.	2020년 5월 13일
Amazon Cognito Pinpoint 문서 업데이트	새 서비스 연결 역할이 추가되었습니다. 'Amazon Cognito 사용자 풀과 함께 Amazon Pinpoint 분석 사용'의 지침이 업데이트되었습니다.	2020년 5월 13일

새로운 Amazon Cognito 전용 보안 장	보안 장은 조직이 AWS 서비스의 기본 제공 보안과 구성 가능한 보안 모두에 대한 심층적인 정보를 얻는 데 도움이 될 수 있습니다. 새로운 장에서는 클라우드 및 클라우드 내 보안에 대한 정보를 제공합니다.	2020년 4월 30일
Amazon Cognito ID 풀에서 Sign in with Apple 지원	Sign in with Apple은 cn-north-1 리전을 제외하고 Amazon Cognito가 운영되는 모든 리전에서 사용할 수 있습니다.	2020년 4월 7일
새로운 Facebook API 버전 관리	Facebook API에 대한 버전 선택이 추가되었습니다.	2020년 4월 3일
사용자 이름 대/소문자 구분 업데이트	사용자 풀을 생성하기 전에 사용자 이름의 대/소문자를 구분하지 않는 것에 대한 권장 사항이 추가되었습니다.	2020년 2월 11일
에 대한 새로운 정보 AWS Amplify	AWS Amplify SDK 및 라이브러리를 사용하여 웹 또는 모바일 앱과 Amazon Cognito를 통합하는 방법에 대한 정보가 추가되었습니다. AWS Amplify앞에 있는 Amazon Cognito SDK 사용에 대한 정보를 제거했습니다.	2019년 11월 22일

사용자 풀 트리거에 대한 새로운 속성	Amazon Cognito는 이제 대부분의 사용자 풀 트리거에 대한 AWS Lambda 함수에 전달하는 이벤트 정보에 `clientMetadata` 파라미터를 포함합니다. 이 파라미터를 사용하면 추가 데이터로 사용자 지정 인증 워크플로우를 개선할 수 있습니다.	2019년 10월 4일
업데이트된 한도	ListUsers API 작업에 대한 스로틀 제한이 업데이트되었습니다.	2019년 6월 25일
신규 한도	사용자 풀의 소프트 제한에는 이제 사용자 수에 대한 제한이 포함됩니다.	2019년 6월 17일
Amazon Cognito 사용자 풀의 Amazon SES 이메일 설정	Amazon SES 구성을 사용하여 Amazon Cognito에서 사용자에게 이메일을 보내도록 사용자 풀을 구성할 수 있습니다. 이 설정을 사용하면 그렇지 않은 경우보다 Amazon Cognito에서 더 많은 볼륨의 이메일을 전송할 수 있습니다.	2019년 4월 8일
태그 지정 지원	Amazon Cognito 리소스 태깅과 관련한 정보가 추가되었습니다.	2019년 3월 26일
사용자 지정 도메인의 인증서 변경	사용자 지정 도메인을 사용하여 Amazon Cognito 호스트된 UI를 호스트하는 경우 필요에 따라 이 도메인의 SSL 인증서를 변경할 수 있습니다.	2018년 12월 19일

신규 한도	각 사용자가 소속될 수 있는 최대 그룹 수에 새로운 한도가 추가되었습니다.	2018년 12월 14일
업데이트된 한도	사용자 풀의 소프트 한도가 업데이트되었습니다.	2018년 12월 11일
이메일 주소 및 전화번호 확인에 대한 설명서 업데이트	사용자가 앱에 가입할 때 이메일 또는 전화 확인을 요구하도록 사용자 풀을 구성하는 방법에 대한 정보를 추가했습니다.	2018년 11월 20일
이메일 테스트에 대한 설명서 업데이트	앱을 테스트할 때 Amazon Cognito에서 이메일을 시작하는 방법에 대한 지침을 추가했습니다.	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 호스트된 UI의 사용자 지정 도메인	Amazon Cognito 사용자 풀의 호스트된 UI에 개발자의 사용자 지정 도메인을 사용하도록 허용합니다.	2018년 6월 4일
Amazon Cognito 사용자 풀 OIDC ID 제공업체	추가된 사용자 풀이 Salesforce 또는 Ping Identity 같은 OpenID Connect(OIDC) 자격 증명 공급자를 통해 로그인합니다.	2018년 5월 17일

Amazon Cognito Lambda 마이그레이션 트리거	Lambda 마이그레이션 트리거 기능을 다루는 페이지가 추가되었습니다.	2018년 4월 8일
Amazon Cognito 개발자 가이드 업데이트	최상위 'Amazon Cognito란' 및 'Amazon Cognito 시작하기'가 추가되었습니다. 또한 일반적인 시나리오가 추가되고 사용자 풀 TOC가 재구성되었습니다. 'Amazon Cognito 사용자 풀 시작하기' 섹션이 추가되었습니다.	2018년 4월 6일
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 사용하여 Amazon Cognito 사용자 풀 앱에 대한 분석을 제공하고 Amazon Pinpoint 캠페인에 대한 사용자 데이터를 보강하는 기능이 추가되었습니다.	2017년 9월 26일

Amazon Cognito 사용자 풀의 페더레이션 및 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.	2016년 12월 15일
문서 업데이트	사용자 풀에서 AWS Lambda 트리거를 사용하는 방법을 보여주는 예제가 업데이트되었습니다.	2016년 11월 27일
문서 업데이트	iOS 코드 예제가 업데이트되었습니다.	2016년 11월 18일
문서 업데이트	사용자 계정의 확인 흐름에 대한 정보가 추가되었습니다.	2016년 11월 9일
사용자 계정 생성 기능	Amazon Cognito 콘솔 및 API를 통해 사용자 계정을 생성하기 위해 관리 기능이 추가되었습니다.	2016년 10월 6일

사용자 가져오기 기능	Cognito 사용자 풀에 대한 대량 가져오기 기능이 추가되었습니다. 이 기능을 사용하여 기존 자격 증명 공급자에서 Amazon Cognito 사용자 풀로 사용자를 마이그레이션합니다.	2016년 9월 1일
Cognito 사용자 풀의 일반 가용성	Cognito 사용자 풀 기능이 추가되었습니다. 이 기능을 사용하여 사용자 디렉터리를 생성 및 유지 관리하고 사용자 풀을 통해 모바일 앱 또는 웹 애플리케이션에 가입 및 로그인을 추가합니다.	2016년 7월 28일
SAML 지원	SAML 2.0(Security Assertion Markup Language 2.0)을 통해 자격 증명 공급자의 인증에 대한 지원이 추가되었습니다.	2016년 6월 23일
CloudTrail 통합	와의 통합을 추가했습니다 AWS CloudTrail.	2016년 2월 18일
Lambda와 이벤트 통합	Amazon Cognito의 중요한 이벤트에 대한 응답으로 AWS Lambda 함수를 실행할 수 있습니다.	2015년 4월 9일
Amazon Kinesis로의 데이터 스트림	데이터 스트림에 제어 및 통찰력을 제공합니다.	2015년 3월 4일
OpenID Connect 지원	OpenID Connect 공급자에 대한 지원을 활성화합니다.	2014년 11월 23일
푸시 동기화	자동 푸시 동기화에 대한 지원을 활성화합니다.	2014년 11월 6일

개발자 인증 ID 지원이 추가됨	Amazon Cognito에서 자체 인증 및 자격 증명 관리 시스템이 있는 개발자를 자격 증명 공급자로 처리할 수 있습니다.	2014년 9월 29일
Amazon Cognito 정식 출시		2014년 7월 10일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.