사용자 가이드

AWS CodeCommit

API 버전 2015-04-13

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS CodeCommit: 사용자 가이드

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon의 상표 및 브랜드 디자인은 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 관계와 관계없이 해당 소유자의 자산입니다.

Table of Contents

..... xiv
CodeCommit은 무엇인가요? 1
CodeCommit 소개 1
CodeCommit, Git 및 필요에 적합한 AWS 서비스 선택 2
CodeCommit은 어떻게 작동하나요? 5
CodeCommit은 Amazon S3의 파일 버전 관리와 어떻게 다른가요? 7
CodeCommit은 어떻게 시작할 수 있나요? 7
Git에 대해 더 학습하려면 어떻게 해야 하나요? 7
설정 8
보안 인증 정보 보기 및 관리 8
Git 보안 인증 정보를 사용하여 설정 9
다른 방법을 사용하여 설정 9
CodeCommit, Git 및 기타 구성 요소에 대한 호환성 11
Git 보안 인증 정보를 사용하는 HTTPS 사용자의 경우 11
1단계: CodeCommit에 대한 초기 구성 12
2단계: Git 설치 13
3단계: CodeCommit에 대한 HTTPS 접속을 위한 Git 보안 인증 정보 생성 13
4단계: CodeCommit 콘솔 연결 및 리포지토리 복제 15
다음 단계 17
git-remote-codecommit을 사용한 HTTPS 연결의 경우 17
0단계: git-remote-codecommit에 대한 사전 조건 설치 18
1단계: CodeCommit에 대한 초기 구성 19
2단계: git-remote-codecommit 설치 22
3단계: CodeCommit 콘솔 연결 및 리포지토리 복제 23
다음 단계 24
개발 도구에서 연결 25
AWS Cloud9 와 통합 AWS CodeCommit 27
Visual Studio를와 통합 AWS CodeCommit 31
를 사용하지 않는 SSH 사용자의 경우 AWS CLI 32
1단계: 퍼블릭 키를 IAM 사용자와 연결 33
2단계: SSH 구성에 CodeCommit 추가 34
다음 단계 35
Linux, macOS, Unix에서 SSH 연결 35
1단계: CodeCommit에 대한 초기 구성 35

2단계: Git 설치 ... 36
3단계: Linux, macOS, Unix에서 보안 인증 정보 구성 ... 37
4단계: CodeCommit 콘솔 연결 및 리포지토리 복제 ... 41
다음 단계 ... 42
Windows에서 SSH 연결 ... 42
1단계: CodeCommit에 대한 초기 구성 ... 42
2단계: Git 설치 ... 44
3단계: Git과 CodeCommit에 사용되는 퍼블릭 키와 프라이빗 키 설정 ... 44
4단계: CodeCommit 콘솔 연결 및 리포지토리 복제 ... 48
다음 단계 ... 49
AWS CLI 자격 증명 헬퍼를 사용하여 Linux, macOS 또는 Unix에서 HTTPS 연결 ... 49
1단계: CodeCommit에 대한 초기 구성 ... 50
2단계: Git 설치 ... 54
3단계: 보안 인증 도우미 설정 ... 54
4단계: CodeCommit 콘솔 연결 및 리포지토리 복제 ... 56
다음 단계 ... 57
자격 AWS CLI 증명 헬퍼를 사용하여 Windows에서 HTTPS 연결 ... 57
1단계: CodeCommit에 대한 초기 구성 ... 58
2단계: Git 설치 ... 62
3단계: 보안 인증 도우미 설정 ... 62
4단계: CodeCommit 콘솔 연결 및 리포지토리 복제 ... 64
다음 단계 ... 65
시작 ... 66
CodeCommit 시작하기 ... 66
사전 조건 ... 67
1단계: CodeCommit 리포지토리 생성 ... 68
2단계: 리포지토리에 파일 추가 ... 70
3단계: 리포지토리의 콘텐츠 검색 ... 73
4단계: 풀 요청 생성 및 협업 ... 77
5단계: 정리 ... 83
6단계: 다음 단계 ... 84
Git 및 CodeCommit 시작하기 ... 84
1단계: CodeCommit 리포지토리 생성 ... 85
2단계: 로컬 리포지토리 생성 ... 87
3단계: 첫 커밋 생성 ... 88
4단계: 첫 커밋 푸시 ... 89

5단계: CodeCommit 리포지토리를 공유하고 또 다른 커밋을 푸시 및 풀하기 90
6단계: 브랜치 생성 및 공유 92
7단계: 태그 생성 및 공유 94
8단계: 액세스 권한 설정 95
9단계: 정리 98
제품 및 서비스 통합 100
다른 AWS 서비스와의 통합 100
커뮤니티의 통합 예제 107
블로그 게시물 107
코드 샘플 111
리포지토리 작업 112
리포지토리 생성 113
리포지토리 생성 (콘솔) 114
리포지토리 생성 (AWS CLI) 115
리포지토리에 연결 118
CodeCommit 리포지토리에 연결하기 위한 사전 조건 118
리포지토리를 복제하여 CodeCommit 리포지토리에 연결 119
로컬 리포지토리를 CodeCommit 리포지토리에 연결 121
리포지토리 공유 122
사용자들과 공유할 접속 프로토콜 선택 123
리포지토리를 위한 IAM 정책 생성 124
리포지토리 사용자를 위한 IAM 그룹 생성 125
해당 사용자들과 접속 정보 공유 126
리포지토리 이벤트에 대한 알림 구성 127
리포지토리 알림 규칙 사용 129
알림 규칙 생성 129
알림 변경 또는 비활성화 132
알림 삭제 133
리포지토리 태그 지정 134
리포지토리에 태그 추가 135
리포지토리에 대한 태그 보기 137
리포지토리에 대한 태그 편집 138
리포지토리에서 태그 제거 140
리포지토리 트리거 관리 141
리소스 생성 및 CodeCommit에 대한 권한 추가 142
Amazon SNS 주제에 대한 트리거 생성 142

Lambda 함수를 위한 트리거 생성 149
기존 Lambda 함수를 위한 트리거 생성 154
리포지토리 트리거 편집 161
리포지토리 트리거 163
리포지토리에서 트리거 삭제 165
리포지토리를 Amazon CodeGuru Reviewer와 연결 또는 연결 해제 167
리포지토리를 CodeGuru Reviewer와 연결 169
CodeGuru Reviewer에서 리포지토리 연결 해제 170
리포지토리 세부 정보 보기 171
리포지토리 세부 정보 보기 (콘솔) 171
CodeCommit 리포지토리 세부 정보 보기 (Git) 172
CodeCommit 리포지토리 세부 정보 보기 (AWS CLI) 173
리포지토리 설정 변경 177
리포지토리 설정 변경 (콘솔) 177
AWS CodeCommit 리포지토리 설정 변경(AWS CLI) 179
리포지토리 간 변경 사항 동기화 181
두 리포지토리로 커밋 푸시 182
역할을 사용하여 리포지토리에 대한 크로스 계정 액세스 구성 187
크로스 계정 리포지토리 액세스: AccountA의 관리자를 위한 작업 188
교차 계정 리포지토리 액세스: AccountB의 관리자 작업 191
크로스 계정 리포지토리 액세스: AccountB의 리포지토리 사용자를 위한 작업 193
리포지토리 삭제 198
CodeCommit 리포지토리 삭제 (콘솔) 199
로컬 리포지토리 삭제 200
CodeCommit 리포지토리 삭제 (AWS CLI) 200
파일 작업하기 201
리포지토리에서 파일 검색 202
CodeCommit 리포지토리 탐색 202
파일 생성 또는 추가 203
파일 생성 또는 업로드 (콘솔) 204
파일 추가 (AWS CLI) 205
파일 추가 (Git) 207
파일의 콘텐츠 편집 207
파일 편집 (콘솔) 208
파일 편집 또는 삭제 (AWS CLI) 209
파일 편집 (Git) 210

풀 요청 작업 211
풀 요청 생성 215
풀 요청 생성 (콘솔) 215
풀 요청 생성 (AWS CLI) 217
승인 규칙 생성 219
풀 요청에 대한 승인 규칙 생성 (콘솔) 220
풀 요청에 대한 승인 규칙 생성 (AWS CLI) 222
풀 요청 보기 224
풀 요청 보기 (콘솔) 224
풀 요청 보기 (AWS CLI) 225
풀 요청 검토 229
풀 요청 검토 (콘솔) 230
풀 요청 검토 (AWS CLI) 235
풀 요청 업데이트 240
풀 요청 업데이트 (콘솔) 240
풀 요청 업데이트 (AWS CLI) 241
승인 규칙 편집 또는 삭제 243
풀 요청에 대한 승인 규칙 편집 또는 삭제 (콘솔) 244
풀 요청에 대한 승인 규칙 편집 또는 삭제 (AWS CLI) 245
풀 요청에 대한 승인 규칙 재정의 247
승인 규칙 재정의 (콘솔) 248
승인 규칙 재정의 (AWS CLI) 248
풀 요청 병합 249
풀 요청 병합 (콘솔) 250
풀 요청 병합 (AWS CLI) 253
풀 요청의 충돌 해결 259
풀 요청의 충돌 해결 (콘솔) 260
풀 요청의 충돌 해결 (AWS CLI) 262
풀 요청 닫기 269
풀 요청 닫기 (콘솔) 270
풀 요청 닫기(AWS CLI) 271
승인 규칙 템플릿 작업 273
승인 규칙 템플릿 생성 275
승인 규칙 템플릿 생성 (콘솔) 275
승인 규칙 템플릿 생성 (AWS CLI) 279
승인 규칙 템플릿을 리포지토리와 연결 280

승인 규칙 템플릿 연결 (콘솔) ... 281
승인 규칙 템플릿 연결 (AWS CLI) ... 281
승인 규칙 템플릿 관리 ... 282
승인 규칙 템플릿 관리 (콘솔) ... 283
승인 규칙 템플릿 관리 (AWS CLI) ... 283
승인 규칙 템플릿 연결 해제 ... 288
승인 규칙 템플릿 연결 해제 (콘솔) ... 288
승인 규칙 템플릿 연결 해제 (AWS CLI) ... 288
승인 규칙 템플릿 삭제 ... 289
승인 규칙 템플릿 삭제 (콘솔) ... 290
승인 규칙 템플릿 삭제 (AWS CLI) ... 290
커밋 작업 ... 291
커밋 생성 ... 292
를 사용하여 리포지토리에 대한 첫 번째 커밋 생성 AWS CLI ... 292
Git 클라이언트를 사용하여 커밋 생성 ... 294
를 사용하여 커밋 생성 AWS CLI ... 297
커밋 세부 정보 보기 ... 300
리포지토리의 커밋 검색 ... 301
커밋 세부 정보 보기 (AWS CLI) ... 304
커밋 세부 정보 보기 (Git) ... 310
커밋 비교 ... 312
한 커밋을 상위 커밋과 비교 ... 313
임의의 두 커밋 지정자 비교 ... 315
커밋에 대한 주석 추가 ... 317
리포지토리의 커밋에 대한 주석 보기 ... 318
리포지토리의 커밋에 대한 주석 추가 및 댓글 달기 ... 318
설명 보기, 추가, 업데이트, 답변 (AWS CLI) ... 323
Git 태그 생성 ... 332
Git을 사용한 태그 생성 ... 332
태그 세부 정보 보기 ... 333
태그 세부 정보 보기 (콘솔) ... 333
Git 태그 세부 정보 보기 (Git) ... 334
태그 삭제 ... 336
Git을 사용하여 Git 태그 삭제 ... 337
브랜치 작업 ... 338
브랜치 생성 ... 339

브랜치 생성 (콘솔) ... 340
브랜치 생성 (Git) ... 341
브랜치 생성 (AWS CLI) ... 342
브랜치에 대한 푸시 및 병합 제한 ... 344
브랜치에 대한 푸시 및 병합을 제한하는 IAM 정책 구성 ... 344
IAM 그룹 또는 역할에 IAM 정책 적용 ... 346
정책 테스트 ... 347
브랜치 세부 정보 보기 ... 347
브랜치 세부 정보 보기 (콘솔) ... 347
브랜치 세부 정보 보기 (Git) ... 348
브랜치 세부 정보 보기 (AWS CLI) ... 349
브랜치 비교 및 병합 ... 351
임의의 브랜치와 기본 브랜치 간의 비교 ... 351
두 특정 브랜치 비교 ... 351
두 브랜치 병합 (AWS CLI) ... 352
브랜치 설정 변경 ... 355
기본 브랜치 변경 (콘솔) ... 355
기본 브랜치 변경 (AWS CLI) ... 356
브랜치 삭제 ... 357
브랜치 삭제 (콘솔) ... 357
브랜치 삭제 (AWS CLI) ... 358
브랜치 삭제 (Git) ... 358
사용자 기본 설정으로 작업 ... 360
CodeCommit으로 마이그레이션 ... 361
Git 리포지토리를 로 마이그레이션 AWS CodeCommit ... 361
0단계: CodeCommit 액세스에 필요한 설정 ... 362
1단계: CodeCommit 리포지토리 생성 ... 367
2단계: 리포지토리를 복제하여 CodeCommit 리포지토리로 푸시 ... 369
3단계: CodeCommit에서 파일 보기 ... 371
4단계: CodeCommit 리포지토리 공유 ... 371
CodeCommit으로 콘텐츠 마이그레이션 ... 374
0단계: CodeCommit 액세스에 필요한 설정 ... 375
1단계: CodeCommit 리포지토리 생성 ... 380
2단계: 로컬 콘텐츠를 CodeCommit 리포지토리로 마이그레이션 ... 381
3단계: CodeCommit에서 파일 보기 ... 383
4단계: CodeCommit 리포지토리 공유 ... 383

리포지토리를 증분으로 마이그레이션하기 ... 385
0단계: 증분 마이그레이션 여부 결정 ... 386
1단계: 필수 구성 요소를 설치하고 CodeCommit 리포지토리를 원격으로 추가 ... 387
2단계: 증분 마이그레이션에 사용할 스크립트 생성 ... 389
3단계: 스크립트 실행 및 CodeCommit으로 증분 마이그레이션 ... 389
부록: 샘플 스크립트 `incremental-repo-migration.py` ... 390
보안 ... 399
데이터 보호 ... 399
AWS KMS 및 암호화 ... 400
교체 보안 인증 정보 사용 ... 402
IAM(ID 및 액세스 관리) ... 407
대상 ... 408
ID를 통한 인증 ... 408
정책을 사용하여 액세스 관리 ... 411
인증 및 액세스 제어 ... 413
AWS CodeCommit 에서 IAM을 사용하는 방법 ... 480
CodeCommit 리소스 기반 정책 ... 481
CodeCommit 태그 기반 권한 부여 ... 481
CodeCommit IAM 역할 ... 484
자격 증명 기반 정책 예제 ... 485
문제 해결 ... 488
복원성 ... 490
인프라 보안 ... 491
CodeCommit 모니터링 ... 492
CcodeComts 이벤트 모니터링 ... 492
referenceCreated event ... 494
referenceUpdated event ... 494
referenceDeleted 이벤트 ... 495
unreferencedMergeCommitCreated 이벤트 ... 496
commentOnCommitCreated 이벤트 ... 497
commentOnCommitUpdated 이벤트 ... 498
commentOnPullRequestCreated 이벤트 ... 499
commentOnPullRequestUpdated 이벤트 ... 500
pullRequestCreated 이벤트 ... 501
pullRequestSourceBranchUpdated 이벤트 ... 502
pullRequestStatusChanged 이벤트 ... 503

pullRequestMergeStatusUpdated 이벤트 504
approvalRuleTemplateCreated 이벤트 505
approvalRuleTemplateUpdated 이벤트 505
approvalRuleTemplateDeleted 이벤트 506
approvalRuleTemplateAssociatedWithRepository 이벤트 507
approvalRuleTemplateDisassociatedWithRepository 이벤트 508
approvalRuleTemplateBatchAssociatedWithRepositories 이벤트 509
approvalRuleTemplateBatchDisassociatedFromRepositories 이벤트 509
pullRequestApprovalRuleCreated 이벤트 510
pullRequestApprovalRuleDeleted 이벤트 511
pullRequestApprovalRuleOverridden 이벤트 513
pullRequestApprovalStateChanged 이벤트 515
pullRequestApprovalRuleUpdated 이벤트 517
reactionCreated 이벤트 518
reactionUpdated 이벤트 519
를 사용하여 AWS CodeCommit API 호출 로깅 AWS CloudTrail 519
CloudTrail의 CodeCommit 정보 520
CodeCommit 로그 파일 항목 이해 521
AWS CloudFormation 리소스 529
CodeCommit 및 AWS CloudFormation 템플릿 529
템플릿 예제 530
AWS CloudFormation, CodeCommit 및 AWS 클라우드 개발 키트 (AWS CDK) 531
에 대해 자세히 알아보기 AWS CloudFormation 532
문제 해결 533
Git 보안 인증 정보(HTTPS) 문제 해결 533
에 대한 Git 자격 증명 AWS CodeCommit: 터미널 또는 명령줄에서 CodeCommit 리포지토리에 연결할 때 자격 증명 프롬프트가 계속 표시됨 533
에 대한 Git 자격 증명 AWS CodeCommit: Git 자격 증명을 설정했지만 시스템이 이를 사용하고 있지 않습니다. 534
git-remote-codecommit 문제 해결 534
오류가 표시됩니다: git: 'remote-codecommit'은 git 명령어가 아닙니다 535
오류가 표시됩니다: fatal: 'codecommit'의 원격 도우미를 찾을 수 없습니다. 535
복제 오류: IDE에서 CodeCommit 리포지토리를 복제할 수 없습니다 536
푸시 또는 풀 오류: IDE에서 CodeCommit 리포지토리로 커밋을 푸시하거나 풀할 수 없습니다 536
설치 오류: git-remote-codecommitexternally-managed-environment 오류가 표시됩니다. 536

SSH 연결 문제 해결 536
액세스 오류: 퍼블릭 키가 IAM에 성공적으로 업로드되었지만 Linux, macOS, Unix 시스템에서 연결이 실패했습니다 537
액세스 오류: 퍼블릭 키가 IAM에 성공적으로 업로드되고 SSH가 성공적으로 테스트되었지만 Windows 시스템에서 연결이 실패했습니다 538
인증 문제: CodeCommit 리포지토리에 연결할 때 호스트의 인증을 설정할 수 없습니다 539
IAM 오류: 퍼블릭 키를 IAM에 추가하려고 시도할 때 '잘못된 형식' 오류가 나타납니다 546
SSH 보안 인증 정보를 사용하여 여러 Amazon Web Services 계정의 CodeCommit 리포지토리에 액세스해야 합니다 546
Windows에서의 Git: SSH를 사용하여 연결을 시도할 때 Bash 에뮬레이터 또는 명령줄이 중지됩니다 547
일부 Linux 배포판에 공개 키 형식을 지정해야 합니다 548
액세스 오류: CodeCommit 리포지토리에 연결할 때 SSH 퍼블릭 키가 거부됩니다 548
보안 인증 도우미(HTTPS) 문제 해결 548
보안 인증 도우미를 구성하는 `git config` 명령을 실행할 때 오류가 발생합니다. 549
리포지토리를 복제하려고 할 때 사용자 이름을 읽는 동안 오류가 표시됩니다. 549
보안 인증 도우미를 사용할 때 Windows에서 명령을 찾을 수 없음 오류가 발생합니다. 550
CodeCommit 리포지토리에 연결할 때 사용자 이름을 묻는 메시지가 표시됩니다. 550
macOS용 Git: 보안 인증 도우미를 성공적으로 구성했지만 현재 내 리포지토리에 액세스가 거부되었습니다(403). 551
Windows용 Git: Windows용 Git을 설치했지만 내 리포지토리에 액세스가 거부되었습니다(403). 554
Git 클라이언트 문제 해결 556
Git 오류: RPC failed; result=56, HTTP code = 200 fatal: 원격 엔드가 예기치 않게 끊겼습니다 556
Git 오류: 참조 업데이트 명령이 너무 많습니다 557
Git 오류: 일부 Git 버전에서 HTTPS를 통한 푸시가 손상되었습니다 557
Git 오류: 'gnutls_handshake()가 실패했습니다' 557
Git 오류: Git이 CodeCommit 리포지토리를 찾을 수 없거나 리포지토리에 액세스할 권한이 없습니다 557
Windows의 Git: 지원되는 인증 방법이 없습니다 (퍼블릭키) 558
액세스 오류 문제 해결 558
액세스 오류: Windows에서 CodeCommit 리포지토리에 연결할 때 사용자 이름과 암호를 묻는 메시지가 나타납니다. 559
액세스 오류: CodeCommit 리포지토리에 연결할 때 퍼블릭 키가 거부됩니다. 559

액세스 오류: CodeCommit 리포지토리에 연결할 때 “Rate Exceeded” 또는 “429” 메시지가 표시됩니다. ... 559
구성 오류 문제 해결 ... 560
구성 오류: macOS에서 AWS CLI 자격 증명을 구성할 수 없음 ... 561
콘솔 오류 문제 해결 ... 561
액세스 오류: 콘솔 또는에서 CodeCommit 리포지토리에 대한 암호화 키 액세스가 거부됨 AWS CLI ... 559
암호화 오류: 리포지토리를 해독할 수 없습니다. ... 562
콘솔 오류: 콘솔에서 CodeCommit 리포지토리의 코드를 검색할 수 없습니다. ... 562
디스플레이 오류: 파일을 열람하거나 파일 간 비교를 볼 수 없습니다. ... 562
트리거 문제 해결 ... 563
트리거 오류: 리포지토리 트리거가 예상대로 실행되지 않습니다 ... 563
디버깅 활성화 ... 563
CodeCommit 참조 ... 565
리전 및 Git 연결 엔드포인트 ... 565
CodeCommit AWS 리전 지원 ... 565
Git 연결 엔드포인트 ... 567
CodeCommit의 서버 지문 ... 574
인터페이스 VPC 엔드포인트 AWS CodeCommit 와 함께 사용 ... 581
가용성 ... 582
CodeCommit의 VPC 엔드포인트 생성 ... 583
CodeCommit을 위한 VPC 엔드포인트 정책 생성 ... 584
할당량 ... 585
명령줄 참조 ... 592
기본 Git 명령 ... 598
구성 변수 ... 598
원격 리포지토리 ... 599
커밋 ... 600
브랜치 ... 601
Tags ... 603
문서 기록 ... 604
이전 업데이트 ... 613
AWS 용어집 ... 619

AWS CodeCommit 는 더 이상 신규 고객이 사용할 수 없습니다. AWS CodeCommit 의 기존 고객은 정상적으로 서비스를 계속 이용할 수 있습니다. 자세히 알아보기

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.

AWS CodeCommit란 무엇인가요?

AWS CodeCommit 는 Amazon Web Services에서 호스팅하는 버전 관리 서비스로, 클라우드에서 자산(예: 문서, 소스 코드, 바이너리 파일)을 비공개로 저장하고 관리하는 데 사용할 수 있습니다. CodeCommit에 대한 요금 정보는 요금을 참조하세요.

> **Note**
>
> CodeCommit은 여러 규정 준수 프로그램의 요구 사항을 충족합니다. AWS 및 규정 준수 노력에 대한 자세한 내용은 AWS 규정 준수 프로그램 제공 범위 내 서비스를 참조하세요.
> 이것은 HIPAA 적격 서비스입니다. AWS미국 HIPAA(Health Insurance Portability and Accountability Act of 1996) 및 AWS 서비스를 사용하여 보호 대상 건강 정보(PHI)를 처리, 저장 및 전송하는 방법에 대한 자세한 내용은 HIPAA 개요를 참조하세요.
> 이 서비스 및 보안 관리 모범 사례를 지정하는 보안 관리 표준인 ISO 27001에 대한 자세한 내용은 ISO 27001 개요를 참조하세요.
> 이 서비스와 PCI DSS(Payment Card Industry Data Security Standard)에 대한 자세한 내용은 PCI DSS 개요 단원을 참조하세요.
> 이 서비스에 대한 자세한 내용 및 기밀 정보를 보호하는 암호 모듈의 보안 요건을 규정하고 있는 Federal Information Processing Standard(FIPS) Publication 140-2 US 정부 표준에 대한 정보는 Federal Information Processing Standard(FIPS) 140-2 개요 및 Git 연결 엔드포인트 섹션을 참조하세요.

주제

- CodeCommit 소개
- CodeCommit, Git 및 필요에 적합한 AWS 서비스 선택
- CodeCommit은 어떻게 작동하나요?
- CodeCommit은 Amazon S3의 파일 버전 관리와 어떻게 다른가요?
- CodeCommit은 어떻게 시작할 수 있나요?
- Git에 대해 더 학습하려면 어떻게 해야 하나요?

CodeCommit 소개

CodeCommit은 프라이빗 Git 리포지토리를 호스팅하는 안전하고 확장성 높은 소스 관리형 서비스입니다. CodeCommit을 사용하면 자체 소스 제어 시스템을 관리하거나 인프라 확장에 대해 걱정할

필요가 없습니다. CodeCommit을 사용하면 코드부터 바이너리까지 무엇이든 저장할 수 있습니다. CodeCommit은 Git의 표준 기능을 지원하므로 기존 Git 기반 도구와도 원활하게 연동됩니다.

CodeCommit을 사용하면 다음과 같은 것들이 가능합니다.

- AWS에서 호스트하는 완전 관리형 서비스의 혜택을 누릴 수 있습니다. CodeCommit은 높은 서비스 가용성과 내구성을 제공하며 하드웨어와 소프트웨어를 자체적으로 관리해야 하는 부담을 덜어 줍니다. 하드웨어를 프로비저닝 및 확장하고 서버 소프트웨어를 설치, 구성, 업데이트할 필요가 없습니다.
- 코드를 안전하게 저장할 수 있습니다. CodeCommit 리포지토리는 유휴 상태는 물론 전송 중에도 암호화됩니다.
- 코드를 공동으로 작업할 수 있습니다. CodeCommit 리포지토리는 브랜치에 병합하기 전에 사용자들이 서로 코드 변경 내용을 검토하고 그에 대한 설명을 추가할 수 있는 풀 요청 기능과 풀 요청 및 주석에 대한 이메일을 사용자에게 자동으로 전송하는 알림 기능 등을 지원합니다.
- 버전 제어 프로젝트를 손쉽게 확장할 수 있습니다. CodeCommit 리포지토리는 개발 요구 사항에 맞게 스케일 업을 할 수 있습니다. 서비스를 통해 대량 파일 또는 브랜치, 대용량 파일 크기 및 긴 개정 기록을 포함하는 리포지토리를 처리할 수 있습니다.
- 무엇이든, 언제든 저장할 수 있습니다. CodeCommit은 저장할 수 있는 리포지토리 크기 또는 파일 형식에 제한이 없습니다.
- 를 다른 AWS 및 타사 서비스와 통합합니다. CodeCommit은 리포지토리를 AWS 클라우드의 다른 프로덕션 리소스에 가깝게 유지하므로 개발 수명 주기의 속도와 빈도를 높이는 데 도움이 됩니다. IAM과 통합되며 다른 AWS 서비스와 함께 다른 리포지토리와 병렬로 사용할 수 있습니다. 자세한 내용은 와의 제품 및 서비스 통합 AWS CodeCommit 단원을 참조하십시오.
- 다른 원격 리포지토리에서 파일을 쉽게 마이그레이션할 수 있습니다. Git 기반 리포지토리에서 CodeCommit을 마이그레이션할 수 있습니다.
- 이미 알고 있는 Git 도구를 사용할 수 있습니다. CodeCommit은 Git 명령과 자체 AWS CLI 명령 및 APIs 지원합니다.

CodeCommit, Git 및 필요에 적합한 AWS 서비스 선택

CodeCommit은 Git 기반 서비스로서 대부분의 버전 관리 필요에 적합합니다. 파일 크기, 파일 형식 및 리포지토리 크기에 대한 임의적인 제한은 없습니다. 그러나 특정 종류의 작업 수행, 특히 시간에 따른 작업 수행에 부정적인 영향을 미칠 수 있는 Git에 대해서는 내재된 제약이 있습니다. 다른 AWS 서비스가 작업에 더 적합한 사용 사례에 CodeCommit 리포지토리 성능을 사용하지 않음으로써 CodeCommit 리포지토리 성능이 저하되는 것을 방지할 수 있습니다. 또한 복잡한 리포지토리에 대한 Git 성능을 최

적화할 수 있습니다. 일부 사용 사례에서는 Git, 즉 CodeCommit가 최상의 솔루션이 아닐 수 있으며 Git를 최적화하기 위해 추가적인 조치를 취해야 할 수 있습니다.

사용 사례	설명	고려해야 할 다른 서비스
자주 변경되는 대용량 파일	Git는 Delta 인코딩을 사용하여 파일 버전 사이의 차이를 저장합니다. 예를 들어 문서의 몇 단어만을 변경하는 경우, Git는 이러한 변경된 단어만을 저장합니다. 변경이 많이 가해진 5MB 이상의 파일 또는 객체가 있는 경우 Git가 Delta 차이의 상당한 체인을 다시 구성해야 할 수 있습니다. 따라서 이러한 파일이 시간의 경과에 따라 증가하면서 로컬 컴퓨터와 CodeCommit 양쪽에서 컴퓨팅 리소스의 소비량이 늘어날 수 있습니다.	대용량 파일을 버전 관리하려면, Amazon Simple Storage Service(S3)를 고려해 보세요. 자세한 내용은 Amazon Simple Storage Service 사용 설명서에서 버전 관리 사용을 참조하세요.
데이터베이스	Git 리포지토리의 크기는 시간이 흐를수록 증가합니다. 버전 관리 트랙은 모두 변경되므로 모든 변경 사항이 리포지토리 크기를 증가시킵니다. 다시 말해 데이터를 커밋할 때 커밋에서 데이터를 삭제하더라도 데이터는 리포지토리에 추가됩니다. 처리하고 시간의 경과에 따라 전송해야 할 더 많은 데이터가 있으므로 Git의 속도가 느려집니다. 이러한 현상은 특히 데이터베이스 사용 사례에 불리한 영향을 줍니다. Git는 데이터베이스로서 설계되지 않았습니다.	크기와 상관 없이 일관된 성능을 지닌 데이터베이스를 생성해 사용하려면 Amazon DynamoDB를 고려해 보세요. 자세한 내용은 Amazon DynamoDB 시작 안내서를 참조하세요.

사용 사례	설명	고려해야 할 다른 서비스
감사 추적	일반적으로 감사 추적은 긴 시간 동안 유지되며 빈도가 매우 높은 케이던스로 시스템 프로세스에 의해 지속적으로 생성됩니다. Git는 개발 주기에서 개발자 그룹에 의해 생성된 소스 코드를 안전하게 저장하도록 설계되었습니다. 프로그래밍 방식으로 생성된 시스템 변경을 지속적으로 저장하는 리포지토리를 빠르게 변경하면 시간이 흐름에 따라 성능 저하를 초래하게 됩니다.	감사 추적을 저장하려면 Amazon Simple Storage Service(S3)를 고려해 보세요. 사용 사례에 따라 AWS 활동을 감사하려면 , AWS CloudTrail AWS Config또는 Amazon CloudWatch를 사용하는 것이 좋습니다.
백업	Git는 개발자가 작성한 소스 코드 버전에 맞추어 설계되었습니다. 사용자는 백업 전략으로 CodeCommit 리포지토리를 포함한 두 개의 원격 리포지토리에 커밋을 푸시할 수 있습니다. 그러나 Git는 컴퓨터 파일 시스템, 데이터베이스 덤프 또는 유사한 백업 콘텐츠의 백업을 처리하도록 설계되지 않았습니다. 그렇게 하면 시스템의 속도를 둔화시키고 리포지토리를 복제하고 푸시하는 데 소요되는 시간이 증가할 수 있습니다.	AWS 클라우드 백업에 대한 자세한 내용은 백업 및 복원을 참조하세요.

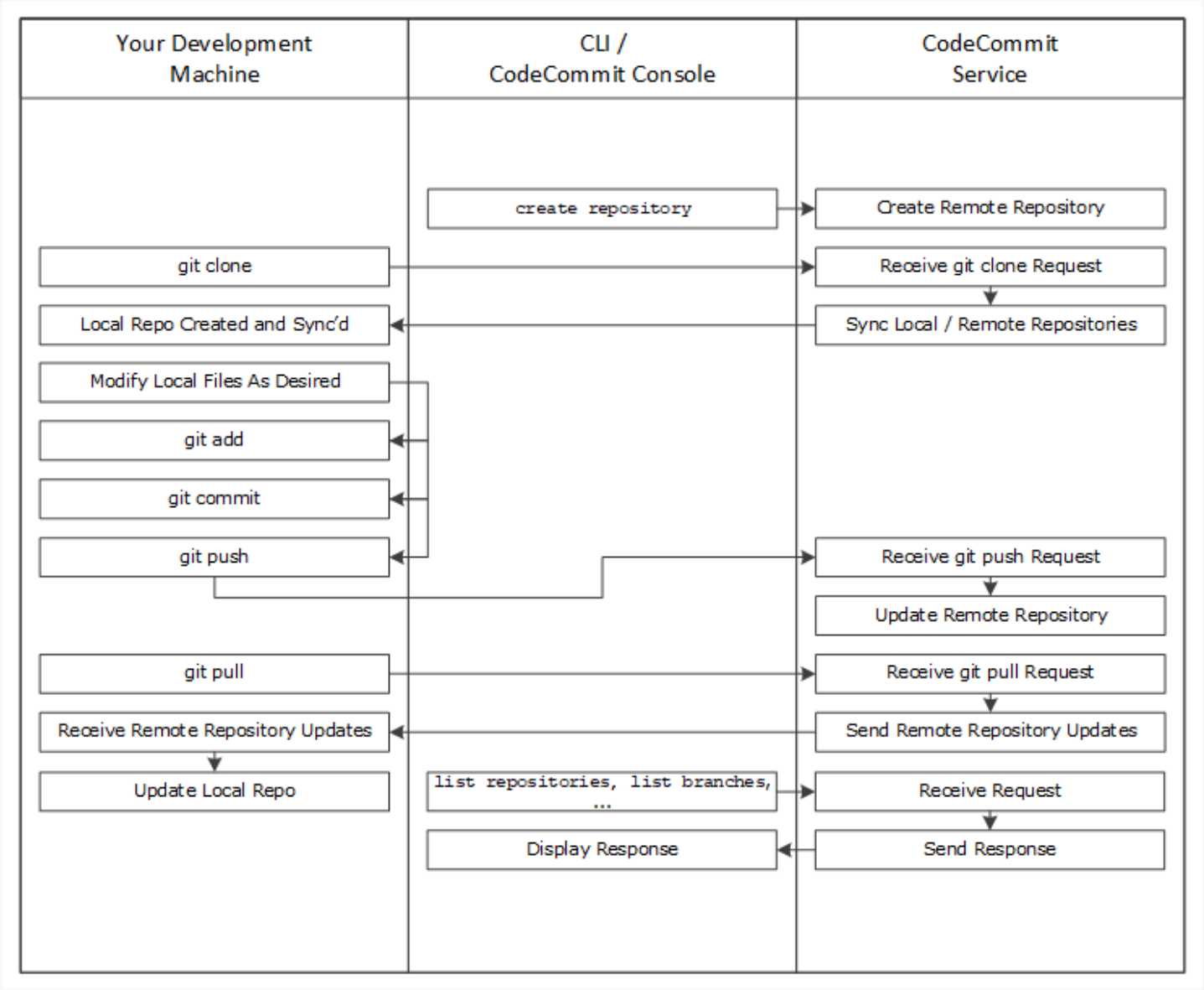
사용 사례	설명	고려해야 할 다른 서비스

다수의 브랜치 또는 참조	Git 클라이언트가 리포지토리 데이터를 푸시하거나 가져올 때 단일 브랜치에만 관심이 있더라도, 원격 서버는 모든 브랜치와 태그 등의 참조를 전송해야 합니다. 수천 개의 브랜치와 참조가 있는 경우, 이를 처리하고 전송하는 데(팩 협상) 시간이 소요될 수 있으며 명백히 느린 리포지토리 응답을 초래할 수 있습니다. 브랜치와 태그가 많을수록 이러한 프로세스가 오래 걸릴 수 있습니다. CodeCommit을 사용하되 더 이상 필요하지 않은 브랜치와 태그는 삭제하는 것이 좋습니다.	필요치 않은 브랜치와 태그를 확인하기 위해 CodeCommit 리포지토리의 참조 수를 분석하려면 다음 명령 중 하나를 사용할 수 있습니다.
		• Linux, macOS, Unix, Windows의 Bash 에뮬레이터:
		`git ls-remote | wc -l`
		• Powershell:
		`git ls-remote | Measure-Object -line`

CodeCommit은 어떻게 작동하나요?

CodeCommit은 Git 기반 리포지토리 사용자에게 익숙한 서비스이지만, 익숙하지 않은 사용자라도 비교적 간단하게 CodeCommit으로 전환할 수 있습니다. CodeCommit은 리포지토리를 쉽게 생성하고 기존 리포지토리와 브랜치를 나열할 수 있는 콘솔을 제공합니다. 로컬 리포지토리를 생성하여 간단한 몇 단계로 리포지토리에 대한 정보를 찾아서 컴퓨터에 복제할 수 있습니다. 로컬 리포지토리에서 원하는 항목을 변경한 다음 변경 내용을 CodeCommit 리포지토리로 푸시할 수 있습니다. 로컬 컴퓨터의 명령줄에서 작업하거나 GUI 기반 편집기를 사용할 수 있습니다.

다음 그림은 개발 머신, AWS CLI 또는 CodeCommit 콘솔 및 CodeCommit 서비스를 사용하여 리포지토리를 생성하고 관리하는 방법을 보여줍니다.



1. AWS CLI 또는 CodeCommit 콘솔을 사용하여 CodeCommit 리포지토리를 생성합니다.
2. 개발 시스템에서 Git을 통해 git clone을 실행하여 CodeCommit 리포지토리의 이름을 지정합니다. 그러면 CodeCommit 리포지토리에 연결되는 로컬 리포지토리가 생성됩니다.
3. 개발 시스템에서 로컬 리포지토리를 사용하여 파일을 수정(추가, 편집, 삭제)한 다음 git add를 실행하여 수정된 파일을 로컬로 스테이징합니다. git commit을 실행하여 파일을 로컬로 커밋한 다음 git push를 실행하여 파일을 CodeCommit 리포지토리로 전송합니다.
4. 다른 사용자가 변경한 내용을 다운로드합니다. git pull을 실행하여 CodeCommit 리포지토리의 파일을 로컬 리포지토리와 동기화합니다. 그러면 최신 버전의 파일을 사용하여 작업할 수 있습니다.

AWS CLI 또는 CodeCommit 콘솔을 사용하여 리포지토리를 추적하고 관리할 수 있습니다.

CodeCommit은 Amazon S3의 파일 버전 관리와 어떻게 다른가요?

CodeCommit은 팀 소프트웨어 개발용으로 최적화되었습니다. 여러 파일에 걸쳐 다른 개발자가 변경한 내용과 병렬로 발생할 수 있는 변경 사항들을 일괄 관리합니다. Amazon S3 버전 관리는 파일의 이전 버전에 대한 복구를 지원하지만, 소프트웨어 개발 팀에 필요한 협업 파일 추적 기능에는 초점을 맞추지 않습니다.

CodeCommit은 어떻게 시작할 수 있나요?

CodeCommit을 시작하려면 다음과 같이 합니다.

1. 설정 의 단계에 따라 개발 시스템을 준비합니다.
2. 시작의 자습서 하나 이상에 나오는 단계를 따르십시오.
3. CodeCommit에서 버전 관리 프로젝트를 생성하거나, 버전 관리 프로젝트를 CodeCommit으로 마이그레이션합니다.

Git에 대해 더 학습하려면 어떻게 해야 하나요?

아직도 잘 모르겠다면 Git 사용 방법 배우기를 참조하세요. 다음과 같은 몇 가지 유용한 자료가 있습니다.

- Pro Git(Pro Git 서적의 온라인 버전). 작성자: Scott Chacon. 발행: Apress
- Git Immersion(Git 사용에 대한 기초 정보를 안내하는 자습서). 발행: Neo Innovation, Inc.
- Git Reference(상세한 Git 자습서로도 사용 가능한 온라인 빠른 참조). 발행: GitHub 팀
- Git Cheat Sheet(기본 Git 명령 구문). 발행: GitHub 팀
- Git Pocket Guide. 작성자: Richard E. Silverman 발행: O'Reilly Media, Inc.

에 대한 설정 AWS CodeCommit

에 로그인하여 AWS CodeCommit 콘솔에서 직접 리포지토리에 파일을 AWS Management Console 업로드, 추가 또는 편집할 수 있습니다. ??? 이 방법을 사용하면 빠르게 변경이 가능합니다. 한편, 여러 파일이나 브랜치 전반의 파일에서 작업을 하고 싶은 경우에는 리포지토리에서 작업이 가능하도록 로컬 컴퓨터를 설정하는 것을 고려해야 합니다. CodeCommit을 설정하는 가장 쉬운 방법은 HTTPS Git 자격 증명을 구성하는 것입니다 AWS CodeCommit. 이 HTTPS 인증 방법은,

- 정적 사용자 이름 및 암호를 사용합니다.
- CodeCommit이 지원하는 모든 운영 체제에서 작동됩니다.
- 또한 Git 보안 인증 정보를 지원하는 IDE(통합 개발 환경) 및 기타 개발 도구와도 호환됩니다.

운영상 이유로 Git 보안 인증 정보를 사용하기 원치 않거나 불가능한 경우 다른 방법을 사용할 수 있습니다. 예를 들어 페더레이션 액세스, 임시 보안 인증 정보 또는 웹 자격 증명 공급자를 사용하여 CodeCommit 리포지토리에 액세스하는 경우 Git 보안 인증 정보를 사용할 수 없습니다. `git-remote-codecommit` 명령을 사용하여 로컬 컴퓨터를 설정하는 것이 좋습니다. 다른 옵션을 세심하게 읽고 가장 적합한 대안을 선택하시기 바랍니다.

- Git 보안 인증 정보를 사용하여 설정
- 다른 방법을 사용하여 설정
- CodeCommit, Git 및 기타 구성 요소에 대한 호환성

CodeCommit 및 Amazon Virtual Private Cloud를 사용하는 방법에 대해 자세히 알아보려면 인터페이스 VPC 엔드포인트 AWS CodeCommit 와 함께 사용 섹션을 참조하세요.

보안 인증 정보 보기 및 관리

AWS 콘솔에서 내 보안 자격 증명을 통해 CodeCommit 자격 증명을 보고 관리할 수 있습니다.

> **Note**
>
> 이 옵션은 페더레이션 액세스, 임시 보안 인증 정보 또는 웹 자격 증명 공급자를 사용하는 사용자는 이용할 수 없습니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/://https://https://https://https://://https://://https://://https://://https://https://://://
2. 오른쪽 상단의 탐색 모음에서 사용자 이름을 선택한 다음 보안 자격 증명을 선택합니다.
3. AWS CodeCommit 보안 인증 정보 탭을 선택합니다.

Git 보안 인증 정보를 사용하여 설정

HTTPS 연결과 Git 보안 인증 정보를 사용하여 IAM에서 정적 사용자 이름 및 암호를 생성합니다. 그런 다음 Git 사용자 이름 및 암호 인증을 지원하는 Git 및 타사 도구에서 이 보안 인증 정보들을 사용합니다. 이 방법은 대부분의 IDE 및 개발 도구에서 지원합니다. CodeCommit에서 사용할 수 있는 가장 간단하고 간편한 연결 방법입니다.

- Git 보안 인증 정보를 사용하는 HTTPS 사용자의 경우: 다음 지침에 따라 Git 보안 인증 정보를 사용하여 로컬 컴퓨터와 CodeCommit 리포지토리 간 연결을 설정합니다.
- 개발 도구에서 연결: 다음 지침에 따라 Git 보안 인증 정보를 사용하여 IDE 또는 기타 개발 도구와 CodeCommit 리포지토리 간 연결을 설정합니다. Git 자격 증명을 지원하는 IDEs에는 Visual Studio, Xcode 및 IntelliJ가 포함됩니다(이에 국한되지 않음).

다른 방법을 사용하여 설정

HTTPS 대신 SSH 프로토콜을 사용하여 CodeCommit 리포지토리에 연결할 수 있습니다. SSH 연결을 사용하여 Git 및 CodeCommit에서 SSH 인증에 사용하는 퍼블릭 및 프라이빗 키 파일을 로컬 시스템에서 생성합니다. 퍼블릭 키를 IAM 사용자와 연결합니다. 로컬 컴퓨터에 프라이빗 키를 저장합니다. SSH는 퍼블릭 및 프라이빗 키 파일을 수동으로 생성 및 관리해야 하므로 Git 보안 인증 정보가 CodeCommit에서 사용하기 더 간단하고 간편할 수 있습니다.

Git 보안 인증 정보와 달리 SSH 연결 설정은 로컬 컴퓨터의 운영 체제에 따라 달라집니다.

- 를 사용하지 않는 SSH 사용자의 경우 AWS CLI: 이미 퍼블릭-프라이빗 키 페어가 있고 로컬 컴퓨터에서의 SSH 연결에 익숙한 경우 다음의 간략한 지침을 따릅니다.
- Linux, macOS, Unix에서 SSH 연결: Linux, macOS, Unix 운영 체제에서 퍼블릭-프라이빗 키 페어를 생성하고 연결을 설정하는 단계별 과정은 다음 지침을 따릅니다.
- Windows에서 SSH 연결: Windows 운영 체제에서 퍼블릭-프라이빗 키 페어를 생성하고 연결을 설정하는 단계별 과정은 다음 지침을 따릅니다.

CodeCommit에 연결하고 페더레이션 액세스, 자격 증명 공급자 또는 임시 자격 증명을 AWS 사용하거나 IAM 사용자 또는 IAM 사용자를 위한 Git 자격 증명을 구성하지 않으려면 다음 두 가지 방법 중 하나로 CodeCommit 리포지토리에 대한 연결을 설정할 수 있습니다.

- git-remote-codecommit를 설치하고 사용합니다(권장).
- AWS CLI에 포함된 보안 인증 도우미를 설치하고 사용합니다.

두 방법 모두 IAM 사용자 없이도 CodeCommit 리포지토리에 액세스할 수 있습니다. 즉, 페더레이션 액세스 및 보안 인증 정보를 사용하여 리포지토리에 연결할 수 있습니다. git-remote-codecommit 유틸리티가 권장되는 접근 방식입니다. Git를 확장하고 다양한 Git 버전 및 보안 인증 도우미와 호환됩니다. 그러나 모든 IDE가 `git-remote-codecommit`에서 사용하는 복제 URL 형식을 지원하는 것은 아닙니다. IDE에서 리포지토리를 사용하기 전에 로컬 컴퓨터에 수동으로 리포지토리를 복제해야 할 수 있습니다.

- git-remote-codecommit을 사용하여 AWS CodeCommit 리포지토리에 대한 HTTPS 연결을 위한 설정 단계의 지침에 따라 Windows, Linux, macOS 또는 Unixgit-remote-codecommit에를 설치하고 설정합니다.

에 포함된 자격 증명 헬퍼 AWS CLI 를 사용하면 Git이 CodeCommit 리포지토리와 상호 작용하기 위해와 인증해야 할 때마다 HTTPS와 암호화 방식으로 서명된 버전의 IAM 사용자 자격 증명 또는 Amazon EC2 인스턴스 역할을 사용할 AWS 수 있습니다. 일부 운영 체제 및 Git 버전은 자체 보안 인증 도우미를 제공하는데, 이러한 도우미가 AWS CLI에 포함된 보안 인증 도우미와 충돌을 일으킵니다. 이로 인해 CodeCommit 연결 문제가 발생할 수 있습니다.

- AWS CLI 자격 증명 헬퍼를 사용하여 Linux, macOS 또는 Unix에서 HTTPS 연결: Linux, macOS, Unix 시스템에서 보안 인증 도우미를 설치 및 생성하는 단계별 과정은 다음 지침을 따릅니다.
- 자격 AWS CLI 증명 헬퍼를 사용하여 Windows에서 HTTPS 연결: Windows 시스템에서 보안 인증 도우미를 설치 및 설정하는 단계별 과정은 다음 지침을 따릅니다.

다른 Amazon Web Services 계정에 호스팅된 CodeCommit 리포지토리에 연결하려는 경우에는 역할, 정책, 그리고 AWS CLI에 포함된 보안 인증 도우미 등을 사용하여 액세스를 구성하고 연결을 설정할 수 있습니다.

- 역할을 사용하여 AWS CodeCommit 리포지토리에 대한 교차 계정 액세스 구성: 한 Amazon Web Services 계정에서 다른 Amazon Web Services 계정의 IAM 그룹 사용자에 대한 크로스 계정 액세스를 구성하는 단계별 과정은 다음 지침을 따릅니다.

CodeCommit, Git 및 기타 구성 요소에 대한 호환성

CodeCommit으로 작업할 때 Git을 사용합니다. 다른 프로그램도 사용할 수 있습니다. 다음 표는 버전 호환성에 대한 최신 정보를 제공합니다. 가장 좋은 방법은 최신 버전의 Git 및 기타 소프트웨어를 사용하는 것입니다.

에 대한 버전 호환성 정보 AWS CodeCommit

구성 요소	버전
Git	CodeCommit은 Git 버전 1.7.9 이상을 지원합니다. Git 버전 2.28에서는 초기 커밋을 위한 브랜드 이름을 구성할 수 있습니다. 최신 버전의 Git를 사용하는 것이 좋습니다.
Curl	CodeCommit에는 curl 7.33 이상이 필요합니다. 단, HTTPS 및 curl 업데이트 7.41.0에는 알려진 문제가 있습니다. 자세한 내용은 문제 해결 단원을 참조하십시오.
Python(git-remote-codecommit 전용)	git-remote-codecommit에는 버전 3 이상이 필요합니다.
Pip(git-remote-codecommit 전용)	git-remote-codecommit에는 버전 9.0.3 이상이 필요합니다.
AWS CLI (git-remote-codecommit만 해당)	모든 CodeCommit 사용자에게 최신 AWS CLI 버전의 버전 2를 사용하는 것이 좋습니다. git-remote-codecommit에서는 연동 사용자와 같이 임시 자격 증명이 필요한 SSO 및 연결을 지원하기 AWS 위해 AWS CLI 버전 2가 필요합니다.

Git 보안 인증 정보를 사용하는 HTTPS 사용자를 위한 설정

AWS CodeCommit 리포지토리에 대한 연결을 설정하는 가장 간단한 방법은 IAM 콘솔에서 CodeCommit에 대한 Git 자격 증명을 구성한 다음 HTTPS 연결에 해당 자격 증명을 사용하는 것입니다. 이 보안 인증 정보는 사용자 이름 및 암호를 사용한 HTTPS 인증을 지원하는 타사 도구 또는 통합 개발 환경(IDE)에서도 사용할 수 있습니다. 예시는 개발 도구에서 연결 섹션을 참조하세요.

> **Note**
>
> 이전에 로컬 컴퓨터에서 CodeCommit에 보안 인증 도우미를 사용하도록 구성하였다면 Git 보안 인증 정보를 사용하기 전에 먼저 .gitconfig 파일을 편집하여 이 파일에서 보안 인증 도우미 정보를 제거해야 합니다. 로컬 컴퓨터에서 macOS를 실행 중인 경우에는 Keychain Access에서 캐시된 보안 인증 정보를 지워야 할 수도 있습니다.

1단계: CodeCommit에 대한 초기 구성

이 절차에 따라 Amazon Web Services 계정을 설정하고 IAM 사용자를 생성하며 CodeCommit에 대한 액세스를 구성합니다.

CodeCommit에 액세스할 IAM 사용자를 생성 및 구성하려면

1. http://aws.amazon.com에서 가입을 선택하여 Amazon Web Services 계정을 만듭니다.
2. Amazon Web Services 계정에서 IAM 사용자를 생성하거나 기존 사용자를 사용합니다. 액세스 키 ID가 있고 해당 IAM 사용자와 연결된 비밀 액세스 키가 있는지 확인합니다. 자세한 내용은 Amazon Web Services 계정에서 IAM 사용자 생성하기를 참조하세요.

   > **Note**
   >
   > CodeCommit에는가 필요합니다 AWS Key Management Service. 기존 IAM 사용자를 사용하는 경우 CodeCommit에 필요한 AWS KMS 작업을 명시적으로 거부하는 정책이 사용자에게 연결되어 있지 않은지 확인합니다. 자세한 내용은 AWS KMS 및 암호화 단원을 참조하십시오.

3. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/://https://https://https://https://://https://https://://https://://https://://https://://https://://http
4. IAM 콘솔의 탐색 창에서 사용자를 선택한 다음 CodeCommit 액세스에 대해 구성할 IAM 사용자를 선택합니다.
5. 권한 탭에서 권한 추가를 선택합니다.
6. 권한 부여에서 기존 정책 직접 연결을 선택합니다.
7. 정책 목록에서, CodeCommit 액세스에 대한 AWSCodeCommitPowerUser 또는 다른 관리형 정책을 선택합니다. 자세한 내용은 AWS CodeCommit에 대한 관리형 정책 단원을 참조하십시오.

연결할 정책을 선택한 후 다음: 검토를 선택하여 IAM 사용자에게 연결할 정책의 목록을 검토합니다. 목록이 올바르면 권한 추가를 선택합니다.

CodeCommit 관리형 정책, 그리고 리포지토리에 대한 액세스를 다른 그룹 및 사용자와 공유하는 방법에 대해 자세히 알아보려면 리포지토리 공유, AWS CodeCommit에 대한 인증 및 액세스 제어 섹션을 참조하세요.

CodeCommit에서 AWS CLI 명령을 사용하려면를 설치합니다 AWS CLI. CodeCommit AWS CLI 에서를 사용하기 위한 프로파일을 생성하는 것이 좋습니다. 자세한 내용은 명령줄 참조 및 명명된 프로필 사용을 참조하세요.

2단계: Git 설치

CodeCommit 리포지토리에서 파일과 커밋 및 기타 정보를 사용하려면 로컬 시스템에 Git를 설치해야 합니다. CodeCommit은 Git 버전 1.7.9 이상을 지원합니다. Git 버전 2.28에서는 초기 커밋을 위한 브랜드 이름을 구성할 수 있습니다. 최신 버전의 Git를 사용하는 것이 좋습니다.

Git 설치를 위해서는 Git 다운로드와 같은 웹 사이트를 권장합니다.

> **Note**
>
> Git은 계속 개선되며 정기적으로 업데이트되는 플랫폼입니다. 기능 변경에 따라 CodeCommit 사용 방식이 달라지는 경우가 있습니다. Git과 CodeCommit의 특정 버전에서 문제가 발생할 경우 문제 해결의 내용을 검토하세요.

3단계: CodeCommit에 대한 HTTPS 접속을 위한 Git 보안 인증 정보 생성

Git을 설치한 후에는 IAM에 IAM 사용자를 위한 Git 보안 인증 정보를 생성합니다.

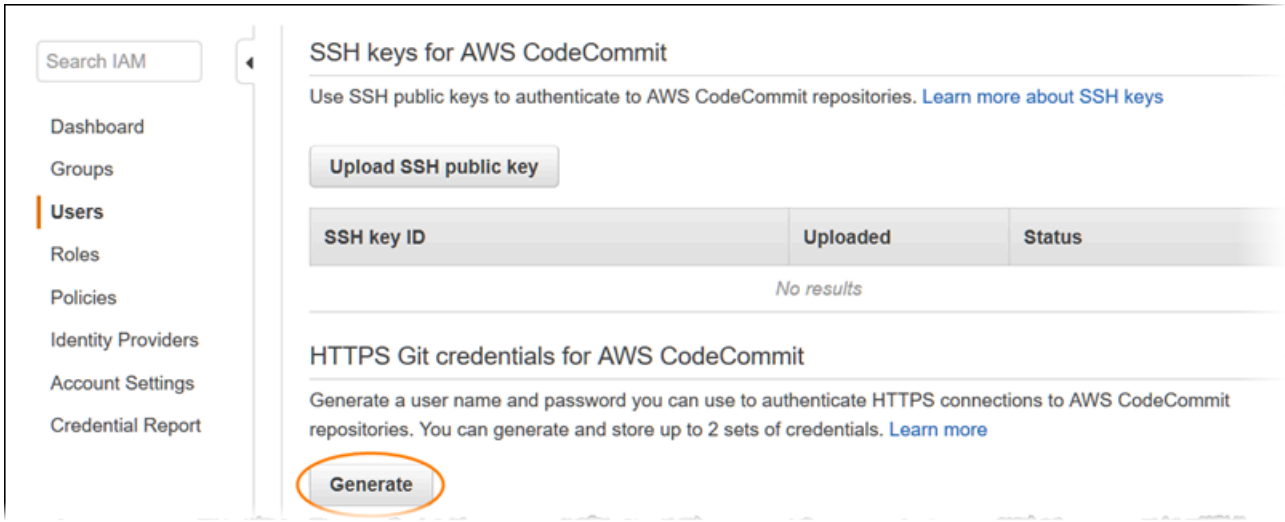
CodeCommit을 위한 HTTPS Git 보안 인증 정보를 설정하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/://https://https://https://://https://://https://://https://://https://://https://://https://://https

   CodeCommit에 연결하기 위한 Git 보안 인증 정보를 생성하고 사용할 IAM 사용자로 로그인해야 합니다.
2. IAM 콘솔의 탐색 창에서 사용자를 선택하고 사용자 목록에서 해당 IAM 사용자를 선택합니다.

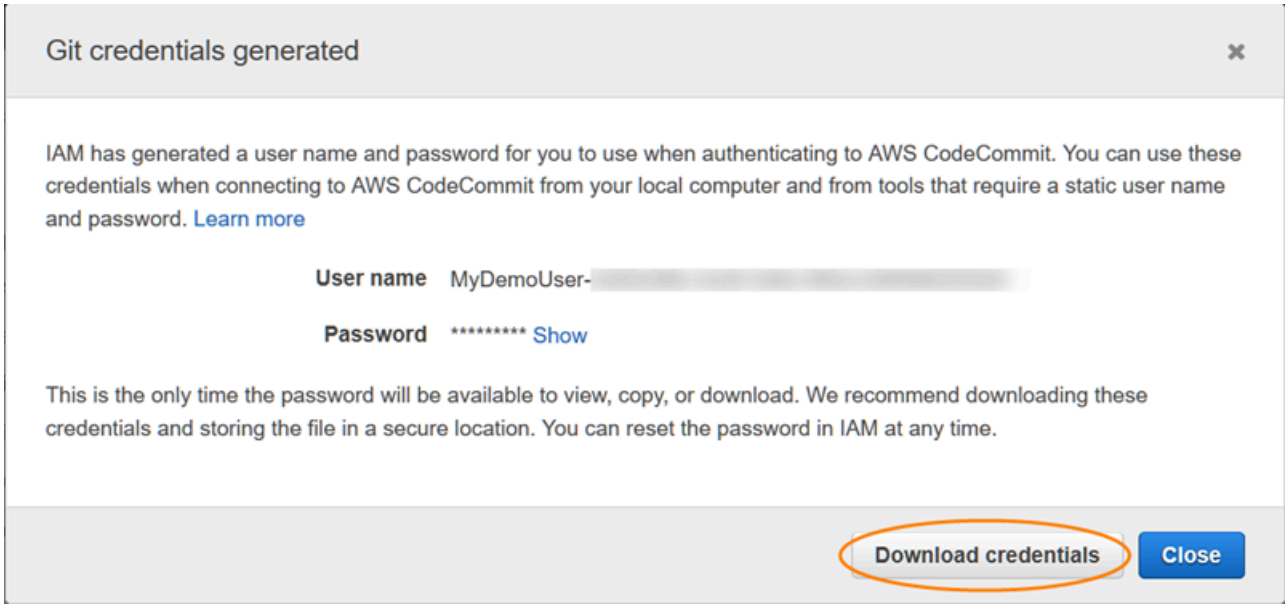
> **Note**
>
> 내 보안 인증 정보에서 CodeCommit 보안 인증 정보를 직접 보고 관리할 수 있습니다. 자세한 내용은 보안 인증 정보 보기 및 관리 단원을 참조하십시오.

3. 사용자 세부 정보 페이지에서 보안 자격 증명 탭을 선택하고 HTTPS Git 자격 증명 AWS CodeCommit에서 생성을 선택합니다.



> **Note**
>
> Git 보안 인증 정보에 대해 본인의 사용자 이름 또는 암호를 선택할 수 없습니다. 이에 관한 자세한 내용은 CodeCommit으로 Git 보안 인증 정보 및 HTTPS 사용을 참조하세요.

4. IAM이 생성한 사용자 이름과 암호를 복제해 둡니다. 이 정보를 표시해 복사한 다음 로컬 컴퓨터의 안전한 파일에 붙여넣기하거나, 아니면 보안 인증 정보 다운로드를 선택해 .CSV 파일로 다운로드합니다. CodeCommit에 접속하려면 이 정보가 필요합니다.



보안 인증 정보를 저장한 후 닫기를 선택합니다.

> **Important**
>
> 이때가 사용자 이름과 암호를 저장할 수 있는 유일한 기회입니다. 이 정보를 저장하지 않는 경우, 사용자 이름은 IAM 콘솔에서 복사할 수 있지만 암호는 찾을 수 없습니다. 그러므로 암호를 재설정한 후 저장해야 합니다.

4단계: CodeCommit 콘솔 연결 및 리포지토리 복제

관리자가 CodeCommit 리포지토리에 대한 이름과 연결 세부 정보를 이미 전송한 경우 이 단계를 건너뛰고 리포지토리를 직접 복제할 수 있습니다.

CodeCommit 리포지토리에 연결하려면

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리전 선택기에서 리포지토리가 생성된 AWS 리전 를 선택합니다. 리포지토리는에 고유합니다 AWS 리전. 자세한 내용은 리전 및 Git 연결 엔드포인트 단원을 참조하십시오.
3. 목록에서 연결하려는 리포지토리를 찾아서 선택합니다. URL 복제를 선택한 다음 리포지토리를 복제하거나 연결할 때 사용할 프로토콜을 선택합니다. 이것으로 복제 URL가 복사됩니다.
   - IAM 사용자를 통한 Git 보안 인증 정보를 활용하거나 AWS CLI에 포함된 보안 인증 도우미를 사용하는 경우 HTTPS URL을 복사합니다.

- 로컬 컴퓨터에서 git-remote-codecommit 명령을 사용하는 경우 HTTPS(GRC) URL을 복사합니다.
- IAM 사용자와 SSH 퍼블릭/프라이빗 키 페어를 사용하는 경우 SSH URL을 복사합니다.

> **Note**
>
> 리포지토리 목록 대신 시작 페이지가 표시되면 로그인한에 AWS 계정과 연결된 AWS 리전 리포지토리가 없는 것입니다. 리포지토리를 만들려면 the section called "리포지토리 생성"을(를) 참조하거나 Git 및 CodeCommit 시작하기 자습서의 다음 단계를 따릅니다.

4. 터미널, 명령줄 또는 Git 셸을 엽니다. 복사한 HTTPS 복제 URL을 사용하여 git clone 명령을 실행하여 리포지토리를 복제합니다. 예를 들어, *MyDemoRepo*라는 리포지토리를 미국 동부(오하이오) 리전의 *my-demo-repo*라는 로컬 리포지토리에 복제하려면 다음과 같이 합니다.

```
git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-
demo-repo
```

처음 접속하는 경우에는 리포지토리에 대한 사용자 이름과 암호를 묻는 메시지가 표시됩니다. 로컬 컴퓨터의 구성에 따라 이 메시지는 운영 체제에 대한 보안 인증 정보 관리 시스템이나 해당 Git 버전의 보안 인증 정보 관리자 유틸리티(예: Windows용 Git에 포함된 Git 보안 인증 정보 관리자), 해당 IDE 또는 Git 자체에서 출력됩니다. IAM에서 Git 보안 인증 정보에 대해 생성한 사용자 이름과 암호를 입력합니다(3단계: CodeCommit에 대한 HTTPS 접속을 위한 Git 보안 인증 정보 생성에서 생성한 것). 해당 운영 체제 및 기타 소프트웨어에 따라 이 정보는 보안 인증 정보 스토어 또는 보안 인증 정보 관리 유틸리티에 저장됩니다. 일단 저장되면 IAM에서 암호를 변경하거나 Git 보안 인증 정보을 비활성화하거나 Git 보안 인증 정보를 삭제하지 않는 한 이 정보를 입력하라는 메시지가 다시 표시되지 않습니다.

로컬 컴퓨터에 보안 인증 정보 스토어 또는 보안 인증 정보 관리 유틸리티를 구성하지 않은 경우, 이를 설치할 수 있습니다. Git과 Git의 보안 인증 정보 관리 방식에 대한 자세한 내용은 Git 설명서의 보안 인증 정보 스토리지를 참조하세요.

자세한 내용은 리포지토리를 복제하여 CodeCommit 리포지토리에 연결 및 커밋 생성 단원을 참조하세요.

다음 단계

사전 필수 단계를 완료했습니다. CodeCommit 시작하기 에 나와 있는 단계에 따라 CodeCommit의 사용을 시작합니다.

첫 커밋을 생성하고 푸시하는 방법에 대해 알아보려면 에서 커밋 생성 AWS CodeCommit 단원을 참조하세요. Git를 처음 사용하는 경우, Git에 대해 더 학습하려면 어떻게 해야 하나요? 및 Git 및 시작하기 AWS CodeCommit의 정보를 검토하는 것이 좋습니다.

git-remote-codecommit을 AWS CodeCommit 사용하여에 HTTPS 연결을 위한 설정 단계

루트 계정, 페더레이션 액세스 또는 임시 보안 인증 정보를 사용하여 CodeCommit에 연결하려는 경우, git-remote-codecommit을 사용하여 액세스를 설정해야 합니다. 이 유틸리티는 Git를 확장하여 CodeCommit 리포지토리에서 코드를 푸시하고 풀하는 간단한 방법을 제공합니다. 페더레이션 액세스, 자격 증명 공급자 및 임시 보안 인증 정보로 이루어진 연결을 지원하는 데 권장되는 방법입니다. 페더레이션 자격 증명 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 페더레이션 자격 증명이 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 관련 역할에 대한 자세한 내용은 IAM 사용 설명서의 Create a role for a third-party identity provider (federation)를 참조하세요. IAM Identity Center를 사용하는 경우, 권한 집합을 구성합니다. 인증 후 ID가 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 집합을 IAM의 역할과 연관짓습니다. 권한 집합에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 집합을 참조하세요.

git-remote-codecommit을 IAM 사용자와 함께 사용할 수도 있습니다. 다른 HTTPS 연결 방법과 달리 git-remote-codecommit에서는 사용자에 대한 Git 보인 인증 정보를 설정할 필요가 없습니다.

> **Note**
>
> 일부 IDE는 `git-remote-codecommit`에서 사용하는 복제 URL 형식을 지원하지 않습니다. 원하는 IDE에서 리포지토리를 사용하려면 먼저 로컬 컴퓨터에 리포지토리를 수동으로 복제해야 할 수 있습니다. 자세한 내용은 git-remote-codecommit 및 문제 해결 AWS CodeCommit 단원을 참조하십시오.

이 절차들은 Amazon Web Services 계정이 있고 CodeCommit에 리포지토리를 하나 이상 생성했으며 CodeCommit 리포지토리에 연결할 때 관리형 정책이 있는 IAM 사용자를 사용한다는 가정 하에 작성되었습니다. 연합된 사용자 및 기타 교체 자격 증명 유형에 대한 액세스를 구성하는 방법에 대한 자세한 내용은 교체 자격 증명을 사용하여 AWS CodeCommit 리포지토리에 연결 단원을 참조하세요.

주제

- 0단계: git-remote-codecommit에 대한 사전 조건 설치
- 1단계: CodeCommit에 대한 초기 구성
- 2단계: git-remote-codecommit 설치
- 3단계: CodeCommit 콘솔 연결 및 리포지토리 복제
- 다음 단계

0단계: git-remote-codecommit에 대한 사전 조건 설치

git-remote-codecommit를 사용하려면 먼저 로컬 컴퓨터에 몇 가지 사전 조건을 설치해야 합니다. 다음이 포함됩니다.

- 지원되는 Python 버전입니다. 지원되는 Python 버전에 대한 자세한 내용은 git-remote-codecommit을 참조하세요.

  git-remote-codecommit 설정 및 사용에 대한 자세한 내용은 git-remote-codecommit을 AWS CodeCommit 사용하여에 HTTPS 연결을 위한 설정 단계 단원을 참조하세요.
- Git

> **Note**
>
> Windows에서 Python을 설치할 때 경로에 Python을 추가하는 옵션을 선택해야 합니다.

git-remote-codecommit에는 pip 버전 9.0.3 이상이 필요합니다. pip 버전을 확인하려면 터미널 또는 명령줄을 열고 다음 명령을 실행합니다.

```
pip --version
```

다음 두 명령을 실행하여 pip 버전을 최신 버전으로 업데이트할 수 있습니다.

```
curl -O https://bootstrap.pypa.io/get-pip.py
python3 get-pip.py --user
```

CodeCommit 리포지토리에서 파일과 커밋 및 기타 정보를 사용하려면 로컬 시스템에 Git를 설치해야 합니다. CodeCommit은 Git 버전 1.7.9 이상을 지원합니다. Git 버전 2.28에서는 초기 커밋을 위한 브랜드 이름을 구성할 수 있습니다. 최신 버전의 Git를 사용하는 것이 좋습니다.

Git 설치를 위해서는 Git 다운로드와 같은 웹 사이트를 권장합니다.

> **Note**
>
> Git은 계속 개선되며 정기적으로 업데이트되는 플랫폼입니다. 기능 변경에 따라 CodeCommit 사용 방식이 달라지는 경우가 있습니다. Git과 CodeCommit의 특정 버전에서 문제가 발생할 경우 문제 해결의 내용을 검토하세요.

1단계: CodeCommit에 대한 초기 구성

다음 단계에 따라 IAM 사용자를 생성하고, 적절한 정책으로 구성하고, 액세스 키와 보안 키를 얻고, AWS CLI를 설치 및 구성합니다.

CodeCommit에 액세스할 IAM 사용자를 생성 및 구성하려면

1. http://aws.amazon.com에서 가입을 선택하여 Amazon Web Services 계정을 만듭니다.
2. Amazon Web Services 계정에서 IAM 사용자를 생성하거나 기존 사용자를 사용합니다. 액세스 키 ID가 있고 해당 IAM 사용자와 연결된 비밀 액세스 키가 있는지 확인합니다. 자세한 내용은 Amazon Web Services 계정에서 IAM 사용자 생성하기를 참조하세요.

   > **Note**
   >
   > CodeCommit에는가 필요합니다 AWS Key Management Service. 기존 IAM 사용자를 사용하는 경우 CodeCommit에 필요한 AWS KMS 작업을 명시적으로 거부하는 정책이 사용자에게 연결되어 있지 않은지 확인합니다. 자세한 내용은 AWS KMS 및 암호화 단원을 참조하십시오.

3. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/://https://https://https://https://https://https://://https://://https://://https://https://https://://
4. IAM 콘솔의 탐색 창에서 사용자를 선택한 다음 CodeCommit 액세스에 대해 구성할 IAM 사용자를 선택합니다.
5. 권한 탭에서 권한 추가를 선택합니다.
6. 권한 부여에서 기존 정책 직접 연결을 선택합니다.

7. 정책 목록에서, CodeCommit 액세스에 대한 AWSCodeCommitPowerUser 또는 다른 관리형 정책을 선택합니다. 자세한 내용은 AWS CodeCommit에 대한 관리형 정책 단원을 참조하십시오.

   연결할 정책을 선택한 후 다음: 검토를 선택하여 IAM 사용자에게 연결할 정책의 목록을 검토합니다. 목록이 올바르면 권한 추가를 선택합니다.

   CodeCommit 관리형 정책, 그리고 리포지토리에 대한 액세스를 다른 그룹 및 사용자와 공유하는 방법에 대해 자세히 알아보려면 리포지토리 공유, AWS CodeCommit에 대한 인증 및 액세스 제어 섹션을 참조하세요.

설치 및 구성 AWS CLI

1. 로컬 시스템에서를 다운로드하여 설치합니다 AWS CLI. 이는 명령줄에서 CodeCommit과 상호 작용하기 위한 사전 조건입니다. AWS CLI 버전 2를 설치하는 것이 좋습니다. 의 최신 메이저 버전이며 모든 최신 기능을 AWS CLI 지원합니다. 에서 루트 계정, 페더레이션 액세스 또는 임시 자격 증명 사용을 AWS CLI 지원하는 유일한 버전의 입니다git-remote-codecommit.

   자세한 내용은 AWS 명령줄 인터페이스 설정 단원을 참조하십시오.

   > **Note**
   >
   > CodeCommit은 AWS CLI 버전 1.7.38 이상에서만 작동합니다. 가장 좋은 방법은를 설치하거나 사용 가능한 최신 버전으로 업그레이드 AWS CLI 하는 것입니다. 설치 AWS CLI 한 버전을 확인하려면 aws --version 명령을 실행합니다.
   > 의 이전 버전을 AWS CLI 최신 버전으로 업그레이드하려면 설치를 참조하세요 AWS Command Line Interface.

2. 이 명령을 실행하여에 대한 CodeCommit 명령 AWS CLI 이 설치되어 있는지 확인합니다.

   ```
   aws codecommit help
   ```

   이 명령은 CodeCommit 명령 목록을 반환합니다.

3. 다음과 같이 configure 명령을 사용하여 프로파일 AWS CLI 로를 구성합니다.

   ```
   aws configure
   ```

   메시지가 표시되면 CodeCommit과 함께 사용할 IAM 사용자의 AWS 액세스 키와 AWS 보안 액세스 키를 지정합니다. 또한와 같이 리포지토리 AWS 리전 가 있는를 지정해야 합니다`us-east-2`.

기본 출력 형식을 묻는 메시지가 표시되면 `json`을 지정합니다. 예를 들어, IAM 사용자의 프로필을 구성하는 경우에는 다음과 같이 합니다.

```
AWS Access Key ID [None]: Type your IAM user AWS access key ID here, and then press
 Enter
AWS Secret Access Key [None]: Type your IAM user AWS secret access key here, and
 then press Enter
Default region name [None]: Type a supported region for CodeCommit here, and then
 press Enter
Default output format [None]: Type json here, and then press Enter
```

에서 사용할 프로필을 생성하고 구성하는 방법에 대한 자세한 내용은 다음을 AWS CLI참조하세요.

- 명명된 프로파일
- 에서 IAM 역할 사용 AWS CLI
- 설정 명령
- 교체 자격 증명을 사용하여 AWS CodeCommit 리포지토리에 연결

리포지토리 또는 다른의 리소스에 연결하려면 기본 리전 이름으로 AWS CLI 를 재구성 AWS 리전해야 합니다. CodeCommit에서 지원되는 기본 리전 이름은 다음과 같습니다.

- us-east-2
- us-east-1
- eu-west-1
- us-west-2
- ap-northeast-1
- ap-southeast-1
- ap-southeast-2
- ap-southeast-3
- me-central-1
- eu-central-1
- ap-northeast-2
- sa-east-1
- us-west-1

- eu-west-2
- ap-south-1
- ap-south-1
- ca-central-1
- us-gov-west-1
- us-gov-east-1
- eu-north-1
- ap-east-1
- me-south-1
- cn-north-1
- cn-northwest-1
- eu-south-1
- ap-northeast-3
- af-south-1
- il-central-1

CodeCommit 및에 대한 자세한 내용은 섹션을 AWS 리전참조하세요리전 및 Git 연결 엔드포인트. IAM, 액세스 키, 비밀 키에 대한 자세한 내용을 확인하려면 보안 인증 정보 얻는 방법 및 IAM 사용자의 액세스 키 관리를 참조하세요. AWS CLI 및 프로파일에 대한 자세한 내용은 명명된 프로파일을 참조하세요.

2단계: git-remote-codecommit 설치

다음 단계에 따라 git-remote-codecommit를 설치합니다.

git-remote-codecommit를 설치하려면

1. 터미널 또는 명령줄에서 다음 명령을 실행합니다.

```
pip install git-remote-codecommit
```

> **Note**
>
> 운영 체제 및 구성에 따라 사용자는 sudo와 같은 승격된 권한으로 이 명령을 실행해야 하거나, 아니면 --user 파라미터를 사용하여 현재 사용자 계정과 같이 특별한 권한이 필요하지 않은 디렉터리에 설치해야 할 수 있습니다. 예를 들어, Linux, macOS, Unix를 실행하는 컴퓨터는 다음과 같습니다.
>
> ```
> sudo pip install git-remote-codecommit
> ```
>
> Windows를 실행하는 컴퓨터는 다음과 같습니다.
>
> ```
> pip install --user git-remote-codecommit
> ```

2. 성공 메시지가 나타날 때까지 설치 프로세스를 모니터링합니다.

3단계: CodeCommit 콘솔 연결 및 리포지토리 복제

관리자가 CodeCommit 리포지토리에 git-remote-codecommit과 함께 사용할 복제 URL을 이미 전송한 경우에는 콘솔 연결을 건너뛰고 리포지토리를 직접 복제할 수 있습니다.

CodeCommit 리포지토리에 연결하려면

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리전 선택기에서 리포지토리가 생성된 AWS 리전 를 선택합니다. 리포지토리는에 고유합니다 AWS 리전. 자세한 내용은 리전 및 Git 연결 엔드포인트 단원을 참조하십시오.
3. 목록에서 연결하려는 리포지토리를 찾아서 선택합니다. URL 복제를 선택한 다음 리포지토리를 복제하거나 연결할 때 사용할 프로토콜을 선택합니다. 이것으로 복제 URL가 복사됩니다.
   - IAM 사용자를 통한 Git 보안 인증 정보를 활용하거나 AWS CLI에 포함된 보안 인증 도우미를 사용하는 경우 HTTPS URL을 복사합니다.
   - 로컬 컴퓨터에서 git-remote-codecommit 명령을 사용하는 경우 HTTPS(GRC) URL을 복사합니다.
   - IAM 사용자와 SSH 퍼블릭/프라이빗 키 페어를 사용하는 경우 SSH URL을 복사합니다.

> **Note**
>
> 리포지토리 목록 대신 시작 페이지가 표시되면 로그인 AWS 리전 한에 AWS 계정과 연결된 리포지토리가 없는 것입니다. 리포지토리를 만들려면 the section called "리포지토리 생성"을(를) 참조하거나 Git 및 CodeCommit 시작하기 자습서의 다음 단계를 따릅니다.

4. 터미널 또는 명령 프롬프트에서 git clone 명령을 사용하여 리포지토리를 복제합니다. 명명된 AWS CLI 프로필을 생성한 경우 복사한 HTTPS git-remote-codecommit URL과 프로필 이름을 사용합니다. 프로파일을 지정하지 않으면 명령이 기본 프로파일을 가정합니다. 그러면 명령을 실행한 디렉터리의 하위 디렉터리에 로컬 리포지토리가 생성됩니다. 예를 들어, *MyDemoRepo*라는 리포지토리를 *my-demo-repo*라는 로컬 리포지토리에 복제하려면 다음과 같이 합니다.

```
git clone codecommit://MyDemoRepo my-demo-repo
```

*CodeCommitProfile*이라는 프로파일을 사용하여 동일한 리포지토리를 복제하려면:

```
git clone codecommit://CodeCommitProfile@MyDemoRepo my-demo-repo
```

프로파일에 구성된 리포지토리 AWS 리전 와 다른에 리포지토리를 복제하려면 AWS 리전 이름을 포함합니다. 예시:

```
git clone codecommit::ap-northeast-1://MyDemoRepo my-demo-repo
```

다음 단계

사전 필수 단계를 완료했습니다. CodeCommit 시작하기 에 나와 있는 단계에 따라 CodeCommit의 사용을 시작합니다.

첫 커밋을 생성하고 푸시하는 방법에 대해 알아보려면 에서 커밋 생성 AWS CodeCommit 단원을 참조하세요. Git를 처음 사용하는 경우, Git에 대해 더 학습하려면 어떻게 해야 하나요? 및 Git 및 시작하기 AWS CodeCommit의 정보를 검토하는 것이 좋습니다.

Git 보안 인증을 사용하여 개발 도구에서 연결 설정

IAM 콘솔 AWS CodeCommit 에서에 대한 Git 자격 증명을 구성한 후 Git 자격 증명을 지원하는 모든 개발 도구와 함께 해당 자격 증명을 사용할 수 있습니다. 예를 들어, Visual Studio, Xcode AWS Cloud9, IntelliJ 또는 Git 자격 증명을 통합하는 통합 개발 환경(IDE)에서 CodeCommit 리포지토리에 대한 액세스를 구성할 수 있습니다. 액세스를 구성한 후 코드 편집, 변경 내용 커밋, IDE 또는 기타 개발 도구에서 직접 푸시할 수 있습니다.

> **Note**
>
> 패더레이션 액세스, 임시 보안 인증 또는 웹 ID 공급자를 사용하여 CodeCommit 리포지토리에 액세스하는 경우 Git 보안 인증을 사용할 수 없습니다. `git-remote-codecommit` 명령을 사용하여 로컬 컴퓨터를 설정하는 것이 좋습니다. 그러나 모든 IDE가 git-remote-codecommit와 같은 Git 원격 헬퍼와 완벽하게 호환되는 것은 아닙니다. 문제가 발생하면 git-remote-codecommit 및 문제 해결 AWS CodeCommit 단원을 참조하세요.

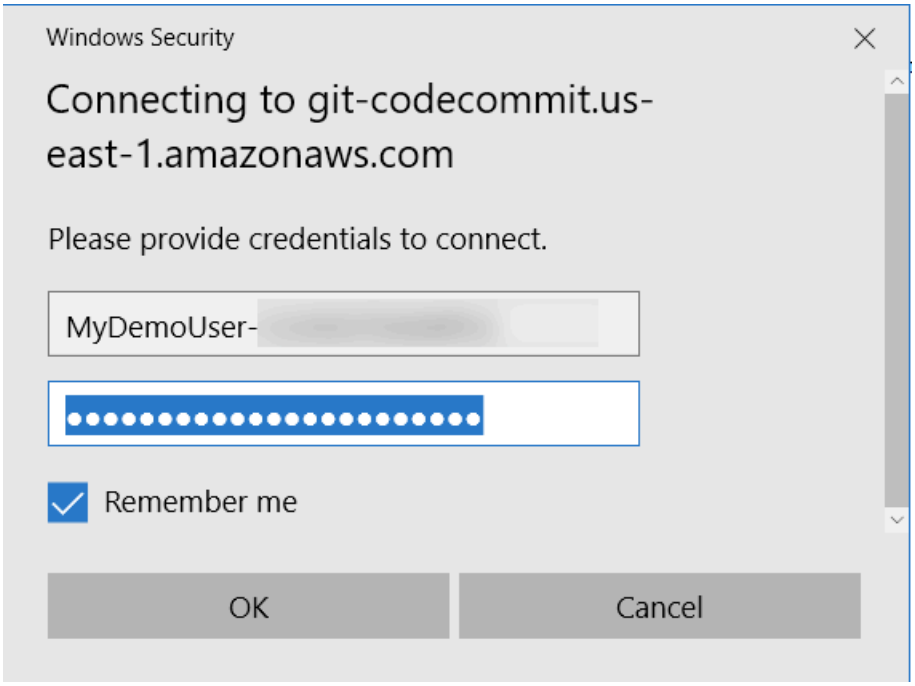
주제

- AWS Cloud9 와 통합 AWS CodeCommit
- Visual Studio를와 통합 AWS CodeCommit

IDE 또는 개발 도구에서, CodeCommit 리포지토리에 연결하는 데 사용된 사용자 이름과 암호를 입력하라는 메시지가 표시되면, IAM에서 생성한 사용자 이름 및 암호에 대한 Git 보안 인증 정보를 제공합니다.

CodeCommit의 AWS 리전 및 엔드포인트에 대한 자세한 내용은 섹션을 참조하세요리전 및 Git 연결 엔드포인트.

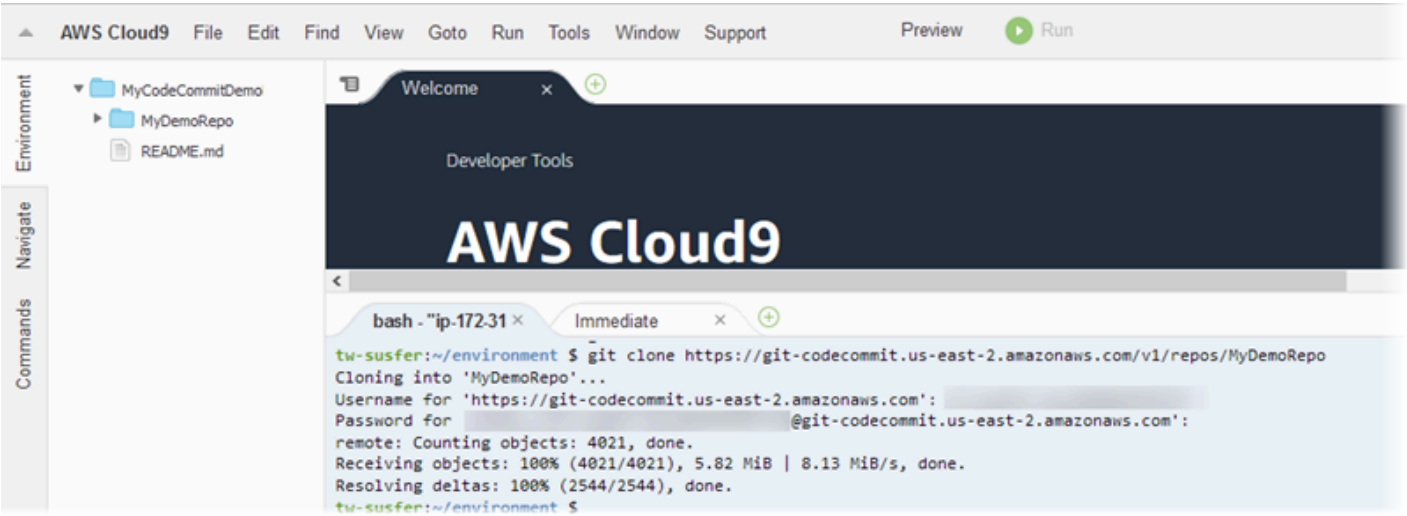
운영 체제에서 사용자 이름과 암호를 저장하라는 메시지가 나타날 수도 있습니다. 예를 들어, Windows에서 다음과 같이 Git 보안 인증 정보를 제공합니다.



특정 소프트웨어 프로그램 또는 개발 도구의 Git 보안 인증 정보를 구성하는 방법에 대한 자세한 내용은 해당 제품 설명서를 참조하세요.

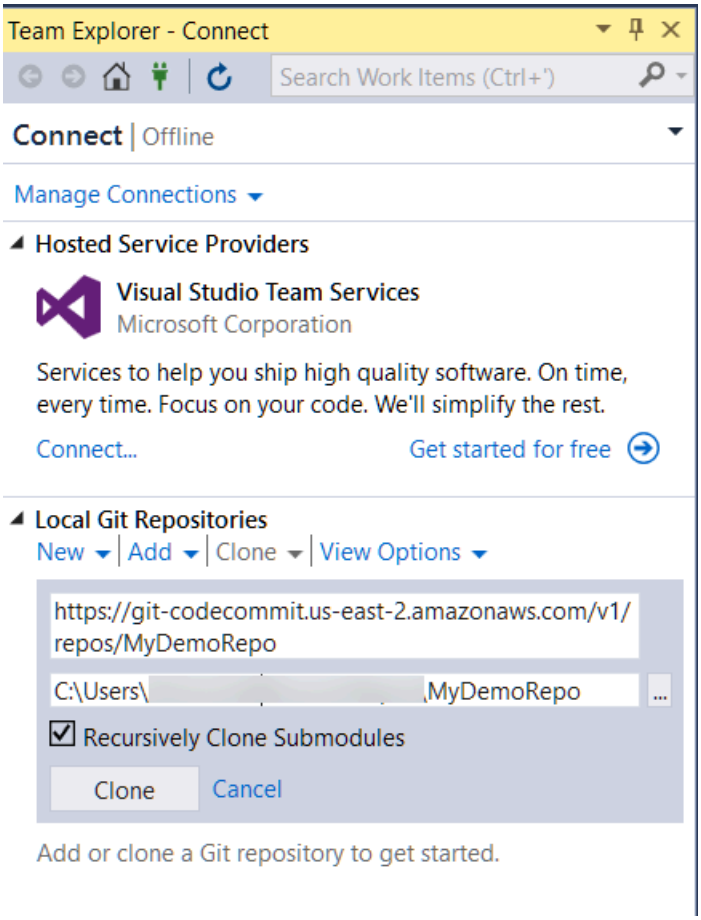
다음은 IDE 전체 목록이 아닙니다. 링크는 이러한 도구에 대해 자세히 알아보는 데 도움이 되는 용도로만 제공됩니다. AWS 는 이러한 주제의 내용에 대해 책임을 지지 않습니다.

- AWS Cloud9



- Visual Studio

  또는를 설치합니다 AWS Toolkit for Visual Studio. 자세한 내용은 Visual Studio를와 통합 AWS CodeCommit 단원을 참조하십시오.



- XCode

AWS Cloud9 와 통합 AWS CodeCommit

AWS Cloud9 를 사용하여 CodeCommit 리포지토리에서 코드를 변경할 수 있습니다. AWS Cloud9 에는 코드를 작성하고 소프트웨어를 빌드, 실행, 테스트, 디버깅 및 릴리스하는 데 사용할 수 있는 도구 모음이 포함되어 있습니다. AWS Cloud9 EC2 개발 환경에서 기존 리포지토리를 복제하고, 리포지토리를 생성하고, 코드 변경 사항을 리포지토리에 커밋하고 푸시하는 등의 작업을 수행할 수 있습니다. AWS Cloud9 EC2 개발 환경은 일반적으로 AWS CLI Amazon EC2 역할 및 Git으로 미리 구성되어 있으므로 대부분의 경우 몇 가지 간단한 명령을 실행하고 리포지토리와 상호 작용을 시작할 수 있습니다.

CodeCommit AWS Cloud9 에서를 사용하려면 다음이 필요합니다.

- Amazon Linux에서 실행되는 An AWS Cloud9 EC2 개발 환경.
- AWS Cloud9 IDE가 웹 브라우저에서 열립니다.
- CodeCommit 관리형 정책 중 하나와 관리 AWS Cloud9 형 정책 중 하나가 적용된 IAM 사용자.

자세한 내용은 AWS CodeCommit에 대한 관리형 정책 및 보안 보안 인증 정보 이해 및 얻기를 참조하세요.

> **Note**
>
> 이 주제에서는 CodeCommit과의 통합 설정 및 인터넷에서의 일반적인 액세스 AWS Cloud9에 대해 설명합니다. 격리된 환경에서 CodeCommit 및 AWS Cloud9 에 대한 액세스를 설정할 수 있지만 추가 단계가 필요합니다. 자세한 내용은 다음을 참조하세요.
>
> - 인터페이스 VPC 엔드포인트 AWS CodeCommit 와 함께 사용
> - 를 사용하여 수신하지 않는 Amazon EC2 인스턴스에 액세스 AWS Systems Manager
> - 공유 환경 사용
> - 다른 계정과 VPC 공유
> - 블로그 게시물: AWS Cloud9 환경에 대한 네트워크 액세스 격리

주제

- 1단계: AWS Cloud9 개발 환경 생성
- 2단계: AWS Cloud9 EC2 개발 환경에서 AWS CLI 보안 인증 도우미 구성
- 3단계: CodeCommit 리포지토리를 AWS Cloud9 EC2 개발 환경에 복제
- 다음 단계

1단계: AWS Cloud9 개발 환경 생성

AWS Cloud9 는 Amazon EC2 인스턴스에서 개발 환경을 호스팅합니다. 인스턴스의 AWS 관리형 임시 자격 증명을 사용하여 CodeCommit 리포지토리에 연결할 수 있으므로 가장 쉽게 통합할 수 있습니다. 자체 서버를 대신 사용하려면 AWS Cloud9 사용 설명서를 참조하세요.

AWS Cloud9 환경을 생성하려면

1. 구성한 IAM 사용자 AWS 로에 로그인하고 콘솔을 AWS Cloud9 엽니다.
2. AWS Cloud9 콘솔에서 환경 생성을 선택합니다.
3. 1단계: 환경 이름 지정에 환경의 이름과 설명(선택 사항)을 입력한 후, 다음 단계를 선택합니다.
4. 2단계: 설정 구성에서 다음과 같이 환경을 구성합니다.
   - 환경 유형에서 환경의 새 인스턴스 생성(EC2)을 선택합니다.

- 인스턴스 유형에서 개발 환경에 적합한 인스턴스 유형을 선택합니다. 예를 들어 서비스를 탐색하려는 경우 기본 유형인 t2.micro를 선택할 수 있습니다. 이 환경을 사용하여 개발 작업을 하려는 경우에는 더 큰 인스턴스 유형을 선택합니다.
- 다른 것을 선택할 이유(예: 조직에서 특정 VPC를 사용하거나 Amazon Web Services 계정에 구성된 VPC가 없는 경우)가 없다면 다른 기본 설정을 수락하고 다음 단계를 선택합니다.

5. 3단계: 검토에서 설정을 검토합니다. 변경하려면 이전 단계를 선택합니다. 그렇지 않다면 환경 생성을 선택합니다.

   처음으로 환경을 만들고 연결하는 경우 몇 분 정도 걸릴 수 있습니다. 시간이 비정상적으로 오래 걸리는 것 같으면 AWS Cloud9 사용 설명서에서 문제 해결을 참조하세요.

6. 환경에 연결되면 터미널 창에서 git --version 명령을 실행하여 Git이 설치되어 있으며 지원되는 버전인지 확인합니다.

   Git이 설치되어 있지 않거나 지원되는 버전이 아니면 지원되는 버전을 설치합니다. CodeCommit은 Git 버전 1.7.9 이상을 지원합니다. Git 버전 2.28에서는 초기 커밋을 위한 브랜드 이름을 구성할 수 있습니다. 최신 버전의 Git를 사용하는 것이 좋습니다. Git 설치를 위해서는 Git 다운로드와 같은 웹 사이트를 권장합니다.

   > **Tip**
   >
   > 환경의 운영 체제에 따라 sudo 옵션과 함께 yum 명령을 사용하여 Git을 포함한 업데이트를 설치할 수 있습니다. 예를 들어 다음 세 가지 명령과 같은 관리자 명령 시퀀스를 사용할 수 있습니다.
   >
   > ```
   > sudo yum -y update
   > sudo yum -y install git
   > git --version
   > ```

7. git config 명령을 실행하여 Git 커밋에 연결할 사용자 이름과 이메일을 구성합니다. 예시:

   ```
   git config --global user.name "Mary Major"
   git config --global user.email mary.major@example.com
   ```

2단계: AWS Cloud9 EC2 개발 환경에서 AWS CLI 보안 인증 도우미 구성

AWS Cloud9 환경을 생성한 후 CodeCommit 리포지토리에 연결하기 위한 AWS CLI 자격 증명을 관리하도록 자격 증명 도우미를 구성할 수 있습니다. 개발 환경은 AWS Cloud9 IAM 사용자와 연결된 AWS 관리형 임시 자격 증명과 함께 제공됩니다. 자격 증명 헬퍼와 함께 이러한 자격 AWS CLI 증명을 사용합니다.

1. 터미널 창을 열고 다음 명령을 실행하여 AWS CLI 가 설치되어 있는지 확인합니다.

   ```
   aws --version
   ```

   성공하면 이 명령은 현재 설치된 AWS CLI버전을 반환합니다. 이전 버전의 AWS CLI 를 최신 버전으로 업그레이드하려면 설치를 참조하세요 AWS Command Line Interface.

2. 터미널에서 다음 명령을 실행하여 HTTPS 연결을 위한 AWS CLI 자격 증명 헬퍼를 구성합니다.

   ```
   git config --global credential.helper '!aws codecommit credential-helper $@'
   git config --global credential.UseHttpPath true
   ```

   > **Tip**
   > 보안 인증 도우미는 개발 환경에 기본 Amazon EC2 인스턴스 역할을 사용합니다. 개발 환경을 사용하여 CodeCommit에서 호스팅되지 않은 리포지토리에 연결하려는 경우, 해당 리포지토리에 대한 SSH 연결을 구성하거나 다른 리포지토리에 연결할 때 대체 보안 인증 정보 관리 시스템을 사용하도록 로컬 `.gitconfig` 파일을 구성합니다. 자세한 내용은 Git 웹사이트의 Git Tools - Credential Storage를 참조하세요.

3단계: CodeCommit 리포지토리를 AWS Cloud9 EC2 개발 환경에 복제

AWS CLI 자격 증명 헬퍼를 구성한 후 CodeCommit 리포지토리를 복제할 수 있습니다. 그런 다음 코드 작업을 시작할 수 있습니다.

1. 터미널에서 git clone 명령을 실행하여 복제하려는 리포지토리의 HTTPS 복제 URL을 지정합니다. 예를 들어 미국 동부(오하이오) 리전에서 MyDemoRepo라는 리포지토리를 복제하려면 다음을 입력합니다.

   ```
   git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
   ```

> **Tip**
>
> CodeCommit 콘솔에서 복제 URL을 선택하여 리포지토리의 복제 URL을 찾을 수 있습니다.

2. 복제가 완료되면 옆의 탐색 창에서 리포지토리 폴더를 확장하고, 편집하려는 파일을 선택하여 엽니다. 또는 파일을 선택한 다음 새 파일을 선택하여 파일을 만듭니다.
3. 파일을 편집하거나 만들었으면 터미널 창에서 복제된 리포지토리가 있는 디렉터리로 변경한 후 변경 내용을 커밋하고 푸시합니다. 예를 들어 *MyFile.py*라는 새 파일을 추가한 경우:

```
cd MyDemoRepo
git commit -a MyFile.py
git commit -m "Added a new file with some code improvements"
git push
```

다음 단계

자세한 내용은 AWS Cloud9 사용 설명서 및 AWS Cloud9용 CodeCommit 샘플을 참조하세요. CodeCommit에 Git을 사용하는 것에 대한 자세한 내용은 Git 및 시작하기 AWS CodeCommit 섹션을 참조하세요.

Visual Studio를와 통합 AWS CodeCommit

Visual Studio를 사용하여 CodeCommit 리포지토리에서 코드를 변경할 수 있습니다. AWS Toolkit for Visual Studio 이제 에는 Visual Studio에서 작업할 때 CodeCommit을 더 쉽고 편리하게 사용할 수 있는 기능이 포함되어 있습니다. Visual Studio용 Toolkit 통합은 Git 보안 인증 정보 및 IAM 사용자로 작업하기 위한 것입니다. 기존 리포지토리를 복제하고 리포지토리를 생성하며 코드 변경 내용을 리포지토리로 커밋 및 푸시하는 등의 작업을 수행할 수 있습니다.

> **Important**
>
> Visual Studio용 Toolkit은 Windows 운영 체제에만 설치할 수 있습니다. Visual Studio Code로 작업하는 방법에 대해 자세히 알아보려면 AWS Toolkit for Visual Studio Code을 참조하세요.

이전에 Visual Studio용 Toolkit을 사용했다면, 아마 액세스 키와 보안 키를 포함하는 AWS 보안 인증 프로필을 설정하는 방법에 대해 잘 알고 있을 것입니다. 자격 증명 프로필은 Toolkit for Visual Studio에서

AWS 서비스 APIs(예: 버킷을 나열하는 Amazon S3 또는 리포지토리를 나열하는 CodeCommit)에 대한 호출을 활성화하는 데 사용됩니다. CodeCommit 리포지토리로 코드를 풀 및 푸시하려는 경우에도 Git 보안 인증 정보가 필요합니다. Git 보안 인증 정보가 없으면 Visual Studio용 Toolkit에서 이 보안 인증 정보를 생성하고 적용할 수 있습니다. 이렇게 하면 시간을 크게 절약할 수 있습니다.

CodeCommit과 함께 Visual Studio를 사용하려면 다음이 필요합니다.

- 유효한 보안 인증 정보 세트(액세스 키와 보안 키)가 구성되어 있는 IAM 사용자. 또한 이 IAM 사용자는 다음을 갖추어야 합니다.

  CodeCommit 관리형 정책 한 개와 그 정책에 적용된 IAMSelfManageServiceSpecificCredentials 관리형 정책.

  OR

  IAM 사용자에게 Git 보안 인증 정보가 이미 구성된 경우, CodeCommit 관리형 정책 중 하나 또는 이에 상응하는 권한.

  자세한 내용은 AWS CodeCommit에 대한 관리형 정책 및 보안 인증 정보 이해 및 얻기 단원을 참조하세요.
- Visual Studio를 AWS Toolkit for Visual Studio 설치한 컴퓨터에가 설치되어 있습니다. 자세한 내용은 AWS Toolkit for Visual Studio설정을 참조하세요.

CodeCommit AWS Toolkit for Visual Studio 에서를 사용하는 방법에 대한 자세한 내용은 Toolkit for Visual Studio 사용 설명서의 Using AWS CodeCommit with Visual Studio Team Explorer를 참조하세요.

를 사용하지 않는 SSH 사용자를 위한 설정 AWS CLI

리포지토리에 SSH 연결을 사용하려면를 설치 AWS CodeCommit 하지 않고에 연결할 수 있습니다 AWS CLI. 에는 CodeCommit 리포지토리를 사용하고 관리할 때 유용한 명령이 AWS CLI 포함되어 있지만 초기 설정에는 필요하지 않습니다.

이 주제에서는 다음을 가정합니다.

- CodeCommit에 필요한 정책 또는 권한을 사용하여, 그리고 키 업로드에 필요한 IAMUsersshKeys 관리형 정책 또는 동등한 권한을 사용하여 IAM 사용자를 설정했습니다. 자세한 내용은 CodeCommit에 대한 자격 증명 기반 정책(IAM 정책) 사용 단원을 참조하십시오.

- 퍼블릭-프라이빗 키 페어를 이미 가지고 있거나 만드는 방법을 알고 있습니다. SSH 키에는 보안 암호를 사용할 것을 적극 권장합니다.
- SSH, Git 클라이언트 및 그 구성 파일에 익숙합니다.
- Windows를 사용하는 경우에는 bash 쉘을 에뮬레이션하는 Git Bash와 같은 명령줄 유틸리티를 설치했습니다.

추가 안내가 필요한 경우 Linux, macOS, Unix에서 SSH 연결 또는 Windows에서 SSH 연결의 지침을 따릅니다.

주제

- 1단계: 퍼블릭 키를 IAM 사용자와 연결
- 2단계: SSH 구성에 CodeCommit 추가
- 다음 단계

1단계: 퍼블릭 키를 IAM 사용자와 연결

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/://https://https://https://https://://https://://https://://https://://https://https://://://
2. IAM 콘솔의 탐색 창에서 사용자를 선택하고 사용자 목록에서 해당 IAM 사용자를 선택합니다.
3. 보안 인증 정보 탭에서 SSH 퍼블릭 키 업로드를 선택합니다.
4. SSH 퍼블릭 키의 콘텐츠를 필드에 붙여넣은 후 SSH 키 업로드를 선택합니다.

> **Tip**
>
> 퍼블릭-프라이빗 키 페어는 OpenSSH 형식의 SSH-2 RSA여야 하며 2048비트를 포함해야 합니다. 해당 키는 다음과 비슷합니다.

```
ssh-rsa EXAMPLE-
AfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJB
gNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb2
5zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhc
NMTEwNDI1MjA0NTIxWhcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAw
DgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDAS=EXAMPLE user-
name@ip-192-0-2-137
```

IAM은 OpenSSH 형식의 퍼블릭 키만 수락합니다. 퍼블릭 키를 다른 형식으로 제공하면, 키 형식이 잘못되었다는 오류 메시지가 표시됩니다.

5. SSH 키 ID(예: *APKAEIBAERJR2EXAMPLE*)를 복사하고 콘솔을 닫습니다.

2단계: SSH 구성에 CodeCommit 추가

1. 터미널(Linux, macOS, Unix) 또는 bash 에뮬레이터(Windows)에서 cat>> ~/.ssh/config를 입력하여 SSH 구성 파일을 편집합니다.

```
Host git-codecommit.*.amazonaws.com
User Your-SSH-Key-ID, such as APKAEIBAERJR2EXAMPLE
IdentityFile Your-Private-Key-File, such as ~/.ssh/codecommit_rsa or ~/.ssh/id_rsa
```

> **Tip**
>
> SSH 구성이 두 개 이상인 경우 콘텐츠 앞뒤에 빈 줄을 포함해야 합니다. `Ctrl` 및 `d` 키를 동시에 눌러 파일을 저장합니다.

2. 다음 명령을 실행하여 SSH 구성을 테스트합니다.

```
ssh git-codecommit.us-east-2.amazonaws.com
```

메시지가 표시되면 SSH 키 파일의 암호를 입력합니다. 모두 다 올바르게 구성되면 다음과 같은 성공 메시지가 표시되어야 합니다.

```
You have successfully authenticated over SSH. You can use Git to interact with
 CodeCommit.
```

다음 단계

사전 필수 단계를 완료했습니다. CodeCommit 시작하기 에 나와 있는 단계에 따라 CodeCommit을 사용하기 시작합니다.

리포지토리에 연결하려면 리포지토리에 연결의 단계를 따릅니다. 리포지토리를 생성하려면 리포지토리 생성의 단계를 따릅니다.

Linux, macOS 또는 Unix의 AWS CodeCommit 리포지토리에 대한 SSH 연결 설정 단계

CodeCommit에 처음으로 연결하려면 초기 구성 단계를 일부 완료해야 합니다. 컴퓨터와 AWS 프로필을 설정한 후 CodeCommit 리포지토리에 연결하고 해당 리포지토리를 컴퓨터에 복제할 수 있습니다(로컬 리포지토리 생성이라고도 함). Git을 처음 사용하는 경우, Git에 대해 더 학습하려면 어떻게 해야 하나요?의 정보를 검토하는 것이 좋습니다.

주제

- 1단계: CodeCommit에 대한 초기 구성
- 2단계: Git 설치
- 3단계: Linux, macOS, Unix에서 보안 인증 정보 구성
- 4단계: CodeCommit 콘솔 연결 및 리포지토리 복제
- 다음 단계

1단계: CodeCommit에 대한 초기 구성

이 절차에 따라 Amazon Web Services 계정을 설정하고 IAM 사용자를 생성하며 CodeCommit에 대한 액세스를 구성합니다.

CodeCommit에 액세스할 IAM 사용자를 생성 및 구성하려면

1. http://aws.amazon.com에서 가입을 선택하여 Amazon Web Services 계정을 만듭니다.
2. Amazon Web Services 계정에서 IAM 사용자를 생성하거나 기존 사용자를 사용합니다. 액세스 키 ID가 있고 해당 IAM 사용자와 연결된 비밀 액세스 키가 있는지 확인합니다. 자세한 내용은 Amazon Web Services 계정에서 IAM 사용자 생성하기를 참조하세요.

> **Note**
>
> CodeCommit에는가 필요합니다 AWS Key Management Service. 기존 IAM 사용자를 사용하는 경우 CodeCommit에 필요한 AWS KMS 작업을 명시적으로 거부하는 정책이 사용자에게 연결되어 있지 않은지 확인합니다. 자세한 내용은 AWS KMS 및 암호화 단원을 참조하십시오.

3. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/://https://https://https://https://://https://://https://://https://://https://://https://://
4. IAM 콘솔의 탐색 창에서 사용자를 선택한 다음 CodeCommit 액세스에 대해 구성할 IAM 사용자를 선택합니다.
5. 권한 탭에서 권한 추가를 선택합니다.
6. 권한 부여에서 기존 정책 직접 연결을 선택합니다.
7. 정책 목록에서, CodeCommit 액세스에 대한 AWSCodeCommitPowerUser 또는 다른 관리형 정책을 선택합니다. 자세한 내용은 AWS CodeCommit에 대한 관리형 정책 단원을 참조하십시오.

   연결할 정책을 선택한 후 다음: 검토를 선택하여 IAM 사용자에게 연결할 정책의 목록을 검토합니다. 목록이 올바르면 권한 추가를 선택합니다.

   CodeCommit 관리형 정책, 그리고 리포지토리에 대한 액세스를 다른 그룹 및 사용자와 공유하는 방법에 대해 자세히 알아보려면 리포지토리 공유, AWS CodeCommit에 대한 인증 및 액세스 제어 섹션을 참조하세요.

> **Note**
>
> CodeCommit에서 AWS CLI 명령을 사용하려면를 설치합니다 AWS CLI. 자세한 내용은 명령줄 참조 단원을 참조하십시오.

2단계: Git 설치

CodeCommit 리포지토리에서 파일과 커밋 및 기타 정보를 사용하려면 로컬 시스템에 Git를 설치해야 합니다. CodeCommit은 Git 버전 1.7.9 이상을 지원합니다. Git 버전 2.28에서는 초기 커밋을 위한 브랜드 이름을 구성할 수 있습니다. 최신 버전의 Git를 사용하는 것이 좋습니다.

Git 설치를 위해서는 Git 다운로드와 같은 웹 사이트를 권장합니다.

> **Note**
>
> Git은 계속 개선되며 정기적으로 업데이트되는 플랫폼입니다. 기능 변경에 따라 CodeCommit 사용 방식이 달라지는 경우가 있습니다. Git과 CodeCommit의 특정 버전에서 문제가 발생할 경우 문제 해결의 내용을 검토합니다.

3단계: Linux, macOS, Unix에서 보안 인증 정보 구성

SSH와 Linux, macOS Unix: Git과 CodeCommit에 사용되는 퍼블릭 키와 프라이빗 키를 설정합니다.

Git과 CodeCommit에 사용되는 퍼블릭 키와 프라이빗 키를 설정하려면

1. 로컬 컴퓨터의 터미널에서 ssh-keygen 명령을 실행하고 지침에 따라 파일을 프로필의 .ssh 디렉터리에 저장합니다.

   > **Note**
   >
   > 키 파일을 저장할 위치와 사용할 파일 이름 지정 패턴은 시스템 관리자에게 문의하세요.

   예시:

```
$ ssh-keygen

Generating public/private rsa key pair.
Enter file in which to save the key (/home/user-name/.ssh/id_rsa): Type /home/
your-user-name/.ssh/ and a file name here, for example /home/your-user-name/.ssh/
codecommit_rsa

Enter passphrase (empty for no passphrase): <Type a passphrase, and then press
 Enter>
Enter same passphrase again: <Type the passphrase again, and then press Enter>

Your identification has been saved in /home/user-name/.ssh/codecommit_rsa.
Your public key has been saved in /home/user-name/.ssh/codecommit_rsa.pub.
The key fingerprint is:
45:63:d5:99:0e:99:73:50:5e:d4:b3:2d:86:4a:2c:14 user-name@client-name
The key's randomart image is:
```

```
+--[ RSA 2048]----+
|        E.+.o*.++|
|        .o .=.=o.|
|       . ..  *. +|
|        ..o . +..|
|        So . . . |
|          .      |
|                 |
|                 |
|                 |
+-----------------+
```

다음이 생성됩니다.

- *codecommit_rsa* 파일: 프라이빗 키 파일입니다.
- *codecommit_rsa*.pub 파일: 퍼블릭 키 파일입니다.

> **Tip**
>
> 기본 설정에 따라 ssh-keygen은 2048비트 키를 생성합니다. 매개 변수 -t와 -b를 사용하여 키의 유형과 길이를 지정할 수 있습니다. rsa 형식의 4,096비트 키를 원하는 경우, 해당 명령 실행 시 다음의 파라미터를 사용합니다.
>
> ```
> ssh-keygen -t rsa -b 4096
> ```
>
> SSH 키에 필요한 형식과 길이에 대해 자세히 알아보려면 CodeCommit에서 IAM 사용하기를 참조하세요.

2. 다음 명령을 실행하여 퍼블릭 키 파일(*codecommit_rsa*.pub)의 값을 표시합니다.

```
cat ~/.ssh/codecommit_rsa.pub
```

이 값을 복사합니다. 내용은 다음처럼 보일 것입니다.

```
ssh-rsa EXAMPLE-AfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJB
gNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb2
5zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhc
NMTEwNDI1MjA0NTIxWhcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAw
DgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDAS=EXAMPLE user-name@ip-192-0-2-137
```

3. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/://https://https://https://https://://https://://https://://https://://https://://https://://https://://

> **Note**
>
> 내 보안 인증 정보에서 CodeCommit 보안 인증 정보를 직접 보고 관리할 수 있습니다. 자세한 내용은 보안 인증 정보 보기 및 관리 섹션을 참조하세요.

4. IAM 콘솔의 탐색 창에서 사용자를 선택하고 사용자 목록에서 해당 IAM 사용자를 선택합니다.
5. 사용자 세부 정보 페이지에서 보안 인증 정보 탭을 선택한 다음 SSH 퍼블릭 키 업로드를 선택합니다.
6. SSH 퍼블릭 키의 콘텐츠를 필드에 붙여넣은 후 SSH 퍼블릭 키 업로드를 선택합니다.
7. SSH 키 ID(예: *APKAEIBAERJR2EXAMPLE*)에 정보를 복사하거나 저장합니다.

> **Note**
>
> SSH 키 ID를 두 개 이상 업로드한 경우, 키가 업로드 날짜가 아니라 키 ID를 기준으로 영문 자순으로 나열됩니다. 올바른 업로드 날짜와 연결된 키 ID를 복사했는지 확인하세요.

8. 로컬 컴퓨터에서 텍스트 편집기를 사용하여 ~/.ssh 디렉터리에 config 파일을 생성한 후 다음 줄을 파일에 추가합니다. 여기서 *User*는 이전에 복사한 SSH 키 ID입니다.

```
Host git-codecommit.*.amazonaws.com
  User APKAEIBAERJR2EXAMPLE
  IdentityFile ~/.ssh/codecommit_rsa
```

> **Note**
>
> 프라이빗 키 파일에 *codecommit_rsa* 이외의 이름을 지정한 경우, 여기서 해당 이름을 사용해야 합니다.

> Amazon Web Services의 여러 계정에서 리포지토리에 대한 SSH 액세스를 설정할 수 있습니다. 자세한 내용은 AWS CodeCommit과의 SSH 연결 문제 해결 단원을 참조하세요.

이 파일을 저장하고 이름을 `config`로 지정합니다.

9. 터미널에서 명령을 실행하여 config 파일에 대한 권한을 변경합니다.

```
chmod 600 config
```

10. 다음 명령을 실행하여 SSH 구성을 테스트합니다.

```
ssh git-codecommit.us-east-2.amazonaws.com
```

`git-codecommit.us-east-2.amazonaws.com`이 알려진 호스트 파일에 아직 포함되어 있지 않으므로 연결을 확인하라는 메시지가 표시됩니다. CodeCommit 서버 지문이 인증 과정에서 표시됩니다(MD5의 경우 `a9:6d:03:ed:08:42:21:be:06:e1:e0:2a:d1:75:31:5e` 또는 SHA256의 경우 `3lBlW2g5xn/NA2Ck6dyeJIrQOWvn7n8UEs56fG6ZIzQ`).

> **Note**
>
> CodeCommit 서버 지문은 모든에 대해 고유합니다 AWS 리전. 에 대한 서버 지문을 보려면 섹션을 AWS 리전참조하세요CodeCommit의 서버 지문.

연결을 확인한 후 서버를 알려진 호스트 파일에 추가했다는 확인 메시지와 연결이 성공했다는 메시지가 표시되어야 합니다. 성공 메시지가 표시되지 않을 경우, CodeCommit 액세스를 위해 구성한 IAM 사용자의 ~/.ssh 디렉터리에 `config` 파일을 저장하고 올바른 프라이빗 키 파일을 지정했는지 확인합니다.

문제를 해결하는 데 도움이 되는 정보를 보려면 `-v` 파라미터를 사용하여 `ssh` 명령을 실행합니다. 예시:

```
ssh -v git-codecommit.us-east-2.amazonaws.com
```

연결 문제를 해결하기 위한 자세한 방법은 AWS CodeCommit과의 SSH 연결 문제 해결 단원을 참조하세요.

4단계: CodeCommit 콘솔 연결 및 리포지토리 복제

관리자가 CodeCommit 리포지토리에 대한 이름과 연결 세부 정보를 이미 전송한 경우 이 단계를 건너뛰고 리포지토리를 직접 복제할 수 있습니다.

CodeCommit 리포지토리에 연결하려면

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리전 선택기에서 리포지토리가 생성된 AWS 리전 를 선택합니다. 리포지토리는에 고유합니다 AWS 리전. 자세한 내용은 리전 및 Git 연결 엔드포인트 단원을 참조하십시오.
3. 목록에서 연결하려는 리포지토리를 찾아서 선택합니다. URL 복제를 선택한 다음 리포지토리를 복제하거나 연결할 때 사용할 프로토콜을 선택합니다. 이것으로 복제 URL가 복사됩니다.

   - IAM 사용자를 통한 Git 보안 인증 정보를 활용하거나 AWS CLI에 포함된 보안 인증 도우미를 사용하는 경우 HTTPS URL을 복사합니다.
   - 로컬 컴퓨터에서 git-remote-codecommit 명령을 사용하는 경우 HTTPS(GRC) URL을 복사합니다.
   - IAM 사용자와 SSH 퍼블릭/프라이빗 키 페어를 사용하는 경우 SSH URL을 복사합니다.

   > **Note**
   >
   > 리포지토리 목록 대신 시작 페이지가 표시되면 로그인한에 AWS 계정과 연결된 AWS 리전 리포지토리가 없는 것입니다. 리포지토리를 만들려면 the section called “리포지토리 생성”을(를) 참조하거나 Git 및 CodeCommit 시작하기 자습서의 다음 단계를 따릅니다.

4. 터미널을 엽니다. /tmp 디렉터리에서 복사한 SSH URL로 git clone 명령을 실행하여 리포지토리를 복제합니다. 예를 들어, *MyDemoRepo*라는 리포지토리를 미국 동부(오하이오) 리전의 *my-demo-repo*라는 로컬 리포지토리에 복제하려면 다음과 같이 합니다.

   ```
   git clone ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-demo-repo
   ```

   > **Note**
   >
   > 연결을 성공적으로 테스트했지만 복제 명령이 실패하는 경우, config 파일에 필요한 액세스 권한이 없거나 다른 설정이 config 파일과 충돌하는 것일 수 있습니다. 이번에는 SSH 키 ID를 명령에 포함하여 연결을 다시 시도하십시오. 예시:

```
git clone ssh://Your-SSH-Key-ID@git-codecommit.us-east-2.amazonaws.com/v1/
repos/MyDemoRepo my-demo-repo
```

자세한 내용은 액세스 오류: 퍼블릭 키가 IAM에 성공적으로 업로드되었지만 Linux, macOS, Unix 시스템에서 연결이 실패했습니다 단원을 참조하십시오.

리포지토리에 연결하는 방법에 대한 자세한 내용은 리포지토리를 복제하여 CodeCommit 리포지토리에 연결 섹션을 참조하세요.

다음 단계

사전 필수 단계를 완료했습니다. CodeCommit 시작하기 에 나와 있는 단계에 따라 CodeCommit의 사용을 시작합니다.

Windows의 AWS CodeCommit 리포지토리에 대한 SSH 연결 설정 단계

AWS CodeCommit 에 처음 연결하려면 먼저 몇 가지 초기 구성 단계를 완료해야 합니다. 컴퓨터와 AWS 프로필을 설정한 후 CodeCommit 리포지토리에 연결하고 해당 리포지토리를 컴퓨터에 복제할 수 있습니다(로컬 리포지토리 생성이라고도 함). Git을 처음 사용하는 경우, Git에 대해 더 학습하려면 어떻게 해야 하나요?의 정보를 검토하는 것이 좋습니다.

주제

- 1단계: CodeCommit에 대한 초기 구성
- 2단계: Git 설치
- 3단계: Git과 CodeCommit에 사용되는 퍼블릭 키와 프라이빗 키 설정
- 4단계: CodeCommit 콘솔 연결 및 리포지토리 복제
- 다음 단계

1단계: CodeCommit에 대한 초기 구성

이 절차에 따라 Amazon Web Services 계정을 설정하고 IAM 사용자를 생성하며 CodeCommit에 대한 액세스를 구성합니다.

CodeCommit에 액세스할 IAM 사용자를 생성 및 구성하려면

1. http://aws.amazon.com에서 가입을 선택하여 Amazon Web Services 계정을 만듭니다.
2. Amazon Web Services 계정에서 IAM 사용자를 생성하거나 기존 사용자를 사용합니다. 액세스 키 ID가 있고 해당 IAM 사용자와 연결된 비밀 액세스 키가 있는지 확인합니다. 자세한 내용은 Amazon Web Services 계정에서 IAM 사용자 생성하기를 참조하세요.

   > **Note**
   >
   > CodeCommit에는가 필요합니다 AWS Key Management Service. 기존 IAM 사용자를 사용하는 경우 CodeCommit에 필요한 AWS KMS 작업을 명시적으로 거부하는 정책이 사용자에게 연결되어 있지 않은지 확인합니다. 자세한 내용은 AWS KMS 및 암호화 단원을 참조하십시오.

3. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/://https://https://https://https://://https://https://://https://://https://://https://://https://://http
4. IAM 콘솔의 탐색 창에서 사용자를 선택한 다음 CodeCommit 액세스에 대해 구성할 IAM 사용자를 선택합니다.
5. 권한 탭에서 권한 추가를 선택합니다.
6. 권한 부여에서 기존 정책 직접 연결을 선택합니다.
7. 정책 목록에서, CodeCommit 액세스에 대한 AWSCodeCommitPowerUser 또는 다른 관리형 정책을 선택합니다. 자세한 내용은 AWS CodeCommit에 대한 관리형 정책 단원을 참조하십시오.

   연결할 정책을 선택한 후 다음: 검토를 선택하여 IAM 사용자에게 연결할 정책의 목록을 검토합니다. 목록이 올바르면 권한 추가를 선택합니다.

   CodeCommit 관리형 정책, 그리고 리포지토리에 대한 액세스를 다른 그룹 및 사용자와 공유하는 방법에 대해 자세히 알아보려면 리포지토리 공유, AWS CodeCommit에 대한 인증 및 액세스 제어 섹션을 참조하세요.

> **Note**
>
> CodeCommit에서 AWS CLI 명령을 사용하려면를 설치합니다 AWS CLI. 자세한 내용은 명령줄 참조 단원을 참조하십시오.

2단계: Git 설치

CodeCommit 리포지토리에서 파일과 커밋 및 기타 정보를 사용하려면 로컬 시스템에 Git를 설치해야 합니다. CodeCommit은 Git 버전 1.7.9 이상을 지원합니다. Git 버전 2.28에서는 초기 커밋을 위한 브랜드 이름을 구성할 수 있습니다. 최신 버전의 Git를 사용하는 것이 좋습니다.

Git 설치를 위해서는 Git 다운로드와 같은 웹 사이트를 권장합니다.

> **Note**
>
> Git은 계속 개선되며 정기적으로 업데이트되는 플랫폼입니다. 기능 변경에 따라 CodeCommit 사용 방식이 달라지는 경우가 있습니다. Git과 CodeCommit의 특정 버전에서 문제가 발생할 경우 문제 해결의 내용을 검토합니다.

설치한 Git 버전에 Git Bash와 같은 Bash 에뮬레이터가 포함되지 않다면 에뮬레이터를 설치합니다. SSH 연결을 구성할 때 Windows 명령줄 대신 이 에뮬레이터를 사용합니다.

3단계: Git과 CodeCommit에 사용되는 퍼블릭 키와 프라이빗 키 설정

Windows에서 Git과 CodeCommit에 사용되는 퍼블릭 키와 프라이빗 키를 설정하려면

1. Bash 에뮬레이터를 엽니다.

   > **Note**
   >
   > 관리자 권한을 사용하여 에뮬레이터를 실행해야 할 수 있습니다.

   에뮬레이터에서 ssh-keygen 명령을 실행하고 지침에 따라 파일을 프로필의 .ssh 디렉터리에 저장합니다.

   예시:

```
$ ssh-keygen

Generating public/private rsa key pair.
Enter file in which to save the key (/drive/Users/user-name/.ssh/id_rsa): Type a
 file name here, for example /c/Users/user-name/.ssh/codecommit_rsa
```

```
Enter passphrase (empty for no passphrase): <Type a passphrase, and then press
 Enter>
Enter same passphrase again: <Type the passphrase again, and then press Enter>

Your identification has been saved in drive/Users/user-name/.ssh/codecommit_rsa.
Your public key has been saved in drive/Users/user-name/.ssh/codecommit_rsa.pub.
The key fingerprint is:
45:63:d5:99:0e:99:73:50:5e:d4:b3:2d:86:4a:2c:14 user-name@client-name
The key's randomart image is:
+--[ RSA 2048]----+
|        E.+.o*.++|
|        .o .=.=o.|
|       . ..  *. +|
|        ..o . +..|
|        So . . . |
|          .      |
|                 |
|                 |
|                 |
+-----------------+
```

다음이 생성됩니다.

- *codecommit_rsa* 파일: 프라이빗 키 파일입니다.
- *codecommit_rsa*.pub 파일: 퍼블릭 키 파일입니다.

> **Tip**
>
> 기본 설정에 따라 ssh-keygen은 2048비트 키를 생성합니다. 매개 변수 -t와 -b를 사용하여 키의 유형과 길이를 지정할 수 있습니다. rsa 형식의 4,096비트 키를 원하는 경우, 해당 명령 실행 시 다음의 파라미터를 사용합니다.
>
> ```
> ssh-keygen -t rsa -b 4096
> ```
>
> SSH 키에 필요한 형식과 길이에 대해 자세히 알아보려면 CodeCommit에서 IAM 사용하기를 참조하세요.

2. 다음 명령을 실행하여 퍼블릭 키 파일(*codecommit_rsa*.pub)의 값을 표시합니다.

```
cd .ssh
```

```
notepad codecommit_rsa.pub
```

파일 콘텐츠를 복사한 다음 메모장을 저장하지 않고 닫습니다. 파일 콘텐츠는 다음과 비슷합니다.

```
ssh-rsa EXAMPLE-AfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJB
gNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb2
5zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhc
NMTEwNDI1MjA0NTIxWhcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAw
DgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDAS=EXAMPLE user-name@computer-name
```

3. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/://https://https://https://https://://https://://https://://https://://https://://https://https://://://

> **Note**
>
> 내 보안 인증 정보에서 CodeCommit 보안 인증 정보를 직접 보고 관리할 수 있습니다. 자세한 내용은 보안 인증 정보 보기 및 관리 섹션을 참조하세요.

4. IAM 콘솔의 탐색 창에서 사용자를 선택하고 사용자 목록에서 해당 IAM 사용자를 선택합니다.
5. 사용자 세부 정보 페이지에서 보안 인증 정보 탭을 선택한 다음 SSH 퍼블릭 키 업로드를 선택합니다.
6. SSH 퍼블릭 키의 콘텐츠를 필드에 붙여넣은 후 SSH 퍼블릭 키 업로드를 선택합니다.
7. SSH 키 ID(예: *APKAEIBAERJR2EXAMPLE*)에 정보를 복사하거나 저장합니다.

> **Note**
>
> SSH 키 ID를 두 개 이상 업로드한 경우, 키가 업로드 날짜가 아니라 키 ID를 기준으로 영문 자순으로 나열됩니다. 올바른 업로드 날짜와 연결된 키 ID를 복사했는지 확인하세요.

8. Bash 에뮬레이터에서 다음 명령을 실행하여 ~/.ssh 디렉터리에서 config 파일을 생성하거나 이미 있는 경우 편집합니다.

```
notepad ~/.ssh/config
```

9. 파일에 다음 행을 추가합니다. 여기서 *User* 값은 앞서 복사한 SSH 키 ID이고 *IdentityFile*은 프라이빗 키 파일의 경로 및 이름입니다.

```
Host git-codecommit.*.amazonaws.com
  User APKAEIBAERJR2EXAMPLE
  IdentityFile ~/.ssh/codecommit_rsa
```

> **Note**
>
> 프라이빗 키 파일에 *codecommit_rsa* 이외의 이름을 지정한 경우, 여기서 해당 이름을 사용해야 합니다.
>
> Amazon Web Services의 여러 계정에서 리포지토리에 대한 SSH 액세스를 설정할 수 있습니다. 자세한 내용은 AWS CodeCommit과의 SSH 연결 문제 해결 단원을 참조하세요.

파일을 config 파일(config.txt가 아님)로 저장하고 메모장을 닫습니다.

> **Important**
>
> 파일 이름은 파일 확장명 없이 `config`여야 합니다. 그렇지 않으면 SSH 연결이 실패합니다.

10. 다음 명령을 실행하여 SSH 구성을 테스트합니다.

```
ssh git-codecommit.us-east-2.amazonaws.com
```

`git-codecommit.us-east-2.amazonaws.com`이 알려진 호스트 파일에 아직 포함되어 있지 않으므로 연결을 확인하라는 메시지가 표시됩니다. CodeCommit 서버 지문이 인증 과정에서 표시됩니다(MD5의 경우 `a9:6d:03:ed:08:42:21:be:06:e1:e0:2a:d1:75:31:5e` 또는 SHA256의 경우 `3lBlW2g5xn/NA2Ck6dyeJIrQOWvn7n8UEs56fG6ZIzQ`).

> **Note**
>
> CodeCommit 서버 지문은 모든에 대해 고유합니다 AWS 리전. 의 서버 지문을 보려면 섹션을 AWS 리전참조하세요CodeCommit의 서버 지문.

연결을 확인한 후 서버를 알려진 호스트 파일에 추가했다는 확인 메시지와 연결이 성공했다는 메시지가 표시되어야 합니다. 성공 메시지가 표시되지 않는 경우, CodeCommit에 액세스하도록 구성한 IAM 사용자의 ~/.ssh 디렉터리에 `config` 파일을 저장했는지, `config` 파일에 파일 확장자가 없는지(예: 이름을 config.txt로 짓지 말 것), 올바른 프라이빗 키 파일을 지정했는지 (*`codecommit_rsa`*.pub가 아닌 *`codecommit_rsa`*) 등을 다시 확인합니다.

문제를 해결하려면 `-v` 파라미터를 사용하여 `ssh` 명령을 실행합니다. 예시:

```
ssh -v git-codecommit.us-east-2.amazonaws.com
```

연결 문제를 해결하기 위한 자세한 방법은 AWS CodeCommit과의 SSH 연결 문제 해결 단원을 참조하세요.

4단계: CodeCommit 콘솔 연결 및 리포지토리 복제

관리자가 CodeCommit 리포지토리에 대한 이름과 연결 세부 정보를 이미 전송한 경우 이 단계를 건너뛰고 리포지토리를 직접 복제할 수 있습니다.

CodeCommit 리포지토리에 연결하려면

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리전 선택기에서 리포지토리가 생성된 AWS 리전 를 선택합니다. 리포지토리는에 고유합니다 AWS 리전. 자세한 내용은 리전 및 Git 연결 엔드포인트 단원을 참조하십시오.
3. 목록에서 연결하려는 리포지토리를 찾아서 선택합니다. URL 복제를 선택한 다음 리포지토리를 복제하거나 연결할 때 사용할 프로토콜을 선택합니다. 이것으로 복제 URL가 복사됩니다.
   - IAM 사용자를 통한 Git 보안 인증 정보를 활용하거나 AWS CLI에 포함된 보안 인증 도우미를 사용하는 경우 HTTPS URL을 복사합니다.
   - 로컬 컴퓨터에서 git-remote-codecommit 명령을 사용하는 경우 HTTPS(GRC) URL을 복사합니다.
   - IAM 사용자와 SSH 퍼블릭/프라이빗 키 페어를 사용하는 경우 SSH URL을 복사합니다.

> **Note**
>
> 리포지토리 목록 대신 시작 페이지가 표시되면 로그인한에 AWS 계정과 연결된 AWS 리전 리포지토리가 없는 것입니다. 리포지토리를 만들려면 the section called "리포지토리 생성"를 참조하거나 Git 및 CodeCommit 시작하기 자습서의 다음 단계를 따릅니다.

4. Bash 에뮬레이터에서 복사한 SSH URL로 git clone 명령을 실행하여 리포지토리를 복제합니다. 이 명령은 명령을 실행한 디렉터리의 하위 디렉터리에 로컬 리포지토리를 생성합니다. 예를 들어, *MyDemoRepo*라는 리포지토리를 미국 동부(오하이오) 리전의 *my-demo-repo*라는 로컬 리포지토리에 복제하려면 다음과 같이 합니다.

```
git clone ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-demo-
repo
```

또는 명령 프롬프트를 열고 IAM에 업로드한 퍼블릭 키의 URL 및 SSH 키 ID를 사용하여 git clone 명령을 실행합니다. 그러면 명령을 실행한 디렉터리의 하위 디렉터리에 로컬 리포지토리가 생성됩니다. 예를 들어, *MyDemoRepo*라는 리포지토리를 *my-demo-repo*라는 로컬 리포지토리에 복제하려면 다음과 같이 합니다.

```
git clone ssh://Your-SSH-Key-ID@git-codecommit.us-east-2.amazonaws.com/v1/repos/
MyDemoRepo my-demo-repo
```

자세한 내용은 리포지토리를 복제하여 CodeCommit 리포지토리에 연결 및 커밋 생성 단원을 참조하세요.

다음 단계

사전 필수 단계를 완료했습니다. CodeCommit 시작하기 에 나와 있는 단계에 따라 CodeCommit의 사용을 시작합니다.

AWS CLI 자격 증명 헬퍼를 사용하여 Linux, macOS 또는 Unix의 AWS CodeCommit 리포지토리에 HTTPS 연결을 위한 설정 단계

AWS CodeCommit 에 처음 연결하려면 먼저 초기 구성 단계를 완료해야 합니다. 대부분의 사용자는 Git 보안 인증 정보를 사용하는 HTTPS 사용자의 경우의 단계를 따라 가장 쉽게 진행할 수 있습니다. 하

지만 루트 계정, 페더레이션 액세스 또는 임시 보안 인증 정보를 사용하여 CodeCommit에 연결하려는 경우, AWS CLI에 포함된 보안 인증 도우미를 사용할 수 있습니다.

> **Note**
>
> 보안 인증 도우미는 페더레이션 액세스, ID 공급자 또는 임시 보안 인증 정보를 사용하여 CodeCommit에 연결하는 데 지원되는 방법이지만 git-remote-codecommit 유틸리티를 설치하고 사용하는 것이 좋습니다. 자세한 내용은 git-remote-codecommit을 AWS CodeCommit 사용하여에 HTTPS 연결을 위한 설정 단계 단원을 참조하십시오.

주제

- 1단계: CodeCommit에 대한 초기 구성
- 2단계: Git 설치
- 3단계: 보안 인증 도우미 설정
- 4단계: CodeCommit 콘솔 연결 및 리포지토리 복제
- 다음 단계

1단계: CodeCommit에 대한 초기 구성

이 절차에 따라 Amazon Web Services account 계정을 설정하고 IAM 사용자를 생성 및 구성하며 AWS CLI를 설치합니다.

CodeCommit에 액세스할 IAM 사용자를 생성 및 구성하려면

1. http://aws.amazon.com에서 가입을 선택하여 Amazon Web Services 계정을 만듭니다.
2. Amazon Web Services 계정에서 IAM 사용자를 생성하거나 기존 사용자를 사용합니다. 액세스 키 ID가 있고 해당 IAM 사용자와 연결된 비밀 액세스 키가 있는지 확인합니다. 자세한 내용은 Amazon Web Services 계정에서 IAM 사용자 생성하기를 참조하세요.

   > **Note**
   >
   > CodeCommit에는가 필요합니다 AWS Key Management Service. 기존 IAM 사용자를 사용하는 경우 CodeCommit에 필요한 AWS KMS 작업을 명시적으로 거부하는 정책이 사용자에게 연결되어 있지 않은지 확인합니다. 자세한 내용은 AWS KMS 및 암호화 단원을 참조하십시오.

3. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/://https://https://https://https://://https://https://://https://://https://://https://://https://://http
4. IAM 콘솔의 탐색 창에서 사용자를 선택한 다음 CodeCommit 액세스에 대해 구성할 IAM 사용자를 선택합니다.
5. 권한 탭에서 권한 추가를 선택합니다.
6. 권한 부여에서 기존 정책 직접 연결을 선택합니다.
7. 정책 목록에서, CodeCommit 액세스에 대한 AWSCodeCommitPowerUser 또는 다른 관리형 정책을 선택합니다. 자세한 내용은 AWS CodeCommit에 대한 관리형 정책 단원을 참조하십시오.

   연결할 정책을 선택한 후 다음: 검토를 선택하여 IAM 사용자에게 연결할 정책의 목록을 검토합니다. 목록이 올바르면 권한 추가를 선택합니다.

   CodeCommit 관리형 정책, 그리고 리포지토리에 대한 액세스를 다른 그룹 및 사용자와 공유하는 방법에 대해 자세히 알아보려면 리포지토리 공유, AWS CodeCommit에 대한 인증 및 액세스 제어 섹션을 참조하세요.

를 설치하고 구성하려면 AWS CLI

1. 로컬 시스템에서를 다운로드하여 설치합니다 AWS CLI. 이는 명령줄에서 CodeCommit과 상호 작용하기 위한 사전 조건입니다. AWS CLI 버전 2를 설치하는 것이 좋습니다. 의 최신 메이저 버전이며 모든 최신 기능을 AWS CLI 지원합니다. 에서 루트 계정, 페더레이션 액세스 또는 임시 자격 증명 사용을 AWS CLI 지원하는 유일한 버전의 입니다git-remote-codecommit.

   자세한 내용은 AWS 명령줄 인터페이스 설정 단원을 참조하십시오.

   > **Note**
   >
   > CodeCommit은 AWS CLI 버전 1.7.38 이상에서만 작동합니다. 가장 좋은 방법은를 설치하거나 사용 가능한 최신 버전으로 업그레이드 AWS CLI 하는 것입니다. 설치 AWS CLI 한 버전을 확인하려면 aws --version 명령을 실행합니다.
   > 이전 버전의 AWS CLI 를 최신 버전으로 업그레이드하려면 설치를 참조하세요 AWS Command Line Interface.

2. 이 명령을 실행하여에 대한 CodeCommit 명령 AWS CLI 이 설치되어 있는지 확인합니다.

```
aws codecommit help
```

이 명령은 CodeCommit 명령 목록을 반환합니다.

3. 다음과 같이 configure 명령을 사용하여 프로파일 AWS CLI 로를 구성합니다.

```
aws configure
```

메시지가 표시되면 CodeCommit과 함께 사용할 IAM 사용자의 AWS 액세스 키와 AWS 보안 액세스 키를 지정합니다. 또한와 같이 리포지토리 AWS 리전 가 있는를 지정해야 합니다`us-east-2`. 기본 출력 형식을 묻는 메시지가 표시되면 `json`을 지정합니다. 예를 들어, IAM 사용자의 프로필을 구성하는 경우에는 다음과 같이 합니다.

```
AWS Access Key ID [None]: Type your IAM user AWS access key ID here, and then press
 Enter
AWS Secret Access Key [None]: Type your IAM user AWS secret access key here, and
 then press Enter
Default region name [None]: Type a supported region for CodeCommit here, and then
 press Enter
Default output format [None]: Type json here, and then press Enter
```

에서 사용할 프로필을 생성하고 구성하는 방법에 대한 자세한 내용은 다음을 AWS CLI참조하세요.

- 명명된 프로파일
- 에서 IAM 역할 사용 AWS CLI
- 설정 명령
- 교체 자격 증명을 사용하여 AWS CodeCommit 리포지토리에 연결

리포지토리 또는 다른의 리소스에 연결하려면 기본 리전 이름으로 AWS CLI 를 재구성 AWS 리전 해야 합니다. CodeCommit에서 지원되는 기본 리전 이름은 다음과 같습니다.

- us-east-2
- us-east-1
- eu-west-1
- us-west-2
- ap-northeast-1
- ap-southeast-1

- ap-southeast-2
- ap-southeast-3
- me-central-1
- eu-central-1
- ap-northeast-2
- sa-east-1
- us-west-1
- eu-west-2
- ap-south-1
- ap-south-1
- ca-central-1
- us-gov-west-1
- us-gov-east-1
- eu-north-1
- ap-east-1
- me-south-1
- cn-north-1
- cn-northwest-1
- eu-south-1
- ap-northeast-3
- af-south-1
- il-central-1

CodeCommit 및에 대한 자세한 내용은 섹션을 AWS 리전참조하세요리전 및 Git 연결 엔드포인트. IAM, 액세스 키, 비밀 키에 대한 자세한 내용을 확인하려면 보안 인증 정보 얻는 방법 및 IAM 사용자의 액세스 키 관리를 참조하세요. AWS CLI 및 프로파일에 대한 자세한 내용은 명명된 프로파일을 참조하세요.

2단계: Git 설치

CodeCommit 리포지토리에서 파일과 커밋 및 기타 정보를 사용하려면 로컬 시스템에 Git를 설치해야 합니다. CodeCommit은 Git 버전 1.7.9 이상을 지원합니다. Git 버전 2.28에서는 초기 커밋을 위한 브랜드 이름을 구성할 수 있습니다. 최신 버전의 Git를 사용하는 것이 좋습니다.

Git 설치를 위해서는 Git 다운로드와 같은 웹 사이트를 권장합니다.

> **Note**
>
> Git은 계속 개선되며 정기적으로 업데이트되는 플랫폼입니다. 기능 변경에 따라 CodeCommit 사용 방식이 달라지는 경우가 있습니다. Git과 CodeCommit의 특정 버전에서 문제가 발생할 경우 문제 해결의 내용을 검토하세요.

3단계: 보안 인증 도우미 설정

1. 터미널에서 Git을 사용하여를 실행하고git config, 자격 증명 프로필과 함께 Git AWS 자격 증명 헬퍼 사용을 지정하고, Git 자격 증명 헬퍼가 리포지토리 경로를 전송하도록 활성화합니다.

```
git config --global credential.helper '!aws codecommit credential-helper $@'
git config --global credential.UseHttpPath true
```

> **Tip**
>
> 자격 증명 도우미는 기본 AWS 자격 증명 프로필 또는 Amazon EC2 인스턴스 역할을 사용합니다. CodeCommit에서 사용할 AWS 보안 인증 프로필을 생성한 경우에는 `CodeCommitProfile`과 같이 사용할 프로필을 다음과 같이 지정할 수 있습니다.
>
> ```
> git config --global credential.helper '!aws --profile CodeCommitProfile
> codecommit credential-helper $@'
> ```
>
> 프로필 이름에 공백이 포함되어 있는 경우에는 그 이름을 인용 부호(")로 묶어주어야 합니다.
> `--global` 대신에 `--local`을 사용함으로써 전역적으로가 아닌 리포지토리당 프로필을 구성할 수 있습니다.

Git 보안 인증 도우미는 다음 값을 `~/.gitconfig`에 작성합니다.

```
[credential]
    helper = !aws --profile CodeCommitProfile codecommit credential-helper $@
    UseHttpPath = true
```

> **Important**
>
> CodeCommit에 대해 동일한 로컬 시스템에서 다른 IAM 사용자를 사용하려면 git config 명령을 다시 실행하고 다른 AWS 자격 증명 프로파일을 지정해야 합니다.

2. git config --global --edit를 실행하여 이전 값이 `~/.gitconfig`에 기록되었는지 확인합니다. 기록되었다면 Git 전역 구성 파일에 이미 있는 값뿐 아니라 이전 값도 보여야 합니다. 종료하려면 일반적으로 **`:q`**를 입력한 다음 Enter를 누릅니다.

   보안 인증 도우미를 구성한 후에 문제가 있으면 문제 해결 단원을 참조하세요.

   > **Important**
   >
   > macOS를 사용하고 있다면 다음 절차를 통해 보안 인증 도우미가 올바르게 구성되도록 할 수 있습니다.

3. macOS를 사용하고 있다면 HTTPS를 사용해 CodeCommit 리포지토리에 접속합니다. HTTPS로 CodeCommit 리포지토리에 처음 연결한 후 약 15분이 지나면 후속 액세스가 불가능해집니다. macOS의 기본 Git 버전에서는 Keychain Access 유틸리티를 사용하여 보안 인증 정보을 저장합니다. 보안 조치를 위해 해당 CodeCommit 리포지토리에 액세스하기 위해 생성된 암호가 임시용으로 설정되어 있으므로 키 체인에 저장된 보안 인증 정보은 약 15분 후에 작동을 멈춥니다. 이처럼 만료된 보안 인증 정보을 사용하지 못하도록 방지하려면 다음 중 한 가지를 수행해야 합니다.

   - 키 체인을 사용하도록 기본 설정되어 있지 않은 Git 버전을 설치합니다.
   - CodeCommit 리포지토리에 보안 인증 정보을 제공하지 않도록 Keychain Access 유틸리티를 구성합니다.

   1. Keychain Access 유틸리티를 실행합니다. (Finder를 사용하여 이 유틸리티를 찾을 수 있습니다.)

2. `git-codecommit.us-east-2.amazonaws.com`를 찾습니다. 행을 강조 표시하고 컨텍스트 메뉴를 열거나 마우스 오른쪽 버튼으로 클릭한 다음 정보 가져오기를 선택합니다.
3. 액세스 제어 탭을 선택합니다.
4. 액세스 허용 전 확인에서 `git-credential-osxkeychain`을 선택한 후 마이너스 기호를 선택하여 목록에서 제거합니다.

> **Note**
>
> 목록에서 `git-credential-osxkeychain`을 제거하면 Git 명령을 실행할 때마다 팝업 메시지가 표시됩니다. 거부를 선택하여 계속 진행합니다. 팝업이 너무 방해가 되는 경우, 다음과 같은 다른 옵션을 선택할 수 있습니다.
>
> - HTTPS 대신에 SSH를 사용하여 CodeCommit에 접속합니다. 자세한 내용은 Linux, macOS, Unix에서 SSH 연결 단원을 참조하십시오.
> - Keychain Access 유틸리티 내, `git-codecommit.us-east-2.amazonaws.com`에 대한 액세스 제어 탭에서 모든 애플리케이션이 이 항목에 액세스하도록 허용(이 항목에 대한 액세스는 제한되지 않음) 옵션을 선택합니다. 이렇게 하면 팝업이 표시되지 않지만 보안 인증 정보은 결국 만료되며(평균 약 15분 소요) 그 다음에는 403 오류 메시지가 표시됩니다. 이 경우 기능을 복원하려면 키체인 항목을 삭제해야 합니다.
> - 자세한 내용은 macOS용 Git: 보안 인증 도우미를 성공적으로 구성했지만 현재 내 리포지토리에 액세스가 거부되었습니다(403). 단원을 참조하십시오.

4단계: CodeCommit 콘솔 연결 및 리포지토리 복제

관리자가 CodeCommit 리포지토리에 대한 이름과 연결 세부 정보를 이미 전송한 경우 이 단계를 건너뛰고 리포지토리를 직접 복제할 수 있습니다.

CodeCommit 리포지토리에 연결하려면

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리전 선택기에서 리포지토리가 생성된 AWS 리전 를 선택합니다. 리포지토리는에 고유합니다 AWS 리전. 자세한 내용은 리전 및 Git 연결 엔드포인트 단원을 참조하십시오.
3. 목록에서 연결하려는 리포지토리를 찾아서 선택합니다. URL 복제를 선택한 다음 리포지토리를 복제하거나 연결할 때 사용할 프로토콜을 선택합니다. 이것으로 복제 URL가 복사됩니다.

- IAM 사용자를 통한 Git 보안 인증 정보를 활용하거나 AWS CLI에 포함된 보안 인증 도우미를 사용하는 경우 HTTPS URL을 복사합니다.
- 로컬 컴퓨터에서 git-remote-codecommit 명령을 사용하는 경우 HTTPS(GRC) URL을 복사합니다.
- IAM 사용자와 SSH 퍼블릭/프라이빗 키 페어를 사용하는 경우 SSH URL을 복사합니다.

> **Note**
>
> 리포지토리 목록 대신 시작 페이지가 표시되면 로그인 AWS 리전 한에 AWS 계정과 연결된 리포지토리가 없는 것입니다. 리포지토리를 만들려면 the section called "리포지토리 생성"을(를) 참조하거나 Git 및 CodeCommit 시작하기 자습서의 다음 단계를 따릅니다.

4. 터미널을 열고 복사한 HTTPS URL로 git clone 명령을 실행합니다. 예를 들어, *MyDemoRepo*라는 리포지토리를 미국 동부(오하이오) 리전의 *my-demo-repo*라는 로컬 리포지토리에 복제하려면 다음과 같이 합니다.

```
git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-
demo-repo
```

다음 단계

사전 필수 단계를 완료했습니다. CodeCommit 시작하기 에 나와 있는 단계에 따라 CodeCommit의 사용을 시작합니다.

AWS CLI 자격 증명 헬퍼를 사용하여 Windows의 AWS CodeCommit 리포지토리에 HTTPS 연결을 위한 설정 단계

AWS CodeCommit 에 처음 연결하려면 먼저 초기 구성 단계를 완료해야 합니다. 대부분의 사용자는 Git 보안 인증 정보를 사용하는 HTTPS 사용자의 경우의 단계를 따라 가장 쉽게 진행할 수 있습니다. 하지만 루트 계정, 페더레이션 액세스 또는 임시 보안 인증 정보를 사용하여 CodeCommit에 연결하려는 경우, AWS CLI에 포함된 보안 인증 도우미를 사용할 수 있습니다.

> **Note**
>
> 보안 인증 도우미는 페더레이션 액세스, ID 공급자 또는 임시 보안 인증 정보를 사용하여 CodeCommit에 연결하는 데 지원되는 방법이지만 git-remote-codecommit 유틸리티를 설치하고 사용하는 것이 좋습니다. 자세한 내용은 git-remote-codecommit을 AWS CodeCommit 사용하여에 HTTPS 연결을 위한 설정 단계 단원을 참조하십시오.

이 주제에서는 로컬 리포지토리 생성이라고도 하는를 설치하고 AWS CLI, 컴퓨터 및 AWS 프로필을 설정하고, CodeCommit 리포지토리에 연결하고, 해당 리포지토리를 컴퓨터에 복제하는 단계를 안내합니다. Git을 처음 사용하는 경우, Git에 대해 더 학습하려면 어떻게 해야 하나요?의 정보를 검토하는 것이 좋습니다.

주제

- 1단계: CodeCommit에 대한 초기 구성
- 2단계: Git 설치
- 3단계: 보안 인증 도우미 설정
- 4단계: CodeCommit 콘솔 연결 및 리포지토리 복제
- 다음 단계

1단계: CodeCommit에 대한 초기 구성

이 절차에 따라 Amazon Web Services account 계정을 설정하고 IAM 사용자를 생성 및 구성하며 AWS CLI를 설치합니다. AWS CLI 에는 CodeCommit 리포지토리에 대한 HTTPS 연결을 구성하는 보안 인증 도우미가 포함됩니다.

CodeCommit에 액세스할 IAM 사용자를 생성 및 구성하려면

1. http://aws.amazon.com에서 가입을 선택하여 Amazon Web Services 계정을 만듭니다.
2. Amazon Web Services 계정에서 IAM 사용자를 생성하거나 기존 사용자를 사용합니다. 액세스 키 ID가 있고 해당 IAM 사용자와 연결된 비밀 액세스 키가 있는지 확인합니다. 자세한 내용은 Amazon Web Services 계정에서 IAM 사용자 생성하기를 참조하세요.

   > **Note**
   >
   > CodeCommit에는가 필요합니다 AWS Key Management Service. 기존 IAM 사용자를 사용하는 경우 CodeCommit에 필요한 AWS KMS 작업을 명시적으로 거부하는 정책이 사용

자에게 연결되어 있지 않은지 확인합니다. 자세한 내용은 AWS KMS 및 암호화 단원을 참조하십시오.

3. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/://https://https://https://https://://https://://https://://https://://https://://https://://https://://
4. IAM 콘솔의 탐색 창에서 사용자를 선택한 다음 CodeCommit 액세스에 대해 구성할 IAM 사용자를 선택합니다.
5. 권한 탭에서 권한 추가를 선택합니다.
6. 권한 부여에서 기존 정책 직접 연결을 선택합니다.
7. 정책 목록에서, CodeCommit 액세스에 대한 AWSCodeCommitPowerUser 또는 다른 관리형 정책을 선택합니다. 자세한 내용은 AWS CodeCommit에 대한 관리형 정책 단원을 참조하십시오.

   연결할 정책을 선택한 후 다음: 검토를 선택하여 IAM 사용자에게 연결할 정책의 목록을 검토합니다. 목록이 올바르면 권한 추가를 선택합니다.

   CodeCommit 관리형 정책, 그리고 리포지토리에 대한 액세스를 다른 그룹 및 사용자와 공유하는 방법에 대해 자세히 알아보려면 리포지토리 공유, AWS CodeCommit에 대한 인증 및 액세스 제어 섹션을 참조하세요.

설치 및 구성 AWS CLI

1. 로컬 시스템에서를 다운로드하여 설치합니다 AWS CLI. 이는 명령줄에서 CodeCommit과 상호 작용하기 위한 사전 조건입니다. AWS CLI 버전 2를 설치하는 것이 좋습니다. 의 최신 메이저 버전이며 모든 최신 기능을 AWS CLI 지원합니다. 에서 루트 계정, 페더레이션 액세스 또는 임시 자격 증명 사용을 AWS CLI 지원하는 유일한 버전의 입니다git-remote-codecommit.

   자세한 내용은 AWS 명령줄 인터페이스 설정 단원을 참조하십시오.

   > **Note**
   >
   > CodeCommit은 AWS CLI 버전 1.7.38 이상에서만 작동합니다. 가장 좋은 방법은를 설치하거나 사용 가능한 최신 버전으로 업그레이드 AWS CLI 하는 것입니다. 설치 AWS CLI 한 버전을 확인하려면 aws --version 명령을 실행합니다.
   > 의 이전 버전을 AWS CLI 최신 버전으로 업그레이드하려면 설치를 참조하세요 AWS Command Line Interface.

2. 이 명령을 실행하여에 대한 CodeCommit 명령 AWS CLI 이 설치되어 있는지 확인합니다.

```
aws codecommit help
```

이 명령은 CodeCommit 명령 목록을 반환합니다.

3. 다음과 같이 configure 명령을 사용하여 프로파일 AWS CLI 로를 구성합니다.

```
aws configure
```

메시지가 표시되면 CodeCommit과 함께 사용할 IAM 사용자의 AWS 액세스 키와 AWS 보안 액세스 키를 지정합니다. 또한와 같이 리포지토리 AWS 리전 가 있는를 지정해야 합니다`us-east-2`. 기본 출력 형식을 묻는 메시지가 표시되면 `json`을 지정합니다. 예를 들어, IAM 사용자의 프로필을 구성하는 경우에는 다음과 같이 합니다.

```
AWS Access Key ID [None]: Type your IAM user AWS access key ID here, and then press
 Enter
AWS Secret Access Key [None]: Type your IAM user AWS secret access key here, and
 then press Enter
Default region name [None]: Type a supported region for CodeCommit here, and then
 press Enter
Default output format [None]: Type json here, and then press Enter
```

에서 사용할 프로필을 생성하고 구성하는 방법에 대한 자세한 내용은 다음을 AWS CLI참조하세요.

- 명명된 프로파일
- 에서 IAM 역할 사용 AWS CLI
- 설정 명령
- 교체 자격 증명을 사용하여 AWS CodeCommit 리포지토리에 연결

리포지토리 또는 다른의 리소스에 연결하려면 기본 리전 이름으로 AWS CLI 를 재구성 AWS 리전 해야 합니다. CodeCommit에서 지원되는 기본 리전 이름은 다음과 같습니다.

- us-east-2
- us-east-1
- eu-west-1
- us-west-2

- ap-northeast-1
- ap-southeast-1
- ap-southeast-2
- ap-southeast-3
- me-central-1
- eu-central-1
- ap-northeast-2
- sa-east-1
- us-west-1
- eu-west-2
- ap-south-1
- ap-south-1
- ca-central-1
- us-gov-west-1
- us-gov-east-1
- eu-north-1
- ap-east-1
- me-south-1
- cn-north-1
- cn-northwest-1
- eu-south-1
- ap-northeast-3
- af-south-1
- il-central-1

CodeCommit 및에 대한 자세한 내용은 섹션을 AWS 리전참조하세요리전 및 Git 연결 엔드포인트. IAM, 액세스 키, 비밀 키에 대한 자세한 내용을 확인하려면 보안 인증 정보 얻는 방법 및 IAM 사용자의 액세스 키 관리를 참조하세요. AWS CLI 및 프로파일에 대한 자세한 내용은 명명된 프로파일을 참조하세요.

2단계: Git 설치

CodeCommit 리포지토리에서 파일과 커밋 및 기타 정보를 사용하려면 로컬 시스템에 Git를 설치해야 합니다. CodeCommit은 Git 버전 1.7.9 이상을 지원합니다. Git 버전 2.28에서는 초기 커밋을 위한 브랜드 이름을 구성할 수 있습니다. 최신 버전의 Git를 사용하는 것이 좋습니다.

Git를 설치하려면 Git for Windows 등의 웹 사이트를 사용하는 것이 좋습니다. 이 링크를 사용하여 Git을 설치하는 경우 다음을 제외한 설치 기본 설정을 모두 수락할 수 있습니다.

- PATH 환경 조정 단계에서 메시지가 표시될 경우 명령줄에서 Git를 사용하는 옵션을 선택합니다.
- (선택 사항) CodeCommit에 대한 Git 자격 증명을 구성하는 AWS CLI 대신에 포함된 자격 증명 헬퍼와 함께 HTTPS를 사용하려면 추가 옵션 구성 페이지에서 Git 자격 증명 관리자 활성화 옵션이 선택 취소되어 있는지 확인합니다. IAM 사용자가 Git 보안 인증 정보를 구성할 경우 Git 보안 인증 관리자는 오직 CodeCommit과 호환됩니다. 자세한 내용은 Git 보안 인증 정보를 사용하는 HTTPS 사용자의 경우 및 Windows용 Git: Windows용 Git을 설치했지만 내 리포지토리에 액세스가 거부되었습니다(403). 단원을 참조하세요.

> **Note**
>
> Git은 계속 개선되며 정기적으로 업데이트되는 플랫폼입니다. 기능 변경에 따라 CodeCommit 사용 방식이 달라지는 경우가 있습니다. Git과 CodeCommit의 특정 버전에서 문제가 발생할 경우 문제 해결의 내용을 검토하세요.

3단계: 보안 인증 도우미 설정

에는 CodeCommit과 함께 사용할 수 있는 Git 자격 증명 헬퍼가 AWS CLI 포함되어 있습니다. Git 자격 증명 헬퍼에는 IAM 사용자의 AWS 액세스 키 ID 및 AWS 보안 액세스 키(기본 AWS 리전 이름 및 기본 출력 형식 포함)의 사본을 저장하는 자격 증명 프로필이 필요합니다 AWS . Git 보안 인증 도우미는 이 정보를 사용하여 CodeCommit에 자동으로 인증하므로 Git을 사용하여 CodeCommit과 상호 작용할 때마다 이 정보를 입력할 필요가 없습니다.

1. 명령 프롬프트를 열고 Git을 사용하여를 실행합니다. Git 자격 증명 도우미를 AWS 자격 증명 프로필과 함께 사용하여 Git 자격 증명 도우미가 리포지토리 경로를 보낼 수 있도록 git config지정합니다.

```
git config --global credential.helper "!aws codecommit credential-helper $@"
```

```
git config --global credential.UseHttpPath true
```

Git 보안 인증 도우미는 .gitconfig 파일에 다음 사항을 작성합니다.

```
[credential]
    helper = !aws codecommit credential-helper $@
    UseHttpPath = true
```

Important

- Windows 명령줄 대신 Bash 에뮬레이터를 사용하는 경우 큰따옴표 대신 작은따옴표를 사용해야 합니다.
- 자격 증명 도우미는 기본 AWS 프로필 또는 Amazon EC2 인스턴스 역할을 사용합니다. *CodeCommitProfile*과 같이 사용할 AWS 자격 증명 프로파일을 생성한 경우 다음과 같이 명령을 수정하여 대신 사용할 수 있습니다.

  ```
  git config --global credential.helper "!aws codecommit credential-helper
   --profile CodeCommitProfile $@"
  ```

  이것은 .gitconfig 파일에 다음 사항을 작성합니다.

  ```
  [credential]
      helper = !aws codecommit credential-helper --profile=CodeCommitProfile
   $@
      UseHttpPath = true
  ```

- 프로필 이름에 공백이 있으면 이 명령을 실행하여 작은따옴표(')로 묶은 이후에 .gitconfig 파일을 편집해야 합니다. 그렇지 않으면 보안 인증 도우미가 동작하지 않습니다.
- Windows용 Git 프로그램에 Git Credential Manager 유틸리티가 포함되어 있으면 처음 몇 번의 연결 시도 이후에 Credentials Manager 유틸리티에 보안 인증 정보를 제공하라는 403 오류 또는 프롬프트가 표시됩니다. CodeCommit과 호환되지 않으므로 이 문제를 해결하는 가장 좋은 방법은 Git Credential Manager 유틸리티 옵션이 없는 Windows용 Git을 제거했다가 다시 설치하는 것입니다. Git Credential Manager 유틸리티를 유지하려면 CodeCommit에 연결할 AWS CodeCommit 때에 대한 자격 증명 헬퍼 사용을 지정하도록 .gitconfig 파일을 수동으로 수정하는 등 CodeCommit도 사용하기 위한 추가

구성 단계를 수행해야 합니다. Credential Manager 유틸리티에서 저장된 보안 인증 정보를 제거합니다. 이 유틸리티는 제어판에서 찾을 수 있습니다. 저장된 보안 인증 정보를 제거했으면 .gitconfig 파일에 다음 사항을 추가하고 저장한 다음 새 명령 프롬프트 창에서 다시 연결합니다.

```
[credential "https://git-codecommit.us-east-2.amazonaws.com"]
    helper = !aws codecommit credential-helper $@
    UseHttpPath = true

[credential "https://git-codecommit.us-east-1.amazonaws.com"]
    helper = !aws codecommit credential-helper $@
    UseHttpPath = true
```

모든 연결이 예상대로 작동하기 전에 --global 또는 --local 대신 --system을 지정하여 git config 설정을 다시 구성해야 할 수도 있습니다.

- CodeCommit에 대해 동일한 로컬 시스템에서 다른 IAM 사용자를 사용하려면 대신를 지정 git config --localgit config --global하고 각 AWS 자격 증명 프로파일에 대한 구성을 실행해야 합니다.

2. git config --global --edit를 실행하여 이전 값이 사용자 프로필의 .gitconfig 파일에 기록되었는지 확인합니다(기본적으로 `%HOME%\.gitconfig` 또는 *`drive`*`:\Users\`*`UserName`*`\.gitconfig`). 기록되었다면 Git 전역 구성 파일에 이미 있는 값뿐 아니라 이전 값도 보여야 합니다. 종료하려면 일반적으로 **`:q`**를 입력한 다음 Enter를 누릅니다.

4단계: CodeCommit 콘솔 연결 및 리포지토리 복제

관리자가 CodeCommit 리포지토리에 대한 이름과 연결 세부 정보를 이미 전송한 경우 이 단계를 건너뛰고 리포지토리를 직접 복제할 수 있습니다.

CodeCommit 리포지토리에 연결하려면

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리전 선택기에서 리포지토리가 생성된 AWS 리전 를 선택합니다. 리포지토리는에 고유합니다 AWS 리전. 자세한 내용은 리전 및 Git 연결 엔드포인트 단원을 참조하십시오.
3. 목록에서 연결하려는 리포지토리를 찾아서 선택합니다. URL 복제를 선택한 다음 리포지토리를 복제하거나 연결할 때 사용할 프로토콜을 선택합니다. 이것으로 복제 URL가 복사됩니다.

- IAM 사용자를 통한 Git 보안 인증 정보를 활용하거나 AWS CLI에 포함된 보안 인증 도우미를 사용하는 경우 HTTPS URL을 복사합니다.
- 로컬 컴퓨터에서 git-remote-codecommit 명령을 사용하는 경우 HTTPS(GRC) URL을 복사합니다.
- IAM 사용자와 SSH 퍼블릭/프라이빗 키 페어를 사용하는 경우 SSH URL을 복사합니다.

> **Note**
>
> 리포지토리 목록 대신 시작 페이지가 표시되면 로그인한에 AWS 계정과 연결된 AWS 리전 리포지토리가 없는 것입니다. 리포지토리를 만들려면 the section called "리포지토리 생성"를 참조하거나 Git 및 CodeCommit 시작하기 자습서의 다음 단계를 따르세요.

4. 명령 프롬프트를 열고 복사한 HTTPS URL을 사용하여 git clone 명령을 실행합니다. 그러면 명령을 실행한 디렉터리의 하위 디렉터리에 로컬 리포지토리가 생성됩니다. 예를 들어, *MyDemoRepo*라는 리포지토리를 미국 동부(오하이오) 리전의 *my-demo-repo*라는 로컬 리포지토리에 복제하려면 다음과 같이 합니다.

```
git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-
demo-repo
```

일부 Windows 버전에서는 사용자 이름과 암호를 묻는 팝업 메시지가 표시될 수 있습니다. 이것은 Windows용 기본 보안 인증 정보 관리 시스템이지만 AWS CodeCommit의 보안 인증 도우미와 호환되지 않습니다. 취소를 선택합니다.

다음 단계

사전 필수 단계를 완료했습니다. CodeCommit 시작하기 에 나와 있는 단계에 따라 CodeCommit의 사용을 시작합니다.

시작

CodeCommit을 시작하는 가장 쉬운 방법은 CodeCommit 시작하기 에 제시된 단계를 따르는 것입니다. Git과 CodeCommit을 처음 접하는 경우에는 Git 및 CodeCommit 시작하기에 제시된 단계도 수행하는 것이 좋습니다. 각 단계를 모두 완료하면 CodeCommit에 익숙해질 수 있으며 CodeCommit 리포지토리와 상호 작용할 때 Git을 사용하기 위한 기본 사항들을 익힐 수 있습니다.

또 CodePipeline과 CodeCommit으로 하는 간단한 파이프라인 연습의 자습서를 따라하면, CodeCommit 리포지토리를 지속적인 전달 파이프라인의 일부로 사용하는 방법에 대해 배울 수 있습니다.

이 단원의 자습서는 다음과 같은 필수 구성 요소 및 설정을 완료했다는 가정 하에 작성되었습니다.

- IAM 사용자에게 권한 할당
- 이 자습서에 사용하는 로컬 시스템에 HTTPS 또는 SSH 연결을 위한 자격 증명 관리 설정
- 리포지토리 생성을 포함한 모든 작업에 명령줄 또는 터미널을 사용하려는 AWS CLI 경우를 구성합니다.

주제

- 시작하기 AWS CodeCommit
- Git 및 시작하기 AWS CodeCommit

시작하기 AWS CodeCommit

본 자습서는 CodeCommit의 몇 가지 주요 기능에 대한 사용 방법을 소개합니다. 먼저 리포지토리를 생성하고 몇 가지 변경 사항을 커밋합니다. 그런 다음 파일을 검색하고 해당 변경 사항을 확인합니다. 다른 사용자에게 코드 변경 내용을 검토하고 의견을 제시할 수 있게 풀 요청을 생성할 수도 있습니다.

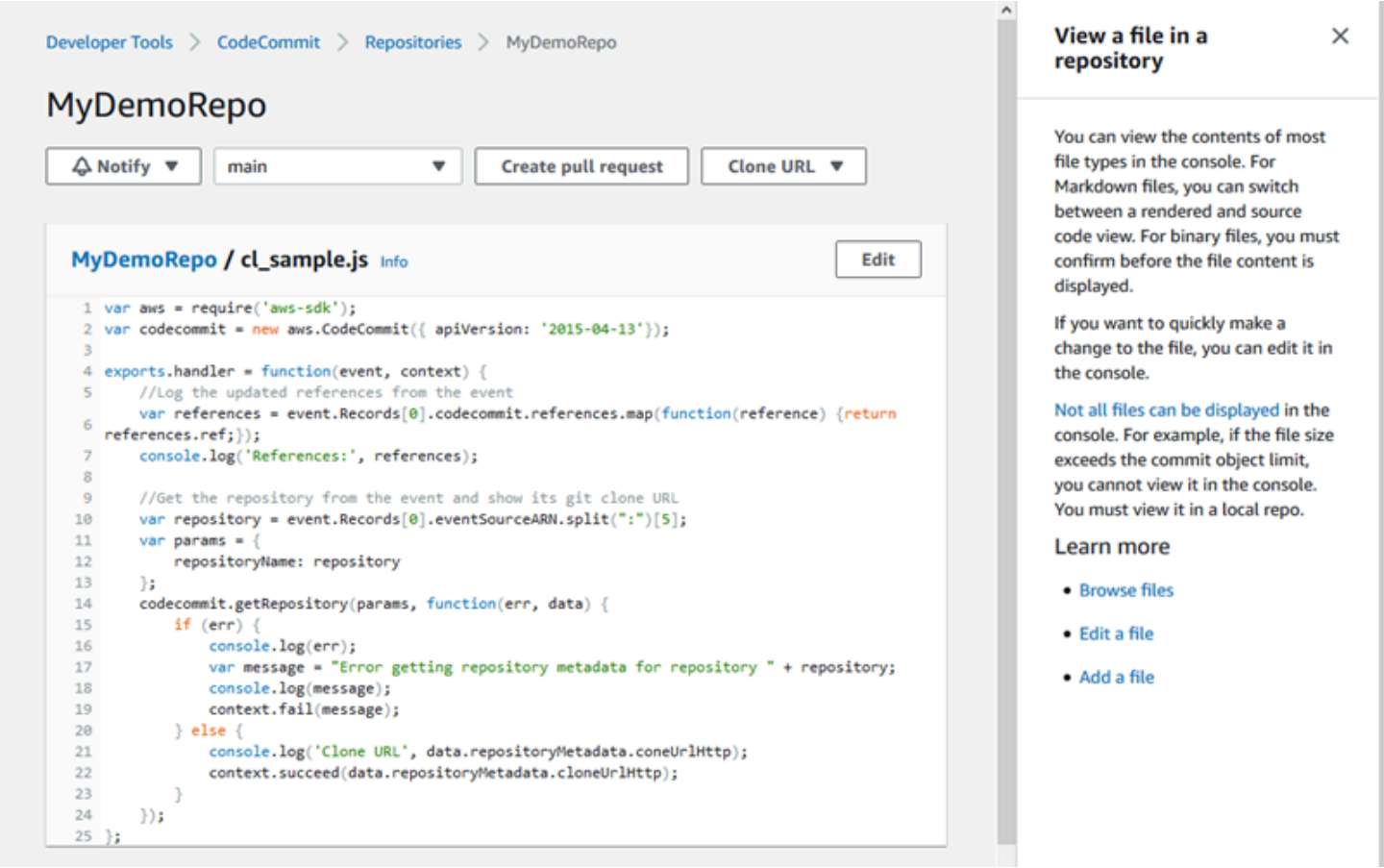
기존 코드를 CodeCommit으로 마이그레이션하려면 로 마이그레이션 AWS CodeCommit 단원을 참조하세요.

Git에 익숙하지 않다면 Git 및 CodeCommit 시작하기 섹션도 완료하는 것이 좋습니다. 이 자습서를 완료한 후 충분한 연습을 거치면 CodeCommit을 자체 프로젝트 및 팀 환경에서 사용할 수 있습니다.

CodeCommit 콘솔은 축소 가능한 패널에 유용한 정보를 담고 있습니다. 이 패널은 해당 페이지의 정보 아이콘

( )

이나 정보 링크를 통해 열 수 있습니다. 사용자는 이 패널을 언제든 닫을 수 있습니다.



CodeCommit 콘솔에서는 리포지토리, 구축 프로젝트, 배포 애플리케이션, 파이프라인과 같은 리소스를 신속하게 검색할 수도 있습니다. 리소스로 이동을 선택하거나 / 키를 누른 후 리소스 이름을 입력하세요. 목록에 일치 항목이 나타납니다. 검색은 대/소문자를 구분하지 않습니다. 보기 권한이 있는 리소스만 표시됩니다. 자세한 내용은 콘솔에서 리소스 보기 섹션을 참조하세요.

사전 조건

시작에 앞서 다음과 같은 사전 조건 및 설정 절차를 완료합니다.

- IAM 사용자에게 권한 할당
- 본 자습서에 사용되는 로컬 시스템에서 HTTPS 또는 SSH 접속에 필요한 보안 인증 정보 관리 설정하기.
- 리포지토리 생성을 포함한 모든 작업에 명령줄 또는 터미널을 AWS CLI 사용하려면를 구성합니다.

주제

- 1단계: CodeCommit 리포지토리 생성
- 2단계: 리포지토리에 파일 추가
- 3단계: 리포지토리의 콘텐츠 검색
- 4단계: 풀 요청 생성 및 협업
- 5단계: 정리
- 6단계: 다음 단계

1단계: CodeCommit 리포지토리 생성

CodeCommit 콘솔을 사용하여 CodeCommit 리포지토리를 만들 수 있습니다. 본 자습서에 활용할 리포지토리를 이미 보유하고 있으면 이 단계를 건너뛰어도 됩니다.

> **Note**
>
> 사용자는 사용량에 따라 리포지토리 생성 또는 액세스에 대한 비용이 부과될 수 있습니다. 자세한 내용은 CodeCommit 제품 정보 페이지에서 요금을 참조하세요.

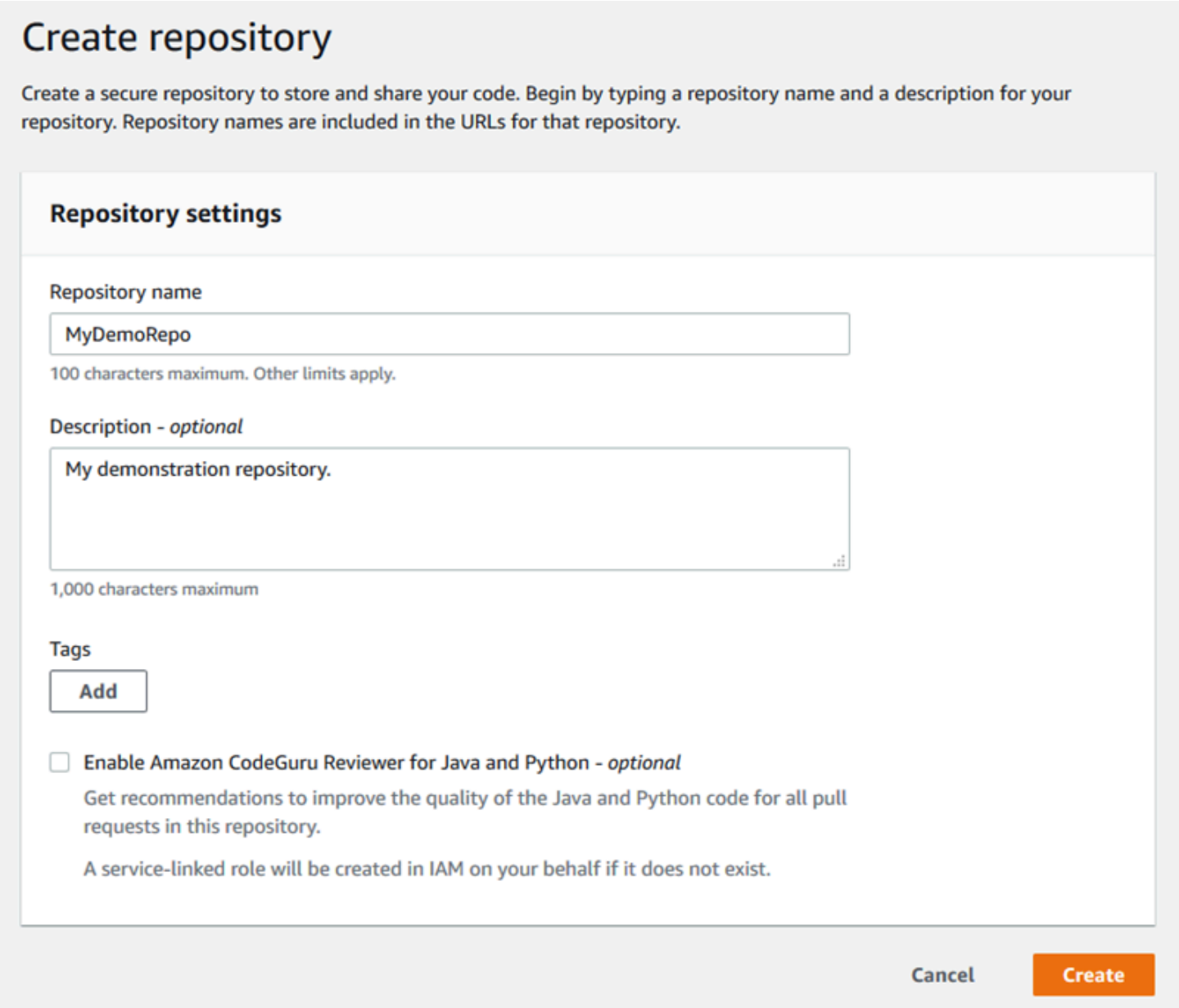
CodeCommit 리포지토리를 생성하려면

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리전 선택기를 사용하여 리포지토리를 생성할 AWS 리전 를 선택합니다. 자세한 내용은 리전 및 Git 연결 엔드포인트 단원을 참조하십시오.
3. 리포지토리 페이지에서 리포지토리 생성을 선택합니다.
4. 리포지토리 생성 페이지에서 리포지토리 이름에 해당 리포지토리의 이름(예: **`MyDemoRepo`**)을 입력합니다.

   > **Note**
   >
   > 리포지토리 이름은 대/소문자를 구분하며 100자를 초과할 수 없습니다. 자세한 내용은 제한을 참조하세요.

5. (선택 사항) 설명 란에 설명 내용을 입력합니다(예: **`My demonstration repository`**). 그러면 사용자들이 리포지토리의 용도를 식별하는 데 도움이 됩니다.

6. (선택 사항) 태그 추가를 선택하여 하나 이상의 리포지토리 태그(AWS 리소스를 구성하고 관리하는 데 도움이 되는 사용자 지정 속성 레이블)를 리포지토리에 추가합니다. 자세한 내용은 에서 리포지토리 태그 지정 AWS CodeCommit 단원을 참조하십시오.
7. (선택 사항) 추가 구성을 확장하여이 리포지토리의 데이터를 암호화하고 복호화하기 위해 기본 AWS 관리형 키 또는 자체 고객 관리형 키를 사용할지 여부를 지정합니다. 자체 고객 관리형 키를 사용하기로 선택한 경우 리포지토리를 생성하는 AWS 리전 에서 사용할 수 있고 키가 활성 상태인지 확인해야 합니다. 자세한 내용은 AWS Key Management Service 및 AWS CodeCommit 리포지토리 암호화 단원을 참조하십시오.
8. (선택 사항) 이 리포지토리에 Java나 Python 코드가 포함될 예정이고 해당 코드를 CodeGuru Reviewer로 분석하고 싶다면, Java 또는 Python용 Amazon CodeGuru Reviewer 활성화를 선택합니다. CodeGuru Reviewer는 다양한 기계 학습 모델을 사용하여 코드의 결함을 찾아내고 풀 요청의 개선점과 해결책을 자동으로 제안합니다. 자세한 내용은 Amazon CodeGuru Reviewer 사용 설명서를 참조하세요.
9. 생성(Create)을 선택합니다.



> **Note**
>
> 해당 리포지토리에 `MyDemoRepo`이 아닌 다른 이름을 사용하는 경우, 나머지 단계에서 그 이름을 사용해야 합니다.

리포지토리가 열리면 CodeCommit 콘솔에서 직접 파일을 추가하는 방법에 대한 내용이 표시됩니다.

2단계: 리포지토리에 파일 추가

리포지토리에 파일을 추가하려면 다음을 수행해야 합니다.

- CodeCommit 콘솔에서 파일 생성하기. 콘솔에서 임의의 리포지토리에 파일을 처음으로 생성하면 main이라는 브랜치가 자동으로 생성됩니다. 이 브랜치는 해당 리포지토리의 기본 브랜치입니다.
- CodeCommit 콘솔을 사용하여 로컬 컴퓨터에서 파일을 업로드하기. 콘솔에서 리포지토리에 파일을 처음으로 업로드하면 main이라는 브랜치가 자동으로 생성됩니다. 이 브랜치는 해당 리포지토리의 기본 브랜치입니다.
- Git 클라이언트를 사용하여 로컬 컴퓨터로 리포지토리를 복제한 다음, CodeCommit 리포지토리에 파일을 추가, 커밋, 푸시합니다. Git에서 처음 커밋할 때는 브랜치가 자동으로 생성되며 이 브랜치가 리포지토리의 기본 브랜치로 설정됩니다. 브랜치 이름은 Git 클라이언트의 기본 설정에 따라 결정됩니다. 첫 브랜치의 이름으로 main을 사용하도록 Git 클라이언트 구성합니다.

> **Note**
>
> 사용자는 언제든 브랜치를 생성하고 리포지토리의 기본 브랜치를 변경할 수 있습니다. 자세한 내용은 AWS CodeCommit 리포지토리의 브랜치 작업 단원을 참조하세요.

가장 간단한 시작 방법은 CodeCommit 콘솔을 열고 파일을 추가하는 것입니다. 이렇게 하면 리포지토리에 main이라는 기본 브랜치도 생성됩니다. 를 사용하여 리포지토리에 파일을 추가하고 첫 번째 커밋을 생성하는 방법에 대한 지침은를 사용하여 리포지토리에 대한 첫 번째 커밋 생성을 AWS CLI참조하세요. AWS CLI

리포지토리에 파일을 추가하려면

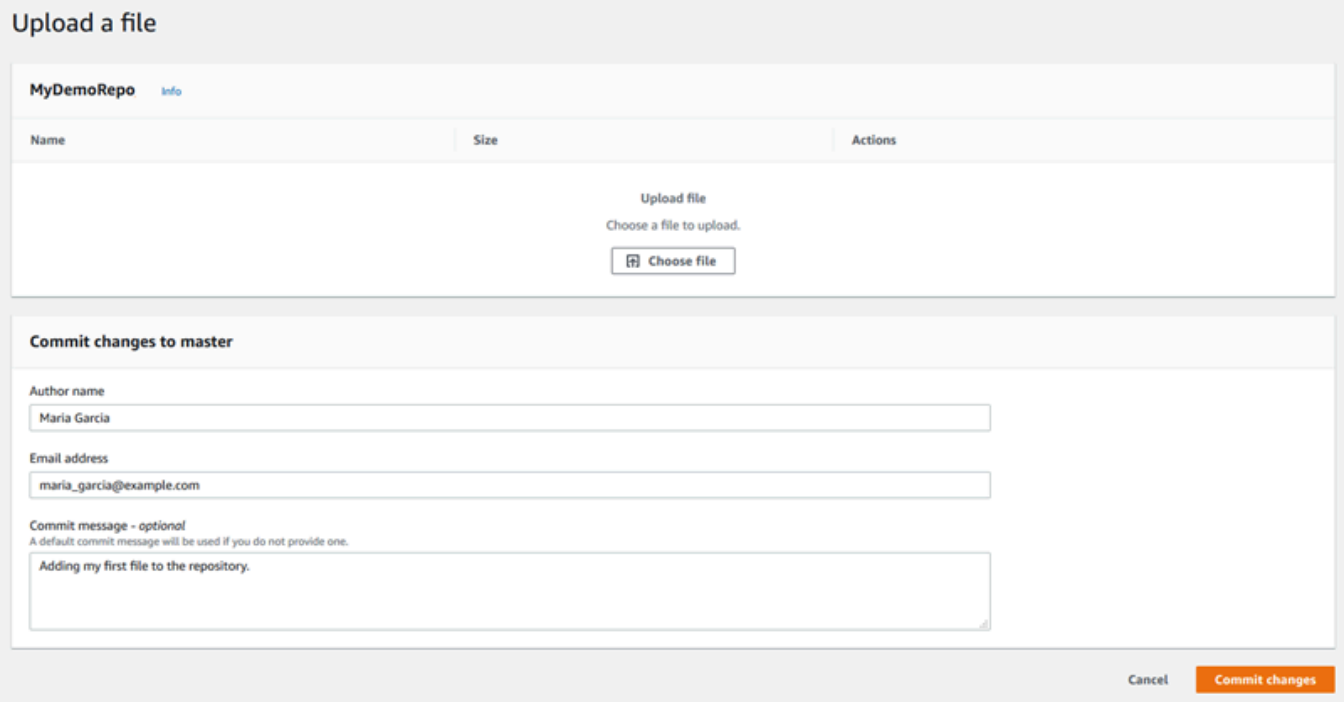
1. 해당 리포지토리의 탐색 모음에서 코드를 선택합니다.
2. 파일 추가를 선택한 다음, 컴퓨터에서 파일 생성 또는 파일을 업로드를 선택합니다. 이 자습서에서는 두 작업을 수행하는 방법을 모두 보여 줍니다.
3. 파일을 추가하려면 다음을 수행합니다.
   a. 브랜치 드롭다운 목록에서 파일을 추가할 브랜치를 선택합니다. 기본 브랜치가 자동으로 선택됩니다. 여기서 소개하는 예제에서는 기본 브랜치의 명칭이 main으로 지정됩니다. 파일을 다른 브랜치에 추가하려면 다른 브랜치를 선택합니다.
   b. 파일 이름에 파일의 이름을 입력합니다. 코드 편집기에서 파일의 코드를 입력합니다.
   c. 작성자 이름에 다른 리포지토리 사용자에게 표시할 이름을 입력합니다.
   d. 이메일 주소에 이메일 주소를 입력합니다.

e. (선택 사항) 커밋 메시지에 간단한 메시지를 입력합니다. 이것은 선택 사항이지만 이 파일을 추가한 이유를 팀원들이 이해하는 데 도움이 되므로 커밋 메시지를 추가하는 것이 좋습니다. 커밋 메시지를 입력하지 않으면 기본 메시지가 사용됩니다.

f. 변경 사항 커밋을 선택합니다.

파일을 업로드하려면 다음을 수행합니다.

- 파일을 업로드하려는 경우에는 업로드할 파일을 선택합니다.



- 작성자 이름에 다른 리포지토리 사용자에게 표시할 이름을 입력합니다.
- 이메일 주소에 이메일 주소를 입력합니다.
- (선택 사항) 커밋 메시지에 간단한 메시지를 입력합니다. 이것은 선택 사항이지만 이 파일을 추가한 이유를 팀원들이 이해하는 데 도움이 되므로 커밋 메시지를 추가하는 것이 좋습니다. 커밋 메시지를 입력하지 않으면 기본 메시지가 사용됩니다.
- 변경 사항 커밋을 선택합니다.

자세한 내용은 AWS CodeCommit 리포지토리의 파일 작업 단원을 참조하십시오.

Git 클라이언트를 사용해 리포지토리를 복제하려면, 로컬 컴퓨터에 Git을 설치한 다음 CodeCommit 리포지토리를 복제합니다. 다음으로 로컬 리포지토리에 파일을 몇 개 추가하고 그 파일들을 CodeCommit 리포지토리로 푸시합니다. 자세한 설명은 Git 및 CodeCommit 시작하기 섹션을 참조하세

요. Git에는 익숙하지만 CodeCommit 리포지토리에서 이 작업을 수행하는 방법을 잘 모르는 경우에는 커밋 생성, 2단계: 로컬 리포지토리 생성, 리포지토리에 연결 섹션에서 예제와 지침을 참조할 수 있습니다.

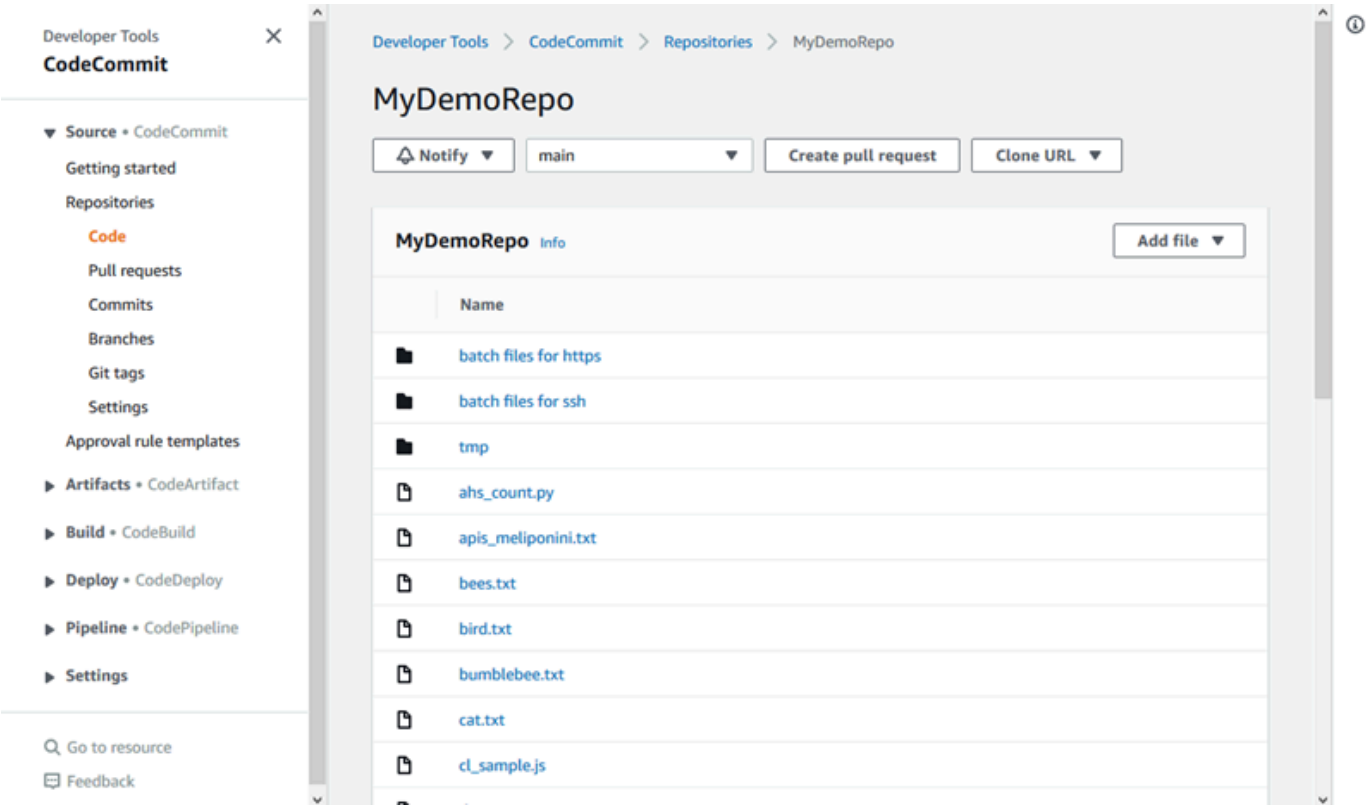
CodeCommit 리포지토리에 파일을 추가했으면 콘솔에서 해당 파일을 볼 수 있습니다.

3단계: 리포지토리의 콘텐츠 검색

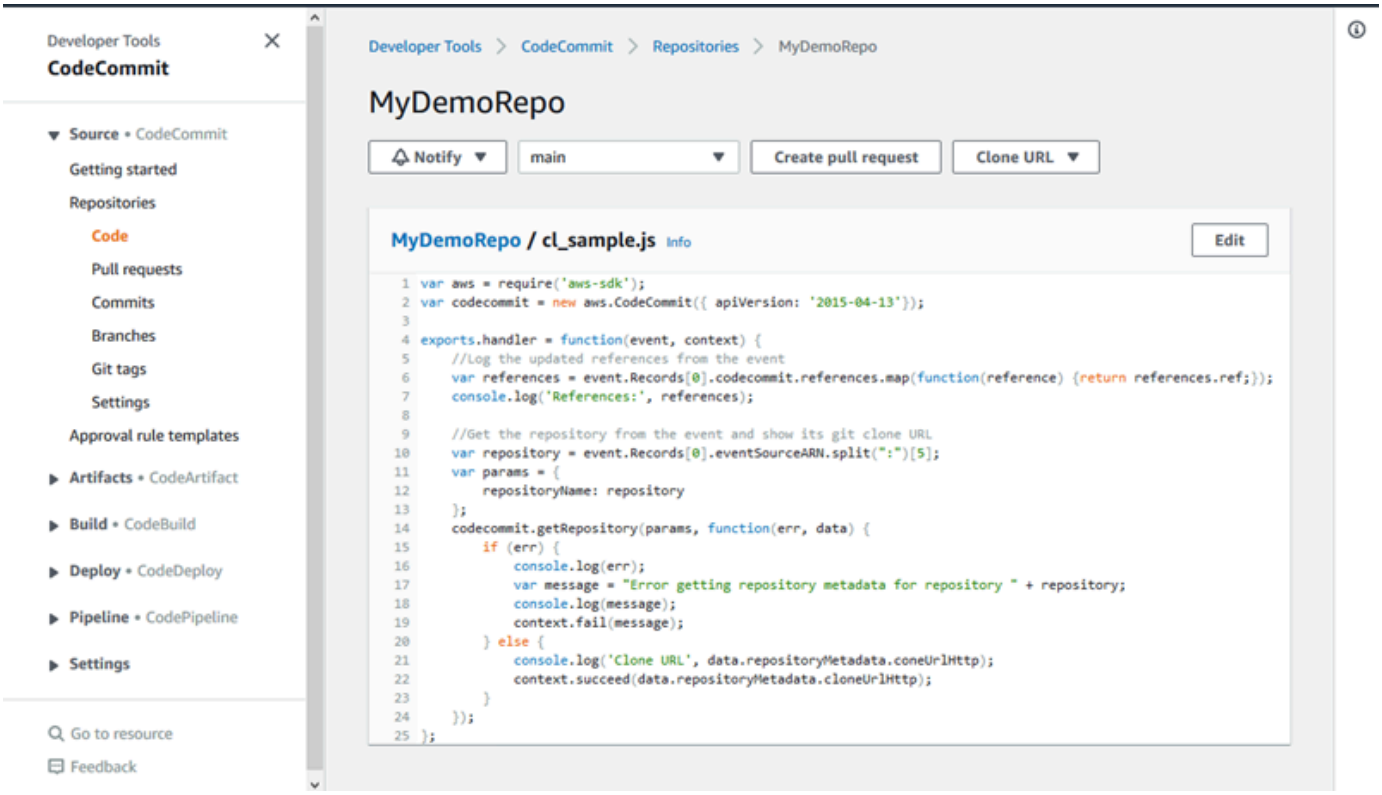
사용자는 CodeCommit 콘솔을 사용하여 리포지토리의 파일을 검토하거나 파일의 콘텐츠를 빠르게 훑어볼 수 있습니다. 이렇게 하면 체크아웃할 브랜치를 결정하거나 리포지토리의 로컬 복사본 생성 여부를 결정하는 데 도움이 됩니다.

리포지토리를 검색하려면

1. 리포지토리에서 MyDemorepo를 선택합니다.
2. 이 페이지에는 리포지토리의 기본 브랜치에 있는 콘텐츠가 표시됩니다. 다른 브랜치를 보거나 특정 태그의 코드를 보려면 목록에서 보려는 브랜치 또는 태그를 선택합니다. 다음 스크린샷에서 보기는 main 브랜치로 설정되어 있습니다.

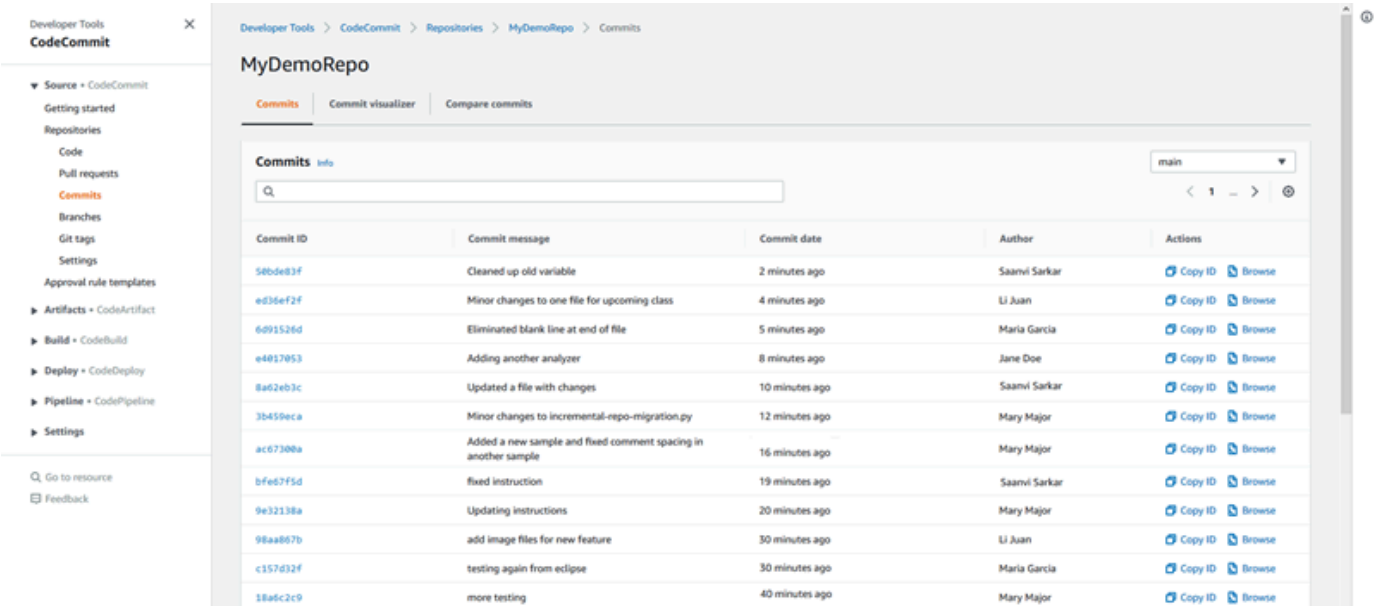


3. 자신의 리포지토리에 있는 파일의 콘텐츠를 보려면 목록에서 해당 파일을 선택합니다. 표시된 코드의 색상을 변경하려면 설정 아이콘을 선택합니다.



자세한 내용은 리포지토리에서 파일 검색 단원을 참조하십시오.

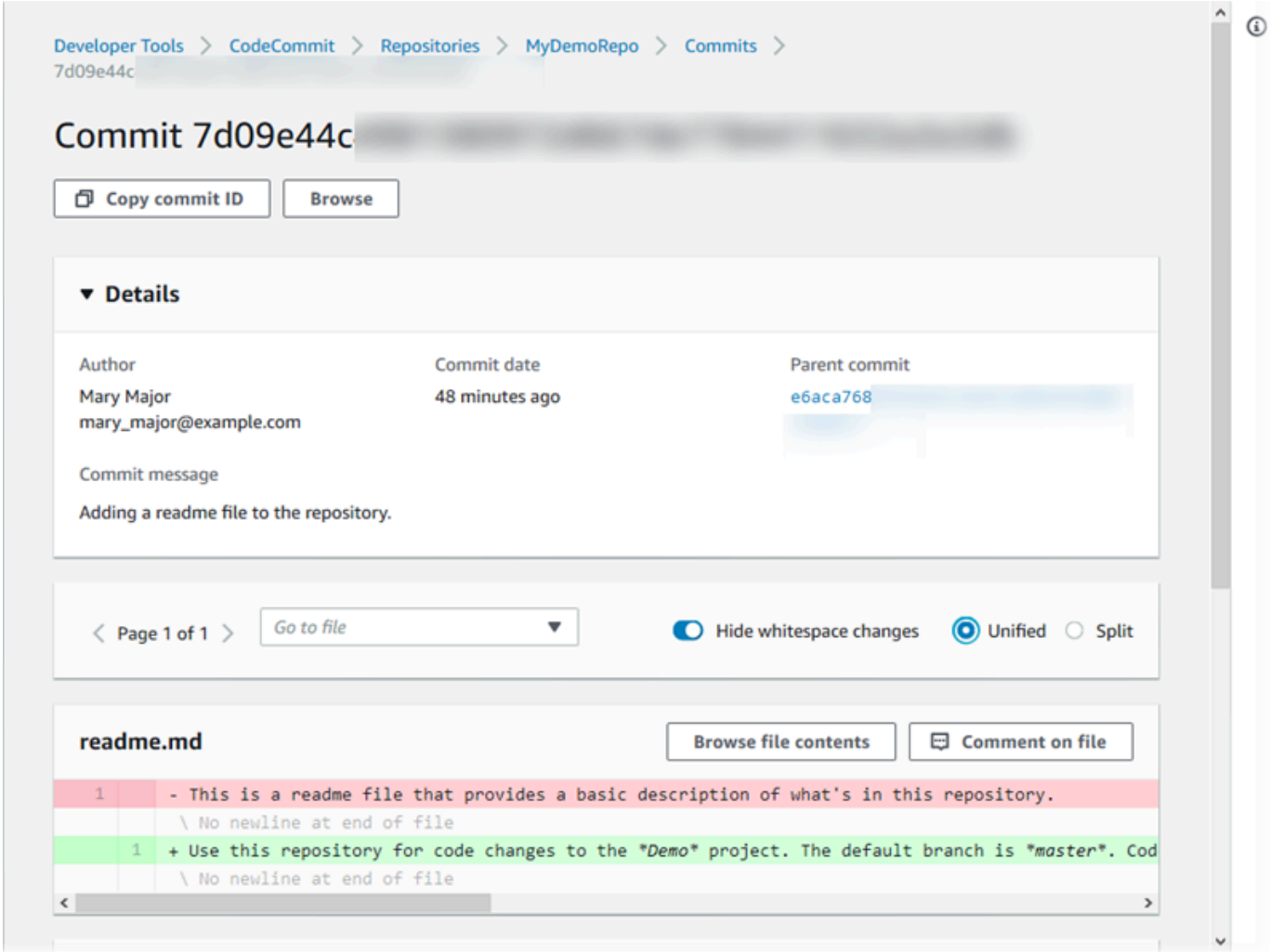
4. 리포지토리의 커밋 이력을 검색하려면 커밋을 선택합니다. 콘솔은 기본 브랜치의 커밋 이력을 시간 역순으로 표시합니다. 작성자, 날짜 등을 기준으로 커밋 세부 정보를 검토합니다.



5. 브랜치 또는 Git 태그별로 커밋 이력을 보려면 목록에서 보려는 브랜치 또는 태그를 선택합니다.

6. 커밋과 그 부모 커밋 간 차이를 보려면 축약된 커밋 ID를 선택하십시오. 공백 변경 사항 표시 여부와 변경을 인라인으로 표시할지(Unified 보기) 아니면 나란히 표시할지(Split 보기) 여부 등 변경 내용의 표시 방식을 선택할 수 있습니다.

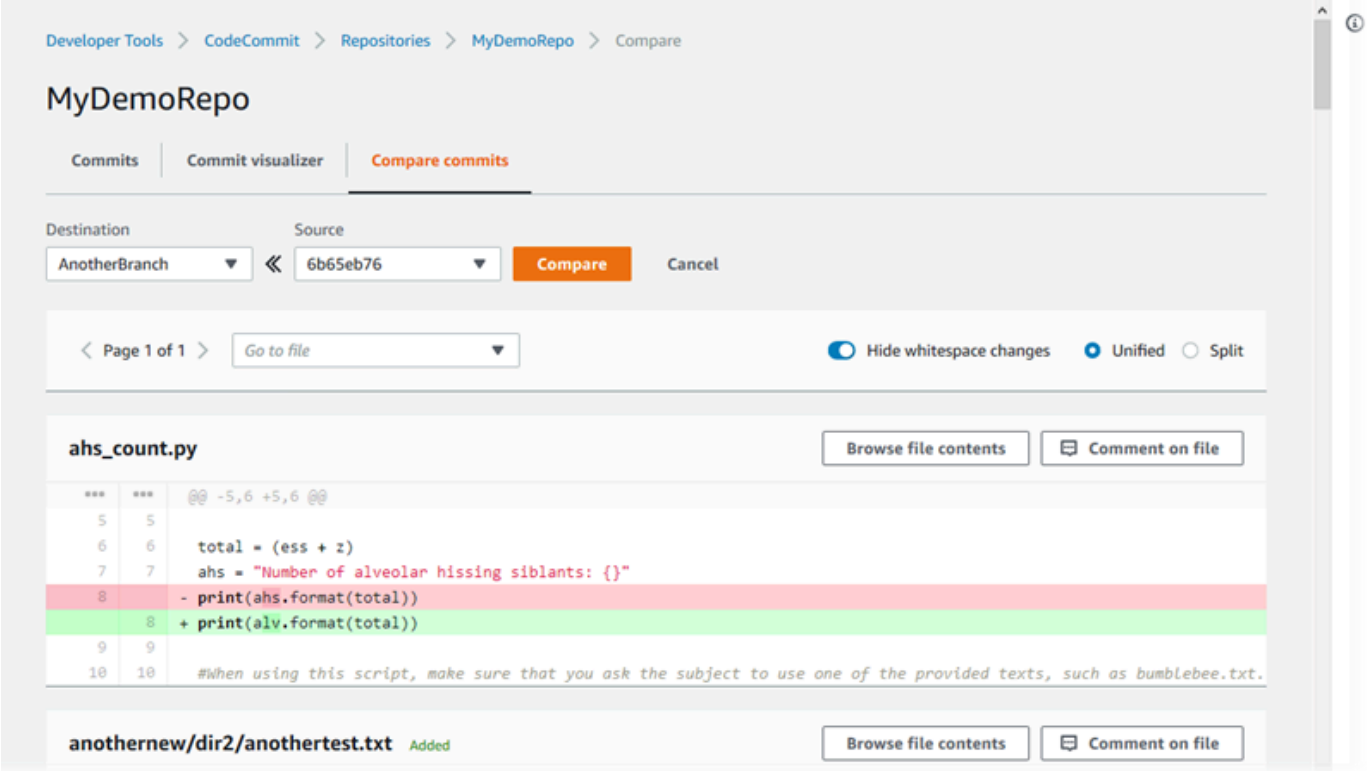
> **Note**
>
> 코드 및 다른 콘솔 설정 보기에 대한 기본 설정은 변경이 생길 때마다 브라우저 쿠키로 저장됩니다. 자세한 내용은 사용자 기본 설정으로 작업 단원을 참조하십시오.



7. 커밋에 대한 모든 주석을 보려면 해당 커밋을 선택한 후 변경 내용을 스크롤하여 인라인으로 표시합니다. 자신의 주석을 추가하고 다른 사람의 주석에 답변할 수도 있습니다.

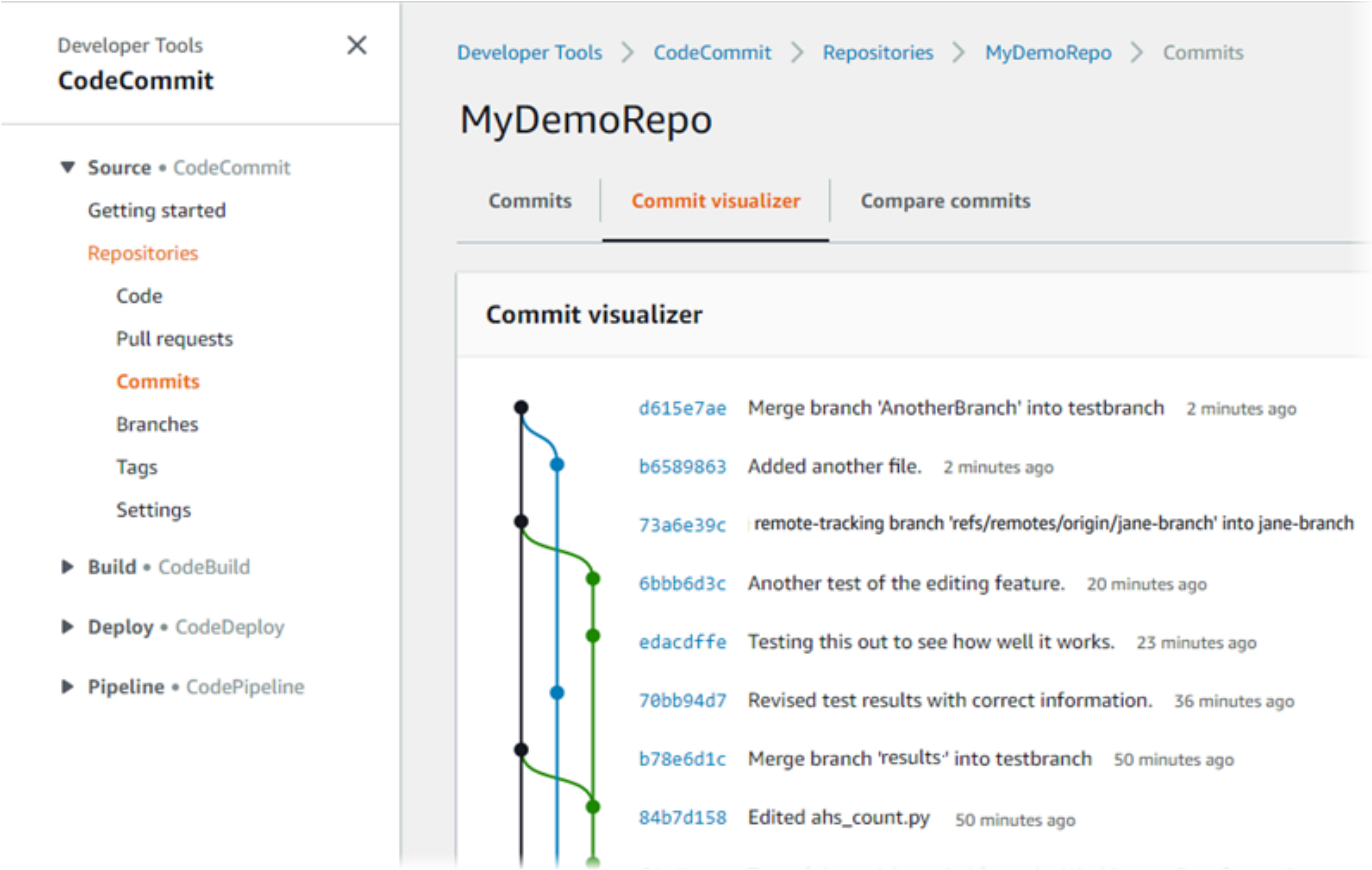
   자세한 내용은 커밋에 대한 주석 추가 단원을 참조하십시오.

8. 탐색 창에서 태그, 브랜치, 커밋 ID 등 두 커밋 지정자 간의 차이를 보려면 커밋을 선택한 다음 커밋 비교를 선택합니다.



자세한 정보는 리포지토리의 커밋 이력 검색 및 커밋 비교 섹션을 참조하세요.

9. 커밋에서 커밋 시각화 도우미 탭을 선택합니다.



그래프의 각 시점 옆에 커밋의 제목이 있는 커밋 그래프가 표시됩니다. 제목 표시는 80자로 제한됩니다.

10. 커밋에 대한 자세한 내용을 보려면 축약된 커밋 ID를 선택합니다. 특정 커밋에서 그래프를 렌더링하려면 그래프에서 해당 점을 선택합니다. 자세한 내용은 리포지토리의 커밋 이력을 그래프 형태로 보기 단원을 참조하십시오.

4단계: 풀 요청 생성 및 협업

리포지토리에서 다른 사용자와 함께 작업할 때 코드를 공동 작업하고 변경 내용을 검토할 수 있습니다. 다른 사용자가 브랜치의 코드 변경 사항을 검토하고 의견을 제시할 수 있도록 풀 요청을 생성할 수 있습니다. 풀 요청에 대해 하나 이상의 승인 규칙을 생성할 수도 있습니다. 예를 들어 두 명 이상의 다른 사용자가 풀 요청을 승인해야 병합할 수 있는 승인 규칙을 생성할 수 있습니다. 풀 요청이 승인되면 해당 변경 내용을 대상 브랜치에 병합할 수 있습니다. 리포지토리에 대한 알림을 설정하면, 리포지토리 사용자는 리포지토리 이벤트(예: 풀 요청 또는누군가가 코드에 주석을 다는 경우)에 대한 이메일을 받을 수 있습니다. 자세한 내용은 AWS CodeCommit 리포지토리에서 이벤트 알림 구성 단원을 참조하십시오.

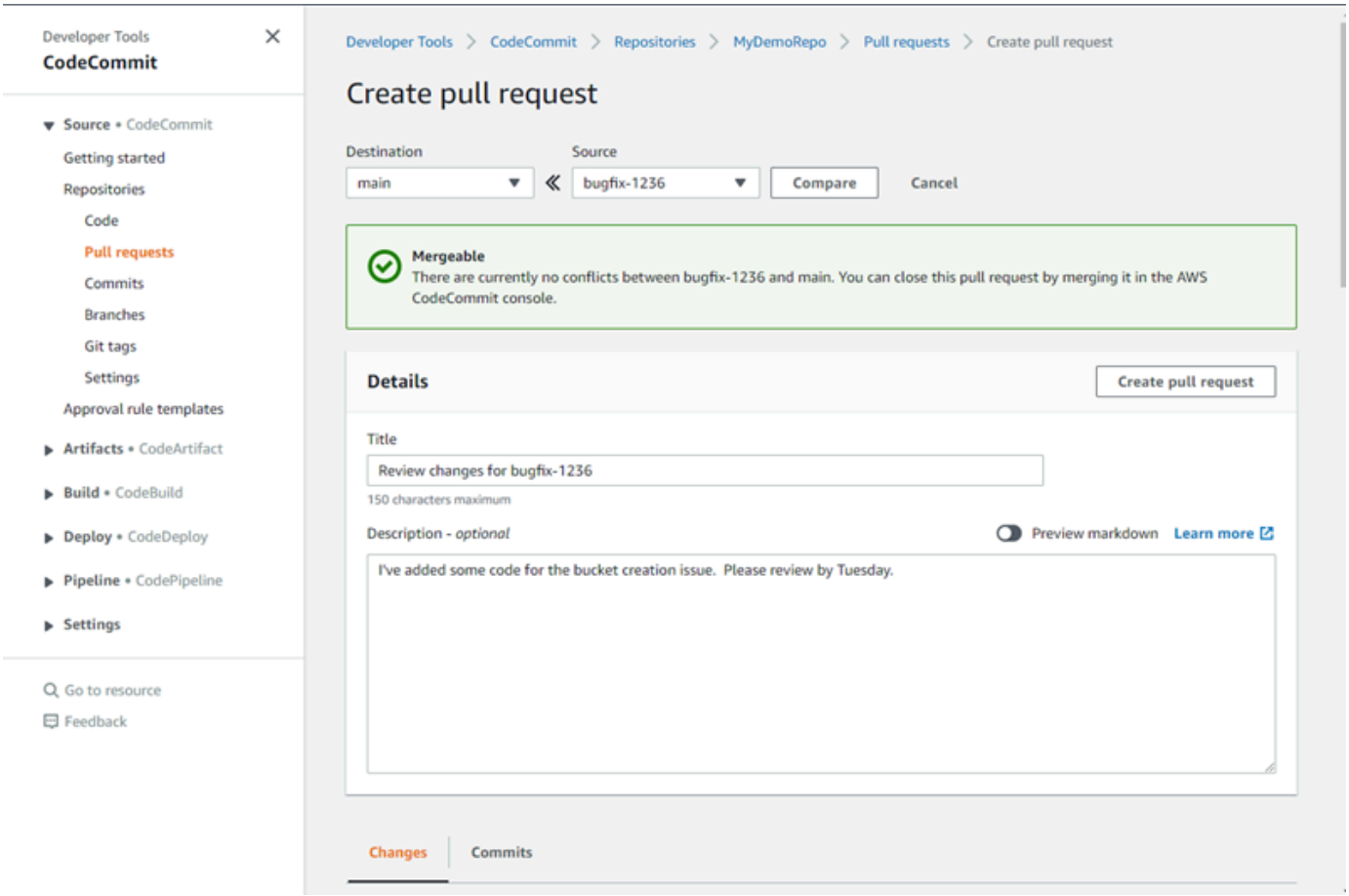
> **Important**
>
> 풀 요청을 생성하려면 먼저 검토할 코드 변경 내용을 포함하는 브랜치를 생성해야 합니다. 자세한 내용은 브랜치 생성 단원을 참조하십시오.

풀 요청을 생성하고 공동 작업하려면

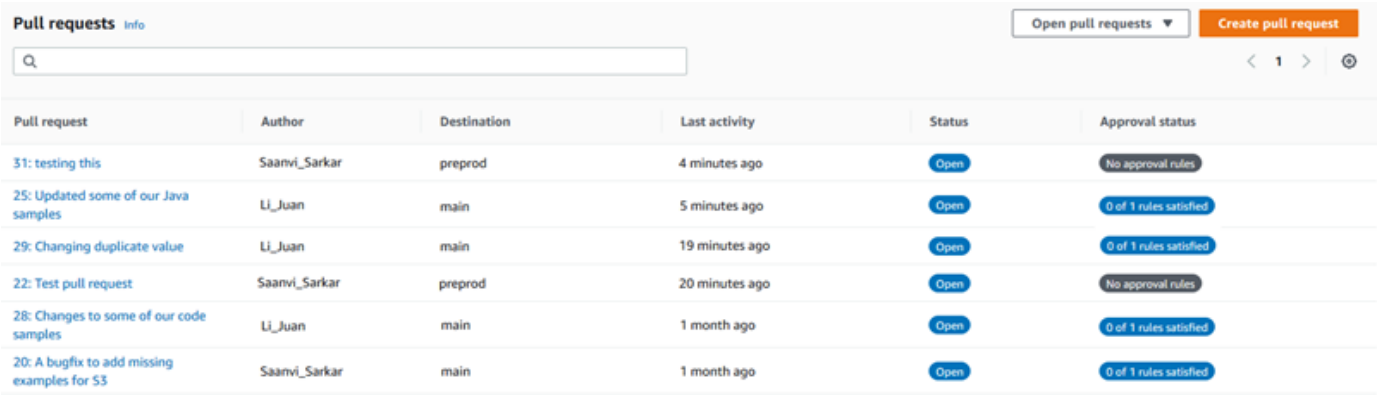
1. 탐색 창에서 풀 요청을 선택합니다.
2. 풀 요청에서 풀 요청 생성을 선택합니다.

   > **Tip**
   >
   > 또한 브랜치 및 코드에서 풀 요청을 생성할 수도 있습니다.

   풀 요청 생성의 소스에서, 검토하려는 변경 사항이 포함된 브랜치를 선택합니다. 대상 주소에서, 풀 요청이 닫힐 때 검토한 코드를 병합할 브랜치를 선택합니다. 비교를 선택합니다.
3. 병합 세부 정보와 변경 내용을 검토하여 풀 요청에 검토를 요청할 변경 내용과 커밋이 포함되어 있는지 확인합니다. 모두 포함되었으면 제목에서 이 검토 항목의 제목을 지정합니다. 이 제목은 리포지토리의 풀 요청 목록에 표시되는 제목입니다. 설명에 이 검토 사항에 대한 세부 정보와 검토자에 대한 기타 유용한 정보를 입력합니다. 생성(Create)을 선택합니다.



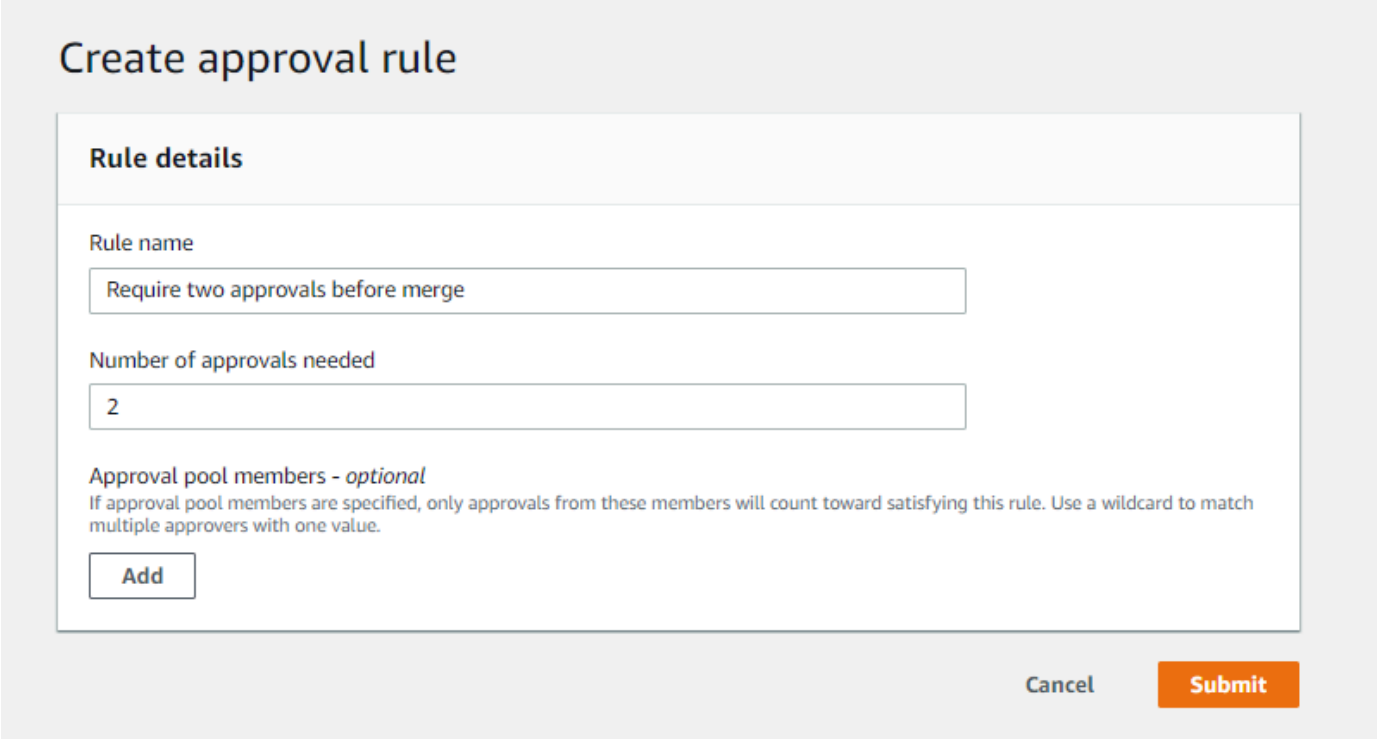
4. 풀 요청이 리포지토리의 풀 요청 목록에 나타납니다. 보기를 필터링하여 열린 요청, 닫힌 요청, 본인이 생성한 요청 등만 표시할 수 있습니다.



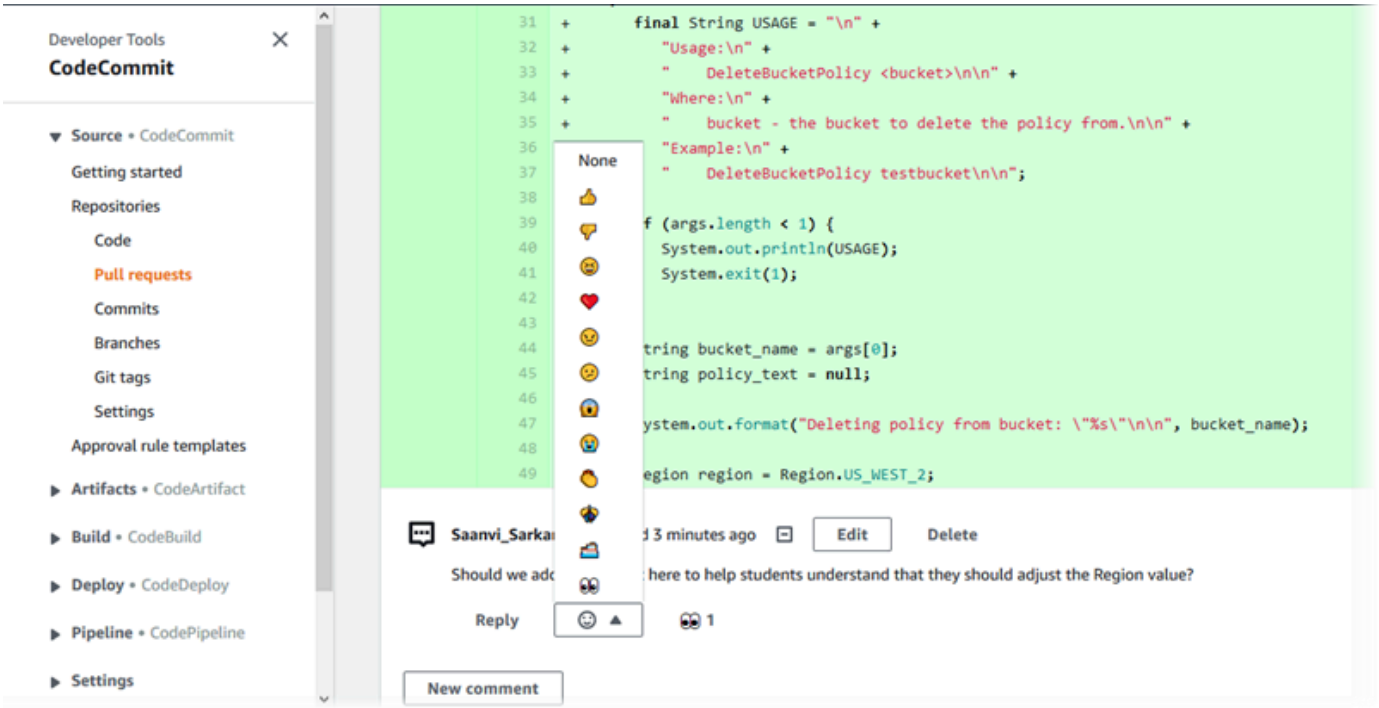
5. 풀 요청을 병합하기 전에 특정 조건이 충족되도록 풀 요청에 승인 규칙을 추가할 수 있습니다. 풀 요청에 승인 규칙을 추가하려면 목록에서 풀 요청을 선택합니다. 승인 탭에서 승인 규칙 생성을 선택합니다.

6. 규칙 이름에서 규칙에 설명이 포함된 이름을 지정합니다. 예를 들어 풀 요청을 병합하기 전에 두 사람이 승인하도록 요구하려면 규칙 이름으로 **`Require two approvals before merge`**를 지정할 수 있습니다. 필요한 승인 수에 원하는 숫자 **2**를 입력합니다. 기본 값은 1입니다. 제출을 선

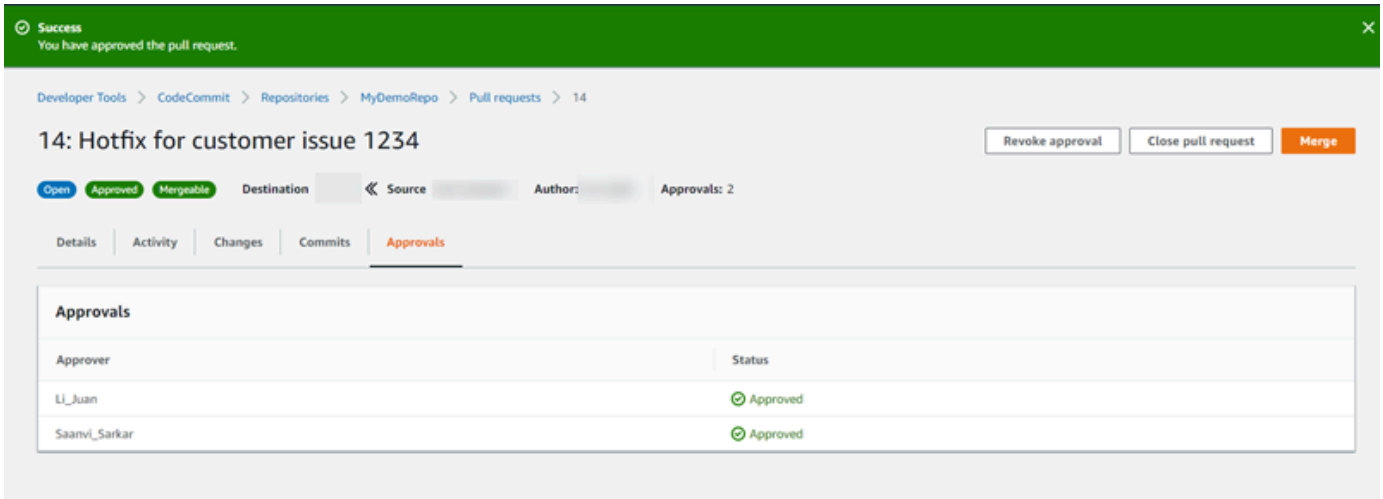
택합니다. 승인 규칙 및 승인 풀 멤버에 대한 자세한 내용은 풀 요청에 대한 승인 규칙 생성 단원을 참조하세요.



7. 리포지토리에 대한 알림을 구성했고 풀 요청 이벤트를 사용자에게 알리기로 한 경우에는 새 풀 요청에 대한 이메일이 사용자에게 발송됩니다. 사용자는 변경 내용을 살펴보고 특정 코드 행, 파일, 풀 요청 자체 등에 대해 주석을 남길 수 있습니다. 주석에 텍스트와 이모티콘으로 댓글을 다는 것도 가능합니다. 필요할 경우, 변경 내용을 풀 요청 브랜치로 푸시할 수 있습니다. 이렇게 하면 풀 요청이 업데이트됩니다.



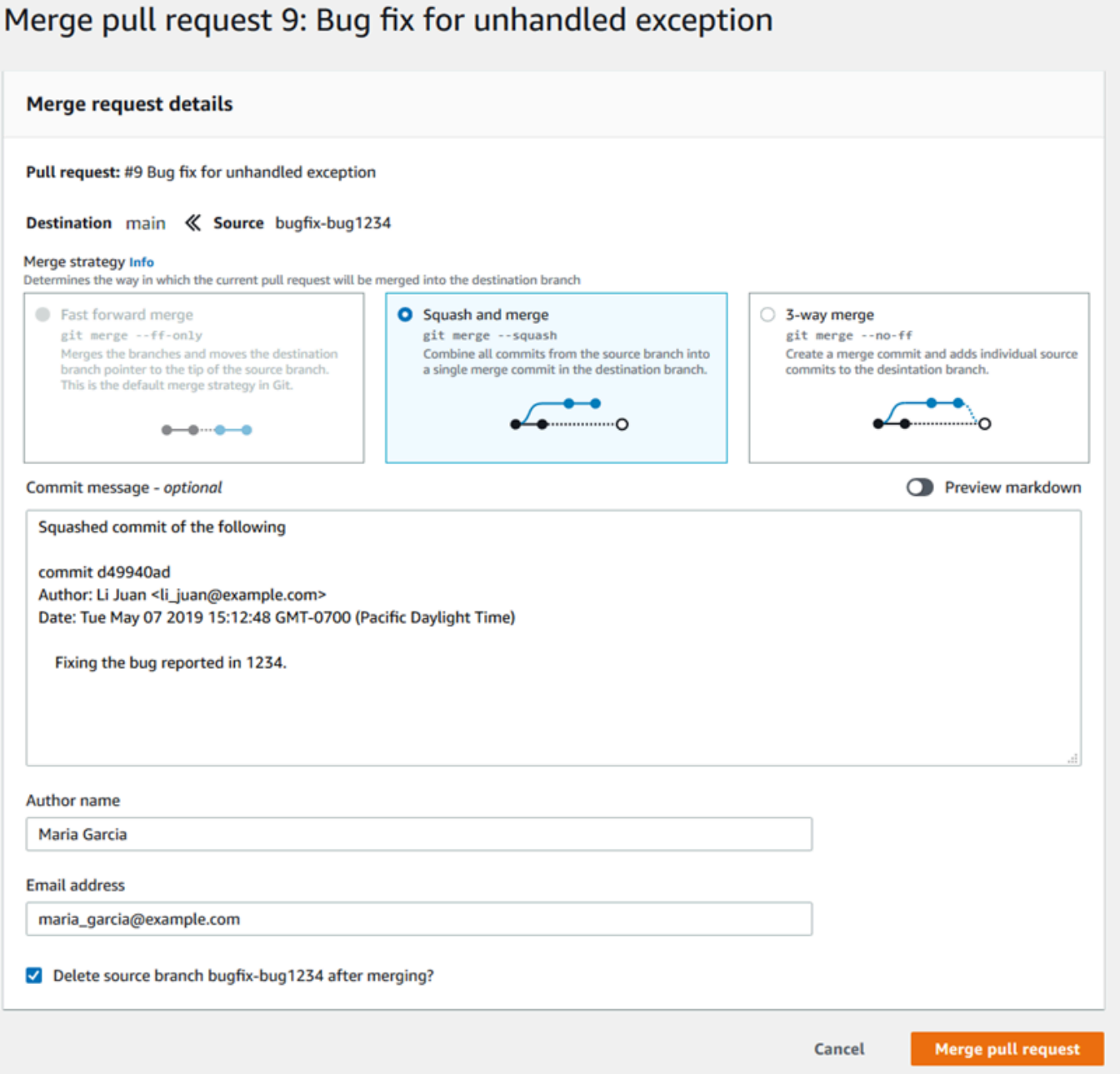
8. 요청에서 변경한 내용이 만족스러우면 승인을 선택합니다. 해당 풀 요청에 대해 승인 규칙이 구성되어 있지 않은 경우에도 풀 요청을 승인하도록 선택할 수 있습니다. 그러면 풀 요청을 검토한 결과와 변경 내용에 대한 승인이 명확하게 기록될 수 있습니다. 생각이 바뀔 경우 승인을 취소하도록 선택할 수도 있습니다.



> **Note**
>
> 풀 요청을 생성한 사용자는 풀 요청을 승인할 수 없습니다.

9. 풀 요청에서 모든 코드 변경 내용을 검토하고 동의했다면 다음 중 하나를 수행합니다.

- 브랜치를 병합하지 않고 풀 요청을 닫으려면 풀 요청 닫기를 선택합니다.
- 브랜치를 병합하고 풀 요청을 종료하려면 병합을 선택합니다. 코드에 사용 가능한 병합 전략 중에서 선택할 수 있습니다. 병합 전략은 소스 브랜치와 대상 브랜치 간 차이 및 병합이 완료된 후 소스 브랜치를 자동으로 삭제할지 여부에 따라 달라집니다. 선택한 후 풀 요청 병합을 선택하여 병합을 완료합니다.



- 브랜치에서 자동으로 해결할 수 없는 병합 충돌이 발생한 경우, CodeCommit 콘솔에서 이러한 충돌을 해결하거나 로컬 Git 클라이언트를 사용하여 브랜치를 병합한 다음 병합을 푸시할 수 있

습니다. 자세한 내용은 AWS CodeCommit 리포지토리에서 풀 요청의 충돌 해결 단원을 참조하십시오.

> **Note**
> 언제라도 로컬 리포지토리에서 git merge 명령을 사용하고 변경 내용을 푸시하여 브랜치(풀 요청 브랜치 포함)를 수동으로 병합할 수 있습니다.

자세한 내용은 풀 요청 작업 및 승인 규칙 템플릿 작업 단원을 참조하세요.

5단계: 정리

CodeCommit 리포지토리가 더 이상 필요하지 않을 경우에는 스토리지 공간에 대한 비용이 계속 부과되지 않도록 본 연습에서 사용한 CodeCommit 리포지토리와 기타 리소스를 삭제하는 것이 좋습니다.

> **Important**
> 이 작업은 실행을 취소할 수 없습니다. 이 리포지토리를 삭제한 후에는 이 리포지토리를 로컬이나 공유 리포지토리에 더 이상 복제할 수 없습니다. 또 이 리포지토리를 대상으로 데이터를 풀하거나 푸시할 수 없으며 로컬 또는 공유 리포지토리에서 어떠한 Git 연산도 수행할 수 없습니다.
> 리포지토리에 대해 알림을 설정한 경우에는 리포지토리 삭제 시 해당 리포지토리에 대해 생성된 Amazon CloudWatch Events 규칙도 함께 삭제됩니다. 단, 해당 규칙의 대상으로 사용된 Amazon SNS 주제는 삭제되지 않습니다.
> 리포지토리에 트리거를 구성한 경우에는 해당 리포지토리를 삭제해도 트리거의 대상으로 설정한 Amazon SNS 주제나 Lambda 함수는 삭제되지 않습니다. 필요 없는 리소스는 반드시 삭제하세요. 자세한 내용은 리포지토리에서 트리거 삭제 단원을 참조하십시오.

CodeCommit 리포지토리를 삭제하려면

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서 삭제할 리포지토리를 선택합니다. 본 주제의 이름 지정 규칙을 따른 경우 해당 리포지토리는 MyDemoRepo로 명명됩니다.
3. 탐색 창에서 설정을 선택합니다.
4. 설정 페이지의 리포지토리 삭제에서 리포지토리 삭제를 선택합니다.

5. **delete**를 입력한 다음 삭제를 선택합니다. 이로써 리포지토리가 영구적으로 삭제됩니다.

6단계: 다음 단계

CodeCommit과 그 일부 기능에 익숙해졌으니 이제 다음의 사항들을 고려해 보세요.

- Git과 CodeCommit을 처음 접하거나 CodeCommit으로 Git을 활용한 사례를 검토하고 싶다면, 계속해서 Git 및 CodeCommit 시작하기 자습서를 참조하세요.
- CodeCommit 리포지토리에 다른 사람과 작업을 수행하고 싶다면, 리포지토리 공유 섹션을 참조하세요. (다른 Amazon Web Services 계정의 사용자와 자신의 리포지토리를 공유하고 싶다면 역할을 사용하여 AWS CodeCommit 리포지토리에 대한 교차 계정 액세스 구성 단원을 참조하세요.)
- 리포지토리를 CodeCommit으로 마이그레이션하려면 CodeCommit으로 마이그레이션 섹션에 제시된 단계를 따릅니다.
- 지속적인 배포 파이프라인에 리포지토리를 추가하려면 간단한 파이프라인 안내 자습서의 절차를 따릅니다.
- 커뮤니티의 사례를 포함하여 CodeCommit과 통합되는 제품 및 서비스에 대해 자세히 알아보려면 제품 및 서비스 통합 섹션을 참조하세요.

Git 및 시작하기 AWS CodeCommit

Git과 CodeCommit을 처음 사용한다면 이 자습서에서 몇 가지 간단한 명령을 배워 시작할 수 있습니다. 이미 Git에 익숙한 경우 이 자습서를 건너뛰고 CodeCommit 시작하기 로 이동할 수 있습니다.

이 자습서에서는 CodeCommit 리포지토리의 로컬 사본에 해당하는 리포지토리를 생성합니다. 이를 로컬 지포지토리라고 부릅니다.

먼저 로컬 리포지토리를 만든 후에 몇 가지를 변경합니다. 그런 다음 변경 사항을 CodeCommit 리포지토리로 전송(푸시)합니다.

또한 두 사용자가 독립적으로 변경 사항을 로컬 리포지토리에 커밋하고 CodeCommit 리포지토리에 푸시하는 팀 환경을 시뮬레이션합니다. 그런 다음 각 사용자는 다른 사용자의 변경 사항을 보기 위해 해당 변경 사항을 CodeCommit 리포지토리에서 로컬 리포지토리로 풀합니다.

또한 CodeCommit 리포지토리에서 브랜치와 태그를 생성하고 몇 가지 액세스 권한도 관리합니다.

이 자습서를 완료한 후에는 Git과 CodeCommit에 관한 핵심 개념을 자신의 프로젝트에서 활용할 수 있도록 충분한 연습을 해야 합니다.

다음을 포함하여 사전 필수 및 설정을 완료합니다.

- IAM 사용자에게 권한을 할당합니다.
- CodeCommit이 임의의 리포지토리에 연결되도록 설정합니다. 이때 HTTPS나 SSH 또는 git-remote-codecommit을 활용합니다. 어떤 것을 활용할지에 대해서는 에 대한 설정 AWS CodeCommit 단원을 참조하세요.
- 리포지토리 생성을 포함한 모든 작업에 명령줄 또는 터미널을 AWS CLI 사용하려면를 구성합니다.

주제

- 1단계: CodeCommit 리포지토리 생성
- 2단계: 로컬 리포지토리 생성
- 3단계: 첫 커밋 생성
- 4단계: 첫 커밋 푸시
- 5단계: CodeCommit 리포지토리를 공유하고 또 다른 커밋을 푸시 및 풀하기
- 6단계: 브랜치 생성 및 공유
- 7단계: 태그 생성 및 공유
- 8단계: 액세스 권한 설정
- 9단계: 정리

1단계: CodeCommit 리포지토리 생성

이 단계에서는 CodeCommit 콘솔을 사용하여 리포지토리를 생성합니다.

사용할 CodeCommit 리포지토리가 이미 있으면 이 단계를 건너뛸 수 있습니다.

> **Note**
> 사용량에 따라, 리포지토리를 생성하거나 액세스하는 것에 대한 비용이 부과될 수 있습니다. 자세한 내용은 CodeCommit 제품 정보 페이지에서 요금을 참조하세요.

CodeCommit 리포지토리를 생성하려면

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.

2. 리전 선택기를 사용하여 리포지토리를 생성할 AWS 리전 를 선택합니다. 자세한 내용은 리전 및 Git 연결 엔드포인트 단원을 참조하십시오.
3. 리포지토리 페이지에서 리포지토리 생성을 선택합니다.
4. 리포지토리 생성 페이지에서 리포지토리 이름에 해당 리포지토리의 이름(예: **MyDemoRepo**)을 입력합니다.

> **Note**
>
> 리포지토리 이름은 대/소문자를 구분하며 100자를 초과할 수 없습니다. 자세한 내용은 제한을 참조하세요.

5. (선택 사항) 설명 란에 설명 내용을 입력합니다(예: **My demonstration repository**). 그러면 사용자들이 리포지토리의 용도를 식별하는 데 도움이 됩니다.
6. (선택 사항) 태그 추가를 선택하여 하나 이상의 리포지토리 태그(AWS 리소스를 구성하고 관리하는 데 도움이 되는 사용자 지정 속성 레이블)를 리포지토리에 추가합니다. 자세한 내용은 에서 리포지토리 태그 지정 AWS CodeCommit 단원을 참조하십시오.
7. (선택 사항) 추가 구성을 확장하여이 리포지토리의 데이터를 암호화하고 복호화하기 위해 기본 AWS 관리형 키 또는 자체 고객 관리형 키를 사용할지 여부를 지정합니다. 자체 고객 관리형 키를 사용하기로 선택한 경우 리포지토리를 생성하는 AWS 리전 에서 사용할 수 있고 키가 활성 상태인지 확인해야 합니다. 자세한 내용은 AWS Key Management Service 및 AWS CodeCommit 리포지토리 암호화 단원을 참조하십시오.
8. (선택 사항) 이 리포지토리에 Java나 Python 코드가 포함될 예정이고 해당 코드를 CodeGuru Reviewer로 분석하고 싶다면, Java 또는 Python용 Amazon CodeGuru Reviewer 활성화를 선택합니다. CodeGuru Reviewer는 다양한 기계 학습 모델을 사용하여 코드의 결함을 찾아내고 풀 요청의 개선점과 해결책을 자동으로 제안합니다. 자세한 내용은 Amazon CodeGuru Reviewer 사용 설명서를 참조하세요.
9. 생성(Create)을 선택합니다.

> **Note**
>
> 이 자습서의 나머지 단계에서는 CodeCommit 리포지토리의 이름으로 `MyDemoRepo`를 사용합니다. 다른 이름을 선택하는 경우 이 자습서 전체에서 이를 사용해야 합니다.

터미널 또는 명령줄에서 리포지토리를 생성하는 방법을 포함하여 리포지토리를 생성하는 데 대한 자세한 내용은 리포지토리 생성를 참조하세요.

2단계: 로컬 리포지토리 생성

이 단계에서는 로컬 컴퓨터에서 로컬 리포지토리를 설정하여 리포지토리에 연결합니다. 이렇게 하려면 로컬 리포지토리를 나타내는 로컬 컴퓨터에서 디렉터리를 선택합니다. Git을 활용해 해당 디렉터리 내에 빈 CodeCommit 리포지토리의 사본을 복제하고 초기화합니다. 그런 다음 커밋에 주석을 다는 데 사용하는 Git 사용자명과 이메일 주소를 지정합니다.

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리전 선택기에서 리포지토리가 생성된 AWS 리전 를 선택합니다. 리포지토리는에 고유합니다 AWS 리전. 자세한 내용은 리전 및 Git 연결 엔드포인트 단원을 참조하십시오.
3. 목록에서 연결하려는 리포지토리를 찾아서 선택합니다. URL 복제를 선택한 다음 리포지토리를 복제하거나 연결할 때 사용할 프로토콜을 선택합니다. 이것으로 복제 URL가 복사됩니다.

   - IAM 사용자를 통한 Git 보안 인증 정보를 활용하거나 AWS CLI에 포함된 보안 인증 도우미를 사용하는 경우 HTTPS URL을 복사합니다.
   - 로컬 컴퓨터에서 git-remote-codecommit 명령을 사용하는 경우 HTTPS(GRC) URL을 복사합니다.
   - IAM 사용자와 SSH 퍼블릭/프라이빗 키 페어를 사용하는 경우 SSH URL을 복사합니다.

   > **Note**
   >
   > 리포지토리 목록 대신 시작 페이지가 표시되면 로그인한에 AWS 계정과 연결된 AWS 리전 리포지토리가 없는 것입니다. 리포지토리를 만들려면 the section called “리포지토리 생성”을(를) 참조하거나 Git 및 CodeCommit 시작하기 자습서의 다음 단계를 따르세요.

4. (선택 사항) 로컬 Git 클라이언트는 리포지토리의 기본 브랜치에 **main**이라는 이름을 쓰도록 구성하는 것이 좋습니다. 이 이름은 본 가이드에서 모든 예제의 기본 브랜치에 사용됩니다. 또한 콘솔에서 첫 커밋을 만들 때 CodeCommit이 사용하는 기본 브랜치에도 똑같이 쓰입니다. 다음 명령을 실행하여 시스템의 기본 브랜치 이름을 전역으로 구성합니다.

   ```
   git config --global init.defaultBranch main
   ```

   만약 모든 리포지토리의 기본 브랜치 이름을 전부 다르게 사용하고 싶다면 **main**을 원하는 이름으로 변경합니다. 본 자습서에서는 기본 브랜치의 이름을 main으로 가정합니다.

   리포지토리별로 기본 브랜치 이름을 다르게 사용하려면 이 속성을 글로벌(--global)이 아닌 로컬(--local)로 설정하면 됩니다.

5. 터미널 또는 명령 프롬프트에서, git clone 명령으로 리포지토리를 복제하고 3단계에서 복사한 복제 URL을 입력합니다. 복제 URL은 사용하는 프로토콜 및 구성에 따라 결정됩니다. 예를 들어, 미국 동부(오하이오) 리전에서 Git 보안 인증 정보와 함께 HTTPS를 사용하여 *MyDemorepo*라는 리포지토리를 복제하려는 경우는 다음과 같습니다.

```
git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-
demo-repo
```

git-remote-codecommit와 함께 HTTPS를 사용하는 경우는 다음과 같습니다.

```
git clone codecommit://MyDemoRepo my-demo-repo
```

SSH를 사용 중인 경우는 다음과 같습니다.

```
git clone ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-demo-
repo
```

> **Note**
>
> 리포지토리를 복제하려고 할 때 오류가 표시되면 로컬 컴퓨터에 필요한 설정을 완료하지 않았을 수 있습니다. 자세한 내용은 에 대한 설정 AWS CodeCommit 단원을 참조하십시오.

3단계: 첫 커밋 생성

이 단계에서는 로컬 리포지토리에 첫 커밋을 생성합니다. 이를 위해 로컬 리포지토리에 예제 파일을 2개 생성합니다. Git을 사용하여 로컬 리포지토리에서 변경 사항을 준비하고, 로컬 리포지토리에 해당 변경 사항을 커밋합니다.

1. 텍스트 편집기를 사용하여 디렉터리에 다음과 같은 예제 텍스트 파일을 2개 생성합니다. `cat.txt` 및 `dog.txt` 파일의 이름 지정:

```
cat.txt
-------
The domestic cat (Felis catus or Felis silvestris catus) is a small, usually furry,
 domesticated, and carnivorous mammal.
```

```
dog.txt
-------
The domestic dog (Canis lupus familiaris) is a canid that is known as man's best
 friend.
```

2. git config를 실행하여 자리 표시자 *your-user-name* 및 *your-email-address*로 나타내는 사용자 이름 및 이메일 주소를 로컬 리포지토리에 추가합니다. 이렇게 하면 커밋을 쉽게 식별할 수 있습니다.

```
git config --local user.name "your-user-name"
git config --local user.email your-email-address
```

3. 로컬 리포지토리를 생성할 때 기본 브랜치 이름을 전역으로 설정하지 않은 경우, 다음 명령을 실행하여 기본 브랜치 이름을 **main**으로 설정합니다.

```
git config --local init.defaultBranch main
```

4. git add를 실행하여 변경 내용을 준비합니다.

```
git add cat.txt dog.txt
```

5. git commit를 실행하여 변경 내용을 커밋합니다.

```
git commit -m "Added cat.txt and dog.txt"
```

> **Tip**
> 생성한 커밋의 세부 정보를 보려면 git log를 실행합니다.

4단계: 첫 커밋 푸시

이 단계에서는 커밋을 로컬 리포지토리에서 CodeCommit 리포지토리로 푸시합니다.

git push를 실행하여, Git이 CodeCommit 리포지토리에 사용하는 기본 원격 이름(`origin`)을 통해 로컬 리포지토리의 기본 브랜치(`main`)에서 커밋을 푸시합니다.

```
git push -u origin main
```

> **Tip**
>
> CodeCommit 리포지토리에 파일을 푸시하고 나면, CodeCommit 콘솔을 사용하여 해당 콘텐츠를 볼 수 있습니다. 자세한 내용은 리포지토리에서 파일 검색 단원을 참조하십시오.

5단계: CodeCommit 리포지토리를 공유하고 또 다른 커밋을 푸시 및 풀하기

이 단계에서는 팀원 한 명과 함께 CodeCommit 리포지토리에 대한 정보를 공유합니다. 팀원은 이 정보를 사용하여 로컬 사본을 생성하고 몇 가지 사항을 변경하며 수정된 로컬 사본을 CodeCommit 리포지토리에 푸시합니다. 그런 다음 CodeCommit 리포지토리에서 로컬 리포지토리로 변경 사항을 풀합니다.

이 자습서에서는 2단계에서 생성한 것과는 별개인 디렉터리를 생성하는 Git로 사용자 따르기를 시뮬레이션합니다. (일반적으로 이 디렉터리는 다른 시스템에 있습니다.) 이 새 디렉터리는 CodeCommit 리포지토리의 복사본입니다. 기존 디렉터리나 이 새 디렉터리에 대한 변경 사항은 모두 독립적으로 이루어집니다. 이 디렉터리에 대한 변경 사항을 식별하는 유일한 방법은 CodeCommit 리포지토리에서 풀하는 것입니다.

동일한 로컬 시스템에 있다고 해도 기존 디렉터리 local repo 및 새 디렉터리 shared repo를 호출합니다.

새 디렉터리에서 CodeCommit 리포지토리의 사본을 별도로 하나 만듭니다. 그런 다음 새 예제 파일을 추가하고, 공유 리포지토리에 변경 내용을 커밋하며, 공유 리포지토리의 커밋을 CodeCommit 리포지토리로 푸시합니다.

마지막으로 리포지토리의 변경 내용을 로컬 리포지토리로 풀하고 다른 사용자가 커밋한 변경 내용을 볼 수 있도록 살펴봅니다.

1. `/tmp` 디렉터리 또는 `c:\temp` 디렉터리로 전환합니다.
2. git clone을 실행하여 리포지토리 사본을 공유 리포지토리로 폴다운합니다.

   HTTPS의 경우에는 다음과 같이 합니다.

   ```
   git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
    shared-demo-repo
   ```

   git-remote-codecommit를 사용하는 HTTPS의 경우:

```
git clone codecommit://MyDemoRepo shared-demo-repo
```

SSH의 경우에는 다음과 같이 합니다.

```
git clone ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo shared-
demo-repo
```

> **Note**
>
> Windows 운영 체제에서 SSH를 사용하여 리포지토리를 복제할 경우 다음과 같이 SSH 키 ID를 연결 문자열에 추가해야 할 수 있습니다.
>
> ```
> git clone ssh://Your-SSH-Key-ID@git-codecommit.us-east-2.amazonaws.com/v1/
> repos/MyDemoRepo my-demo-repo
> ```
>
> 자세한 내용은 Windows에서 SSH 연결 단원을 참조하십시오.

이 명령에서 `MyDemoRepo`는 CodeCommit 리포지토리의 이름입니다. `shared-demo-repo`는 `/tmp` 디렉터리 또는 `c:\temp` 디렉터리에서 Git이 생성하는 디렉터리의 이름입니다. Git에서 디렉터리가 생성된 후에는 리포지토리의 복사본이 `shared-demo-repo` 디렉터리로 풀다운됩니다.

3. `shared-demo-repo` 디렉터리로 전환합니다.

```
(For Linux, macOS, or Unix) cd /tmp/shared-demo-repo
(For Windows) cd c:\temp\shared-demo-repo
```

4. git config를 실행하여 자리 표시자 *`other-user-name`* 및 *`other-email-address`*로 나타내는 다른 사용자 이름 및 이메일 주소를 추가합니다. 이렇게 하면 다른 사용자의 커밋을 쉽게 식별할 수 있습니다.

```
git config --local user.name "other-user-name"
git config --local user.email other-email-address
```

5. 텍스트 편집기를 사용하여 다음 예제 텍스트 파일을 `shared-demo-repo`에서 생성합니다. `horse.txt` 파일의 이름을 지정합니다.

```
horse.txt
```

```
-------
The horse (Equus ferus caballus) is one of two extant subspecies of Equus ferus.
```

6. git add를 실행하여 공유 리포지토리로 변경 내용을 준비합니다.

```
git add horse.txt
```

7. git commit를 실행하여 공유 리포지토리로 변경 내용을 커밋합니다.

```
git commit -m "Added horse.txt"
```

8. git push를 실행하여, Git이 CodeCommit 리포지토리에 사용하는 기본 원격 이름(`origin`)을 통해 로컬 리포지토리의 기본 브랜치(`main`)에서 초기 커밋을 푸시합니다.

```
git push -u origin main
```

9. 로컬 리포지토리로 전환하고 git pull을 실행하여, 공유 리포지토리가 CodeCommit 리포지토리에 만든 커밋을 로컬 리포지토리로 풀합니다. 그런 다음 git log를 실행하여 공유 리포지토리에서 초기화된 커밋을 확인합니다.

6단계: 브랜치 생성 및 공유

이 단계에서는 로컬 리포지토리에서 브랜치를 생성하고, 몇 가지 사항을 변경한 다음, 이 브랜치를 CodeCommit 리포지토리로 푸시합니다. 그런 다음 CodeCommit 리포지토리에서 공유 리포지토리에 브랜치를 풀합니다.

브랜치를 사용하면 서로 다른 버전의 리포지토리 콘텐츠를 독립적으로 개발할 수 있습니다(예를 들어, 팀원 작업에 영향을 주지 않고 새 소프트웨어 기능으로 작업할 수 있음). 이 기능이 안정화되면 브랜치를 더욱 안정화된 소프트웨어 브랜치로 병합합니다.

Git을 사용하여 브랜치를 생성하고 만들어 놓은 첫 커밋을 가리키도록 합니다. Git을 사용하여 브랜치를 CodeCommit 리포지토리로 푸시합니다. 그런 다음 공유 리포지토리로 전환하고 Git을 사용하여 새 브랜치를 공유 로컬 리포지토리로 풀하고 해당 브랜치를 탐색합니다.

1. 로컬 리포지토리에서 git checkout을 실행하고, 브랜치의 이름(예: `MyNewBranch`)과 로컬 리포지토리에서 만든 첫 커밋의 ID를 지정합니다.

   커밋 ID를 모르면 git log를 실행하여 확인합니다. 커밋에 다른 사용자의 사용자 이름과 이메일 주소가 아닌 본인의 사용자 이름과 이메일 주소가 있는지 확인합니다. 이렇게 하여 `main`이

CodeCommit 리포지토리의 안정화 버전인지, 그리고 `MyNewBranch` 브랜치가 새롭지만 상대적으로 안정화되지 않은 기능인지 시뮬레이션할 수 있습니다.

```
git checkout -b MyNewBranch commit-ID
```

2. git push를 실행하여 로컬 리포지토리에서 CodeCommit 리포지토리로 새 브랜치를 전송합니다.

```
git push origin MyNewBranch
```

3. 이제 브랜치를 공유 리포지토리로 풀하고 결과를 확인합니다.

   1. 공유 리포지토리 디렉터리(shared-demo-repo)로 전환합니다.
   2. 새 브랜치(git fetch origin)를 풀합니다.
   3. 새 브랜치(git branch --all은 리포지토리에 대한 모든 브랜치 목록을 표시함)를 풀했는지 확인합니다.
   4. 새로운 브랜치(git checkout MyNewBranch)로 전환합니다.
   5. git status 또는 git branch를 실행하여 `MyNewBranch` 브랜치로 전환했는지 확인합니다. 현재 위치한 브랜치가 결과에 표시됩니다. 이 경우 `MyNewBranch`가 되어야 합니다.
   6. 브랜치(git log)에서 커밋 목록을 확인합니다.

   호출할 Git 명령 목록은 다음과 같습니다.

```
git fetch origin
git branch --all
git checkout MyNewBranch
git branch or git status
git log
```

4. `main` 브랜치로 다시 전환하여 커밋 목록을 확인합니다. Git 명령은 다음과 같은 형식이어야 합니다.

```
git checkout main
git log
```

5. 로컬 리포지토리에서 `main` 브랜치로 전환합니다. git status 또는 git branch를 실행할 수 있습니다. 현재 위치한 브랜치가 결과에 표시됩니다. 이 경우 `main`가 되어야 합니다. Git 명령은 다음과 같은 형식이어야 합니다.

```
git checkout main
git branch or git status
```

7단계: 태그 생성 및 공유

이 단계에서는 로컬 리포지토리에 두 개의 태그를 생성해 커밋과 연결한 다음 CodeCommit 리포지토리로 푸시합니다. 그런 다음 CodeCommit 리포지토리에서 공유 리포지토리로 변경 사항을 풀합니다.

태그는 커밋(또는 분기나 다른 태그)에 사람이 읽을 수 있는 이름을 제공하는 데 사용됩니다. 예를 들어 커밋에 `v2.1`과 같은 태그를 지정하려면 이렇게 할 수 있습니다. 커밋, 브랜치 또는 태그에 연결할 수 있는 태그 수는 제한이 없지만 개별 태그는 하나의 커밋, 브랜치 또는 태그에만 연결할 수 있습니다. 이 자습서에서는 한 커밋에는 `release`라는 태그를 지정하고, 다른 커밋에는 `beta`라는 태그를 지정합니다.

Git을 사용하여 태그를 생성하고 `release` 태그로 만들었던 첫 커밋을 가리키며, `beta` 태그로 다른 사용자가 만든 커밋을 가리킵니다. 다음으로 Git을 사용하여 CodeCommit 리포지토리로 태그를 푸시합니다. 다음으로 공유 리포지토리로 전환하고, Git을 사용하여 공유 로컬 리포지토리로 태그를 풀한 다음, 해당 태그를 탐색합니다.

1. 로컬 리포지토리에서 git tag를 실행하고, 새 태그의 이름(`release`)과 로컬 리포지토리에서 만든 첫 커밋의 ID를 지정합니다.

   커밋 ID를 모르면 git log를 실행하여 확인합니다. 커밋에 다른 사용자의 사용자 이름과 이메일 주소가 아닌 본인의 사용자 이름과 이메일 주소가 있는지 확인합니다. 이렇게 하여 커밋이 CodeCommit 리포지토리의 안정화 버전인지 시뮬레이션할 수 있습니다.

   ```
   git tag release commit-ID
   ```

   git tag를 다시 실행하여 `beta` 태그로 다른 사용자의 커밋에 태그를 지정합니다. 이렇게 커밋이 새롭지만 상대적으로 안정화되지 않은 기능인지 시뮬레이션합니다.

   ```
   git tag beta commit-ID
   ```

2. git push --tags를 실행하여 CodeCommit 리포지토리로 태그를 전송합니다.
3. 이제 태그를 공유 리포지토리로 풀하고 결과를 확인합니다.

1. 공유 리포지토리 디렉터리(shared-demo-repo)로 전환합니다.
2. 새 태그(git fetch origin)에서 풀합니다.
3. 새 태그가 (git tag은 리포지토리에 대한 모든 태그 목록을 표시함)에서 풀했는지 확인합니다.
4. 각 태그(git log release 및 git log beta)에 대한 정보를 확인합니다.

호출할 Git 명령 목록은 다음과 같습니다.

```
git fetch origin
git tag
git log release
git log beta
```

4. 이 과정을 로컬 리포지토리에서도 시도합니다.

```
git log release
git log beta
```

8단계: 액세스 권한 설정

이 단계에서는 공유 리포지토리를 CodeCommit 리포지토리와 동기화하기 위해 사용자 권한을 제공합니다. 이 단계는 선택 사항입니다. CodeCommit 리포지토리에 대한 액세스를 제어하는 방법을 배우려는 사용자에게 권장합니다. 사용자가 Git 보안 인증 정보를 사용하거나 IAM 사용자가 CodeCommit 리포지토리에 액세스하기 위해 SSH 키 페어를 사용할 때 수행하는 것이 좋습니다.

> **Note**
> 페더레이션 액세스, 임시 보안 인증 정보, 또는 IAM Identity Center 같은 웹 자격 증명 공급자를 사용하는 경우에는 ID 공급자의 사용자, 액세스. 권한 등을 설정한 다음 git-remote-codecommit을 사용합니다. 자세한 정보는 git-remote-codecommit을 AWS CodeCommit 사용하여에 HTTPS 연결을 위한 설정 단계 및 교체 자격 증명을 사용하여 AWS CodeCommit 리포지토리에 연결 섹션을 참조하세요.

이렇게 하려면 IAM 콘솔을 사용하여 사용자를 생성합니다. 기본적으로 이 사용자에게는 공유 리포지토리를 CodeCommit 리포지토리와 동기화할 수 있는 권한이 없습니다. git pull을 실행하여 이를 확인할 수 있습니다. 새 사용자에게 동기화할 권한이 없는 경우 명령이 실행되지 않습니다. 그러면 IAM 콘

솔로 돌아와 사용자가 git pull을 사용할 수 있도록 허용하는 정책을 적용합니다. 다시 git pull을 실행하여 이를 확인할 수 있습니다.

이 단계는 Amazon Web Services 계정에서 IAM 사용자를 생성할 수 있는 권한이 있다는 가정 하에 작성되었습니다. 이 권한이 없으면 이 단계의 절차를 수행할 수 없습니다. 자습서에 사용한 리소스를 정리하려면 9단계: 정리으로 건너뜁니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/://https://https://https://https://://https://://https://://https://://https://://https://://https://://

   설정 에서 사용한 동일한 사용자 이름과 암호로 로그인했는지 확인합니다.
2. 탐색 창에서 사용자와 새 사용자 만들기를 차례대로 선택합니다.
3. 첫 번째 사용자 이름 입력 상자에서 예제 사용자 이름(예: **JaneDoe-CodeCommit**)을 입력합니다. 각 사용자의 액세스 키 생성 상자를 선택한 후 생성을 선택합니다.
4. 사용자 보안 인증 정보 표시를 선택합니다. 액세스 키 ID 및 비밀 액세스 키를 기록하거나 보안 인증 정보 다운로드를 선택합니다.
5. Git 보안 인증 정보를 사용하는 HTTPS 사용자의 경우의 지침에 따라 IAM 사용자의 보안 인증 정보를 생성하고 제공합니다.

   SSH를 사용하려면 SSH와 Linux, macOS Unix: Git과 CodeCommit에 사용되는 퍼블릭 키와 프라이빗 키를 설정합니다. 또는 3단계: Git과 CodeCommit에 사용되는 퍼블릭 키와 프라이빗 키 설정의 지침을 따라 퍼블릭 및 프라이빗 키로 사용자를 설정합니다.
6. git pull을(를) 실행합니다. 다음 오류가 나타납니다.

   HTTPS의 경우에는 다음과 같이 합니다.

   ```
   fatal: unable to access 'https://git-codecommit.us-
   east-2.amazonaws.com/v1/repos/repository-name/': The requested URL
   returned error: 403.
   ```

   SSH의 경우에는 다음과 같이 합니다.

   ```
   fatal: unable to access 'ssh://git-codecommit.us-east-2.amazonaws.com/
   v1/repos/repository-name/': The requested URL returned error: 403.
   ```

   새 사용자에게 공유 리포지토리를 CodeCommit 리포지토리와 동기화할 권한이 없어 이런 오류가 발생하는 것입니다.

7. IAM 콘솔로 돌아갑니다. 탐색 창에서 정책을 선택한 후 정책 생성을 선택합니다. (시작하기 버튼이 표시되면 이 버튼을 선택한 후 정책 생성을 선택합니다.)
8. 자체 정책 생성 옆의 선택을 선택합니다.
9. 정책 이름 상자에 이름을 입력합니다(예: **CodeCommitAccess-GettingStarted**).
10. IAM 사용자가 자신과 연결된 리포지토리에서 풀할 수 있도록 정책 문서 상자에서 다음을 입력합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecommit:GitPull"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Tip**
>
> IAM 사용자가 자신과 연결된 리포지토리로 커밋을 푸시할 수 있도록 하려면 다음을 대신 입력합니다.
>
> ```
> {
>  "Version": "2012-10-17",
>  "Statement": [
>  {
>  "Effect": "Allow",
>  "Action": [
>  "codecommit:GitPull",
>  "codecommit:GitPush"
>  ],
>  "Resource": "*"
>  }
>  ]
> }
> ```

다른 CodeCommit 작업과, 사용자에게 제공할 수 있는 리소스 권한에 대해 자세히 알아보려면 AWS CodeCommit에 대한 인증 및 액세스 제어 섹션을 참조하세요.

11. 탐색 창에서 사용자를 선택합니다.
12. 정책에 첨부할 예제 사용자 이름(예: **`JaneDoe-CodeCommit`**)을 선택합니다.
13. 권한 탭을 선택합니다.
14. 관리형 정책에서 정책 연결을 선택합니다.
15. 방금 생성한 **`CodeCommitAccess-GettingStarted`** 정책을 선택한 후 정책 연결을 선택합니다.
16. git pull을(를) 실행합니다. 이제 명령이 실행되면서 `Already up-to-date` 메시지가 표시되어야 합니다.
17. HTTPS를 사용하는 경우 원래 Git 자격 증명으로 전환하거나 git-remote-codecommit를 사용하는 경우 일반 프로파일로 전환합니다. 자세한 내용은 Git 보안 인증 정보를 사용하는 HTTPS 사용자를 위한 설정 단원 또는 git-remote-codecommit을 AWS CodeCommit 사용하여에 HTTPS 연결을 위한 설정 단계 단원을 참조하세요.

    SSH를 사용하는 경우 원래 키로 전환합니다. 자세한 내용은 SSH와 Linux, macOS Unix: Git과 CodeCommit에 사용되는 퍼블릭 키와 프라이빗 키를 설정합니다. 또는 3단계: Git과 CodeCommit에 사용되는 퍼블릭 키와 프라이빗 키 설정을 참조하세요.

본 자습서를 이수하셨습니다.

9단계: 정리

이 단계에서는 이 자습서에서 사용한 CodeCommit 리포지토리를 삭제하여 스토리지 공간에 대해 요금이 청구되지 않도록 합니다.

CodeCommit 리포지토리를 삭제하면 로컬 리포지토리와 공유 리포지토리는 더 이상 필요하지 않으므로 로컬 컴퓨터에서 제거해도 됩니다.

> **Important**
>
> 이 리포지토리를 삭제하면 이 리포지토리를 로컬 또는 공유 리포지토리에 복제할 수 없게 됩니다. 또한 로컬 또는 공유 리포지토리를 대상으로 데이터를 풀하거나 푸시할 수 없습니다. 이 작업은 실행을 취소할 수 없습니다.

CodeCommit 리포지토리를 삭제하려면 (콘솔)

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 대시보드 페이지에서 리포지토리 목록 중 MyDemoRepo를 선택합니다.
3. 탐색 창에서 설정을 선택합니다.
4. 설정 페이지의 리포지토리 삭제에서 리포지토리 삭제를 선택합니다.
5. 리포지토리의 이름을 입력하여 삭제를 확인 옆의 상자에 **MyDemoRepo**를 입력한 다음 삭제를 선택합니다.

CodeCommit 리포지토리를 삭제하려면 (AWS CLI)

리포지토리 삭제 명령을 실행합니다.

```
aws codecommit delete-repository --repository-name MyDemoRepo
```

로컬 및 공유 리포지토리를 삭제하려면

Linux, macOS, Unix의 경우:

```
cd /tmp
rm -rf /tmp/my-demo-repo
rm -rf /tmp/shared-demo-repo
```

Windows의 경우:

```
cd c:\temp
rd /s /q c:\temp\my-demo-repo
rd /s /q c:\temp\shared-demo-repo
```

와의 제품 및 서비스 통합 AWS CodeCommit

기본적으로 CodeCommit은 여러 AWS 서비스와 통합됩니다. CodeCommit을 외부의 제품 및 서비스와 함께 사용할 수도 있습니다 AWS. 다음 정보를 활용하면 사용 중인 제품과 서비스를 통합할 CodeCommit을 구성할 수 있습니다.

주제

- 다른 AWS 서비스와의 통합
- 커뮤니티의 통합 예제

다른 AWS 서비스와의 통합

CodeCommit은 다음 AWS 서비스와 통합됩니다.

AWS Amplify	AWS Amplify를 사용하면 AWS로 구동되는 확장 가능 모바일 애플리케이션을 손쉽게 생성, 구성, 구현할 수 있습니다. Amplify는 모바일 백엔드를 원활하게 프로비저닝하고 관리하며, 백엔드를 iOS, Android, 웹 및 React Native 프런트엔드와 손쉽게 통합할 수 있는 간단한 프레임워크를 제공합니다. 또한, 프런트엔드와 백엔드 모두의 애플리케이션 릴리스 프로세스를 자동화하므로 기능을 더 빠르게 제공할 수 있습니다. Amplify 콘솔에서 CodeCommit 리포지토리를 연결할 수 있습니다. Amplify 콘솔에 권한을 부여하면 Amplify는 리포지토리 공급자로부터 액세스 토큰을 가져오지만 AWS 서버에 토큰을 저장하지 않습니다. Amplify는 특정 리포지토리에만 설치된 배포 키를 사용하여 리포지토리에 액세스합니다. 자세히 알아보기: • AWS Amplify 사용 설명서

	• 시작하기
AWS Cloud9	AWS Cloud9에는 클라우드에서 코드를 작성하고 소프트웨어를 빌드, 실행, 테스트, 디버그, 릴리스하는 데 사용할 수 있는 다양한 도구가 포함되어 있습니다. 이러한 도구 모음을 AWS Cloud9 통합 개발 환경 또는 IDE라고 합니다. 웹 브라우저를 통해 AWS Cloud9 IDE에 액세스합니다. IDE는 여러 프로그래밍 언어와 런타임 디버거 및 기본 제공 터미널을 갖춘 강력한 코드 편집 환경을 제공합니다. 자세히 알아보기: • AWS Cloud9 사용 설명서 • AWS CodeCommit 용 샘플 AWS Cloud9 • AWS Cloud9 와 통합 AWS CodeCommit
AWS CloudFormation	AWS CloudFormation는 AWS 리소스를 모델링하고 설정하는 데 도움이 되는 서비스로, 리소스를 관리하는 데 소요되는 시간을 줄이고 애플리케이션에 더 많은 시간을 할애할 수 있습니다. CodeCommit 리포지토리를 포함한 리소스를 설명하는 템플릿을 생성하면 AWS CloudFormation 가 해당 리소스를 프로비저닝하고 구성합니다. 자세히 알아보기: • AWS CodeCommit 리포지토리 리소스 페이지

AWS CloudTrail	CloudTrail은 Amazon Web Services 계정에 의해 또는 Amazon Web Services 계정을 대신하여 수행된 AWS API 호출 및 관련 이벤트를 캡처하고 사용자가 지정한 Amazon S3 버킷에 로그 파일을 전송합니다. AWS CodeCommit 콘솔, 의 CodeCommit 명령, 로컬 Git 클라이언트 및 CodeCommit API에서 API 호출 AWS CLI을 캡처하도록 CloudTrail을 구성할 수 있습니다. 자세히 알아보기: • 를 사용하여 AWS CodeCommit API 호출 로깅 AWS CloudTrail

Amazon CloudWatch Events	CloudWatch Events는 AWS 리소스의 변경 사항을 설명하는 시스템 이벤트의 스트림을 거의 실시간으로 제공합니다. 신속하게 설정할 수 있는 단순 규칙을 사용하여 일치하는 이벤트를 검색하고 하나 이상의 대상 함수 또는 스트림으로 이를 라우팅할 수 있습니다. CloudWatch Events는 운영 변경 사항이 발생할 때 이를 인식하게 됩니다. 또한 CloudWatch Events는 환경에 응답하기 위한 메시지를 전송하고 함수를 활성화하고 변경을 수행하고 상태 정보를 기록하는 등 이러한 운영 변경 사항에 응답하고 필요에 따라 조치를 취합니다. Amazon Simple Queue Service, Amazon Kinesis 등과 같은 다른 AWS 서비스의 스트림, 함수, 작업 또는 기타 프로세스를 대상으로 하여 CodeCommit 리포지토리 AWS Lambda를 모니터링하고 리포지토리 이벤트에 응답하도록 CloudWatch Events를 구성할 수 있습니다. 자세히 알아보기: • CloudWatch 이벤트 사용자 설명서 • AWS CodeCommit Events • 블로그 게시물: Amazon CloudWatch Events 및 JGit을 사용하여 서버리스 AWS CodeCommit 워크플로 빌드

AWS CodeBuild	CodeBuild는 소스 코드를 컴파일하고 유닛 테스트를 실행하며 배포 준비가 완료된 아티팩트를 생성하는 클라우드 상의 종합 관리형 빌드 서비스입니다. CodeCommit 리포지토리에 빌드할 소스 코드와 빌드 사양을 저장할 수 있습니다. CodeBuild를 CodeCommit으로 직접 사용하거나, CodePipeline을 통한 지속적 전달 파이프라인에 CodeBuild와 CodeCommit을 모두 통합할 수 있습니다. 자세히 알아보기: • 빌드 계획 • 빌드 프로젝트 생성 • CodePipeline을 AWS CodeBuild 와 함께 사용하여 빌드 실행
Amazon CodeGuru Reviewer	Amazon CodeGuru Reviewer는 프로그램 분석 및 기계 학습을 사용하여 Java 또는 Python 코드에서 일반적인 문제를 감지하고 권장 수정 사항을 제공하는 자동화된 코드 검토 서비스입니다. 사용자는 Amazon Web Services 계정의 리포지토리를 CodeGuru Reviewer와 연결할 수 있습니다. 이렇게 하면 CodeGuru Reviewer는 CodeGuru Reviewer가 연결 설정 후 생성된 모든 풀 요청에서 코드를 분석할 수 있도록 허용하는 서비스 연결 역할을 생성합니다. 자세히 알아보기: • AWS CodeCommit 리포지토리를 Amazon CodeGuru Reviewer와 연결 또는 연결 해제 • Amazon CodeGuru Reviewer 사용자 설명서

AWS CodePipeline	CodePipeline은 소프트웨어 릴리스에 필요한 단계를 모델링, 시각화 및 자동화하는 데 사용할 수 있는 지속적 전달 서비스입니다. 사용자는 파이프라인에서 CodeCommit 리포지토리를 소스 작업으로 사용하고 변경 사항의 빌드, 테스트, 배포를 자동화하도록 CodePipeline을 구성할 수 있습니다. 자세히 알아보기: • CodePipeline 및를 사용한 간단한 파이프라인 연습 AWS CodeCommit • CodeCommit 리포지토리가 있는 파이프라인에 대해 Amazon CloudWatch Events 변경 감지로 마이그레이션 • 파이프라인을 자동으로 시작하는 데 사용되는 변경 감지 방법
AWS Elastic Beanstalk	Elastic Beanstalk는 애플리케이션을 실행하는 인프라에 관계없이 AWS 클라우드에서 애플리케이션을 쉽게 배포하고 관리할 수 있는 관리형 서비스입니다. 사용자는 Elastic Beanstalk 명령줄 인터페이스(EB CLI)를 사용하여 새 CodeCommit 리포지토리나 기존의 CodeCommit 리포지토리에서 애플리케이션을 직접 배포할 수 있습니다. 자세히 알아보기: • AWS CodeCommit에서 EB CLI 사용 • 기존 AWS CodeCommit 리포지토리 사용 • eb codesource(EB CLI 명령)

AWS Key Management Service	AWS KMS는 데이터 암호화에 사용하는 암호화 키를 쉽게 생성하고 제어할 수 있게 해주는 관리형 서비스입니다. 기본적으로 CodeCommit은 AWS KMS 를 사용하여 리포지토리를 암호화합니다. 자세히 알아보기: • AWS KMS 및 암호화
AWS Lambda	Lambda를 사용하면 서버를 프로비저닝하거나 관리하지 않고 코드를 실행할 수 있습니다. 사용자는 리포지토리 이벤트에 응답하여 Lambda 함수를 간접 호출하는 트리거를 CodeCommit 리포지토리에 대해 구성할 수 있습니다. 자세히 알아보기: • Lambda 함수를 위한 트리거 생성 • AWS Lambda 개발자 안내서
Amazon Simple Notification Service	Amazon SNS는 애플리케이션, 최종 사용자 및 디바이스가 클라우드에서 알림을 즉시 전송하고 수신할 수 있게 해 주는 웹 서비스입니다. 리포지토리 이벤트에 응답하여 Amazon SNS 알림을 전송하는 트리거를 CodeCommit 리포지토리에 대해 구성할 수 있습니다. Amazon SNS 알림을 사용하여 다른 AWS 서비스와 통합할 수도 있습니다. 예를 들어 Amazon SNS 알림을 사용하여 Amazon Simple Queue Service 대기열에 메시지를 전송할 수 있습니다. 자세히 알아보기: • Amazon SNS 주제에 대한 트리거 생성 • Amazon Simple Notification Service 개발자 안내서

커뮤니티의 통합 예제

다음 단원에서는 블로그 포스트, 자료 및 커뮤니티에서 제공하는 예제를 제공합니다.

> **Note**
>
> 이러한 링크는 정보 제공 목적으로만 제공되며 예제의 콘텐츠에 대한 포괄적인 목록 또는 보증으로 간주해서는 안 됩니다. AWS 는 외부 콘텐츠의 콘텐츠 또는 정확성에 대해 책임을 지지 않습니다.

주제

- 블로그 게시물
- 코드 샘플

블로그 게시물

- AWS CodeCommit에서 풀 요청 승인자로 SonarQube 통합

  풀 요청을 병합하기 전에 성공적인 SonarQube 품질 분석이 필요한 CodeCommit 리포지토리를 생성하는 방법에 대해 알아봅니다.

  2019년 12월 12일 발행

- GitLab으로 AWS CodeCommitAWS CodePipeline, 그리고 GitLab AWS CodeBuild 에서 마이그레이션 GitLab

  AWS CodePipeline 및를 사용하여 GitLab AWS CodeCommit 에서 여러 리포지토리를 로 마이그레이션하고 CI/CD 파이프라인을 설정하는 방법을 알아봅니다 AWS CodeBuild.

  2019년 11월 22일 발행

- AWS CodePipeline, AWS CodeCommitAWS CodeBuild, 및를 사용하여 GitFlow 구현 AWS CodeDeploy

  AWS CodePipeline AWS CodeCommit, AWS CodeBuild및를 사용하여 GitFlow를 구현하는 방법을 알아봅니다 AWS CodeDeploy.

  2019년 2월 22일 발행

- 여러 AWS 계정 AWS CodeCommit 에서 Git 사용

  여러 Amazon Web Services 계정에서 Git 구성을 관리하는 방법을 알아봅니다.

  2019년 2월 12일 발행

- AWS CodeBuild및를 사용하여 AWS CodeCommit 풀 요청 검증 AWS Lambda

  AWS CodeCommit AWS CodeBuild및를 사용하여 풀 요청을 검증하는 방법을 알아봅니다 AWS Lambda. 기본 브랜치에 병합하기 전에 제안된 변경 사항에 대한 테스트를 실행하면 풀 요청에서 높은 수준의 품질을 보장하고, 잠재적인 문제를 포착하며, 변경 사항과 관련하여 개발자의 신뢰를 높일 수 있습니다.

  2019년 2월 11일 발행

- 에서 페더레이션 자격 증명 사용 AWS CodeCommit

  비즈니스에 사용되는 자격 증명을 AWS CodeCommit 사용하여에서 리포지토리에 액세스하는 방법을 알아봅니다.

  2018년 10월 5일 발행

- 에서 브랜치에 대한 액세스 권한 구체화 AWS CodeCommit

  컨텍스트 키를 사용하는 IAM 정책을 생성 및 적용함으로써 리포지토리 브랜치에 대한 커밋을 제한하는 방법을 알아봅니다.

  2018년 5월 16일 발행

- Fargate를 사용하여 AWS 리전 간 AWS CodeCommit 리포지토리 복제

  서버리스 아키텍처를 사용하여 다른 AWS 리전으로의 CodeCommit 리포지토리 연속 복제를 설정하는 방법에 대해 알아봅니다.

  2017년 4월 11일 발행

- AWS OpsWorks for Chef Automate 인프라 배포

  CodePipeline, CodeCommit, CodeBuild 및 AWS Lambda 를 사용하여 쿡북 및 기타 구성이 하나 이상의에 있는 둘 이상의 Chef 서버에 일관되게 배포되도록 하는 방법을 알아봅니다 AWS 리전.

  2018년 3월 9일 발행

- 땅콩버터와 초콜릿: AWS CodeCommit을 활용한 Azure Functions CI/CD 파이프라인

PowerShell 기반 Azure Functions CI/CD 파이프라인을 CodeCommit 리포지토리에 저장된 코드가 있는 장소에 생성하는 방법에 대해 알아봅니다.

2018년 19월 2일 발행

- AWS CodePipeline, AWS CodeCommit, AWS CodeBuild, Amazon ECR 및를 사용하여 Kubernetes에 지속적으로 배포 AWS Lambda

  Kubernetes 및를 AWS 함께 사용하여 컨테이너 기반 애플리케이션을 위한 완전 관리형 연속 배포 파이프라인을 생성하는 방법을 알아봅니다.

2018년 1월 11일 발행

- AWS CodeCommit 풀 요청을 사용하여 코드 검토 요청 및 코드 논의

  풀 요청을 사용하여 CodeCommit 리포지토리에서 코드 변경 사항을 리뷰하고 거기에 설명 추가 및 쌍방향 반복 처리하는 방법에 대해 알아봅니다.

2017년 11월 20일 발행

- Amazon CloudWatch Events 및 JGit을 사용하여 서버리스 AWS CodeCommit 워크플로 구축

  CodeCommit 리포지토리 이벤트 및 다른 AWS 서비스의 대상 작업을 사용하여 리포지토리의 변경 사항을 처리하는 CloudWatch Events 규칙을 생성하는 방법을 알아봅니다. 예제에는 커밋에 대한 Git 커밋 메시지 정책을 적용하고 CodeCommit 리포지토리를 복제하며 CodeCommit 리포지토리를 Amazon S3에 백업하는 AWS Lambda 함수가 포함되어 있습니다.

2017년 8월 3일 발행

- 로 마이그레이션 AWS CodeCommit

  원본 트리를 사용할 때 다른 Git 리포지토리에서 CodeCommit으로 마이그레이션하는 과정에서 두 개의 리포지토리에 코드를 푸시하는 방법에 대해 알아봅니다.

2016년 9월 6일 발행

- Appium, AWS CodeCommit Jenkins 및를 사용하여 연속 테스트 설정 AWS Device Farm

  Appium, CodeCommit, Jenkins, Device Farm을 사용하여 모바일 디바이스의 지속적 테스트 프로세스를 만드는 방법에 대해 알아봅니다.

2016년 2월 2일 발행

- 여러 Amazon Web Services 계정에서 Git 리포지토리와 AWS CodeCommit 함께 사용

CodeCommit 리포지토리를 복제하고 하나의 명령으로 보안 인증 도우미를 구성하여 해당 리포지토리와의 연결에 특정 IAM 역할을 사용하는 방법에 대해 알아봅니다.

2015년 11월 발행

- OpsWorks 및 통합 AWS CodeCommit

  OpsWorks 가 CodeCommit에서 앱 및 Chef 쿡북을 자동으로 가져오는 방법을 알아봅니다.

  2015년 8월 25일 발행

- AWS CodeCommit 과 GitHub 보안 인증 도우미 사용하기

  gitconfig 파일을 구성하여 CodeCommit 및 GitHub 보안 인증 도우미 모두로 작업하는 방법에 대해 알아봅니다.

  2015년 9월 발행

- Eclipse AWS CodeCommit 에서 사용

  Eclipse에서 EGit 도구를 사용하여 CodeCommit으로 작업하는 방법에 대해 알아봅니다.

  2015년 8월 발행

- AWS CodeCommit 과 Amazon EC2 보안 인증

  CodeCommit 리포지토리에 대한 자동화 에이전트 액세스를 구성할 때 Amazon EC2용 인스턴스 프로파일을 사용하는 방법에 대해 알아봅니다.

  2015년 7월 발행

- Jenkins AWS CodeCommit 와 통합

  CodeCommit과 Jenkins를 사용하여 두 개의 간편한 지속적 통합(CI) 시나리오를 지원하는 방법에 대해 알아봅니다.

  2015년 7월 발행

- 검토 위원회 AWS CodeCommit 와 통합

  Review Board 코드 리뷰 시스템을 활용하여 CodeCommit을 개발 워크플로에 통합하는 방법에 대해 알아봅니다.

  2015년 7월 발행

코드 샘플

다음은 CodeCommit 사용자가 관심을 둘 만한 코드 샘플입니다.

- OS X 인증서 스토어에서, 캐시된 보안 인증 정보를 주기적으로 지우는 Mac OS X 스크립트

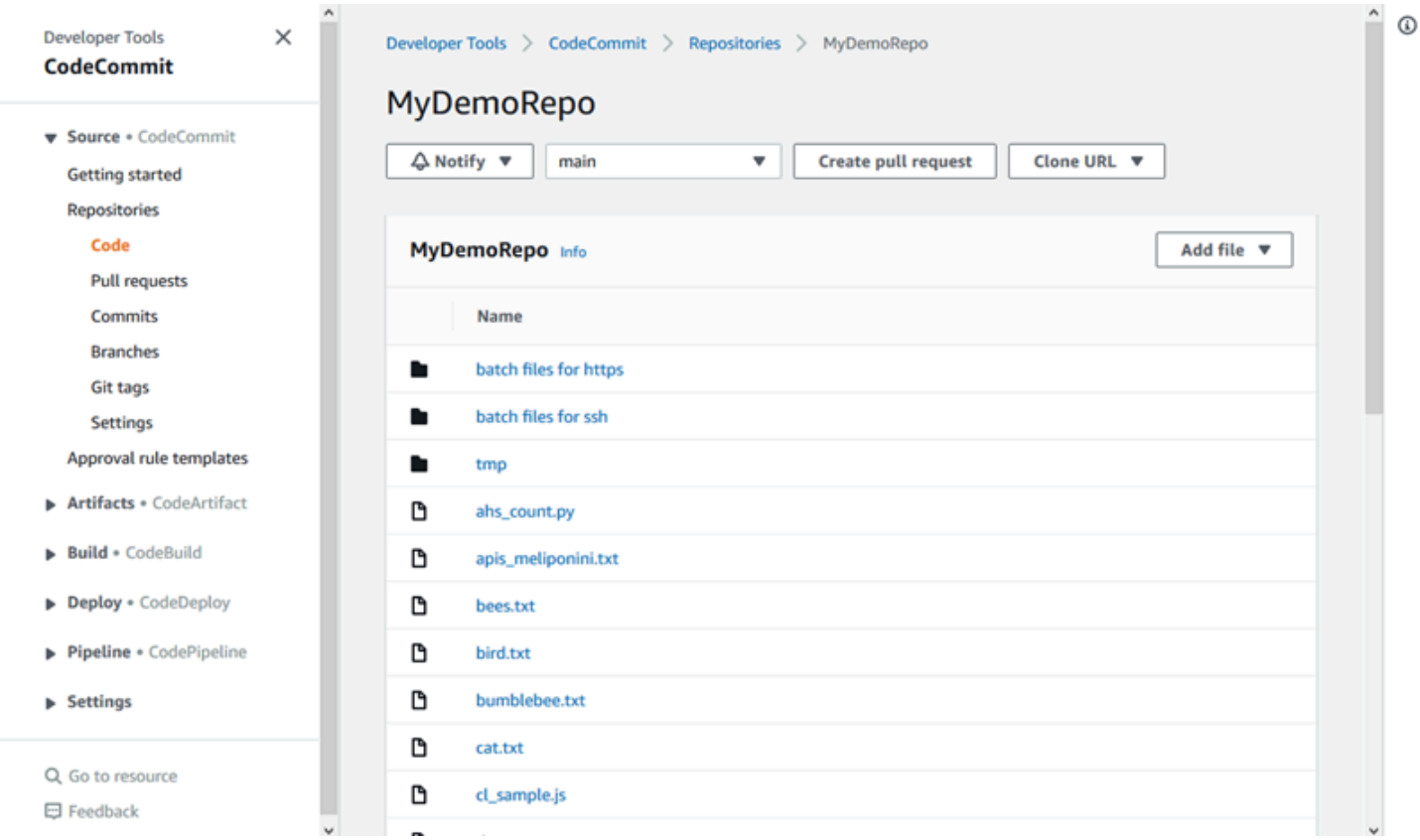
  Mac OS X에서 CodeCommit용 보안 인증 도우미를 사용한다면 캐시된 보안 인증 정보 문제에 익숙할 것입니다. 이 스크립트는 하나의 솔루션을 보여줍니다.

  저자: Nico Coetzee

  2016년 2월 발행

에서 리포지토리 작업 AWS CodeCommit

리포지토리는 CodeCommit의 기본 버전 관리 객체이며, 프로젝트에 대한 코드 및 파일을 안전하게 저장합니다. 또한 첫 번째 커밋부터 마지막 변경 내용까지 프로젝트 기록을 저장합니다. 리포지토리를 다른 사용자와 공유할 수 있으므로 프로젝트에 대해 함께 작업할 수 있습니다. 리포지토리에 AWS 태그를 추가하는 경우 리포지토리 사용자가 이벤트에 대한 이메일(예: 코드에 주석을 달고 있는 다른 사용자)을 수신하도록 알림을 설정할 수 있습니다. 또한 리포지토리에 대한 기본 설정 변경, 콘텐츠 찾아보기 등을 수행할 수 있습니다. 코드 푸시 또는 기타 이벤트가 이메일, 코드 함수 등과 같은 작업을 트리거하도록 리포지토리에 대한 트리거를 생성할 수 있습니다. 변경 내용을 여러 리포지토리로 푸시하도록 로컬 컴퓨터에서 리포지토리(로컬 리포지토리)를 구성할 수도 있습니다.



CodeCommit 리포지토리에 변경 사항을 푸시하려면, 먼저 Amazon Web Services 계정에서 IAM 사용자를 구성하거나 페더레이션 액세스 또는 임시 보안 인증 정보를 위한 액세스를 설정해야 합니다. 자세한 내용은 1단계: CodeCommit에 대한 초기 구성 및 git-remote-codecommit을 AWS CodeCommit 사용하여에 HTTPS 연결을 위한 설정 단계 섹션을 참조하세요.

CodeCommit에서 리포지토리의 다른 측면으로 작업하는 방법에 대해 자세히 알아보려면 파일 작업하기, 풀 요청 작업, 커밋 작업, 브랜치 작업, 사용자 기본 설정으로 작업 섹션을 참조하세요.

CodeCommit으로 마이그레이션하기에 대한 자세한 내용은 CodeCommit으로 마이그레이션 섹션을 참조하세요.

주제

- AWS CodeCommit 리포지토리 생성
- AWS CodeCommit 리포지토리에 연결
- AWS CodeCommit 리포지토리 공유
- AWS CodeCommit 리포지토리에서 이벤트 알림 구성
- 에서 리포지토리 태그 지정 AWS CodeCommit
- AWS CodeCommit 리포지토리에 대한 트리거 관리
- AWS CodeCommit 리포지토리를 Amazon CodeGuru Reviewer와 연결 또는 연결 해제
- CodeCommit 리포지토리 세부 정보 보기
- AWS CodeCommit 리포지토리 설정 변경
- 로컬 리포지토리와 AWS CodeCommit 리포지토리 간의 변경 사항 동기화
- 추가 Git 리포지토리로 커밋 푸시
- 역할을 사용하여 AWS CodeCommit 리포지토리에 대한 교차 계정 액세스 구성
- AWS CodeCommit 리포지토리 삭제

AWS CodeCommit 리포지토리 생성

AWS CodeCommit 콘솔 또는 AWS Command Line Interface (AWS CLI)를 사용하여 빈 CodeCommit 리포지토리를 생성합니다. 리포지토리 생성 후 리포지토리에 태그를 추가하려면 리포지토리에 태그 추가 단원을 참조하세요.

이러한 지침에서는 설정 의 단계를 완료한 것으로 가정합니다.

> **Note**
>
> 사용량에 따라, 리포지토리를 생성하거나 액세스하는 것에 대한 비용이 부과될 수 있습니다. 자세한 내용은 CodeCommit 제품 정보 페이지에서 요금을 참조하세요.

주제

- 리포지토리 생성 (콘솔)
- 리포지토리 생성 (AWS CLI)

리포지토리 생성 (콘솔)

CodeCommit 리포지토리 생성

> **Important**
> 신규 고객은 더 이상에서 리포지토리를 생성할 수 AWS CodeCommit 없습니다. 의 기존 고객은 평소와 같이 API를 계속 사용할 AWS CodeCommit 수 있습니다. 자세히 알아보기

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리전 선택기에서 리포지토리를 생성할 AWS 리전 을 선택합니다. 자세한 내용은 리전 및 Git 연결 엔드포인트 단원을 참조하십시오.
3. 리포지토리 페이지에서 리포지토리 생성을 선택합니다.
4. 리포지토리 생성 페이지의 리포지토리 이름에 리포지토리 이름을 입력합니다.

   > **Note**
   > 리포지토리 이름은 대소문자를 구분합니다. 이 이름은 Amazon Web Services 계정이 속한 AWS 리전 에서 고유해야 합니다.

5. (선택 사항) 설명에 리포지토리에 대한 설명을 입력합니다. 그러면 사용자들이 리포지토리의 용도를 식별하는 데 도움이 됩니다.

   > **Note**
   > 설명 필드에는 콘솔의 마크다운이 표시되며, 모든 HTML 문자와 유효한 Unicode 문자를 모두 사용할 수 있습니다. `GetRepository` 또는 `BatchGetRepositories` API를 사용하는 애플리케이션 개발자인 경우 웹 브라우저에 리포지토리 설명 필드를 표시하려면 CodeCommit API 참조를 참조하세요.

6. (선택 사항) 태그 추가를 선택하여 하나 이상의 리포지토리 태그(AWS 리소스를 구성하고 관리하는 데 도움이 되는 사용자 지정 속성 레이블)를 리포지토리에 추가합니다. 자세한 내용은 에서 리포지토리 태그 지정 AWS CodeCommit 단원을 참조하십시오.
7. (선택 사항) 추가 구성을 확장하여이 리포지토리의 데이터 암호화 및 복호화에 기본 AWS 관리형 키 또는 자체 고객 관리형 키를 사용할지 여부를 지정합니다. 자체 고객 관리형 키를 사용하기로 선택한 경우 리포지토리를 생성하는 AWS 리전 에서 사용할 수 있고 키가 활성 상태인지 확인해야

합니다. 자세한 내용은 AWS Key Management Service 및 AWS CodeCommit 리포지토리 암호화 단원을 참조하십시오.

8. (선택 사항) 이 리포지토리에 Java나 Python 코드가 들어가고 CodeGuru Reviewer로 해당 코드를 분석하려 한다면, Java 및 Python용 Amazon CodeGuru Reviewer 활성화를 선택합니다. CodeGuru Reviewer는 다양한 기계 학습 모델을 사용하여 풀 요청의 코드 결함을 찾아내고 개선점과 해결책을 제안합니다. 자세한 내용은 Amazon CodeGuru Reviewer 사용 안내서를 참조하십시오.
9. 생성(Create)을 선택합니다.

리포지토리를 생성한 후에는 이 리포지토리에 연결한 다음, CodeCommit 콘솔을 활용하거나 로컬 Git 클라이언트를 활영하거나 또는 CodeCommit 리포지토리를 원하는 IDE와 통합함으로써 코드 추가를 시작할 수 있습니다. 자세한 내용은 에 대한 설정 AWS CodeCommit 단원을 참조하십시오. 또한 지속적인 배포 파이프라인에 리포지토리를 추가할 수 있습니다. 자세한 내용은 단일 파이프라인 예제를 참조하세요.

리포지토리를 복제할 때 사용하는 URL과 같은 새 CodeCommit 리포지토리에 대한 정보를 얻으려면 리포지토리 이름을 목록에서 선택하거나, 아니면 해당 리포지토리 이름 옆에 사용할 연결 프로토콜을 선택하면 됩니다.

이 리포지토리를 다른 사람과 공유하려면 사용할 HTTPS 또는 SSH 링크를 전송하여 리포지토리를 복제해야 합니다. 리포지토리 액세스에 필요한 권한이 있는지 확인하세요. 자세한 내용은 리포지토리 공유 및 AWS CodeCommit에 대한 인증 및 액세스 제어 섹션을 참조하세요.

리포지토리 생성 (AWS CLI)

AWS CLI 를 사용하여 CodeCommit 리포지토리를 생성할 수 있습니다. 콘솔과 달리, AWS CLI를 사용하여 리포지토리를 생성할 경우 리포지토리에 태그를 추가할 수 있습니다.

> **Important**
>
> 신규 고객은 더 이상에서 리포지토리를 생성할 수 AWS CodeCommit 없습니다. 의 기존 고객은 평소와 같이 API를 계속 사용할 AWS CodeCommit 수 있습니다. 자세히 알아보기

1. 리포지토리가 AWS 리전 있는 AWS CLI 로를 구성했는지 확인합니다. 리전을 확인하려면 다음 명령을 명령줄 또는 터미널에서 실행하고 기본 리전 이름에 대한 정보를 검토하세요.

```
aws configure
```

기본 리전 이름은 CodeCommit의 리포지토리에 AWS 리전 대한와 일치해야 합니다. 자세한 내용은 리전 및 Git 연결 엔드포인트 단원을 참조하십시오.

2. 다음을 지정하여 create-repository 명령을 실행합니다.

   - CodeCommit 리포지토리를 고유하게 식별하는 이름(`--repository-name` 옵션 사용).

     > **Note**
     > 이 이름은 Amazon Web Services 계정 전체에서 고유해야 합니다.

   - CodeCommit 리포지토리에 대한 선택적 주석(`--repository-description` 옵션 사용).
   - CodeCommit 리포지토리에 대해 태그로 사용할 선택적 키 값 페어(`--tags` 옵션 사용).
   - 이 리포지토리를 암호화하고 복호화할 때 사용할 고객 관리형 키(선택 사항)입니다. 모든 리포지토리는 전송 중 및 미사용 시 AWS KMS의 키를 사용하여 암호화됩니다. 키가 지정되지 않은 경우 기본 AWS 관리형 키`aws/codecommit`가 사용됩니다.

   예를 들어, `MyDemoRepo`라는 설명이 달려 있고 *`Team`*이라는 키 이름에 *`Saanvi`*라는 키 값의 태그가 존재하는 `"My demonstration repository"`라는 명칭의 CodeCommit 리포지토리를 생성하려면, 다음 명령을 사용합니다.

```
aws codecommit create-repository --repository-name MyDemoRepo --repository-
description "My demonstration repository" --tags Team=Saanvi
```

   > **Note**
   > 설명 필드에는 콘솔의 마크다운이 표시되며, 모든 HTML 문자와 유효한 Unicode 문자를 모두 사용할 수 있습니다. `GetRepository` 또는 `BatchGetRepositories` API를 사용하는 애플리케이션 개발자인 경우 웹 브라우저에 리포지토리 설명 필드를 표시하려면 CodeCommit API 참조를 참조하세요.

3. 이 명령이 제대로 실행되면 다음 정보를 포함하는 `repositoryMetadata` 객체가 출력됩니다.

   - 설명(`repositoryDescription`)
   - 고유한 시스템 생성 ID(`repositoryId`).
   - 이름(`repositoryName`)
   - CodeCommit 리포지토리 (`accountId`)와 연결된 Amazon Web Services 계정의 ID.

다음은 위 예제 명령의 출력 예입니다.

```
{
    "repositoryMetadata": {
        "repositoryName": "MyDemoRepo",
        "cloneUrlSsh": "ssh://git-codecommit.us-east-2.amazonaws.com/v1/
repos/MyDemoRepo",
        "lastModifiedDate": 1446071622.494,
        "repositoryDescription": "My demonstration repository",
        "cloneUrlHttp": "https://git-codecommit.us-east-2.amazonaws.com/v1/
repos/MyDemoRepo",
        "defaultBranch": main,
        "kmsKeyId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "creationDate": 1446071622.494,
        "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE",
        "Arn": "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
        "accountId": "111111111111"
    }
}
```

> **Note**
>
> 리포지토리 생성 시 추가된 태그는 출력에서 반환되지 않습니다. 리포지토리와 연결된 태그 목록을 보려면 list-tags-for-resource 명령을 실행합니다.

4. CodeCommit 리포지토리의 이름과 ID를 기록해 둡니다. CodeCommit 리포지토리에 대한 정보를 모니터링하고 변경하는 데 필요합니다. AWS CLI를 사용할 경우 특히 필요합니다.

   이름 또는 ID를 잊어버린 경우 CodeCommit 리포지토리 세부 정보 보기 (AWS CLI)의 지침을 따릅니다.

리포지토리를 생성하고 리포지토리에 연결 및 코드 추가를 시작할 수 있습니다. 자세한 내용은 리포지토리에 연결 단원을 참조하십시오. 또한 지속적인 배포 파이프라인에 리포지토리를 추가할 수 있습니다. 자세한 내용은 단일 파이프라인 예제를 참조하세요.

AWS CodeCommit 리포지토리에 연결

CodeCommit 리포지토리에 처음으로 연결하는 경우 일반적으로 리포지토리의 콘텐츠를 로컬 시스템에 복제합니다. CodeCommit 콘솔에서 직접 리포지토리에 파일을 추가하고 편집할 수도 있습니다. 또는 로컬 리포지토리가 이미 있는 경우 CodeCommit 리포지토리를 원격으로 추가할 수 있습니다. 이 주제에서는 CodeCommit 리포지토리에 대한 연결 지침을 제공합니다. 기존 리포지토리를 CodeCommit으로 마이그레이션하려면 CodeCommit으로 마이그레이션 단원을 참조하세요.

> **Note**
>
> 사용량에 따라, 리포지토리를 생성하거나 액세스하는 것에 대한 비용이 부과될 수 있습니다. 자세한 내용은 CodeCommit 제품 정보 페이지에서 요금을 참조하세요.

주제

- CodeCommit 리포지토리에 연결하기 위한 사전 조건
- 리포지토리를 복제하여 CodeCommit 리포지토리에 연결
- 로컬 리포지토리를 CodeCommit 리포지토리에 연결

CodeCommit 리포지토리에 연결하기 위한 사전 조건

CodeCommit 리포지토리를 복제하거나 로컬 리포지토리를 CodeCommit 리포지토리에 연결하기 전에 다음을 수행합니다.

- CodeCommit에 연결하는 데 필요한 소프트웨어 및 설정을 사용하여 로컬 컴퓨터를 구성해야 합니다. 여기에는 Git의 설치 및 구성이 포함됩니다. 자세한 내용은 설정 및 Git 및 시작하기 AWS CodeCommit 섹션을 참조하세요.
- 연결하고 싶은 CodeCommit 리포지토리의 복제 URL이 있어야 합니다. 자세한 내용은 리포지토리 세부 정보 보기 단원을 참조하십시오.

  CodeCommit 리포지토리를 아직 생성하지 않은 경우에는 리포지토리 생성의 지침을 따르고 CodeCommit 리포지토리의 복제 URL을 복사한 다음 이 페이지로 돌아옵니다.

  CodeCommit 리포지토리가 있지만 이름을 모르는 경우에는 리포지토리 세부 정보 보기의 지침을 따릅니다.
- 연결을 원하는 CodeCommit 리포지토리의 로컬 복사본을 저장하려면 로컬 시스템에 저장 공간이 있어야 합니다. (CodeCommit 리포지토리의 이 로컬 복사본을 로컬 리포지토리라고 합니다.) 해당 위

치로 전환하여 그 곳에서 Git 명령을 실행합니다. 예를 들어, 테스트 목적으로 임시 복제를 만들려면 `/tmp`(Linux, macOS, Unix) 또는 `c:\temp`(Windows)를 사용할 수 있습니다. 이는 이러한 예에서 사용되는 디렉터리 경로입니다.

> **Note**
> 원하는 디렉터리를 사용할 수 있습니다. 장기적인 용도로 리포지토리를 복제하는 경우 작업 디렉터리에서 복제물을 생성해 보되 임시 파일용으로는 생성하지 마시기 바랍니다. `/tmp` 또는 `c:\temp`와 다른 디렉터리를 사용하는 경우 다음 지침을 따를 때 여기의 디렉터리 대신에 해당 디렉터리를 사용해야 합니다.

리포지토리를 복제하여 CodeCommit 리포지토리에 연결

로컬 리포지토리가 아직 없는 경우 이 절차의 단계에 따라 CodeCommit 리포지토리를 로컬 시스템에 복제합니다.

1. 설정 을 포함한 사전 필수 단계를 완료합니다.

   > **Important**
   > 설정을 완료하지 않았으면 리포지토리에 연결하거나 리포지토리를 복제할 수 없습니다.

2. `/tmp` 디렉터리 또는 `c:\temp` 디렉터리에서 Git을 사용하여 clone 명령을 실행합니다. 다음 예제는 미국 동부(오하이오) 리전에서 *`MyDemoRepo`*라는 리포지토리를 복제하는 방법을 보여 줍니다.

   Git 보안 인증 정보 또는 AWS CLI에 포함된 보안 인증 도우미를 사용하는 HTTPS의 경우에는 다음과 같이 합니다.

   ```
   git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-
   demo-repo
   ```

   기본 프로파일과 AWS 리전 이 AWS CLI에 구성되어 있다고 가정하며 git-remote-codecommit를 사용하는 HTTPS의 경우에는 다음과 같이 합니다.

   ```
   git clone codecommit://MyDemoRepo my-demo-repo
   ```

   SSH의 경우에는 다음과 같이 합니다.

```
git clone ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-demo-repo
```

이 예제에서 `git-codecommit.us-east-2.amazonaws.com`은 리포지토리가 있는 미국 동부(오하이오) 리전에 대한 Git 연결 지점이고, `MyDemoRepo`는 CodeCommit 리포지토리의 이름을 나타내며, `my-demo-repo`는 `/tmp` 디렉터리 또는 `c:\temp` 디렉터리에 Git이 생성하는 디렉터리의 이름을 나타냅니다. CodeCommit을 AWS 리전 지원하는 및 이에 대한 Git 연결에 대한 자세한 내용은 섹션을 AWS 리전참조하세요리전 및 Git 연결 엔드포인트.

> **Note**
>
> Windows 운영 체제에서 SSH를 사용하여 리포지토리를 복제할 경우 다음과 같이 SSH 키 ID를 연결 문자열에 추가해야 할 수 있습니다.
>
> ```
> git clone ssh://Your-SSH-Key-ID@git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-demo-repo
> ```
>
> 자세한 내용은 Windows에서 SSH 연결 및 문제 해결 섹션을 참조하세요.

Git에서 디렉터리를 생성한 후 CodeCommit 리포지토리의 복사본을 새로 생성된 디렉터리로 풀다운합니다.

CodeCommit 리포지토리가 새 리포지토리이거나 비어 있는 경우 빈 리포지토리를 복제하고 있다는 메시지가 표시됩니다. 이는 예상된 동작입니다.

> **Note**
>
> Git에서 CodeCommit 리포지토리를 찾을 수 없거나 CodeCommit 리포지토리에 연결할 권한이 없다는 오류가 표시되는 경우, 사전 조건(IAM 사용자에게 권한 할당, 로컬 시스템에서 Git 및 CodeCommit에 대한 IAM 사용자 보안 인증 정보 설정 등)을 완료했는지 확인합니다. 또한 올바른 리포지토리 이름을 지정했는지 확인합니다.

로컬 리포지토리를 CodeCommit 리포지토리에 성공적으로 연결한 후에는 로컬 리포지토리에서 Git 명령을 실행하여 커밋, 분기 및 태그를 생성한 다음 CodeCommit 리포지토리를 대상으로 푸시하거나 풀할 수 있습니다.

로컬 리포지토리를 CodeCommit 리포지토리에 연결

로컬 리포지토리가 이미 있고 CodeCommit 리포지토리를 원격 리포지토리로 추가하려면, 다음 단계를 완료합니다. 원격 리포지토리가 이미 있고 커밋을 CodeCommit 및 다른 원격 리포지토리에 푸시하려면, 두 리포지토리로 커밋 푸시의 단계를 따릅니다.

1. 사전 조건을 완료합니다.
2. 명령 프롬프트 또는 터미널에서 로컬 리포지토리 디렉터리로 전환한 다음 git remote add 명령을 실행하여 CodeCommit 리포지토리를 로컬 리포지토리에 대한 원격 리포지토리로 추가합니다.

   예를 들어, 다음 명령은 **origin**라는 원격 리포지토리를 'https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo:'에 추가합니다.

   HTTPS의 경우에는 다음과 같이 합니다.

   ```
   git remote add origin https://git-codecommit.us-east-2.amazonaws.com/v1/repos/
   MyDemoRepo
   ```

   SSH의 경우에는 다음과 같이 합니다.

   ```
   git remote add origin ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/
   MyDemoRepo
   ```

   이 명령은 아무 것도 반환하지 않습니다.
3. CodeCommit 리포지토리를 로컬 리포지토리에 대해 원격으로 추가했는지 확인하려면 git remote -v 명령을 실행하여 다음과 유사한 출력이 생성되어야 합니다.

   HTTPS의 경우에는 다음과 같이 합니다.

   ```
   origin  https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (fetch)

   origin  https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (push)
   ```

   SSH의 경우에는 다음과 같이 합니다.

```
origin  ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (fetch)

origin  ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (push)
```

로컬 리포지토리를 CodeCommit 리포지토리에 성공적으로 연결한 후에는 로컬 리포지토리에서 Git 명령을 실행하여 커밋, 분기 및 태그를 생성한 다음 CodeCommit 리포지토리를 대상으로 푸시하거나 풀할 수 있습니다.

AWS CodeCommit 리포지토리 공유

CodeCommit 리포지토리를 생성한 후 이를 다른 사용자와 공유할 수 있습니다. 먼저 CodeCommit에 액세스할 때 페더레이션 액세스, 임시 보안 인증 정보, IAM Identity Center와 같은 웹 자격 증명 공급자를 사용할지, 아니면 IAM 사용자로 Git 보안 인증 정보나 SSH 키 페어를 사용할지 결정합니다. 전자를 사용하는 경우에는 자격 증명 공급자에 대한 사용자, 엑세스, 권한 등을 설정한 다음 사용자가 git-remote-codecommit을 사용하는 데 필요한 지침을 제공해야 합니다. 자세한 내용은 git-remote-codecommit을 AWS CodeCommit 사용하여에 HTTPS 연결을 위한 설정 단계 및 교체 자격 증명을 사용하여 AWS CodeCommit 리포지토리에 연결 섹션을 참조하세요.

페더레이션 액세스나 자격 증명 공급자와 함께 Git 보안 인증 정보 또는 SSH 키 페어를 사용할 수는 없지만, 다수의 IDE는 이러한 보안 인증 정보들과 가장 잘 작동합니다. 이런 경우에는 Git 클라이언트 또는 IDE를 복제 및 사용하여 리포지토리에 접속할 때 사용자에게 어떤 프로토콜(HTTPS 또는 SSH)을 권장할 것인지 결정합니다. 그런 다음 리포지토리를 공유하고 싶은 사용자들에게 URL과 접속 정보를 전송합니다. 보안 요건에 따라, 리포지토리를 공유하는 데는 IAM 그룹 생성, 이 그룹에 관리형 정책 적용, 액세스 권한 세분화를 위한 IAM 정책 편집, IAM 역할 생성 및 사용 등이 필요할 수도 있습니다.

> **Note**
>
> 사용자에게 리포지토리에 대한 콘솔 액세스 권한을 부여한 후 사용자는 Git 클라이언트 또는 기타 연결을 설정할 필요 없이 콘솔에서 직접 파일을 추가하거나 편집할 수 있습니다. 자세한 내용은 AWS CodeCommit 리포지토리에 파일 생성 또는 추가 및 AWS CodeCommit 리포지토리에서 파일의 콘텐츠 편집 섹션을 참조하세요.

이 지침들은 사용자가 설정 및 리포지토리 생성의 단계를 이미 완료했다는 가정하에 작성되었습니다.

> **Note**
> 사용량에 따라, 리포지토리를 생성하거나 액세스하는 것에 대한 비용이 부과될 수 있습니다. 자세한 내용은 CodeCommit 제품 정보 페이지에서 요금을 참조하세요.

주제

- 사용자들과 공유할 접속 프로토콜 선택
- 리포지토리를 위한 IAM 정책 생성
- 리포지토리 사용자를 위한 IAM 그룹 생성
- 해당 사용자들과 접속 정보 공유

사용자들과 공유할 접속 프로토콜 선택

CodeCommit에 리포지토리를 생성하면 엔드포인트가 두 개 생성되는데, 하나는 HTTPS 접속을 위한 것이고 다른 하나는 SSH 접속을 위한 것입니다. 둘 다 네트워크를 통한 안전한 접속이 가능합니다. 사용자는 둘 중 한 가지 프로토콜을 사용할 수 있습니다. 해당 사용자들에게 어떤 프로토콜을 권장하든지 상관없이 두 엔드포인트 모두 활성화된 상태를 유지합니다.

HTTPS 연결에는 다음 중 하나가 필요합니다.

- IAM 사용자가 IAM에서 직접 생성할 수 있는 Git 보안 인증 정보. Git 보안 인증 정보는 리포지토리 사용자들이 설정하여 사용하기 가장 쉬운 방법입니다.
- 리포지토리 사용자가 자격 증명 프로필에서 구성해야 하는 수임할 AWS 액세스 키 또는 역할입니다. git-remote-codecommit(권장)를 구성하거나 AWS CLI에 포함된 보안 인증 도우미를 구성할 수 있습니다. 이는 루트 계정 또는 페더레이션형 사용자에 사용할 수 있는 유일한 방법입니다.

SSH 연결을 위해서는 사용자가 다음 작업을 수행해야 합니다.

- 퍼블릭-프라이빗 키 페어를 생성합니다.
- 퍼블릭 키를 저장합니다.
- 퍼블릭 키를 IAM 사용자와 연결합니다.
- 로컬 컴퓨터에서 알려진 호스트 파일을 구성합니다.
- 로컬 컴퓨터에서 config 파일을 생성하고 유지합니다.

이것은 더 복잡한 구성 절차이므로 CodeCommit 연결에는 HTTPS 및 Git 보안 인증 정보를 선택하는 것이 좋습니다.

HTTPS, SSH, Git, git-remote-codecommit 및 원격 리포지토리에 대한 자세한 내용은 설정 , 교체 자격 증명을 사용하여 AWS CodeCommit 리포지토리에 연결 단원 또는 해당 Git 문서를 참조하세요.

> **Note**
>
> Git에서는 다양한 접속 프로토콜을 지원하지만 CodeCommit에서는 로컬 프로토콜 또는 일반 HTTP와 같은 비보안 프로토콜을 통한 접속은 지원하지 않습니다.

리포지토리를 위한 IAM 정책 생성

AWS 는 CodeCommit용 IAM에서 세 가지 관리형 정책을 제공합니다. 이 정책은 편집할 수 없으며 Amazon Web Services 계정에 연결된 모든 리포지토리에 적용됩니다. 그러나 이러한 정책을 템플릿으로 사용함으로써 자신이 공유하고 싶은 리포지토리에만 적용되는 고유한 사용자 지정 관리형 정책을 생성할 수 있습니다. 이렇게 생성한 사용자 지정 관리형 정책은 자신이 공유하고 싶은 리포지토리에 한해 적용할 수 있습니다. 자세한 내용은 관리형 정책 및 IAM 사용자 및 그룹을 참조하세요.

> **Tip**
>
> 리포지토리에 대한 액세스를 더 미세하게 관리하기 위해 고객 관리형 정책을 하나 이상 생성하여 이 정책을 여러 IAM 사용자 및 그룹에 적용할 수 있습니다.

관리형 정책 내용 검토 및 정책을 사용하여 권한 생성 및 적용에 대한 자세한 내용은 AWS CodeCommit에 대한 인증 및 액세스 제어 단원을 참조하세요.

리포지토리를 위한 고객 관리형 정책 만들기

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/ IAM 콘솔을 엽니다.
2. 대시보드 탐색 영역에서 정책을 선택한 다음 정책 생성을 선택합니다.
3. 정책 생성 페이지에서 관리형 정책 가져오기를 선택합니다.
4. 관리형 정책 가져오기 페이지의 필터 정책에 **`AWSCodeCommitPowerUser`**를 입력합니다. 정책 이름 옆에 있는 버튼을 선택한 다음 가져오기를 선택합니다.

5. 정책 생성 페이지에서 JSON을 선택합니다. 다음과 같이 CodeCommit 작업을 위한 `Resource` 줄의 “*” 부분을 CodeCommit 리포지토리의 Amazon 리소스 이름(ARN)으로 교체합니다.

```
"Resource": [
 "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo"
 ]
```

> **Tip**
>
> CodeCommit 리포지토리의 ARN을 찾으려면, CodeCommit 콘솔로 이동하고 목록에서 리포지토리 이름을 선택한 다음 설정을 선택합니다. 자세한 내용은 리포지토리 세부 정보 보기 단원을 참조하십시오.

이 정책을 둘 이상의 리포지토리에 적용하려면, ARN을 지정하여 각 리포지토리를 리소스로 추가합니다. 다음과 같이 각 리소스 명령문 사이에 쉼표를 넣습니다.

```
"Resource": [
 "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
 "arn:aws:codecommit:us-east-2:111111111111:MyOtherDemoRepo"
 ]
```

편집이 완료되면 정책 검토를 선택합니다.

6. 정책 검토 페이지의 이름에 정책의 새 이름을 입력합니다(예: *`AWSCodeCommitPowerUser-MyDemoRepo`*). 선택 사항으로 이 정책에 대한 설명을 제공할 수 있습니다.
7. 정책 생성을 선택합니다.

리포지토리 사용자를 위한 IAM 그룹 생성

자신의 리포지토리에 대한 액세스를 관리하려면, 리포지토리 사용자에 대한 IAM 그룹을 생성하고, 해당 그룹에 IAM 사용자를 추가한 다음, 이전 단계에서 생성한 고객 관리형 정책을 연결합니다. 또는 연결된 고객 관리형 정책을 사용하여 역할을 생성하고, 사용자에게 해당 역할을 수임하도록 할 수 있습니다.

SSH를 사용하는 경우에는 IAMUserSSHKeys 그룹에 다른 관리형 정책, 즉 사용자들이 자신의 SSH 퍼블릭 키를 업로드하도록 허용하는 IAM 관리형 정책을 연결하고 이를 사용자들이 CodeCommit에 접속할 때 사용할 IAM 사용자와 연동해야 합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/ IAM 콘솔을 엽니다.
2. 대시보드 탐색 영역에서 그룹을 선택한 다음 새 그룹 생성을 선택합니다.
3. 그룹 이름 설정 페이지의 그룹 이름에 그룹 이름(예: *MyDemoRepoGroup*)을 입력한 다음 다음 단계를 선택합니다. 그룹 이름에는 리포지토리 이름을 포함시키는 것이 좋습니다.

   > **Note**
   >
   > 그룹 이름은 Amazon Web Services 계정 전체에서 고유해야 합니다.

4. 이전 섹션에서 생성한 고객 관리형 정책(예: AWSCodeCommitPowerUser-MyDemoRepo) 옆의 상자를 선택합니다.
5. 검토 페이지에서 그룹 생성을 선택합니다. IAM은 지정된 정책이 이미 연결된 상태로 이 그룹을 생성합니다. 해당 그룹이 Amazon Web Services 계정에 연결된 그룹 목록에 나타납니다.
6. 목록에서 그룹을 선택합니다.
7. 그룹 요약 페이지에서 사용자 탭을 선택한 다음 그룹에 사용자 추가를 선택합니다. Amazon Web Services 계정과 연결된 모든 사용자를 표시하는 목록에서, CodeCommit 리포지토리에 대한 액세스를 허용할 사용자 옆의 상자를 선택한 다음 사용자 추가를 선택합니다.

   > **Tip**
   >
   > 검색 상자를 사용하여 이름별로 사용자를 빠르게 찾을 수 있습니다.

8. 사용자를 추가했으면 IAM 콘솔을 닫습니다.

해당 사용자들과 접속 정보 공유

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리전 선택기에서 리포지토리가 생성된 AWS 리전 를 선택합니다. 리포지토리는에 고유합니다 AWS 리전. 자세한 내용은 리전 및 Git 연결 엔드포인트 단원을 참조하십시오.
3. 리포지토리 페이지에서 공유할 리포지토리를 선택합니다.
4. URL 복제에서 사용자가 사용하게 할 프로토콜을 선택합니다. 그러면 연결 프로토콜에 대한 복제 URL이 복사됩니다.
5. 사용자에게 복제 URL을 설치, 프로필 AWS CLI구성 또는 Git 설치와 같은 다른 지침과 함께 전송합니다. 연결 프로토콜(예: HTTPS)에 대한 구성 정보를 포함해야 합니다.

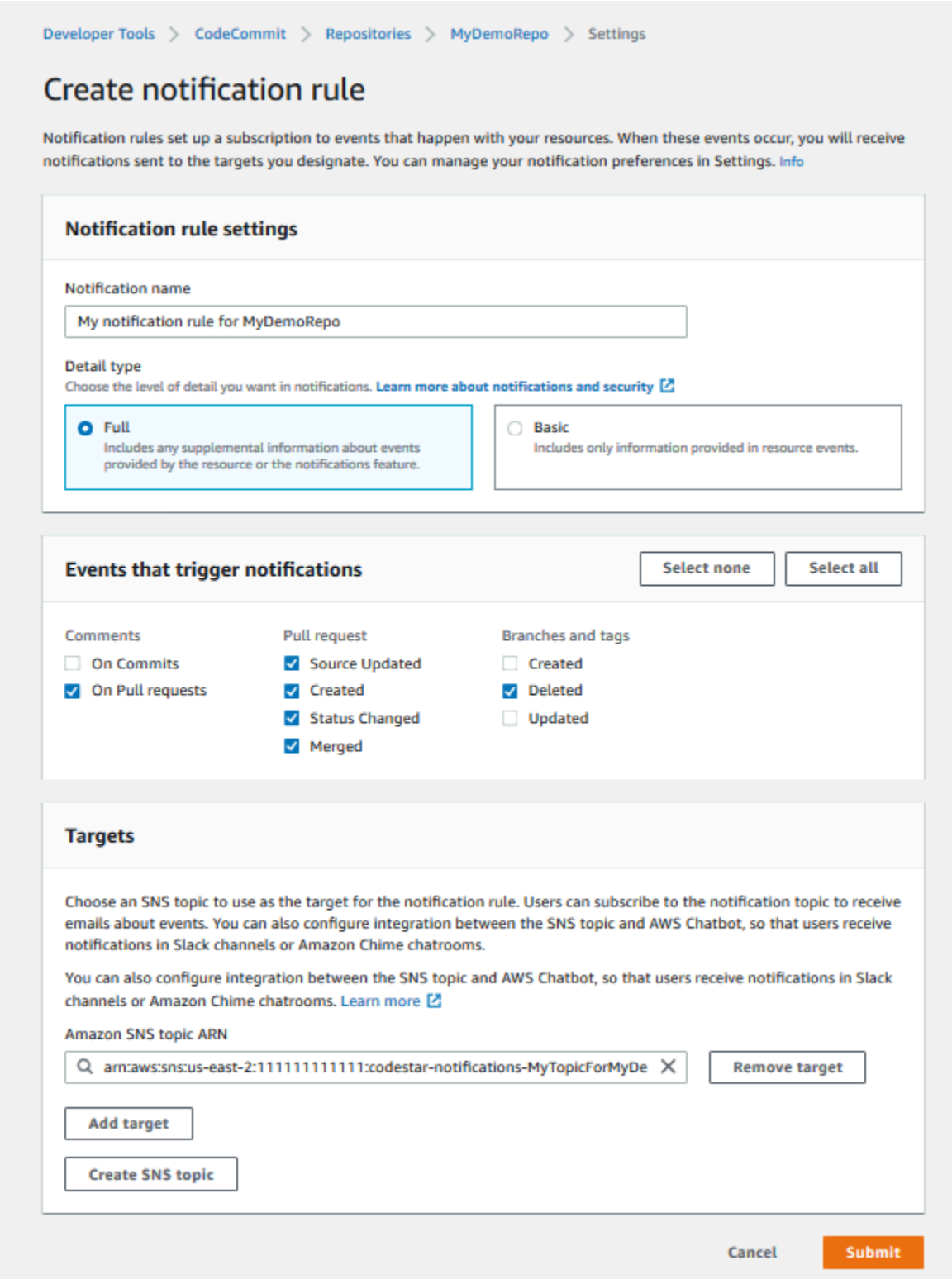
다음 예시 이메일에서는 미국 동부(오하이오)(us-east-2) 리전에서 HTTPS 접속 프로토콜 및 Git 보안 인증 정보를 통해 MyDemoRepo 리포지토리에 접속하는 사용자들에게 관련 정보를 제공합니다. 이 이메일은 사용자가 이미 Git을 설치했고 Git 사용에 익숙하다는 가정 하에 작성되었습니다.

```
I've created a CodeCommit repository for us to use while working on our project.
The name of the repository is MyDemoRepo, and
it is in the  US East (Ohio) (us-east-2) region.
Here's what you need to do in order to get started using it:

1. Make sure that your version of Git on your local computer is 1.7.9 or later.
2. Generate Git credentials for your IAM user by signing into the IAM console
 here: https://console.aws.amazon.com/iam/  .
Switch to the Security credentials tab for your IAM user and choose the Generate button
 in HTTPS Git credentials for CodeCommit.
Make sure to save your credentials in a secure location!
3. Switch to a directory of your choice and clone the CodeCommit repository to your
 local machine by running the following command:
    git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-
demo-repo
4. When prompted for user name and password, use the Git credentials you just saved.


That's it!  If you'd like to learn more about using CodeCommit, you can start with the
 tutorial here.
```

설정 에서 완전한 설정 관련 지침을 볼 수 있습니다.

AWS CodeCommit 리포지토리에서 이벤트 알림 구성

개발자가 지정하는 리포지토리 이벤트 유형에 대한 이메일을 리포지토리 사용자가 수신하도록 리포지토리에 대한 알림 규칙을 설정할 수 있습니다. 이벤트가 알림 규칙 설정과 일치하면 알림이 전송됩니다. 알림을 위해 사용하거나 Amazon Web Services 계정에 기존 알림을 사용하기 위해 Amazon SNS 항목을 만들 수 있습니다. CodeCommit 콘솔과를 사용하여 알림 규칙을 AWS CLI 구성할 수 있습니다.



주제

- 리포지토리 알림 규칙 사용
- 알림 규칙 생성

- 알림 변경 또는 비활성화
- 알림 삭제

리포지토리 알림 규칙 사용

알림을 구성하면 누군가가 다른 사용자에게 영향을 주는 작업을 수행하는 경우 사용자에게 이메일을 전송하여 리포지토리 사용자를 도울 수 있습니다. 예를 들어 커밋에 대한 주석이 작성될 때 알림을 전송하도록 알림 규칙을 구성할 수 있습니다. 이 구성에서는 리포지토리 사용자가 커밋의 코드 행에 주석을 추가하면 다른 리포지토리 사용자에게 이메일이 발송됩니다. 이들 리포지토리 사용자는 로그인하여 주석을 볼 수 있습니다. 주석에 댓글을 달아도 이메일이 생성되며 리포지토리 사용자에게 알려집니다.

알림 규칙은 리포지토리 트리거와 다르며 2019년 11월 5일 이전까지 CodeCommit 콘솔에서 구성할 수 있었던 알림과도 다릅니다.

- 일부 저장소 이벤트에 대한 이메일을 보내기 위해 Amazon SNS를 사용하도록 트리거를 구성 할 수 있지만 이러한 이벤트는 분기 작성 및 분기로 코드 푸시와 같은 조작 이벤트로 제한됩니다. 트리거는 CloudWatch Events 규칙을 사용하여 리포지토리 이벤트를 평가하지 않습니다. 범위가 더욱 제한됩니다. 트리거 사용에 대한 자세한 내용은 리포지토리 트리거 관리 단원을 참조하세요.
- 2019년 11월 5일 이전까지 구성된 알림은 사용 가능한 이벤트 유형이 적었으며 Amazon Chime 채팅룸 또는 Slack 채널과 통합되도록 구성할 수 없었습니다. 2019년 11월 5일 이전까지 구성된 알림을 계속해서 사용할 수는 있지만 이러한 유형의 알림을 생성할 수 없습니다. 대신에 알림 규칙을 생성하고 사용합니다. 2019년 11월 5일 이전까지 생성된 알림을 비활성화하거나 삭제하고 알림 규칙을 사용하는 것이 좋습니다. 자세한 내용은 알림 규칙 생성 및 알림 삭제 섹션을 참조하세요.

알림 규칙 생성

알림 규칙을 사용하면 풀 요청이 리포지토리에 생성된 경우와 같이 중요한 변경 사항을 사용자에게 알릴 수 있습니다. 알림 규칙은 알림을 보내는 데 사용되는 이벤트와 Amazon SNS 주제를 모두 지정합니다. 자세한 내용은 알림이란 무엇입니까?를 참조하세요.

> **Note**
>
> 이 기능은 유럽(밀라노) 리전에서 사용할 수 없습니다. 해당 지역에서 사용 가능한 환경에서 알림을 구성하는 방법에 대해 알아보려면 리포지토리 알림 구성을 참조하세요.

콘솔 또는를 사용하여 알림 규칙을 AWS CLI 생성할 수 있습니다 AWS CodeCommit.

알림 규칙을 생성하려면 (콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codecommit/ CodeCommit 콘솔을 엽니다.
2. 리포지토리를 선택하고 알림 규칙을 추가하려는 리포지토리를 선택합니다.
3. 리포지토리 페이지에서 알림을 선택하고 알림 규칙 생성을 선택합니다. 리포지토리의 설정 페이지로 이동하여 알림 규칙 생성을 선택할 수도 있습니다.
4. 알림 이름에 규칙에 대한 이름을 입력합니다.
5. 알림에 포함된 Amazon EventBridge에 제공된 정보만 원하는 경우 세부 정보 유형에서 기본을 선택합니다. Amazon EventBridge에 제공된 정보와 CodeCommit 또는 알림 관리자에서 제공할 수 있는 정보를 포함하려는 경우 전체를 선택합니다.

   자세한 내용은 알림 내용 및 보안 이해를 참조하세요.
6. 알림을 트리거하는 이벤트에서 알림을 보내고자 하는 이벤트를 선택합니다. 자세한 내용은 리포지토리의 알림 규칙 이벤트를 참조하시기 바랍니다.
7. 대상에서 다음 중 하나를 수행합니다.
   - 알림에 사용할 리소스를 이미 구성한 경우 대상 유형 선택에서 채팅 애플리케이션에서 Amazon Q Developer(Slack) 또는 SNS 주제를 선택합니다. 대상 선택에서 클라이언트 이름(채팅 애플리케이션의 Amazon Q Developer에 구성된 Slack 클라이언트의 경우) 또는 Amazon SNS 주제의 Amazon 리소스 이름(ARN)(알림에 필요한 정책으로 이미 구성된 Amazon SNS 주제의 경우)을 선택합니다.
   - 알림과 함께 사용할 리소스를 구성하지 않은 경우 대상 생성을 선택한 다음 SNS 주제를 선택합니다. codestar-notifications- 뒤에 주제 이름을 입력한 다음 생성을 선택합니다.

> **Note**
>
> - 알림 규칙을 만드는 과정에서 Amazon SNS 주제를 생성하면 알림 기능이 주제에 이벤트를 게시할 수 있도록 허용하는 정책이 적용됩니다. 알림 규칙에 대해 생성된 주제를 사용하면 이 리소스에 대한 알림을 받기를 원하는 사용자만 구독할 수 있습니다.
> - 알림 규칙 생성의 일부로 채팅 애플리케이션 클라이언트에서 Amazon Q Developer를 생성할 수 없습니다. 채팅 애플리케이션에서 Amazon Q Developer(Slack)를 선택하면 채팅 애플리케이션에서 Amazon Q Developer의 클라이언트를 구성하라는 버튼이 표시

됩니다. 이 옵션을 선택하면 채팅 애플리케이션 콘솔에서 Amazon Q Developer가 열립니다. 자세한 내용은 채팅 애플리케이션에서 알림과 Amazon Q Developer 간의 통합 구성을 참조하세요.

- 기존 Amazon SNS 주제를 대상으로 사용하려면 해당 주제에 대해 존재할 수 있는 다른 정책 외에 AWS CodeStar Notifications에 필요한 정책을 추가해야 합니다. 자세한 내용은 알림에 대한 Amazon SNS 주제 구성과 알림 내용 및 보안 이해를 참조하세요.

8. 규칙 생성을 완료하려면 제출을 선택합니다.
9. 사용자가 알림을 받으려면 먼저 규칙에 대한 Amazon SNS 주제를 사용자가 구독하도록 해야 합니다. 자세한 내용은 대상인 Amazon SNS 주제에 사용자 구독을 참조하세요. 채팅 애플리케이션에서 알림과 Amazon Q Developer 간의 통합을 설정하여 Amazon Chime 채팅룸으로 알림을 보낼 수도 있습니다. 자세한 내용은 채팅 애플리케이션에서 알림과 Amazon Q Developer 간의 통합 구성을 참조하세요.

알림 규칙을 생성하려면(AWS CLI)

1. 터미널 또는 명령 프롬프트에서 create-notification rule 명령을 실행하여 JSON 스켈레톤을 생성합니다.

```
aws codestar-notifications create-notification-rule --generate-cli-skeleton
 > rule.json
```

원하는 대로 파일 이름을 지정할 수 있습니다. 이 예에서는 *rule.json*으로 파일 이름을 지정합니다.

2. 일반 텍스트 편집기에서 JSON 파일을 열고 규칙에 대해 원하는 리소스, 이벤트 유형 및 대상을 포함하도록 편집합니다. 다음 예제는 ID가 *123456789012*인 계정의 *MyDemoRepo*라는 AWS 리포지토리에 **MyNotificationRule** 대한 라는 알림 규칙을 보여줍니다. 전체 세부 정보 유형의 알림은 브랜치 및 태그가 생성될 때 *MyNotificationTopic*이라는 Amazon SNS 주제로 전송됩니다.

```
{
    "Name": "MyNotificationRule",
    "EventTypeIds": [
        "codecommit-repository-branches-and-tags-created"
    ],
    "Resource": "arn:aws:codecommit:us-east-1:123456789012:MyDemoRepo",
    "Targets": [
```

```
        {
            "TargetType": "SNS",
            "TargetAddress": "arn:aws:sns:us-
east-1:123456789012:MyNotificationTopic"
        }
    ],
    "Status": "ENABLED",
    "DetailType": "FULL"
}
```

파일을 저장합니다.

3. 터미널 또는 명령줄에서 create-notification-rule 명령을 다시 실행하여 조금 전 편집한 파일을 사용해 알림 규칙을 생성합니다.

```
aws codestar-notifications create-notification-rule --cli-input-json
 file://rule.json
```

4. 성공한 경우 명령에서 다음과 유사한 알림 규칙의 ARN을 반환합니다.

```
{
    "Arn": "arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/
dc82df7a-EXAMPLE"
}
```

알림 변경 또는 비활성화

AWS CodeCommit 콘솔을 사용하여 사용자에게 이메일을 보내는 이벤트 유형과 리포지토리에 대한 이메일을 보내는 데 사용되는 Amazon SNS 주제를 포함하여 2019년 11월 5일 이전에 생성된 알림의 구성 방식을 변경할 수 있습니다. 또한 CodeCommit 콘솔을 사용하여 주제에 가입한 이메일 주소 및 엔드포인트의 목록을 관리하거나 알림을 비활성화할 수 있습니다.

알림 설정을 변경하려면

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서, 2019년 11월 5일 이전까지 생성된 알림을 구성하려는 리포지토리의 이름을 선택합니다.
3. 탐색 창에서 설정을 선택한 다음 알림을 선택합니다. 알림 규칙 대신에 알림이 있음을 알리는 배너가 표시되는 경우 기존 알림 관리를 선택합니다.

4. 편집을 선택합니다.
5. 내용을 변경하고 저장을 선택합니다.

알림 비활성화는 사용자가 일시적으로 리포지토리 이벤트에 대한 이메일을 수신할 수 없도록 하는 쉬운 방법입니다.

2019년 11월 5일 이전까지 생성된 알림을 영구적으로 삭제하려면 알림 삭제 단원의 단계를 따릅니다.

알림을 비활성화하려면

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서, 알림을 비활성화하려는 리포지토리의 이름을 선택합니다.
3. 탐색 창에서 설정을 선택한 다음 알림을 선택합니다. 기존 알림 관리를 선택합니다.
4. 편집을 선택하고 이벤트 상태에서 슬라이더를 사용하여 알림 활성화를 해제합니다. 저장을 선택합니다.
5. 이벤트 상태가 비활성화됨으로 변경됩니다. 이벤트에 대한 이메일이 전송되지 않습니다. 알림을 비활성화하면 리포지토리에 대한 CloudWatch Events 규칙도 자동으로 비활성화됩니다. CloudWatch Events 콘솔에서 상태를 수동으로 변경하지 마세요.

알림 삭제

2019년 11월 5일 이전까지 레포지토리에 생성된 알림을 더 이상 사용하지 않으려는 경우 알림과 연결된 Amazon CloudWatch Events 규칙을 삭제할 수 있습니다. 이렇게 하면 알림이 자동으로 삭제됩니다. 구독이나 알림에 사용된 Amazon SNS 주제는 삭제되지 않습니다.

> **Note**
>
> 콘솔에서 리포지토리의 이름을 변경하면 2019년 11월 5일 이전까지 생성된 알림이 수정하지 않아도 계속 작동합니다. 하지만 명령줄이나 API를 사용하여 리포지토리의 이름을 변경할 경우 알림이 더 이상 작동하지 않습니다. 알림을 복원하는 가장 쉬운 방법은 알림 설정을 삭제하고 나서 다시 구성하는 것입니다.

알림 설정을 삭제하려면

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.

2. 리포지토리에서, 2019년 11월 5일 이전까지 생성된 알림을 제거하려는 리포지토리의 이름을 선택합니다.
3. 탐색 창에서 설정을 선택한 다음 알림을 선택합니다. 알림 규칙 대신에 알림이 있음을 알리는 배너가 표시되는 경우 기존 알림 관리를 선택합니다.
4. CloudWatch 이벤트 규칙에서, 알림에 생성된 규칙의 이름을 복사합니다.
5. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/cloudwatch/ CloudWatch 콘솔을 엽니다.
6. 이벤트에서 규칙을 선택합니다. 이름에서, 알림에 생성된 규칙의 이름을 붙여 넣습니다. 규칙을 선택하고 작업에서 삭제를 선택합니다.
7. (선택 사항) 알림 설정을 삭제한 후 알림에 사용되는 Amazon SNS 주제를 변경하거나 삭제하려면, https://console.aws.amazon.com/sns/v3/home에서 Amazon SNS 콘솔로 이동합니다. 자세한 정보는 Amazon Simple Notification Service 개발자 안내서에서 정리를 참조하세요.

에서 리포지토리 태그 지정 AWS CodeCommit

태그는 사용자 또는가 AWS 리소스에 AWS 할당하는 사용자 지정 속성 레이블입니다. AWS 태그는 커밋에 적용할 수 있는 Git 태그와 다릅니다. 각 AWS 태그는 두 부분으로 구성됩니다.

- 태그 키(예: `CostCenter`, `Environment`, `Project` 또는 `Secret`). 태그 키는 대소문자를 구별합니다.
- 태그 값(예: `111122223333`, `Production` 또는 팀 이름)으로 알려진 선택적 필드. 태그 값을 생략하는 것은 빈 문자열을 사용하는 것과 같습니다. 태그 키처럼 태그 값은 대/소문자를 구별합니다.

태그 키와 태그 값을 합해서 키 값 페어라고 합니다. 리포지토리에 포함할 수 있는 태그 수 제한 및 태그 키 및 값에 대한 제한은 제한을 참조하세요.

태그는 AWS 리소스를 식별하고 구성하는 데 도움이 됩니다. 많은 AWS 서비스가 태그 지정을 지원하므로 서로 다른 서비스의 리소스에 동일한 태그를 할당하여 리소스가 관련이 있음을 나타낼 수 있습니다. 예를 들어 Amazon S3 버킷에 할당한 것과 동일한 태그를 CodeCommit 리포지토리에 할당할 수 있습니다. 태그 지정 전략에 대한 자세한 내용은 AWS 리소스 태그 지정을 참조하세요.

CodeCommit에서는 리포지토리가 기본 리소스입니다. CodeCommit 콘솔, AWS CLI, CodeCommit APIs 또는 AWS SDKs 사용하여 리포지토리에 대한 태그를 추가, 관리 및 제거할 수 있습니다. 태그로 리포지토리를 식별, 구성 및 추적하는 것 외에도 IAM 정책의 태그를 사용하여 리포지토리를 보고 상호작용할 수 있는 사용자를 제어할 수 있습니다. 태그 기반 액세스 정책의 예는 예제 5: 태그에 따라 리포지토리에 대한 작업 거부 또는 허용 단원을 참조하세요.

주제

- 리포지토리에 태그 추가
- 리포지토리에 대한 태그 보기
- 리포지토리에 대한 태그 편집
- 리포지토리에서 태그 제거

리포지토리에 태그 추가

리포지토리에 태그를 추가하면 AWS 리소스를 식별 및 구성하고 리소스에 대한 액세스를 관리하는 데 도움이 될 수 있습니다. 먼저 리포지토리에 하나 이상의 태그(키-값 쌍)를 추가합니다. 리포지토리에 태그 수에 대한 제한이 있음을 알아 두세요. 키 및 값 필드에서 사용할 수 있는 문자에 대한 제한이 있습니다. 자세한 설명은 제한을 참조하십시오. 태그가 생성된 후 해당 태그를 기준으로 리포지토리에 대한 액세스를 관리하는 IAM 정책을 생성할 수 있습니다. CodeCommit 콘솔 또는를 사용하여 리포지토리 AWS CLI 에 태그를 추가할 수 있습니다.

> **Important**
>
> 리포지토리에 태그를 추가하면 해당 리포지토리에 대한 액세스에 영향을 줄 수 있습니다. 리포지토리에 태그를 추가하기 전에 리포지토리와 같은 리소스에 대한 액세스를 제어하는 태그를 사용할 수도 있는 모든 IAM 정책을 검토합니다. 태그 기반 액세스 정책의 예는 예제 5: 태그에 따라 리포지토리에 대한 작업 거부 또는 허용 단원을 참조하세요.

리포지토리를 생성할 때 리포지토리에 태그를 추가하는 방법에 대한 자세한 내용은 리포지토리 생성 (콘솔) 단원을 참조하세요.

주제

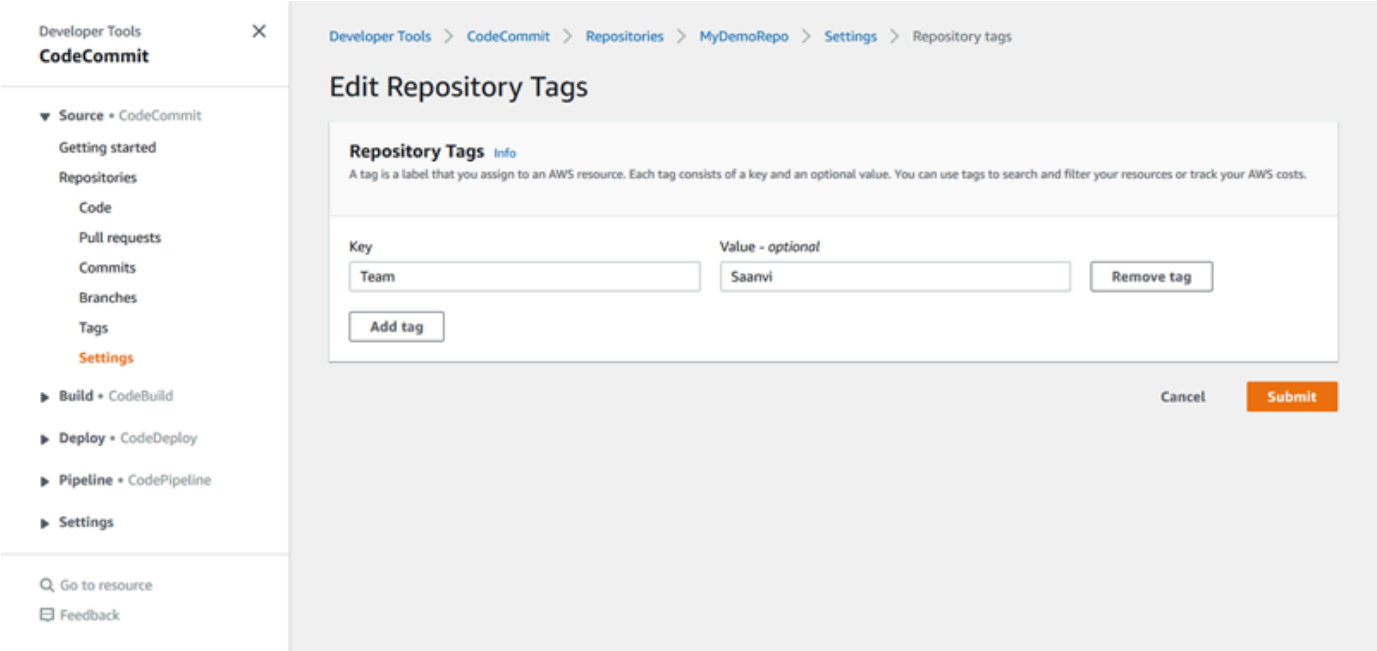
- 리포지토리에 태그 추가 (콘솔)
- 리포지토리에 태그 추가 (AWS CLI)

리포지토리에 태그 추가 (콘솔)

CodeCommit 콘솔을 사용하여 CodeCommit 리포지토리에 하나 이상의 태그를 추가할 수 있습니다.

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서 태그를 추가할 리포지토리의 이름을 선택합니다.

3. 탐색 창에서 설정을 선택합니다. 리포지토리 태그를 선택합니다.
4. 리포지토리에 추가된 태그가 없는 경우 태그 추가를 선택합니다. 또는 편집을 선택한 다음 태그 추가를 선택합니다.
5. 키에 태그 이름을 입력합니다. 값에 태그의 선택적 값을 추가할 수 있습니다.



6. (선택 사항) 다른 태그를 추가하려면 다시 태그 추가를 선택합니다.
7. 태그 추가를 마쳤으면 제출을 선택합니다.

리포지토리에 태그 추가 (AWS CLI)

다음 단계에 따라 AWS CLI 를 사용하여 CodeCommit 리포지토리에 태그를 추가합니다. 리포지토리를 생성할 때 태그를 추가하려면 리포지토리 생성 (AWS CLI) 단원을 참조하세요.

이 단계에서는 사용자가 이미 AWS CLI 최신 버전을 설치했거나 현재 버전으로 업데이트했다고 가정합니다. 자세한 정보는 AWS Command Line Interface설치 섹션을 참조하세요.

터미널이나 명령줄에서 tag-resource 명령을 실행하여, 태그를 추가할 리포지토리의 Amazon 리소스 이름(ARN)과 추가할 태그의 키와 값을 지정합니다. 리포지토리에 두 개 이상의 태그를 추가할 수 있습니다. 예를 들어, *MyDemoRepo*라는 리포지토리를 2개 태그로, *Status*라는 태그 키를 *Secret*의 태그 값으로, *Team*이라는 태그 키를 *Saanvi*의 태그 값으로 태그 지정하려면 다음을 수행합니다.

```
aws codecommit tag-resource --resource-arn arn:aws:codecommit:us-
west-2:111111111111:MyDemoRepo --tags Status=Secret,Team=Saanvi
```

성공한 경우 이 명령은 아무 것도 반환하지 않습니다.

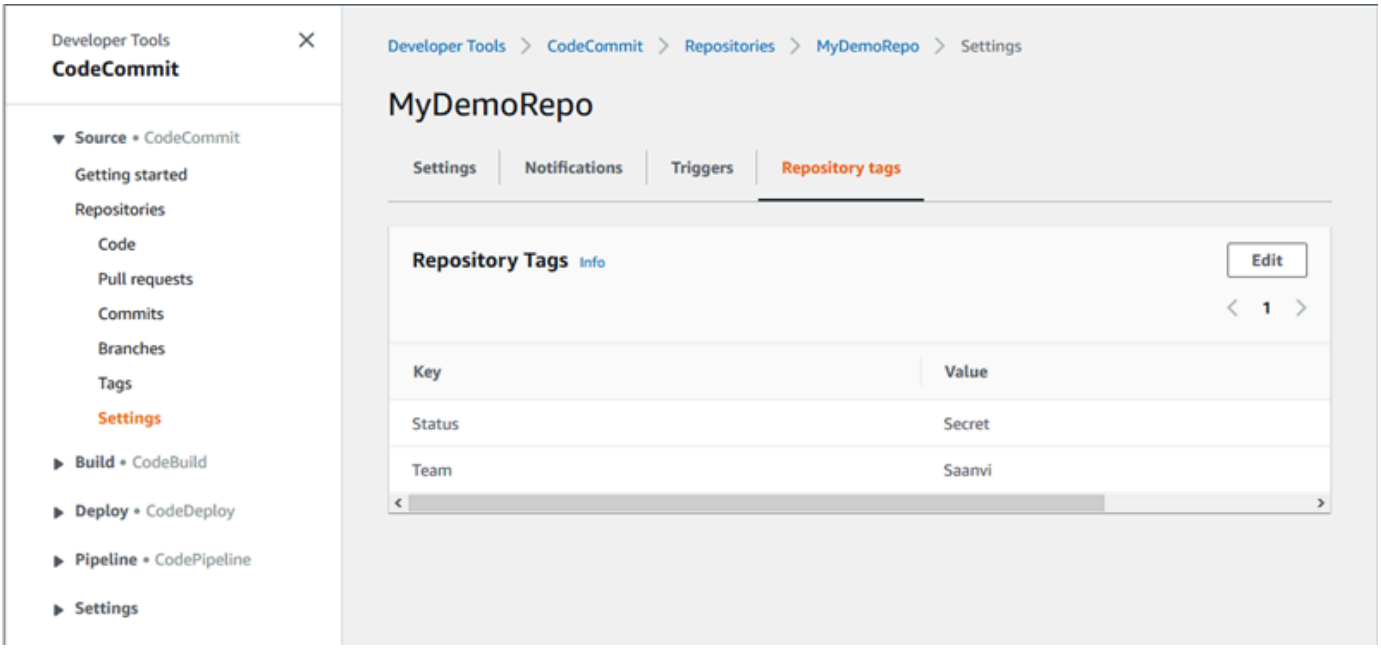
리포지토리에 대한 태그 보기

태그는 AWS 리소스를 식별 및 구성하고 리소스에 대한 액세스를 관리하는 데 도움이 될 수 있습니다. 태그 지정 전략에 대한 자세한 내용은 AWS 리소스 태그 지정을 참조하세요. 태그 기반 액세스 정책의 예는 예제 5: 태그에 따라 리포지토리에 대한 작업 거부 또는 허용 단원을 참조하세요.

리포지토리에 대한 태그 보기 (콘솔)

CodeCommit 콘솔을 사용하여 CodeCommit 리포지토리와 연결된 태그를 볼 수 있습니다.

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서, 태그를 보려는 리포지토리의 이름을 선택합니다.
3. 탐색 창에서 설정을 선택합니다. 리포지토리 태그를 선택합니다.



리포지토리에 대한 태그 보기 (AWS CLI)

다음 단계에 따라 AWS CLI 를 사용하여 CodeCommit 리포지토리의 AWS 태그를 확인합니다. 태그가 추가되지 않은 경우 반환되는 목록은 비어 있습니다.

터미널 또는 명령줄에서 list-tags-for-resource 명령을 실행합니다. 예를 들어, ARN이 `arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo`인 `MyDemoRepo`라는 리포지토리의 태그 키 및 태그 값에 대한 목록을 보려면 다음과 같이 합니다.

```
aws codecommit list-tags-for-resource --resource-arn arn:aws:codecommit:us-
west-2:111111111111:MyDemoRepo
```

이 명령이 제대로 실행되면 다음과 비슷한 정보를 반환합니다.

```
{
    "tags": {
        "Status": "Secret",
        "Team": "Saanvi"
    }
}
```

리포지토리에 대한 태그 편집

리포지토리와 연결된 태그에 대한 값을 변경할 수 있습니다. 또한 키 이름을 변경할 수 있습니다. 이는 현재 태그를 제거하고 새 이름 및 다른 키와 동일한 값을 가진 다른 태그를 추가하는 것과 동일합니다. 키 및 값 필드에서 사용할 수 있는 문자에 대한 제한이 있음을 알아 두세요. 자세한 설명은 제한을 참조하십시오.

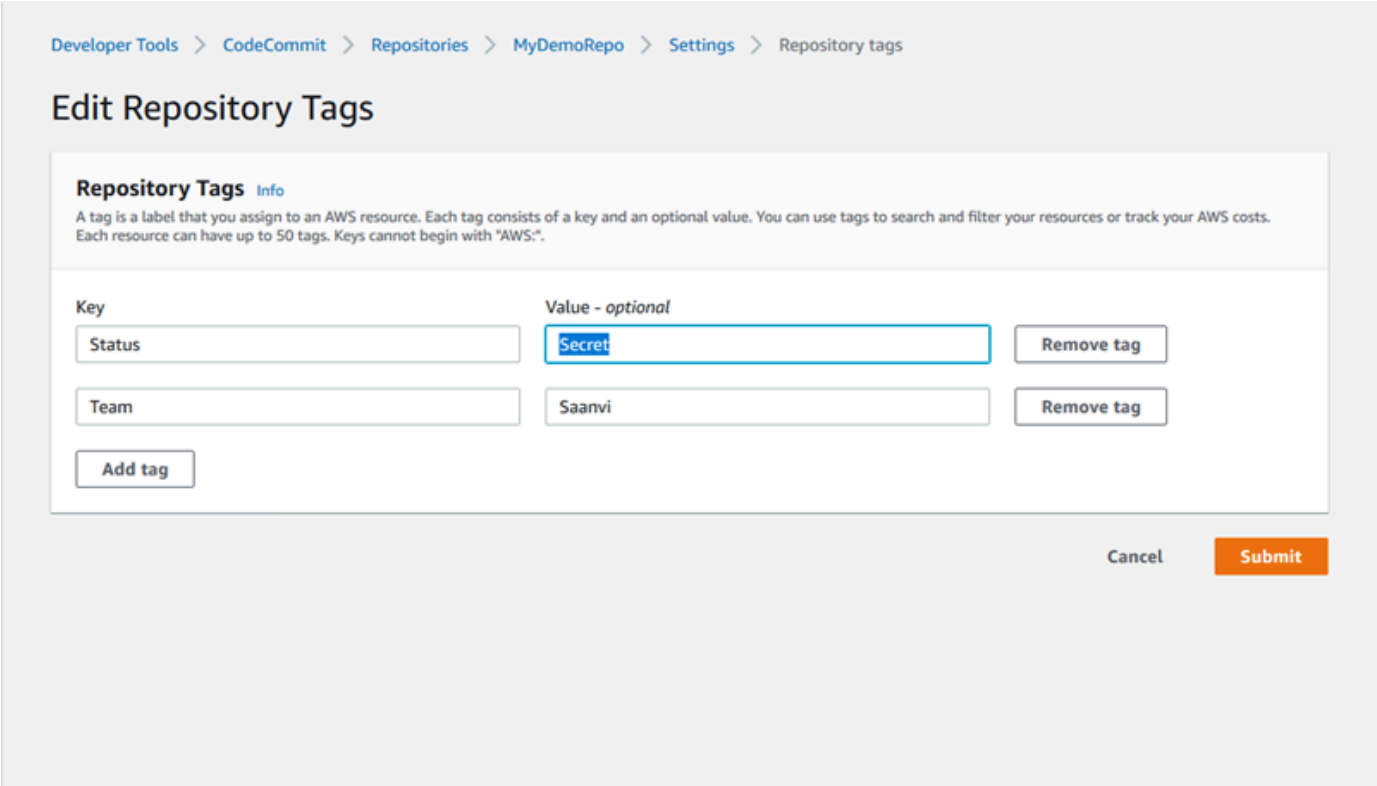
> **Important**
>
> 리포지토리에 대한 태그를 편집하면 해당 리포지토리에 대한 액세스에 영향을 줄 수 있습니다. 리포지토리에 대한 태그의 이름(키) 또는 값을 편집하기 전에 리포지토리와 같은 리소스에 대한 액세스를 제어하는 태그의 키 또는 값을 사용할 수도 있는 모든 IAM 정책을 검토합니다. 태그 기반 액세스 정책의 예는 예제 5: 태그에 따라 리포지토리에 대한 작업 거부 또는 허용 단원을 참조하세요.

리포지토리에 대한 태그 편집 (콘솔)

CodeCommit 콘솔을 사용하여 CodeCommit 리포지토리와 연결된 태그를 편집할 수 있습니다.

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서 태그를 편집할 리포지토리의 이름을 선택합니다.
3. 탐색 창에서 설정을 선택합니다. 리포지토리 태그를 선택합니다.
4. 편집을 선택합니다.

5.



다음 중 하나를 수행합니다.

- 태그를 변경하려면 키에 새 이름을 입력합니다. 태그 이름을 변경하는 것은 태그를 제거하고 새 키 이름의 새 태그를 추가하는 것과 동일합니다.
- 태그 값을 변경하려면 새 값을 입력합니다. 값을 어떤 것으로도 변경하지 않으려면 현재 값을 삭제하고 필드를 비워둡니다.

6. 태그 편집을 마쳤으면 제출을 선택합니다.

리포지토리에 대한 태그 편집 (AWS CLI)

다음 단계에 따라 AWS CLI 를 사용하여 CodeCommit 리포지토리의 태그를 업데이트합니다. 기존 키의 값을 변경하거나 다른 키를 추가할 수 있습니다.

터미널이나 명령줄에서 tag-resource 명령을 실행하여, 태그를 업데이트하고 태그 키 및 태그 값을 지정할 리포지토리의 Amazon 리소스 이름(ARN)을 지정합니다.

```
aws codecommit tag-resource --resource-arn arn:aws:codecommit:us-
west-2:111111111111:MyDemoRepo --tags Team=Li
```

리포지토리에서 태그 제거

리포지토리와 연결된 태그를 하나 이상 제거할 수 있습니다. 태그를 제거해도 해당 태그와 연결된 다른 AWS 리소스에서 태그는 삭제되지 않습니다.

> **Important**
>
> 리포지토리에 대한 태그를 제거하면 해당 리포지토리에 대한 액세스에 영향을 줄 수 있습니다. 리포지토리에서 태그를 제거하기 전에 리포지토리와 같은 리소스에 대한 액세스를 제어하는 태그의 키 또는 값을 사용할 수도 있는 모든 IAM 정책을 검토합니다. 태그 기반 액세스 정책의 예는 예제 5: 태그에 따라 리포지토리에 대한 작업 거부 또는 허용 단원을 참조하세요.

리포지토리에서 태그 제거 (콘솔)

CodeCommit 콘솔을 사용하여 태그와 CodeCommit 리포지토리 간의 연결을 제거할 수 있습니다.

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서 태그를 제거할 리포지토리의 이름을 선택합니다.
3. 탐색 창에서 설정을 선택합니다. 리포지토리 태그를 선택합니다.
4. 편집을 선택합니다.
5. 제거할 태그를 찾은 다음 태그 제거를 선택합니다.
6. 태그 제거를 마쳤으면 제출을 선택합니다.

리포지토리에서 태그 제거 (AWS CLI)

다음 단계에 따라 AWS CLI 를 사용하여 CodeCommit 리포지토리에서 태그를 제거합니다. 태그를 제거하면 태그는 삭제되지 않고 태그와 리포지토리 간의 연결만 제거됩니다.

> **Note**
>
> CodeCommit 리포지토리를 삭제하면 삭제된 리포지토리에서 모든 태그 연결이 제거됩니다. 리포지토리를 삭제하기 전에 태그를 제거하지 않아도 됩니다.

터미널이나 명령줄에서 untag-resource 명령을 실행하여, 태그를 제거할 리포지토리의 Amazon 리소스 이름(ARN)과 제거할 태그의 태그 키를 지정합니다. *## ##, MyDemoRepo## ###### ## ## Status*인 태그를 제거하려면 다음과 같이 합니다.

```
aws codecommit untag-resource --resource-arn arn:aws:codecommit:us-
west-2:111111111111:MyDemoRepo --tag-keys Status
```

성공한 경우 이 명령은 아무 것도 반환하지 않습니다. 리포지토리와 연결된 태그를 확인하려면 list-tags-for-resource 명령을 실행합니다.

AWS CodeCommit 리포지토리에 대한 트리거 관리

코드 푸시 또는 기타 이벤트가 Amazon Simple Notification Service(SNS)의 알림 전송이나 AWS Lambda의 함수 간접 호출과 같은 작업을 트리거하도록 CodeCommit 리포지토리를 구성할 수 있습니다. 각 CodeCommit 리포지토리에 트리거를 최대 10개까지 만들 수 있습니다.

트리거는 일반적으로 다음과 같이 구성됩니다.

- 누군가 리포지토리로 푸시할 때마다 구독한 사용자들에게 이메일을 보냅니다.
- 누군가 리포지토리의 main 브랜치로 푸시하면 외부 빌드 시스템에 빌드를 시작하도록 알립니다.

외부 빌드 시스템에 알림을 보내는 것과 같은 시나리오에서는 다른 애플리케이션과 상호 작용하기 위해 Lambda 함수를 작성해야 합니다. 이메일 시나리오에서는 Amazon SNS 주제를 생성하기만 하면 됩니다.

이 주제에서는 CodeCommit이 Amazon SNS 및 Lambda에서 작업을 트리거할 수 있도록 권한을 설정하는 방법에 대해 설명합니다. 이 주제에는 또한 트리거 생성, 편집, 테스트 및 삭제에 대한 예제로 연결되는 링크도 포함되어 있습니다.

주제

- 리소스 생성 및 CodeCommit에 대한 권한 추가
- 예: Amazon SNS 주제에 대한 AWS CodeCommit 트리거 생성
- 예: AWS Lambda 함수에 대한 AWS CodeCommit 트리거 생성
- 예:에서 기존 AWS Lambda 함수에 AWS CodeCommit 대한 트리거 생성
- 리포지 AWS CodeCommit 토리에 대한 트리거 편집
- 리포지 AWS CodeCommit 토리에 대한 트리거 테스트

- AWS CodeCommit 리포지토리에서 트리거 삭제

리소스 생성 및 CodeCommit에 대한 권한 추가

CodeCommit에서는 Amazon SNS 주제 및 Lambda 함수를 트리거와 통합할 수 있지만, 먼저 CodeCommit에 해당 리소스와 상호 작용할 권한을 부여하는 정책을 사용하여 리소스를 생성하고 구성해야 합니다. CodeCommit 리포지토리 AWS 리전 와 동일한에서 리소스를 생성해야 합니다. 예를 들어, 리포지토리가 미국 동부 (오하이오)(us-east-2)에 있는 경우, Amazon SNS 주제나 Lambda 함수 역시 미국 동부(오하이오)에 있어야 합니다.

- Amazon SNS 주제의 경우, CodeCommit 리포지토리와 동일한 계정을 사용하여 Amazon SNS 주제를 생성한다면 추가 IAM 정책 또는 권한을 구성할 필요가 없습니다. 사용자는 Amazon SNS 주제를 생성하고 구독하는 즉시 CodeCommit 트리거를 생성할 수 있습니다.
  - Amazon SNS에서 주제를 생성하는 방법에 대해 자세히 알아보려면 Amazon SNS 설명서를 참조하세요.
  - Amazon SNS를 사용하여 Amazon SQS 대기열에 메시지를 보내는 방법에 대해 자세히 알아보려면 Amazon SNS 개발자 안내서에서 Amazon SQS 대기열로 Amazon SNS 메시지 전송을 참조하세요.
  - Amazon SNS 사용하여 Lambda 함수를 간접 호출하는 방법에 대한 자세한 내용은 Amazon SNS 개발자 안내서에서 Amazon SNS 알림을 사용하여 Lambda 함수 호출을 참조하세요.
- 다른 AWS 계정에서 Amazon SNS 주제를 사용하도록 트리거를 구성하려면, CodeCommit이 해당 주제를 게시하도록 허용하는 정책으로 해당 주제를 구성해야 합니다. 자세한 내용은 예제 1: Amazon SNS 주제에 대한 크로스 계정 액세스 권한을 허용하는 정책 생성 단원을 참조하십시오.
- Lambda 콘솔에서 트리거를 함수의 일부로 생성하여 Lambda 함수를 구성할 수 있습니다. 이 방법이 가장 간단한 이유는 Lambda 콘솔에서 생성된 트리거에는 CodeCommit에서 Lambda 함수를 간접 호출하는 데 필요한 권한이 자동으로 포함되기 때문입니다. CodeCommit에서 트리거를 생성하는 경우에는 CodeCommit에서 함수를 간접 호출하도록 허용하는 정책을 포함해야 합니다. 자세한 내용은 기존 Lambda 함수를 위한 트리거 생성 및 예제 3: CodeCommit 트리거와 AWS Lambda 통합하기 위한 정책 생성 섹션을 참조하세요.

예: Amazon SNS 주제에 대한 AWS CodeCommit 트리거 생성

사용자는 CodeCommit 리포지토리의 이벤트가 Amazon Simple Notification Service(SNS) 주제의 알림을 트리거하도록 CodeCommit 리포지토리의 트리거를 생성할 수 있습니다. Amazon SNS 주제에 대한 트리거를 생성하여 사용자가 브랜치 삭제와 같은 리포지토리 이벤트에 대한 알림을 구독할 수 있도

록 하는 것이 좋을 수도 있습니다. Amazon Simple Queue Service(Amazon SQS) 및와 같은 다른 서비스와 Amazon SNS 주제의 통합을 활용할 수도 있습니다 AWS Lambda.Amazon SQS

> **Note**
>
> - 트리거는 기존 Amazon SNS 주제, 즉 리포지토리 이벤트에 대한 응답으로 취해진 조치를 가리켜야 합니다. Amazon SNS 주제 생성 및 구독에 대한 자세한 내용은 Amazon Simple Notification Service 시작하기를 참조하세요.
> - Amazon SNS FIFO(선입선출) 주제는 CodeCommit 트리거에 대해 지원되지 않습니다.

주제

- CodeCommit 리포지토리에 Amazon SNS 주제를 위한 트리거 생성 (콘솔)
- CodeCommit 리포지토리에 Amazon SNS 주제를 위한 트리거 생성 (AWS CLI)

CodeCommit 리포지토리에 Amazon SNS 주제를 위한 트리거 생성 (콘솔)

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서, 리포지토리 이벤트 트리거를 생성하려는 리포지토리를 선택합니다.
3. 리포지토리의 탐색 창에서 설정을 선택한 후 트리거를 선택합니다.
4. 정책 생성을 선택한 다음 다음을 수행합니다.

   - 트리거 이름에서 트리거 이름(예: `MyFirstTrigger`)을 입력합니다.
   - 이벤트에서, Amazon SNS 주제가 알림을 전송하도록 트리거할 리포지토리 이벤트를 선택합니다.

     모든 리포지토리 이벤트를 선택하면, 다른 이벤트는 선택할 수 없습니다. 이벤트의 하위 집합을 선택하려면 모든 리포지토리 이벤트를 제거한 다음 목록에서 하나 이상의 이벤트를 선택합니다. 예를 들어, 사용자가 CodeCommit 리포지토리에 브랜치나 태그를 생성할 때만 해당 트리거가 실행되도록 하려면, 모든 리포지토리 이벤트를 제거한 다음 브랜치 또는 태그 생성을 선택합니다.
   - 트리거를 리포지토리의 모든 브랜치에 적용하려면, 브랜치에서 선택 항목을 비워 둡니다. 이 기본 옵션으로 트리거를 모든 브랜치에 자동으로 적용할 수 있기 때문입니다. 이 트리거를 특정 브랜치에만 적용하려면 리포지토리 브랜치 목록에서 브랜치 이름을 최대 10개까지 선택합니다.
   - 사용할 서비스 선택에서 Amazon SNS를 선택합니다.

- Amazon SNS에서는 목록의 주제 이름을 선택하거나 주제에 대한 ARN을 입력합니다.

> **Note**
>
> Amazon SNS FIFO(선입선출) 주제는 CodeCommit 트리거에 대해 지원되지 않습니다. 유형이 표준으로 설정된 Amazon SNS 주제를 선택해야 합니다.

- 사용자 지정 데이터에서, Amazon SNS 주제에서 전송하는 알림에 포함하려는 선택적 정보(예: 개발자가 이 리포지토리에서의 개발을 논의할 때 사용하는 IRC 채널 이름)를 입력합니다. 이 필드는 문자열입니다. 동적 파라미터를 전달하는 데 사용할 수 없습니다.

5. (선택 사항) 트리거 테스트를 선택합니다. 이 단계를 통해 CodeCommit과 Amazon SNS 주제 간의 액세스 구성이 올바른지 확인할 수 있습니다. 이 단계에서는 Amazon SNS 주제를 사용하여 리포지토리의 데이터(있는 경우)로 테스트 알림을 보냅니다. 실제 데이터가 없는 경우에는 테스트 알림에 샘플 데이터가 포함됩니다.
6. 트리거 생성을 선택하여 트리거 생성하기를 완료합니다.

CodeCommit 리포지토리에 Amazon SNS 주제를 위한 트리거 생성 (AWS CLI)

사용자는 명령줄을 사용하여, CodeCommit 리포지토리의 이벤트(예: 리포지토리에 커밋을 푸시하는 경우)에 응답해 Amazon SNS 주제를 위한 트리거를 생성할 수도 있습니다.

Amazon SNS 주제에 대한 트리거를 생성하려면

1. 일반 텍스트 편집기를 열고 다음을 지정하는 JSON 파일을 생성합니다.

   - Amazon SNS 주제 이름.

   > **Note**
   >
   > Amazon SNS FIFO(선입선출) 주제는 CodeCommit 트리거에 대해 지원되지 않습니다. 유형이 표준으로 설정된 Amazon SNS 주제를 선택해야 합니다.

   - 이 트리거로 모니터링하려는 리포지토리와 브랜치. (브랜치를 지정하지 않는 경우 트리거가 리포지토리의 모든 브랜치에 적용됩니다.)
   - 이 트리거를 활성화하는 이벤트.

파일을 저장합니다.

예를 들어, *MyDemoRepo*라는 리포지토리의 *main*과 *preprod*라는 두 브랜치에서 발생하는 모든 리포지토리 이벤트를 *MySNSTopic*이라는 Amazon SNS 주제에 게시하는 트리거를 생성하려면 다음과 같이 합니다.

```
{
    "repositoryName": "MyDemoRepo",
    "triggers": [
        {
            "name": "MyFirstTrigger",
            "destinationArn": "arn:aws:sns:us-east-2:111122223333:MySNSTopic",
            "customData": "",
            "branches": [
                "main", "preprod"
            ],
            "events": [
                "all"
            ]
        }
    ]
}
```

JSON에는 리포지토리의 각 트리거에 대한 트리거 블록이 있어야 합니다. 리포지토리에 대한 트리거를 두 개 이상 만들려면 JSON에 두 개 이상의 트리거 블록을 포함합니다. 이 파일에 만든 모든 트리거는 지정된 리포지토리를 위한 것임을 기억하세요. 단일 JSON 파일에는 다수의 리포지토리에 대한 트리거를 생성할 수 없습니다. 예를 들어, 리포지토리에 대한 트리거를 두 개 생성하려는 경우 두 개의 트리거 블록이 있는 JSON 파일을 만들 수 있습니다. 다음 예제에서는 두 번째 트리거에 지정된 브랜치가 없으므로, 해당 트리거는 모든 브랜치에 적용됩니다.

```
{
    "repositoryName": "MyDemoRepo",
    "triggers": [
        {
            "name": "MyFirstTrigger",
            "destinationArn": "arn:aws:sns:us-east-2:111122223333:MySNSTopic",
            "customData": "",
            "branches": [
                "main", "preprod"
```

```
            ],
            "events": [
                "all"
            ]
        },
        {
            "name": "MySecondTrigger",
            "destinationArn": "arn:aws:sns:us-east-2:111122223333:MySNSTopic2",
            "customData": "",
            "branches": [],
            "events": [
                "updateReference", "deleteReference"
            ]
        }
    ]
}
```

커밋이 리포지토리에 푸시되는 경우와 같이 지정한 이벤트에 대해 트리거를 만들 수 있습니다. 이벤트 유형에는 다음이 포함됩니다.

- `all` - 지정된 리포지토리 및 브랜치의 모든 이벤트.
- `updateReference` - 커밋이 지정된 리포지토리 및 브랜치로 푸시되는 경우.
- `createReference` - 새 브랜치 또는 태그가 지정된 리포지토리에 생성되는 경우.
- `deleteReference` - 브랜치 또는 태그가 지정된 리포지토리에서 삭제되는 경우.

> **Note**
>
> 트리거에는 이벤트 유형을 두 개 이상 사용할 수 있습니다. 하지만 `all`을 지정하면 다른 이벤트를 지정할 수 없습니다.

유효한 이벤트 유형의 전체 목록을 보려면 터미널 또는 명령 프롬프트에서 aws codecommit put-repository-triggers help를 입력합니다.

또한 `customData`에 문자열을 포함할 수 있습니다(예: 개발자가 이 리포지토리에서의 개발을 논의할 때 사용하는 IRC 채널 이름). 이 필드는 문자열입니다. 동적 파라미터를 전달하는 데 사용할 수 없습니다. 이 문자열은 트리거에 대한 응답으로 반환되는 CodeCommit JSON에 속성으로 추가됩니다.

2. (선택 사항) 터미널 또는 명령 프롬프트에서 test-repository-triggers 명령을 실행합니다. 이 테스트에서는 리포지토리의 샘플 데이터를 사용하여(또는 사용 가능한 데이터가 없는 경우 샘플 데이터를 생성) Amazon SNS 주제 구독자에게 알림을 보냅니다. 예를 들어, 다음은 *trigger.json*이라는 JSON 파일이 유효하고 CodeCommit이 Amazon SNS 주제에 게시를 할 수 있는지 테스트하는 데 사용됩니다.

```
aws codecommit test-repository-triggers --cli-input-json file://trigger.json
```

이 명령이 제대로 실행되면 다음과 비슷한 정보를 반환합니다.

```
{
    "successfulExecutions": [
        "MyFirstTrigger"
    ],
    "failedExecutions": []
}
```

3. 터미널 또는 명령 프롬프트에서 put-repository-triggers 명령을 실행하여 CodeCommit에서 트리거를 생성합니다. 예를 들어 *trigger.json*이라는 JSON 파일을 사용하여 트리거를 생성하려면 다음과 같이 합니다.

```
aws codecommit put-repository-triggers --cli-input-json
file://trigger.json
```

이 명령은 다음과 유사한 구성 ID를 반환합니다.

```
{
    "configurationId": "0123456-I-AM-AN-EXAMPLE"
}
```

4. 트리거의 구성을 보려면, 리포지토리 이름을 지정하여 get-repository-triggers 명령을 실행합니다.

```
aws codecommit get-repository-triggers --repository-name MyDemoRepo
```

이 명령은 리포지토리에 대해 구성된 모든 트리거의 다음과 비슷한 구조를 반환합니다.

```
{
    "configurationId": "0123456-I-AM-AN-EXAMPLE",
    "triggers": [
        {
            "events": [
```

```
                "all"
            ],
            "destinationArn": "arn:aws:sns:us-east-2:111122223333:MySNSTopic",
            "branches": [
                "main",
                "preprod"
            ],
            "name": "MyFirstTrigger",
            "customData": "Project ID 12345"
        }
    ]
}
```

5. 트리거의 기능을 좀 더 테스트하려면, 커밋을 하나 생성한 다음 트리거를 구성한 리포지토리에 푸시합니다. 그러면 Amazon SNS 주제에서 응답을 볼 수 있습니다. 예를 들어, 이메일을 보내도록 Amazon SNS 주제를 구성한 경우에는 해당 주제를 구독하는 이메일 계정에서 Amazon SNS의 이메일을 볼 수 있습니다.

   다음은 CodeCommit 리포지토리로의 푸시에 대한 응답으로 Amazon SNS에서 보낸 이메일을 출력한 예시입니다.

```
{
  "Records":[
     {
        "awsRegion":"us-east-2",
        "codecommit":{
           "references" : [
              {
                    "commit":"317f8570EXAMPLE",
                    "created":true,
                    "ref":"refs/heads/NewBranch"
              },
              {
                    "commit":"4c925148EXAMPLE",
                    "ref":"refs/heads/preprod",
              }
            ]
          },
        "eventId":"11111-EXAMPLE-ID",
        "eventName":"ReferenceChange",
        "eventPartNumber":1,
        "eventSource":"aws:codecommit",
        "eventSourceARN":"arn:aws:codecommit:us-east-2:111122223333:MyDemoRepo",
```

```
        "eventTime":"2016-02-09T00:08:11.743+0000",
        "eventTotalParts":1,
        "eventTriggerConfigId":"0123456-I-AM-AN-EXAMPLE",
        "eventTriggerName":"MyFirstTrigger",
        "eventVersion":"1.0",
        "customData":"Project ID 12345",
        "userIdentityARN":"arn:aws:iam::111122223333:user/JaneDoe-CodeCommit",
    }
  ]
}
```

예: AWS Lambda 함수에 대한 AWS CodeCommit 트리거 생성

CodeCommit 리포지토리에 적용되는 트리거를 생성하여 리포지토리의 해당 이벤트가 Lambda 함수를 간접 호출하게 할 수 있습니다 이 예제에서는 해당 리포지토리를 Amazon CloudWatch 로그로 복제하는 데 사용할 URL을 반환하는 Lambda 함수를 생성합니다.

주제

- Lambda 함수 생성
- AWS CodeCommit 리포지토리에서 Lambda 함수에 대한 트리거 보기

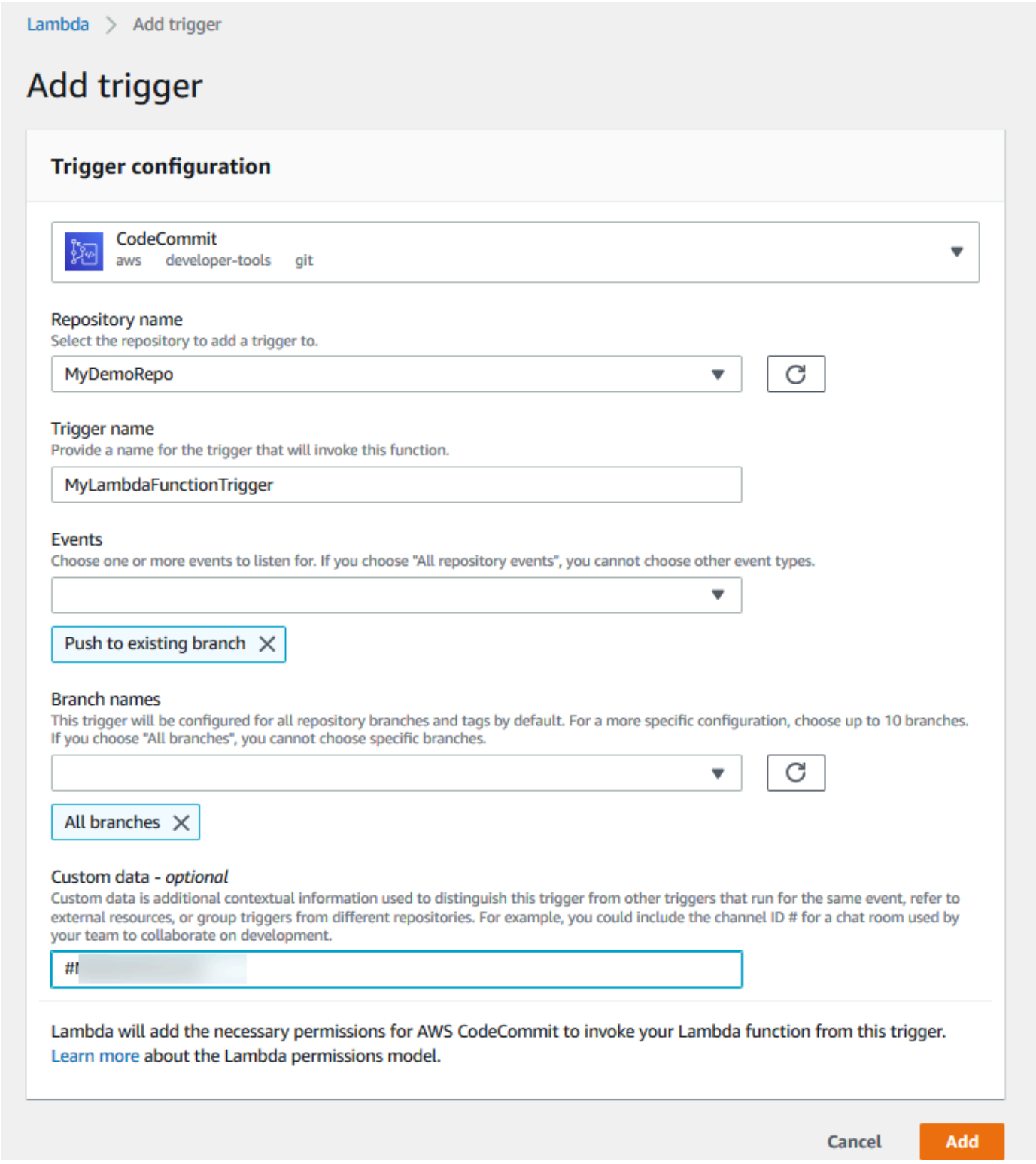
Lambda 함수 생성

Lambda 콘솔을 사용하여 함수를 생성하는 경우에는 Lambda 함수에 대한 CodeCommit 트리거 또한 생성할 수 있습니다. 다음 절차에는 샘플 Lambda 함수가 포함되어 있습니다. 이 샘플은 JavaScript 및 Python 두 언어 버전으로 제공됩니다. 이 함수는 리포지토리를 CloudWatch 로그로 복제하는 데 사용할 URL을 반환합니다.

Lambda 청사진을 사용하여 Lambda 함수를 생성하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/lambda/ AWS Lambda 콘솔을 엽니다.
2. Lambda 함수 페이지에서 함수 생성을 선택합니다. (이전에 Lambda를 사용해본 적이 없다면 지금 시작을 선택합니다.)
3. 함수 생성 페이지에서 처음부터 새로 작성을 선택합니다. 함수 이름에 함수 이름을 입력합니다(예: *MyLambdaFunctionforCodeCommit*). 런타임에서 함수를 작성하는 데 사용할 언어를 선택한 다음 함수 생성을 선택합니다.

4. 구성 탭에서 트리거 추가를 선택합니다.
5. 트리거 구성의 서비스 드롭다운 목록에서 CodeCommit을 선택합니다.



- 리포지토리 이름에서, 리포지토리 이벤트에 반응하여 Lambda 함수를 사용하는 트리거를 구성하게 될 리포지토리의 이름을 선택합니다.
- 트리거 이름에 해당 트리거의 이름을 입력합니다(예: *MyLambdaFunctionTrigger*).
- 이벤트에서 Lambda 함수를 트리거할 리포지토리 이벤트를 선택합니다. 모든 리포지토리 이벤트를 선택하면, 다른 이벤트는 선택할 수 없습니다. 이벤트의 하위 세트를 선택하려면 모든 리포지토리 이벤트를 취소한 후 목록에서 원하는 이벤트를 선택하면 됩니다. 예를 들어 사용자가 리포지 AWS CodeCommit 토리에서 태그 또는 브랜치를 생성할 때만 트리거를 실행하려면 모든 리포지토리 이벤트를 제거한 다음 브랜치 또는 태그 생성을 선택합니다.
- 해당 트리거가 리포지토리의 모든 브랜치에 적용되도록 하고 싶다면 브랜치에서 모든 브랜치를 선택하면 됩니다. 이를 원하지 않은 경우 특정 브랜치를 선택합니다. 리포지토리의 기본 브랜치는 기본적으로 추가됩니다. 이 기본 브랜치는 그대로 유지하거나 목록에서 삭제할 수 있습니다. 리포지토리 브랜치 목록에서 최대 10개의 브랜치 이름을 선택합니다.
- (선택 사항) 사용자 지정 데이터에서 Lambda 함수에 포함하려는 정보를 입력합니다(예: 리포지토리에서의 개발을 논의하기 위해 개발자들이 사용하는 IRC 채널의 이름). 이 필드는 문자열입니다. 동적 파라미터를 전달하는 데 사용할 수 없습니다.

추가를 선택합니다.

6. 구성 페이지의 함수 코드에 있는 코드 입력 유형에서 코드 인라인 편집을 선택합니다. 런타임에서 Node.js를 선택합니다. 샘플 Python 함수를 생성하려면 Python을 선택합니다.
7. 코드 항목 유형에서 코드 인라인 편집을 선택한 후 hello world 코드를 다음 두 샘플 중 하나로 교체합니다.

   Node.js의 경우:

```
import {
  CodeCommitClient,
  GetRepositoryCommand,
} from "@aws-sdk/client-codecommit";

const codecommit = new CodeCommitClient({ region: "your-region" });

/**
 * @param {{ Records: { codecommit: { references: { ref: string }[] },
 eventSourceARN: string  }[]} event
 */
export const handler = async (event) => {
  // Log the updated references from the event
```

```
  const references = event.Records[0].codecommit.references.map(
    (reference) => reference.ref,
  );
  console.log("References:", references);

  // Get the repository from the event and show its git clone URL
  const repository = event.Records[0].eventSourceARN.split(":")[5];
  const params = {
    repositoryName: repository,
  };

  try {
    const data = await codecommit.send(new GetRepositoryCommand(params));
    console.log("Clone URL:", data.repositoryMetadata.cloneUrlHttp);
    return data.repositoryMetadata.cloneUrlHttp;
  } catch (error) {
    console.error("Error:", error);
    throw new Error(
      `Error getting repository metadata for repository ${repository}`,
    );
  }
};
```

Python의 경우:

```
import json
import boto3

codecommit = boto3.client("codecommit")


def lambda_handler(event, context):
    # Log the updated references from the event
    references = {
        reference["ref"]
        for reference in event["Records"][0]["codecommit"]["references"]
    }
    print("References: " + str(references))

    # Get the repository from the event and show its git clone URL
    repository = event["Records"][0]["eventSourceARN"].split(":")[5]
    try:
```

```
        response = codecommit.get_repository(repositoryName=repository)
        print("Clone URL: " + response["repositoryMetadata"]["cloneUrlHttp"])
        return response["repositoryMetadata"]["cloneUrlHttp"]
    except Exception as e:
        print(e)
        print(
            "Error getting repository {}. Make sure it exists and that your
repository is in the same region as this function.".format(
                repository
            )
        )
        raise e
```

8. 권한 탭의 실행 역할에서, IAM 콘솔에서 열 역할을 선택합니다. 트리거를 사용할 리포지토리에 대한 `GetRepository` 권한을 추가하도록 연결된 정책을 편집합니다.

AWS CodeCommit 리포지토리에서 Lambda 함수에 대한 트리거 보기

Lambda 함수를 생성했으면 AWS CodeCommit에서 해당 트리거를 확인하고 테스트할 수 있습니다. 트리거를 테스트하면 지정한 리포지토리 이벤트에 반응하여 함수가 실행됩니다.

Lambda 함수에 대한 트리거를 조회하고 테스트하려면

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서 트리거를 보려는 리포지토리를 선택합니다.
3. 리포지토리의 탐색 창에서 설정을 선택한 후 트리거를 선택합니다.
4. 리포지토리의 트리거 목록을 검토합니다. Lambda 콘솔에서 생성한 트리거가 보일 것입니다. 목록에서 트리거를 선택한 후 트리거 테스트를 선택합니다. 이 옵션을 선택하면 해당 리포지토리의 가장 최근 커밋 ID를 포함해 리포지토리에 대한 샘플 데이터로 함수 간접 호출을 시도합니다. (커밋 이력이 없으면 그 대신 0으로 이루어진 샘플 값이 생성됩니다.) 이렇게 하면 AWS CodeCommit 와 Lambda 함수 간에 액세스를 올바르게 구성했는지 확인할 수 있습니다.
5. 트리거의 기능을 좀 더 확인하려면 커밋을 만들어 트리거를 구성한 리포지토리에 푸시합니다. Lambda 콘솔의 해당 함수에 대한 모니터링 탭에 Lambda 함수의 응답이 표시되어야 합니다. 모니터링 탭에서 CloudWatch에서 로그 보기를 선택합니다. 새 탭에서 CloudWatch 콘솔이 열려 해당 함수에 관한 이벤트를 표시합니다. 커밋을 푸시한 시점에 상응하는 로그 스트림을 목록에서 선택합니다. 다음과 유사한 이벤트 데이터가 출력되어야 합니다.

```
START RequestId: 70afdc9a-EXAMPLE Version: $LATEST
2015-11-10T18:18:28.689Z 70afdc9a-EXAMPLE References: [ 'refs/heads/main' ]
2015-11-10T18:18:29.814Z 70afdc9a-EXAMPLE Clone URL: https://git-codecommit.us-
east-2.amazonaws.com/v1/repos/MyDemoRepo
END RequestId: 70afdc9a-EXAMPLE
REPORT RequestId: 70afdc9a-EXAMPLE Duration: 1126.87 ms Billed Duration: 1200 ms
 Memory Size: 128 MB Max Memory Used: 14 MB
```

예:에서 기존 AWS Lambda 함수에 AWS CodeCommit 대한 트리거 생성

Lambda 함수를 호출하는 트리거를 생성하는 가장 쉬운 방법은 Lambda 콘솔에서 해당 트리거를 생성하는 것입니다. 이 내장 통합 기능을 통해 CodeCommit은 함수를 실행하는 데 필요한 권한을 갖게 됩니다. 기존 Lambda 함수에 트리거를 추가하려면, Lambda 콘솔로 이동하여 해당 함수를 선택합니다. 해당 함수의 트리거 탭에서 트리거 추가의 단계를 따릅니다. 이 단계는 Lambda 함수 생성의 단계와 비슷합니다.

CodeCommit 리포지토리에 Lambda 함수를 위한 트리거를 생성할 수도 있습니다. 이렇게 하려면 간접 호출할 기존 Lambda 함수를 선택해야 합니다. 또한 CodeCommit에서 해당 함수를 실행하는 데 필요한 권한을 수동으로 구성해야 합니다.

주제

- CodeCommit이 Lambda 함수를 실행할 수 있도록 권한을 수동으로 구성합니다.
- CodeCommit 리포지토리에 Lambda 함수를 위한 트리거 생성 (콘솔)
- CodeCommit 리포지토리에 Lambda 함수를 위한 트리거 생성 (AWS CLI)

CodeCommit이 Lambda 함수를 실행할 수 있도록 권한을 수동으로 구성합니다.

CodeCommit에서 Lambda 함수를 호출하는 트리거를 생성하는 경우에는 CodeCommit에서 해당 Lambda 함수를 실행하도록 허용하는 권한을 수동으로 구성해야 합니다. 이러한 수동 구성을 피하려면, Lambda 콘솔에서 함수에 대한 트리거를 생성하는 것이 좋습니다.

CodeCommit이 Lambda 함수를 실행할 수 있게 하려면

1. 일반 텍스트 편집기를 열고 JSON 파일을 생성하되, 다음과 같이 Lambda 함수 이름, CodeCommit 리포지토리의 세부 정보, Lambda에 허용하려는 작업 등을 지정합니다.

```
{
```

```
    "FunctionName": "MyCodeCommitFunction",
    "StatementId": "1",
    "Action": "lambda:InvokeFunction",
    "Principal": "codecommit.amazonaws.com",
    "SourceArn": "arn:aws:codecommit:us-east-1:111122223333:MyDemoRepo",
    "SourceAccount": "111122223333"
}
```

2. 파일은 기억하기 쉬운 이름을 가진 JSON 파일로 저장합니다(예: *AllowAccessfromMyDemoRepo*.json).
3. 터미널(Linux, macOS, Windows) 또는 명령줄에서 방금 생성한 JSON 파일로 aws lambda add-permissions 명령을 실행하여 Lambda 함수와 연결된 리소스 정책에 권한을 추가합니다.

```
aws lambda add-permission --cli-input-json file://AllowAccessfromMyDemoRepo.json
```

이 명령은 다음과 비슷하게 방금 추가한 정책 명령문의 JSON을 반환합니다.

```
{
    "Statement": "{\"Condition\":{\"StringEquals\":{\"AWS:SourceAccount
\":\"111122223333\"},\"ArnLike\":{\"AWS:SourceArn\":
\"arn:aws:codecommit:us-east-1:111122223333:MyDemoRepo\"}},\"Action
\":[\"lambda:InvokeFunction\"],\"Resource\":\"arn:aws:lambda:us-
east-1:111122223333:function:MyCodeCommitFunction\",\"Effect\":\"Allow\",
\"Principal\":{\"Service\":\"codecommit.amazonaws.com\"},\"Sid\":\"1\"}"
}
```

Lambda 함수의 리소스 정책에 대해 자세히 알아보려면 AWS Lambda 사용 설명서에서 AddPermission 및 풀/푸시 이벤트 모델을 참조하세요.

4. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/ IAM 콘솔을 엽니다.
5. 대시보드 탐색 창에서 역할을 선택하고, 역할 목록에서 *lambda_basic_execution*을 선택합니다.
6. 역할에 대한 요약 페이지에서 권한 탭을 선택하고, 인라인 정책에서 역할 정책 생성을 선택합니다.
7. 권한 설정 페이지에서 정책 생성기를 선택한 후 선택을 선택합니다.
8. 권한 편집 페이지에서 다음을 수행합니다.
   - 효과에서 허용을 선택합니다.
   - AWS 서비스에서 AWS CodeCommit을 선택합니다.

- 작업에서 리포지토리를 선택합니다.
- Amazon 리소스 이름(ARN)에 리포지토리의 ARN을 입력합니다(예: `arn:aws:codecommit:us-east-1:111122223333:MyDemoRepo`).

설명문 추가를 선택하고 다음 단계를 선택합니다.

9. 정책 검토 페이지에서 정책 적용을 선택합니다.

   정책 명령문은 다음 예제와 비슷할 것입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt11111111",
            "Effect": "Allow",
            "Action": [
                "codecommit:GetRepository"
            ],
            "Resource": [
                "arn:aws:codecommit:us-east-1:111122223333:MyDemoRepo"
            ]
        }
    ]
}
```

CodeCommit 리포지토리에 Lambda 함수를 위한 트리거 생성 (콘솔)

Lambda 함수를 생성한 후에는 CodeCommit에, 사용자가 지정한 리포지토리 이벤트에 응답해 함수를 실행하는 트리거를 생성할 수 있습니다.

> **Note**
>
> 예제에 대한 트리거를 성공적으로 테스트하거나 실행하려면, 먼저 CodeCommit이 함수를 간접 호출하고 Lambda 함수가 리포지토리에 대한 정보를 가져오도록 허용하는 정책을 구성해야 합니다. 자세한 내용은 CodeCommit이 Lambda 함수를 실행할 수 있게 하려면 단원을 참조하십시오.

Lambda 함수를 위한 트리거 생성

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서, 리포지토리 이벤트 트리거를 생성하려는 리포지토리를 선택합니다.
3. 리포지토리의 탐색 창에서 설정을 선택한 후 트리거를 선택합니다.
4. 트리거 생성을 선택합니다.
5. 트리거 생성에서 다음과 같이 합니다.

   - 트리거 이름에 해당 트리거의 이름을 입력합니다(예: `MyLambdaFunctionTrigger`).
   - 이벤트에서 Lambda 함수를 트리거할 리포지토리 이벤트를 선택합니다.

     모든 리포지토리 이벤트를 선택하면, 다른 이벤트는 선택할 수 없습니다. 이벤트의 하위 세트를 선택하려면 모든 리포지토리 이벤트를 취소한 후 목록에서 원하는 이벤트를 선택하면 됩니다. 예를 들어, 사용자가 CodeCommit 리포지토리에 태그나 브랜치를 생성할 때만 해당 트리거가 실행되도록 하려면, 모든 리포지토리 이벤트를 제거한 다음 브랜치 또는 태그 생성을 선택합니다.
   - 트리거를 리포지토리의 모든 브랜치에 적용하려면, 브랜치에서 선택 항목을 비워 둡니다. 이 기본 옵션으로 트리거를 모든 브랜치에 자동으로 적용할 수 있기 때문입니다. 이 트리거를 특정 브랜치에만 적용하려면 리포지토리 브랜치 목록에서 브랜치 이름을 최대 10개까지 선택합니다.
   - 사용할 서비스 선택에서 AWS Lambda를 선택합니다.
   - Lambda 함수에서 목록의 함수 이름을 선택하거나 함수의 ARN을 입력합니다.
   - (선택 사항) 사용자 지정 데이터에서 Lambda 함수에 포함하려는 정보를 입력합니다(예: 리포지토리에서의 개발을 논의하기 위해 개발자들이 사용하는 IRC 채널의 이름). 이 필드는 문자열입니다. 동적 파라미터를 전달하는 데 사용할 수 없습니다.
6. (선택 사항) 트리거 테스트를 선택합니다. 이 옵션을 선택하면 해당 리포지토리의 가장 최근 커밋 ID를 포함해 리포지토리에 대한 샘플 데이터로 함수 간접 호출을 시도합니다. (커밋 이력이 없으면 그 대신 0으로 이루어진 샘플 값이 생성됩니다.) 이를 통해 CodeCommit과 Lambda 함수 간의 액세스가 올바르게 구성되었는지 확인할 수 있습니다.
7. 트리거 생성을 선택하여 트리거 생성하기를 완료합니다.
8. 트리거의 기능을 확인하려면 커밋을 만들어 트리거를 구성한 리포지토리에 푸시합니다. Lambda 콘솔의 해당 함수에 대한 모니터링 탭에 Lambda 함수의 응답이 표시되어야 합니다.

CodeCommit 리포지토리에 Lambda 함수를 위한 트리거 생성 (AWS CLI)

사용자는 명령줄을 사용하여, CodeCommit 리포지토리의 이벤트(예: 리포지토리에 커밋을 푸시하는 경우)에 응답해 Lambda 함수를 위한 트리거를 생성할 수도 있습니다.

Lambda 함수를 위한 트리거 생성

1. 일반 텍스트 편집기를 열고 다음을 지정하는 JSON 파일을 생성합니다.

   - Lambda 함수의 이름입니다.
   - 이 트리거로 모니터링하려는 리포지토리와 브랜치. (브랜치를 지정하지 않는 경우 트리거가 리포지토리의 모든 브랜치에 적용됩니다.)
   - 이 트리거를 활성화하는 이벤트.

   파일을 저장합니다.

   예를 들어, *MyDemoRepo*라는 리포지토리의 *main*과 *preprod*라는 두 브랜치에서 발생하는 모든 리포지토리 이벤트를 *MyCodeCommitFunction*이라는 Lambda 함수에 게시하는 트리거를 생성하려면 다음과 같이 합니다.

```
{
    "repositoryName": "MyDemoRepo",
    "triggers": [
        {
            "name": "MyLambdaFunctionTrigger",
            "destinationArn": "arn:aws:lambda:us-
east-1:111122223333:function:MyCodeCommitFunction",
            "customData": "",
            "branches": [
                "main", "preprod"
            ],
            "events": [
                "all"
            ]
        }
    ]
}
```

   JSON에는 리포지토리의 각 트리거에 대한 트리거 블록이 있어야 합니다. 리포지토리에 대한 트리거를 두 개 이상 만들려면 JSON에 추가 블록을 포함합니다. 이 파일에 만든 모든 트리거는 지정

된 리포지토리를 위한 것임을 기억하세요. 단일 JSON 파일에는 다수의 리포지토리에 대한 트리거를 생성할 수 없습니다. 예를 들어, 리포지토리에 대한 트리거를 두 개 생성하려는 경우 두 개의 트리거 블록이 있는 JSON 파일을 만들 수 있습니다. 다음 예제에서는 두 번째 트리거 블록에 지정된 브랜치가 없으므로, 해당 트리거는 모든 브랜치에 적용됩니다.

```
{
    "repositoryName": "MyDemoRepo",
    "triggers": [
        {
            "name": "MyLambdaFunctionTrigger",
            "destinationArn": "arn:aws:lambda:us-
east-1:111122223333:function:MyCodeCommitFunction",
            "customData": "",
            "branches": [
                "main", "preprod"
            ],
            "events": [
                "all"
            ]
        },
        {
            "name": "MyOtherLambdaFunctionTrigger",
            "destinationArn": "arn:aws:lambda:us-
east-1:111122223333:function:MyOtherCodeCommitFunction",
            "customData": "",
            "branches": [],
            "events": [
                "updateReference", "deleteReference"
            ]
        }
    ]
}
```

커밋이 리포지토리에 푸시되는 경우와 같이 지정한 이벤트에 대해 트리거를 만들 수 있습니다. 이벤트 유형에는 다음이 포함됩니다.

- `all` - 지정된 리포지토리 및 브랜치의 모든 이벤트.
- `updateReference` - 커밋이 지정된 리포지토리 및 브랜치로 푸시되는 경우.
- `createReference` - 새 브랜치 또는 태그가 지정된 리포지토리에 생성되는 경우.
- `deleteReference` - 브랜치 또는 태그가 지정된 리포지토리에서 삭제되는 경우.

> **Note**
>
> 트리거에는 이벤트 유형을 두 개 이상 사용할 수 있습니다. 하지만 `all`을 지정하면 다른 이벤트를 지정할 수 없습니다.

유효한 이벤트 유형의 전체 목록을 보려면 터미널 또는 명령 프롬프트에서 aws codecommit put-repository-triggers help를 입력합니다.

또한 `customData`에 문자열을 포함할 수 있습니다(예: 개발자가 이 리포지토리에서의 개발을 논의할 때 사용하는 IRC 채널 이름). 이 필드는 문자열입니다. 동적 파라미터를 전달하는 데 사용할 수 없습니다. 이 문자열은 트리거에 대한 응답으로 반환되는 CodeCommit JSON에 속성으로 추가됩니다.

2. (선택 사항) 터미널 또는 명령 프롬프트에서 test-repository-triggers 명령을 실행합니다. 예를 들어, 다음은 *`trigger.json`*이라는 JSON 파일이 유효하고 CodeCommit이 Lambda 함수를 트리거할 수 있는지 테스트하는 데 사용됩니다. 이 테스트에서는 실제 데이터가 없는 경우 샘플 데이터를 사용하여 함수를 트리거합니다.

```
aws codecommit test-repository-triggers --cli-input-json file://trigger.json
```

이 명령이 제대로 실행되면 다음과 비슷한 정보를 반환합니다.

```
{
    "successfulExecutions": [
        "MyLambdaFunctionTrigger"
    ],
    "failedExecutions": []
}
```

3. 터미널 또는 명령 프롬프트에서 put-repository-triggers 명령을 실행하여 CodeCommit에서 트리거를 생성합니다. 예를 들어 *`trigger.json`*이라는 JSON 파일을 사용하여 트리거를 생성하려면 다음과 같이 합니다.

```
aws codecommit put-repository-triggers --cli-input-json
file://trigger.json
```

이 명령은 다음과 유사한 구성 ID를 반환합니다.

```
{
    "configurationId": "0123456-I-AM-AN-EXAMPLE"
}
```

4. 트리거의 구성을 보려면, 리포지토리 이름을 지정하여 get-repository-triggers 명령을 실행합니다.

   ```
   aws codecommit get-repository-triggers --repository-name MyDemoRepo
   ```

   이 명령은 리포지토리에 대해 구성된 모든 트리거의 다음과 비슷한 구조를 반환합니다.

   ```
   {
       "configurationId": "0123456-I-AM-AN-EXAMPLE",
       "triggers": [
           {
               "events": [
                   "all"
               ],
               "destinationArn": "arn:aws:lambda:us-
   east-1:111122223333:MyCodeCommitFunction",
               "branches": [
                   "main",
                   "preprod"
               ],
               "name": "MyLambdaFunctionTrigger",
               "customData": "Project ID 12345"
           }
       ]
   }
   ```

5. 트리거의 기능을 확인하려면 커밋을 만들어 트리거를 구성한 리포지토리에 푸시합니다. Lambda 콘솔의 해당 함수에 대한 모니터링 탭에 Lambda 함수의 응답이 표시되어야 합니다.

리포지 AWS CodeCommit 토리에 대한 트리거 편집

CodeCommit 리포지토리에 대해 생성된 트리거를 편집할 수 있습니다. 트리거를 위한 이벤트와 브랜치, 이벤트 및 기타 설정에 맞게 취해진 조치를 변경할 수 있습니다.

주제

- 리포지토리 트리거 편집 (콘솔)
- 리포지토리 트리거 편집 (AWS CLI)

리포지토리 트리거 편집 (콘솔)

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서, 리포지토리 이벤트 트리거를 편집하려는 리포지토리를 선택합니다.
3. 리포지토리의 탐색 창에서 설정을 선택한 후 트리거를 선택합니다.
4. 리포지토리에 대한 트리거 목록에서 편집하려는 트리거를 선택한 다음 편집을 선택합니다.
5. 원하는 대로 트리거를 변경한 후 저장을 선택합니다.

리포지토리 트리거 편집 (AWS CLI)

1. 터미널(Linux, macOS, Unix) 또는 명령 프롬프트(Windows)에서 get-repository-triggers 명령을 실행하여, 리포지토리에 대해 구성된 모든 트리거의 구조를 갖춘 JSON 파일을 생성합니다. 예를 들어, *MyDemoRepo*라는 리포지토리에 대해 구성된 모든 트리거의 구조를 갖춘 *MyTriggers.json*라는 JSON 파일을 생성하려면 다음과 같이 합니다.

   ```
   aws codecommit get-repository-triggers --repository-name MyDemoRepo
    >MyTriggers.json
   ```

   이 명령은 아무 것도 반환하지 않지만 명령을 실행한 디렉터리에 *MyTriggers.json* 파일을 생성합니다.
2. 평문 편집기에서 JSON 파일을 편집하고 편집할 트리거에 대한 트리거 블록을 변경합니다. `configurationId` 쌍을 `repositoryName` 쌍으로 바꿉니다. 파일을 저장합니다.

   예를 들어, *MyDemoRepo*라는 리포지토리의 *MyFirstTrigger*라는 트리거를 편집해서 모든 브랜치에 적용되도록 하려면, `configurationId`를 `repositoryName`으로 교체한 다음, *### ### ###*의 지정된 `main` 및 `preprod` 브랜치를 제거합니다. 기본적으로 브랜치를 지정하지 않을 경우 트리거가 리포지토리 내 모든 브랜치에 적용됩니다.

   ```
   {
       "repositoryName": "MyDemoRepo",
       "triggers": [
           {
               "destinationArn": "arn:aws:sns:us-
   east-2:111122223333:MyCodeCommitTopic",
               "branches": [
                   "main",
                   "preprod"
               ],
   ```

```
            "name": "MyFirstTrigger",
            "customData": "",
            "events": [
                "all"
            ]
        }
    ]
}
```

3. 터미널 또는 명령줄에서 put-repository-triggers 명령을 실행합니다. 그러면 *MyFirstTrigger* 트리거의 변경 내용을 포함하여 리포지토리에 대한 모든 트리거가 업데이트됩니다.

```
aws codecommit put-repository-triggers --repository-name MyDemoRepo
 file://MyTriggers.json
```

이 명령은 다음과 유사한 구성 ID를 반환합니다.

```
{
    "configurationId": "0123456-I-AM-AN-EXAMPLE"
}
```

리포지 AWS CodeCommit 토리에 대한 트리거 테스트

CodeCommit 리포지토리에 대해 생성된 트리거를 테스트할 수 있습니다. 테스트에는 가장 최근 커밋 ID를 비롯한 리포지토리 샘플 데이터로 트리거를 실행하는 과정이 포함됩니다. (커밋 이력이 없으면 그 대신 0으로 이루어진 샘플 값이 생성됩니다.) 트리거를 테스트하면 CodeCommit과 트리거 대상 간의 액세스가 AWS Lambda 함수이든 Amazon Simple Notification Service 알림이든 올바르게 구성되었는지 확인할 수 있습니다.

주제

- 리포지토리 트리거 테스트 (콘솔)
- 리포지토리 트리거 테스트 (AWS CLI)

리포지토리 트리거 테스트 (콘솔)

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서, 리포지토리 이벤트 트리거를 테스트하려는 리포지토리를 선택합니다.
3. 리포지토리의 탐색 창에서 설정을 선택한 후 트리거를 선택합니다.

4. 테스트하려는 트리거를 선택한 다음 트리거 테스트를 선택합니다. 성공 또는 실패 메시지가 표시되어야 합니다. 성공하면, Lambda 함수 또는 Amazon SNS 주제의 해당 작업 응답도 확인할 수 있습니다.

리포지토리 트리거 테스트 (AWS CLI)

1. 터미널(Linux, macOS, Unix) 또는 명령 프롬프트(Windows)에서 get-repository-triggers 명령을 실행하여, 리포지토리에 대해 구성된 모든 트리거의 구조를 갖춘 JSON 파일을 생성합니다. 예를 들어, MyDemoRepo라는 리포지토리에 대해 구성된 모든 트리거의 구조를 갖춘 *TestTrigger.json*라는 JSON 파일을 생성하려면 다음과 같이 합니다.

```
aws codecommit get-repository-triggers --repository-name MyDemoRepo
 >TestTrigger.json
```

   이 명령은 명령을 실행한 디렉터리에 *TestTriggers.json*이라는 파일을 생성합니다.

2. 일반 텍스트 편집기에서 JSON 파일을 편집하고 트리거 명령문을 변경합니다. `configurationId` 쌍을 `repositoryName` 쌍으로 바꿉니다. 파일을 저장합니다.

   예를 들어, *MyDemoRepo*라는 리포지토리의 *MyFirstTrigger*라는 트리거를 테스트해서 모든 브랜치에 적용되도록 하려면, `configurationId`를 `repositoryName`으로 교체한 다음, 다음과 유사한 내용의 파일을 *TestTrigger.json*으로 저장합니다.

```
{
    "repositoryName": "MyDemoRepo",
    "triggers": [
        {
            "destinationArn": "arn:aws:sns:us-
east-2:111122223333:MyCodeCommitTopic",
            "branches": [
                "main",
                "preprod"
            ],
            "name": "MyFirstTrigger",
            "customData": "",
            "events": [
                "all"
            ]
        }
    ]
```

```
}
```

3. 터미널 또는 명령줄에서 test-repository-triggers 명령을 실행합니다. 그러면 *MyFirstTrigger* 트리거의 변경 내용을 포함하여 리포지토리에 대한 모든 트리거가 업데이트됩니다.

```
aws codecommit test-repository-triggers --cli-input-json file://TestTrigger.json
```

이 명령은 다음과 비슷한 응답을 반환합니다.

```
{
    "successfulExecutions": [
        "MyFirstTrigger"
    ],
    "failedExecutions": []
}
```

AWS CodeCommit 리포지토리에서 트리거 삭제

더 이상 사용되지 않는 트리거는 삭제하는 것이 좋을 수도 있습니다. 트리거를 삭제하면 되돌릴 수 없지만, 트리거를 다시 생성할 수는 있습니다.

> **Note**
> 리포지토리에 트리거를 구성한 경우, 리포지토리를 삭제해도 그 트리거의 대상으로 구성한 Amazon SNS 주제 또는 Lambda 함수는 삭제되지 않습니다. 더 이상 필요하지 않은 리소스도 삭제해야 합니다.

주제

- 리포지토리에서 트리거 삭제 (콘솔)
- 리포지토리에서 트리거 삭제 (AWS CLI)

리포지토리에서 트리거 삭제 (콘솔)

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서, 리포지토리 이벤트 트리거를 삭제하려는 리포지토리를 선택합니다.
3. 리포지토리의 탐색 창에서 설정을 선택합니다. 설정에서 트리거를 선택합니다.

4. 트리거 목록에서 삭제할 트리거를 선택한 다음 삭제를 선택합니다.
5. 대화 상자에 삭제를 입력해 확인합니다.

리포지토리에서 트리거 삭제 (AWS CLI)

1. 터미널(Linux, macOS, Unix) 또는 명령 프롬프트(Windows)에서 get-repository-triggers 명령을 실행하여, 리포지토리에 대해 구성된 모든 트리거의 구조를 갖춘 JSON 파일을 생성합니다. 예를 들어, MyDemoRepo라는 리포지토리에 대해 구성된 모든 트리거의 구조를 갖춘 *MyTriggers.json*라는 JSON 파일을 생성하려면 다음과 같이 합니다.

```
aws codecommit get-repository-triggers --repository-name MyDemoRepo
 >MyTriggers.json
```

   이 명령은 명령을 실행한 디렉터리에 *MyTriggers.json*이라는 파일을 생성합니다.
2. 일반 텍스트 편집기에서 JSON 파일을 편집하고, 삭제할 트리거에 대한 트리거 블록을 제거합니다. `configurationId` 쌍을 `repositoryName` 쌍으로 바꿉니다. 파일을 저장합니다.

   예를 들어, *MyDemoRepo*라는 리포지토리에서 *MyFirstTrigger*라는 트리거를 제거하려면, `configurationId`를 `repositoryName`으로 교체한 다음, *### ### ###*의 명령문을 제거합니다.

```
{
    "repositoryName": "MyDemoRepo",
    "triggers": [
        {
            "destinationArn": "arn:aws:sns:us-
east-2:111122223333:MyCodeCommitTopic",
            "branches": [
                "main",
                "preprod"
            ],
            "name": "MyFirstTrigger",
            "customData": "",
            "events": [
                "all"
            ]
        },
        {
            "destinationArn": "arn:aws:lambda:us-
east-2:111122223333:function:MyCodeCommitJSFunction",
```

```
            "branches": [],
            "name": "MyLambdaTrigger",
            "events": [
                "all"
            ]
        }
    ]
}
```

3. 터미널 또는 명령줄에서 put-repository-triggers 명령을 실행합니다. 그러면 리포지토리에 대한 트리거가 업데이트되고, *MyFirstTrigger* 트리거가 삭제됩니다.

```
aws codecommit put-repository-triggers --repository-name MyDemoRepo
 file://MyTriggers.json
```

이 명령은 다음과 유사한 구성 ID를 반환합니다.

```
{
    "configurationId": "0123456-I-AM-AN-EXAMPLE"
}
```

> **Note**
>
> *MyDemorepo*라는 리포지토리의 모든 트리거를 삭제하려면, JSON 파일이 다음과 같이 표시됩니다.
>
> ```
> {
> "repositoryName": "MyDemoRepo",
> "triggers": []
> }
> ```

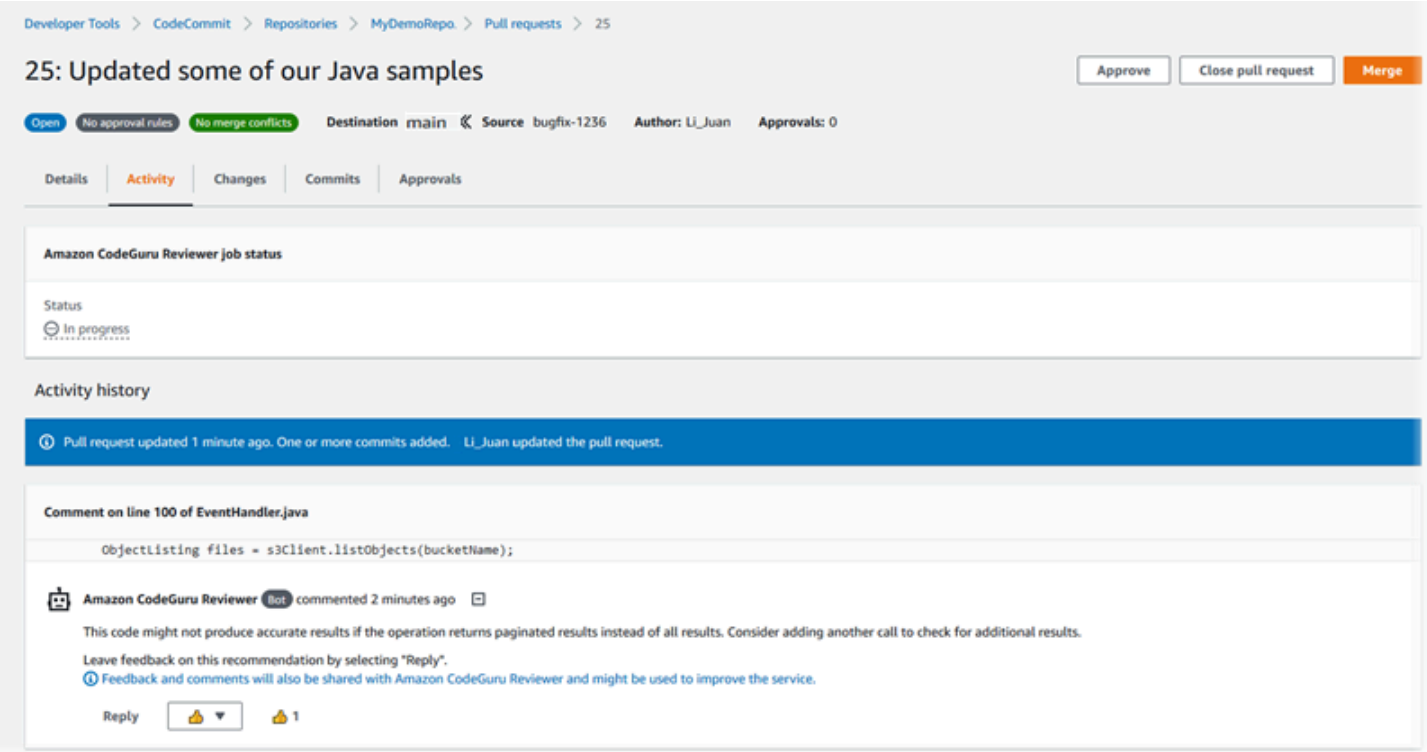
AWS CodeCommit 리포지토리를 Amazon CodeGuru Reviewer와 연결 또는 연결 해제

Amazon CodeGuru Reviewer는 프로그램 분석 및 기계 학습을 사용하여 Java 또는 Python 코드에서 일반적인 문제를 감지하고 권장 수정 사항을 제공하는 자동화된 코드 검토 서비스입니다. 사용자는 Amazon Web Services 계정의 리포지토리를 CodeGuru Reviewer와 연결할 수 있습니다. 이렇게 하면

CodeGuru Reviewer는 CodeGuru Reviewer가 연결 설정 후 생성된 모든 풀 요청에서 코드를 분석할 수 있도록 허용하는 서비스 연결 역할을 생성합니다.

리포지토리를 연결한 후 CodeGuru Reviewer는 사용자가 풀 요청을 생성할 때 발견된 문제를 분석하고 의견을 제공합니다. 각 주석은 Amazon CodeGuru Reviewer라는 명칭의 CodeGuru Reviewer가 작성한 것으로 명확히 표시됩니다. 풀 요청에서 다른 의견에 응답할 때와 마찬가지로 이러한 의견에 응답할 수 있습니다. 또한 제안의 품질에 대한 피드백을 제공할 수도 있습니다. 이 피드백은 CodeGuru Reviewer와 공유되며 서비스 및 제안 사항을 개선하는 데 도움이 될 수 있습니다.

> **Note**
>
> 리포지토리가 연결되기 전에 작성된 풀 요청에 대한 CodeGuru Reviewer의 주석은 표시되지 않습니다. 다음은 연결 이후에 생성된 풀 요청에 의견이 표시되지 않을 수 있는 이유입니다.
>
> - 풀 요청에 Java 또는 Python 코드가 포함되어 있지 않습니다.
> - CodeGuru Reviewer가 풀 요청에서 코드를 실행하고 검토할 시간이 충분하지 않았습니다. 이 프로세스는 최대 30분이 걸릴 수 있습니다. 주석은 검토가 진행되면서 나타날 수 있지만, 주석 작성은 작업 상태가 완료됨으로 표시되기 전까지 완료되지 않습니다.
> - CodeGuru Reviewer가 풀 요청에서 Java 또는 Python 코드의 문제를 찾지 못했습니다.
> - 코드 검토 작업을 실행하지 못했습니다. 풀 요청에 대한 검토 상태를 검토하려면, 풀 요청의 활동 탭을 참조하세요.
> - 변경 사항 탭에서 풀 요청에 대한 변경 사항을 검토하고 있고, 풀 요청이 업데이트되었으며, Amazon CodeGuru Reviewer가 변경 사항에서 어떤 문제도 찾지 못했습니다. Amazon CodeGuru Reviewer 주석은 풀 요청의 가장 최근 수정본에 대해 작성된 경우에만 변경 사항 탭에 표시됩니다. 이러한 주석은 활동 탭에 항상 표시됩니다.



자세한 내용은 AWS CodeCommit 리포지토리에서 풀 요청 작업, 풀 요청 검토, Amazon CodeGuru Reviewer 사용 안내서를 참조하세요.

> **Note**
>
> 리포지토리를 CodeGuru Reviewer와 연결하거나 연결 해제할 수 있는 충분한 권한이 있는 IAM 사용자 또는 역할로 로그인해야 합니다. 이러한 권한이 포함된 CodeCommit용 관리형 정책에 대한 자세한 내용은 AWS CodeCommit에 대한 관리형 정책 및 AWS CodeCommit 관리형 정책 및 Amazon CodeGuru Reviewer 섹션을 참조하세요. CodeGuru Reviewer 권한 및 보안에 대한 자세한 내용은 Amazon CodeGuru Reviewer 사용 설명서를 참조하세요.

주제

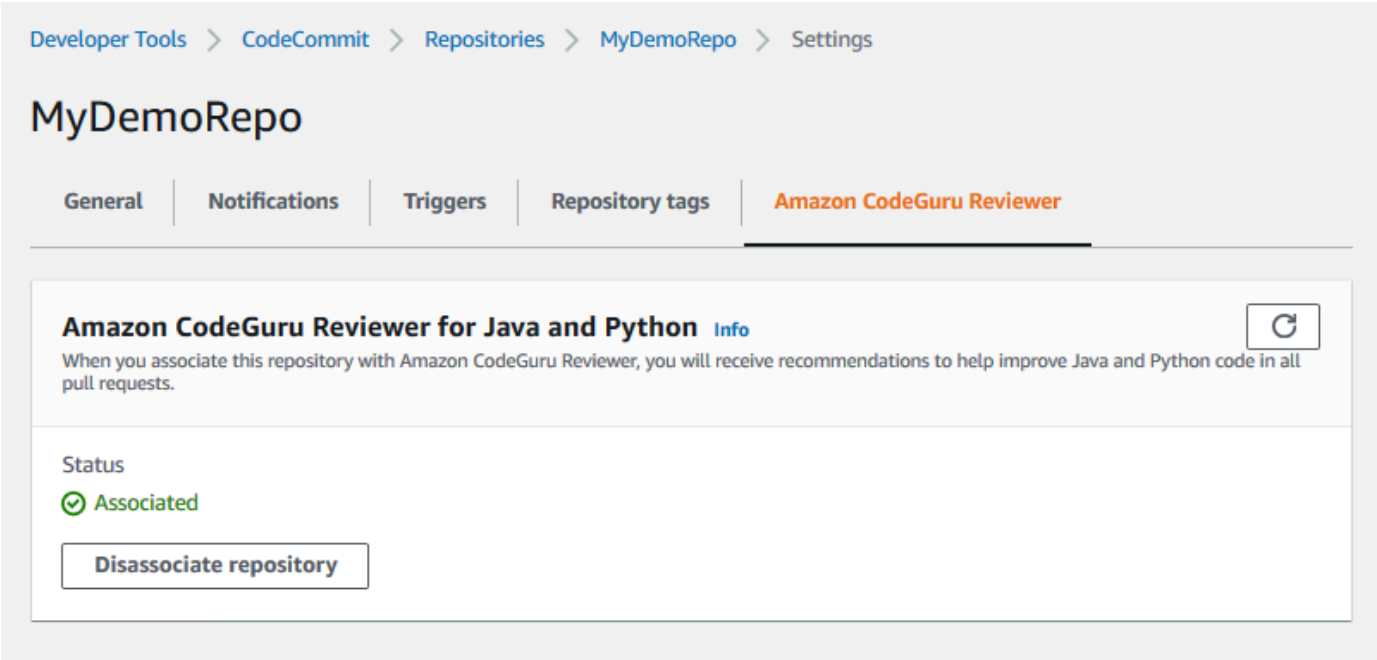
- 리포지토리를 CodeGuru Reviewer와 연결
- CodeGuru Reviewer에서 리포지토리 연결 해제

리포지토리를 CodeGuru Reviewer와 연결

AWS CodeCommit 콘솔을 사용하여 리포지토리를 CodeGuru Reviewer와 빠르게 연결합니다. 다른 방법에 대해서는 Amazon CodeGuru Reviewer 사용 설명서를 참조하세요.

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서 CodeGuru Reviewer와 연결할 리포지토리의 이름을 선택합니다.
3. 설정을 선택한 다음 Amazon CodeGuru Reviewer를 선택합니다.
4. 리포지토리 연결을 선택합니다.

> **Note**
>
> 리포지토리를 CodeGuru Reviewer와 완전히 연결하는 데 최대 10분이 걸릴 수 있습니다. 상태는 자동으로 업데이트되지 않습니다. 현재 상태를 보려면 새로 고침 버튼을 선택합니다.



CodeGuru Reviewer에서 리포지토리 연결 해제

AWS CodeCommit 콘솔을 사용하여 CodeGuru Reviewer에서 리포지토리의 연결을 빠르게 해제합니다. 다른 방법에 대해서는 Amazon CodeGuru Reviewer 사용 설명서를 참조하세요.

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서 CodeGuru Reviewer와의 연결을 해제할 리포지토리의 이름을 선택합니다.
3. 설정을 선택한 다음 Amazon CodeGuru Reviewer를 선택합니다.
4. 리포지토리 연결 해제를 선택합니다.

CodeCommit 리포지토리 세부 정보 보기

AWS CodeCommit 콘솔 AWS CLI또는 CodeCommit 리포지토리에 연결된 로컬 리포지토리의 Git을 사용하여 사용 가능한 리포지토리에 대한 정보를 볼 수 있습니다.

이러한 지침을 따르기에 앞서 설정 의 단계를 완료합니다.

주제

- 리포지토리 세부 정보 보기 (콘솔)
- CodeCommit 리포지토리 세부 정보 보기 (Git)
- CodeCommit 리포지토리 세부 정보 보기 (AWS CLI)

리포지토리 세부 정보 보기 (콘솔)

AWS CodeCommit 콘솔을 사용하여 Amazon Web Services 계정으로 생성된 모든 리포지토리를 빠르게 볼 수 있습니다.

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서 로그인한 AWS 리전 의 리포지토리에 대한 세부 정보를 볼 수 있습니다. 리전 선택기를 사용하여 다른 AWS 리전을 선택하면 해당 리전의 리포지토리를 볼 수 있습니다.
3. 자세한 내용을 보려는 리포지토리의 이름을 선택한 다음 다음 중 하나를 수행합니다.
   - 리포지토리 복제를 위한 URL을 보려면, URL 복제를 선택한 다음, 리포지토리를 복제할 때 사용할 프로토콜을 선택합니다. 이렇게 하면 복제 URL이 복사됩니다. 이를 검토하려면 일반 텍스트 편집기에 붙여넣습니다.
   - 리포지토리의 구성 가능한 옵션과 세부 정보(예: 리포지토리 ARN 및 리포지토리 ID)를 보려면 탐색 창에서 설정을 선택합니다.

> **Note**
>
> IAM 사용자로 로그인한 경우 코드 및 다른 콘솔 설정 보기에 대한 기본 설정을 구성하고 저장할 수 있습니다. 자세한 내용은 사용자 기본 설정으로 작업 단원을 참조하십시오.

CodeCommit 리포지토리 세부 정보 보기 (Git)

로컬 리포지토리의 Git을 사용하여 CodeCommit 리포지토리에 대한 세부 정보를 보려면 git remote show 명령을 실행합니다.

이 단계를 수행하기에 앞서 로컬 리포지토리를 CodeCommit 리포지토리에 연결합니다. 지침은 리포지토리에 연결 단원을 참조하세요.

1. git remote show ***remote-name*** 명령을 실행합니다. 여기서 *remote-name*은 CodeCommit 리포지토리의 별칭입니다. (기본값 `origin`)

   > **Tip**
   >
   > CodeCommit 리포지토리 이름의 목록과 그 URL을 가져오려면 git remote -v 명령을 실행합니다.

   예를 들어, `origin`라는 별칭이 있는 CodeCommit 리포지토리의 세부 정보를 보려면 다음과 같이 합니다.

   ```
   git remote show origin
   ```

2. HTTPS의 경우에는 다음과 같이 합니다.

   ```
   * remote origin
     Fetch URL: https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
     Push  URL: https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
     HEAD branch: (unknown)
     Remote branches:
       MyNewBranch tracked
       main tracked
     Local ref configured for 'git pull':
       MyNewBranch merges with remote MyNewBranch (up to date)
     Local refs configured for 'git push':
       MyNewBranch pushes to MyNewBranch (up to date)
       main pushes to main (up to date)
   ```

   SSH의 경우에는 다음과 같이 합니다.

   ```
   * remote origin
   ```

```
  Fetch URL: ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
  Push  URL: ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
  HEAD branch: (unknown)
  Remote branches:
    MyNewBranch tracked
    main tracked
  Local ref configured for 'git pull':
    MyNewBranch merges with remote MyNewBranch (up to date)
  Local refs configured for 'git push':
    MyNewBranch pushes to MyNewBranch (up to date)
    main pushes to main (up to date)
```

> **Tip**
>
> IAM 사용자에 대한 SSH 키 ID를 조회하려면, IAM 콘솔을 열고 IAM 사용자 세부 정보 페이지에서 보안 인증 정보를 확장합니다. SSH 키 ID는 SSH 키에서 찾을 수 있습니다 AWS CodeCommit.

다른 옵션들에 대해서는 Git 설명서를 참조하세요.

CodeCommit 리포지토리 세부 정보 보기 (AWS CLI)

CodeCommit에서 AWS CLI 명령을 사용하려면를 설치합니다 AWS CLI. 자세한 내용은 명령줄 참조 단원을 참조하십시오.

AWS CLI 를 사용하여 리포지토리 세부 정보를 보려면 다음 명령을 실행합니다.

- CodeCommit 리포지토리 이름 및 해당 ID의 목록을 보려면 list-repositories를 실행합니다.
- 단일 CodeCommit 리포지토리에 대한 정보를 보려면 get-repository를 실행합니다.
- CodeCommit에서 다수의 리포지토리에 대한 정보를 보려면 batch-get-repositories를 실행합니다.

CodeCommit 리포지토리의 목록을 보려면

1. list-repositories 명령을 실행합니다.

   ```
   aws codecommit list-repositories
   ```

   선택적 `--sort-by` 또는 `--order` 옵션을 사용하여 반환되는 정보의 순서를 변경할 수 있습니다.

2. 성공할 경우 이 명령은 Amazon Web Services 계정과 연결된 CodeCommit에 있는 모든 리포지토리의 이름과 ID를 포함하는 `repositories` 객체를 출력합니다.

   다음은 위의 명령을 기반으로 하는 출력 예시입니다.

```
{
    "repositories": [
        {
           "repositoryName": "MyDemoRepo",
           "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE"
        },
        {
           "repositoryName": "MyOtherDemoRepo",
           "repositoryId": "cfc29ac4-b0cb-44dc-9990-f6f51EXAMPLE"
        }
    ]
}
```

단일 CodeCommit 리포지토리의 세부 정보를 보려면

1. CodeCommit 리포지토리의 이름과 `--repository-name` 옵션을 지정하여 get-repository 명령을 실행합니다.

   > **Tip**
   > CodeCommit 리포지토리의 이름을 가져오려면 list-repositories 명령을 실행합니다.

   예를 들어, `MyDemoRepo`라는 CodeCommit 리포지토리의 세부 정보를 보려면 다음과 같이 합니다.

```
aws codecommit get-repository --repository-name MyDemoRepo
```

2. 이 명령이 제대로 실행되면 다음 정보를 포함하는 `repositoryMetadata` 객체가 출력됩니다.

   - 리포지토리의 이름 (`repositoryName`)
   - 리포지토리의 설명 (`repositoryDescription`)
   - 리포지토리의 고유 시스템 생성 ID (`repositoryId`)
   - 리포지토리와 연결된 Amazon Web Services 계정의 ID (`accountId`)

다음은 위 예제 명령의 출력 예입니다.

```
{
        "repositoryMetadata": {
            "creationDate": 1429203623.625,
            "defaultBranch": "main",
            "repositoryName": "MyDemoRepo",
            "cloneUrlSsh": "ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/
MyDemoRepo",
            "lastModifiedDate": 1430783812.0869999,
            "repositoryDescription": "My demonstration repository",
            "cloneUrlHttp": "https://codecommit.us-east-2.amazonaws.com/v1/repos/
MyDemoRepo",
            "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE",
            "Arn": "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
            "accountId": "111111111111"
        }
}
```

다수의 CodeCommit 리포지토리에 대한 세부 정보를 보려면

1. batch-get-repositories 옵션으로 `--repository-names` 명령을 실행합니다. 각 CodeCommit 리포지토리 이름 사이에 스페이스를 추가합니다.

> **Tip**
>
> CodeCommit 리포지토리의 이름을 가져오려면 list-repositories 명령을 실행합니다.

예를 들어, `MyDemoRepo` 및 `MyOtherDemoRepo`라는 두 CodeCommit 리포지토리의 세부 정보를 보려면 다음과 같이 합니다.

```
aws codecommit batch-get-repositories --repository-names MyDemoRepo MyOtherDemoRepo
```

2. 이 명령이 제대로 실행되면 다음 정보를 포함하는 객체가 출력됩니다.

   - 찾을 수 없는 CodeCommit 리포지토리의 목록 (`repositoriesNotFound`)

- CodeCommit 리포지토리 목록 (`repositories`) 각 CodeCommit 리포지토리 이름 뒤에는 다음이 붙습니다.
  - 리포지토리의 설명 (`repositoryDescription`)
  - 리포지토리의 고유 시스템 생성 ID (`repositoryId`)
  - 리포지토리와 연결된 Amazon Web Services 계정의 ID (`accountId`)

다음은 위 예제 명령의 출력 예입니다.

```
{
        "repositoriesNotFound": [],
        "repositories": [
             {
                "creationDate": 1429203623.625,
                "defaultBranch": "main",
                "repositoryName": "MyDemoRepo",
                "cloneUrlSsh": "ssh://git-codecommit.us-east-2.amazonaws.com/v1/
repos/MyDemoRepo",
                "lastModifiedDate": 1430783812.0869999,
                "repositoryDescription": "My demonstration repository",
                "cloneUrlHttp": "https://codecommit.us-east-2.amazonaws.com/v1/
repos/MyDemoRepo",
                "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE",
                "Arn": "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
                "accountId": "111111111111"
            },
            {
                "creationDate": 1429203623.627,
                "defaultBranch": "main",
                "repositoryName": "MyOtherDemoRepo",
                "cloneUrlSsh": "ssh://git-codecommit.us-east-2.amazonaws.com/v1/
repos/MyOtherDemoRepo",
                "lastModifiedDate": 1430783812.0889999,
                "repositoryDescription": "My other demonstration repository",
                "cloneUrlHttp": "https://codecommit.us-east-2.amazonaws.com/v1/
repos/MyOtherDemoRepo",
                "repositoryId": "cfc29ac4-b0cb-44dc-9990-f6f51EXAMPLE",
                "Arn": "arn:aws:codecommit:us-east-2:111111111111:MyOtherDemoRepo",
                "accountId": "111111111111"
            }
        ],
        "repositoriesNotFound": []
```

```
    }
```

AWS CodeCommit 리포지토리 설정 변경

AWS CLI 및 AWS CodeCommit 콘솔을 사용하여 설명 또는 이름과 같은 CodeCommit 리포지토리의 설정을 변경할 수 있습니다.

> **Important**
>
> 리포지토리 이름을 변경하면 원격 URL에 기존 이름을 사용하는 모든 로컬 리포지토리가 손상될 수 있습니다. git remote set-url 명령을 실행하여 원격 URL이 새 리포지토리의 이름을 사용하도록 업데이트합니다.

주제

- 리포지토리 설정 변경 (콘솔)
- AWS CodeCommit 리포지토리 설정 변경(AWS CLI)

리포지토리 설정 변경 (콘솔)

AWS CodeCommit 콘솔을 사용하여에서 CodeCommit 리포지토리의 설정을 변경하려면 다음 단계를 AWS CodeCommit따릅니다.

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서, 설정을 변경하려는 리포지토리의 이름을 선택합니다.
3. 탐색 창에서 설정을 선택합니다.
4. 리포지토리 이름을 변경하려면 리포지토리 이름에서 이름 텍스트 상자에 이름을 입력하고 저장을 선택합니다. 메시지가 표시되면 선택 내용을 확인합니다.

   > **Important**
   >
   > AWS CodeCommit 리포지토리의 이름을 변경하면 사용자가 리포지토리에 연결하는 데 필요한 SSH 및 HTTPS URLs이 변경됩니다. 사용자는 연결 설정을 업데이트할 때까지 이 리포지토리에 연결할 수 없습니다. 또한 리포지토리의 ARN이 변경되므로, 리포지토리 이름을 변경하면 이 리포지토리의 ARN을 사용하는 모든 IAM 사용자 정책이 무효화됩니다.

이름을 변경한 후 리포지토리에 연결하려면, 각 사용자가 git remote set-url 명령을 사용하고 새 URL을 지정해야 합니다. 예를 들어, 리포지토리의 이름을 MyDemoRepo에서 MyRenamedDemoRepo로 변경한 경우, HTTPS를 사용하여 리포지토리에 연결하는 사용자는 다음 Git 명령을 실행합니다.

```
git remote set-url origin https://git-codecommit.us-east-2.amazonaws.com/v1/
repos/MyRenamedDemoRepo
```

SSH를 사용하여 리포지토리에 연결하는 사용자는 다음 Git 명령을 실행합니다.

```
git remote set-url origin ssh://git-codecommit.us-east-2.amazonaws.com/v1/
repos/MyRenamedDemoRepo
```

다른 옵션들에 대해서는 Git 설명서를 참조하세요.

5. 리포지토리의 설명을 변경하려면 설명 텍스트 상자에서 텍스트를 수정한 다음 저장을 선택합니다.

> **Note**
>
> 설명 필드에는 콘솔의 마크다운이 표시되며, 모든 HTML 문자와 유효한 Unicode 문자를 모두 사용할 수 있습니다. `GetRepository` 또는 `BatchGetRepositories` API를 사용하는 애플리케이션 개발자인 경우 웹 브라우저에 리포지토리 설명 필드를 표시하려면 CodeCommit API 참조를 참조하세요.

6. 기본 브랜치를 변경하려면 기본 브랜치에서 브랜치 드롭다운 목록을 선택한 다음 다른 브랜치를 선택합니다. 저장을 선택합니다.
7. 리포지토리의 데이터를 암호화하고 복호화하는 데 사용되는 암호화 AWS KMS 키를 변경하려면 리포지토리 암호화 키에서 AWS 관리형 키 또는 고객 관리형 키를 선택하여 사용할 키 유형을 지정합니다. 고객 관리형 키를 선택하는 경우 키의 ARN을 입력합니다. 저장을 선택합니다.
8. 리포지토리를 삭제하려면 리포지토리 삭제를 선택합니다. 리포지토리의 이름을 입력하여 삭제를 확인 옆의 상자에 **`delete`**를 입력한 다음 삭제를 선택합니다.

> **Important**
>
> 에서이 리포지토리를 삭제 AWS CodeCommit한 후에는 더 이상 로컬 리포지토리 또는 공유 리포지토리에 복제할 수 없습니다. 또한 어떠한 로컬 리포지토리 또는 공유 리포지토리

를 대상으로도 데이터를 풀하거나 푸시할 수 없습니다. 이 작업은 실행을 취소할 수 없습니다.

AWS CodeCommit 리포지토리 설정 변경(AWS CLI)

CodeCommit에서 AWS CLI 명령을 사용하려면를 설치합니다 AWS CLI. 자세한 내용은 명령줄 참조 단원을 참조하십시오.

AWS CLI 를 사용하여에서 CodeCommit 리포지토리의 설정을 변경하려면 다음 명령 중 하나 이상을 AWS CodeCommit실행합니다.

- update-repository-description - CodeCommit 리포지토리의 설명을 변경합니다.
- update-repository-name - CodeCommit 리포지토리의 이름을 변경합니다.

CodeCommit 리포지토리의 설명을 변경하려면

1. 다음을 지정하여 update-repository-description 명령을 실행합니다.

   - CodeCommit 리포지토리의 이름(`--repository-name` 옵션 사용)

     > **Tip**
     > CodeCommit 리포지토리의 이름을 가져오려면 list-repositories 명령을 실행합니다.

   - 새 리포지토리 설명(`--repository-description` 옵션 사용)

     > **Note**
     > 설명 필드에는 콘솔의 마크다운이 표시되며, 모든 HTML 문자와 유효한 Unicode 문자를 모두 사용할 수 있습니다. `GetRepository` 또는 `BatchGetRepositories` API를 사용하는 애플리케이션 개발자인 경우 웹 브라우저에 리포지토리 설명 필드를 표시하려면 CodeCommit API 참조를 참조하세요.

   예를 들어, `MyDemoRepo`라는 CodeCommit 리포지토리에 대한 설명을 `This description was changed`로 변경하려면 다음과 같이 합니다.

```
aws codecommit update-repository-description --repository-name MyDemoRepo --
repository-description "This description was changed"
```

이 명령은 오류가 있는 경우에만 출력을 생성합니다.

2. 변경된 설명을 확인하려면 `--repository-name` 옵션을 사용하여, 설명을 변경한 CodeCommit 리포지토리의 이름을 지정하여 get-repository 명령을 실행합니다.

   이 명령의 출력은 `repositoryDescription`에서 변경된 텍스트를 보여 줍니다.

CodeCommit 리포지토리의 이름을 변경하려면

1. 다음을 지정하여 update-repository-name 명령을 실행합니다.

   - CodeCommit 리포지토리의 현재 이름(`--old-name` 옵션 사용)

     > **Tip**
     > CodeCommit 리포지토리 이름을 가져오려면 list-repositories 명령을 실행합니다.

   - CodeCommit 리포지토리의 새 이름(`--new-name` 옵션 사용)

   예를 들어, 리포지토리 이름을 `MyDemoRepo`에서 `MyRenamedDemoRepo`로 변경하려면 다음과 같이 합니다.

```
aws codecommit update-repository-name --old-name MyDemoRepo --new-name
 MyRenamedDemoRepo
```

   이 명령은 오류가 있는 경우에만 출력을 생성합니다.

> **Important**
> AWS CodeCommit 리포지토리의 이름을 변경하면 사용자가 리포지토리에 연결하는 데 필요한 SSH 및 HTTPS URLs이 변경됩니다. 사용자는 연결 설정을 업데이트할 때까지 이 리포지토리에 연결할 수 없습니다. 또한 리포지토리의 ARN이 변경되므로, 리포지토리 이름을 변경하면 이 리포지토리의 ARN을 사용하는 모든 IAM 사용자 정책이 무효화됩니다.

2. 변경된 이름을 확인하려면 list-repositories 명령을 실행한 다음 리포지토리 이름의 목록을 검토합니다.

로컬 리포지토리와 AWS CodeCommit 리포지토리 간의 변경 사항 동기화

Git을 사용하여, 로컬 리포지토리와 이 로컬 리포지토리에 연결된 CodeCommit 리포지토리 간의 변경 사항을 동기화합니다.

변경 사항을 로컬 리포지토리에서 CodeCommit 리포지토리로 푸시하려면 git push ***remote-name branch-name***을 실행합니다.

변경 사항을 로컬 CodeCommit 리포지토리에서 로컬 리포지토리로 풀하려면 git pull ***remote-name branch-name***을 실행합니다.

푸시의 경우든 풀의 경우든, *remote-name*은 로컬 리포지토리가 CodeCommit 리포지토리에 사용하는 별명입니다. *branch-name*은 푸시 및 풀의 대상이 되는 CodeCommit 리포지토리의 브랜치 이름입니다.

> **Tip**
>
> 로컬 리포지토리가 CodeCommit 리포지토리에 대해 사용하는 별명을 가져오려면 git remote를 실행합니다. 브랜치 이름 목록을 가져오려면 git branch 명령을 실행합니다. 현재 설정된 브랜치의 이름 옆에 별표(*)가 표시됩니다. (git status 명령을 실행해 현재 브랜치 이름을 표시할 수도 있습니다.)

> **Note**
>
> 리포지토리를 복제한 경우 로컬 리포지토리의 관점에서 *remote-name*은 CodeCommit 리포지토리의 이름이 아닙니다. 리포지토리를 복제할 때 *remote-name*은 자동으로 `origin`으로 설정됩니다.

예를 들어, `origin`라는 별명을 사용하여 변경 사항을 로컬 리포지토리에서 CodeCommit 리포지토리의 `main` 브랜치로 푸시하려면, 다음과 같이 합니다.

```
git push origin main
```

마찬가지로, `origin`라는 별명을 사용하여 변경 사항을 CodeCommit 리포지토리의 `main` 브랜치에서 로컬 리포지토리로 풀하려면, 다음과 같이 합니다.

```
git pull origin main
```

> **Tip**
>
> git push에 `-u` 옵션을 추가하면 업스트림 추적 정보가 설정됩니다. 예를 들어, git push -u origin main을 실행하면 나중에 *`remote-name`*, *`branch-name`*을 사용하지 않고 git push와 git pull을 실행할 수 있습니다. 업스트림 추적 정보를 보려면 git remote show ***`remote-name`***을 실행합니다(예: git remote show origin).

다른 옵션들에 대해서는 Git 설명서를 참조하세요.

추가 Git 리포지토리로 커밋 푸시

변경 내용을 두 개의 원격 리포지토리에 푸시하도록 로컬 리포지토리를 구성할 수 있습니다. 예를 들어, 사용자는 AWS CodeCommit을 사용해 보는 동안 기존 Git 리포지토리 솔루션을 계속 사용하려 할 수 있습니다. 다음 기본 단계에 따라 로컬 리포지토리의 변경 사항을 CodeCommit 및 별도의 Git 리포지토리로 푸시합니다.

> **Tip**
>
> Git 리포지토리가 없는 경우에는 CodeCommit 이외의 서비스에 빈 리포지토리를 생성한 다음 CodeCommit 리포지토리를 해당 리포지토리로 마이그레이션할 수 있습니다. CodeCommit으로 마이그레이션의 단계와 유사한 단계를 따라야 합니다.

1. 명령 프롬프트나 터미널에서, 로컬 리포지토리 디렉터리로 전환한 다음 git remote -v 명령을 실행합니다. 다음과 유사한 출력 화면이 표시되어야 합니다.

   HTTPS의 경우에는 다음과 같이 합니다.

   ```
   origin  https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (fetch)
   ```

```
origin  https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (push)
```

SSH의 경우에는 다음과 같이 합니다.

```
origin  ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (fetch)
origin  ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (push)
```

2. git remote set-url --add --push origin ***git-repository-name*** 명령을 실행합니다. 여기서 *git-repository-name*은 코드를 호스트하려는 Git 리포지토리의 URL이자 이름입니다. 이렇게 하면 `origin`의 푸시 대상이 해당 Git 리포지토리로 변경됩니다.

> **Note**
>
> git remote set-url --add --push은 푸시에 대한 기본 URL을 재정의하므로, 사용자는 이후 단계에서 설명하는 것처럼 이 명령을 두 번 실행해야 합니다.

예를 들어, 다음 명령은 origin의 푸시 대상을 *some-URL*/MyDestinationRepo로 변경합니다.

```
git remote set-url --add --push origin some-URL/MyDestinationRepo
```

이 명령은 아무 것도 반환하지 않습니다.

> **Tip**
>
> 보안 인증 정보가 필요한 Git 리포지토리로 푸시하는 경우에는 해당 보안 인증 정보를 보안 인증 도우미 또는 *some-URL* 문자열 구성에서 구성해야 합니다. 그러지 않으면 해당 리포지토리로의 푸시가 실패합니다.

3. git remote -v 명령을 다시 실행하면 다음과 비슷한 출력이 생성됩니다.

HTTPS의 경우에는 다음과 같이 합니다.

```
origin  https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (fetch)
origin  some-URL/MyDestinationRepo (push)
```

SSH의 경우에는 다음과 같이 합니다.

```
origin  ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (fetch)
origin  some-URL/MyDestinationRepo (push)
```

4. 이제 CodeCommit 리포지토리를 추가합니다. git remote set-url --add --push origin을 다시 실행합니다. 이번에는 CodeCommit 리포지토리의 URL 및 리포지토리 이름을 사용합니다.

   예를 들어, 다음 명령은 origin의 푸시를 'https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo'에 추가합니다.

   HTTPS의 경우에는 다음과 같이 합니다.

```
git remote set-url --add --push origin https://git-codecommit.us-
east-2.amazonaws.com/v1/repos/MyDemoRepo
```

   SSH의 경우에는 다음과 같이 합니다.

```
git remote set-url --add --push origin ssh://git-codecommit.us-
east-2.amazonaws.com/v1/repos/MyDemoRepo
```

   이 명령은 아무 것도 반환하지 않습니다.

5. git remote -v 명령을 다시 실행하면 다음과 비슷한 출력이 생성됩니다.

   HTTPS의 경우에는 다음과 같이 합니다.

```
origin  https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (fetch)
origin  some-URL/MyDestinationRepo (push)
origin  https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (push)
```

   SSH의 경우에는 다음과 같이 합니다.

```
origin  ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (fetch)
origin  some-URL/MyDestinationRepo (push)
origin  ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (push)
```

이제 푸시의 대상으로 두 개의 Git 리포지토리를 보유하고 있지만, 먼저 푸시를 *some-URL*/MyDestinationRepo로 이동합니다. 해당 리포지토리로의 푸시가 실패하면 커밋은 어느 리포지토리로도 푸시되지 않습니다.

> **Tip**
> 다른 리포지토리에서, 수동으로 입력하려는 보안 인증 정보를 요구하는 경우에는 CodeCommit으로 먼저 푸시하도록 푸시 순서를 변경하는 것이 좋습니다. git remote set-url --delete를 실행하여 첫 번째로 푸시된 리포지토리를 삭제한 다음 git remote set-url --add를 실행하여 해당 리포지토리를 다시 추가합니다. 그러면 해당 리포지토리가 목록에서 두 번째 푸시 대상이 됩니다.
> 다른 옵션들에 대해서는 Git 설명서를 참조하세요.

6. 이제 두 원격 리포지토리로 푸시하고 있는지 확인하려면, 텍스트 편집기를 사용하여 로컬 리포지토리에 다음 텍스트 파일을 생성합니다.

```
bees.txt
-------
Bees are flying insects closely related to wasps and ants, and are known for their
 role in pollination and for producing honey and beeswax.
```

7. git add를 실행하여 로컬 리포지토리에 변경 사항을 스테이징합니다.

```
git add bees.txt
```

8. git commit을 실행하여 로컬 리포지토리에 변경 사항을 커밋합니다.

```
git commit -m "Added bees.txt"
```

9. 커밋을 로컬 리포지토리에서 원격 리포지토리로 푸시하기 위해 git push -u ***remote-name branch-name***을 실행합니다. 이때 *remote-name*은 로컬 리포지토리가 원격 리포지토리에 대해 사용하는 별명이고 *branch-name*은 리포지토리로 푸시할 브랜치의 이름입니다.

> **Tip**
> 처음 푸시할 때만 -u 옵션을 사용하면 됩니다. 그러면 업스트림 추적 정보가 설정됩니다.

예를 들어, git push -u origin main을 실행하면 푸시가 예상 브랜치의 두 원격 리포지토리로 이동한 것으로 표시되며, 출력은 다음과 유사합니다.

HTTPS의 경우에는 다음과 같이 합니다.

```
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 5.61 KiB | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
To some-URL/MyDestinationRepo
   a5ba4ed..250f6c3  main -> main
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 5.61 KiB | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote:
To https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
   a5ba4ed..250f6c3  main -> main
```

SSH의 경우에는 다음과 같이 합니다.

```
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 5.61 KiB | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
To some-URL/MyDestinationRepo
   a5ba4ed..250f6c3  main -> main
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 5.61 KiB | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote:
To ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
   a5ba4ed..250f6c3  main -> main
```

다른 옵션들에 대해서는 Git 설명서를 참조하세요.

역할을 사용하여 AWS CodeCommit 리포지토리에 대한 교차 계정 액세스 구성

다른 AWS 계정의 IAM 사용자 및 그룹에 대해 CodeCommit 리포지토리 액세스를 구성할 수 있습니다. 이를 종종 교차 계정 액세스라고 합니다. 이 섹션에서는 AWS 계정(AccountA라고 함)의 미국 동부(오하이오) 리전에 있는 *MySharedDemoRepo*라는 리포지토에 대한 크로스 계정 액세스를 다른 AWS 계정(AccountB라고 함)의 *DevelopersWithCrossAccountRepositoryAccess*라는 IAM 그룹에 속한 IAM 사용자를 위해 구성할 수 있도록 그 방법과 단계별 지침을 제공합니다.

이 단원은 세 가지 부분으로 나누어져 있습니다.

- AccountA의 관리자를 위한 작업.
- AccountB의 관리자를 위한 작업.
- AccountB의 리포지토리 사용자를 위한 작업.

크로스 계정 액세스를 구성하려면

- AccountA의 관리자는 CodeCommit에서 리포지토리를 생성 및 관리하고 IAM에서 역할을 생성하는 데 필요한 권한을 가진 IAM 사용자로 로그인합니다. 관리형 정책을 사용하고 있는 경우에는 IAMFullAccess 및 AWSCodeCommitFullAccess를 이 IAM 사용자에게 적용합니다.

  AccountA의 예시 계정 ID는 *111122223333*입니다.
- AccountB의 관리자는 IAM 사용자 및 그룹을 생성 및 관리하고 사용자 및 그룹에 대한 정책을 구성하는 데 필요한 권한을 가진 IAM 사용자로 로그인합니다. 관리형 정책을 사용하고 있는 경우에는 IAMFullAccess를 이 IAM 사용자에게 적용합니다.

  AccountB의 예시 계정 ID는 *888888888888*입니다.
- AccountB의 리포지토리 사용자는 개발자 활동을 에뮬레이션하기 위해 AccountA의 CodeCommit 리포지토리 액세스를 허용하기 위해 생성된 IAM 그룹의 멤버인 IAM 사용자로 로그인합니다. 이 계정은 다음으로 구성되어야 합니다.
  - AWS 관리 콘솔 액세스.
  - AWS 리소스에 연결할 때 사용할 액세스 키 및 보안 키와 AccountA의 리포지토리에 액세스할 때 수임할 역할의 ARN입니다.
  - 리포지토리가 복제된 로컬 컴퓨터의 git-remote-codecommit 유틸리티입니다. 이 유틸리티에는 Python과 해당 설치 프로그램인 pip가 필요합니다. Python 패키지 인덱스 웹사이트에서 git-remote-codecommit에서 유틸리티를 다운로드할 수 있습니다.

자세한 내용은 git-remote-codecommit을 AWS CodeCommit 사용하여에 HTTPS 연결을 위한 설정 단계 및 IAM 사용자를 참조하세요.

주제

- 크로스 계정 리포지토리 액세스: AccountA의 관리자를 위한 작업
- 교차 계정 리포지토리 액세스: AccountB의 관리자 작업
- 크로스 계정 리포지토리 액세스: AccountB의 리포지토리 사용자를 위한 작업

크로스 계정 리포지토리 액세스: AccountA의 관리자를 위한 작업

AccountB의 사용자나 그룹이 AccountA의 리포지토리에 액세스할 수 있도록 허용하려면 AccountA 관리자가 다음과 같이 해야 합니다.

- AccountA에서 리포지토리에 대한 액세스 권한을 부여하는 정책을 생성합니다.
- AccountA에서 AccountB의 IAM 사용자 및 그룹이 수임할 수 있는 역할을 생성합니다.
- 책을 역할에 연결합니다.

다음 단원에는 절차와 예제가 나와 있습니다.

주제

- 1단계: AccountA에서 리포지토리 액세스를 위한 정책 생성
- 2단계: AccountA에서 리포지토리 액세스를 위한 역할 생성

1단계: AccountA에서 리포지토리 액세스를 위한 정책 생성

AccountA에서 AccountB의 사용자에게 AccountA의 리포지토리에 대한 액세스 권한을 부여하는 정책을 생성할 수 있습니다. AccountB 허용할 액세스 수준에 따라 다음 중 하나를 수행합니다.

- AccountB 사용자가 특정 리포지토리에 액세스는 할 수 있지만 AccoutA의 모든 리포지토리 목록을 보는 것은 허용하지 않도록 정책을 구성합니다.
- AccountB 사용자가 AccountA의 모든 리포지토리 목록에서 리포지토리를 선택할 수 있도록 추가 액세스를 구성합니다.

리포지토리 액세스를 위한 정책을 생성하려면

1. AccountA에서 정책을 생성할 권한이 있는 IAM 사용자로 AWS Management Console에 로그인합니다.
2. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
3. 탐색 창에서 Policies를 선택합니다.
4. 정책 생성을 선택합니다.
5. JSON 탭에서 다음과 같은 JSON 정책 문서를 JSON 텍스트 상자에 붙여 넣습니다. *us-east-2*를 해당 리포지토리의 AWS 리전 으로, *111122223333*을 AccountA의 계정 ID로, *MySharedDemoRepo*를 AccountA의 CodeCommit 리포지토리 이름으로 교체합니다.

```
{
"Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "codecommit:BatchGet*",
            "codecommit:Create*",
            "codecommit:DeleteBranch",
            "codecommit:Get*",
            "codecommit:List*",
            "codecommit:Describe*",
            "codecommit:Put*",
            "codecommit:Post*",
            "codecommit:Merge*",
            "codecommit:Test*",
            "codecommit:Update*",
            "codecommit:GitPull",
            "codecommit:GitPush"
        ],
        "Resource": [
            "arn:aws:codecommit:us-east-2:111122223333:MySharedDemoRepo"
        ]
    }
]
}
```

이 역할을 수임하는 사용자가 CodeCommit 콘솔 홈 페이지에서 리포지토리 목록을 볼 수 있도록 하고 싶으면 정책에 다음과 같이 설명문을 추가합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codecommit:BatchGet*",
                "codecommit:Create*",
                "codecommit:DeleteBranch",
                "codecommit:Get*",
                "codecommit:List*",
                "codecommit:Describe*",
                "codecommit:Put*",
                "codecommit:Post*",
                "codecommit:Merge*",
                "codecommit:Test*",
                "codecommit:Update*",
                "codecommit:GitPull",
                "codecommit:GitPush"
            ],
            "Resource": [
                "arn:aws:codecommit:us-east-2:111122223333:MySharedDemoRepo"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "codecommit:ListRepositories",
            "Resource": "*"
        }
    ]
}
```

이 액세스 권한은 이 정책을 사용하여 이 역할을 수임하는 사용자가 액세스 권한을 가진 리포지토리를 손쉽게 찾을 수 있도록 해줍니다. 목록에서 리포지토리 이름을 선택하면 공유 리포지토리(Code)의 홈 페이지로 전달됩니다. 사용자는 목록에서 표시되는 다른 리포지토리에 액세스할 수는 없지만, 대시보드 페이지에서 AccountA의 리포지토리를 볼 수 있습니다.

역할을 수임하는 사용자가 AccountA의 모든 리포지토리 목록을 볼 수 있도록 허용하고 싶지 않은 경우에는 첫 번째 정책 예제를 사용하되, 이들 사용자에게 CodeCommit 콘솔의 공유 리포지토리 홈 페이지에 대한 직접 링크를 전송해야 합니다.

6. 정책 검토를 선택합니다. 정책 검사기는 구문 오류를 보고합니다(예를 들어, Amazon Web Services 계정 ID 및 리포지토리 이름을 Amazon Web Services 계정 ID 및 리포지토리 이름으로 교체하는 것을 잊은 경우).
7. 정책 검토 페이지에서 정책 이름(예: *CrossAccountAccessForMySharedDemoRepo*)을 입력합니다. 또한 선택에 따라 이 정책에 대한 설명을 제공할 수 있습니다. 정책 생성을 선택합니다.

2단계: AccountA에서 리포지토리 액세스를 위한 역할 생성

정책을 구성한 후에는 AccountB의 IAM 사용자 및 그룹이 수임할 수 있는 역할을 생성하고 이 역할에 정책을 연결합니다.

리포지토리 액세스를 위한 역할을 생성하려면

1. IAM 콘솔에서 역할을 선택합니다.
2. 역할 생성을 선택합니다.
3. 다른 Amazon Web Services 계정을 선택합니다.
4. 계정 ID에서 AccountB의 *Amazon Web Services ## ID(#: 888888888888*)를 입력합니다. 다음: 권한을 선택합니다.
5. 정책 권한 연결에서 이전 절차에서 생성한 정책 (*CrossAccountAccessForMySharedDemoRepo*)을 선택합니다. (다음: 검토)를 선택합니다.
6. 역할 이름에 역할 이름(예: *MyCrossAccountRepositoryContributorRole*)을 입력합니다. 또한 선택에 따라 설명을 입력하여 다른 사용자가 역할의 목적을 이해하도록 도울 수 있습니다.
7. 역할 생성을 선택합니다.
8. 방금 생성한 역할을 열고 역할 ARN(예: `arn:aws:iam::`*111122223333*`:role/`*MyCrossAccountRepositoryContributorRole*) 을 복사합니다. AccountB 관리자에게 이 ARN을 제공해야 합니다.

교차 계정 리포지토리 액세스: AccountB의 관리자 작업

AccountB의 사용자나 그룹이 AccountA의 리포지토리에 액세스할 수 있도록 허용하려면 AccountB 관리자가 AccountB에 그룹을 생성해야 합니다. 그룹 구성원들이 AccountA 관리자가 생성한 역할을 수임하도록 허용하는 정책을 통해 이 그룹을 구성해야 합니다.

다음 단원에는 절차와 예제가 나와 있습니다.

주제

- 1단계: AccountB 사용자의 리포지토리 액세스를 위한 IAM 그룹 생성
- 2단계: 정책을 생성하고 IAM 그룹에 사용자 추가

1단계: AccountB 사용자의 리포지토리 액세스를 위한 IAM 그룹 생성

AccountA 리포지토리에 액세스할 수 있는 AccountB의 IAM 사용자를 관리하는 가장 간단한 방법은 AccountA에서 역할을 수임할 권한을 가진 IAM 그룹을 AccountB에 생성한 다음 해당 그룹에 IAM 사용자를 추가하는 것입니다.

교차 계정 리포지토리 액세스를 위한 그룹을 생성하려면

1. IAM 그룹 및 정책을 생성하고 AccountB에서 IAM 사용자를 관리하는 데 필요한 권한을 가진 IAM 사용자로 AWS Management Console에 로그인합니다.
2. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
3. IAM 콘솔에서 역할을 선택합니다.
4. 새 그룹 생성을 선택합니다.
5. 그룹 이름에 해당 그룹의 이름(예: *DevelopersWithCrossAccountRepositoryAccess*)을 입력합니다. 다음 단계를 선택합니다.
6. 정책 연결에서 다음 단계를 선택합니다. 다음 단계에서 크로스 계정 정책을 생성합니다. 그룹 생성을 완료합니다.

2단계: 정책을 생성하고 IAM 그룹에 사용자 추가

그룹을 생성했다면 이제는 이 그룹의 구성원들이 AccountA의 리포지토리에 대한 액세스 권한을 부여하는 역할을 수임하도록 허용하는 정책을 생성합니다. 그런 다음, 그룹에 AccountA에서 액세스를 허용할 AccountB의 IAM 사용자를 추가합니다.

그룹에 대한 정책을 생성하고 여기에 사용자를 추가하려면

1. IAM 콘솔에서 그룹을 선택한 다음, 방금 생성한 그룹의 이름(예: *DevelopersWithCrossAccountRepositoryAccess*)을 선택합니다.
2. 권한 탭을 선택합니다. 인라인 정책을 확장하고 링크를 선택하여 인라인 정책을 생성합니다 (이미 인라인 정책을 가지고 있는 그룹을 구성 중인 경우에는 그룹 정책 생성을 선택).
3. 사용자 지정 정책을 선택한 후 선택을 선택합니다.
4. 정책 이름에서 정책의 이름(예: *AccessPolicyForSharedRepository*)을 입력합니다.

5. 정책 문서에 다음 정책을 붙여 넣습니다. `Resource`에서 ARN을 AccountA에서 관리자가 생성한 정책의 ARN(예: arn:aws:iam::*111122223333*:role/*MyCrossAccountRepositoryContributorRole*)으로 바꾼 다음 정책 적용을 선택합니다. AccountA에서 관리자가 생성한 정책에 대한 자세한 내용은 1단계: AccountA에서 리포지토리 액세스를 위한 정책 생성 단원을 참조하세요.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource":
 "arn:aws:iam::111122223333:role/MyCrossAccountRepositoryContributorRole"
  }
}
```

6. 사용자 탭을 선택합니다. 그룹에 사용자 추가를 선택한 다음 AccountB IAM 사용자를 추가합니다. 예를 들어, 사용자 이름이 *Saanvi_Sarkar*인 IAM 사용자를 그룹에 추가할 수 있습니다.

> **Note**
>
> AccountB의 사용자는 액세스 키 및 비밀 키를 포함하여 공유 CodeCommit 리포지토리에 액세스할 수 있도록 로컬 컴퓨터를 구성하기 위한 프로그래밍 방식의 액세스 권한을 가져야 합니다. IAM 사용자를 생성 중인 경우에는 액세스 키와 비밀 키를 반드시 저장해야 합니다. AWS 계정의 보안을 유지하기 위해 비밀 액세스 키는 생성 시에만 액세스할 수 있습니다.

크로스 계정 리포지토리 액세스: AccountB의 리포지토리 사용자를 위한 작업

AccountA의 리포지토리에 액세스하려면 AccountB 그룹의 사용자가 리포지토리 액세스가 가능하도록 로컬 컴퓨터를 구성해야 합니다. 다음 단원에는 절차와 예제가 나와 있습니다.

주제

- 1단계: AccountB 사용자가 AccountA의 리포지토리에 액세스하도록 AWS CLI 및 Git 구성
- 2단계: AccountA의 CodeCommit 리포지토리 복제 및 액세스

1단계: AccountB 사용자가 AccountA의 리포지토리에 액세스하도록 AWS CLI 및 Git 구성

또 다른 Amazon Web Services 계정에서 리포지토리에 액세스하기 위해 SSH 키나 Git 보안 인증 정보를 사용할 수 없습니다. AccountB 사용자는 git-remote-codecommit(권장) 또는 보안 인증 도우미를 사용하여 AccountA의 공유 CodeCommit 리포지토리에 액세스하도록 컴퓨터를 구성해야 합니다. 그러나 AccountB에서 리포지토리에 액세스할 때 SSH 키나 Git 자격 증명을 계속 사용할 수 있습니다.

git-remote-codecommit를 사용하여 액세스를 구성하려면 다음 단계를 수행합니다. git-remote-codecommit를 아직 설치하지 않은 경우 Python 패키지 인덱스 웹사이트의 git-remote-codecommit에서 다운로드하세요.

교차 계정 액세스를 위해 AWS CLI 및 Git를 구성하려면

1. 로컬 컴퓨터에 AWS CLI 를 설치합니다. AWS CLI설치 단원의 운영 체제 지침을 참조하세요.
2. 로컬 컴퓨터에 Git를 설치합니다. Git를 설치하려면 Git 다운로드 또는 Windows용 Git 같은 웹 사이트를 이용하세요.

> **Note**
>
> CodeCommit은 Git 버전 1.7.9 이상을 지원합니다. Git 버전 2.28에서는 초기 커밋을 위한 브랜드 이름을 구성할 수 있습니다. 최신 버전의 Git를 사용하는 것이 좋습니다. Git은 계속 개선되며 정기적으로 업데이트되는 플랫폼입니다. 기능 변경에 따라 CodeCommit 사용 방식이 달라지는 경우가 있습니다. Git과 CodeCommit의 특정 버전에서 문제가 발생할 경우 문제 해결의 내용을 검토하세요.

3. 리포지토리를 복제할 디렉터리 위치의 터미널이나 명령줄에서 git config --local user.name 및 git config --local user.email 명령을 실행하여 리포지토리에 수행할 커밋의 사용자 이름과 이메일을 설정합니다. 예시:

```
git config --local user.name "Saanvi Sarkar"
git config --local user.email saanvi_sarkar@example.com
```

   이 명령은 어떤 결과도 반환하지 않지만, 지정한 이메일 및 사용자 이름은 AccountA의 리포지토리에서 수행하는 커밋에 연결됩니다.

4. aws configure --profile 명령을 실행하여 AccountB의 리소스에 연결할 때 사용할 기본 프로필을 구성합니다. 메시지가 표시되면 IAM 사용자의 액세스 키와 비밀 키를 입력합니다.

> **Note**
>
> 를 이미 설치하고 프로파일을 AWS CLI 구성한 경우이 단계를 건너뛸 수 있습니다.

예를 들어 다음 명령을 실행하여 미국 동부(오하이오)(us-east-2)의 AccountB에서 AWS 리소스에 액세스하는 데 사용하는 기본 AWS CLI 프로필을 생성합니다.

```
aws configure
```

요청 메시지가 나타나면 다음 정보를 입력합니다.

```
AWS Access Key ID [None]: Your-IAM-User-Access-Key
AWS Secret Access Key ID [None]: Your-IAM-User-Secret-Access-Key
Default region name ID [None]: us-east-2
Default output format [None]: json
```

5. aws configure --profile 명령을 다시 실행하여 AccountA의 리포지토리에 연결할 때 사용할 명명된 프로필을 구성합니다. 메시지가 표시되면 IAM 사용자의 액세스 키와 비밀 키를 입력합니다. 예를 들어, 다음 명령을 실행하여 *Account Account Profile*이라는 AWS CLI 프로필을 생성합니다. 이 프로필은 미국 동부(오하이오)(us-east-2)의 AccountA에 있는 리포지토리에 액세스하는 데 사용합니다.

```
aws configure --profile MyCrossAccountAccessProfile
```

요청 메시지가 나타나면 다음 정보를 입력합니다.

```
AWS Access Key ID [None]: Your-IAM-User-Access-Key
AWS Secret Access Key ID [None]: Your-IAM-User-Secret-Access-Key
Default region name ID [None]: us-east-2
Default output format [None]: json
```

6. 일반 텍스트 편집기에서 AWS CLI 구성 파일이라고도 하는 `config` 파일을 엽니다. 운영 체제에 따라 이 파일의 위치는 다를 수 있습니다. Linux, macOS, Unix의 경우 `~/.aws/config`에, Windows의 경우 *drive*:\Users*USERNAME*\.aws\config에 위치할 수 있습니다.
7. 파일에서 AccountB의 리포지토리에 액세스하도록 구성한 기본 프로파일에 해당하는 항목을 찾습니다. 예를 들면 다음과 같아야 합니다.

```
[default]
region = us-east-2
output = json
```

`account`를 프로파일 구성에 추가합니다. AccountB의 AWS 계정 ID를 제공합니다. 예시:

```
[default]
account = 888888888888
region = us-east-2
output = json
```

8. 파일에서 방금 생성한 *MyCrossAccountAccessProfile* 프로필에 해당되는 항목을 찾습니다. 예를 들면 다음과 같아야 합니다.

```
[profile MyCrossAccountAccessProfile]
region = us-east-2
output = json
```

`account`, `role_arn` 및 `source_profile`을 프로파일 구성에 추가합니다. AccountA의 Amazon Web Services 계정 ID, 다른 계정의 리포지토리에 액세스하기 위해 AccountA에서 수임하는 역할 ARN, 그리고 AccountB의 기본 AWS CLI 프로파일의 이름을 입력합니다. 예시:

```
[profile MyCrossAccountAccessProfile]
region = us-east-2
account = 111122223333
role_arn = arn:aws:iam::111122223333:role/MyCrossAccountRepositoryContributorRole
source_profile = default
output = json
```

변경 사항을 저장하고 일반 텍스트 편집기를 닫습니다.

2단계: AccountA의 CodeCommit 리포지토리 복제 및 액세스

크로스 계정 CodeCommit 리포지토리를 복제하거나 그 리포지토리에 푸시하거나 그 리포지토리에서 가져오려면 git clone, git push, git pull 명령을 실행합니다. AWS 관리 콘솔에 로그인하고, 역할을 전환하고, CodeCommit 콘솔을 사용하여 다른 계정의 리포지토리와 상호 작용할 수도 있습니다.

> **Note**
>
> IAM 역할이 어떻게 구성되었는지에 따라 CodeCommit의 기본 페이지에서 리포지토리를 확인할 수 있습니다. 리포지토리를 볼 수 없는 경우에는 리포지토리 관리자에게 CodeCommit 콘솔에 있는 공유 리포지토리의 코드 페이지로 연결되는 URL 링크를 이메일로 보내달라고 요청하세요. URL은 다음과 비슷합니다.
>
> ```
> https://console.aws.amazon.com/codecommit/home?region=us-east-2#/
> repository/MySharedDemoRepo/browse/HEAD/--/
> ```

교차 계정 리포지토리를 로컬 컴퓨터에 복제하려면

1. 리포지토리를 복제할 디렉터리의 명령줄 또는 터미널에서 HTTPS (GRC) 복제 URL을 사용하여 git clone 명령을 실행합니다. 예시:

   ```
   git clone codecommit://MyCrossAccountAccessProfile@MySharedDemoRepo
   ```

   달리 지정하지 않는 한, 리포지토리는 리포지토리와 같은 이름으로 하위 디렉터리에 복제됩니다.
2. 복제된 리포지토리로 디렉터리를 변경한 다음, 파일을 추가하거나 변경합니다. 예를 들어 *NewFile.txt*라는 이름의 파일을 추가할 수 있습니다.
3. 로컬 리포지토리에서 추적한 변경 사항에 파일을 추가하고 변경 내용을 커밋한 다음 CodeCommit 리포지토리로 파일을 푸시합니다. 예시:

   ```
   git add NewFile.txt
   git commit -m "Added a file to test cross-account access to this repository"
   git push
   ```

   자세한 내용은 Git 및 시작하기 AWS CodeCommit 단원을 참조하십시오.

파일을 추가했다면 CodeCommit 콘솔로 이동하여 커밋을 확인하고 리포지토리에 대한 다른 사용자의 변경 사항을 검토하며 pull 요청에 참여합니다.

CodeCommit 콘솔에서 크로스 계정 리포지토리에 액세스하려면

1. AccountA의 리포지토리 AWS Management Console 에 대한 교차 계정 액세스 권한이 부여된 IAM 사용자로 AccountA(*888888888888*)의에 로그인합니다. AccountB

2. 탐색 표시줄에서 사용자 이름을 선택하고 드롭다운 메뉴로 가서 역할 전환을 선택합니다.

> **Note**
>
> 이 옵션을 선택한 것이 처음인 경우에는 해당 페이지에서 정보를 검토하고 다시 역할 전환을 선택합니다.

3. 역할 전환 페이지에서 다음을 수행합니다.

   - 계정에서 AccountA의 계정 ID(예: *111122223333*))를 입력합니다.
   - 역할에서 AccountA의 리포지토리에 액세스하기 위해 수임하려는 역할의 이름(예: *MyCrossAccountRepositoryContributorRole*)을 입력합니다.
   - 표시 이름에 이 역할의 표시 이름을 입력합니다. 이 이름은 역할을 수임할 때 콘솔에 표시됩니다. 또한 나중에 콘솔에서 역할을 전환할 때 수임된 역할 목록에 나타납니다.
   - (선택 사항) 색상에서 디스플레이 이름의 컬러 레이블을 선택합니다.
   - 역할 전환을 선택합니다.

   자세한 내용은 역할 전환(AWS Management Console) 단원을 참조하세요.

4. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.

   수임된 역할이 AccountA의 리포지토리 이름을 볼 수 있는 권한을 가지고 있는 경우에는 리포지토리 목록과 상태 보기 권한이 없음을 알리는 오류 메시지가 표시됩니다. 이는 예상된 동작입니다. 목록에서 공유 리포지토리의 이름을 선택합니다.

   수임된 역할이 AccountA의 리포지토리 이름을 볼 수 있는 권한이 없는 경우에는 오류 메시지와 리포지토리가 없이 빈 목록이 표시됩니다. 리포지토리에 대한 URL 링크를 붙여 넣거나 콘솔 링크를 수정한 다음, `/list`를 공유 리포지토리 이름(예: `/`*MySharedDemoRepo*)으로 변경합니다.

5. 코드에서 로컬 컴퓨터에서 추가한 파일 이름을 찾습니다. 파일에서 코드를 검색한 다음, 리포지토리의 나머지 부분을 검색하고 기능을 사용하기 시작하려면 이를 선택합니다.

   자세한 내용은 시작하기 AWS CodeCommit 단원을 참조하십시오.

AWS CodeCommit 리포지토리 삭제

CodeCommit 콘솔 또는를 사용하여 CodeCommit AWS CLI 리포지토리를 삭제할 수 있습니다.

> **Note**
> 리포지토리를 삭제한다고 해서 해당 리포지토리의 모든 로컬 사본(로컬 리포지토리)이 삭제되는 것은 아닙니다. 로컬 리포지토리를 삭제하려면 로컬 시스템의 디렉터리와 파일 관리 도구를 사용하세요.

주제

- CodeCommit 리포지토리 삭제 (콘솔)
- 로컬 리포지토리 삭제
- CodeCommit 리포지토리 삭제 (AWS CLI)

CodeCommit 리포지토리 삭제 (콘솔)

CodeCommit 콘솔을 사용하여 CodeCommit 리포지토리를 삭제하려면 다음 단계를 따르세요.

> **Important**
> CodeCommit 리포지토리를 삭제한 후에는 이를 어떠한 로컬 리포지토리 또는 공유 리포지토리에도 복제할 수 없습니다. 또한 어떠한 로컬 리포지토리 또는 공유 리포지토리를 대상으로도 데이터를 풀하거나 푸시할 수 없습니다. 이 작업은 실행을 취소할 수 없습니다.

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서 삭제할 리포지토리의 이름을 선택합니다.
3. 탐색 창에서 설정을 선택합니다.
4. 일반 탭의 리포지토리 삭제에서 리포지토리 삭제를 선택합니다. **`delete`**를 입력한 다음 삭제를 선택합니다. 이로써 리포지토리가 영구적으로 삭제됩니다.

   > **Note**
   > CodeCommit에서 리포지토리를 삭제하더라도 모든 로컬 리포지토리가 삭제되지는 않습니다.

로컬 리포지토리 삭제

로컬 시스템의 디렉터리와 파일 관리 도구를 사용하여 로컬 리포지토리가 포함된 디렉터리를 삭제합니다.

로컬 리포지토리를 삭제한다고 해서 이에 연결되었을 가능성이 있는 CodeCommit 리포지토리까지 삭제되지는 않습니다.

CodeCommit 리포지토리 삭제 (AWS CLI)

CodeCommit에서 AWS CLI 명령을 사용하려면를 설치합니다 AWS CLI. 자세한 내용은 명령줄 참조 단원을 참조하십시오.

AWS CLI 를 사용하여 CodeCommit 리포지토리를 삭제하려면 delete-repository 명령을 실행하여 삭제할 CodeCommit 리포지토리의 이름을 지정합니다(`--repository-name` 옵션 사용).

> **Important**
>
> CodeCommit 리포지토리를 삭제한 후에는 이를 어떠한 로컬 리포지토리 또는 공유 리포지토리에도 복제할 수 없습니다. 또한 어떠한 로컬 리포지토리 또는 공유 리포지토리를 대상으로도 데이터를 풀하거나 푸시할 수 없습니다. 이 작업은 실행을 취소할 수 없습니다.

> **Tip**
>
> CodeCommit 리포지토리의 이름을 가져오려면 list-repositories 명령을 실행합니다.

예를 들어 `MyDemoRepo`이라는 리포지토리를 삭제하려면 다음 명령을 실행합니다.

```
aws codecommit delete-repository --repository-name MyDemoRepo
```

명령이 제대로 실행되면 영구 삭제된 CodeCommit 리포지토리의 ID가 다음과 같이 출력됩니다.

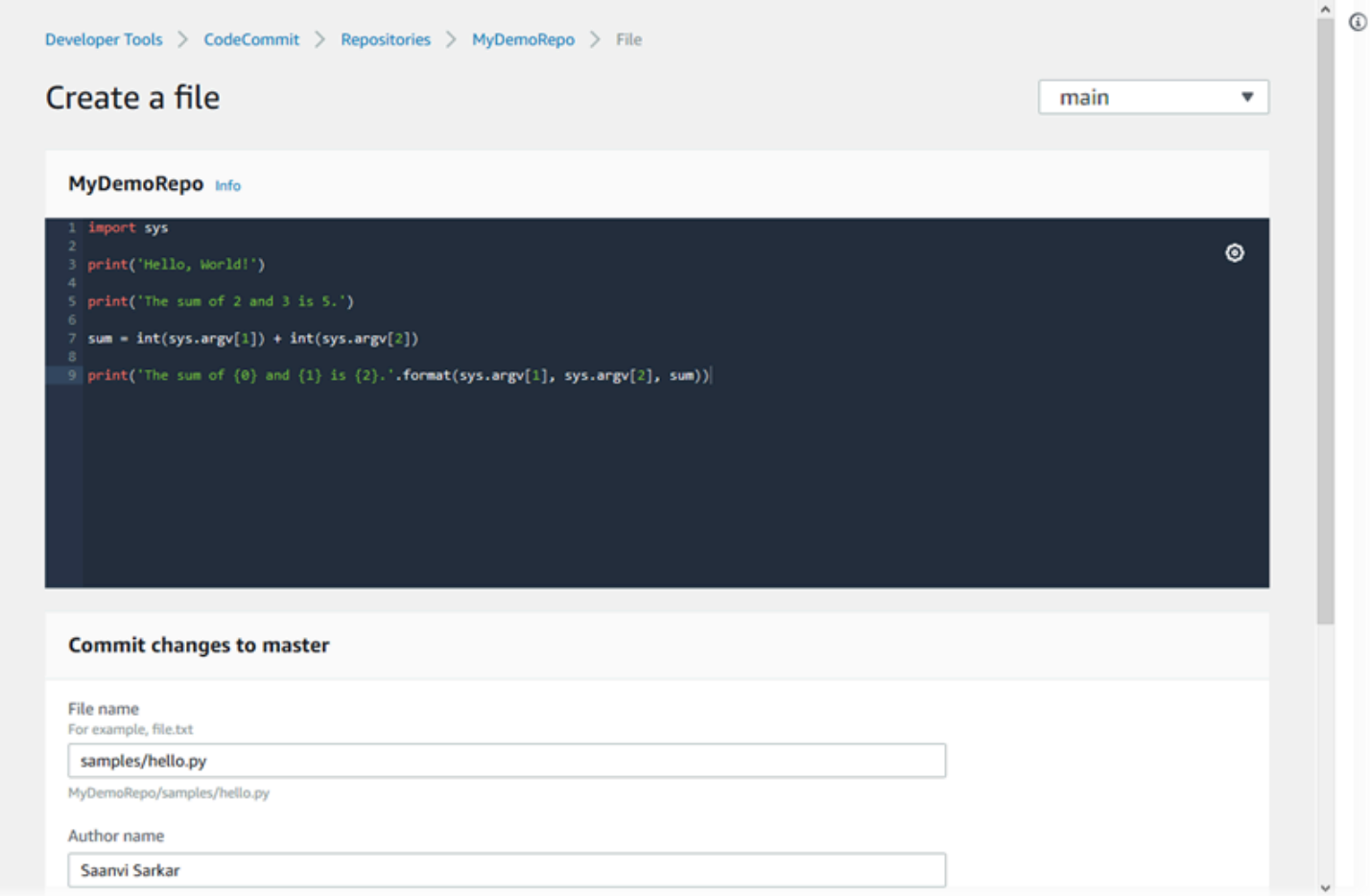
```
{
    "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE"
}
```

CodeCommit 리포지토리를 삭제한다고 해서 이에 연결되었을 가능성이 있는 로컬 리포지토리까지 삭제되지는 않습니다.

AWS CodeCommit 리포지토리의 파일 작업

CodeCommit에서 파일은 버전이 관리되고 독립적으로 작동하는 정보입니다. 본래의 사용자 외에, 해당 파일이 저장된 리포지토리와 브렌치의 다른 사용자들도 이용할 수 있습니다. 사용자는 개인용 컴퓨터처럼 디렉터리 구조를 활용하여 리포지토리 파일을 정리할 수 있습니다. CodeCommit은 개인용 컴퓨터와 달리 파일에 대한 모든 변경 내용을 자동으로 추적합니다. 사용자는 파일의 버전을 비교할 수 있으며 서로 다른 버전의 파일을 서로 다른 리포지토리 브랜치에 저장할 수 있습니다.

리포지토리에 파일을 추가하거나 기존 파일을 편집할 때는 Git 클라이언트를 활용합니다. CodeCommit 콘솔 AWS CLI, 또는 CodeCommit API를 사용할 수도 있습니다.



CodeCommit에서 리포지토리의 다른 측면을 다루는 방식에 대해 자세히 알아보려면 리포지토리 작업, 풀 요청 작업, 브랜치 작업, 커밋 작업, 사용자 기본 설정으로 작업 섹션을 참조하세요.

주제

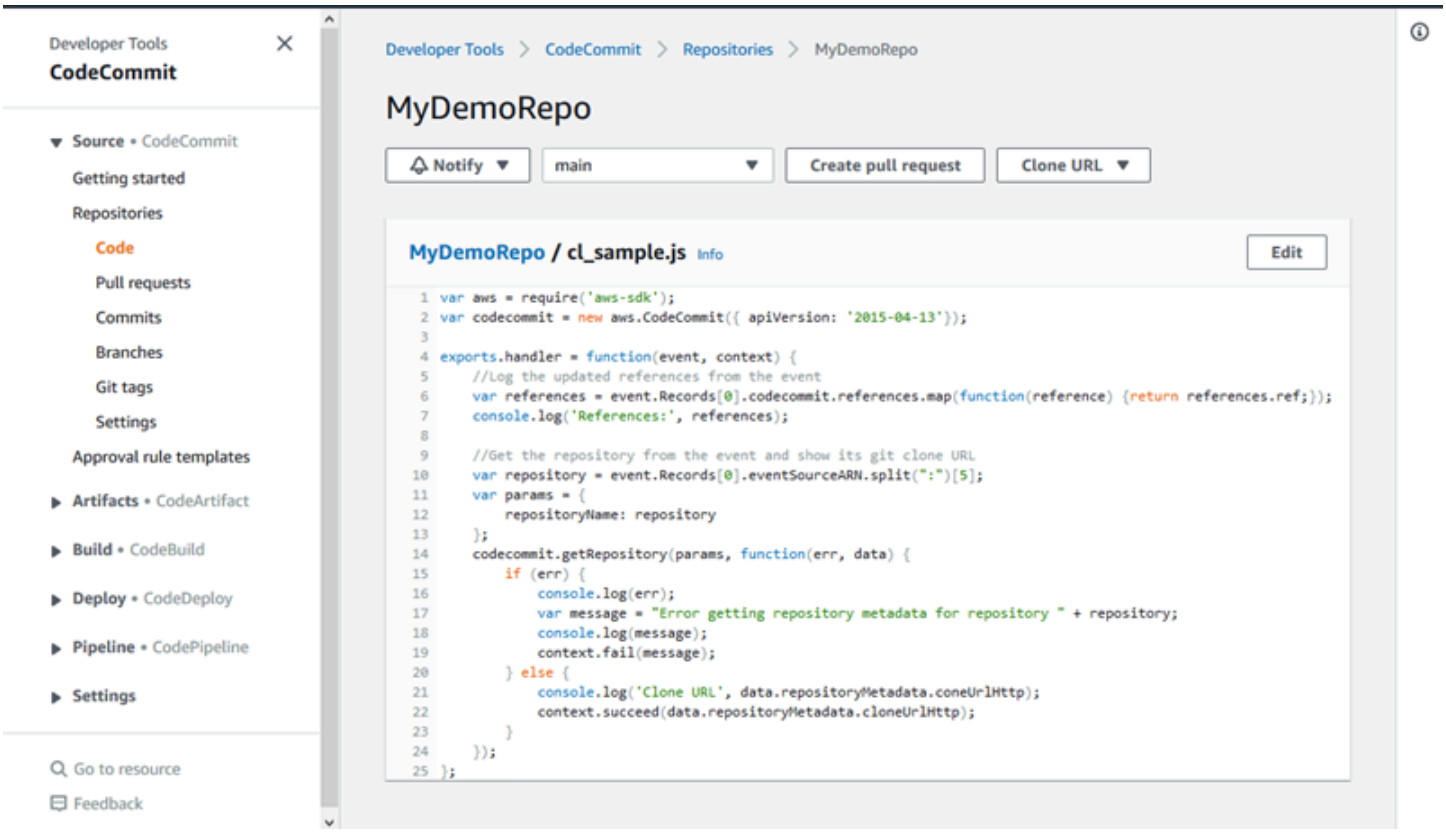
- AWS CodeCommit 리포지토리에서 파일 찾아보기
- AWS CodeCommit 리포지토리에 파일 생성 또는 추가

- AWS CodeCommit 리포지토리에서 파일의 콘텐츠 편집

AWS CodeCommit 리포지토리에서 파일 찾아보기

CodeCommit 리포지토리에 연결한 후, CodeCommit 리포지토리를 로컬 리포지토리에 복제하거나 CodeCommit 콘솔을 사용하여 그 콘텐츠를 검색할 수 있습니다. 이 주제에서는 CodeCommit 콘솔을 사용하여 CodeCommit 리포지토리의 콘텐츠를 검색하는 방법에 대해 설명합니다.

> **Note**
>
> 활성 CodeCommit 사용자는 CodeCommit 콘솔에서 코드를 검색하는 데 요금이 부과되지 않습니다. 요금이 부과될 수 있는 상황에 대해 자세히 알아보려면 요금을 참조하세요.



CodeCommit 리포지토리 탐색

CodeCommit 콘솔을 사용하여 특정 리포지토리에 포함된 파일을 검토하거나 파일의 콘텐츠를 빠르게 훑어볼 수 있습니다.

리포지토리의 콘텐츠를 검색하려면

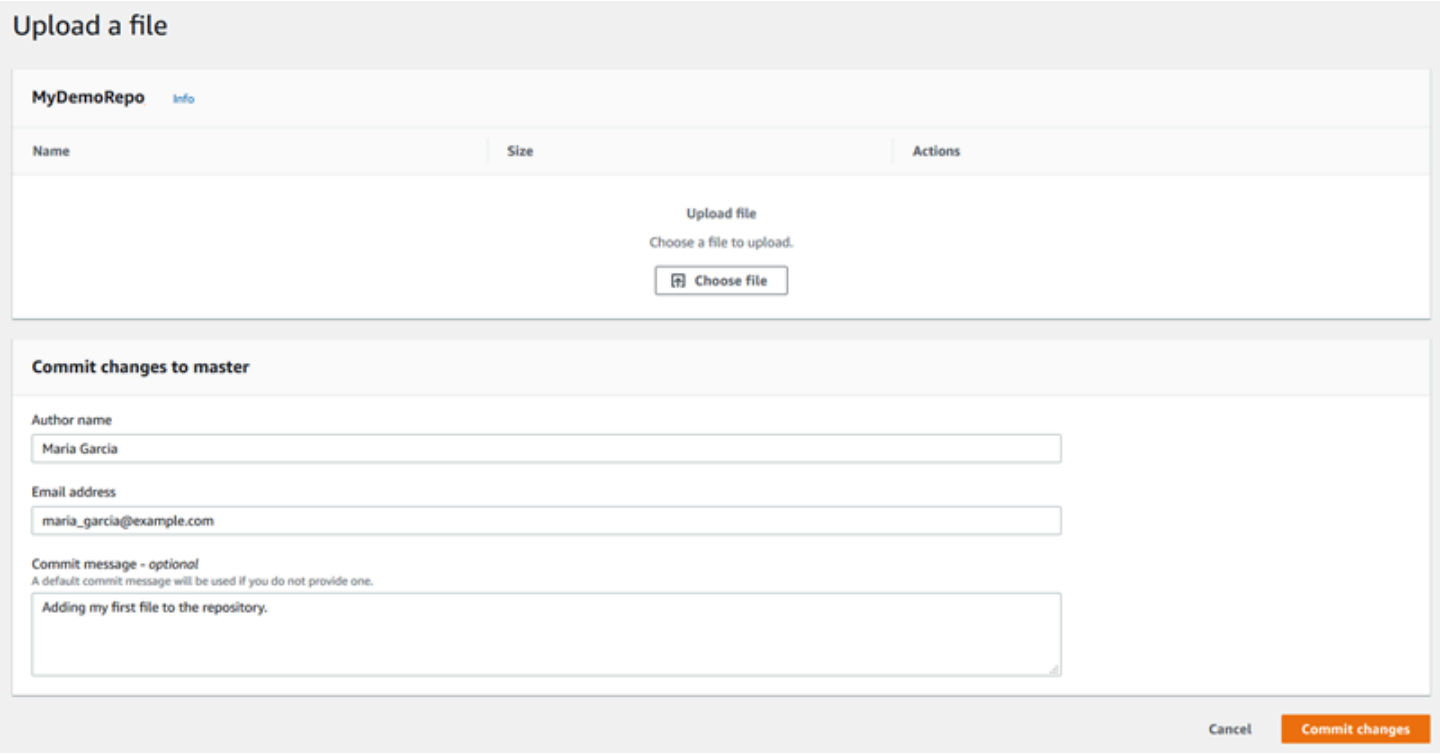
1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리 페이지의 리포지토리 목록에서 검색하려는 리포지토리를 선택합니다.
3. 코드 보기에서 리포지토리의 기본 브랜치의 콘텐츠를 검색합니다.

   보기를 다른 브랜치나 태그로 변경하려면 보기 선택 버튼을 선택합니다. 드롭다운 목록에서 브랜치나 태그 이름을 선택하거나 필터 상자에서 브랜치나 태그 이름을 입력한 다음, 목록에서 선택합니다.
4. 다음 중 하나를 수행합니다.
   - 디렉터리의 콘텐츠를 보려면 목록에서 선택합니다. 탐색 목록에서 아무 디렉터리나 선택하면 해당 디렉터리 보기로 돌아갈 수 있습니다. 디렉토리 목록 상단에 있는 위쪽 화살표를 활용할 수도 있습니다.
   - 파일의 콘텐츠를 보려면 목록에서 선택합니다. 파일이 커밋 객체 한도보다 크면 콘솔에 표시할 수 없으며, 해당 파일은 대신 로컬 리포지토리에서 봐야 합니다. 자세한 내용은 할당량 단원을 참조하십시오. 파일 보기를 종료하려면, 코드 탐색 표시줄에서 살펴보고 싶은 디렉터리를 선택합니다.

> **Note**
>
> 콘솔에서 모든 바이너리 파일을 볼 수 있는 것은 아닙니다. 이진 파일을 선택하고 그 파일을 잠재적으로 볼 수 있는 경우, 콘텐츠를 표시할 것인지 여부를 확인하라는 경고 메시지가 나타납니다. 파일을 보려면 파일 콘텐츠 표시를 선택합니다. 파일을 보지 않으려면 코드 탐색 표시줄에서 살펴보고 싶은 디렉터리를 선택합니다.
> 마크다운 파일(.md)을 선택한 경우, 렌더링된 마크다운 및 마크다운 소스 버튼을 사용하여 렌더링된 뷰와 구문 뷰 사이를 전환할 수 있습니다. 자세한 내용은 콘솔에서 마크다운 사용하기를 참조하세요.

AWS CodeCommit 리포지토리에 파일 생성 또는 추가

CodeCommit 콘솔 AWS CLI또는 Git 클라이언트를 사용하여 리포지토리에 파일을 추가할 수 있습니다. 로컬 컴퓨터에서 리포지토리로 파일을 업로드하거나, 콘솔의 코드 편집기를 사용하여 파일을 생성할 수 있습니다. 편집기를 사용하면 readme.md 파일과 같은 간단한 파일을 리포지토리의 브랜치에 빠르고 쉽게 추가할 수 있습니다.



주제

- 파일 생성 또는 업로드 (콘솔)
- 파일 추가 (AWS CLI)
- 파일 추가 (Git)

파일 생성 또는 업로드 (콘솔)

CodeCommit 콘솔을 사용하여 파일을 생성하고 CodeCommit 리포지토리의 브랜치에 추가할 수 있습니다. 파일 생성 과정 중에 사용자 이름과 이메일 주소를 입력할 수 있습니다. 또한 파일을 추가한 사용자와 그 이유를 다른 사용자가 파악할 수 있도록 커밋 메시지를 추가할 수도 있습니다. 로컬 컴퓨터에서 리포지토리의 브랜치로 직접 파일을 업로드할 수도 있습니다.

리포지토리에 파일을 추가하려면

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서 파일을 추가할 리포지토리를 선택합니다.
3. 코드 보기에서 파일을 추가할 브랜치를 선택합니다. 코드 보기를 열면 기본 브랜치의 콘텐츠가 표시되도록 기본 설정되어 있습니다.

보기를 다른 브랜치로 변경하려면 보기 선택 버튼을 선택합니다. 드롭다운 목록에서 브랜치 이름을 선택하거나 필터 상자에서 브랜치 이름을 입력한 다음, 목록에서 선택합니다.

4. 파일 추가를 선택하고 다음 옵션 중 하나를 선택합니다.
   - 코드 편집기를 사용하여 파일의 콘텐츠를 만들고 리포지토리에 추가하려면 파일 생성을 선택합니다.
   - 로컬 컴퓨터에서 리포지토리로 파일을 업로드하려면 파일 업로드를 선택합니다.
5. 이 파일을 리포지토리에 추가한 사람과 그 이유에 대한 정보를 다른 사용자에게 제공합니다.
   - 작성자 이름에 이름을 입력합니다. 이 이름은 커밋 정보에서 작성자 이름과 커미터 이름으로 모두 사용됩니다. CodeCommit은 IAM 사용자 이름이나 콘솔 로그인의 파생물을 작성자 이름으로 사용하도록 기본 설정되어 있습니다.
   - 이메일 주소에는 다른 리포지토리 사용자가 이러한 변경에 대해 문의할 수 있도록 이메일 주소를 입력합니다.
   - 커밋 메시지에는 간단한 설명을 입력합니다. 이는 선택 사항이지만 적극 권장됩니다. 입력하지 않을 경우, 기본 커밋 메시지가 사용됩니다.
6. 다음 중 하나를 수행합니다.
   - 파일을 업로드하는 경우에는 로컬 컴퓨터에서 파일을 선택합니다.
   - 파일을 생성하려는 경우에는 추가할 콘텐츠를 코드 편집기에서 입력한 다음 파일의 이름을 제공합니다.
7. 변경 사항 커밋을 선택합니다.

파일 추가 (AWS CLI)

AWS CLI 및 put-file 명령을 사용하여 CodeCommit 리포지토리에 파일을 추가할 수 있습니다. put-file 명령을 사용하여 파일의 디렉터리 또는 경로 구조를 추가할 수도 있습니다.

> **Note**
>
> CodeCommit에서 AWS CLI 명령을 사용하려면를 설치합니다 AWS CLI. 자세한 내용은 명령줄 참조 단원을 참조하십시오.

리포지토리에 파일을 추가하려면

1. 로컬 컴퓨터에서, CodeCommit 리포지토리에 추가할 파일을 생성합니다.
2. 터미널 또는 명령줄에서 다음을 지정하여 put-file 명령을 실행합니다.

   - 파일을 추가할 리포지토리.
   - 파일을 추가할 브랜치.
   - 해당 브랜치에서 수행된 가장 최근 커밋의 축약되지 않은 커밋 ID(팁 또는 헤드 커밋)
   - 파일의 로컬 위치. 이 위치에 사용되는 구문은 로컬 운영 체제에 따라 달라집니다.
   - 업데이트된 파일이 리포지토리에 저장된 경로(있는 경우)를 포함하여 추가할 파일의 이름.
   - 이 파일에 연결할 사용자 이름과 이메일.
   - 이 파일을 추가한 이유를 설명하는 커밋 메시지.

   사용자 이름, 이메일 주소 및 커밋 메시지는 선택 사항이지만, 다른 사용자에게 변경을 수행한 사용자와 변경 이유를 알려줍니다. 사용자 이름을 제공하지 않는 경우, CodeCommit은 IAM 사용자 이름이나 콘솔 로그인 파생물을 작성자 이름으로 사용하도록 기본 설정되어 있습니다.

   *## ##, MyDemorepo## ##### ## ## ## ID# 4c925148EXAMPLE*이고 이름이 *feature-randomizationfeature*인 브랜치에 *ExampleSolution.py*라는 파일을 추가하려면 다음과 같이 합니다.

```
aws codecommit put-file --repository-name MyDemoRepo --branch-name feature-
randomizationfeature --file-content file://MyDirectory/ExampleSolution.py --file-
path /solutions/ExampleSolution.py --parent-commit-id 4c925148EXAMPLE --name "María
 García" --email "maría_garcía@example.com" --commit-message "I added a third
 randomization routine."
```

> **Note**
>
> 바이너리 파일을 추가할 때는 반드시 `fileb://`를 사용하여 파일의 로컬 위치를 지정해야 합니다.

   이 명령이 제대로 실행되면 다음과 비슷한 출력이 반환됩니다.

```
{
    "blobId": "2eb4af3bEXAMPLE",
```

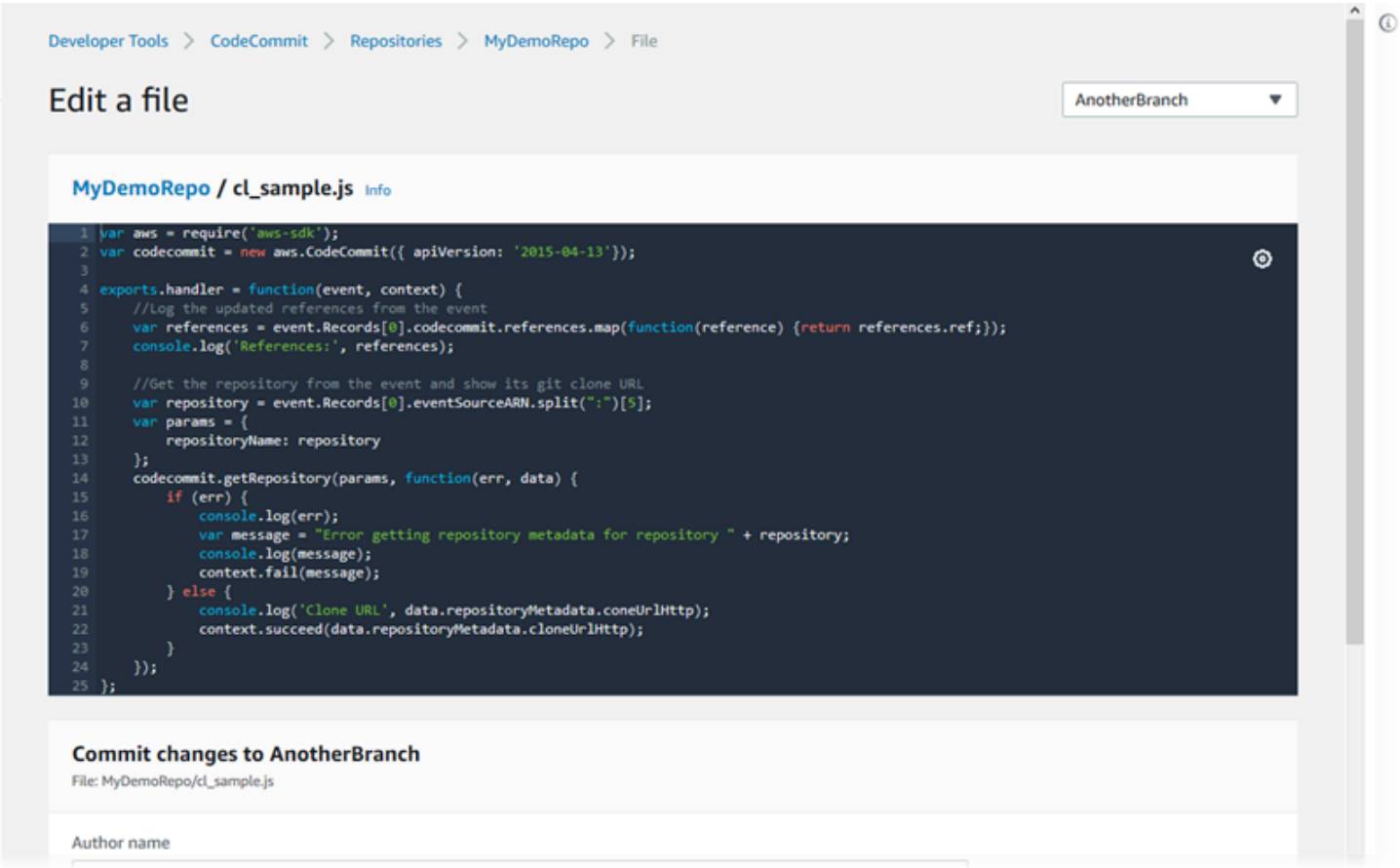
```
    "commitId": "317f8570EXAMPLE",
    "treeId": "347a3408EXAMPLE"
}
```

파일 추가 (Git)

로컬 리포지토리에서 파일을 추가해서 CodeCommit 리포지토리에 변경 내용을 푸시할 수 있습니다. 자세한 내용은 Git 및 시작하기 AWS CodeCommit 단원을 참조하십시오.

AWS CodeCommit 리포지토리에서 파일의 콘텐츠 편집

CodeCommit 콘솔 AWS CLI또는 Git 클라이언트를 사용하여 CodeCommit 리포지토리에 있는 파일의 콘텐츠를 편집할 수 있습니다.



주제

- 파일 편집 (콘솔)
- 파일 편집 또는 삭제 (AWS CLI)

- 파일 편집 (Git)

파일 편집 (콘솔)

CodeCommit 콘솔을 사용하여 CodeCommit 리포지토리의 브랜치에 추가된 파일을 편집할 수 있습니다. 파일 편집 프로세스의 한 부분으로 사용자 이름과 이메일 주소를 입력할 수 있습니다. 또한 다른 사용자가 변경을 수행한 사용자와 변경 이유를 이해할 수 있도록 커밋 메시지를 추가할 수도 있습니다.

리포지토리에서 파일을 편집하려면

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서 파일을 편집할 리포지토리를 선택합니다.
3. 코드 보기에서 파일을 편집할 브랜치를 선택합니다. 코드 보기를 열면 기본 브랜치의 콘텐츠가 표시되도록 기본 설정되어 있습니다.

   보기를 다른 브랜치로 변경하려면 보기 선택 버튼을 선택합니다. 드롭다운 목록에서 브랜치 이름을 선택하거나 필터 상자에서 브랜치 이름을 입력한 다음, 목록에서 선택합니다.
4. 브랜치의 콘텐츠를 검색하고 편집할 파일을 선택합니다. 파일 보기에서 편집을 선택합니다.

   > **Note**
   > 이진 파일을 선택하는 경우, 콘텐츠를 표시할 것인지 여부를 확인하라는 경고 메시지가 나타납니다. 이진 파일 편집에 CodeCommit 콘솔을 사용하면 안 됩니다.

5. 파일을 편집하고 이러한 변경을 수행한 사용자와 변경 이유에 대한 정보를 다른 사용자에게 제공합니다.
   - 작성자 이름에 이름을 입력합니다. 이 이름은 커밋 정보에서 작성자 이름과 커미터 이름으로 모두 사용됩니다. CodeCommit은 IAM 사용자 이름이나 콘솔 로그인의 파생물을 작성자 이름으로 사용하도록 기본 설정되어 있습니다.
   - 이메일 주소에는 다른 리포지토리 사용자가 이러한 변경에 대해 문의할 수 있도록 이메일 주소를 입력합니다.
   - 커밋 메시지에 변경에 대한 간단한 설명을 입력합니다.
6. 변경 사항 커밋을 선택하여 파일의 변경 사항을 저장하고 리포지토리에 커밋합니다.

파일 편집 또는 삭제 (AWS CLI)

AWS CLI 및 put-file 명령을 사용하여 CodeCommit 리포지토리의 파일을 변경할 수 있습니다. 또한 변경된 파일을 원래와 다른 위치에 저장하고 싶은 경우에는 put-file 명령을 사용하여 변경된 파일에 대한 디렉터리나 경로 구조를 추가할 수 있습니다. 파일을 완전히 삭제하려면 delete-file 명령을 사용하면 됩니다.

> **Note**
>
> CodeCommit에서 AWS CLI 명령을 사용하려면를 설치합니다 AWS CLI. 자세한 내용은 명령줄 참조 단원을 참조하십시오.

리포지토리에서 파일을 편집하려면

1. 파일의 로컬 복사본을 사용하여 CodeCommit 리포지토리에 추가할 내용을 변경합니다.
2. 터미널 또는 명령줄에서 다음을 지정하여 put-file 명령을 실행합니다.

   - 편집된 파일을 추가할 리포지토리.
   - 편집된 파일을 추가할 브랜치.
   - 해당 브랜치에서 수행된 가장 최근 커밋의 축약되지 않은 커밋 ID(팁 또는 헤드 커밋)
   - 파일의 로컬 위치.
   - 업데이트된 파일이 리포지토리에 저장된 경로(있는 경우)를 포함하여 추가할 업데이트 파일의 이름.
   - 이 파일 변경에 연결할 사용자 이름과 이메일.
   - 수행한 변경을 설명하는 커밋 메시지.

   사용자 이름, 이메일 주소 및 커밋 메시지는 선택 사항이지만, 다른 사용자에게 변경을 수행한 사용자와 변경 이유를 알려줍니다. 사용자 이름을 제공하지 않는 경우, CodeCommit은 IAM 사용자 이름이나 콘솔 로그인 파생물을 사용하도록 기본 설정됩니다.

   *## ##, MyDemorepo## ##### ## ## ## ID# 4c925148EXAMPLE*이고 이름이 *feature-randomizationfeature*인 브랜치에 *ExampleSolution.py*라는 파일을 편집한 내용을 추가하려면 다음과 같이 합니다.

```
aws codecommit put-file --repository-name MyDemoRepo --branch-name feature-
randomizationfeature --file-content file://MyDirectory/ExampleSolution.py --file-
```

```
path /solutions/ExampleSolution.py --parent-commit-id 4c925148EXAMPLE --name "María
 García" --email "maría_garcía@example.com" --commit-message "I fixed the bug Mary
 found."
```

> **Note**
>
> 변경된 이진 파일을 추가하려면 **fileb**://*MyDirectory*/*MyFile*.raw 표기법을 따르는 --file-content를 사용해야 합니다.

이 명령이 제대로 실행되면 다음과 비슷한 출력이 반환됩니다.

```
{
   "blobId": "2eb4af3bEXAMPLE",
   "commitId": "317f8570EXAMPLE",
   "treeId": "347a3408EXAMPLE"
}
```

파일을 삭제하려면 delete-file 명령을 사용합니다. 예를 들어, *MyDemoRepo*라는 리포지토리에서 최근 커밋 ID가 *c5709475EXAMPLE*이고 이름이 main인 브랜치에서 *README.md*라는 파일을 삭제하려면 다음과 같이 합니다.

```
aws codecommit delete-file --repository-name MyDemoRepo --branch-name main --file-
path README.md --parent-commit-id c5709475EXAMPLE
```

이 명령이 제대로 실행되면 다음과 비슷한 출력이 반환됩니다.

```
{
  "blobId":"559b44fEXAMPLE",
  "commitId":"353cf655EXAMPLE",
  "filePath":"README.md",
  "treeId":"6bc824cEXAMPLE"
}
```

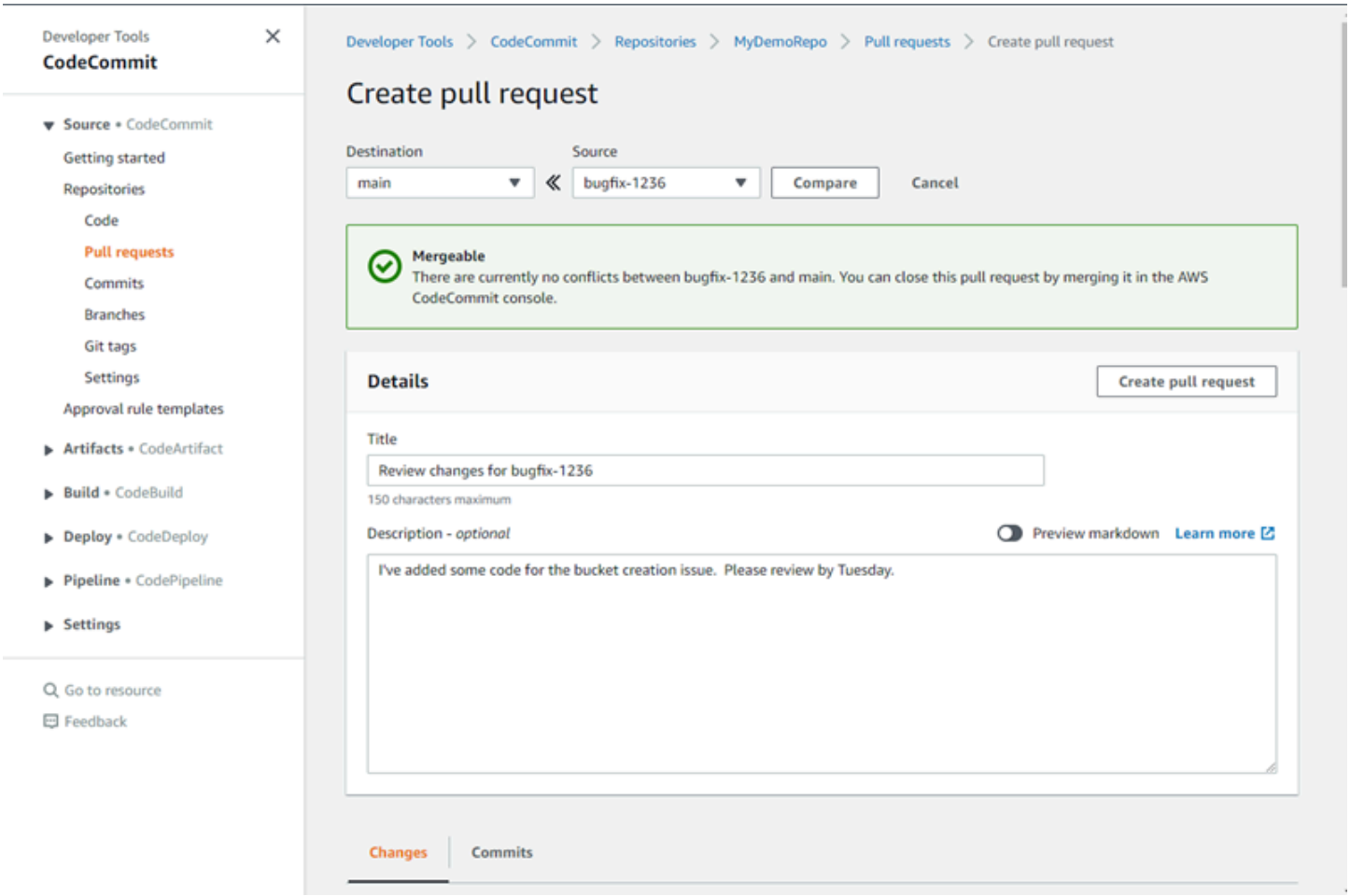
파일 편집 (Git)

로컬 리포지토리에서 파일을 편집해서 CodeCommit 리포지토리에 변경 내용을 푸시할 수 있습니다. 자세한 내용은 Git 및 시작하기 AWS CodeCommit 단원을 참조하십시오.

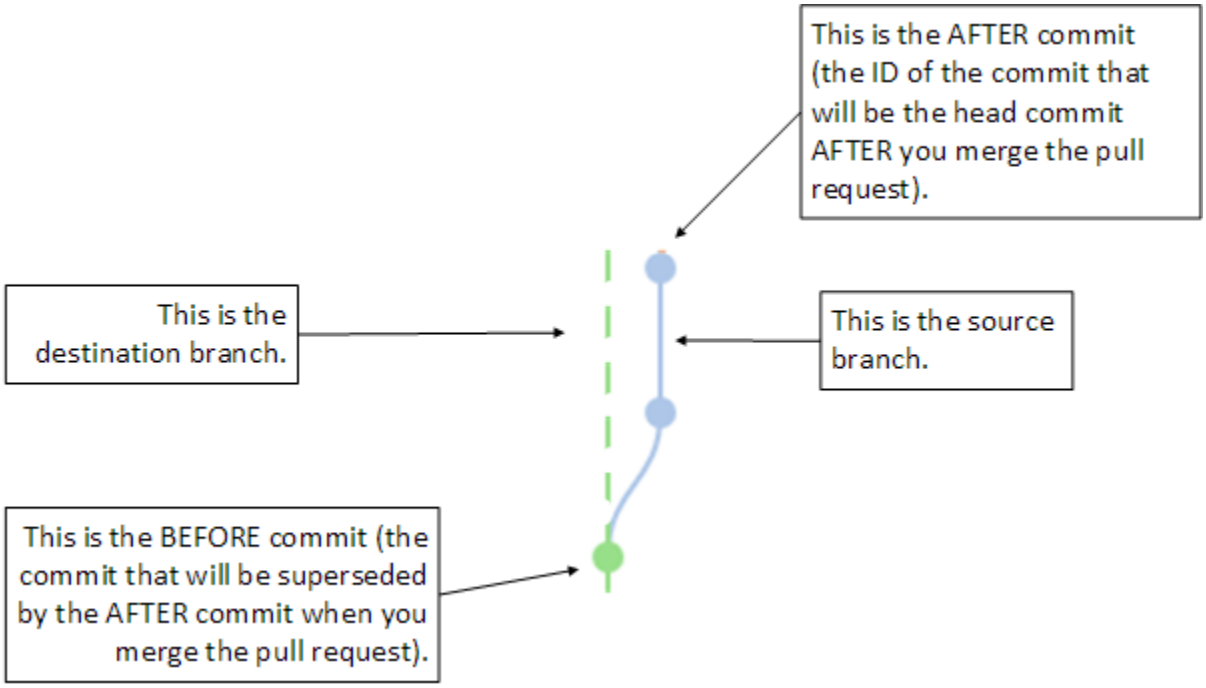
AWS CodeCommit 리포지토리에서 풀 요청 작업

풀 요청은 나와 리포지토리의 다른 사용자가 검토하고, 주석을 추가하며, 한 브랜치에서 다른 브랜치로 코드를 변경하기 위한 기본적인 방법입니다. 풀 요청을 사용하여 중요하지 않은 변경 내용이나 수정 사항, 중요 기능 추가 또는 릴리스된 소프트웨어의 새 버전에 대한 코드 변경 내용을 공동으로 검토할 수 있습니다. 다음은 풀 요청에 대한 한 가지 워크플로우입니다.

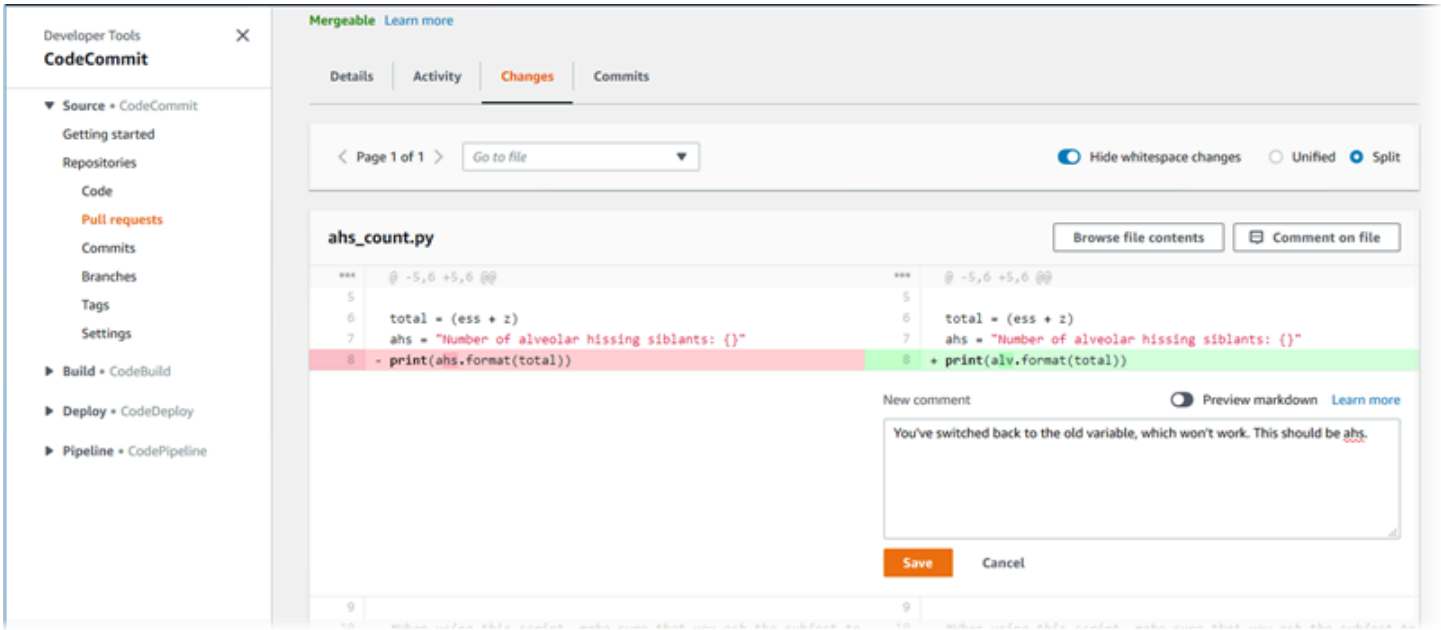
MyDemoRepo라는 리포지토리에서 작업 중인 개발자 Li Juan은 제품의 향후 버전에 추가될 새 기능 작업을 수행하려고 합니다. 그는 프로덕션 지원 코드와 별도로 유지하기 위해 기본 브랜치와 분리된 이 작업용 브랜치를 생성하여 `feature-randomizationfeature`라는 이름을 지정합니다. 그녀는 코드를 작성하여 설명을 추가하고 새 기능 코드를 이 브랜치에 푸시합니다. 변경 내용을 기본 브랜치에 병합하기 전에 다른 리포지토리 사용자가 품질 코드를 검토하게 하려 합니다. 이렇게 하기 위해 그녀는 풀 요청을 생성합니다. 풀 요청에는 작업용 브랜치와 자신이 변경 내용을 병합할 코드의 브랜치(이 경우 기본 브랜치) 간의 비교가 포함됩니다. 또한 지정된 수의 사용자가 풀 요청을 승인해야 하는 승인 규칙을 생성할 수 있습니다. 승인 사용자 풀을 지정할 수도 있습니다. 다른 사용자가 그녀의 코드와 변경 내용을 검토하고 주석과 제안을 추가합니다. 그녀는 다른 사용자의 주석을 반영하여 작업용 브랜치를 코드 변경 내용으로 여러 번 업데이트할 수 있습니다. 변경 내용은 CodeCommit에서 해당 브랜치에 푸시할 때마다 풀 요청에 통합됩니다. 또한 풀 요청이 미해결 상태인 동안 의도한 대상 브랜치에서 변경한 내용을 통합할 수 있으므로 사용자는 제안한 모든 변경 내용을 상황에 맞게 검토할 수 있습니다. 그녀와 검토자가 만족하고 승인 규칙 조건(있는 경우)이 충족되면 그녀 또는 검토자 중 한 명이 코드를 병합하고 풀 요청을 종료합니다.



풀 요청에는 검토를 요청할 코드를 포함하는 소스 브랜치와 검토한 코드를 변경할 대상 브랜치, 이렇게 두 개 브랜치가 필요합니다. 소스 브랜치에는 AFTER 커밋이 포함되며 이 커밋은 대상 브랜치에 병합하려는 변경 내용을 포함합니다. 대상 브랜치에는 BEFORE 커밋이 포함되며, 이 커밋은 풀 요청 브랜치가 대상 브랜치에 병합되기 전 코드의 상태를 나타냅니다. 병합 전략의 선택은 CodeCommit 콘솔에서 소스 브랜치와 대상 브랜치 간에 커밋이 병합되는 방법의 세부 사항에 영향을 줍니다. CodeCommit의 병합 전략에 대한 자세한 내용은 풀 요청 병합 (콘솔) 섹션을 참조하세요.



풀 요청은 소스 브랜치의 말단과 풀 요청이 생성될 당시의 대상 브랜치의 마지막 커밋 간의 차이를 표시하므로, 사용자는 변경 내용을 보고 그에 대한 설명을 추가할 수 있습니다. 주석에 따라 변경 내용을 소스 브랜치로 커밋하고 푸시함으로써 풀 요청을 업데이트할 수 있습니다.



코드가 검토되고 승인 규칙 요구 사항(있는 경우)이 충족되면 다음과 같은 여러 방법 중 하나로 풀 요청을 종료할 수 있습니다.

- 브랜치를 로컬에서 병합하여 변경 내용을 푸시합니다. 이렇게 하면 fast-forward merge 병합 전략을 사용하여 병합 충돌이 없을 경우 요청이 자동으로 닫힙니다.
- AWS CodeCommit 콘솔을 사용하여 병합하지 않고 풀 요청을 종료하거나, 병합에서 충돌을 해결하거나, 충돌이 없는 경우 사용 가능한 병합 전략 중 하나를 사용하여 브랜치를 닫고 병합합니다.
- AWS CLI를 사용합니다.

풀 요청을 생성하려면 먼저 다음을 수행해야 합니다.

- 검토하려는 코드 변경 내용을 반드시 특정 브랜치(소스 브랜치)로 커밋해 놓고 푸시해 놓아야 합니다.
- 다른 사용자가 풀 요청 및 변경 내용에 대해 알 수 있도록 리포지토리에 대한 알림을 설정합니다. (이 단계는 선택 사항이며, 권장 사항은 아닙니다.)
- 승인 규칙 템플릿을 생성하여 리포지토리와 연결합니다. 그러면 코드 품질을 보장하기 위해 풀 요청에 대한 승인 규칙이 자동으로 생성됩니다. 자세한 내용은 승인 규칙 템플릿 작업 단원을 참조하십시오.

Amazon Web Services 계정의 리포지토리 사용자에 대한 IAM 사용자를 설정하면 풀 요청을 더욱 효율적으로 할 수 있습니다. 어떤 사용자가 어떤 의견을 작성했는지 식별하기 더 쉽습니다. 다른 장점은 IAM 사용자가 리포지토리 액세스에 Git 보안 인증 정보를 사용할 수 있다는 것입니다. 자세한 내용은 1 단계: CodeCommit에 대한 초기 구성 단원을 참조하십시오. 페더레이션 액세스 사용자를 비롯한 다른 종류의 사용자와 함께 풀 요청을 사용할 수 있습니다.

CodeCommit에서 리포지토리의 다른 측면으로 작업하는 방법에 대해 자세히 알아보려면 리포지토리 작업, 승인 규칙 템플릿 작업, 파일 작업하기, 커밋 작업, 브랜치 작업,사용자 기본 설정으로 작업 섹션을 참조하세요.

주제

- 풀 요청 생성
- 풀 요청에 대한 승인 규칙 생성
- AWS CodeCommit 리포지토리의 풀 요청 보기
- 풀 요청 검토
- 풀 요청 업데이트
- 풀 요청에 대한 승인 규칙 편집 또는 삭제
- 풀 요청에 대한 승인 규칙 재정의

- AWS CodeCommit 리포지토리에서 풀 요청 병합
- AWS CodeCommit 리포지토리에서 풀 요청의 충돌 해결
- AWS CodeCommit 리포지토리에서 풀 요청 닫기

풀 요청 생성

풀 요청을 생성하면 변경 내용을 다른 브랜치에 병합하기 전에 다른 사용자가 코드 변경을 보고 검토할 수 있습니다. 먼저 코드 변경을 위한 브랜치를 생성합니다. 이 브랜치를 풀 요청의 소스 브랜치라고도 합니다. 이러한 변경 내용을 리포지토리로 커밋하고 푸시한 후, 해당 브랜치(소스 브랜치)의 내용을 풀 요청이 닫히고 나서 변경 내용을 병합할 브랜치(대상 브랜치)와 비교하는 풀 요청을 생성할 수 있습니다.

AWS CodeCommit 콘솔 또는를 사용하여 리포지토리 AWS CLI 에 대한 풀 요청을 생성할 수 있습니다.

주제

- 풀 요청 생성 (콘솔)
- 풀 요청 생성 (AWS CLI)

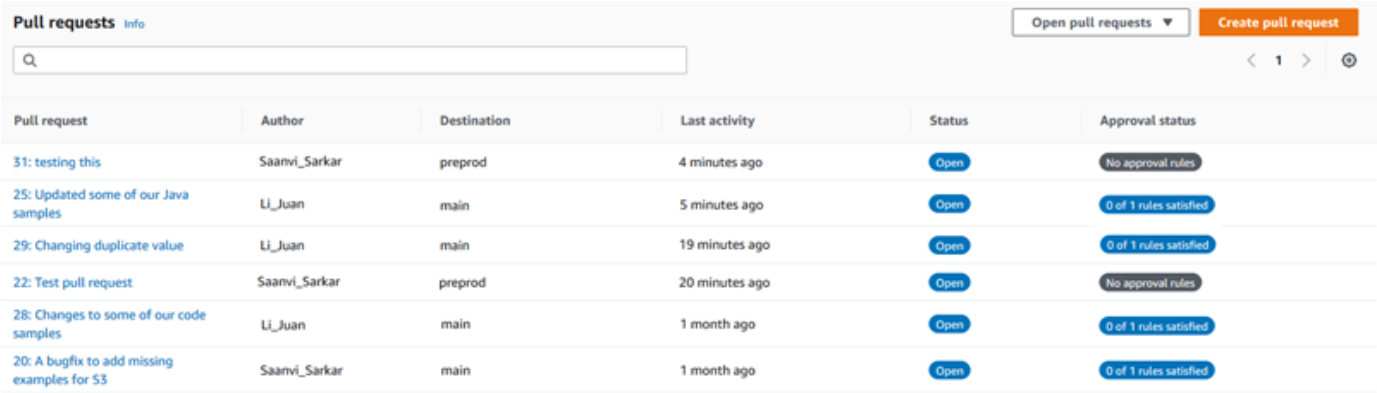
풀 요청 생성 (콘솔)

CodeCommit 콘솔을 사용하여 CodeCommit 리포지토리에 풀 요청을 생성할 수 있습니다. 리포지토리가 알림으로 구성된 경우, 구독 사용자는 풀 요청 생성 시 이메일을 받게 됩니다.

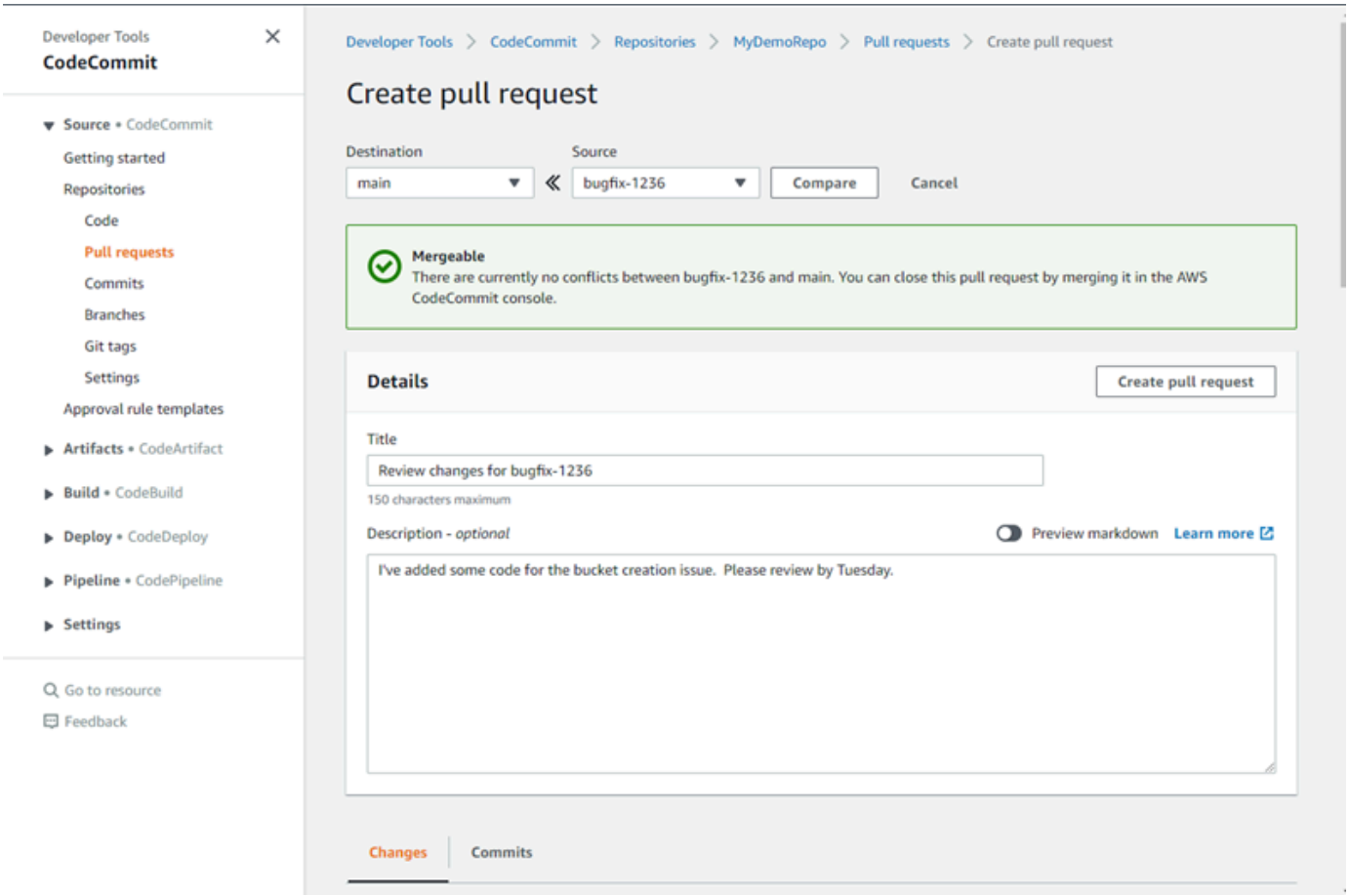
1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서, 풀 요청을 생성하려는 리포지토리의 이름을 선택합니다.
3. 탐색 창에서 풀 요청을 선택합니다.

   > **Tip**
   > 또한 브랜치 및 코드에서 풀 요청을 생성할 수도 있습니다.

4. 풀 요청 생성을 선택합니다.



5. 풀 요청 생성의 소스에서 검토하려는 변경 내용이 포함된 브랜치를 선택합니다.
6. 대상에서, 풀 요청이 닫힐 때 코드 변경 내용을 병합할 브랜치를 선택합니다.
7. 비교를 선택합니다. 두 브랜치에 대한 비교가 실행되고 그 차이가 표시됩니다. 풀 요청이 닫힐 때 두 브랜치를 자동으로 병합할지 여부를 결정하기 위한 분석도 수행됩니다.
8. 비교 세부 정보와 변경 내용을 검토하여, 검토를 요청할 변경 내용 및 커밋이 풀 요청에 포함되었는지 확인합니다. 포함되지 않은 경우 소스 및 대상 브랜치 선택을 조정하고 비교를 다시 선택합니다.
9. 풀 요청에 대한 비교 결과에 만족하면 제목에 이 검토 항목을 잘 설명하는 간단한 제목을 입력합니다. 이 제목은 리포지토리의 풀 요청 목록에 표시되는 제목입니다.
10. (선택 사항) 설명에 이 검토 항목에 대한 세부 정보와 검토자에 대한 기타 유용한 정보를 입력합니다.
11. 생성을 선택합니다.



풀 요청이 리포지토리의 풀 요청 목록에 나타납니다. 알림을 구성한 경우, Amazon SNS 주제 구독자는 새로 생성된 풀 요청에 대해 알려 주는 이메일을 받습니다.

풀 요청 생성 (AWS CLI)

CodeCommit에서 AWS CLI 명령을 사용하려면를 설치합니다 AWS CLI. 자세한 내용은 명령줄 참조 단원을 참조하십시오.

AWS CLI 를 사용하여 CodeCommit 리포지토리에서 풀 요청을 생성하려면

1. 다음을 지정하여 create-pull-request 명령을 실행합니다.
   - 풀 요청의 이름(--title 옵션 사용).
   - 풀 요청의 설명(--description 옵션 사용).
   - create-pull-request 명령의 대상 목록에는 다음이 포함됩니다.
     - 풀 요청을 생성할 CodeCommit 리포지토리의 이름(repositoryName 속성 사용).

- 소스 브랜치라고도 하는, 검토를 요청할 코드 변경 내용을 포함하는 브랜치의 이름(sourceReference 속성 사용).
- (선택 사항) 기본 브랜치로 병합하지 않으려는 경우, 대상 브랜치라고도 하는 코드 변경 내용을 병합할 브랜치의 이름(destinationReference 속성 사용).

- 클라이언트에서 생성된 고유한 멱등성 토큰(--client-request-token 옵션 사용).

이 예제에서는 *jane-branch*라는 소스 브랜치를 대상으로 *Please review these changes by Tuesday*라는 설명을 곁들여, *Pronunciation difficulty analyzer*라는 풀 요청을 생성합니다. 풀 요청은 MyDemoRepo라는 CodeCommit 리포지토리의 기본 브랜치 main에 병합됩니다.

```
aws codecommit create-pull-request --title "Pronunciation difficulty analyzer"
 --description "Please review these changes by Tuesday" --client-request-token
 123Example --targets repositoryName=MyDemoRepo,sourceReference=jane-branch
```

2. 이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
{
    "pullRequest": {
        "approvalRules": [
            {
                "approvalRuleContent": "{\"Version\": \"2018-11-08\",
\"DestinationReferences\": [\"refs/heads/main\"],\"Statements\": [{\"Type
\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers\":
 [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
                "approvalRuleId": "dd8b17fe-EXAMPLE",
                "approvalRuleName": "2-approver-rule-for-main",
                "creationDate": 1571356106.936,
                "lastModifiedDate": 571356106.936,
                "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
                "originApprovalRuleTemplate": {
                    "approvalRuleTemplateId": "dd3d22fe-EXAMPLE",
                    "approvalRuleTemplateName": "2-approver-rule-for-main"
                },
                "ruleContentSha256": "4711b576EXAMPLE"
            }
        ],
        "authorArn": "arn:aws:iam::111111111111:user/Jane_Doe",
        "description": "Please review these changes by Tuesday",
        "title": "Pronunciation difficulty analyzer",
```

```
        "pullRequestTargets": [
            {
                "destinationCommit": "5d036259EXAMPLE",
                "destinationReference": "refs/heads/main",
                "repositoryName": "MyDemoRepo",
                "sourceCommit": "317f8570EXAMPLE",
                "sourceReference": "refs/heads/jane-branch",
                "mergeMetadata": {
                    "isMerged": false
                }
            }
        ],
        "lastActivityDate": 1508962823.285,
        "pullRequestId": "42",
        "clientRequestToken": "123Example",
        "pullRequestStatus": "OPEN",
        "creationDate": 1508962823.285
    }
}
```

풀 요청에 대한 승인 규칙 생성

풀 요청에 대한 승인 규칙을 생성하면 코드를 대상 브랜치로 병합하기 전에 사용자가 풀 요청을 승인하도록 하여 코드의 품질을 보장할 수 있습니다. 풀 요청을 승인해야 하는 사용자 수를 지정할 수 있습니다. 규칙에 대한 승인 사용자 풀을 지정할 수도 있습니다. 이렇게 하면 해당 사용자의 승인만 규칙에 필요한 승인 수에 합산됩니다.

> **Note**
>
> 또한 승인 규칙 템플릿을 생성하여, 리포지토리 간의 모든 풀 요청에 적용되는 승인 규칙 생성을 자동화할 수 있습니다. 자세한 내용은 승인 규칙 템플릿 작업 단원을 참조하십시오.

AWS CodeCommit 콘솔 또는를 사용하여 리포지토리 AWS CLI 에 대한 승인 규칙을 생성할 수 있습니다.

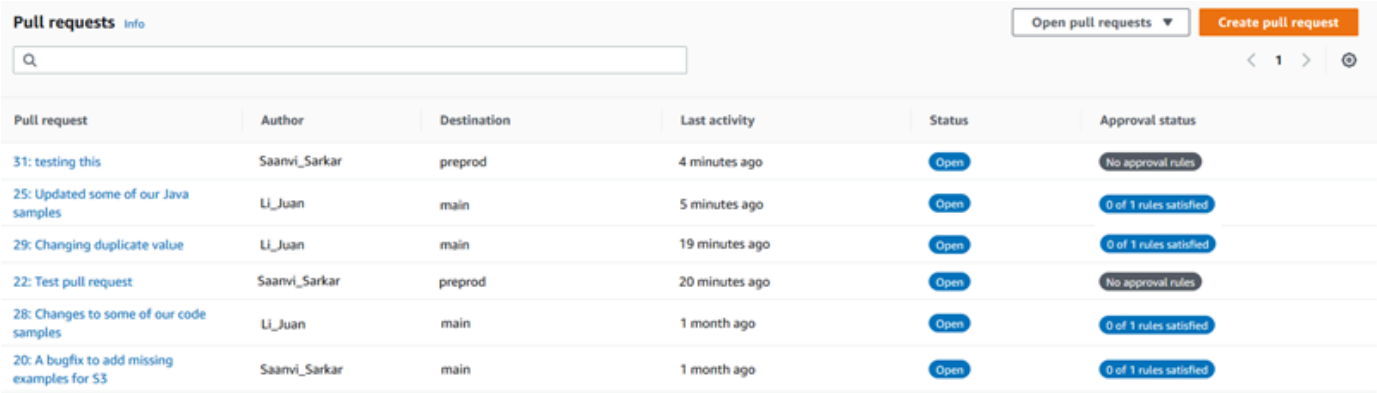
주제

- 풀 요청에 대한 승인 규칙 생성 (콘솔)
- 풀 요청에 대한 승인 규칙 생성 (AWS CLI)

풀 요청에 대한 승인 규칙 생성 (콘솔)

CodeCommit 콘솔을 사용하여 CodeCommit 리포지토리에 풀 요청에 대한 승인 규칙을 생성할 수 있습니다.

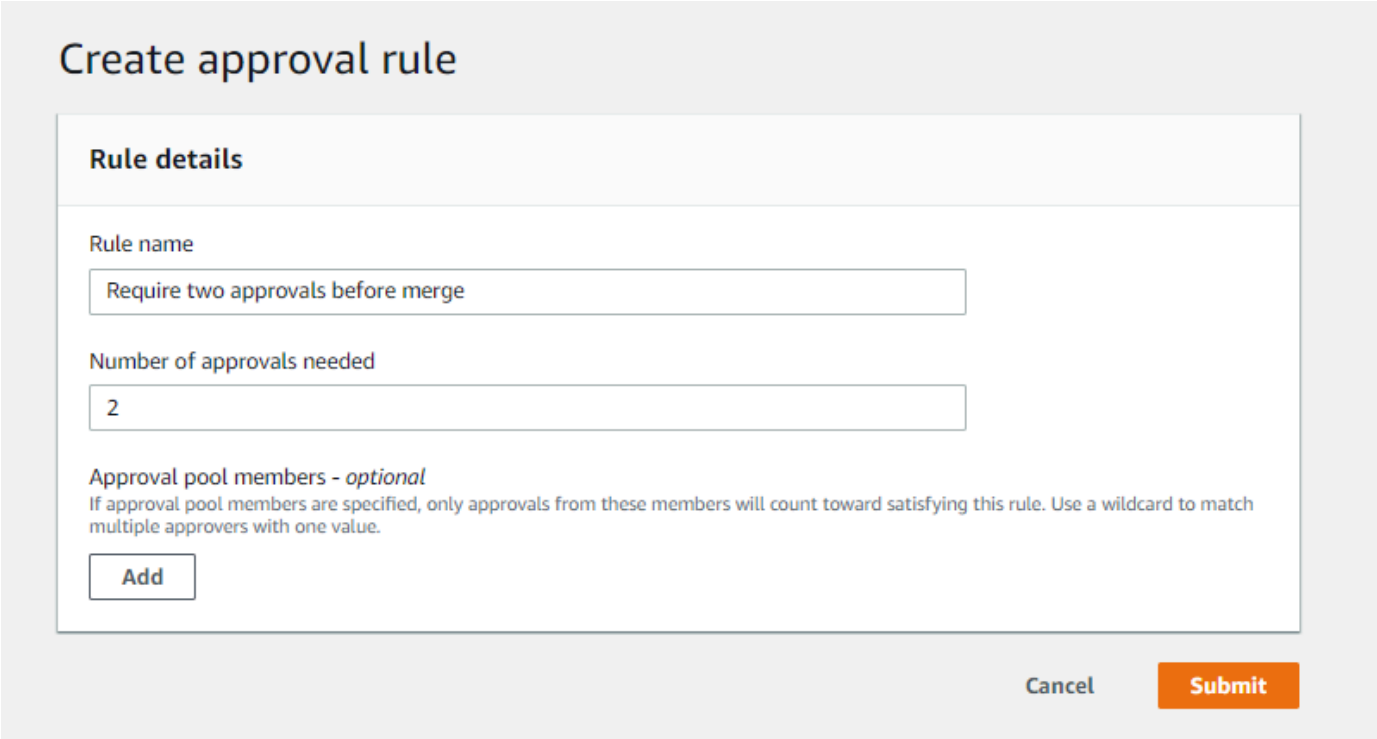
1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서 풀 요청에 대한 승인 규칙을 생성할 리포지토리의 이름을 선택합니다.
3. 탐색 창에서 풀 요청을 선택합니다.
4. 목록에서 승인 규칙을 생성할 풀 요청을 선택합니다. 미결 상태인 풀 요청에 대해서만 승인 규칙을 생성할 수 있습니다.



5. 풀 요청에서 승인을 선택한 다음 승인 규칙 생성을 선택합니다.
6. 규칙 이름에서 규칙에 해당 용도를 알 수 있도록 설명이 포함된 이름을 지정합니다. 예를 들어 풀 요청을 병합하기 전에 두 사람이 승인하도록 요구하려면 규칙 이름으로 **`Require two approvals before merge`**를 지정할 수 있습니다.

> **Note**
>
> 승인 규칙을 생성한 후에는 이름을 변경할 수 없습니다.

필요한 승인 수에 원하는 숫자를 입력합니다. 기본 값은 1입니다.



7. (선택 사항) 특정 사용자 그룹에서 풀 요청에 대한 승인을 받도록 하려면 승인 규칙 멤버에서 추가를 선택합니다. 승인자 유형에서 다음 중 하나를 선택합니다.

   - IAM 사용자 이름 또는 위임된 역할: 이 옵션은 로그인할 때 사용한 계정으로 AWS 계정 ID를 미리 채우며 오로지 이름만 필요로 합니다. 이 옵션은 제공된 이름과 일치하는 IAM 사용자 및 페더레이션 액세스 사용자 모두에 사용할 수 있습니다. 이 옵션은 폭넓은 유연성을 제공하는 매우 강력한 옵션입니다. 예를 들어, Amazon Web Services 계정 123456789012로 로그인하여 이 옵션을 선택하고 **`Mary_Major`**를 지정한 경우, 다음은 모두 해당 사용자의 승인으로 간주됩니다.

     - 해당 계정의 IAM 사용자 (`arn:aws:iam::123456789012:user/Mary_Major`)
     - IAM에서 Mary_Major로 식별되는 페더레이션 사용자 (`arn:aws:sts::123456789012:federated-user/Mary_Major`)

     이 옵션은 와일드카드가 포함되지 않는 한(`*Mary_Major`) 역할 세션 이름 Mary_Major(`arn:aws:sts::123456789012:assumed-role/CodeCommitReview/Mary_Major`)를 사용하여 **`CodeCommitReview`** 역할을 수임한 누군가의 활성 세션을 인식하지 못합니다. 역할 이름(`CodeCommitReview/Mary_Major`)을 명시적으로 지정할 수도 있습니다.

   - 정규화된 ARN: 이 옵션을 사용하면 IAM 사용자 또는 역할의 정규화된 Amazon 리소스 이름(ARN)을 지정할 수 있습니다. 또한이 옵션은 AWS Lambda 및와 같은 다른 AWS 서비스에서 사용하는 수임된 역할을 지원합니다 AWS CodeBuild. 수임된 역할의 경우 ARN 형식

은 역할 및 기능에 대해 각각 `arn:aws:sts::AccountID:assumed-role/RoleName` 및 `arn:aws:sts::AccountID:assumed-role/FunctionName` 형식이어야 합니다.

승인자 유형으로 IAM 사용자 이름 또는 수임된 역할을 선택한 경우, 값에는 IAM 사용자나 역할의 이름 또는 사용자나 역할의 정규화된 ARN을 입력합니다. 필요한 승인 수에 해당 승인이 합산되는 모든 사용자 또는 역할을 추가할 때까지 추가를 다시 선택하여 사용자 또는 역할을 추가합니다.

두 승인자 유형 모두 값에 와일드카드(*)를 사용할 수 있습니다. 예를 들어, IAM 사용자 이름 또는 수임된 역할 옵션을 선택하고 **`CodeCommitReview/*`**를 지정하면, **`CodeCommitReview`**의 역할을 맡은 모든 사용자가 승인 풀에 포함됩니다. 개별 역할 세션 이름은 필요한 승인자 수에 포함됩니다. 이러한 방식으로 Mary_Major 및 Li_Juan은 로그인하여 `CodeCommitReview` 역할을 수임하면 승인으로 합산됩니다. IAM ARN, 와일드카드, 형식 등에 대한 자세한 내용은 IAM 식별자를 참조하세요.

> **Note**
>
> 승인 규칙은 교차 계정 승인을 지원하지 않습니다.

8. 승인 규칙 구성을 완료했으면 제출을 선택합니다.

풀 요청에 대한 승인 규칙 생성 (AWS CLI)

CodeCommit에서 AWS CLI 명령을 사용하려면를 설치합니다 AWS CLI. 자세한 내용은 명령줄 참조 단원을 참조하십시오.

CodeCommit 리포지토리에 풀 요청에 대한 승인 규칙을 생성하려면

1. 다음을 지정하여 create-pull-request-approval-rule 명령을 실행합니다.

   - 풀 요청의 ID(--id 옵션 사용).
   - 승인 규칙의 이름(--approval-rule-name 옵션 사용).
   - 승인 규칙의 내용(--approval-rule-content 옵션 사용).

   승인 규칙을 생성할 때 다음 두 가지 방법 중 하나로 승인 풀에 승인자를 지정할 수 있습니다.

   - CodeCommitApprovers: 이 옵션에는 Amazon Web Services 계정과 리소스만 필요합니다. 이 옵션은 이름이 제공된 리소스 이름과 일치하는 IAM 사용자 및 페더레이션 액세스 사용자 모두

에 사용할 수 있습니다. 이 옵션은 폭넓은 유연성을 제공하는 매우 강력한 옵션입니다. 예를 들어, Amazon Web Services 계정 123456789012 및 **Mary_Major**를 지정한 경우, 다음은 모두 해당 사용자의 승인으로 간주됩니다.

- 해당 계정의 IAM 사용자 (`arn:aws:iam::123456789012:user/Mary_Major`)
- IAM에서 Mary_Major로 식별되는 페더레이션 사용자 (`arn:aws:sts::123456789012:federated-user/Mary_Major`)

이 옵션은 와일드카드가 포함되지 않는 한(`*Mary_Major`) 역할 세션 이름 Mary_Major(`arn:aws:sts::123456789012:assumed-role/CodeCommitReview/Mary_Major`)를 사용하여 **CodeCommitReview** 역할을 수임한 누군가의 활성 세션을 인식하지 못합니다.

- 정규화된 ARN: 이 옵션을 사용하면 IAM 사용자 또는 역할의 정규화된 Amazon 리소스 이름(ARN)을 지정할 수 있습니다.

IAM ARN, 와일드카드, 형식 등에 대한 자세한 내용은 IAM 식별자를 참조하세요.

다음 예제에서는 ID가 27인 풀 요청에 대해 `Require two approved approvers`라는 승인 규칙을 생성합니다. 이 규칙은 하나의 승인 풀에서 두 개의 승인이 필요하도록 지정합니다. 이 풀에는 123456789012 Amazon Web Services 계정에서 **CodeCommitReview**의 역할을 수임하고 CodeCommit에 액세스할 수 있는 모든 사용자가 포함됩니다. 또한 동일한 Amazon Web Services 계정의 `Nikhil_Jayashankar`라는 IAM 사용자 또는 페더레이션 사용자도 포함됩니다.

```
aws codecommit create-pull-request-approval-rule --pull-request-id 27
 --approval-rule-name "Require two approved approvers" --approval-
rule-content "{\"Version\": \"2018-11-08\",\"Statements\": [{\"Type\":
 \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers
\": [\"CodeCommitApprovers:123456789012:Nikhil_Jayashankar\",
 \"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}"
```

2. 이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
{
    "approvalRule": {
        "approvalRuleName": "Require two approved approvers",
        "lastModifiedDate": 1570752871.932,
        "ruleContentSha256": "7c44e6ebEXAMPLE",
        "creationDate": 1570752871.932,
        "approvalRuleId": "aac33506-EXAMPLE",
```

```
        "approvalRuleContent": "{\"Version\": \"2018-11-08\",\"Statements\":
 [{\"Type\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers
\": [\"CodeCommitApprovers:123456789012:Nikhil_Jayashankar\",
 \"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
        "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major"
    }
}
```

AWS CodeCommit 리포지토리의 풀 요청 보기

AWS CodeCommit 콘솔 또는를 사용하여 리포지토리 AWS CLI 에 대한 풀 요청을 볼 수 있습니다. 기본적으로는 진행 중인 풀 요청만 표시되지만, 필터를 변경하여 모든 풀 요청을 표시하거나, 종료된 요청 또는 생성한 풀 요청만 표시할 수 있습니다.

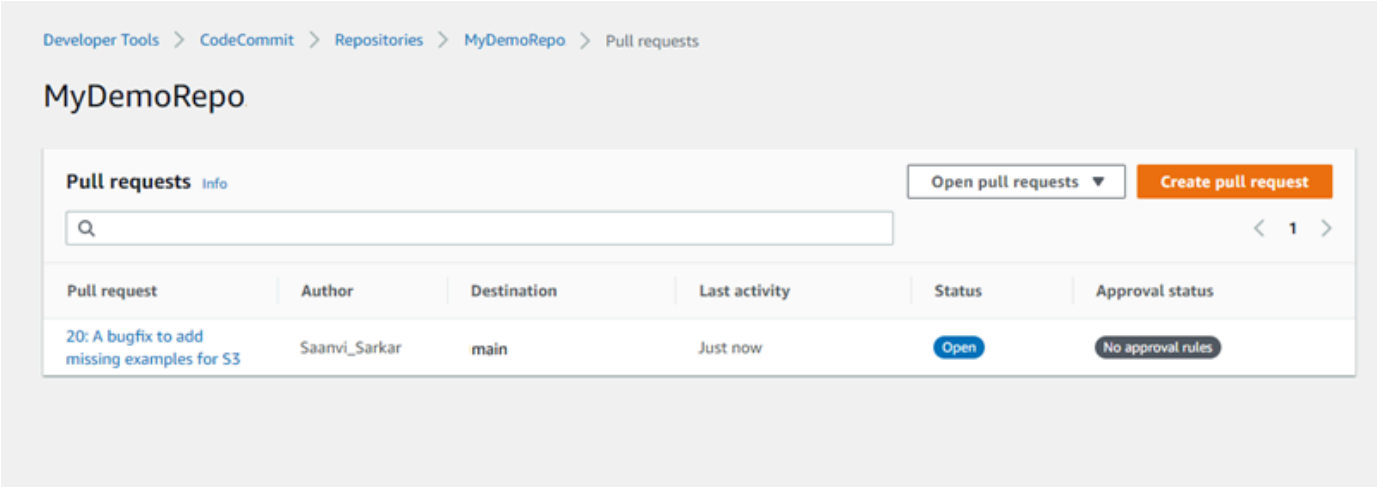
주제

- 풀 요청 보기 (콘솔)
- 풀 요청 보기 (AWS CLI)

풀 요청 보기 (콘솔)

AWS CodeCommit 콘솔을 사용하여 CodeCommit 리포지토리의 풀 요청 목록을 볼 수 있습니다. 필터를 변경하여 특정 풀 요청 세트만 표시되도록 목록 표시 화면을 변경할 수 있습니다. 예를 들면 생성한 풀 요청 중에서 상태가 미해결인 풀 요청의 목록을 표시하도록 선택하거나, 다른 필터를 선택하고 생성한 풀 요청 중에서 상태가 종결인 풀 요청만 볼 수 있습니다.

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서, 풀 요청을 보려 하는 리포지토리의 이름을 선택합니다.
3. 탐색 창에서 풀 요청을 선택합니다.
4. 기본적으로 모든 미해결 풀 요청 목록이 표시됩니다.



5. 표시 필터를 변경하려면 사용할 수 있는 필터 목록에서 선택합니다.
   - 미해결 풀 요청(기본값): 상태가 미해결인 모든 풀 요청을 표시합니다.
   - 모든 풀 요청: 모든 풀 요청을 표시합니다.
   - 종결된 풀 요청: 상태가 종결인 모든 풀 요청을 표시합니다.
   - 나의 풀 요청: 상태와 관계없이 내가 생성한 모든 풀 요청을 표시합니다. 자신의 주석을 작성했거나 참여한 검토는 표시되지 않습니다.
   - 나의 미해결 풀 요청: 내가 생성한 풀 요청 중 상태가 미해결인 풀 요청을 모두 표시합니다.
   - 나의 종결 풀 요청: 내가 생성한 풀 요청 중 상태가 종결인 풀 요청을 모두 표시합니다.
6. 표시되는 목록에서 살펴볼 풀 요청을 찾으면 해당 풀 요청을 선택합니다.

풀 요청 보기 (AWS CLI)

CodeCommit에서 AWS CLI 명령을 사용하려면를 설치합니다 AWS CLI. 자세한 내용은 명령줄 참조 단원을 참조하십시오.

다음 단계에 따라 AWS CLI 를 사용하여 CodeCommit 리포지토리에서 풀 요청을 봅니다.

1. 리포지토리에서 풀 요청 목록을 보려면 다음을 지정하여 list-pull-requests 명령을 실행합니다.
   - 풀 요청을 보려 하는 CodeCommit 리포지토리의 이름(--repository-name 옵션 사용).
   - (선택 사항) 풀 요청의 상태(--pull-request-status 옵션 사용).
   - (선택 사항) 풀 요청을 생성한 IAM 사용자의 Amazon 리소스 이름(ARN)(--author-arn 옵션 사용).
   - (선택 사항) 결과 배치를 반환하는 데 사용할 수 있는 열거형 토큰(--next-token 옵션 사용)

- (선택 사항) 요청당 반환되는 결과 수 제한(--max-results 옵션 사용).

예를 들어, MyDemoRepo라는 CodeCommit 리포지토리에서, ARN이 *arn:aws:iam: :111111111111:user/li_Juan*이고 상태가 *##*이며 IAM 사용자가 생성한 풀 요청을 나열하려면 다음과 같이 합니다.

```
aws codecommit list-pull-requests --author-arn arn:aws:iam::111111111111:user/
Li_Juan --pull-request-status CLOSED --repository-name MyDemoRepo
```

이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
{
   "nextToken": "",
   "pullRequestIds": ["2","12","16","22","23","35","30","39","47"]
}
```

풀 요청 ID는 최근 작업 순서대로 표시됩니다.

2. 풀 요청의 세부 정보를 보려면 풀 요청의 ID를 지정하여 --pull-request-id 옵션과 함께 get-pull-request 명령을 실행합니다. 예를 들어 ID가 *27*인 풀 요청에 대한 정보를 보려면 다음과 같이 합니다.

```
aws codecommit get-pull-request --pull-request-id 27
```

이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
{
    "pullRequest": {
        "approvalRules": [
            {
                "approvalRuleContent": "{\"Version\": \"2018-11-08\",\"Statements
\": [{\"Type\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers
\": [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
                "approvalRuleId": "dd8b17fe-EXAMPLE",
                "approvalRuleName": "2-approver-rule-for-main",
                "creationDate": 1571356106.936,
                "lastModifiedDate": 571356106.936,
                "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
                "ruleContentSha256": "4711b576EXAMPLE"
            }
```

```
        ],
        "lastActivityDate": 1562619583.565,
        "pullRequestTargets": [
            {
                "sourceCommit": "ca45e279EXAMPLE",
                "sourceReference": "refs/heads/bugfix-1234",
                "mergeBase": "a99f5ddbEXAMPLE",
                "destinationReference": "refs/heads/main",
                "mergeMetadata": {
                    "isMerged": false
                },
                "destinationCommit": "2abfc6beEXAMPLE",
                "repositoryName": "MyDemoRepo"
            }
        ],
        "revisionId": "e47def21EXAMPLE",
        "title": "Quick fix for bug 1234",
        "authorArn": "arn:aws:iam::123456789012:user/Nikhil_Jayashankar",
        "clientRequestToken": "d8d7612e-EXAMPLE",
        "creationDate": 1562619583.565,
        "pullRequestId": "27",
        "pullRequestStatus": "OPEN"
    }
}
```

3. 풀 요청에 대한 승인을 보려면 다음을 지정하여 get-pull-request-approval-state 명령을 실행합니다.

   - 풀 요청의 ID(--pull-request-id 옵션 사용).
   - 풀 요청의 개정 ID(--revision-id option) 사용). get-pull-request 명령을 사용하여 풀 요청의 현재 개정 ID를 얻을 수 있습니다.

   예를 들어 ID가 *8*이고 개정 ID가 *9f29d167EXAMPLE*인 풀 요청에 대한 승인을 보려면 다음과 같이 합니다.

```
aws codecommit get-pull-request-approval-state --pull-request-id 8 --revision-
id 9f29d167EXAMPLE
```

   이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
{
    "approvals": [
        {
            "userArn": "arn:aws:iam::123456789012:user/Mary_Major",
            "approvalState": "APPROVE"
        }
    ]
}
```

4. 풀 요청의 이벤트를 보려면 풀 요청의 ID를 지정하여 --pull-request-id 옵션과 함께 describe-pull-request-events 명령을 실행합니다. 예를 들어 ID가 *8*인 풀 요청에 대한 이벤트를 보려면 다음가 같이 합니다.

```
aws codecommit describe-pull-request-events --pull-request-id 8
```

이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
{
    "pullRequestEvents": [
        {
            "pullRequestId": "8",
            "pullRequestEventType": "PULL_REQUEST_CREATED",
            "eventDate": 1510341779.53,
            "actor": "arn:aws:iam::111111111111:user/Zhang_Wei"
        },
        {
            "pullRequestStatusChangedEventMetadata": {
                "pullRequestStatus": "CLOSED"
            },
            "pullRequestId": "8",
            "pullRequestEventType": "PULL_REQUEST_STATUS_CHANGED",
            "eventDate": 1510341930.72,
            "actor": "arn:aws:iam::111111111111:user/Jane_Doe"
        }
    ]
}
```

5. 풀 요청에 대한 병합 충돌이 있는지 보려면 다음을 지정하여 get-merge-conflicts 명령을 실행합니다.

   - CodeCommit 리포지토리의 이름(--repository-name 옵션 사용).

- 브랜치, 태그, HEAD 또는 병합 평가에 사용할 변경 내용 소스의 기타 정규화된 참조(--source-commit-specifier 옵션 사용).
- 브랜치, 태그, HEAD 또는 병합 평가에 사용할 변경 내용 대상의 기타 정규화된 참조(--destination-commit-specifier 옵션 사용).
- 사용할 병합 옵션(--merge-option 옵션 사용)

예를 들어, MyDemoRepo라는 리포지토리에서 *my-feature-branch*라는 소스 브랜츠의 끝과 main이라는 대상 브랜치 사이에 병합 충돌이 있는지 확인하려면 다음과 같이 합니다.

```
aws codecommit get-merge-conflicts --repository-name MyDemoRepo --source-commit-
specifier my-feature-branch --destination-commit-specifier main --merge-option
 FAST_FORWARD_MERGE
```

이 명령이 제대로 실행되면 다음과 비슷한 출력이 반환됩니다.

```
{
    "destinationCommitId": "fac04518EXAMPLE",
    "mergeable": false,
    "sourceCommitId": "16d097f03EXAMPLE"
}
```

풀 요청 검토

AWS CodeCommit 콘솔을 사용하여 풀 요청에 포함된 변경 사항을 검토할 수 있습니다. 요청, 파일, 개별 코드 행에 설명을 추가할 수 있습니다. 또한 다른 사용자가 작성한 주석에 댓글을 달 수도 있습니다. 리포지토리가 알림으로 구성된 경우 내 주석에 사용자가 댓글을 달거나, 풀 요청에 사용자가 주석을 추가할 때마다 이메일을 받게 됩니다.

AWS CLI 를 사용하여 풀 요청에 주석을 달고 주석에 회신할 수 있습니다. 변경 내용을 검토하려면 CodeCommit 콘솔, git diff 명령 또는 diff 도구를 사용해야 합니다.

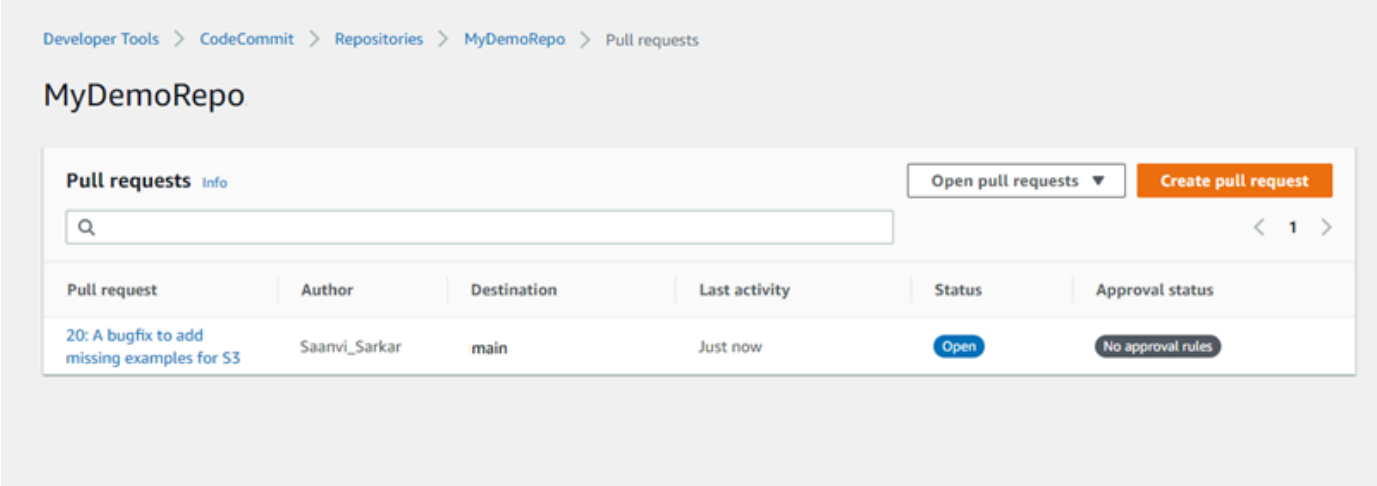
주제

- 풀 요청 검토 (콘솔)
- 풀 요청 검토 (AWS CLI)

풀 요청 검토 (콘솔)

CodeCommit 콘솔을 사용하여 CodeCommit 리포지토리의 풀 요청을 검토할 수 있습니다.

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서 리포지토리의 이름을 선택합니다.
3. 탐색 창에서 풀 요청을 선택합니다.
4. 기본적으로 모든 미해결 풀 요청 목록이 표시됩니다. 검토하려는 진행 중 풀 요청을 선택합니다.

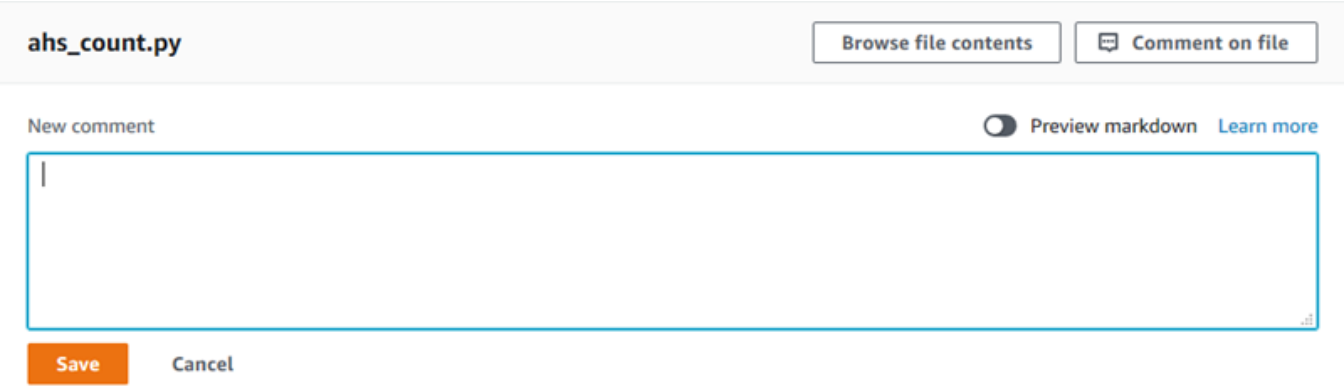


> **Note**
>
> 종료 또는 병합된 풀 요청에 대해 주석을 달 수 있지만 병합하거나 다시 열 수는 없습니다.

5. 풀 요청에서 변경 사항을 선택합니다.
6. 다음 중 하나를 수행합니다.
   - 전체 풀 요청에 대한 전반적인 주석을 추가하려면 변경 사항에 대한 주석의 새 주석에 주석을 입력한 다음 저장을 선택합니다. 마크다운을 사용하거나 설명을 일반 텍스트로 입력할 수 있습니다.

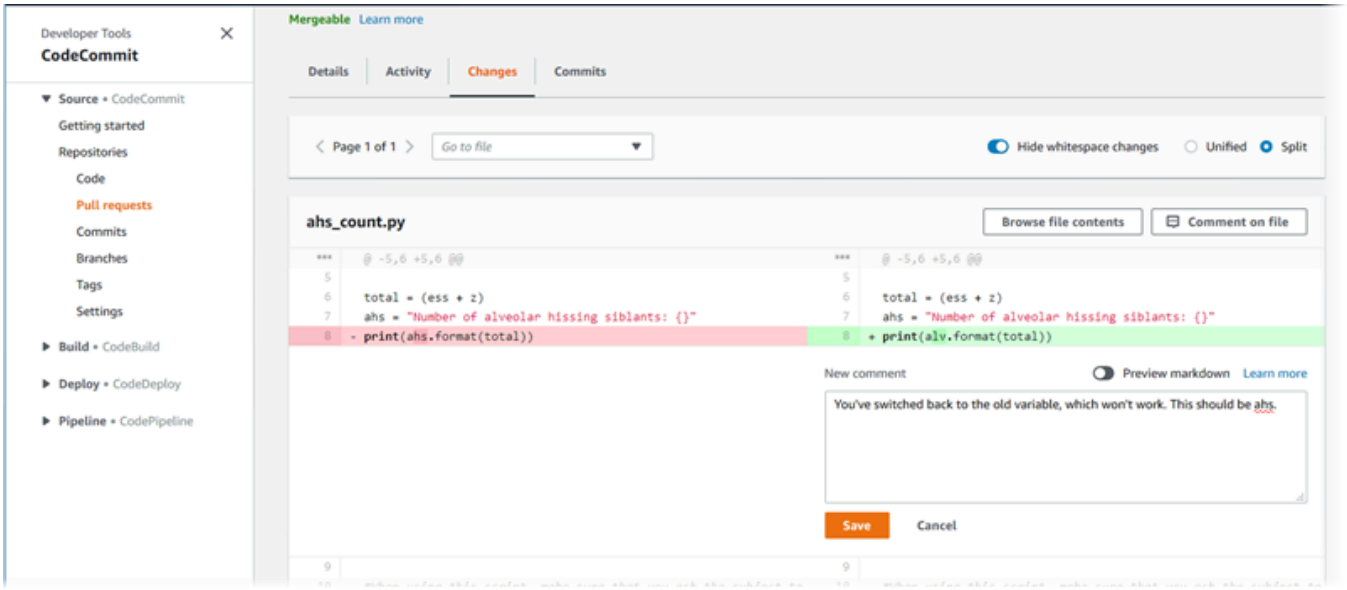
- 커밋의 파일에 주석을 추가하려면 변경 사항에서 해당 파일 이름을 찾습니다. 파일 이름 옆에 나타나는

  주석 아이콘을 선택하고 주석을 입력한 다음 저장을 선택합니다.

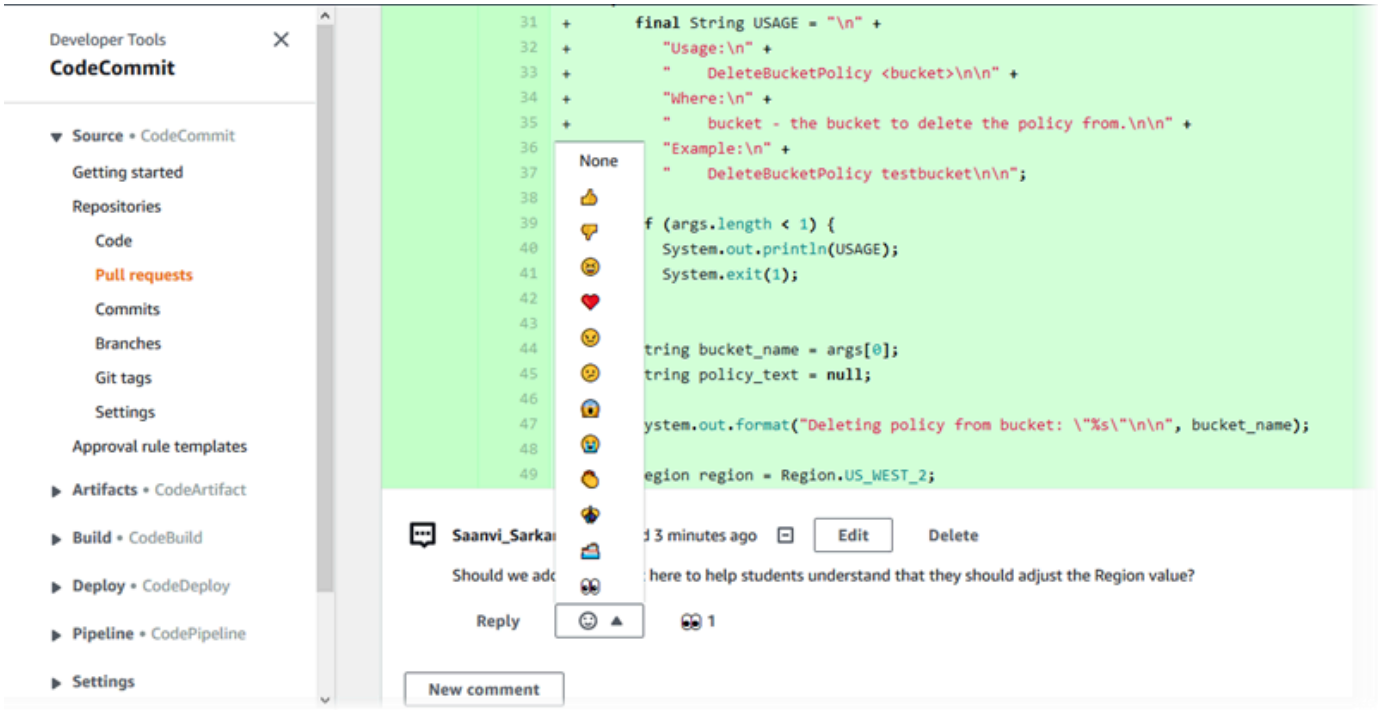


- 풀 요청의 변경된 행에 주석을 추가하려면 변경 사항에서 주석을 작성할 행으로 이동합니다. 해당 행에 나타나는

  주석 아이콘을 선택하고 주석을 입력한 다음 저장을 선택합니다.

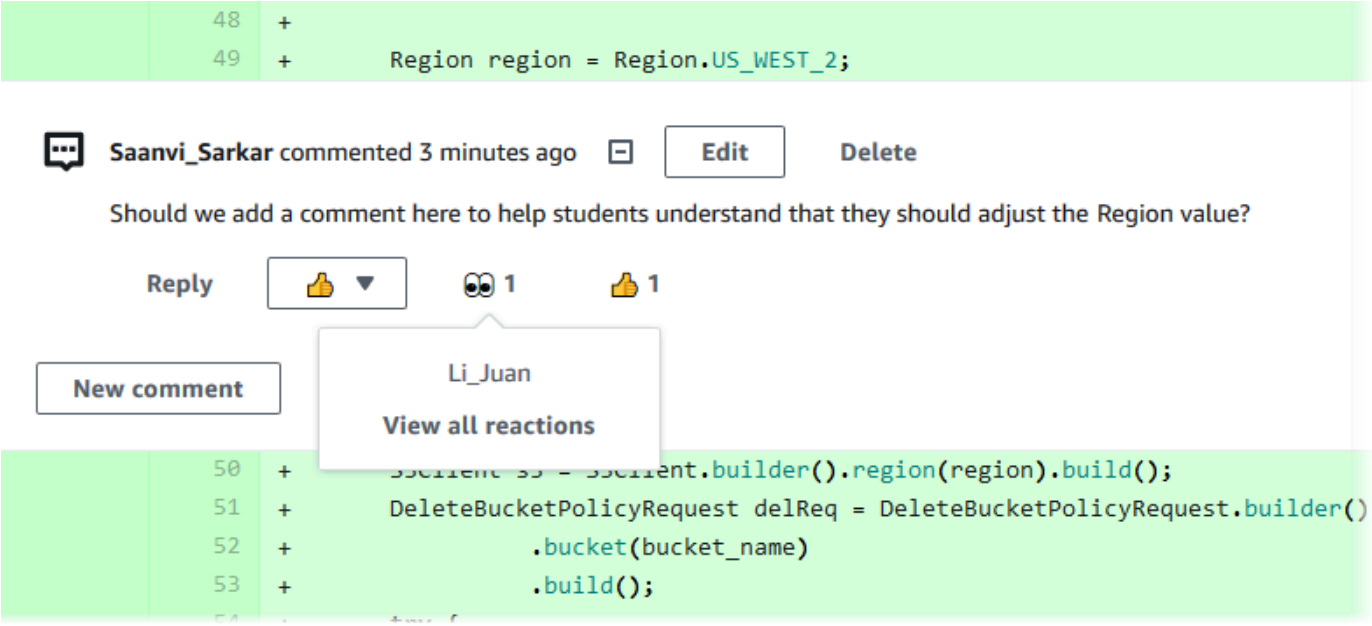


7. 커밋의 주석에 답변하려면 변경 사항 또는 활동에서 댓글을 선택합니다. 텍스트와 이모티콘으로 댓글을 달 수 있습니다.



특정 이모티콘 반응 댓글을 선택하여 응답한 사람들의 이름을 볼 수 있습니다. 모든 이모티콘 반응을 확인하고 누가 어떤 이모티콘으로 응답했는지에 대한 정보를 보려면 모든 반응 보기를 선택합니다. 주석에 이모티콘으로 응답한 경우 해당 응답이 이모티콘 반응 버튼 아이콘에 표시됩니다.

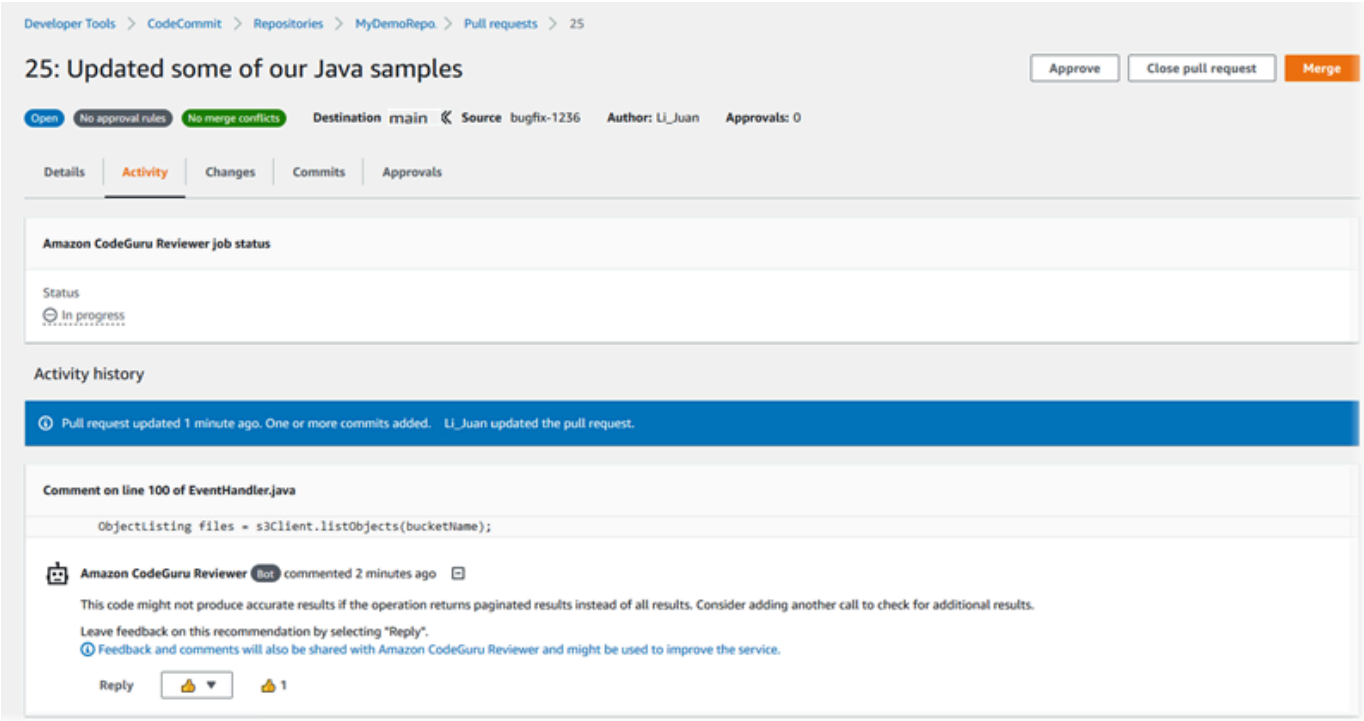
> **Note**
>
> 콘솔에 표시되는 반응 수는 페이지가 로드된 시점에 정확하게 합산됩니다. 이모티콘 반응 수에 대한 최신 정보를 보려면 페이지를 새로 고치거나 모든 반응 보기를 선택합니다.



8. (선택 사항) Amazon CodeGuru Reviewer에서 생성한 권장 사항에 회신하려면(권장 사항 품질에 대한 피드백 제공 포함) 댓글을 선택합니다. 응답 버튼을 사용하여 권장 사항의 승인 또는 거부 여부에 대한 일반 정보를 제공합니다. 의견 필드를 사용하여 응답에 대한 자세한 내용을 제공합니다.

> **Note**
>
> Amazon CodeGuru Reviewer는 프로그램 분석 및 기계 학습을 사용하여 Java 또는 Python 코드에서 일반적인 문제를 감지하고 권장 수정 사항을 제공하는 자동화된 코드 검토 서비스입니다.
>
> - 리포지토리가 Amazon CodeGuru Reviewer와 연결되고, 분석이 완료되고, 풀 요청의 코드가 Java 또는 Python 코드인 경우에만, Amazon CodeGuru Reviewer 주석이 표시됩니다. 자세한 내용은 AWS CodeCommit 리포지토리를 Amazon CodeGuru Reviewer와 연결 또는 연결 해제 단원을 참조하세요.
> - Amazon CodeGuru Reviewer 주석은 풀 요청의 최근 수정본에 대해 작성된 경우에만 변경 사항 탭에 표시됩니다. 이러한 주석은 항상 활동 탭에 표시됩니다.

- Amazon CodeGuru Reviewer 추천에는 사용 가능한 모든 이모티콘 반응으로 응답할 수 있지만, 추천의 유용성을 평가하는 데는 좋아요 및 싫어요 이모티콘 반응만 사용됩니다.

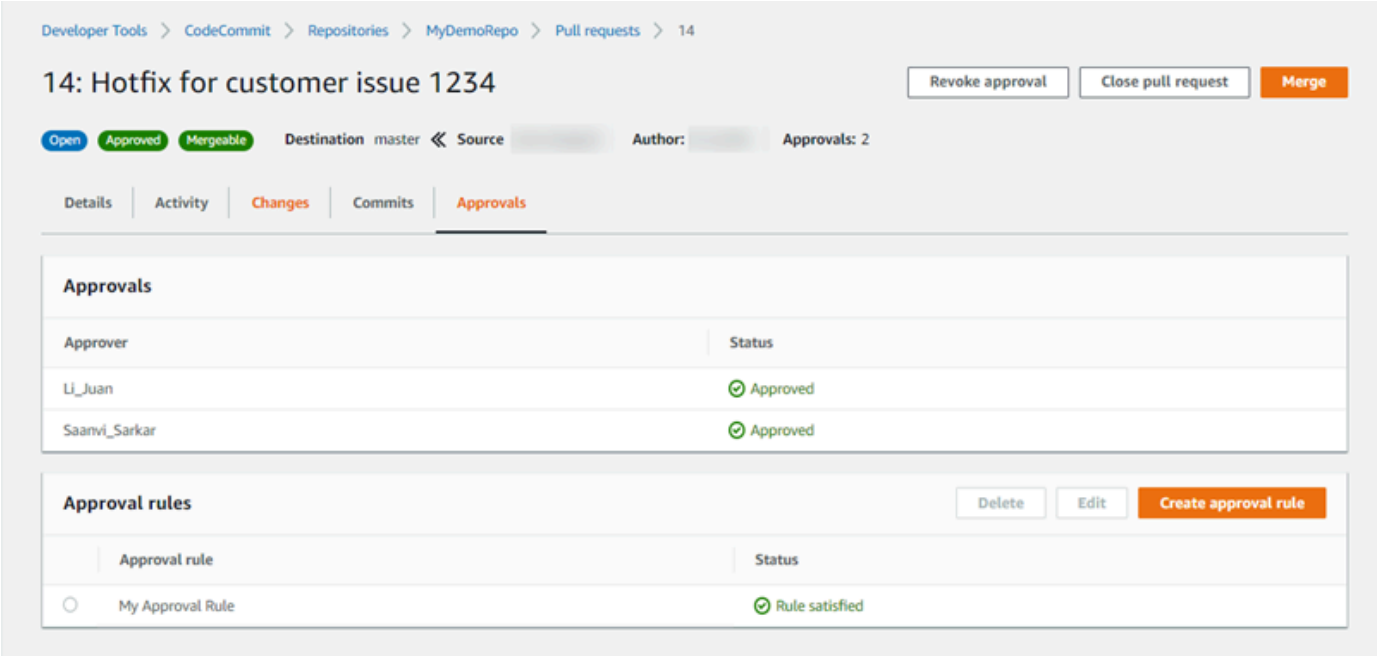


9. 풀 요청에서 변경된 내용을 승인하려면 승인을 선택합니다.

> **Note**
>
> 풀 요청을 생성한 사용자는 풀 요청을 승인할 수 없습니다.

승인에서 승인, 풀 요청에 대한 승인 규칙, 승인 규칙 템플릿에 의해 생성된 승인 규칙을 볼 수 있습니다. 풀 요청을 승인하지 않으려면 승인 취소를 선택할 수 있습니다.

> **Note**
>
> 미결 상태인 풀 요청만 승인하거나 승인을 취소할 수 있습니다. 상태가 병합됨 또는 종료됨인 풀 요청은 승인하거나 승인을 취소할 수 없습니다.



풀 요청 검토 (AWS CLI)

CodeCommit에서 AWS CLI 명령을 사용하려면를 설치합니다 AWS CLI. 자세한 내용은 명령줄 참조 단원을 참조하십시오.

다음 AWS CLI 명령을 사용하여 풀 요청을 검토할 수 있습니다.

- post-comment-for-pull-request - 풀 요청에 주석을 추가합니다.
- get-comments-for-pull-request - 풀 요청에 남긴 주석을 봅니다.
- update-pull-request-approval-state - 풀 요청을 승인하거나 취소합니다.
- post-comment-reply - 풀 요청의 주석에 댓글을 답니다.

다음 명령을 사용하면 풀 요청에 주석이 달린 이모티콘을 사용할 수도 있습니다.

- 주석에 이모티콘으로 반응하려면 put-comment-reaction을 실행합니다.
- 주석에 대한 이모티콘 반응을 보려면 get-comment-reactions을 실행합니다.

AWS CLI 를 사용하여 CodeCommit 리포지토리에서 풀 요청을 검토하려면

1. 리포지토리의 풀 요청에 주석을 추가하려면 다음을 지정하여 post-comment-for-pull-request 명령을 실행합니다.

   - 풀 요청의 ID(--pull-request-id 옵션 사용).
   - 풀 요청을 포함하는 리포지토리의 이름(--repository-name 옵션 사용).
   - 풀 요청이 병합되는 대상 브랜치의 커밋의 전체 커밋 ID(--before-commit-id 옵션 사용).
   - 주석을 게시할 때 풀 요청에 대한 현재 말단 브랜치인 소스 브랜치의 커밋의 전체 커밋 ID(--after-commit-id 옵션 사용).
   - 클라이언트에서 생성된 고유한 멱등성 토큰(--client-request-token 옵션 사용).
   - 설명의 내용(--content 옵션 사용).
   - 다음을 비롯하여 주석을 추가할 위치에 대한 위치 정보 목록.
     - 확장자와 하위 디렉터리를 비롯하여 비교하려는 파일의 이름(filePath 속성 사용).
     - 비교한 파일 내 변경의 행 번호(filePosition 속성 사용).
     - 변경에 대한 주석이 소스와 대상 브랜치 간의 비교 '전'인지 '후'인지 여부(relativeFileVersion 속성 사용).

   예를 들어, *"These don't appear to be used anywhere. Can we remove them?"*라는 주석을 쓴다고 가정하겠습니다. *MyDemoRepo*라는 리포지토리에서 *ahs_count.py* 파일의 변경 사항과 관련해, ID가 *47*인 풀 요청에 이 주석을 남기려면 다음과 같은 명령을 사용합니다.

```
aws codecommit post-comment-for-pull-request --pull-request-id "47" --
repository-name MyDemoRepo --before-commit-id 317f8570EXAMPLE --after-
commit-id 5d036259EXAMPLE --client-request-token 123Example --content
 "These don't appear to be used anywhere. Can we remove them?" --location
 filePath=ahs_count.py,filePosition=367,relativeFileVersion=AFTER
```

   이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
{
        "afterBlobId": "1f330709EXAMPLE",
        "afterCommitId": "5d036259EXAMPLE",
        "beforeBlobId": "80906a4cEXAMPLE",
        "beforeCommitId": "317f8570EXAMPLE",
        "comment": {
              "authorArn": "arn:aws:iam::111111111111:user/Saanvi_Sarkar",
              "clientRequestToken": "123Example",
              "commentId": "abcd1234EXAMPLEb5678efgh",
```

```
                "content": "These don't appear to be used anywhere. Can we remove
 them?",
                "creationDate": 1508369622.123,
                "deleted": false,
                "lastModifiedDate": 1508369622.123,
                "callerReactions": [],
                "reactionCounts": []
            }
             "location": {
                "filePath": "ahs_count.py",
                "filePosition": 367,
                "relativeFileVersion": "AFTER"
             },
        "repositoryName": "MyDemoRepo",
        "pullRequestId": "47"
 }
```

2. 풀 요청에 대한 주석을을 보려면 다음을 지정하여 get-comments-for-pull-request 명령을 실행합니다.

   - CodeCommit 리포지토리의 이름(`--repository-name` 옵션 사용).
   - 풀 요청의 시스템 생성 ID(`--pull-request-id` 옵션 사용).
   - (선택 사항) 다음 결과 세트를 반환하는 열거형 토큰(`--next-token` 옵션 사용).
   - (선택 사항) 반환된 결과의 수를 제한하는 음수가 아닌 정수(`--max-results` 옵션 사용).

   예를 들어 다음 명령을 사용하여 ID가 42인 풀 요청에 대한 의견을 볼 수 있습니다.

```
aws codecommit get-comments-for-pull-request --pull-request-id 42
```

   이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
{
   "commentsForPullRequestData": [
      {
         "afterBlobId": "1f330709EXAMPLE",
         "afterCommitId": "5d036259EXAMPLE",
         "beforeBlobId": "80906a4cEXAMPLE",
         "beforeCommitId": "317f8570EXAMPLE",
         "comments": [
            {
               "authorArn": "arn:aws:iam::111111111111:user/Saanvi_Sarkar",
```

```
                "clientRequestToken": "",
                "commentId": "abcd1234EXAMPLEb5678efgh",
                "content": "These don't appear to be used anywhere. Can we remove
 them?",
                "creationDate": 1508369622.123,
                "deleted": false,
                "lastModifiedDate": 1508369622.123,
                "callerReactions": [],
                "reactionCounts":
                 {
                   "THUMBSUP" : 6,
                   "CONFUSED" : 1
                 }
            },
            {
                "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
                "clientRequestToken": "",
                "commentId": "442b498bEXAMPLE5756813",
                "content": "Good catch. I'll remove them.",
                "creationDate": 1508369829.104,
                "deleted": false,
                "lastModifiedDate": 150836912.273,
                "callerReactions": ["THUMBSUP"]
                "reactionCounts":
                 {
                   "THUMBSUP" : 14
                 }
             }
        ],
        "location": {
            "filePath": "ahs_count.py",
            "filePosition": 367,
            "relativeFileVersion": "AFTER"
        },
        "repositoryName": "MyDemoRepo",
        "pullRequestId": "42"
      }
   ],
   "nextToken": "exampleToken"
}
```

3. 풀 요청을 승인하거나 승인을 취소하려면 다음을 지정하여 update-pull-request-approval-state 명령을 실행합니다.

- 풀 요청의 ID(--pull-request-id 옵션 사용).
- 풀 요청의 개정 ID(--revision-id option) 사용). get-pull-request 명령을 사용하여 풀 요청의 현재 개정 ID를 얻을 수 있습니다.
- 적용할 승인 상태(--approval-state 옵션 사용). 유효한 승인 상태에는 APPROVE 및 REVOKE가 포함됩니다.

예를 들어 다음 명령을 사용하여 ID가 *27*이고 개정 ID가 *9f29d167EXAMPLE*인 풀 요청을 승인합니다.

```
aws codecommit update-pull-request-approval-state --pull-request-id 27 --revision-
id 9f29d167EXAMPLE --approval-state "APPROVE"
```

성공한 경우 이 명령은 아무 것도 반환하지 않습니다.

4. 풀 요청의 주석에 댓글을 달려면 다음을 지정하여 post-comment-reply 명령을 실행합니다.

- 댓글을 달 주석의 시스템 생성 ID(--in-reply-to 옵션 사용).
- 클라이언트에서 생성된 고유한 멱등성 토큰(--client-request-token 옵션 사용).
- 댓글의 내용(--content 옵션 사용).

예를 들어, *"Good catch. I'll remove them."* 라는 댓글을 *abcd1234EXAMPLEb5678efgh*라는 시스템 생성 ID로 해당 주석에 추가하려면 다음 명령을 사용합니다.

```
aws codecommit post-comment-reply --in-reply-to abcd1234EXAMPLEb5678efgh --
content "Good catch. I'll remove them." --client-request-token 123Example
```

이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
{
    "comment": {
        "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
        "clientRequestToken": "123Example",
        "commentId": "442b498bEXAMPLE5756813",
        "content": "Good catch. I'll remove them.",
        "creationDate": 1508369829.136,
        "deleted": false,
```

```
        "lastModifiedDate": 150836912.221,
        "callerReactions": [],
        "reactionCounts": []
    }
}
```

풀 요청 업데이트

풀 요청 업데이트는 공개 풀 요청의 소스 브랜치에 커밋을 푸시하여 추가 코드를 변경하는 것으로 수행할 수 있습니다. 자세한 내용은 에서 커밋 생성 AWS CodeCommit 단원을 참조하십시오.

AWS CodeCommit 콘솔 또는를 사용하여 풀 요청의 제목 또는 설명을 AWS CLI 업데이트할 수 있습니다. 다음과 같은 경우에는 풀 요청의 제목이나 설명을 업데이트 하는 것이 좋을 수도 있습니다.

- 다른 사용자가 설명을 이해하지 못하거나 원래 제목이 오해의 소지가 있을 경우.
- 보류 중인 풀 요청의 소스 브랜치에 대한 변경 사항을 제목이나 설명에 반영하려는 경우.

풀 요청 업데이트 (콘솔)

CodeCommit 콘솔을 사용하여 CodeCommit 리포지토리에서 풀 요청의 제목 및 설명을 업데이트할 수 있습니다. 풀 요청에서 코드를 업데이트하려면 공개 풀 요청의 소스 브랜치에 커밋을 푸시합니다.

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서, 풀 요청을 업데이트하려는 리포지토리의 이름을 선택합니다.
3. 탐색 창에서 풀 요청을 선택합니다.
4. 기본적으로 모든 미해결 풀 요청 목록이 표시됩니다. 업데이트하려는 진행 중 풀 요청을 선택합니다.
5. 풀 요청에서 세부 정보를 선택한 다음 세부 정보 편집을 선택하여 제목 또는 설명을 편집합니다.

> **Note**
>
> 닫히거나 병합된 풀 요청의 제목 또는 설명은 업데이트할 수 없습니다.

풀 요청 업데이트 (AWS CLI)

CodeCommit에서 AWS CLI 명령을 사용하려면를 설치합니다 AWS CLI. 자세한 내용은 명령줄 참조 단원을 참조하십시오.

다음 명령도 사용할 수 있습니다.

- update-pull-request-approval-state - 풀 요청을 승인하거나 승인 취소합니다.
- create-pull-request-approval-rule - 풀 요청에 대한 승인 규칙을 생성합니다.
- delete-pull-request-approval-rule - 풀 요청에 대한 승인 규칙을 삭제합니다.
- 를 사용하여 커밋 생성 AWS CLI 또는 Git 클라이언트를 사용하여 커밋 생성 - 코드의 추가 변경 사항을 생성하여 공개 풀 요청의 소스 브랜치에 푸시합니다.

AWS CLI 를 사용하여 CodeCommit 리포지토리에서 풀 요청을 업데이트하려면

1. 리포지토리에서 풀 요청의 제목을 업데이트하려면 다음을 지정하여 update-pull-request-title 명령을 실행합니다.

   - 풀 요청의 ID(--pull-request-id 옵션 사용).
   - 풀 요청의 이름(--title 옵션 사용).

   예를 들어 ID가 *47*인 풀 요청의 제목을 업데이트하려면:

```
aws codecommit update-pull-request-title --pull-request-id 47 --title
 "Consolidation of global variables - updated review"
```

   이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
{
    "pullRequest": {
        "approvalRules": [
            {
                "approvalRuleContent": "{\"Version\": \"2018-11-08\",
\"DestinationReferences\": [\"refs/heads/main\"],\"Statements\": [{\"Type
\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers\":
 [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
                "approvalRuleId": "dd8b17fe-EXAMPLE",
                "approvalRuleName": "2-approver-rule-for-main",
                "creationDate": 1571356106.936,
```

```
                "lastModifiedDate": 571356106.936,
                "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
                "originApprovalRuleTemplate": {
                    "approvalRuleTemplateId": "dd8b26gr-EXAMPLE",
                    "approvalRuleTemplateName": "2-approver-rule-for-main"
                },
                "ruleContentSha256": "4711b576EXAMPLE"
            }
        ],
        "authorArn": "arn:aws:iam::123456789012:user/Li_Juan",
        "clientRequestToken": "",
        "creationDate": 1508530823.12,
        "description": "Review the latest changes and updates to the global
 variables. I have updated this request with some changes, including removing some
 unused variables.",
        "lastActivityDate": 1508372657.188,
        "pullRequestId": "47",
        "pullRequestStatus": "OPEN",
        "pullRequestTargets": [
            {
                "destinationCommit": "9f31c968EXAMPLE",
                "destinationReference": "refs/heads/main",
                "mergeMetadata": {
                    "isMerged": false,
                },
                "repositoryName": "MyDemoRepo",
                "sourceCommit": "99132ab0EXAMPLE",
                "sourceReference": "refs/heads/variables-branch"
            }
        ],
        "title": "Consolidation of global variables - updated review"
    }
}
```

2. 풀 요청의 설명을 업데이트하려면 다음을 지정하여 update-pull-request-description 명령을 실행합니다.

   - 풀 요청의 ID(--pull-request-id 옵션 사용).
   - 설명(--description 옵션 사용).

   예를 들어 ID가 *47*인 풀 요청의 설명을 업데이트하려면 다음과 같이 합니다.

```
aws codecommit update-pull-request-description --pull-request-id 47 --description
 "Updated the pull request to remove unused global variable."
```

이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
{
   "pullRequest": {
      "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
      "clientRequestToken": "",
      "creationDate": 1508530823.155,
      "description": "Updated the pull request to remove unused global variable.",
      "lastActivityDate": 1508372423.204,
      "pullRequestId": "47",
      "pullRequestStatus": "OPEN",
      "pullRequestTargets": [
         {
            "destinationCommit": "9f31c968EXAMPLE",
            "destinationReference": "refs/heads/main",
            "mergeMetadata": {
               "isMerged": false,
            },
            "repositoryName": "MyDemoRepo",
            "sourceCommit": "99132ab0EXAMPLE",
            "sourceReference": "refs/heads/variables-branch"
         }
      ],
      "title": "Consolidation of global variables"
   }
}
```

풀 요청에 대한 승인 규칙 편집 또는 삭제

풀 요청에 대한 승인 규칙이 있는 경우 해당 조건이 충족될 때까지 해당 풀 요청을 병합할 수 없습니다. 풀 요청에 대한 승인 규칙을 변경하여 해당 조건을 충족하기 더 쉽게 완화하거나 검토의 엄격성을 강화할 수 있습니다. 풀 요청을 승인해야 하는 사용자 수를 변경할 수 있습니다. 규칙에 대한 승인 사용자 풀에서 멤버십을 추가, 제거 또는 변경할 수도 있습니다. 마지막으로, 풀 요청에 대한 승인 규칙을 더 이상 사용하지 않으려면 삭제할 수 있습니다.

> **Note**
>
> 풀 요청에 대한 승인 규칙을 재정의할 수도 있습니다. 자세한 내용은 풀 요청에 대한 승인 규칙 재정의 단원을 참조하십시오.

AWS CodeCommit 콘솔 또는를 사용하여 AWS CLI 리포지토리에 대한 승인 규칙을 편집하고 삭제할 수 있습니다.

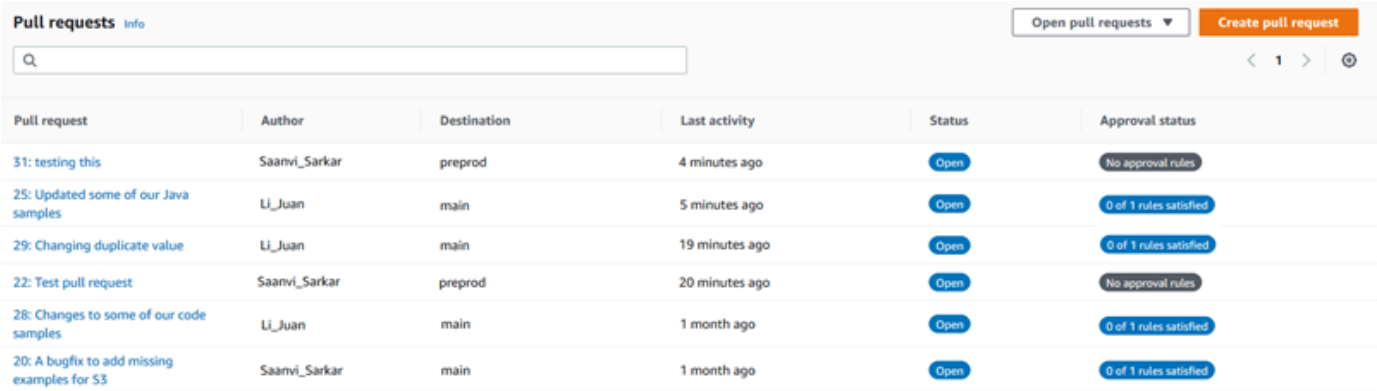
주제

- 풀 요청에 대한 승인 규칙 편집 또는 삭제 (콘솔)
- 풀 요청에 대한 승인 규칙 편집 또는 삭제 (AWS CLI)

풀 요청에 대한 승인 규칙 편집 또는 삭제 (콘솔)

CodeCommit 콘솔을 사용하여 CodeCommit 리포지토리에서 풀 요청에 대한 승인 규칙을 편집 또는 삭제할 수 있습니다.

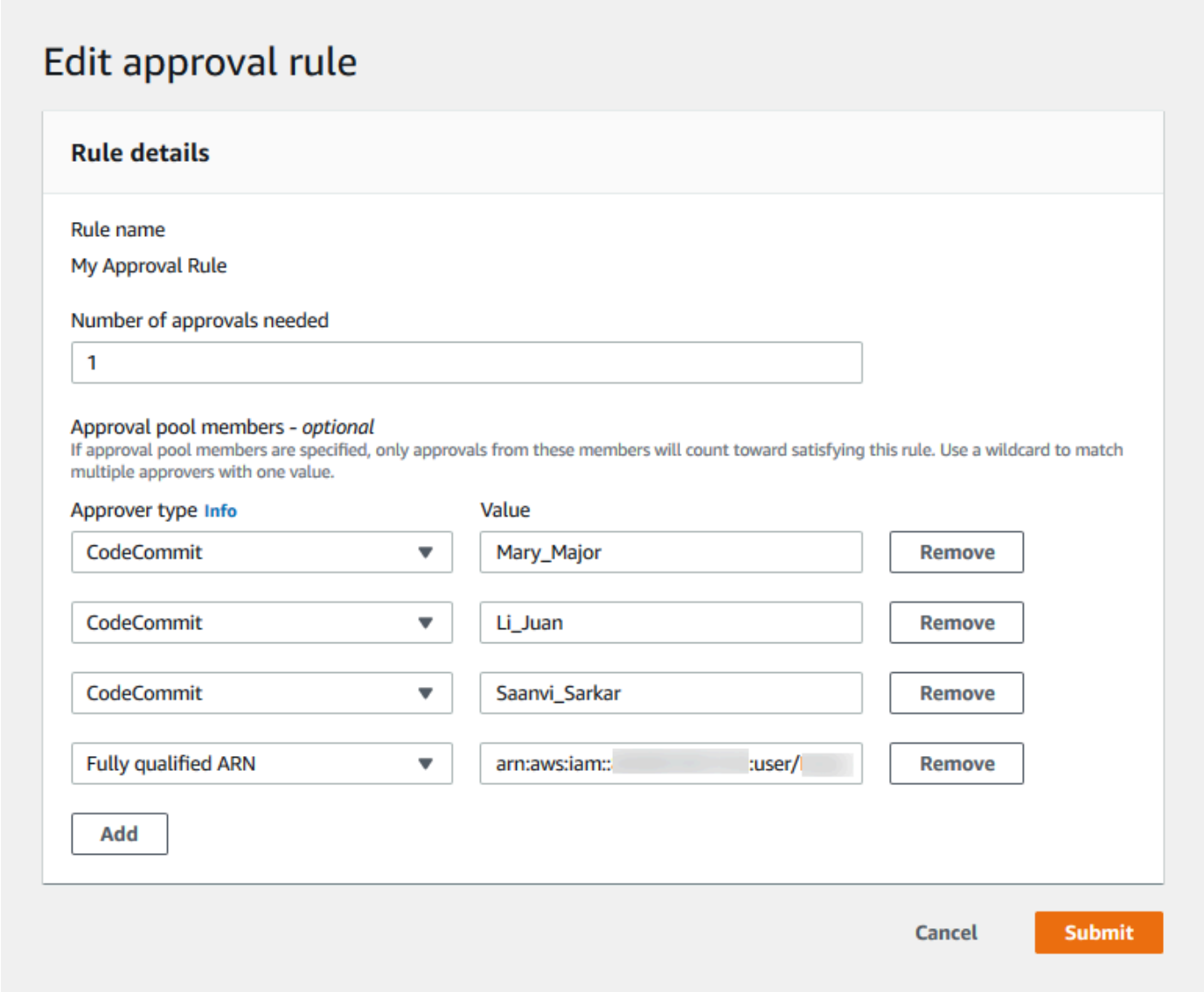
1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서 풀 요청에 대한 승인 규칙을 편집하거나 삭제할 리포지토리의 이름을 선택합니다.
3. 탐색 창에서 풀 요청을 선택합니다.
4. 승인 규칙을 편집하거나 삭제할 풀 요청을 선택합니다. 미결 상태인 풀 요청에 대한 승인 규칙만 편집하거나 삭제할 수 있습니다.



5. 풀 요청에서 승인을 선택한 다음, 목록에서 편집하거나 삭제할 규칙을 선택합니다. 다음 중 하나를 수행합니다.

- 규칙을 편집하려면 편집을 선택합니다.
- 규칙을 삭제하려면 삭제를 선택한 다음 규칙 삭제를 확인하는 지침을 따릅니다.

6. 승인 규칙 편집에서 규칙을 원하는 대로 변경한 다음 제출을 선택합니다.



7. 승인 규칙 구성을 완료했으면 제출을 선택합니다.

풀 요청에 대한 승인 규칙 편집 또는 삭제 (AWS CLI)

CodeCommit에서 AWS CLI 명령을 사용하려면를 설치합니다 AWS CLI. 자세한 내용은 명령줄 참조 단원을 참조하십시오.

AWS CLI 를 사용하여 승인 규칙의 내용을 편집하고 승인 규칙을 삭제할 수 있습니다.

> **Note**
>
> 다음 명령도 사용할 수 있습니다.
>
> - update-pull-request-approval-state - 풀 요청을 승인하거나 승인 취소합니다.
> - get-pull-request-approval-states - 풀 요청에 대한 승인 상태를 확인합니다.
> - evaluate-pull-request-approval-rules - 풀 요청에 대한 승인 규칙이 조건을 만족했는지 여부를 확인합니다.

AWS CLI 를 사용하여 CodeCommit 리포지토리에서 풀 요청에 대한 승인 규칙을 편집하거나 삭제하려면

1. 승인 규칙을 편집하려면 다음을 지정하여 update-pull-request-approval-rule-content 명령을 실행합니다.

   - 풀 요청의 ID(--id 옵션 사용).
   - 승인 규칙의 이름(--approval-rule-name 옵션 사용).
   - 승인 규칙의 내용(--approval-rule-content 옵션 사용).

   이 예제에서는 ID가 *27#* 풀 요청에 대해 *Require two approved approvers*라는 이름의 승인 규칙을 업데이트합니다. 이 규칙에는 *123456789012* Amazon Web Services 계정의 모든 IAM 사용자를 포함하는 승인 풀에서 한 명의 사용자 승인이 필요합니다.

```
aws codecommit update-pull-request-approval-rule-content --pull-request-id 27
 --approval-rule-name "Require two approved approvers" --approval-rule-content
 "{Version: 2018-11-08, Statements: [{Type: \"Approvers\", NumberOfApprovalsNeeded:
 1, ApprovalPoolMembers:[\"CodeCommitApprovers:123456789012:user/*\"]}]}}"
```

2. 이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
{
    "approvalRule": {
        "approvalRuleContent": "{Version: 2018-11-08, Statements:
 [{Type: \"Approvers\", NumberOfApprovalsNeeded: 1, ApprovalPoolMembers:
[\"CodeCommitApprovers:123456789012:user/*\"]}]}}",
        "approvalRuleId": "aac33506-EXAMPLE",
        "originApprovalRuleTemplate": {},
```

```
        "creationDate": 1570752871.932,
        "lastModifiedDate": 1570754058.333,
        "approvalRuleName": Require two approved approvers",
        "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
        "ruleContentSha256": "cd93921cEXAMPLE",
    }
}
```

3. 승인 규칙을 삭제하려면 다음을 지정하여 delete-pull-request-approval-rule 명령을 실행합니다.

   - 풀 요청의 ID(--id 옵션 사용).
   - 승인 규칙의 이름(--approval-rule-name 옵션 사용).

   예를 들어 ID가 *15*인 풀 요청에 대해 *My Approval Rule*이라는 이름의 승인 규칙을 삭제하려면 다음과 같이 합니다.

```
aws codecommit delete-pull-request-approval-rule --pull-request-id 15 --approval-
rule-name "My Approval Rule"
```

   이 명령이 제대로 실행되면 다음과 비슷한 출력이 반환됩니다.

```
{
    "approvalRuleId": "077d8e8a8-EXAMPLE"
}
```

풀 요청에 대한 승인 규칙 재정의

일반적인 개발 과정에서는 풀 요청을 병합하기 전에 사용자가 승인 규칙 조건을 충족하도록 합니다. 그러나 풀 요청 병합을 신속히 처리해야 하는 경우가 있습니다. 예를 들어, 프로덕션 환경에 버그 수정을 넣을 수 있지만 승인 풀의 아무도 풀 요청을 승인할 수 없습니다. 이와 같은 경우 풀 요청에 대한 승인 규칙을 재정의하도록 선택할 수 있습니다. 풀 요청에 대해 특별히 생성되고 승인 규칙 템플릿에서 생성된 승인 규칙을 포함하여 풀 요청에 대한 모든 승인 규칙을 재정의할 수 있습니다. 특정 승인 규칙을 선택적으로 재정의할 수는 없으며 모든 규칙을 재정의할 수만 있습니다. 승인 규칙을 재정의하여 규칙 요구 사항을 무시한 후 풀 요청을 대상 브랜치에 병합할 수 있습니다.

풀 요청에 대한 승인 규칙을 재정의하면 규칙을 재정의한 사용자에 대한 정보가 풀 요청 활동에 기록됩니다. 그러므로 풀 요청 기록으로 돌아가서 규칙을 재정의한 사람을 확인할 수 있습니다. 풀 요청이 미

결 상태인 경우 재정의를 취소하도록 선택할 수도 있습니다. 풀 요청이 병합된 후에는 더 이상 재정의를 취소할 수 없습니다.

주제

- 승인 규칙 재정의 (콘솔)
- 승인 규칙 재정의 (AWS CLI)

승인 규칙 재정의 (콘솔)

콘솔에서 풀 요청 검토의 일환으로 풀 요청에 대한 승인 규칙의 요구 사항을 재정의할 수 있습니다. 생각이 바뀔 경우 재정의를 취소할 수 있으며, 그러면 승인 규칙 요구 사항이 다시 적용됩니다. 풀 요청이 아직 미결 상태인 경우에만 승인 규칙을 재정의하거나 재정의를 취소할 수 있습니다. 병합되거나 닫힌 경우에는 재정의 상태를 변경할 수 없습니다.

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서 리포지토리의 이름을 선택합니다.
3. 탐색 창에서 풀 요청을 선택합니다. 승인 규칙 요구 사항을 재정의하거나 재정의를 취소할 풀 요청을 선택합니다.
4. 승인 탭에서 승인 규칙 재정의를 선택합니다. 요구 사항이 무시되며 버튼 텍스트가 취소 재정의로 변경됩니다. 승인 규칙 요구 사항을 다시 적용하려면 재정의 취소를 선택합니다.

승인 규칙 재정의 (AWS CLI)

를 사용하여 승인 규칙 요구 사항을 재정의 AWS CLI 할 수 있습니다. 또한 풀 요청에 대한 재정의 상태를 확인하는 데 사용할 수 있습니다.

풀 요청에 대한 승인 규칙 요구 사항 재정의

1. 터미널 또는 명령줄에서 다음을 지정하여 override-pull-request-approval-rules 명령을 실행합니다.
   - 풀 요청의 시스템 생성 ID.
   - 풀 요청의 최신 개정 ID. 이 정보를 보려면 get-pull-request를 사용합니다.
   - 원하는 재지정 상태, `OVERRIDE` 또는 `REVOKE`. `REVOKE` 상태는 `OVERRIDE`를 제거하지만 저장되지는 않습니다.

예를 들어 ID가 **34**이고 개정 ID가 **927df8d8EXAMPLE**인 풀 요청에 대한 승인 규칙을 재정의하려면 다음과 같이 합니다.

```
aws codecommit override-pull-request-approval-rules --pull-request-id 34 --
revision-id 927df8d8dEXAMPLE --override-status OVERRIDE
```

2. 성공한 경우 이 명령은 아무 것도 반환하지 않습니다.
3. ID가 **34**이고 개정 ID가 **927df8d8EXAMPLE**인 풀 요청에 대한 재정의를 취소하려면

```
aws codecommit override-pull-request-approval-rules --pull-request-id 34 --
revision-id 927df8d8dEXAMPLE --override-status REVOKE
```

풀 리퀘스트의 재정의 상태에 대한 정보 가져오기

1. 터미널 또는 명령줄에서 다음을 지정하여 get-pull-request-override-state 명령을 실행합니다.

   - 풀 요청의 시스템 생성 ID.
   - 풀 요청의 최신 개정 ID. 이 정보를 보려면 get-pull-request를 사용합니다.

   예를 들어 ID가 **34**이고 개정 ID가 **927df8d8EXAMPLE**인 풀 요청의 재정의 상태를 보려면 다음과 같이 합니다.

```
aws codecommit get-pull-request-override-state --pull-request-id 34 --revision-
id 927df8d8dEXAMPLE
```

2. 이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
{
    "overridden": true,
    "overrider": "arn:aws:iam::123456789012:user/Mary_Major"
}
```

AWS CodeCommit 리포지토리에서 풀 요청 병합

코드 검토가 완료되고 풀 요청의 모든 승인 규칙(있는 경우)이 충족되면 다음과 같은 여러 방법 중 하나로 풀 요청을 병합할 수 있습니다.

- 콘솔에서 사용 가능한 병합 전략 중 하나를 사용하여 소스 브랜치를 대상 브랜치에 병합할 수 있습니다. 그러면 풀 요청도 종료됩니다. 또한 콘솔에서 병합 충돌을 해결할 수 있습니다. 콘솔에는 풀 요청이 병합 가능한지 또는 충돌을 해결해야 하는지 나타내는 메시지가 표시됩니다. 모든 충돌을 해결하고 병합을 선택하면 선택한 병합 전략을 사용하여 병합이 수행됩니다. fast-forward는 기본 병합 전략으로, Git에 대한 기본 옵션입니다. 소스 브랜치와 대상 브랜치의 코드 상태에 따라 그 전략의 사용이 불가능할 수 있지만, squash 또는 3방향 같은 다른 옵션을 사용할 수 있습니다.
- AWS CLI 를 사용하여 빠른 전달, 스쿼시 또는 3방향 병합 전략을 사용하여 풀 요청을 병합하고 닫을 수 있습니다.
- 로컬 컴퓨터에서 git merge 명령을 사용하여 대상 브랜치에 소스 브랜치를 병합한 다음 병합한 코드를 대상 브랜치에 푸시할 수 있습니다. 이 방법에는 신중하게 고려해야 할 단점이 있습니다. 풀 요청에 대한 승인 규칙의 요구 사항이 충족되었는지 여부에 관계없이 풀 요청을 병합하여 해당 컨트롤을 우회합니다. fast-forward 병합 전략을 사용하여 풀 요청을 병합하면 대상 브랜치를 병합하고 푸시할 때 자동으로 풀 요청도 종료됩니다. 이 방법의 한 가지 장점은 git merge 명령을 사용하여 CodeCommit 콘솔에서 사용할 수 없는 병합 옵션이나 전략을 선택할 수 있다는 것입니다. git merge 명령 및 병합 옵션에 대한 자세한 내용은 git-merge 또는 Git 설명서를 참조하세요.

CodeCommit은 풀 요청의 소스 브랜치 또는 대상 브랜치가 삭제될 경우 풀 요청을 자동으로 종료합니다.

주제

- 풀 요청 병합 (콘솔)
- 풀 요청 병합 (AWS CLI)

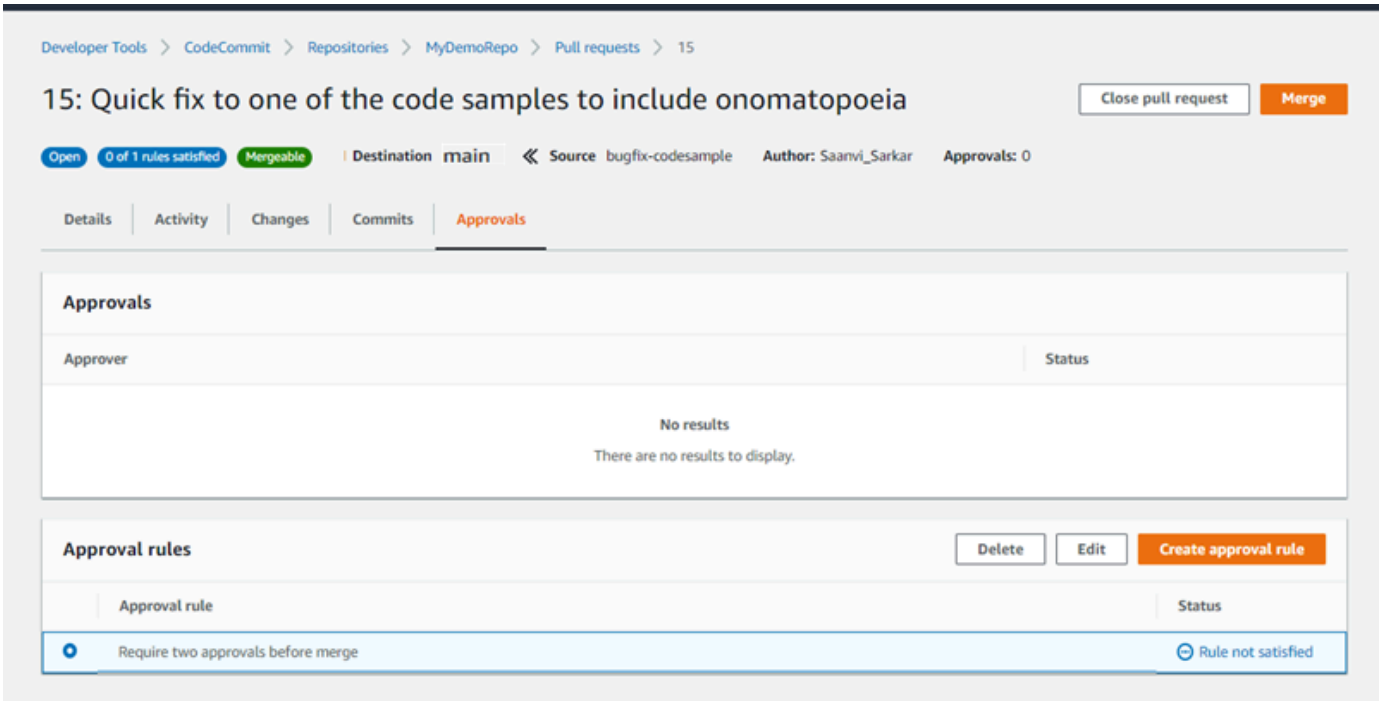
풀 요청 병합 (콘솔)

CodeCommit 콘솔을 사용하여 CodeCommit 리포지토리의 풀 요청을 병합할 수 있습니다. 풀 요청의 상태가 병합됨으로 변경된 후 미해결 풀 요청 목록에 더 이상 표시되지 않습니다. 병합된 풀 요청은 종결로 분류됩니다. 다시 미해결로 변경할 수는 없지만, 여전히 변경에 대한 설명을 남기고 설명에 댓글을 달 수 있습니다. 풀 요청이 병합되거나 종료된 후에는 해당 요청을 승인하거나 승인을 취소하거나 풀 요청에 적용된 승인 규칙을 재정의할 수 없습니다.

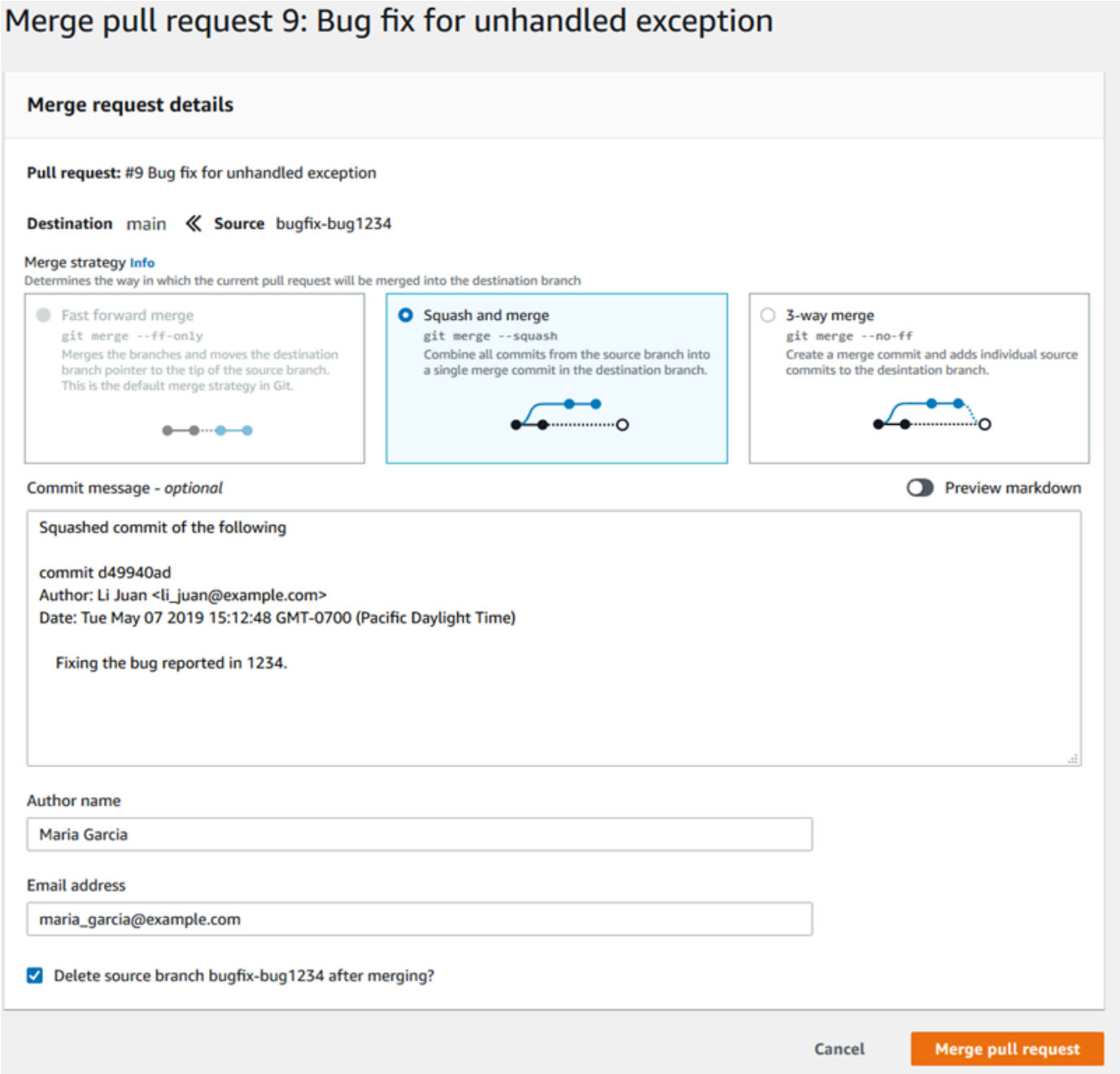
1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서 리포지토리의 이름을 선택합니다.
3. 탐색 창에서 풀 요청을 선택합니다.

4. 기본적으로 모든 미해결 풀 요청 목록이 표시됩니다. 병합하려는 미해결 풀 요청을 선택합니다.
5. 풀 요청에서 승인을 선택합니다. 승인자 목록을 검토하고 모든 승인 규칙(있는 경우)에서 해당 조건이 충족되었는지 확인합니다. 하나 이상의 승인 규칙이 상태가 규칙이 충족되지 않음인 경우 풀 요청을 병합할 수 없습니다. 아무도 풀 요청을 승인하지 않은 경우 병합할 것인지 또는 승인을 기다릴 것인지 고려하세요.

> **Note**
>
> 풀 요청에 대한 승인 규칙을 생성한 경우 이를 편집하거나 삭제하여 병합 차단을 해제할 수 있습니다. 승인 규칙 템플릿을 사용하여 승인 규칙을 생성한 경우에는 편집하거나 삭제할 수 없습니다. 요구 사항을 재정의하도록 선택할 수만 있습니다. 자세한 내용은 풀 요청에 대한 승인 규칙 재정의 단원을 참조하십시오.



6. 병합을 선택합니다.
7. 풀 요청에서 사용 가능한 병합 전략 중에서 선택합니다. 적용할 수 없는 병합 전략은 회색으로 표시됩니다. 병합 전략을 사용할 수 없는 경우 CodeCommit 콘솔에서 충돌을 수동으로 해결하도록 선택하거나 Git 클라이언트를 사용하여 로컬로 충돌을 해결할 수 있습니다. 자세한 내용은 AWS CodeCommit 리포지토리에서 풀 요청의 충돌 해결 단원을 참조하십시오.



- fast-forward 병합은 대상 브랜치의 참조를 소스 브랜치의 가장 최근 커밋으로 전진시킵니다. 가능한 경우 이는 Git의 기본 동작입니다. 병합 커밋은 생성되지 않지만 소스 브랜치의 모든 커밋 이력이 마치 대상 브랜치에서 발생한 것처럼 유지됩니다. fast-forward 병합은 병합 커밋이 생성되지 않기 때문에 대상 브랜치 기록의 커밋 시각화 도우미 보기에 브랜치 병합으로 나타나지 않습니다. 소스 브랜치의 팁은 대상 브랜치의 팁으로 빠르게 넘어갑니다.
- squash 병합은 소스 브랜치의 변경 내용을 포함하는 하나의 커밋을 생성하고 단일 squash된 커밋을 대상 브랜치에 적용합니다. 기본적으로 squash 커밋의 커밋 메시지에는 소스 브랜치의 변경 사항에 대한 모든 커밋 메시지가 포함됩니다. 브랜치 변경 내용의 개별 커밋 이력은 유지되지

않습니다. 이렇게 하면 대상 브랜치 이력의 커밋 시각화 도우미 보기에서 병합의 그래픽 표현을 유지하면서도 리포지토리 이력을 단순화할 수 있습니다.

- 3방향 병합은 대상 브랜치의 병합을 위한 병합 커밋을 생성하지만 대상 브랜치 이력의 일환으로 소스 브랜치에서 실행된 개별 커밋도 유지합니다. 이렇게 하면 리포지토리 변경 내용의 완전한 이력을 유지할 수 있습니다.

8. squash 또는 3방향 병합 전략을 선택하면 정보를 변경하려는 경우 자동으로 생성된 커밋 메시지를 검토하고 수정합니다. 커밋 이력에 대해 사용자의 이름과 이메일 주소를 추가합니다.
9. (선택 사항) 병합의 일환으로 소스 브랜치를 삭제하는 옵션을 선택 취소합니다. 기본값은 풀 요청이 병합될 때 소스 브랜치를 삭제하는 것입니다.
10. 풀 요청 병합을 선택하여 병합을 완료합니다.

풀 요청 병합 (AWS CLI)

CodeCommit에서 AWS CLI 명령을 사용하려면를 설치합니다 AWS CLI. 자세한 내용은 명령줄 참조 단원을 참조하십시오.

AWS CLI 를 사용하여 CodeCommit 리포지토리에서 풀 요청을 병합하려면

1. 풀 요청이 모든 승인 규칙이 충족되고 병합할 준비가 되었는지 여부를 평가하려면 다음을 지정하여 evaluate-pull-request-approval-rules 명령을 실행합니다.

   - 풀 요청의 ID(--pull-request-id 옵션 사용).
   - 풀 요청의 개정 ID(--revision-id option) 사용). get-pull-request 명령을 사용하여 풀 요청에 대한 현재 개정 ID를 가져올 수 있습니다.

   예를 들어 ID가 *27*이고 개정 ID가 *9f29d167EXAMPLE*인 풀 요청의 승인 규칙 상태를 평가하려면 다음과 같이 하십시오.

   ```
   aws codecommit evaluate-pull-request-approval-rules --pull-request-id 27 --
   revision-id 9f29d167EXAMPLE
   ```

   이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

   ```
   {
       "evaluation": {
           "approved": false,
   ```

```
        "approvalRulesNotSatisfied": [
            "Require two approved approvers"
        ],
        "overridden": false,
        "approvalRulesSatisfied": []
    }
}
```

> **Note**
>
> 이 출력은 한 승인 규칙의 요구 사항이 충족되지 않아 풀 요청을 병합할 수 없음을 나타냅니다. 이 풀 요청을 병합하려면 규칙의 조건을 충족하도록 검토자가 요청을 승인하도록 할 수 있습니다. 권한 및 규칙 생성 방법에 따라 규칙을 편집, 재정의 또는 삭제할 수도 있습니다. 자세한 내용은 풀 요청 검토, 풀 요청에 대한 승인 규칙 재정의, 풀 요청에 대한 승인 규칙 편집 또는 삭제 섹션을 참조하세요.

2. fast-forward merge 전략을 사용하여 풀 요청을 병합하고 닫으려면 다음을 지정하여 merge-pull-request-by-fast-forward 명령을 실행합니다.

   - 풀 요청의 ID(--pull-request-id 옵션 사용).
   - 소스 브랜치 말단의 전체 커밋 ID(--source-commit-id 옵션 사용).
   - 리포지토리의 이름(--repository-name 옵션 사용).

   예를 들어, *MyDemoRepo*라는 리포지토리에서 ID가 *47*이고 소스 커밋 ID가 *99132ab0EXAMPLE*인 풀 요청을 병합하고 종료하려면 다음과 같이 합니다.

```
aws codecommit merge-pull-request-by-fast-forward --pull-request-id 47 --source-
commit-id 99132ab0EXAMPLE --repository-name MyDemoRepo
```

   이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
{
    "pullRequest": {
        "approvalRules": [
            {
                "approvalRuleContent": "{\"Version\": \"2018-11-08\",\"Statements
\": [{\"Type\": \"Approvers\",\"NumberOfApprovalsNeeded\": 1,\"ApprovalPoolMembers
\": [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
                "approvalRuleId": "dd8b17fe-EXAMPLE",
```

```
                "approvalRuleName": "I want one approver for this pull request",
                "creationDate": 1571356106.936,
                "lastModifiedDate": 571356106.936,
                "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
                "ruleContentSha256": "4711b576EXAMPLE"
            }
        ],
        "authorArn": "arn:aws:iam::123456789012:user/Li_Juan",
        "clientRequestToken": "",
        "creationDate": 1508530823.142,
        "description": "Review the latest changes and updates to the global
 variables",
        "lastActivityDate": 1508887223.155,
        "pullRequestId": "47",
        "pullRequestStatus": "CLOSED",
        "pullRequestTargets": [
            {
                "destinationCommit": "9f31c968EXAMPLE",
                "destinationReference": "refs/heads/main",
                "mergeMetadata": {
                    "isMerged": true,
                    "mergedBy": "arn:aws:iam::123456789012:user/Mary_Major"
                },
                "repositoryName": "MyDemoRepo",
                "sourceCommit": "99132ab0EXAMPLE",
                "sourceReference": "refs/heads/variables-branch"
            }
        ],
        "title": "Consolidation of global variables"
    }
}
```

3. squash 병합 전략을 사용하여 풀 요청을 병합하고 닫으려면 다음을 지정하여 merge-pull-request-by-squash 명령을 실행합니다.

   - 풀 요청의 ID(--pull-request-id 옵션 사용).
   - 소스 브랜치 말단의 전체 커밋 ID(--source-commit-id 옵션 사용).
   - 리포지토리의 이름(--repository-name 옵션 사용).
   - 사용하려는 충돌 세부 정보 수준(--conflict-detail-level 옵션 사용). 미지정된 경우 기본 **FILE_LEVEL**이 사용됩니다.

- 사용하려는 충돌 해결 전략(--conflict-resolution-strategy 옵션 사용). 미지정된 경우 기본 설정은 NONE이며 충돌은 수동으로 해결해야 합니다.
- 포함시킬 커밋 메시지(--commit-message 옵션 사용).
- 커밋에 사용할 이름(--author-name 옵션 사용).
- 커밋에 사용할 이메일 주소(--email 옵션 사용).
- 빈 폴더의 유지 여부(--keep-empty-folders 옵션 사용).

다음 예제에서는 *MyDemoRepo*라는 리포지토리에서 ID가 *47*이고 소스 커밋 ID가 *99132ab0EXAMPLE*인 풀 요청을 병합하고 종료합니다. 이 예제는 충돌 세부 사항에 LINE_LEVEL을 사용하고 충돌 해결 전략에 ACCEPT_SOURCE를 사용합니다.

```
aws codecommit merge-pull-request-by-squash --pull-request-id 47 --source-commit-
id 99132ab0EXAMPLE --repository-name MyDemoRepo --conflict-detail-level LINE_LEVEL
 --conflict-resolution-strategy ACCEPT_SOURCE --author-name "Jorge Souza" --email
 "jorge_souza@example.com" --commit-message "Merging pull request 47 by squash and
 accepting source in merge conflicts"
```

이 명령이 제대로 실행되면 다음과 유사한 fast-forward 병합의 출력과 같은 종류의 출력이 생성됩니다.

```
{
    "pullRequest": {
        "approvalRules": [
            {
                "approvalRuleContent": "{\"Version\": \"2018-11-08\",
\"DestinationReferences\": [\"refs/heads/main\"],\"Statements\": [{\"Type
\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers\":
 [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
                "approvalRuleId": "dd8b17fe-EXAMPLE",
                "approvalRuleName": "2-approver-rule-for-main",
                "creationDate": 1571356106.936,
                "lastModifiedDate": 571356106.936,
                "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
                "originApprovalRuleTemplate": {
                    "approvalRuleTemplateId": "dd8b17fe-EXAMPLE",
                    "approvalRuleTemplateName": "2-approver-rule-for-main"
                },
                "ruleContentSha256": "4711b576EXAMPLE"
            }
```

```
        ],
        "authorArn": "arn:aws:iam::123456789012:user/Li_Juan",
        "clientRequestToken": "",
        "creationDate": 1508530823.142,
        "description": "Review the latest changes and updates to the global
 variables",
        "lastActivityDate": 1508887223.155,
        "pullRequestId": "47",
        "pullRequestStatus": "CLOSED",
        "pullRequestTargets": [
            {
                "destinationCommit": "9f31c968EXAMPLE",
                "destinationReference": "refs/heads/main",
                "mergeMetadata": {
                    "isMerged": true,
                    "mergedBy": "arn:aws:iam::123456789012:user/Mary_Major"
                },
                "repositoryName": "MyDemoRepo",
                "sourceCommit": "99132ab0EXAMPLE",
                "sourceReference": "refs/heads/variables-branch"
            }
        ],
        "title": "Consolidation of global variables"
    }
}
```

4. 3방향 병합 전략을 사용하여 풀 요청을 병합하고 닫으려면 다음을 지정하여 merge-pull-request-by-three-way 명령을 실행합니다.

   - 풀 요청의 ID(--pull-request-id 옵션 사용).
   - 소스 브랜치 말단의 전체 커밋 ID(--source-commit-id 옵션 사용).
   - 리포지토리의 이름(--repository-name 옵션 사용).
   - 사용하려는 충돌 세부 정보 수준(--conflict-detail-level 옵션 사용). 미지정된 경우 기본 **FILE_LEVEL**이 사용됩니다.
   - 사용하려는 충돌 해결 전략(--conflict-resolution-strategy 옵션 사용). 미지정된 경우 기본 설정은 NONE이며 충돌은 수동으로 해결해야 합니다.
   - 포함시킬 커밋 메시지(--commit-message 옵션 사용).
   - 커밋에 사용할 이름(--author-name 옵션 사용).
   - 커밋에 사용할 이메일 주소(--email 옵션 사용).
   - 빈 폴더의 유지 여부(--keep-empty-folders 옵션 사용).

다음 예제에서는 *MyDemoRepo*라는 리포지토리에서 ID가 *47*이고 소스 커밋 ID가 *99132ab0EXAMPLE*인 풀 요청을 병합하고 종료합니다. 이 예제는 충돌 세부 정보 및 충돌 해결 전략에 기본 옵션을 사용합니다.

```
aws codecommit merge-pull-request-by-three-way --pull-request-id 47 --source-
commit-id 99132ab0EXAMPLE --repository-name MyDemoRepo --author-name "Maria Garcia"
 --email "maria_garcia@example.com" --commit-message "Merging pull request 47 by
 three-way with default options"
```

이 명령이 제대로 실행되면 다음과 유사한 fast-forward 병합의 출력과 같은 종류의 출력이 생성됩니다.

```
{
    "pullRequest": {
        "approvalRules": [
            {
                "approvalRuleContent": "{\"Version\": \"2018-11-08\",
\"DestinationReferences\": [\"refs/heads/main\"],\"Statements\": [{\"Type
\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers\":
 [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
                "approvalRuleId": "dd8b17fe-EXAMPLE",
                "approvalRuleName": "2-approver-rule-for-main",
                "creationDate": 1571356106.936,
                "lastModifiedDate": 571356106.936,
                "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
                "originApprovalRuleTemplate": {
                    "approvalRuleTemplateId": "dd8b17fe-EXAMPLE",
                    "approvalRuleTemplateName": "2-approver-rule-for-main"
                },
                "ruleContentSha256": "4711b576EXAMPLE"
            }
        ],
        "authorArn": "arn:aws:iam::123456789012:user/Li_Juan",
        "clientRequestToken": "",
        "creationDate": 1508530823.142,
        "description": "Review the latest changes and updates to the global
 variables",
        "lastActivityDate": 1508887223.155,
        "pullRequestId": "47",
        "pullRequestStatus": "CLOSED",
```

```
        "pullRequestTargets": [
            {
                "destinationCommit": "9f31c968EXAMPLE",
                "destinationReference": "refs/heads/main",
                "mergeMetadata": {
                    "isMerged": true,
                    "mergedBy": "arn:aws:iam::123456789012:user/Mary_Major"
                },
                "repositoryName": "MyDemoRepo",
                "sourceCommit": "99132ab0EXAMPLE",
                "sourceReference": "refs/heads/variables-branch"
            }
        ],
        "title": "Consolidation of global variables"
    }
}
```

AWS CodeCommit 리포지토리에서 풀 요청의 충돌 해결

풀 요청에 충돌이 발생하여 풀 요청을 병합할 수 없는 경우 다음 방법 중 하나를 사용하여 충돌의 해결을 시도할 수 있습니다.

- 로컬 컴퓨터에서 git diff 명령을 사용하여 2개 브랜치 간의 충돌을 확인하고 충돌을 해결하도록 변경할 수 있습니다. 또한 차이점 도구나 기타 소프트웨어를 사용하여 차이점을 확인하고 해결할 수 있습니다. 충돌을 만족할 수 있을 정도로 해결한 후 해결된 충돌을 포함하는 변경 사항으로 소스 브랜치를 푸시할 수 있으며, 풀 요청이 업데이트됩니다. git diff 및 git difftool에 대한 자세한 내용은 Git 설명서를 참조하세요.
- 콘솔에서 충돌 해결을 선택할 수 있습니다. 이렇게 하면 git diff 명령과 비슷한 방법으로 충돌을 보여주는 일반 텍스트 편집기가 열립니다. 충돌을 포함하는 각 파일의 충돌을 수동으로 검토하고 변경을 실시한 다음 변경 사항으로 풀 요청을 업데이트할 수 있습니다.
- 에서 AWS CLI 를 사용하여 병합 충돌에 대한 정보를 가져오고 참조되지 않은 병합 커밋을 생성하여 병합을 테스트 AWS CLI할 수 있습니다.

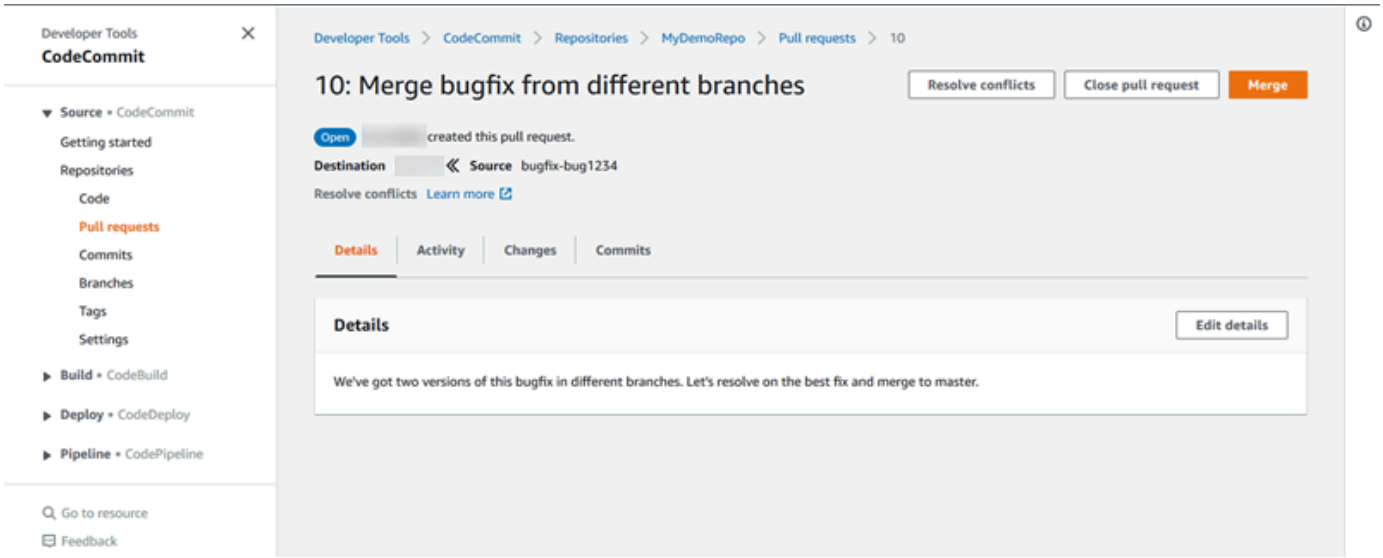
주제

- 풀 요청의 충돌 해결 (콘솔)
- 풀 요청의 충돌 해결 (AWS CLI)

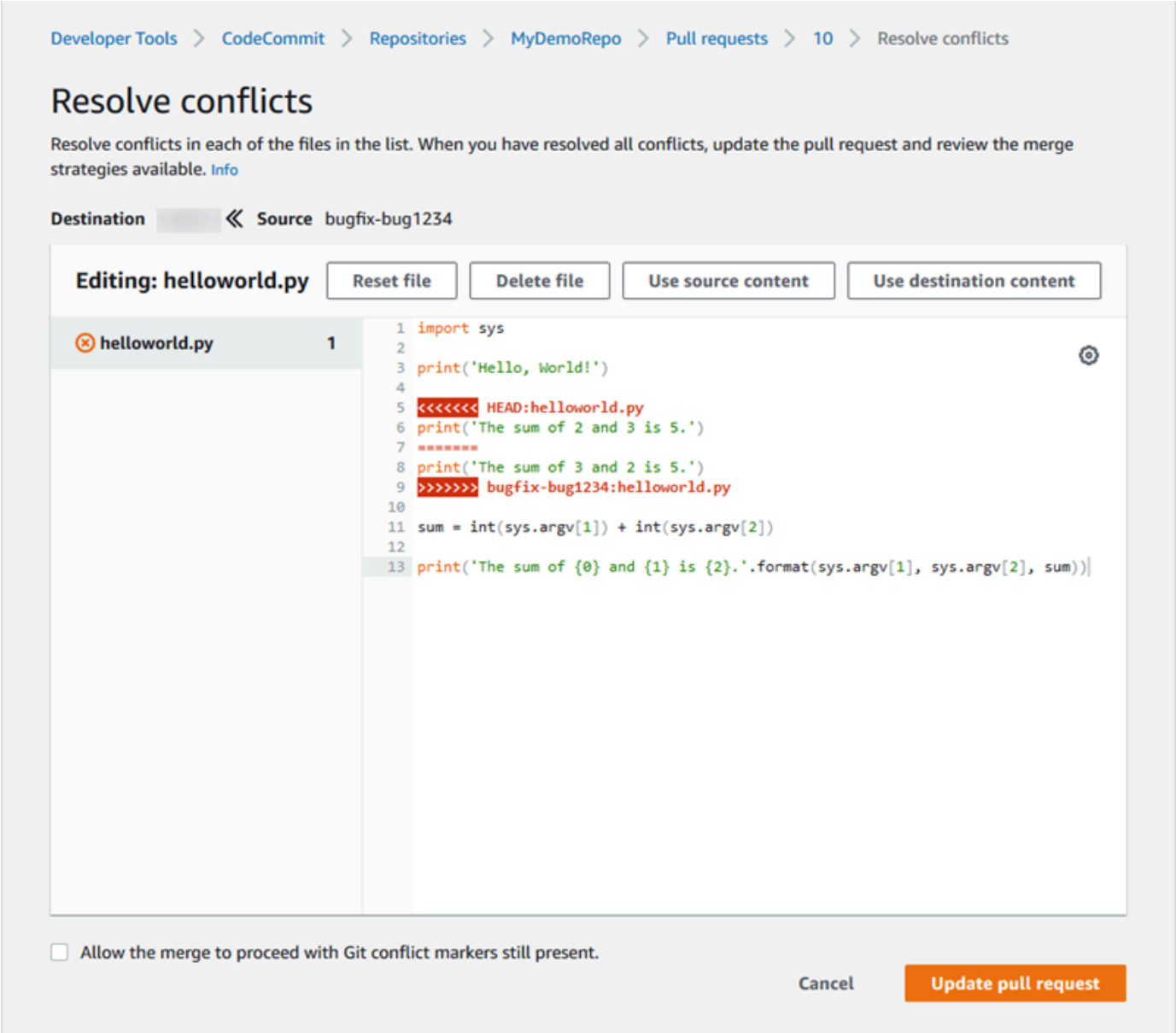
풀 요청의 충돌 해결 (콘솔)

CodeCommit 콘솔을 사용하여 CodeCommit 리포지토리의 풀 요청을 해결할 수 있습니다.

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서 리포지토리의 이름을 선택합니다.
3. 탐색 창에서 풀 요청을 선택합니다.
4. 기본적으로 모든 미해결 풀 요청 목록이 표시됩니다. 병합하려는 미해결 풀 요청을 선택합니다. 이 요청에는 충돌이 포함되어 있습니다.
5. 풀 요청에서 충돌 해결을 선택합니다. 이 옵션은 풀 요청을 병합할 수 있기 전에 해결해야 할 충돌이 있는 경우에만 나타납니다.



6. 충돌 해결 창이 해결해야 할 충돌이 있는 각 파일 목록을 엽니다. 목록에서 충돌을 검토할 각 파일을 선택하고 모든 충돌이 해결될 때까지 필요한 변경을 실시합니다.



- 소스 파일 콘텐츠, 대상 파일 콘텐츠를 사용하도록 선택하거나 파일이 이진 파일이 아닌 경우 원하는 변경 사항만 포함되도록 파일의 콘텐츠를 수동으로 편집하도록 선택할 수 있습니다. 표준 git 차이점 마커는 파일의 대상(HEAD) 브랜치와 소스 브랜치 간의 충돌을 표시하는 데 사용됩니다.
- 파일이 이진 파일, Git 하위 모듈이거나 파일/폴더 이름 충돌이 있는 경우 충돌을 해결하기 위해 소스 파일 또는 대상 파일을 사용하도록 선택해야 합니다. CodeCommit 콘솔에서는 이진 파일을 보거나 편집할 수 없습니다.
- 파일 모드 충돌이 있는 경우 소스 파일의 파일 모드와 대상 파일의 파일 모드 중에서 선택하여 충돌을 해결하는 옵션이 표시됩니다.

- 파일의 변경 사항을 폐기하고 파일을 충돌 상태로 복원하려면 파일 재설정을 선택합니다. 이렇게 하면 다른 방법으로 충돌을 해결할 수 있습니다.

7. 변경 사항에 만족하면 풀 요청 업데이트를 선택합니다.

> **Note**
> 변경 사항으로 풀 요청을 성공적으로 업데이트할 수 있기 전에 모든 파일의 모든 충돌을 해결해야 합니다.

8. 풀 요청은 변경 사항으로 업데이트되며 병합 가능합니다. 병합 페이지가 표시됩니다. 이 시점에 풀 요청을 병합하도록 선택하거나 풀 요청 목록으로 돌아갈 수 있습니다.

풀 요청의 충돌 해결 (AWS CLI)

CodeCommit에서 AWS CLI 명령을 사용하려면를 설치합니다 AWS CLI. 자세한 내용은 명령줄 참조 단원을 참조하십시오.

단일 AWS CLI 명령이 없어 풀 요청의 충돌을 해결하고 해당 요청을 병합할 수 있습니다. 그러나 개별 명령들을 사용하여 충돌을 확인하고 충돌의 해결을 시도하며 풀 요청의 병합 가능 여부를 테스트할 수 있습니다. 다음을 수행할 수 있습니다.

- get-merge-options를 사용하여 2개 커밋 지정자 간의 병합에 사용 가능한 병합 옵션을 알아볼 수 있습니다.
- get-merge-conflicts를 사용하여 2개 커밋 지정자 간의 병합에서 병합 충돌이 있는 파일 목록을 반환할 수 있습니다.
- batch-describe-merge-conflicts를 사용하여, 지정된 병합 전략을 사용하는 2개 커밋 간의 병합에서 파일의 모든 병합 충돌에 대한 정보를 얻을 수 있습니다.
- describe-merge-conflicts를 사용하여, 지정된 병합 전략을 사용하는 2개 커밋 간의 특정 파일의 병합 충돌에 대한 세부 정보를 얻을 수 있습니다.
- create-unreferenced-merge-commit를 사용하여, 지정된 병합 전략을 사용하는 2개 커밋 지정자의 병합 결과를 테스트할 수 있습니다.

1. 2개 커밋 지정자 간의 병합에 사용 가능한 병합 옵션을 확인하려면 다음을 지정하여 get-merge-options 명령을 실행합니다.

   - 병합 소스의 커밋 지정자(--source-commit-specifier 옵션 사용).

- 병합 대상의 커밋 지정자(--destination-commit-specifier 옵션 사용).
- 리포지토리의 이름(--repository-name 옵션 사용).
- (선택 사항) 사용하려는 충돌 해결 전략(--conflict-resolution-strategy 옵션 사용).
- (선택 사항) 충돌에 대해 원하는 세부 사항 수준(--conflict-detail-level 옵션 사용).

예를 들어, *MyDemoRepo*라는 리포지토리에서 *bugfix-1234*라는 소스 브랜치를 main이라는 대상 브랜치와 병합하는 데 활용할 수 있는 병합 전략을 결정하려면, 다음과 같이 합니다.

```
aws codecommit get-merge-options --source-commit-specifier bugfix-1234 --
destination-commit-specifier main --repository-name MyDemoRepo
```

이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
{
    "mergeOptions": [
        "FAST_FORWARD_MERGE",
        "SQUASH_MERGE",
        "THREE_WAY_MERGE"
    ],
    "sourceCommitId": "d49940adEXAMPLE",
    "destinationCommitId": "86958e0aEXAMPLE",
    "baseCommitId": "86958e0aEXAMPLE"
}
```

2. 2개 커밋 지정자 간의 병합에서 병합 충돌을 포함하는 파일 목록을 얻으려면 다음을 지정하여 get-merge-conflicts 명령을 실행합니다.

- 병합 소스의 커밋 지정자(--source-commit-specifier 옵션 사용).
- 병합 대상의 커밋 지정자(--destination-commit-specifier 옵션 사용).
- 리포지토리의 이름(--repository-name 옵션 사용).
- 사용하려는 병합 옵션(--merge-option 옵션 사용).
- (선택 사항) 충돌에 대해 원하는 세부 사항 수준(--conflict-detail-level 옵션 사용).
- (선택 사항) 사용하려는 충돌 해결 전략(--conflict-resolution-strategy 옵션 사용).
- (선택 사항) 반환할 충돌이 있는 파일의 최대 수(--max-conflict-files 옵션 사용).

예를 들어 MyDemoRepo라는 리포지토리에서 three-way merge(삼방향 병합 전략)을 사용하여 feature-randomizationfeature라는 소스 브랜치와 main이라는 대상 브랜치 간의 병합에서 충돌을 포함하는 파일 목록을 얻으려면 다음을 수행합니다.

```
aws codecommit get-merge-conflicts --source-commit-specifier feature-
randomizationfeature --destination-commit-specifier main --merge-option
 THREE_WAY_MERGE --repository-name MyDemoRepo
```

이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
{
    "mergeable": false,
    "destinationCommitId": "86958e0aEXAMPLE",
    "sourceCommitId": "6ccd57fdEXAMPLE",
    "baseCommitId": "767b6958EXAMPLE",
    "conflictMetadataList": [
        {
            "filePath": "readme.md",
            "fileSizes": {
                "source": 139,
                "destination": 230,
                "base": 85
            },
            "fileModes": {
                "source": "NORMAL",
                "destination": "NORMAL",
                "base": "NORMAL"
            },
            "objectTypes": {
                "source": "FILE",
                "destination": "FILE",
                "base": "FILE"
            },
            "numberOfConflicts": 1,
            "isBinaryFile": {
                "source": false,
                "destination": false,
                "base": false
            },
            "contentConflict": true,
            "fileModeConflict": false,
```

```
            "objectTypeConflict": false,
            "mergeOperations": {
                "source": "M",
                "destination": "M"
            }
        }
    ]
}
```

3. 2개 커밋 지정자 간의 병합에서 모든 파일 또는 파일 하위 집합의 병합 충돌에 대한 정보를 얻으려면 다음을 지정하여 batch-describe-merge-conflicts 명령을 실행합니다.

- 병합 소스의 커밋 지정자(--source-commit-specifier 옵션 사용).
- 병합 대상의 커밋 지정자(--destination-commit-specifier 옵션 사용).
- 사용하려는 병합 옵션(--merge-option 옵션 사용).
- 리포지토리의 이름(--repository-name 옵션 사용).
- (선택 사항) 사용하려는 충돌 해결 전략(--conflict-resolution-strategy 옵션 사용).
- (선택 사항) 충돌에 대해 원하는 세부 사항 수준(--conflict-detail-level 옵션 사용).
- (선택 사항) 반환할 병합 헝크의 최대 수(--max-merge-hunks 옵션 사용).
- (선택 사항) 반환할 충돌이 있는 파일의 최대 수(--max-conflict-files 옵션 사용).
- (선택 사항) 충돌을 설명하는 데 사용하기 위한 대상 파일의 경로(--file-paths 옵션 사용).

예를 들어, *MyDemoRepo*라는 리포지토리에서 *### ##(THREE_WAY_MERGE)* 전략을 활용하여 *feature-randomizationfeature*라는 소스 브랜치를 main이라는 대상 브랜치와 병합할 때 발생하는 병합 충돌을 확인하려면, 다음과 같이 합니다.

```
aws codecommit batch-describe-merge-conflicts --source-commit-specifier feature-
randomizationfeature --destination-commit-specifier main --merge-
option THREE_WAY_MERGE --repository-name MyDemoRepo
```

이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
{
    "conflicts": [
        {
            "conflictMetadata": {
                "filePath": "readme.md",
```

```
                "fileSizes": {
                    "source": 139,
                    "destination": 230,
                    "base": 85
                },
                "fileModes": {
                    "source": "NORMAL",
                    "destination": "NORMAL",
                    "base": "NORMAL"
                },
                "objectTypes": {
                    "source": "FILE",
                    "destination": "FILE",
                    "base": "FILE"
                },
                "numberOfConflicts": 1,
                "isBinaryFile": {
                    "source": false,
                    "destination": false,
                    "base": false
                },
                "contentConflict": true,
                "fileModeConflict": false,
                "objectTypeConflict": false,
                "mergeOperations": {
                    "source": "M",
                    "destination": "M"
                }
            },
            "mergeHunks": [
                {
                    "isConflict": true,
                    "source": {
                        "startLine": 0,
                        "endLine": 3,
                        "hunkContent": "VGhpcyBpEXAMPLE=="
                    },
                    "destination": {
                        "startLine": 0,
                        "endLine": 1,
                        "hunkContent": "VXNlIHRoEXAMPLE="
                    }
                }
            ]
```

```
        }
    ],
    "errors": [],
    "destinationCommitId": "86958e0aEXAMPLE",
    "sourceCommitId": "6ccd57fdEXAMPLE",
    "baseCommitId": "767b6958EXAMPLE"
}
```

4. 2개 커밋 지정자 간의 병합에서 특정 파일의 병합 충돌에 대한 세부 정보를 얻으려면 다음을 지정하여 describe-merge-conflicts 명령을 실행합니다.

- 병합 소스의 커밋 지정자(--source-commit-specifier 옵션 사용).
- 병합 대상의 커밋 지정자(--destination-commit-specifier 옵션 사용).
- 사용하려는 병합 옵션(--merge-option 옵션 사용).
- 충돌을 설명하는 데 사용하기 위한 대상 파일의 경로(--file-path 옵션 사용).
- 리포지토리의 이름(--repository-name 옵션 사용).
- (선택 사항) 사용하려는 충돌 해결 전략(--conflict-resolution-strategy 옵션 사용).
- (선택 사항) 충돌에 대해 원하는 세부 사항 수준(--conflict-detail-level 옵션 사용).
- (선택 사항) 반환할 병합 헝크의 최대 수(--max-merge-hunks 옵션 사용).
- (선택 사항) 반환할 충돌이 있는 파일의 최대 수(--max-conflict-files 옵션 사용).

예를 들어, *MyDemoRepo*라는 리포지토리에서 *### ##(THREE_WAY_MERGE)* 전략을 활용하여 *feature-randomizationfeature*라는 소스 브랜치와 main이라는 대상 브랜치에 있는 *readme.md*라는 파일을 병합할 때 발생하는 충돌을 확인하려면, 다음과 같이 합니다.

```
aws codecommit describe-merge-conflicts --source-commit-specifier feature-
randomizationfeature --destination-commit-specifier main --merge-
option THREE_WAY_MERGE --file-path readme.md --repository-name MyDemoRepo
```

이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
{
    "conflictMetadata": {
        "filePath": "readme.md",
        "fileSizes": {
            "source": 139,
            "destination": 230,
```

```
            "base": 85
        },
        "fileModes": {
            "source": "NORMAL",
            "destination": "NORMAL",
            "base": "NORMAL"
        },
        "objectTypes": {
            "source": "FILE",
            "destination": "FILE",
            "base": "FILE"
        },
        "numberOfConflicts": 1,
        "isBinaryFile": {
            "source": false,
            "destination": false,
            "base": false
        },
        "contentConflict": true,
        "fileModeConflict": false,
        "objectTypeConflict": false,
        "mergeOperations": {
            "source": "M",
            "destination": "M"
        }
    },
    "mergeHunks": [
        {
            "isConflict": true,
            "source": {
                "startLine": 0,
                "endLine": 3,
                "hunkContent": "VGhpcyBpEXAMPLE=="
            },
            "destination": {
                "startLine": 0,
                "endLine": 1,
                "hunkContent": "VXNlIHRoEXAMPLE="
            }
        }
    ],
    "destinationCommitId": "86958e0aEXAMPLE",
    "sourceCommitId": "6ccd57fdEXAMPLE",
    "baseCommitId": "767b69580EXAMPLE"
```

```
}
```

5. 2개 커밋 지정자의 병합 결과를 나타내는 참조되지 않은 커밋을 생성하려면 다음을 지정하여 create-unreferenced-merge-commit 명령을 실행합니다.

   - 병합 소스의 커밋 지정자(--source-commit-specifier 옵션 사용).
   - 병합 대상의 커밋 지정자(--destination-commit-specifier 옵션 사용).
   - 사용하려는 병합 옵션(--merge-option 옵션 사용).
   - 리포지토리의 이름(--repository-name 옵션 사용).
   - (선택 사항) 사용하려는 충돌 해결 전략(--conflict-resolution-strategy 옵션 사용).
   - (선택 사항) 충돌에 대해 원하는 세부 사항 수준(--conflict-detail-level 옵션 사용).
   - (선택 사항) 포함시킬 커밋 메시지(--commit-message 옵션 사용).
   - (선택 사항) 커밋에 사용할 이름(--name 옵션 사용).
   - (선택 사항) 커밋에 사용할 이메일 주소(--email 옵션 사용).
   - (선택 사항) 빈 폴더의 유지 여부(--keep-empty-folders 옵션 사용).

   예를 들어 *MyDemoRepo*라는 리포지토리에서 소스 수용(ACCEPT_SOURCE) 전략을 사용하여 *bugfix-1234*라는 소스 브랜치를 main이라는 대상 브랜치와 병합할 때 발생하는 병합 충돌을 확인하려면, 다음과 같이 합니다.

```
aws codecommit create-unreferenced-merge-commit --source-commit-
specifier bugfix-1234 --destination-commit-specifier main --merge-option
 THREE_WAY_MERGE --repository-name MyDemoRepo --name "Maria Garcia" --email
 "maria_garcia@example.com" --commit-message "Testing the results of this merge."
```

   이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
{
    "commitId": "4f178133EXAMPLE",
    "treeId": "389765daEXAMPLE"
}
```

AWS CodeCommit 리포지토리에서 풀 요청 닫기

코드를 병합하지 않고 풀 요청을 닫으려면 다음 여러 방법 중 하나를 사용하여 수행할 수 있습니다.

- 이 콘솔에서 코드를 병합하지 않고 풀 요청을 닫을 수 있습니다. git merge 명령을 사용하여 브랜치를 수동으로 병합하려는 경우, 또는 풀 요청 소스 브랜치의 코드가 대상 브랜치에 병합하려는 코드가 아닌 경우, 이 작업을 수행해야 할 수도 있습니다.
- 풀 요청에서 지정한 소스 브랜치를 삭제할 수 있습니다. CodeCommit은 풀 요청의 소스 브랜치 또는 대상 브랜치가 삭제될 경우 풀 요청을 자동으로 종료합니다.
- 에서 풀 요청의 상태를에서 로 업데이트할 AWS CLI수 `OPEN` 있습니다`CLOSED`. 이렇게 하면 코드를 병합하지 않고 풀 요청이 닫힙니다.

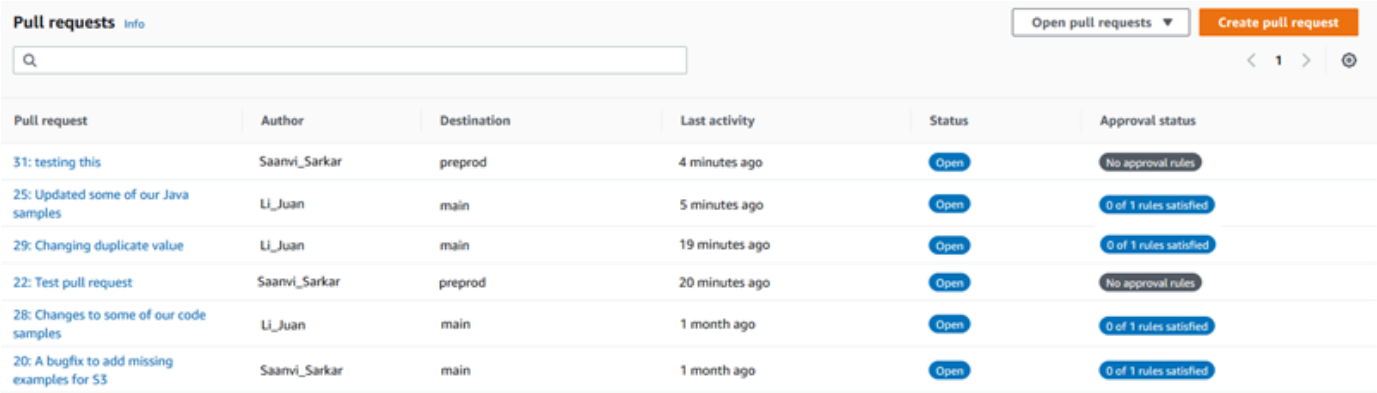
주제

- 풀 요청 닫기 (콘솔)
- 풀 요청 닫기(AWS CLI)

풀 요청 닫기 (콘솔)

CodeCommit 콘솔을 사용하여 CodeCommit 리포지토리에서 풀 요청을 닫을 수 있습니다. 풀 요청의 상태가 종결로 변경되면 다시 미해결로 변경할 수 없지만, 사용자는 여전히 변경 사항에 대한 주석을 남기고 주석에 댓글을 달 수 있습니다.

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서 리포지토리의 이름을 선택합니다.
3. 탐색 창에서 풀 요청을 선택합니다.
4. 기본적으로 모든 미해결 풀 요청 목록이 표시됩니다. 닫으려는 미해결 풀 요청을 선택합니다.



5. 풀 요청에서 풀 요청 닫기를 선택합니다. 이 옵션은 소스 브랜치를 대상 브랜치로 병합하지 않고 풀 요청을 닫습니다. 이 옵션은 풀 요청을 종료하는 과정에서 소스 브랜치를 삭제하는 방법을 제공하지는 않지만, 요청이 종료된 후에 직접 삭제할 수 있습니다.

풀 요청 닫기(AWS CLI)

CodeCommit에서 AWS CLI 명령을 사용하려면를 설치합니다 AWS CLI. 자세한 내용은 명령줄 참조 단원을 참조하십시오.

AWS CLI 를 사용하여 CodeCommit 리포지토리에서 풀 요청을 닫으려면

- 리포지토리의 풀 요청 상태를 OPEN에서 CLOSED로 업데이트하려면 다음을 지정하여 update-pull-request-status 명령을 실행합니다.
  - 풀 요청의 ID(--pull-request-id 옵션 사용).
  - 풀 요청의 상태(--pull-request-status 옵션 사용).

  예를 들어, MyDemoRepo라는 CodeCommit 리포지토리에서 ID가 *42*인 풀 요청의 상태를 *##* 상태로 업데이트하려면 다음과 같이 합니다.

```
aws codecommit update-pull-request-status --pull-request-id 42 --pull-request-
status CLOSED
```

  이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

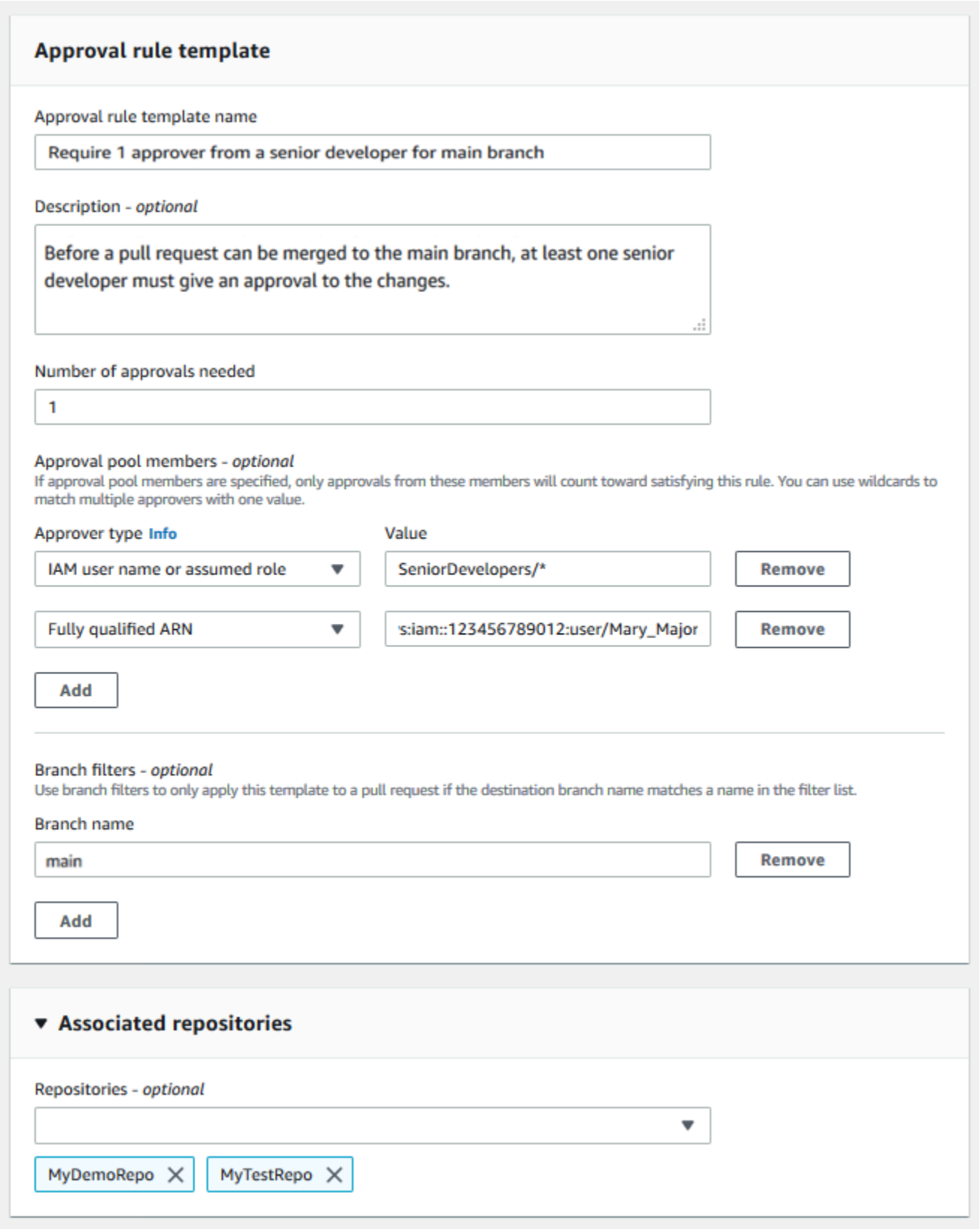
```
{
    "pullRequest": {
        "approvalRules": [
            {
                "approvalRuleContent": "{\"Version\": \"2018-11-08\",\"Statements
\": [{\"Type\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers
\": [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
                "approvalRuleId": "dd8b17fe-EXAMPLE",
                "approvalRuleName": "2-approvers-needed-for-this-change",
                "creationDate": 1571356106.936,
                "lastModifiedDate": 571356106.936,
                "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
                "ruleContentSha256": "4711b576EXAMPLE"
            }
        ],
        "authorArn": "arn:aws:iam::123456789012:user/Li_Juan",
        "clientRequestToken": "",
        "creationDate": 1508530823.165,
        "description": "Updated the pull request to remove unused global
 variable.",
```

```
        "lastActivityDate": 1508372423.12,
        "pullRequestId": "47",
        "pullRequestStatus": "CLOSED",
        "pullRequestTargets": [
            {
                "destinationCommit": "9f31c968EXAMPLE",
                "destinationReference": "refs/heads/main",
                "mergeMetadata": {
                    "isMerged": false,
                },
                "repositoryName": "MyDemoRepo",
                "sourceCommit": "99132ab0EXAMPLE",
                "sourceReference": "refs/heads/variables-branch"
            }
        ],
        "title": "Consolidation of global variables"
    }
}
```

승인 규칙 템플릿 작업

풀 요청에 대한 승인 규칙을 생성할 수 있습니다. 리포지토리에서 생성된 풀 요청의 일부 또는 전부에 승인 규칙을 자동으로 적용하려면 승인 규칙 템플릿을 사용합니다. 승인 규칙 템플릿을 사용하면 리포지토리 사이에서 개발 워크플로를 사용자 지정하여 여러 브랜치에서 적절한 수준의 승인 및 제어 권한을 보유하도록 할 수 있습니다. 프로덕션 및 개발 브랜치에 대해 서로 다른 규칙을 정의할 수 있습니다. 이러한 규칙은 규칙 조건과 일치하는 풀 요청이 생성될 때마다 적용됩니다. 승인 규칙 템플릿에 대한 관리형 정책 및 권한에 대해 자세히 알아보려면 승인 규칙 템플릿에 대한 작업 권한 섹션과 AWS CodeCommit에 대한 관리형 정책 섹션을 참조하세요.

승인 규칙 템플릿을 생성된 AWS 리전 의 하나 이상의 리포지토리와 연결할 수 있습니다. 템플릿이 리포지토리와 연결되어 있으면 풀 요청을 생성하는 과정에서 해당 리포지토리에 풀 요청에 대한 승인 규칙이 자동으로 생성됩니다. 단일 승인 규칙과 마찬가지로 승인 규칙 템플릿은 필요한 승인 수 및 승인을 제공해야 하는 사용자 풀(선택 사항)을 포함하여 승인 규칙 구조를 정의합니다. 승인 규칙과 달리 브랜치 필터라고도 하는 대상 참조(브랜치)를 정의할 수도 있습니다. 대상 참조를 정의하는 경우 대상 브랜치 이름이 템플릿의 지정된 브랜치 이름(대상 참조)과 일치하는 풀 요청에만 규칙이 생성됩니다. 예를 들어 **`refs/heads/main`**을 대상 참조로 지정하면 템플릿에 정의된 승인 규칙이 대상 브랜치가 `main`인 경우의 풀 요청에만 적용됩니다.


Approval rule template
Approval rule template name
Require 1 approver from a senior developer for main branch
Description - optional
Before a pull request can be merged to the main branch, at least one senior developer must give an approval to the changes.
Number of approvals needed
1
Approval pool members - optional
If approval pool members are specified, only approvals from these members will count toward satisfying this rule. You can use wildcards to match multiple approvers with one value.
Approver type Info
Value
IAM user name or assumed role
SeniorDevelopers/*
Remove
Fully qualified ARN
s:iam::123456789012:user/Mary_Major
Remove
Add
Branch filters - optional
Use branch filters to only apply this template to a pull request if the destination branch name matches a name in the filter list.
Branch name
main
Remove
Add
Associated repositories
Repositories - optional
MyDemoRepo
MyTestRepo

주제

- 승인 규칙 템플릿 생성
- 승인 규칙 템플릿을 리포지토리와 연결
- 승인 규칙 템플릿 관리
- 승인 규칙 템플릿 연결 해제
- 승인 규칙 템플릿 삭제

승인 규칙 템플릿 생성

하나 이상의 승인 규칙 템플릿을 생성하여 리포지토리 사이에서 개발 워크플로를 사용자 지정할 수 있습니다. 여러 템플릿을 생성하면 여러 브랜치가 적절한 수준의 승인 및 제어 권한을 보유하도록 승인 규칙의 자동 생성을 구성할 수 있습니다. 예를 들어 프로덕션 브랜치와 개발 브랜치에 서로 다른 템플릿을 생성하여 이러한 템플릿을 하나 이상의 리포지토리에 적용할 수 있습니다. 사용자가 해당 리포지토리에서 풀 요청을 생성하면 해당 템플릿에 대해 요청이 평가됩니다. 요청이 적용된 템플릿의 조건과 일치하면 풀 요청에 대한 승인 규칙이 생성됩니다.

콘솔 또는 AWS CLI 를 사용하여 승인 규칙 템플릿을 생성할 수 있습니다. 승인 규칙 템플릿에 대한 관리형 정책 및 권한에 대해 자세히 알아보려면 승인 규칙 템플릿에 대한 작업 권한 섹션과 AWS CodeCommit에 대한 관리형 정책 섹션을 참조하세요.

주제

- 승인 규칙 템플릿 생성 (콘솔)
- 승인 규칙 템플릿 생성 (AWS CLI)

승인 규칙 템플릿 생성 (콘솔)

승인 규칙 템플릿은 기본적으로 리포지토리와 연결되어 있지 않습니다. 템플릿을 생성할 때 템플릿과 하나 이상의 리포지토리 간에 연결을 생성하거나 나중에 연결을 추가할 수 있습니다.

승인 규칙 템플릿을 생성하려면 (콘솔)

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 승인 규칙 템플릿을 선택한 다음 템플릿 생성을 선택합니다.
3. 승인 규칙 템플릿 이름에서 템플릿에 해당 용도를 알 수 있도록 설명이 포함된 이름을 지정합니다. 예를 들어, 풀 요청을 병합하기 전에 선임 개발자 그룹 중 한 명이 승인하도록 요청하려면 규칙 이름을 **`Require 1 approver from a senior developer`**로 지정할 수 있습니다.

4. (선택 사항) 설명에 이 템플릿의 용도에 대한 설명을 제공합니다. 이렇게 하면 다른 사용자가 이 템플릿이 해당 리포지토리에 적합한지 여부를 결정하는 데 도움이 됩니다.
5. 필요한 승인 수에 원하는 숫자를 입력합니다. 기본 값은 1입니다.
6. (선택 사항) 특정 사용자 그룹에서 풀 요청에 대한 승인을 받도록 하려면 승인 규칙 멤버에서 추가를 선택합니다. 승인자 유형에서 다음 중 하나를 선택합니다.

   - IAM 사용자 이름 또는 수인된 역할: 이 옵션은 로그인하는 데 사용한 계정의 Amazon Web Services 계정 ID를 미리 채워 주며 오로지 이름만 필요로 합니다. 이 옵션은 제공된 이름과 일치하면 IAM 사용자와 페더레이션 액세스 사용자 모두 이용할 수 있습니다. 이 옵션은 폭넓은 유연성을 제공하는 매우 강력한 옵션입니다. 예를 들어, 이 옵션을 선택하고 Amazon Web Services 계정 123456789012로 로그인하여 **Mary_Major**를 지정한 경우 다음은 모두 해당 사용자의 승인으로 간주됩니다.

     - 해당 계정의 IAM 사용자 (`arn:aws:iam::123456789012:user/Mary_Major`)
     - IAM에서 Mary_Major로 식별되는 페더레이션 사용자 (`arn:aws:sts::123456789012:federated-user/Mary_Major`)

     이 옵션은 와일드카드(`*Mary_Major`)가 포함되지 않는 한 역할 세션 이름 Mary_Major(`arn:aws:sts::123456789012:assumed-role/CodeCommitReview/Mary_Major`)를 사용하여 **CodeCommitReview** 역할을 수임한 누군가의 활성 세션을 인식하지 못합니다. 역할 이름(`CodeCommitReview/Mary_Major`)을 명시적으로 지정할 수도 있습니다.

   - 정규화된 ARN: 이 옵션을 사용하면 IAM 사용자 또는 역할의 정규화된 Amazon 리소스 이름(ARN)을 지정할 수 있습니다. 또한이 옵션은 AWS Lambda 및 같은 다른 AWS 서비스에서 사용하는 수임된 역할을 지원합니다 AWS CodeBuild. 수임된 역할의 경우 ARN 형식은 역할 및 기능에 대해 각각 `arn:aws:sts::`*`AccountID`*`:assumed-role/`*`RoleName`* 및 `arn:aws:sts::`*`AccountID`*`:assumed-role/`*`FunctionName` 형식이어야 합니다.

   승인자 유형으로 IAM 사용자 이름 또는 수임된 역할을 선택한 경우, 값에는 IAM 사용자나 역할의 이름 또는 사용자나 역할의 정규화된 ARN을 입력합니다. 필요한 승인 수에 해당 승인이 합산되는 모든 사용자 또는 역할을 추가할 때까지 추가를 다시 선택하여 사용자 또는 역할을 추가합니다.

   두 승인자 유형 모두 값에 와일드카드(*)를 사용할 수 있습니다. 예를 들어, IAM 사용자 이름 또는 수임된 역할 옵션을 선택하고 **CodeCommitReview/***를 지정하면, **CodeCommitReview**의 역할을 맡은 모든 사용자가 승인 풀에 포함됩니다. 개별 역할 세션 이름은 필요한 승인자 수에 포함됩니다. 이러한 방식으로 Mary_Major 및 Li_Juan은 로그인하여 `CodeCommitReview` 역할을 수임

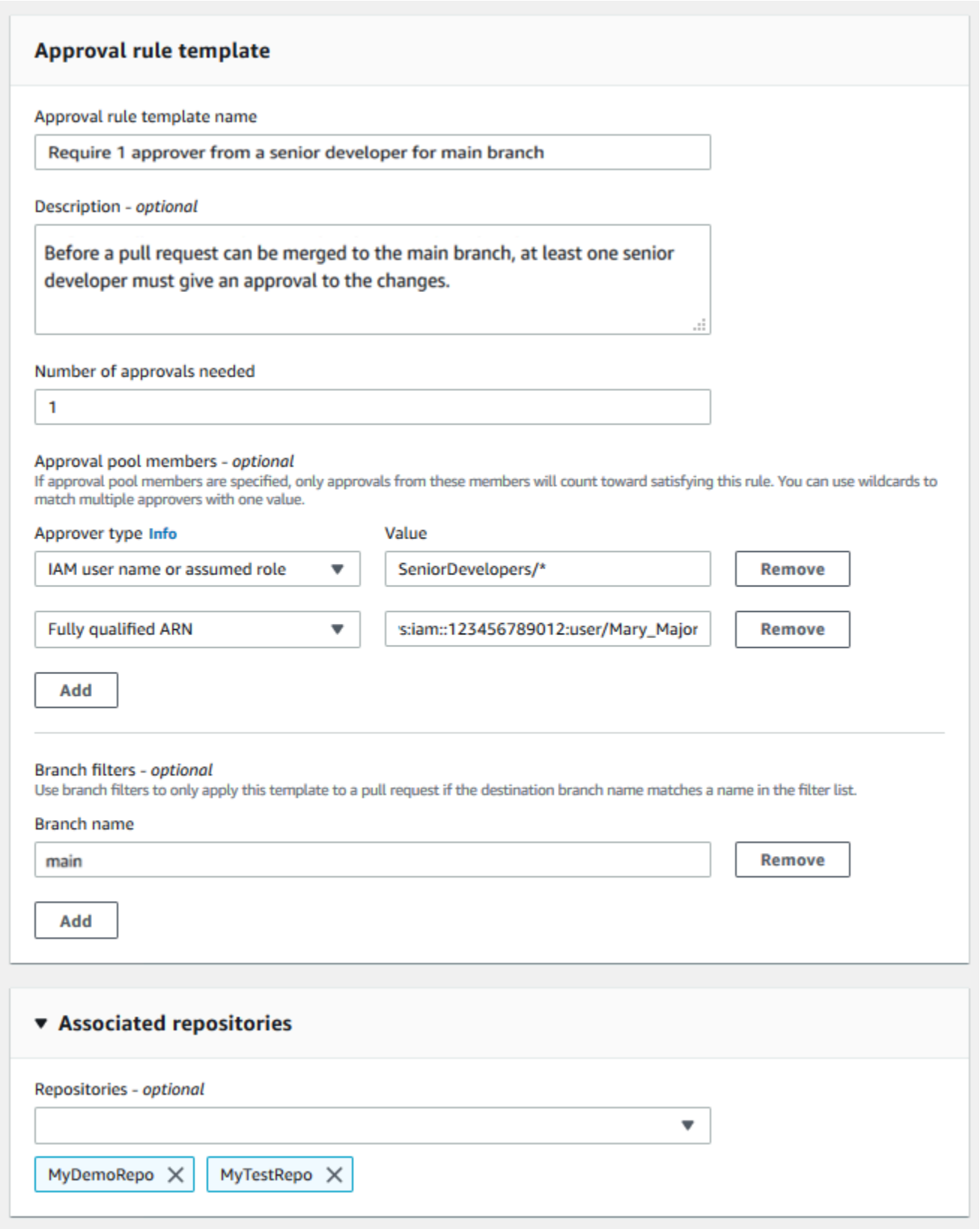
하면 승인으로 합산됩니다. IAM ARN, 와일드카드, 형식 등에 대한 자세한 내용은 IAM 식별자를 참조하세요.

> **Note**
>
> 승인 규칙은 교차 계정 승인을 지원하지 않습니다.

7. (선택 사항) 브랜치 필터에 승인 규칙 생성을 필터링하는 데 사용할 대상 브랜치 이름을 입력합니다. 예를 들어, `main`을 지정할 경우, 풀 요청의 대상 브랜치가 `main`이라는 브랜치인 경우에만 연결된 리포지토리의 풀 요청에 대한 승인 규칙이 생성됩니다. 브랜치 이름에 와일드카드(*)를 사용하여 와일드카드 대소문자와 일치하는 모든 브랜치 이름에 승인 규칙을 적용할 수 있습니다. 그러나 브랜치 이름의 시작 부분에는 와일드카드를 사용할 수 없습니다. 브랜치 이름을 최대 100개까지 지정할 수 있습니다. 필터를 지정하지 않으면 템플릿이 연결된 리포지토리의 모든 브랜치에 적용됩니다.
8. (선택 사항) 연결된 리포지토리의 리포지토리 목록에서 이 승인 규칙과 연결할 이 AWS 리전 의 리포지토리를 선택합니다.

> **Note**
>
> 템플릿을 생성한 후 리포지토리를 연결하도록 선택할 수 있습니다. 자세한 내용은 승인 규칙 템플릿을 리포지토리와 연결 단원을 참조하십시오.

9. 생성을 선택합니다.


Approval rule template
Approval rule template name
Require 1 approver from a senior developer for main branch
Description - optional
Before a pull request can be merged to the main branch, at least one senior developer must give an approval to the changes.
Number of approvals needed
1
Approval pool members - optional
If approval pool members are specified, only approvals from these members will count toward satisfying this rule. You can use wildcards to match multiple approvers with one value.
Approver type Info
Value
IAM user name or assumed role
SeniorDevelopers/*
Remove
Fully qualified ARN
s:iam::123456789012:user/Mary_Major
Remove
Add
Branch filters - optional
Use branch filters to only apply this template to a pull request if the destination branch name matches a name in the filter list.
Branch name
main
Remove
Add
Associated repositories
Repositories - optional
MyDemoRepo
MyTestRepo

승인 규칙 템플릿 생성 (AWS CLI)

AWS CLI 를 사용하여 승인 규칙 템플릿을 생성할 수 있습니다. 를 사용할 때 템플릿에 대한 대상 참조를 지정할 AWS CLI수 있으므로 대상 브랜치가 템플릿의 대상 브랜치와 일치하는 풀 요청에만 적용됩니다.

승인 규칙 템플릿을 만들려면 (AWS CLI)

1. 터미널 또는 명령줄에서 다음을 지정하여 create-approval-rule-template 명령을 실행합니다.

   - 승인 규칙 템플릿을 위한 이름. 용도를 설명하는 이름을 사용하는 것이 좋습니다.
   - 승인 규칙 템플릿에 대한 설명. 이름과 마찬가지로 자세한 설명을 제공하는 것이 좋습니다.
   - 승인 규칙 템플릿의 JSON 구조. 이 구조에는 승인 규칙이 적용되는 풀 요청의 대상 브랜치인 대상 참조에 대한 요구 사항과 필요한 승인 수에 해당 승인이 합산되는 사용자인 승인 풀 멤버가 포함될 수 있습니다.

   승인 규칙의 내용을 작성할 때 다음 두 가지 방법 중 하나로 승인 풀에 승인자를 지정할 수 있습니다.

   - CodeCommitApprovers: 이 옵션에는 Amazon Web Services 계정과 리소스만 필요합니다. 이 옵션은 이름이 제공된 리소스 이름과 일치하는 IAM 사용자 및 페더레이션 액세스 사용자 모두에 사용할 수 있습니다. 이 옵션은 폭넓은 유연성을 제공하는 매우 강력한 옵션입니다. 예를 들어 AWS 계정 123456789012와 **`Mary_Major`**를 지정한 경우 다음은 모두 해당 사용자의 승인으로 간주됩니다.

     - 해당 계정의 IAM 사용자 (`arn:aws:iam::123456789012:user/Mary_Major`)
     - IAM에서 Mary_Major로 식별되는 페더레이션 사용자 (`arn:aws:sts::123456789012:federated-user/Mary_Major`)

     이 옵션은 와일드카드(`*Mary_Major`)가 포함되지 않는 한 역할 세션 이름 *`Mary_Major`*(`arn:aws:sts::123456789012:assumed-role/`*`SeniorDevelopers`*`/`*`Mary_Major`*)를 사용하여 *`SeniorDevelopers`*의 역할을 수임한 누군가의 활성 세션을 인식하지 못합니다.

   - 정규화된 ARN: 이 옵션을 사용하면 IAM 사용자 또는 역할의 정규화된 Amazon 리소스 이름 (ARN)을 지정할 수 있습니다.

   IAM ARN, 와일드카드, 형식 등에 대한 자세한 내용은 IAM 식별자를 참조하세요.

다음 예제에서는 **2-approver-rule-for-main**이라는 승인 규칙 템플릿과 **Requires two developers from the team to approve the pull request if the destination branch is main**에 대한 설명을 생성합니다. 템플릿을 **main** 브랜치에 병합하기 전에 **CodeCommitReview** 역할을 수임한 두 명의 사용자가 풀 요청을 승인해야 합니다.

```
aws codecommit create-approval-rule-template --approval-rule-template-name 2-
approver-rule-for-main --approval-rule-template-description "Requires two
 developers from the team to approve the pull request if the destination branch
 is main" --approval-rule-template-content "{\"Version\": \"2018-11-08\",
\"DestinationReferences\": [\"refs/heads/main\"],\"Statements\": [{\"Type
\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers\":
 [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}"
```

2. 이 명령이 제대로 실행되면 다음과 비슷한 출력이 반환됩니다.

```
{
    "approvalRuleTemplate": {
        "approvalRuleTemplateName": "2-approver-rule-for-main",
        "creationDate": 1571356106.936,
        "approvalRuleTemplateId": "dd8b17fe-EXAMPLE",
        "approvalRuleTemplateContent": "{\"Version\": \"2018-11-08\",
\"DestinationReferences\": [\"refs/heads/main\"],\"Statements\": [{\"Type
\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers\":
 [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
        "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
        "approvalRuleTemplateDescription": "Requires two developers from the team
 to approve the pull request if the destination branch is main",
        "lastModifiedDate": 1571356106.936,
        "ruleContentSha256": "4711b576EXAMPLE"
    }
}
```

승인 규칙 템플릿을 리포지토리와 연결

승인 규칙 템플릿은 특정에서 생성 AWS 리전되지만 연결 AWS 리전 될 때까지 해당의 리포지토리에는 영향을 주지 않습니다. 하나 이상의 리포지토리에 템플릿을 적용하려면 해당 템플릿을 리포지토리와 연결해야 합니다. 한 AWS 리전의 여러 리포지토리에 단일 템플릿을 적용할 수 있습니다. 이렇게 하면 풀 요청을 승인 및 병합하기 위한 일관된 조건을 생성하여 리포지토리의 개발 워크플로를 자동화하고 표준화할 수 있습니다.

승인 규칙 템플릿이 생성된 AWS 리전 의 리포지토리에만 승인 규칙 템플릿을 연결할 수 있습니다.

승인 규칙 템플릿에 대한 관리형 정책 및 권한에 대해 자세히 알아보려면 승인 규칙 템플릿에 대한 작업 권한 섹션과 AWS CodeCommit에 대한 관리형 정책 섹션을 참조하세요.

주제

- 승인 규칙 템플릿 연결 (콘솔)
- 승인 규칙 템플릿 연결 (AWS CLI)

승인 규칙 템플릿 연결 (콘솔)

리포지토리를 생성할 때 승인 규칙 템플릿과 연결할 수 있습니다. (이 단계는 선택 사항입니다.) 템플릿을 편집하여 연결을 추가하거나 제거할 수 있습니다.

승인 규칙 템플릿을 리포지토리와 연결하려면

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 승인 규칙 템플릿을 선택합니다. 템플릿을 선택한 다음 편집을 선택합니다.
3. 연결된 리포지토리의 리포지토리 목록에서 리포지토리를 선택합니다. 연결된 각 리포지토리가 목록 상자 아래에 나타납니다.
4. 저장을 선택합니다. 이제 승인 규칙은 연결된 리포지토리에서 생성된 모든 풀 요청에 적용됩니다.

승인 규칙 템플릿 연결 (AWS CLI)

AWS CLI 를 사용하여 승인 규칙 템플릿을 하나 이상의 리포지토리와 연결할 수 있습니다.

템플릿을 단일 리포지토리와 연결하려면

1. 터미널 또는 명령줄에서 다음을 지정하여 associate-approval-rule-template-with-repository 명령을 실행합니다.

   - 리포지토리와 연결할 승인 규칙 템플릿의 이름.
   - 승인 규칙 템플릿과 연결될 리포지토리의 이름.

   예를 들어 *2-approver-rule-for-main*이라는 승인 규칙 템플릿을 *MyDemoRepo*라는 리포지토리와 연결하려면 다음과 같이 합니다.

```
aws codecommit associate-approval-rule-template-with-repository --repository-
name MyDemoRepo --approval-rule-template-name 2-approver-rule-for-main
```

2. 성공한 경우 이 명령은 아무 것도 반환하지 않습니다.

템플릿을 여러 리포지토리와 연결하려면

1. 터미널 또는 명령줄에서 다음을 지정하여 batch-associate-approval-rule-template-with-repositories 명령을 실행합니다.

   - 리포지토리와 연결할 승인 규칙 템플릿의 이름.
   - 승인 규칙 템플릿과 연결될 리포지토리의 이름.

   예를 들어 **2-approver-rule-for-main**이라는 승인 규칙 템플릿을 **MyDemoRepo** 및 **MyOtherDemoRepo**라는 리포지토리와 연결하려면 다음과 같이 하세요.

```
aws codecommit batch-associate-approval-rule-template-with-repositories --
repository-names "MyDemoRepo", "MyOtherDemoRepo" --approval-rule-template-name 2-
approver-rule-for-main
```

2. 이 명령이 제대로 실행되면 다음과 비슷한 출력이 반환됩니다.

```
{
    "associatedRepositoryNames": [
        "MyDemoRepo",
        "MyOtherDemoRepo"
    ],
    "errors": []
}
```

승인 규칙 템플릿 관리

에서 승인 규칙 템플릿을 관리 AWS 리전 하여 사용 방식과 용도를 파악할 수 있습니다. 예를 들어 승인 규칙 템플릿의 이름과 설명을 편집하여 다른 사용자가 해당 용도를 이해할 수 있도록 할 수 있습니다. 의 모든 승인 규칙 템플릿을 나열 AWS 리전하고 템플릿의 콘텐츠 및 구조에 대한 정보를 가져올 수 있습니다. 한 리포지토리와 연결된 템플릿 그리고 한 템플릿과 연결된 리포지토리를 검토할 수 있습니다.

승인 규칙 템플릿에 대한 관리형 정책 및 권한에 대해 자세히 알아보려면 승인 규칙 템플릿에 대한 작업 권한 섹션과 AWS CodeCommit에 대한 관리형 정책 섹션을 참조하세요.

승인 규칙 템플릿 관리 (콘솔)

CodeCommit 콘솔에서 승인 규칙 템플릿을 보고 관리할 수 있습니다.

승인 규칙 템플릿을 관리하려면

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 승인 규칙 템플릿을 선택하여 로그인 AWS 리전 한의 승인 규칙 템플릿 목록을 봅니다.

   > **Note**
   > 승인 규칙 템플릿은 템플릿 AWS 리전 이 생성된 에서만 사용할 수 있습니다.

3. 템플릿을 변경하려면 목록에서 템플릿을 선택한 다음 편집을 선택합니다.
4. 내용을 변경하고 저장을 선택합니다.

승인 규칙 템플릿 관리 (AWS CLI)

다음 AWS CLI 명령을 사용하여 승인 규칙 템플릿을 관리할 수 있습니다.

- list-approval-rule-templates,의 모든 승인 규칙 템플릿 목록을 보려면 AWS 리전
- get-approval-rule-template - 승인 규칙 템플릿의 내용을 봅니다.
- update-approval-rule-template-content - 승인 규칙 템플릿의 내용을 변경합니다.
- update-approval-rule-template-name - 승인 규칙 템플릿의 이름을 변경합니다.
- update-approval-rule-template-description - 승인 규칙 템플릿의 설명을 변경합니다.
- list-repositories-for-approval-rule-template - 승인 규칙 템플릿과 연결된 모든 리포지토리를 봅니다.
- list-associated-approval-rule-templates-for-repository - 리포지토리와 연결된 모든 승인 규칙 템플릿을 봅니다.

AWS 리전의 모든 승인 규칙 템플릿을 나열하려면

1. 터미널 또는 명령줄에서 list-approval-rule-templates 명령을 실행합니다. 예를 들어, 미국 동부(오하이오) 리전의 모든 승인 규칙 템플릿을 나열하려면 다음과 같이 합니다.

```
aws codecommit list-approval-rule-templates --region us-east-2
```

2. 이 명령이 제대로 실행되면 다음과 비슷한 출력이 반환됩니다.

```
{
    "approvalRuleTemplateNames": [
        "2-approver-rule-for-main",
        "1-approver-rule-for-all-pull-requests"
    ]
}
```

승인 규칙 템플릿의 내용 가져오기

1. 터미널 또는 명령줄에서 승인 규칙 템플릿의 이름을 지정하여 get-approval-rule-template 명령을 실행합니다.

```
aws codecommit get-approval-rule-template --approval-rule-template-name 1-approver-
rule-for-all-pull-requests
```

2. 이 명령이 제대로 실행되면 다음과 비슷한 출력이 반환됩니다.

```
{
    "approvalRuleTemplate": {
        "approvalRuleTemplateContent": "{\"Version\": \"2018-11-08\",\"Statements
\": [{\"Type\": \"Approvers\",\"NumberOfApprovalsNeeded\": 1,\"ApprovalPoolMembers
\": [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
        "ruleContentSha256": "621181bbEXAMPLE",
        "lastModifiedDate": 1571356106.936,
        "creationDate": 1571356106.936,
        "approvalRuleTemplateName": "1-approver-rule-for-all-pull-requests",
        "lastModifiedUser": "arn:aws:iam::123456789012:user/Li_Juan",
        "approvalRuleTemplateId": "a29abb15-EXAMPLE",
        "approvalRuleTemplateDescription": "All pull requests must be approved by
 one developer on the team."
    }
}
```

승인 규칙 템플릿의 내용 업데이트

1. 터미널 또는 명령 프롬프트에서 템플릿 이름과 변경된 내용을 지정하여 update-approval-rule-template-content 명령을 실행합니다. 예를 들어 **1-approver-rule**이라는 승인 규칙 템플릿의 내용을 변경하여 **CodeCommitReview** 역할을 수임하는 사용자에 대한 승인 풀을 재정의하려면 다음과 같이 합니다.

```
aws codecommit update-approval-rule-template-content --approval-rule-template-
name 1-approver-rule --new-rule-content "{\"Version\": \"2018-11-08\",
\"DestinationReferences\": [\"refs/heads/main\"],\"Statements\": [{\"Type
\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers\":
 [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}"
```

2. 이 명령이 제대로 실행되면 다음과 비슷한 출력이 반환됩니다.

```
{
    "approvalRuleTemplate": {
        "creationDate": 1571352720.773,
        "approvalRuleTemplateDescription": "Requires 1 approval for all pull
 requests from the CodeCommitReview pool",
        "lastModifiedDate": 1571358728.41,
        "approvalRuleTemplateId": "41de97b7-EXAMPLE",
        "approvalRuleTemplateContent": "{\"Version\": \"2018-11-08\",\"Statements
\": [{\"Type\": \"Approvers\",\"NumberOfApprovalsNeeded\": 1,\"ApprovalPoolMembers
\": [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
        "approvalRuleTemplateName": "1-approver-rule-for-all-pull-requests",
        "ruleContentSha256": "2f6c21a5EXAMPLE",
        "lastModifiedUser": "arn:aws:iam::123456789012:user/Li_Juan"
    }
}
```

승인 규칙 템플릿의 이름 업데이트

1. 터미널 또는 명령 프롬프트에서 현재 이름과 변경할 이름을 지정하여 update-approval-rule-template-name 명령을 실행합니다. 예를 들어 승인 규칙 템플릿의 이름을 **1-approver-rule**에서 **1-approver-rule-for-all-pull-requests**로 변경하려면 다음과 같이 합니다.

```
aws codecommit update-approval-rule-template-name --old-approval-rule-template-name
 "1-approver-rule" --new-approval-rule-template-name "1-approver-rule-for-all-pull-
requests"
```

2. 이 명령이 제대로 실행되면 다음과 비슷한 출력이 반환됩니다.

```
{
    "approvalRuleTemplate": {
        "approvalRuleTemplateName": "1-approver-rule-for-all-pull-requests",
        "lastModifiedDate": 1571358241.619,
        "approvalRuleTemplateId": "41de97b7-EXAMPLE",
        "approvalRuleTemplateContent": "{\"Version\": \"2018-11-08\",\"Statements
\": [{\"Type\": \"Approvers\",\"NumberOfApprovalsNeeded\": 1,\"ApprovalPoolMembers
\": [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
        "creationDate": 1571352720.773,
        "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
        "approvalRuleTemplateDescription": "All pull requests must be approved by
 one developer on the team.",
        "ruleContentSha256": "2f6c21a5cEXAMPLE"
    }
}
```

승인 규칙 템플릿의 설명 업데이트

1. 터미널 또는 명령줄에서 승인 규칙 템플릿의 이름과 새로운 설명을 지정하여 update-approval-rule-template-description 명령을 실행합니다.

```
aws codecommit update-approval-rule-template-description --approval-rule-template-
name "1-approver-rule-for-all-pull-requests" --approval-rule-template-description
 "Requires 1 approval for all pull requests from the CodeCommitReview pool"
```

2. 이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
{
    "approvalRuleTemplate": {
        "creationDate": 1571352720.773,
        "approvalRuleTemplateDescription": "Requires 1 approval for all pull
 requests from the CodeCommitReview pool",
        "lastModifiedDate": 1571358728.41,
        "approvalRuleTemplateId": "41de97b7-EXAMPLE",
        "approvalRuleTemplateContent": "{\"Version\": \"2018-11-08\",\"Statements
\": [{\"Type\": \"Approvers\",\"NumberOfApprovalsNeeded\": 1,\"ApprovalPoolMembers
\": [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
        "approvalRuleTemplateName": "1-approver-rule-for-all-pull-requests",
        "ruleContentSha256": "2f6c21a5EXAMPLE",
```

```
        "lastModifiedUser": "arn:aws:iam::123456789012:user/Li_Juan"
    }
}
```

템플릿과 연결된 모든 리포지토리 나열

1. 명령줄 또는 터미널에서 템플릿의 이름을 지정하여 list-repositories-for-approval-rule-template 명령을 실행합니다.

```
aws codecommit list-repositories-for-approval-rule-template --approval-rule-
template-name 2-approver-rule-for-main
```

2. 이 명령이 제대로 실행되면 다음과 비슷한 출력이 반환됩니다.

```
{
    "repositoryNames": [
        "MyDemoRepo",
        "MyClonedRepo"
    ]
}
```

리포지토리와 연결된 모든 템플릿 나열

1. 명령줄 또는 터미널에서 리포지토리의 이름을 지정하여 list-associated-approval-rule-templates-for-repository 명령을 실행합니다.

```
aws codecommit list-associated-approval-rule-templates-for-repository --repository-
name MyDemoRepo
```

2. 이 명령이 제대로 실행되면 다음과 비슷한 출력이 반환됩니다.

```
{
    "approvalRuleTemplateNames": [
        "2-approver-rule-for-main",
        "1-approver-rule-for-all-pull-requests"
    ]
}
```

승인 규칙 템플릿 연결 해제

승인 규칙 템플릿으로 생성된 승인 규칙이 리포지토리의 팀 워크플로에 더 이상 적합하지 않은 경우 해당 리포지토리에서 템플릿을 연결 해제할 수 있습니다. 템플릿을 연결 해제해도 해당 템플릿이 리포지토리와 연결된 동안 작성된 승인 규칙은 제거되지 않습니다.

승인 규칙 템플릿에 대한 관리형 정책 및 권한에 대해 자세히 알아보려면 승인 규칙 템플릿에 대한 작업 권한 섹션과 AWS CodeCommit에 대한 관리형 정책 섹션을 참조하세요.

승인 규칙 템플릿 연결 해제 (콘솔)

콘솔을 사용하여 리포지토리와 승인 규칙 템플릿 간의 연결을 제거할 수 있습니다.

리포지토리에서 승인 규칙 템플릿을 연결 해제하려면

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 승인 규칙 템플릿을 선택합니다. 리포지토리에서 연결 해제할 템플릿을 선택한 다음 편집을 선택합니다.
3. 연결된 리포지토리에서 연결 해제할 리포지토리 옆에 있는 X를 선택합니다. 리포지토리 이름이 더 이상 표시되지 않습니다.
4. 저장(Save)을 선택합니다. 승인 규칙은 해당 리포지토리에서 생성된 풀 요청에는 적용되지 않습니다. 연결이 유효한 동안 생성된 풀 요청에는 규칙이 계속 적용됩니다.

승인 규칙 템플릿 연결 해제 (AWS CLI)

를 사용하여 하나 이상의 리포지토리를 승인 규칙 템플릿에서 연결 해제 AWS CLI 할 수 있습니다.

리포지토리에서 승인 규칙 템플릿 연결 해제

1. 터미널 또는 명령줄에서 다음을 지정하여 disassociate-approval-rule-template-from-repository 명령을 실행합니다.

   - 승인 규칙 템플릿의 이름.
   - 리포지토리의 이름.

   예를 들어 **MyDemoRepo**라는 리포지토리에서 **1-approver-rule-for-all-pull-requests**라는 승인 규칙 템플릿을 연결 해제하려면 다음과 같이 하십시오.

```
aws codecommit disassociate-approval-rule-template-from-repository --repository-
name MyDemoRepo --approval-rule-template-name 1-approver-rule-for-all-pull-requests
```

2. 성공한 경우 이 명령은 아무 것도 반환하지 않습니다.

여러 리포지토리에서 승인 규칙 템플릿을 연결 해제하려면

1. 터미널 또는 명령줄에서 다음을 지정하여 batch-disassociate-approval-rule-template-from-repositories 명령을 실행합니다.

   - 승인 규칙 템플릿의 이름.
   - 리포지토리의 이름.

   예를 들어 **MyDemoRepo**라는 리포지토리와 **MyOtherDemoRepo**라는 리포지토리에서 **1-approver-rule-for-all-pull-requests**라는 승인 규칙 템플릿을 연결 해제하려면 다음과 같이 합니다.

```
aws codecommit batch-disassociate-approval-rule-template-from-repositories --
repository-names "MyDemoRepo", "MyOtherDemoRepo" --approval-rule-template-name 1-
approver-rule-for-all-pull-requests
```

2. 이 명령이 제대로 실행되면 다음과 비슷한 출력이 반환됩니다.

```
{
    "disassociatedRepositoryNames": [
        "MyDemoRepo",
        "MyOtherDemoRepo"
    ],
    "errors": []
}
```

승인 규칙 템플릿 삭제

리포지토리에서 사용하지 않는 승인 규칙 템플릿을 삭제할 수 있습니다. 사용하지 않는 승인 규칙 템플릿을 삭제하면 템플릿을 체계적으로 유지하고 워크플로에 적합한 템플릿을 쉽게 찾을 수 있습니다.

승인 규칙 템플릿에 대한 관리형 정책 및 권한에 대해 자세히 알아보려면 승인 규칙 템플릿에 대한 작업 권한 섹션과 AWS CodeCommit에 대한 관리형 정책 섹션을 참조하세요.

주제

- 승인 규칙 템플릿 삭제 (콘솔)
- 승인 규칙 템플릿 삭제 (AWS CLI)

승인 규칙 템플릿 삭제 (콘솔)

승인 규칙 템플릿이 개발 작업과 더 이상 관련이 없는 경우 삭제할 수 있습니다. 콘솔을 사용하여 승인 규칙 템플릿을 삭제하면 삭제 프로세스 중에 해당 템플릿이 리포지토리에서 연결 해제됩니다.

승인 규칙 템플릿 삭제

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 승인 규칙 템플릿을 선택합니다. 삭제할 템플릿을 선택한 다음 삭제를 선택합니다.

승인 규칙 템플릿 삭제 (AWS CLI)

모든 AWS CLI 리포지토리에서 연결 해제된 경우를 사용하여 승인 규칙을 삭제할 수 있습니다. 자세한 내용은 승인 규칙 템플릿 연결 해제 (AWS CLI) 단원을 참조하십시오.

승인 규칙 템플릿 삭제

1. 터미널 또는 명령줄에서 삭제할 승인 규칙 템플릿의 이름을 지정하여 delete-approval-rule-template 명령을 실행합니다.

```
aws codecommit delete-approval-rule-template --approval-rule-template-name 1-approver-for-all-pull-requests
```

2. 이 명령이 성공하면 다음과 비슷한 출력이 반환됩니다. 승인 규칙 템플릿이 이미 삭제된 경우 이 명령은 아무 것도 반환하지 않습니다.

```
{
    "approvalRuleTemplateId": "41de97b7-EXAMPLE"
}
```

AWS CodeCommit 리포지토리에서 커밋 작업

커밋은 리포지토리의 콘텐츠 및 변경 사항에 대한 스냅샷입니다. 사용자가 변경 사항을 커밋하고 푸시할 때마다 해당 정보는 저장 및 보관됩니다. 변경 사항을 커밋한 사용자, 커밋의 날짜 및 시간, 커밋 과정에서 이루어진 변경 사항 등의 정보 역시 마찬가지입니다. 또한 커밋에 태그를 추가하여 특정 커밋을 쉽게 식별할 수 있습니다. CodeCommit에서는 다음을 수행할 수 있습니다.

- 커밋을 검토할 수 있습니다.
- 커밋 이력을 그래프로 볼 수 있습니다.
- 커밋을 상위 또는 다른 지정자와 비교할 수 있습니다.
- 커밋에 주석을 추가하고 다른 사람이 작성한 주석에 댓글을 달 수 있습니다.

CodeCommit 리포지토리로 커밋을 푸시하기 전에 먼저 로컬 컴퓨터를 리포지토리에 연결하도록 설정해야 합니다. 가장 간단한 방법에 대해 알아보려면 Git 보안 인증 정보를 사용하는 HTTPS 사용자의 경우 섹션을 참조하세요.

CodeCommit에서 리포지토리의 다른 측면으로 작업하는 방법에 대해 자세히 알아보려면 리포지토리 작업, 파일 작업하기, 풀 요청 작업, 브랜치 작업, 사용자 기본 설정으로 작업 섹션을 참조하세요.

주제

- 에서 커밋 생성 AWS CodeCommit

- 에서 커밋 세부 정보 보기 AWS CodeCommit
- 에서 커밋 비교 AWS CodeCommit
- 의 커밋에 대한 설명 AWS CodeCommit
- 에서 Git 태그 생성 AWS CodeCommit
- 에서 Git 태그 세부 정보 보기 AWS CodeCommit
- 에서 Git 태그 삭제 AWS CodeCommit

에서 커밋 생성 AWS CodeCommit

새 리포지토리에 대한 첫 번째 커밋을 생성할 때 AWS CLI 및 put-file 명령을 사용합니다. 그러면 첫 번째 커밋이 만들어지며 이에 따라 새 리포지토리의 기본 브랜치를 생성 및 지정할 수 있습니다. Git 또는 AWS CLI 를 사용하여 CodeCommit 리포지토리에서 커밋을 생성할 수 있습니다. 로컬 리포지토리가 CodeCommit 리포지토리에 연결되어 있는 경우에는 Git을 사용하여 로컬 리포지토리의 커밋을 CodeCommit 리포지토리로 푸시합니다. CodeCommit 콘솔에서 직접 커밋을 생성하려면 AWS CodeCommit 리포지토리에 파일 생성 또는 추가 및 AWS CodeCommit 리포지토리에서 파일의 콘텐츠 편집 단원을 참조하세요.

> **Note**
>
> 가장 좋은 방법은 지원되는 최신 버전의 AWS CLI, Git 및 기타 소프트웨어를 사용하는 것입니다. 를 사용하는 경우 `create-commit` 명령이 포함된 버전을 사용하고 있는지 확인하기 위해 최신 버전이 설치되어 있는지 AWS CLI확인합니다.

주제

- 를 사용하여 리포지토리에 대한 첫 번째 커밋 생성 AWS CLI
- Git 클라이언트를 사용하여 커밋 생성
- 를 사용하여 커밋 생성 AWS CLI

를 사용하여 리포지토리에 대한 첫 번째 커밋 생성 AWS CLI

AWS CLI 및 `put-file` 명령을 사용하여 리포지토리에 대한 첫 번째 커밋을 생성할 수 있습니다. put-file를 사용하면 빈 리포지토리에 파일을 추가하는 첫 번째 커밋이 생성되며, 이에 따라 사용자가 지정한 이름의 브랜치가 생성됩니다. 그러면 새 브랜치가 리포지토리의 기본 브랜치로 지정됩니다.

> **Note**
>
> CodeCommit에서 AWS CLI 명령을 사용하려면를 설치합니다 AWS CLI. 자세한 내용은 명령줄 참조 단원을 참조하십시오.

를 사용하여 리포지토리에 대한 첫 번째 커밋을 생성하려면 AWS CLI

1. 로컬 컴퓨터에서 CodeCommit 리포지토리에 첫 번째 파일로 추가할 파일을 만듭니다. 일반적인 방법은 다른 리포지토리 사용자에게 이 리포지토리의 용도를 설명하는 `README.md` 마크다운 파일을 만드는 것입니다. `README.md` 파일을 포함하면, CodeCommit 콘솔의 리포지토리에 대한 코드 페이지 하단에 파일 내용이 자동으로 표시됩니다.
2. 터미널 또는 명령줄에서 다음을 지정하여 put-file 명령을 실행합니다.

   - 첫 파일을 추가할 리포지토리의 이름.
   - 기본 브랜치로 생성하려는 브랜치의 이름.
   - 파일의 로컬 위치. 이 위치에 사용되는 구문은 로컬 운영 체제에 따라 달라집니다.
   - 업데이트된 파일이 리포지토리에 저장된 경로를 포함하여 추가할 파일의 이름.
   - 이 파일에 연결할 사용자 이름과 이메일.
   - 이 파일을 추가한 이유를 설명하는 커밋 메시지.

   사용자 이름, 이메일 주소 및 커밋 메시지는 선택 사항이지만, 다른 사용자에게 변경을 수행한 사용자와 변경 이유를 알려 줄 수 있습니다. 사용자 이름을 제공하지 않는 경우, CodeCommit은 IAM 사용자 이름이나 콘솔 로그인 파생물을 작성자 이름으로 사용하도록 기본 설정되어 있습니다.

   예를 들어, 예제 base 6 인코딩 파일 콘텐츠가 포함된 *`README.md`* 파일을 *`MyDemoRepo`*라는 리포지토리에 *`development`*라는 브랜치에 추가하려면

   ```
   aws codecommit put-file --repository-name MyDemoRepo --branch-name development --
   file-path README.md --file-content "EXAMPLE0123456789example0123456789EXAMPLE1"
    --name "Mary Major" --email "mary_major@example.com" --commit-message "I added a
    quick readme for our new team repository."
   ```

   이 명령이 제대로 실행되면 다음과 비슷한 출력이 반환됩니다.

   ```
   {
   ```

```
    "commitId": "724caa36EXAMPLE",
    "blobId": "a8a94062EXAMPLE",
    "treeId": "08b2fc73EXAMPLE"
}
```

Git 클라이언트를 사용하여 커밋 생성

로컬 컴퓨터에 설치된 Git 클라이언트를 사용하여 커밋을 생성한 후, 그러한 커밋을 CodeCommit 리포지토리에 푸시할 수 있습니다.

1. 설정 을 포함한 사전 필수 단계를 완료합니다.

   > **Important**
   > 설정을 완료하지 않았으면 Git를 사용하여 리포지토리에 연결하거나 커밋할 수 없습니다.

2. 올바른 브랜치에서 커밋을 생성해야 합니다. 사용 가능한 브랜치 목록을 조회하여 현재 어떤 브랜치를 사용하도록 설정되어 있는지 확인하려면 git branch 명령을 실행합니다. 모든 브랜치가 표시됩니다. 현재 설정된 브랜치 옆에 별표(*)가 표시됩니다. 다른 브랜치로 전환하려면 git checkout ***branch-name*** 명령을 실행합니다. 커밋을 처음하는 경우에는 git config 명령을 실행하여 해당 브랜치에 사용할 이름으로 초기 브랜치를 생성하도록 Git 클라이언트를 구성합니다. 예를 들어, 기본 브랜치에 *development*라는 이름을 붙이려면 다음과 같이 합니다.

   ```
   git config --local init.defaultBranch development
   ```

   > **Tip**
   > 이 명령은 CLI 버전 2.28 이상에서만 사용할 수 있습니다.
   > 이 명령을 실행하여, 새로 생성한 모든 리포지토리의 기본 브랜치 이름을 **development**으로 설정할 수도 있습니다.
   >
   > ```
   > git config --global init.defaultBranch development
   > ```

3. 브랜치를 변경합니다(파일 추가, 수정 또는 삭제 등).

   예를 들어 로컬 리포지토리에서 다음 텍스트를 사용하여 `bird.txt`라는 이름의 파일을 생성합니다.

```
bird.txt
--------
Birds (class Aves or clade Avialae) are feathered, winged, two-legged, warm-
blooded, egg-laying vertebrates.
```

4. git status 명령을 실행하여 `bird.txt`가 대기 중인 어떤 커밋에도 포함되지 않았음을 나타내야 합니다.

```
...
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        bird.txt
```

5. git add bird.txt 명령을 실행하여 대기 중인 커밋에 새로운 파일을 포함합니다.
6. git status 명령을 다시 실행한 경우 다음과 비슷한 출력이 표시되어야 합니다. 이 메시지는 `bird.txt`가 이제 대기 중인 커밋의 일부가 되었거나 커밋에 스테이징되었음을 나타냅니다.

```
...
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   bird.txt
```

7. 커밋을 완료하려면 `-m` 옵션으로 git commit 명령을 실행합니다(예: git commit -m "***Adding bird.txt to the repository.***"). `-m` 옵션이 커밋 메시지를 생성합니다.
8. git status 명령을 다시 실행한 경우 다음과 비슷한 출력이 표시되어야 합니다. 이 메시지는 해당 커밋이 로컬 리포지토리에서 CodeCommit 리포지토리로 푸시될 준비가 되었음을 나타냅니다.

```
...
nothing to commit, working directory clean
```

9. 완료된 커밋을 로컬 리포지토리에서 CodeCommit 리포지토리로 푸시하기 전에 git diff --stat ***remote-name/branch-name***을 실행하여 어떤 커밋을 푸시하는지 확인할 수 있습니다. 이때 *remote-name*은 로컬 리포지토리가 CodeCommit 리포지토리에 사용하는 별명이고, *branch-name*은 비교 대상인 브랜치의 이름입니다.

> **Tip**
>
> 별명을 가져오려면 git remote을 실행합니다. 브랜치 이름 목록을 가져오려면 git branch 명령을 실행합니다. 현재 설정된 브랜치 옆에 별표(*)가 표시됩니다. 또한 git status 명령을 실행해도 현재 브랜치 이름을 가져올 수 있습니다.

> **Note**
>
> 리포지토리를 복제한 경우 로컬 리포지토리의 관점에서 *remote-name*은 CodeCommit 리포지토리의 이름이 아닙니다. 리포지토리를 복제할 때 *remote-name*은 자동으로 `origin`으로 설정됩니다.

예를 들어 git diff --stat origin/main 명령을 실행하면 다음과 비슷한 메시지가 출력됩니다.

```
bird.txt | 1 +
1 file changed, 1 insertion(+)
```

출력된 내용은 로컬 리포지토리를 CodeCommit 리포지토리에 이미 연결했다고 가정합니다. (지침은 리포지토리에 연결 단원을 참조하세요.)

10. 해당 커밋을 로컬 리포지토리에서 CodeCommit 리포지토리로 푸시할 준비가 되었으면 git push ***remote-name branch-name*** 명령을 실행합니다. 여기서 *remote-name*은 로컬 리포지토리가 CodeCommit 리포지토리에 사용하는 별명이고, *branch-name*은 CodeCommit 리포지토리에 푸시할 브랜치의 이름입니다.

    예를 들어 git push origin main 명령을 실행하면 다음과 비슷한 메시지가 출력됩니다.

    HTTPS의 경우에는 다음과 같이 합니다.

```
Counting objects: 7, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 516 bytes | 0 bytes/s, done.
Total 5 (delta 2), reused 0 (delta 0)
remote:
To https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
```

```
    b9e7aa6..3dbf4dd main -> main
```

SSH의 경우에는 다음과 같이 합니다.

```
Counting objects: 7, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 516 bytes | 0 bytes/s, done.
Total 5 (delta 2), reused 0 (delta 0)
remote:
To ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
    b9e7aa6..3dbf4dd main -> main
```

> **Tip**
>
> `-u` 옵션을 git push에 추가하면(예: git push -u origin main) 업스트림 추적 정보가 설정되었기 때문에 향후에 git push 명령만 실행하면 됩니다. 업스트림 추적 정보를 보려면 git remote show ***`remote-name`***을 실행합니다(예: git remote show origin).

다른 옵션들에 대해서는 Git 설명서를 참조하세요.

를 사용하여 커밋 생성 AWS CLI

AWS CLI 및 `create-commit` 명령을 사용하여 지정된 브랜치의 끝에 리포지토리에 대한 커밋을 생성할 수 있습니다. 참조되지 않은 병합 커밋을 생성하여 두 커밋 지정자의 병합 결과를 나타낼 수도 있습니다. 자세한 내용은 참조되지 않은 커밋 생성을 참조하세요.

> **Note**
>
> CodeCommit에서 AWS CLI 명령을 사용하려면를 설치합니다 AWS CLI. 자세한 내용은 명령줄 참조 단원을 참조하십시오.

커밋 생성

1. 로컬 컴퓨터에서 CodeCommit 리포지토리에 커밋할 변경을 수행합니다.
2. 터미널 또는 명령줄에서 다음을 지정하여 create-commit 명령을 실행합니다.

- 변경 사항을 커밋할 리포지토리
- 변경 사항을 커밋할 브랜치
- 해당 브랜치에서 수행된 가장 최근 커밋의 축약되지 않은 커밋 ID(팁 또는 헤드 커밋 또는 상위 커밋 ID)
- 수행한 변경으로 인해 그러한 폴더의 내용이 삭제되면 빈 폴더를 유지할지 여부. 기본적으로 이 값은 false입니다.
- 추가, 변경 또는 삭제하려는 파일 정보
- 이러한 변경 사항에 연결할 사용자 이름과 이메일
- 이러한 변경을 수행한 이유를 설명하는 커밋 메시지

사용자 이름, 이메일 주소 및 커밋 메시지는 선택 사항이지만, 다른 사용자에게 변경을 수행한 사용자와 변경 이유를 알려주세요. 사용자 이름을 제공하지 않는 경우, CodeCommit은 IAM 사용자 이름이나 콘솔 로그인 파생물을 작성자 이름으로 사용하도록 기본 설정되어 있습니다.

예를 들어, `README.md` 파일을 *`MyDemorepo`*라는 리포지토리의 *`main`* 브랜치에 추가하는 리포지토리 커밋을 생성하려면 다음과 같이 합니다. 파일 내용은 Base64에 있으며 “Welcome to our team repository!”라고 쓰여 있습니다.

```
aws codecommit create-commit --repository-name MyDemoRepo --
branch-name main --parent-commit-id 4c925148EXAMPLE --put-files
 "filePath=README.md,fileContent=V2VsY29tZSB0byBvdXIgdGVhbSByZXBvc2l0b3J5IQo="
```

> **Tip**
>
> 상위 커밋 ID를 가져오려면 get-branch 명령을 실행합니다.

이 명령이 제대로 실행되면 다음과 비슷한 출력이 반환됩니다.

```
{
    "commitId": "4df8b524-EXAMPLE",
    "treeId": "55b57003-EXAMPLE",
    "filesAdded": [
        {
            "blobId": "5e1c309dEXAMPLE",
            "absolutePath": "meeting.md",
```

```
            "fileMode": "NORMAL"
        }
    ],
    "filesDeleted": [],
    "filesUpdated": []
}
```

file1.py 및 *file2.txt*라는 파일을 변경하는 커밋을 생성하려면, 한 파일의 이름을 *picture.png*에서 *image1.png*로 바꾸고, 그 파일을 *pictures*라는 디렉터리에서 *images ##* 디렉터리로 옮긴 다음, *MyDemoRepo*라는 리포지토리 내에서 가장 최근 커밋의 ID가 *4c925148EXAMPLE*이고 이름이 *MyFeatureBranch*인 브랜치의 *ExampleSolution.py*라는 파일을 삭제합니다.

```
aws codecommit create-commit --repository-name MyDemoRepo --branch-
name MyFeatureBranch --parent-commit-id 4c925148EXAMPLE --author-name "Saanvi
 Sarkar"
 --email "saanvi_sarkar@example.com" --commit-message "I'm creating this commit to
 update a variable name in a number of files."
 --keep-empty-folders false  --put-files '{"filePath": "file1.py", "fileMode":
 "EXECUTABLE", "fileContent": "bucket_name = sys.argv[1] region = sys.argv[2]"}'
'{"filePath": "file2.txt", "fileMode": "NORMAL", "fileContent": "//Adding a comment
 to explain the variable changes in file1.py"}' '{"filePath": "images/image1.png",
"fileMode": "NORMAL", "sourceFile": {"filePath": "pictures/picture.png", "isMove":
 true}}' --delete-files filePath="ExampleSolution.py"
```

> **Note**
>
> --put-files 세그먼트의 구문은 운영 체제에 따라 다릅니다. 위의 예제는 Linux, macOS, Unix 사용자, 그리고 Bash 에뮬레이터가 포함된 Windows 사용자에게 최적화된 것입니다. 명령줄 또는 Powershell에서 Windows 사용자는 해당 시스템에 적합한 구문을 사용해야 합니다.

이 명령이 제대로 실행되면 다음과 비슷한 출력이 반환됩니다.

```
{
   "commitId": "317f8570EXAMPLE",
   "treeId": "347a3408EXAMPLE",
   "filesAdded": [
        {
```

```
        "absolutePath": "images/image1.png",
        "blobId": "d68ba6ccEXAMPLE",
        "fileMode": "NORMAL"
        }
    ],
    "filesUpdated": [
        {
        "absolutePath": "file1.py",
        "blobId": "0a4d55a8EXAMPLE",
        "fileMode": "EXECUTABLE"
        },
        {
        "absolutePath": "file2.txt",
        "blobId": "915766bbEXAMPLE",
        "fileMode": "NORMAL"
        }
    ],
    "filesDeleted": [
        {
        "absolutePath": "ExampleSolution.py",
        "blobId": "4f9cebe6aEXAMPLE",
        "fileMode": "EXECUTABLE"
        },
        {
        "absolutePath": "pictures/picture.png",
        "blobId": "fb12a539EXAMPLE",
        "fileMode": "NORMAL"
        }
    ]
}
```

에서 커밋 세부 정보 보기 AWS CodeCommit

AWS CodeCommit 콘솔을 사용하여 리포지토리의 커밋 기록을 찾아볼 수 있습니다. 이를 통해 다음 사항을 포함한 리포지토리 변경 사항을 파악할 수 있습니다.

- 변경 시점 및 변경 주체
- 특정 커밋이 브랜치에 병합된 시점

브랜치에 대한 커밋 이력을 조회하는 것도 브랜치 간 차이를 이해하는 데 도움이 될 수 있습니다. 태깅을 사용하는 경우에는 태그로 레이블이 지정된 커밋과 태그가 지정된 커밋의 부모를 신속하게 조회할 수도 있습니다. 명령줄에서 Git을 사용해 로컬 리포지토리 또는 CodeCommit 리포지토리의 커밋에 대한 세부 정보를 볼 수 있습니다.

리포지토리의 커밋 검색

AWS CodeCommit 콘솔을 사용하여 리포지토리에 대한 커밋 기록을 찾아볼 수 있습니다. 또한 리포지토리 및 그 브랜치 내 커밋을 시간 경과에 따라 표기한 그래프 형태로 볼 수 있습니다. 이를 통해 변경 시점을 포함한 리포지토리의 이력을 이해할 수 있습니다.

> **Note**
>
> git rebase 명령을 사용하여 리포지토리를 리베이스하면 리포지토리의 이력이 변경되어 커밋의 순서가 흐트러져 보일 수 있습니다. 자세한 내용은 Git Branching-Rebasing 또는 해당 Git 문서를 참조하세요.

주제

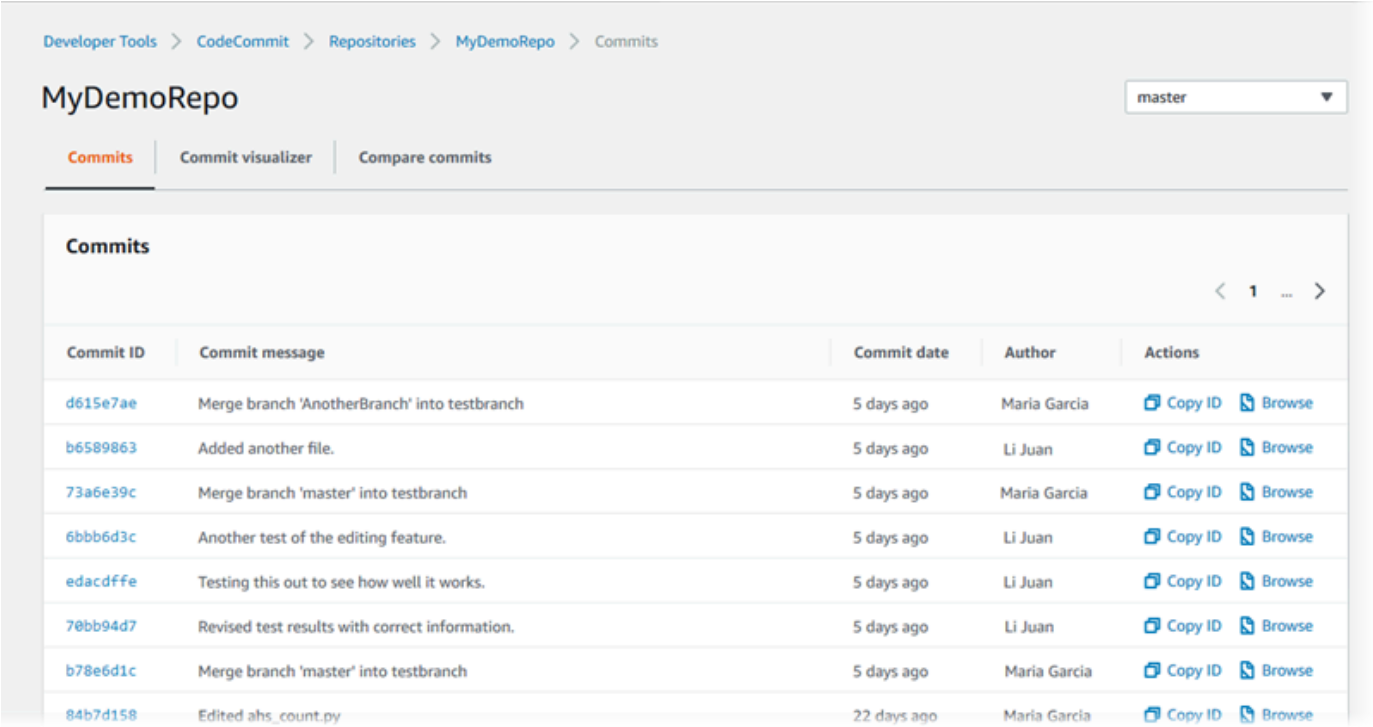
- 리포지토리의 커밋 이력 검색
- 리포지토리의 커밋 이력을 그래프 형태로 보기

리포지토리의 커밋 이력 검색

커미터에 대한 정보와 커밋 메시지를 포함해 해당 리포지토리의 특정 브랜치 또는 태그에 대한 커밋 이력을 조회할 수 있습니다. 또한 커밋에 대한 코드도 볼 수 있습니다.

커밋 이력을 찾아보려면

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서, 커밋 이력을 보고자 하는 리포지토리를 선택합니다.
3. 탐색 창에서 커밋을 선택합니다. 커밋 이력 보기에 기본 브랜치의 리포지토리에 대한 커밋 이력이 최신 커밋 날짜부터 순서대로 표시됩니다. 날짜 및 시간은 협정 세계시(UTC)로 표시됩니다. 보기 선택 버튼을 누른 후 목록에서 다른 브랜치를 선택하면 그 브랜치의 커밋 이력을 볼 수 있습니다. 리포지토리에 태그를 사용하고 있다면 보기 선택 버튼에서 태그를 선택하여 그 태그가 있는 커밋과 그 부모 커밋을 조회할 수 있습니다.



4. 커밋과 그 상위 항목 간의 차이와 변경 내용에 대한 주석을 보려면 약식 커밋 ID를 선택합니다. 자세한 내용은 한 커밋을 상위 커밋과 비교 및 커밋에 대한 주석 추가 섹션을 참조하세요. 브랜치, 태그, 커밋 ID 등 한 커밋과 기타 커밋 지정자 간의 차이를 보려면 임의의 두 커밋 지정자 비교 단원을 참조하세요.
5. 다음 중 한 개 이상을 수행할 수 있습니다.
   - 변경이 이루어진 날짜와 시간을 보려면 커밋 날짜 위에 마우스 포인터를 놓으세요.
   - 축약되지 않은 커밋 ID를 보려면 커밋 ID를 복사하여 텍스트 편집기 또는 기타 위치에 붙여 넣습니다. 복사하려면 ID 복사를 선택합니다.
   - 커밋 당시의 코드를 보려면 찾아보기를 선택합니다. 해당 커밋 시점의 리포지토리 콘텐츠가 코드 보기에 표시됩니다. 보기 선택 버튼에 브랜치 또는 태그 대신에 약식 커밋 ID가 표시됩니다.

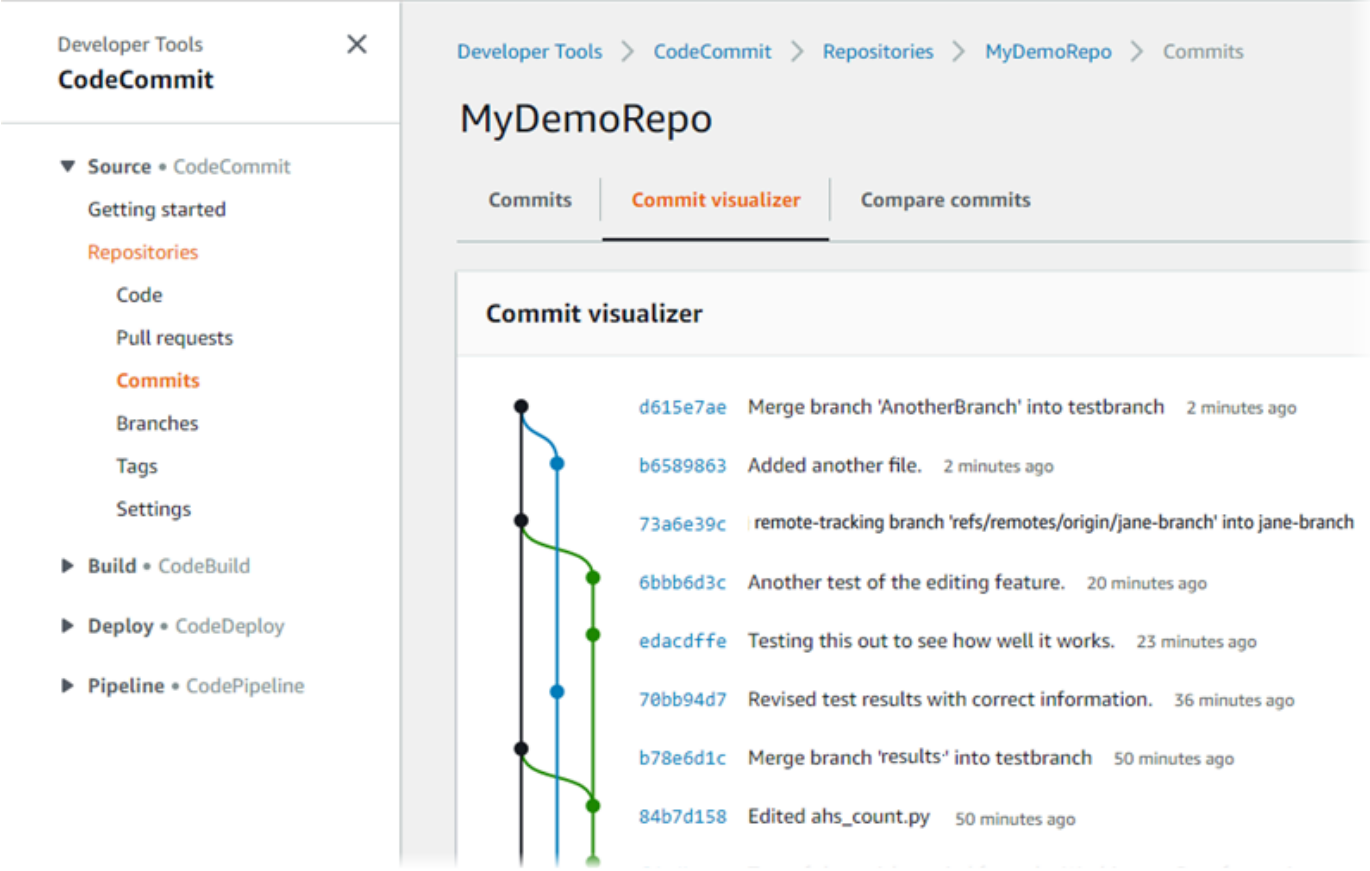
리포지토리의 커밋 이력을 그래프 형태로 보기

특정 리포지토리에 대한 커밋을 그래프 형태로 볼 수 있습니다. 커밋 시각화 도우미 보기에는 해당 리포지토리의 한 브랜치에 대한 모든 커밋이 방향성 비사이클 그래프(DAG) 형태로 표시됩니다. 이 그래프를 통해 커밋과 관련 기능이 추가 또는 병합된 시점을 알 수 있습니다. 또한 다른 변경 사항과 관련해 이루어진 변경 사항을 정확히 식별할 수 있습니다.

> **Note**
> 패스트 포워드 방법을 사용해 병합한 커밋은 커밋 그래프에 별도의 줄로 표시되지 않습니다.

커밋 그래프를 보려면

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서 커밋 그래프를 보고자 하는 리포지토리를 선택합니다.
3. 탐색 창에서 커밋을 선택한 다음 커밋 시각화 도우미 탭을 선택합니다.

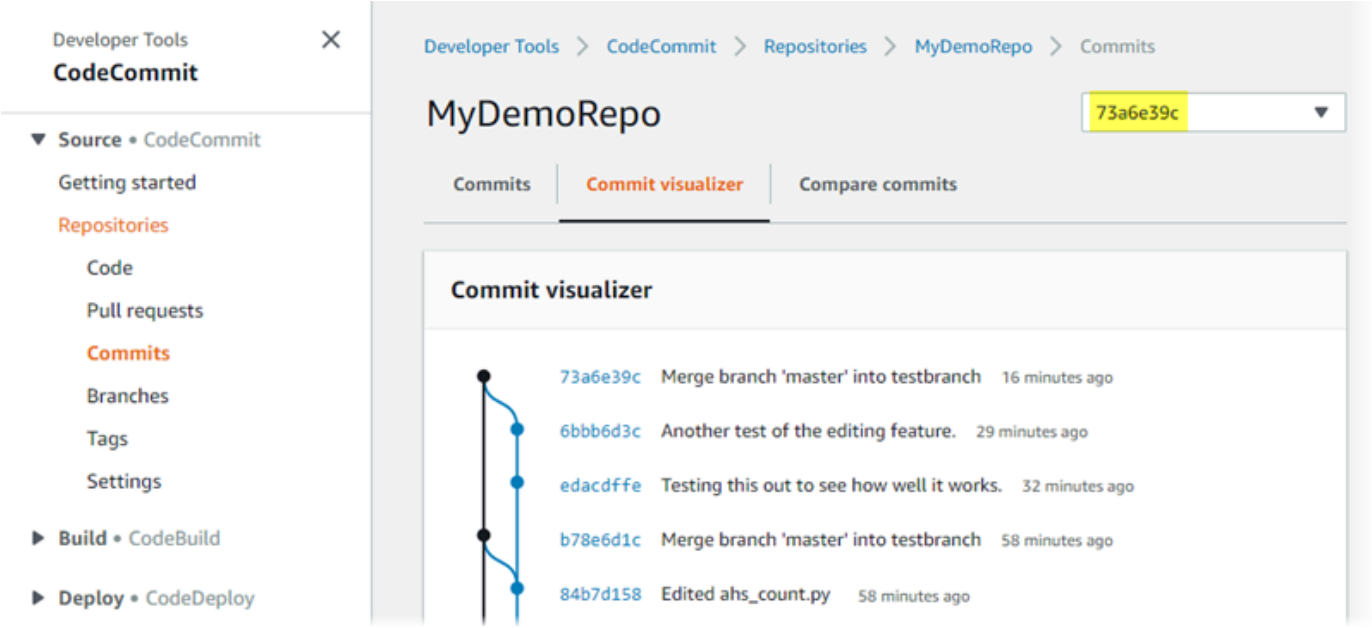


커밋 그래프에서 각 커밋 메시지의 약식 커밋 ID 및 제목이 그래프의 해당 시점 옆에 표시됩니다.

> **Note**
> 그래프는 한 페이지에 최대 35개의 브랜치를 표시할 수 있습니다. 브랜치가 35개를 넘는 경우에는 너무 복잡하여 표시되지 않습니다. 이 경우 보기를 다음과 같은 두 가지 방법으로 단순화할 수 있습니다.

- 보기 선택 버튼을 사용하여 특정 브랜치에 대한 그래프를 표시
- 축약되지 않은 커밋 ID를 검색 상자에 붙여 넣어 해당 커밋에서 그래프를 렌더링

4. 특정 커밋에서 새 그래프를 렌더링하려면 그 커밋에 해당하는 그래프에서 시점을 선택합니다. 보기 선택 버튼이 약식 커밋 ID로 변경됩니다.



커밋 세부 정보 보기 (AWS CLI)

Git를 통해 커밋에 대한 세부 정보를 조회할 수 있습니다. 를 사용하여 다음 명령을 실행 AWS CLI 하여 로컬 리포지토리 또는 CodeCommit 리포지토리의 커밋에 대한 세부 정보를 볼 수도 있습니다.

- 커밋에 대한 정보를 보려면 aws codecommit get-commit을 실행합니다.
- 여러 커밋에 대한 정보를 보려면 aws codecommit batch-get-commits을 실행합니다.
- 병합 커밋에 대한 정보를 보려면 aws codecommit get-merge-commit을 실행합니다.
- 커밋 지정자의 변경 사항에 대한 정보를 보려면(브랜치, 태그, HEAD 또는 커밋 ID와 같은 기타 정규화된 참조) aws codecommit get-differences를 실행합니다.
- 리포지토리에 있는 Git blob 객체의 base64 인코딩 내용을 보려면 aws codecommit get-blob을 실행합니다.

커밋에 대한 정보를 보려면

1. 다음을 지정하여 aws codecommit get-commit 명령을 실행합니다.

   - CodeCommit 리포지토리의 이름(`--repository-name` 옵션 사용)
   - 축약되지 않은 커밋 ID

   예를 들어 `MyDemoRepo`라는 CodeCommit 리포지토리에서 ID가 `317f8570EXAMPLE`인 커밋에 대한 정보를 보려면 다음을 수행합니다.

```
aws codecommit get-commit  --repository-name MyDemoRepo  --commit-id
 317f8570EXAMPLE
```

2. 이 명령이 제대로 실행되면 다음 사항이 출력됩니다.

   - 타임스탬프 형식의 날짜 및 협정 세계시(UTC) 오프셋을 포함한 커밋 작성자(Git에 구성됨) 정보
   - 타임스탬프 형식의 날짜 및 UTC 오프셋을 포함한 커미터(Git에 구성됨) 정보
   - 커밋이 존재하는 Git 트리의 ID
   - 상위 커밋의 커밋 ID
   - 커밋 메시지

   다음은 위 예제 명령의 출력 예입니다.

```
{
    "commit": {
        "additionalData": "",
        "committer": {
            "date": "1484167798 -0800",
            "name": "Mary Major",
            "email": "mary_major@example.com"
        },
        "author": {
            "date": "1484167798 -0800",
            "name": "Mary Major",
            "email": "mary_major@example.com"
        },
        "treeId": "347a3408EXAMPLE",
        "parents": [
            "4c925148EXAMPLE"
```

```
        ],
        "message": "Fix incorrect variable name"
    }
}
```

병합 커밋에 대한 정보를 보려면

1. 다음을 지정하여 get-merge-commit 명령을 실행합니다.

   - 병합 소스의 커밋 지정자(--source-commit-specifier 옵션 사용).
   - 병합 대상의 커밋 지정자(--destination-commit-specifier 옵션 사용).
   - 사용하려는 병합 옵션(--merge-option 옵션 사용).
   - 리포지토리의 이름(--repository-name 옵션 사용).

   예를 들어, *MyDemoRepo*라는 리포지토리에서 *### ##* 전략을 사용하여 *bugfix-bug1234* 소스 브랜치와 *main*이라는 대상 브랜치에 대한 병합 커밋에 대한 정보를 보려면 다음과 같이 합니다.

```
aws codecommit get-merge-commit --source-commit-specifier bugfix-bug1234 --
destination-commit-specifier main --merge-option THREE_WAY_MERGE --repository-
name MyDemoRepo
```

2. 이 명령이 제대로 실행되면 이 명령의 출력으로 다음과 비슷한 정보가 반환됩니다.

```
{
    "sourceCommitId": "c5709475EXAMPLE",
    "destinationCommitId": "317f8570EXAMPLE",
    "baseCommitId": "fb12a539EXAMPLE",
    "mergeCommitId": "ffc4d608eEXAMPLE"
}
```

여러 커밋에 대한 정보 보기

1. 다음을 지정하여 batch-get-commits 명령을 실행합니다.

   - CodeCommit 리포지토리의 이름(`--repository-name` 옵션 사용)
   - 정보를 보려는 모든 커밋에 대한 전체 커밋 ID의 목록

예를 들어 MyDemoRepo라는 CodeCommit 리포지토리에서 ID가 317f8570EXAMPLE 및 4c925148EXAMPLE인 커밋에 대한 정보를 보려면 다음과 같이 합니다.

```
aws codecommit batch-get-commits  --repository-name MyDemoRepo  --commit-ids
 317f8570EXAMPLE 4c925148EXAMPLE
```

2. 이 명령이 제대로 실행되면 다음 사항이 출력됩니다.

- 타임스탬프 형식의 날짜 및 협정 세계시(UTC) 오프셋을 포함한 커밋 작성자(Git에 구성됨) 정보
- 타임스탬프 형식의 날짜 및 UTC 오프셋을 포함한 커미터(Git에 구성됨) 정보
- 커밋이 존재하는 Git 트리의 ID
- 상위 커밋의 커밋 ID
- 커밋 메시지

다음은 위 예제 명령의 출력 예입니다.

```
{
    "commits": [
      {
        "additionalData": "",
        "committer": {
            "date": "1508280564 -0800",
            "name": "Mary Major",
            "email": "mary_major@example.com"
        },
        "author": {
            "date": "1508280564 -0800",
            "name": "Mary Major",
            "email": "mary_major@example.com"
        },
        "commitId": "317f8570EXAMPLE",
        "treeId": "1f330709EXAMPLE",
        "parents": [
            "6e147360EXAMPLE"
        ],
        "message": "Change variable name and add new response element"
    },
    {
        "additionalData": "",
```

```
        "committer": {
            "date": "1508280542 -0800",
            "name": "Li Juan",
            "email": "li_juan@example.com"
        },
        "author": {
            "date": "1508280542 -0800",
            "name": "Li Juan",
            "email": "li_juan@example.com"
        },
        "commitId": "4c925148EXAMPLE",
        "treeId": "1f330709EXAMPLE",
        "parents": [
            "317f8570EXAMPLE"
        ],
        "message": "Added new class"
    }
}
```

커밋 지정자에 대한 변경 사항 관련 정보를 보려면

1. 다음을 지정하여 aws codecommit get-differences 명령을 실행합니다.

   - CodeCommit 리포지토리의 이름(`--repository-name` 옵션 사용)
   - 정보를 조회하고 싶은 커밋 지정자. `--after-commit-specifier`만 있으면 됩니다. `--before-commit-specifier`를 지정하지 않으면 현재 `--after-commit-specifier`인 파일이 모두 표시됩니다.

   예를 들어, `MyDemoRepo`이라는 CodeCommit 리포지토리에서 ID가 `317f8570EXAMPLE` 및 `4c925148EXAMPLE`인 커밋들 사이의 차이에 대한 정보를 보려면 다음을 수행합니다.

   ```
   aws codecommit get-differences  --repository-name MyDemoRepo  --before-commit-
   specifier 317f8570EXAMPLE --after-commit-specifier 4c925148EXAMPLE
   ```

2. 이 명령이 제대로 실행되면 다음 사항이 출력됩니다.

   - 변경 유형(A: 추가, D: 삭제, M: 수정)을 포함한 차이점 목록
   - 파일 변경 유형의 모드
   - 해당 변경 사항이 포함된 Git BLOB 객체의 ID

다음은 위 예제 명령의 출력 예입니다.

```
{
    "differences": [
        {
            "afterBlob": {
                "path": "blob.txt",
                "blobId": "2eb4af3bEXAMPLE",
                "mode": "100644"
            },
            "changeType": "M",
            "beforeBlob": {
                "path": "blob.txt",
                "blobId": "bf7fcf28fEXAMPLE",
                "mode": "100644"
            }
        }
    ]
}
```

Git BLOB 객체에 대한 정보 보기

1. 다음을 지정하여 aws codecommit get-blob 명령을 실행합니다.

   - CodeCommit 리포지토리의 이름(`--repository-name` 옵션 사용)
   - Git BLOB의 ID(`--blob-id` 옵션)

   예를 들어 `MyDemoRepo`라는 CodeCommit 리포지토리에서 ID가 `2eb4af3bEXAMPLE`인 Git BLOB에 대한 정보를 보려면 다음을 수행합니다.

   ```
   aws codecommit get-blob  --repository-name MyDemoRepo  --blob-id 2eb4af3bEXAMPLE
   ```

2. 이 명령이 제대로 실행되면 다음 사항이 출력됩니다.

   - 대개 파일인 BLOB의 base64 인코딩 콘텐츠

   예를 들어 이전 명령의 실행 결과는 다음과 비슷하게 출력될 수 있습니다.

```
{
    "content": "QSBCaW5hcnkgTGFyToEXAMPLE="
}
```

커밋 세부 정보 보기 (Git)

이 단계를 수행하려면 먼저 로컬 리포지토리를 CodeCommit 리포지토리에 연결하고 변경 사항을 커밋한 상태이어야 합니다. 지침은 리포지토리에 연결 섹션을 참조하세요.

리포지토리에 대한 가장 최근 커밋의 변경 사항을 표시하려면 git show 명령을 실행합니다.

```
git show
```

이 명령을 실행하면 다음과 비슷하게 출력됩니다.

```
commit 4f8c6f9d
Author: Mary Major <mary.major@example.com>
Date:   Mon May 23 15:56:48 2016 -0700

    Added bumblebee.txt

diff --git a/bumblebee.txt b/bumblebee.txt
new file mode 100644
index 0000000..443b974
--- /dev/null
+++ b/bumblebee.txt
@@ -0,0 +1 @@
+A bumblebee, also written bumble bee, is a member of the bee genus Bombus, in the
 family Apidae.
\ No newline at end of file
```

> **Note**
>
> 이 예시부터는 커밋 ID가 축약되어 있습니다. 축약되지 않은 커밋 ID는 표시되지 않습니다.

변경된 사항을 보려면 해당 커밋 ID로 git show 명령을 사용합니다.

```
git show 94ba1e60
```

```
commit 94ba1e60
Author: John Doe <johndoe@example.com>
Date:   Mon May 23 15:39:14 2016 -0700

    Added horse.txt

diff --git a/horse.txt b/horse.txt
new file mode 100644
index 0000000..080f68f
--- /dev/null
+++ b/horse.txt
@@ -0,0 +1 @@
+The horse (Equus ferus caballus) is one of two extant subspecies of Equus ferus.
```

두 커밋 사이의 차이점을 보려면 다음과 같이 git diff 명령을 실행하고 두 커밋 ID를 포함시킵니다.

```
git diff ce22850d 4f8c6f9d
```

이 예시에서 두 커밋 간의 차이점은 두 파일이 추가되었다는 것입니다. 이 명령을 실행하면 다음과 비슷하게 출력됩니다.

```
diff --git a/bees.txt b/bees.txt
new file mode 100644
index 0000000..cf57550
--- /dev/null
+++ b/bees.txt
@@ -0,0 +1 @@
+Bees are flying insects closely related to wasps and ants, and are known for their
 role in pollination and for producing honey and beeswax.
diff --git a/bumblebee.txt b/bumblebee.txt
new file mode 100644
index 0000000..443b974
--- /dev/null
+++ b/bumblebee.txt
@@ -0,0 +1 @@
+A bumblebee, also written bumble bee, is a member of the bee genus Bombus, in the
 family Apidae.
\ No newline at end of file
```

Git을 사용해 로컬 리포지토리의 커밋에 대한 세부 정보를 보려면 다음과 같이 git log 명령을 실행합니다.

```
git log
```

이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
commit 94ba1e60
Author: John Doe <johndoe@example.com>
Date:   Mon May 23 15:39:14 2016 -0700

    Added horse.txt

commit 4c925148
Author: Jane Doe <janedoe@example.com>
Date:   Mon May 22 14:54:55 2014 -0700

    Added cat.txt and dog.txt
```

커밋 ID와 메시지만 표시하려면 다음과 같이 git log --pretty=oneline 명령을 실행합니다.

```
git log --pretty=oneline
```

이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
94ba1e60 Added horse.txt
4c925148 Added cat.txt and dog.txt
```

다른 옵션들에 대해서는 Git 설명서를 참조하세요.

에서 커밋 비교 AWS CodeCommit

CodeCommit 콘솔을 사용하여 CodeCommit 리포지토리의 커밋 지정자 간의 차이점을 볼 수 있습니다. 한 커밋과 상위 커밋 간의 차이를 빠르게 확인할 수 있습니다. 커밋 ID를 포함하여 두 참조를 비교할 수도 있습니다.

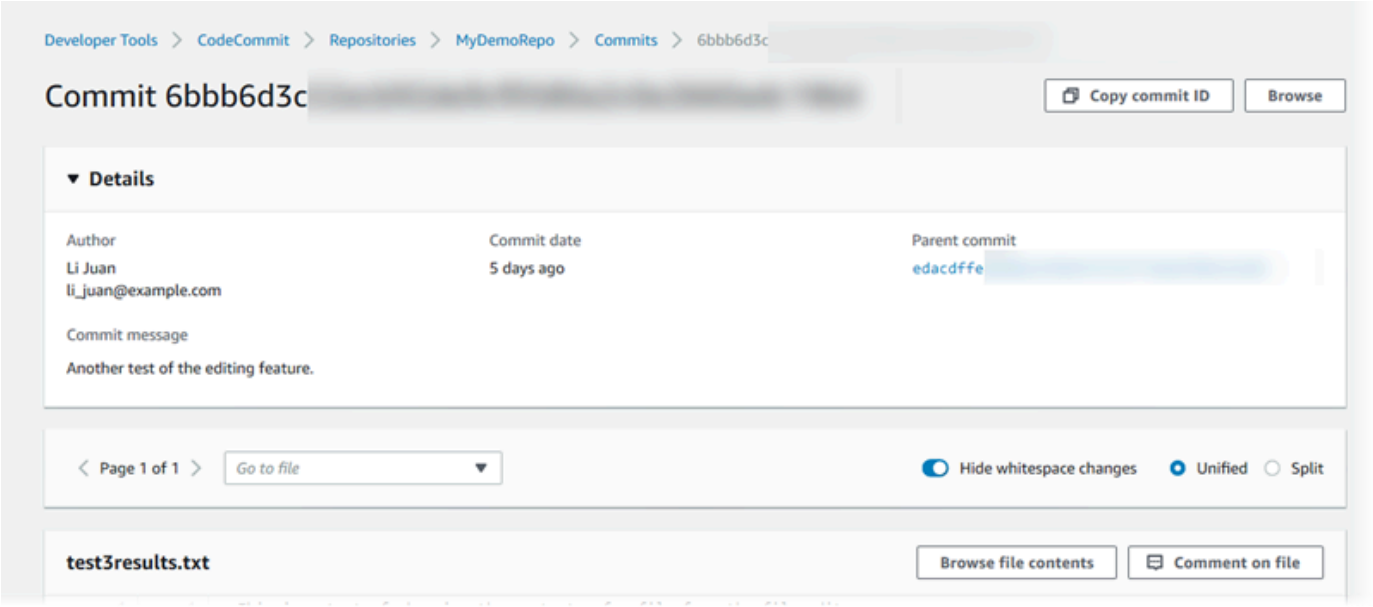
주제

- 한 커밋을 상위 커밋과 비교
- 임의의 두 커밋 지정자 비교

한 커밋을 상위 커밋과 비교

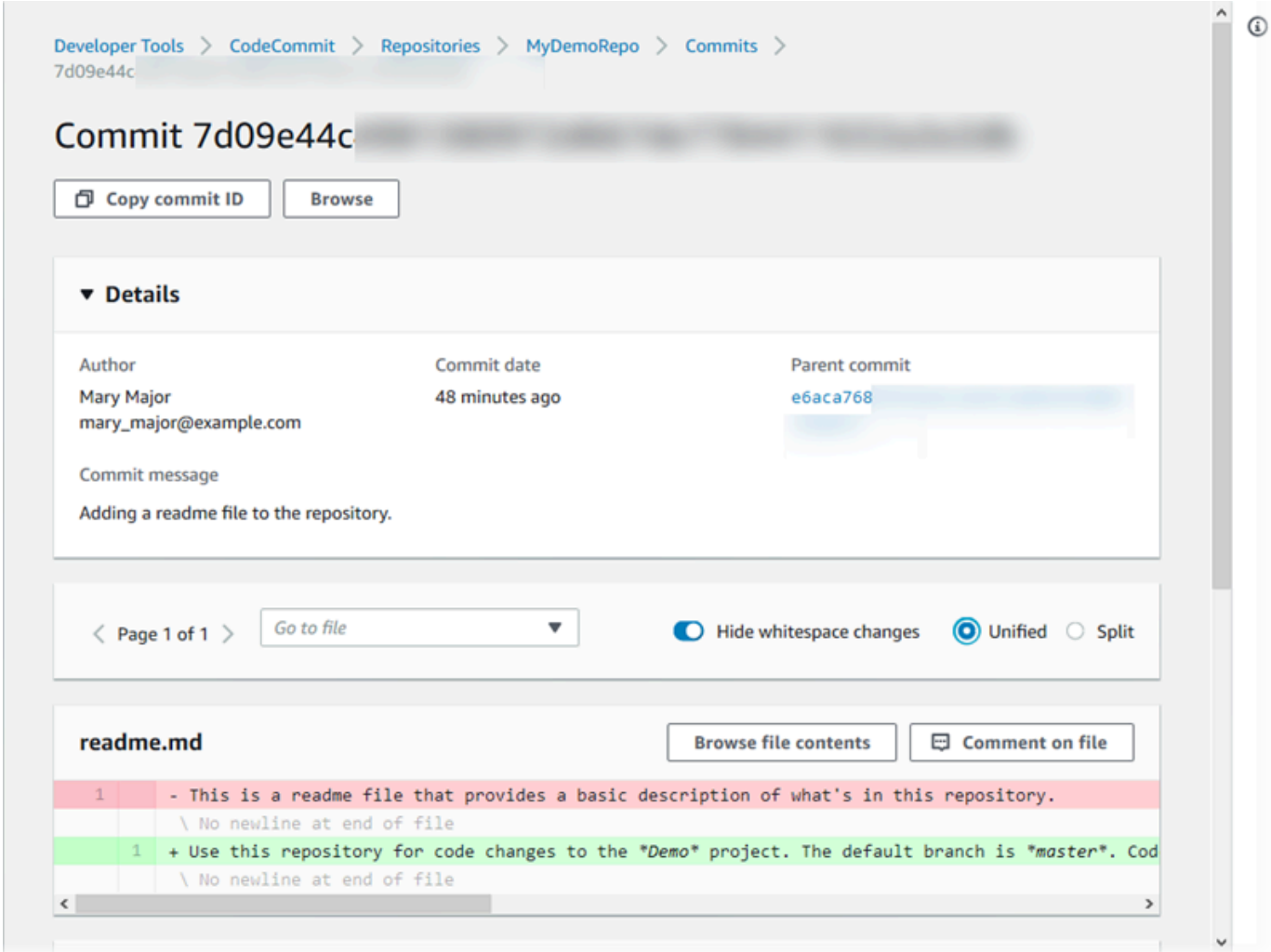
한 커밋과 상위 커밋 간의 차이를 빠르게 확인하여 커밋 메시지, 커미터, 변경 사항 등을 검토할 수 있습니다.

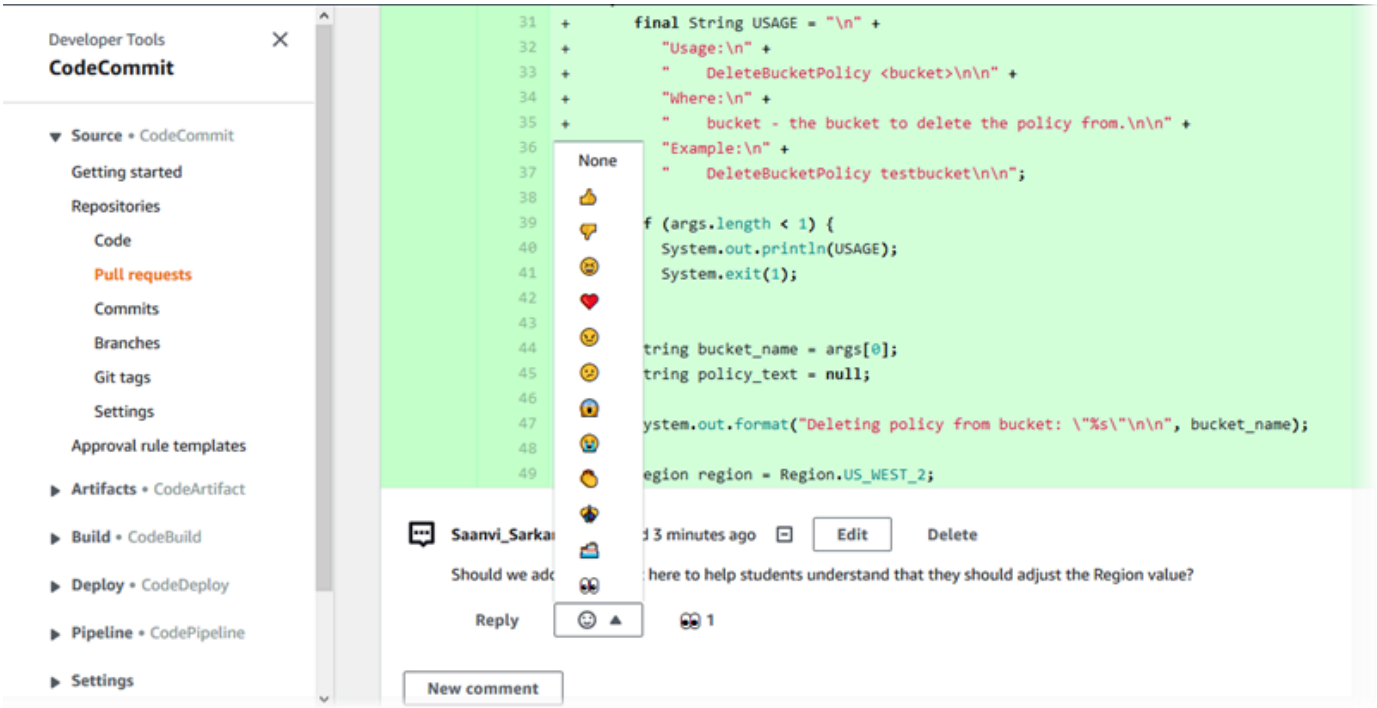
1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리 페이지에서 한 커밋과 상위 커밋 간의 차이를 보려는 리포지토리를 선택합니다.
3. 탐색 창에서 커밋을 선택합니다.
4. 목록에 있는 모든 커밋의 약식 커밋 ID를 선택합니다. 뷰가 변경되어, 해당 커밋과 상위 커밋 간의 차이점을 포함하여 이 커밋에 대한 세부 정보가 표시됩니다.



변경 내용은 나란히(분할 보기) 또는 일렬(통합 보기)로 표시할 수 있습니다. 공백 변경을 숨기거나 표시할 수도 있습니다. 주석을 추가할 수도 있습니다. 자세한 내용은 커밋에 대한 주석 추가 단원을 참조하십시오.

> **Note**
>
> 코드 및 다른 콘솔 설정 보기에 대한 기본 설정은 변경이 생길 때마다 브라우저 쿠키로 저장됩니다. 자세한 내용은 사용자 기본 설정으로 작업 단원을 참조하십시오.


Developer Tools > CodeCommit > Repositories > MyDemoRepo > Commits >
7d09e44c
Commit 7d09e44c
Copy commit ID
Browse
Details
Author
Mary Major
mary_major@example.com
Commit date
48 minutes ago
Parent commit
e6aca768
Commit message
Adding a readme file to the repository.
Page 1 of 1
Go to file
Hide whitespace changes
Unified
Split
readme.md
Browse file contents
Comment on file
1 - This is a readme file that provides a basic description of what's in this repository.
\ No newline at end of file
1 + Use this repository for code changes to the *Demo* project. The default branch is *master*. Cod
\ No newline at end of file



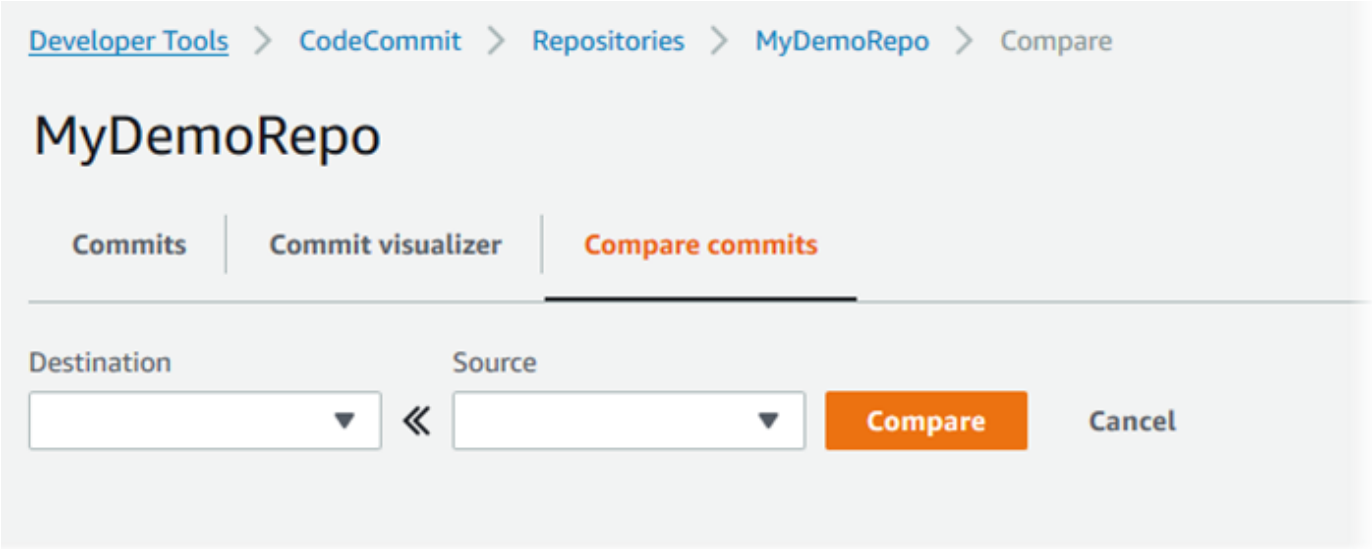
> **Note**
>
> 행 마무리 스타일, 코드 편집기 및 기타 요인 등에 따라 한 행의 특정 변경 내용 대신 전체 행이 추가되거나 삭제될 수 있습니다. 세부 수준은 git show 또는 git diff 명령에서 반환되는 것과 일치합니다.

5. 한 커밋을 상위 커밋과 비교하려면 커밋 시각화 도우미 탭에서 약식 커밋 ID를 선택합니다. 한 커밋과 상위 커밋 간의 변경 사항을 비롯한 커밋 세부 정보가 표시됩니다.

임의의 두 커밋 지정자 비교

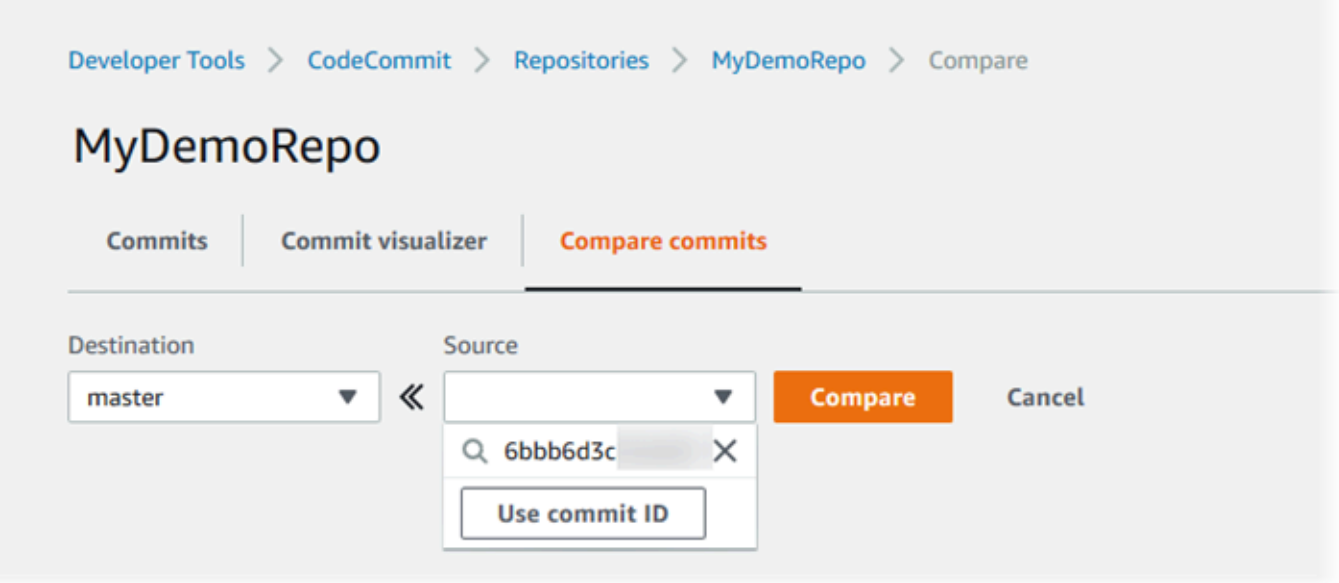
CodeCommit 콘솔에서 두 커밋 지정자 간의 차이점을 확인할 수 있습니다. 커밋 지정자는 브랜치, 태그, 커밋 ID와 같은 참조 정보입니다.

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리 페이지에서, 커밋, 브랜치 또는 태그 지정된 커밋을 비교하려는 리포지토리를 선택합니다.
3. 탐색 창에서 커밋을 선택한 다음 커밋 비교를 선택합니다.

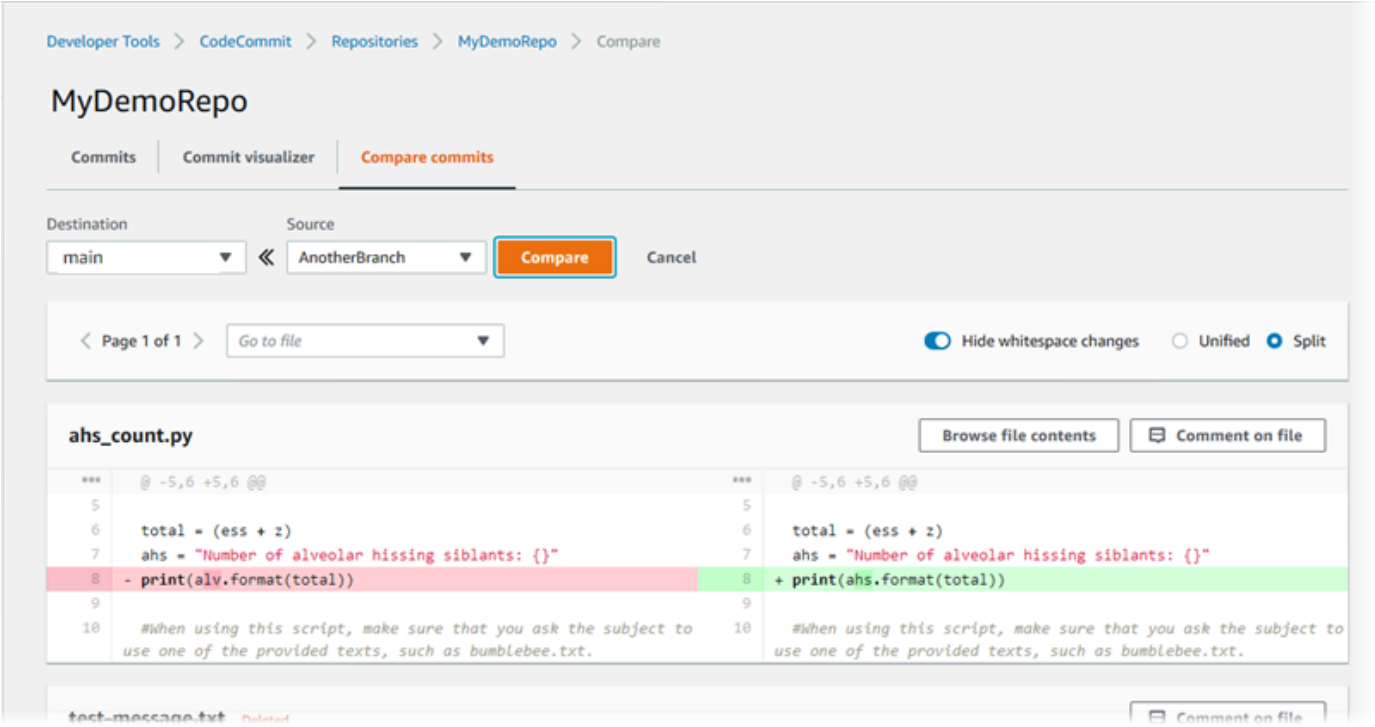


4. 상자를 사용하여 두 커밋 지정자를 비교할 수 있습니다.

   - 브랜치의 팁을 비교하려면 목록에서 브랜치 이름을 선택합니다. 그러면 해당 브랜치에서 비교를 위해 가장 최근 커밋을 선택합니다.
   - 커밋을 관련 특정 태그와 비교하려면 목록에서 태그 이름(있는 경우)을 선택합니다. 그러면 비교를 위해 태그가 지정된 커밋이 선택됩니다.
   - 특정 커밋을 비교하려면 상자에 커밋 ID를 입력하거나 붙여넣습니다. 전체 커밋 ID를 가져오려면, 탐색 표시줄에서 커밋을 선택하고 목록에서 커밋 ID를 복사합니다. 커밋 비교 페이지에서 텍스트 상자에 전체 커밋 ID를 붙여넣고 커밋 ID 사용을 선택합니다.



5. 지정자를 선택한 후 비교를 선택합니다.



차이점은 나란히(분할 보기) 또는 일렬(통합 보기)로 표시할 수 있습니다. 공백 변경을 숨기거나 표시할 수도 있습니다.

6. 비교 선택 항목을 지우려면 취소를 선택합니다.

의 커밋에 대한 설명 AWS CodeCommit

CodeCommit 콘솔을 사용하여 리포지토리의 커밋에 대한 설명을 추가하고, 커밋에 대한 다른 사용자의 설명을 보거나 댓글을 작성할 수 있습니다. 이렇게 하면 다음 사항을 포함한 리포지토리 변경 내용에 대해 논의할 수 있습니다.

- 변경을 수행한 이유
- 추가 변경이 필요한지 여부
- 변경 사항을 다른 브랜치에 병합해야 할지 여부

전체 커밋, 커밋 내 파일 또는 특정 행에 대한 설명을 추가하거나 파일 내에서 변경할 수 있습니다. 코드 행 하나를 선택한 다음 그에 따른 URL을 복사하여 해당 코드 행에 링크를 걸 수 있습니다.

> **Note**
>
> 최상의 결과를 위해, IAM 사용자로 로그인할 때 주석 달기 기능을 사용합니다. 주석 달기 기능은 루트 계정 자격 증명, 연동 액세스 권한 또는 임시 자격 증명으로 로그인하는 사용자에 맞게 최적화되지 않았습니다.

주제

- 리포지토리의 커밋에 대한 주석 보기
- 리포지토리의 커밋에 대한 주석 추가 및 댓글 달기
- 설명 보기, 추가, 업데이트, 답변 (AWS CLI)

리포지토리의 커밋에 대한 주석 보기

CodeCommit 콘솔을 사용하여 커밋에 대한 주석을 볼 수 있습니다.

커밋에 대한 설명 보기

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서, 커밋에 대한 주석을 보고자 하는 리포지토리를 선택합니다.
3. 탐색 창에서 커밋을 선택합니다. 주석을 살펴볼 커밋의 커밋 ID를 선택합니다.

   해당 커밋의 페이지와 주석이 표시됩니다.

리포지토리의 커밋에 대한 주석 추가 및 댓글 달기

CodeCommit 콘솔을 사용하여 한 커밋과 상위 커밋의 비교 또는 지정한 두 커밋 간의 비교에 주석을 추가할 수 있습니다. 한 주석에 이모티콘이나 자신의 주석 또는 둘 다 사용하여 댓글을 달 수 있습니다.

커밋에 대한 주석 추가 및 댓글 달기 (콘솔)

텍스트나 이모티콘으로 커밋에 주석을 추가하거나 주석에 댓글을 달 수 있습니다. 주석과 이모티콘은 콘솔에 로그인하는 데 사용한 역할이나 IAM 사용자의 것으로 표시됩니다.

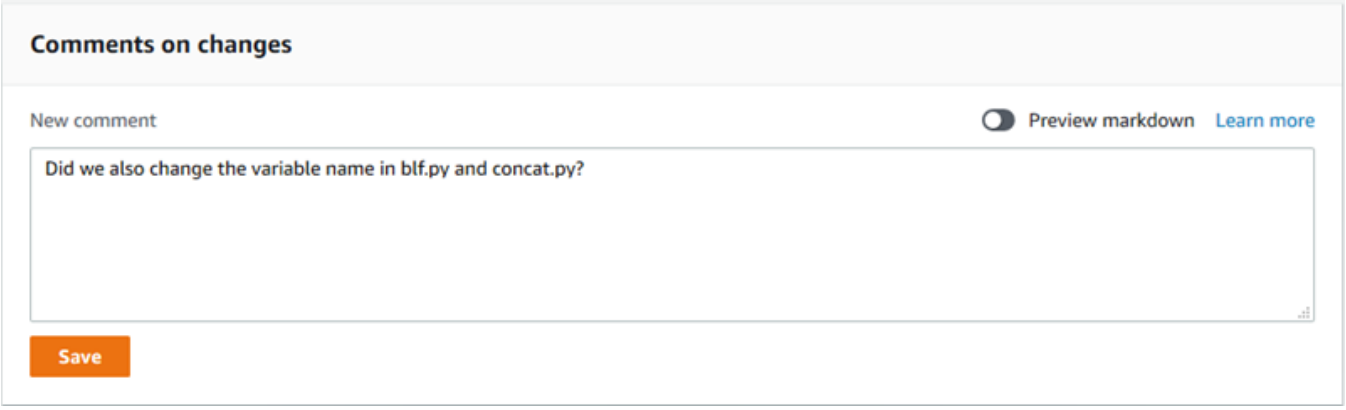
커밋에 주석을 추가하고 주석에 댓글을 달려면

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.

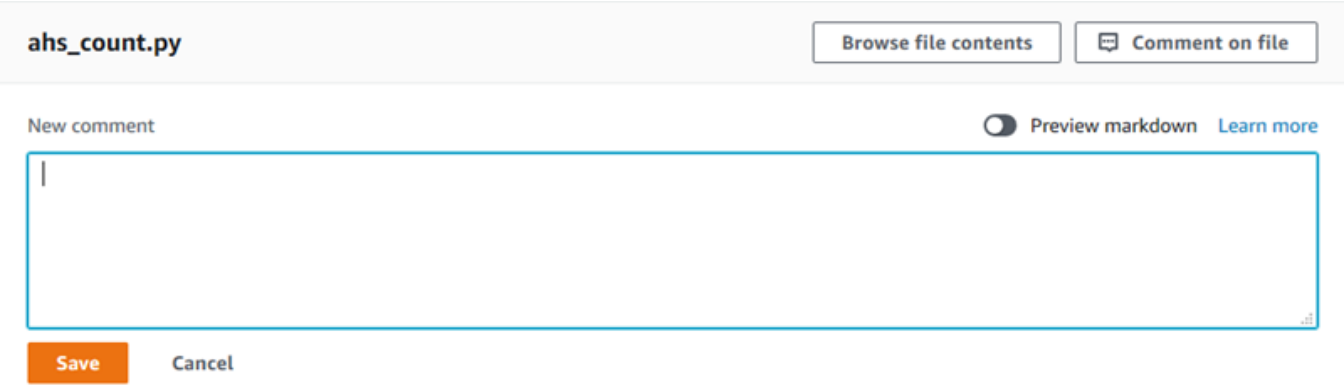
2. 리포지토리에서, 커밋에 대한 주석을 작성하려는 리포지토리를 선택합니다.
3. 탐색 창에서 커밋을 선택합니다. 주석을 추가하거나 주석에 댓글을 달 커밋의 커밋 ID를 선택합니다.

   해당 커밋의 페이지와 주석이 표시됩니다.
4. 주석을 추가하려면 다음 중 하나를 수행합니다.

   - 일반 설명을 추가하려면 변경 사항에 대한 주석에 주석을 입력한 다음 저장을 선택합니다. 마크다운을 사용하거나, 일반 텍스트로 주석을 입력할 수 있습니다.



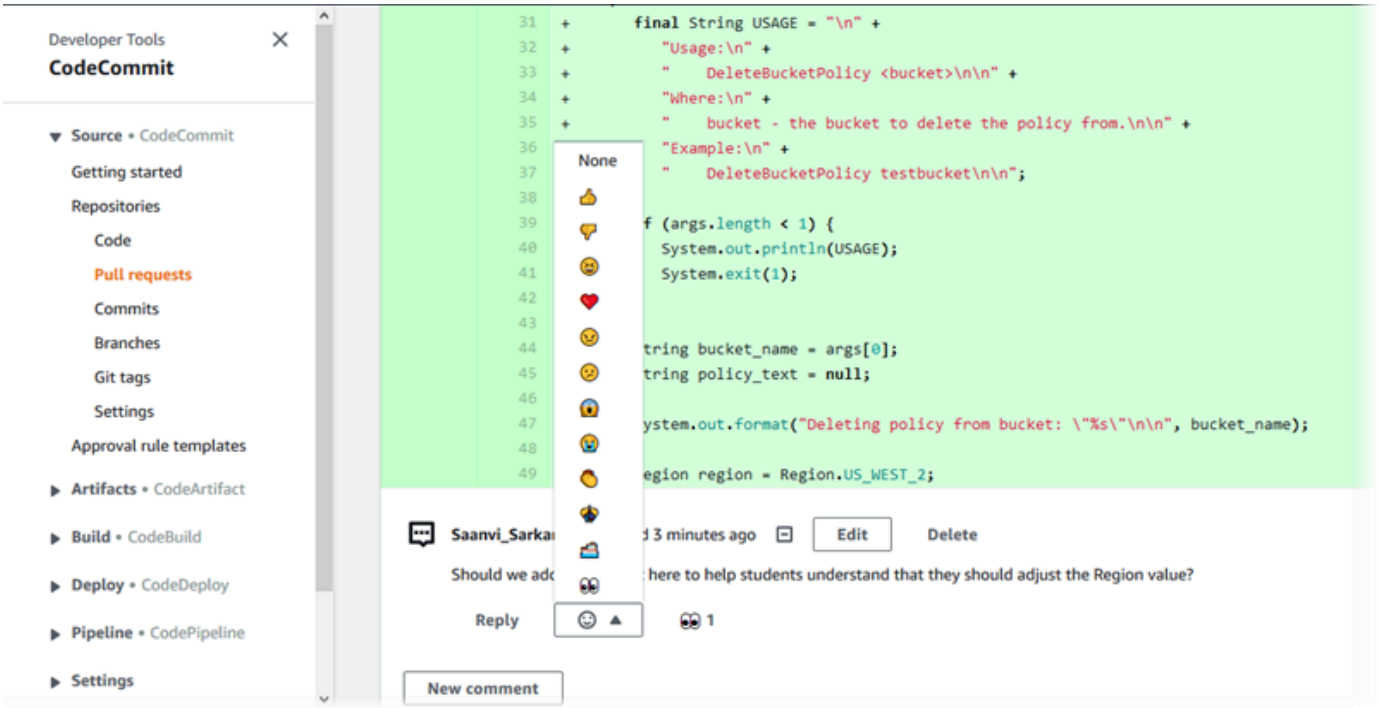
   - 커밋의 파일에 주석을 추가하려면 해당 파일의 이름을 찾습니다. 파일에 대한 주석을 선택하여 주석을 입력한 다음 저장을 선택합니다.



   - 커밋의 변경된 행에 주석을 추가하려면 변경 내용이 표시된 행으로 이동합니다. 주석 말풍선 를 선택하여 주석을 입력한 다음 저장을 선택합니다.

> **Note**
>
> 주석을 저장한 후 수정할 수 있습니다. 주석의 내용을 삭제할 수도 있습니다. 이 경우 내용이 삭제되었다는 메시지가 뜨며 주석은 유지됩니다. 주석을 저장하기 전에 마크다운 미리보기 모드를 사용하여 주석을 확인하는 것이 좋습니다.

5. 커밋에 대한 설명에 답변하려면 회신을 선택합니다. 이모티콘으로 주석에 댓글을 달려면 목록에서 원하는 이모티콘을 선택합니다. 주석 하나당 이모티콘은 하나만 선택할 수 있습니다. 이모티콘 반응을 변경하려면 목록에서 다른 이모티콘을 선택하거나, 반응을 제거하기 위해 없음을 선택합니다.

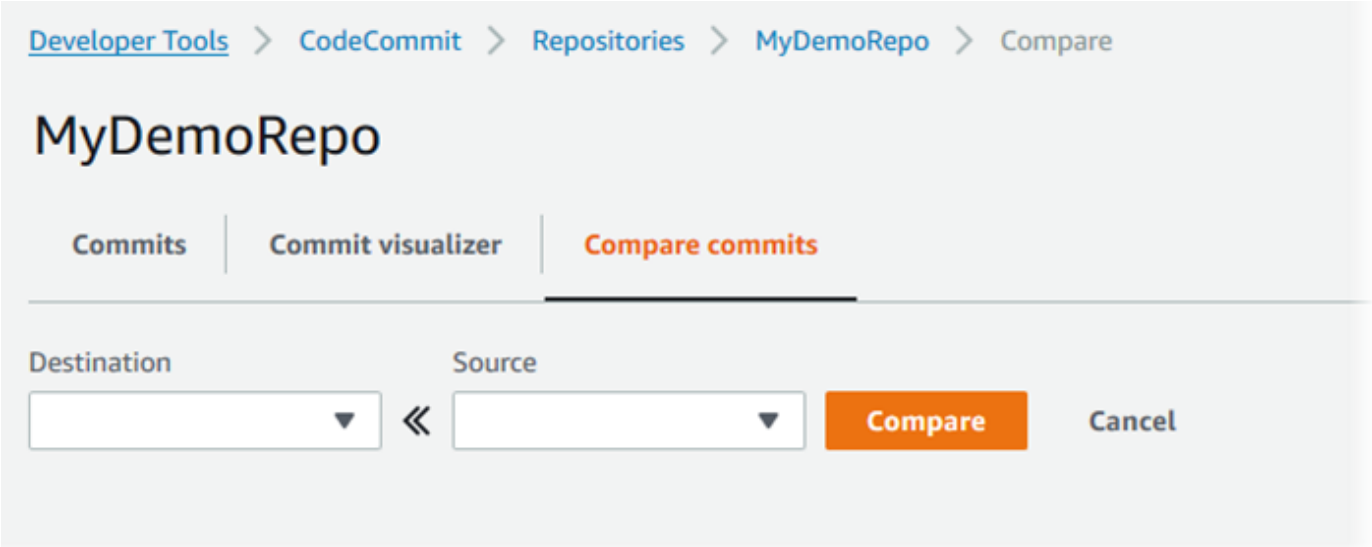


두 커밋 지정자 비교 시 주석 추가 및 주석에 댓글 달기

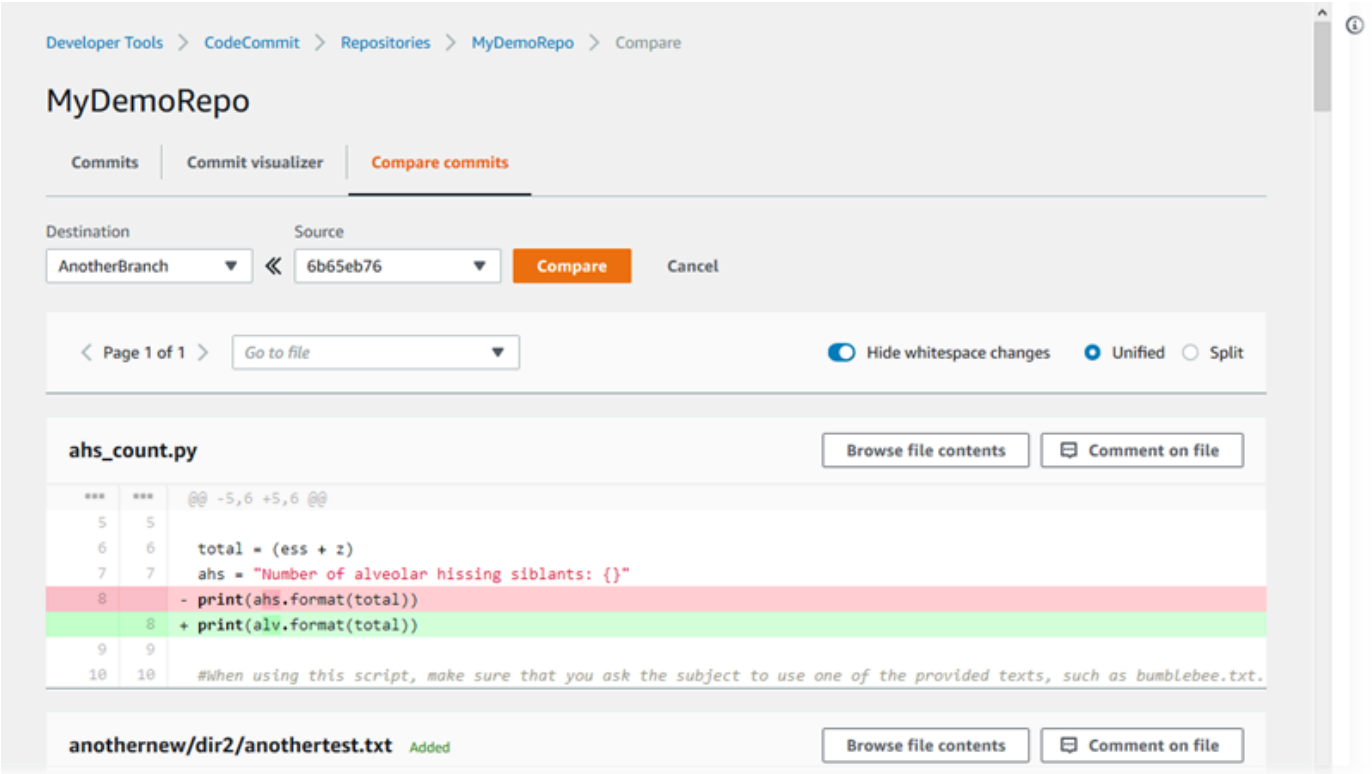
브랜치, 태그 또는 커밋 간 비교에 대한 주석을 추가할 수 있습니다.

커밋 지정자 비교 시 주석을 추가하거나 주석에 대한 댓글을 남기려면

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서, 커밋, 브랜치 또는 태그 지정된 커밋을 비교하려는 리포지토리를 선택합니다.
3. 탐색 창에서 커밋을 선택한 다음 커밋 비교 탭을 선택합니다.



4. 대상 주소 및 소스 필드를 사용하여 두 커밋 지정자를 비교합니다. 드롭다운 목록을 사용하거나 커밋 ID를 붙여 넣습니다. 비교를 선택합니다.



5. 다음 중 한 개 이상을 수행할 수 있습니다.

   - 파일이나 행에 주석을 추가하려면 주석 말풍선을 선택합니다

     .

   - 비교한 변경 사항에 대한 일반적인 주석을 추가하려면 변경 사항에 대한 주석으로 이동합니다.

설명 보기, 추가, 업데이트, 답변 (AWS CLI)

다음 명령을 실행하여 주석을 보고, 추가하고, 답변을 달고, 주석 내용을 삭제할 수 있습니다.

- 두 커밋 간 비교에 대한 주석을 보려면 get-comments-for-compared-commit를 실행합니다.
- 주석에 대한 세부 정보를 보려면 get-comment를 실행합니다.
- 작성한 주석의 내용을 삭제하려면 delete-comment-content를 실행합니다.
- 두 커밋 간 비교에 대한 주석을 작성하려면 post-comment-for-compared-commit을 실행합니다.
- 주석을 업데이트하려면 update-comment를 실행합니다.
- 주석에 댓글을 달려면 post-comment-reply를 실행합니다.
- 주석에 이모티콘으로 반응하려면 put-comment-reaction을 실행합니다.
- 주석에 대한 이모티콘 반응을 보려면 get-comment-reactions을 실행합니다.

커밋에 대한 설명 보기

1. 다음을 지정하여 get-comments-for-compared-commit 명령을 실행합니다.

   - CodeCommit 리포지토리의 이름(`--repository-name` 옵션 사용)
   - 비교 방향을 설정하기 위한 after 커밋의 전체 커밋 ID(`--after-commit-id option` 사용).
   - 비교 방향을 설정하기 위한 before 커밋의 전체 커밋 ID(`--before-commit-id` 옵션 사용).
   - (선택 사항) 다음 결과 세트를 반환하는 열거형 토큰(`--next-token` 옵션 사용).
   - (선택 사항) 반환된 결과의 수를 제한하는 음수가 아닌 정수(`--max-results` 옵션 사용).

   예를 들어 *MyDemoRepo*라는 리포지토리의 두 커밋 간의 비교에 대해 작성된 설명을 보려면 다음을 수행합니다.

   ```
   aws codecommit get-comments-for-compared-commit --repository-name MyDemoRepo --
   before-commit-id 6e147360EXAMPLE --after-commit-id 317f8570EXAMPLE
   ```

2. 이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

   ```
   {
      "commentsForComparedCommitData": [
         {
            "afterBlobId": "1f330709EXAMPLE",
            "afterCommitId": "317f8570EXAMPLE",
   ```

```
        "beforeBlobId": "80906a4cEXAMPLE",
        "beforeCommitId": "6e147360EXAMPLE",
        "comments": [
           {
              "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
              "clientRequestToken": "123Example",
              "commentId": "ff30b348EXAMPLEb9aa670f",
              "content": "Whoops - I meant to add this comment to the line, not
 the file, but I don't see how to delete it.",
              "creationDate": 1508369768.142,
              "deleted": false,
              "CommentId": "123abc-EXAMPLE",
              "lastModifiedDate": 1508369842.278,
              "callerReactions": [],
              "reactionCounts":
               {
                 "SMILE" : 6,
                 "THUMBSUP" : 1
               }
           },
           {
              "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
              "clientRequestToken": "123Example",
              "commentId": "553b509bEXAMPLE56198325",
              "content": "Can you add a test case for this?",
              "creationDate": 1508369612.240,
              "deleted": false,
              "commentId": "456def-EXAMPLE",
              "lastModifiedDate": 1508369612.240,
              "callerReactions": [],
              "reactionCounts":
               {
                 "THUMBSUP" : 2
               }
            }
        ],
        "location": {
           "filePath": "cl_sample.js",
           "filePosition": 1232,
           "relativeFileVersion": "after"
        },
        "repositoryName": "MyDemoRepo"
     }
  ],
```

```
   "nextToken": "exampleToken"
}
```

커밋의 주석에 대한 세부 정보를 보려면

1. 시스템에서 생성한 설명 ID를 지정하여 get-comment 명령을 실행합니다. 예시:

```
aws codecommit get-comment --comment-id ff30b348EXAMPLEb9aa670f
```

2. 이 명령이 제대로 실행되면 다음과 비슷한 출력이 반환됩니다.

```
{
   "comment": {
      "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
      "clientRequestToken": "123Example",
      "commentId": "ff30b348EXAMPLEb9aa670f",
      "content": "Whoops - I meant to add this comment to the line, but I don't see
 how to delete it.",
      "creationDate": 1508369768.142,
      "deleted": false,
      "commentId": "",
      "lastModifiedDate": 1508369842.278,
      "callerReactions": [],
      "reactionCounts":
         {
           "SMILE" : 6,
           "THUMBSUP" : 1
         }
   }
}
```

커밋의 주석 내용을 삭제하려면

1. 시스템에서 생성한 설명 ID를 지정하여 delete-comment-content 명령을 실행합니다. 예시:

```
aws codecommit delete-comment-content --comment-id ff30b348EXAMPLEb9aa670f
```

> **Note**
>
> AWSCodeCommitFullAccess 정책이 적용되었거나 `DeleteCommentContent` 권한이 허용으로 설정된 경우에만 주석의 내용을 삭제할 수 있습니다.

2. 이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
{
   "comment": {
      "creationDate": 1508369768.142,
      "deleted": true,
      "lastModifiedDate": 1508369842.278,
      "clientRequestToken": "123Example",
      "commentId": "ff30b348EXAMPLEb9aa670f",
      "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
      "callerReactions": [],
      "reactionCounts":
         {
           "CLAP" : 1
         }
   }
}
```

커밋에 설명 작성

1. 다음을 지정하여 post-comment-for-compared-commit 명령을 실행합니다.

   - CodeCommit 리포지토리의 이름(`--repository-name` 옵션 사용)
   - 비교 방향을 설정하기 위한 after 커밋의 전체 커밋 ID(`--after-commit-id` 옵션 사용).
   - 비교 방향을 설정하기 위한 before 커밋의 전체 커밋 ID(`--before-commit-id` 옵션 사용).
   - 클라이언트에서 생성된 고유한 멱등성 토큰(--client-request-token 옵션 사용).
   - 설명의 내용(--content 옵션 사용).
   - 다음을 비롯하여 주석을 추가할 위치에 대한 위치 정보 목록.
     - 확장자와 하위 디렉터리를 비롯하여 비교하려는 파일의 이름(filePath 속성 사용).
     - 비교한 파일 내 변경의 행 번호(filePosition 속성 사용).

- 변경에 대한 주석이 소스와 대상 브랜치 간의 비교 전인지 후인지 여부(relativeFileVersion 속성 사용).

예를 들어, *"Can you add a test case for this?"*라는 주석을 *MyDemoRepo*라는 리포지토리의 두 커밋을 비교하여 *cl_sample.js* 파일을 변경한 것에 대해 추가하려면, 다음과 같이 합니다.

```
aws codecommit post-comment-for-compared-commit --repository-name MyDemoRepo
 --before-commit-id 317f8570EXAMPLE --after-commit-id 5d036259EXAMPLE --client-
request-token 123Example --content "Can you add a test case for this?" --location
 filePath=cl_sample.js,filePosition=1232,relativeFileVersion=AFTER
```

2. 이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
{
        "afterBlobId": "1f330709EXAMPLE",
        "afterCommitId": "317f8570EXAMPLE",
        "beforeBlobId": "80906a4cEXAMPLE",
        "beforeCommitId": "6e147360EXAMPLE",
        "comment": {
              "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
              "clientRequestToken": "",
              "commentId": "553b509bEXAMPLE56198325",
              "content": "Can you add a test case for this?",
              "creationDate": 1508369612.203,
              "deleted": false,
              "commentId": "abc123-EXAMPLE",
              "lastModifiedDate": 1508369612.203,
              "callerReactions": [],
              "reactionCounts": []
            },
            "location": {
              "filePath": "cl_sample.js",
              "filePosition": 1232,
              "relativeFileVersion": "AFTER"
            },
        "repositoryName": "MyDemoRepo"
 }
```

커밋에 대한 설명 업데이트

1. 시스템에서 생성한 주석 ID와 기존 내용을 교체하려는 내용을 지정하여 update-comment 명령을 실행합니다.

   예를 들어, *"Fixed as requested. I'll update the pull request."*라는 내용을 ID가 *442b498bEXAMPLE5756813*인 주석으로 추가하려면, 다음과 같이 합니다.

```
aws codecommit update-comment --comment-id 442b498bEXAMPLE5756813 --content "Fixed
 as requested. I'll update the pull request."
```

2. 이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
{
    "comment": {
        "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
        "clientRequestToken": "",
        "commentId": "442b498bEXAMPLE5756813",
        "content": "Fixed as requested. I'll update the pull request.",
        "creationDate": 1508369929.783,
        "deleted": false,
        "lastModifiedDate": 1508369929.287,
        "callerReactions": [],
        "reactionCounts":
          {
            "THUMBSUP" : 2
          }
    }
 }
```

커밋에 대한 주석에 댓글을 달려면

1. 풀 요청의 주석에 댓글을 게시하려면 다음을 지정하여 post-comment-reply 명령을 실행합니다.

   - 댓글을 달 주석의 시스템 생성 ID(--in-reply-to 옵션 사용).
   - 클라이언트에서 생성된 고유한 멱등성 토큰(--client-request-token 옵션 사용).
   - 댓글의 내용(--content 옵션 사용).

예를 들어 *"Good catch. I'll remove them."*라는 댓글을 시스템 생성 ID가 *abcd1234EXAMPLEb5678efgh*인 주석에 달려면 다음과 같이 합니다.

```
aws codecommit post-comment-reply --in-reply-to abcd1234EXAMPLEb5678efgh --
content "Good catch. I'll remove them." --client-request-token 123Example
```

2. 이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
{
    "comment": {
        "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
        "clientRequestToken": "123Example",
        "commentId": "442b498bEXAMPLE5756813",
        "content": "Good catch. I'll remove them.",
        "creationDate": 1508369829.136,
        "deleted": false,
        "CommentId": "abcd1234EXAMPLEb5678efgh",
        "lastModifiedDate": 150836912.221,
        "callerReactions": [],
        "reactionCounts": []
    }
 }
```

커밋에 대한 주석에 이모티콘으로 댓글을 달려면

1. 풀 요청의 주석에 이모티콘으로 댓글을 달거나 이모티콘 반응의 값을 변경하려면 다음과 같이 지정하여 put-comment-reaction 명령을 실행합니다.

   - 이모티콘으로 댓글을 달 주석의 시스템 생성 ID.
   - 추가 또는 업데이트하려는 반응의 값. 허용되는 값에는 지원되는 이모티콘, 단축 코드, 유니코드 값 등이 있습니다.

   CodeCommit에서 이모티콘에 지원되는 값은 다음과 같습니다.

이모티콘	ShortCode	유니코드
#	:thumbsup:	U+1F44D
#	:thumbsdown:	U+1F44E
#	:smile:	U+1F604
❤	:heart:	U+2764
#	:angry:	U+1F620
#	:confused:	U+1F615
#	:scream:	U+1F631
#	:sob:	U+1F62D
#	:clap:	U+1F44F
#	:confetti_ball:	U+1F38A
#	:ship:	U+1F6A2
#	:eyes:	U+1F440
	없음	U+0000

예를 들어, 시스템 생성 ID가 *abcd1234EXAMPLEb5678efgh*인 주석에 이모티콘 *:thumbsup:*을 추가하려면 다음과 같이 합니다.

```
aws codecommit put-comment-reaction --comment-id abcd1234EXAMPLEb5678efgh --
reaction-value :thumbsup:
```

2. 성공하면 이 명령은 출력을 생성하지 않습니다.

주석에 대한 이모티콘 반응을 보려면

1. 이모티콘으로 반응한 사용자를 포함하여 주석에 대한 이모티콘 반응을 보려면 주석의 시스템 생성 ID를 지정하여 get-comment-reactions 명령을 실행합니다.

   예를 들어, 시스템 생성 ID가 *abcd1234EXAMPLEb5678efgh*인 주석에 대한 이모티콘 반응을 보려면 다음과 같이 합니다.

```
aws codecommit get-comment-reactions --comment-id abcd1234EXAMPLEb5678efgh
```

2. 이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
{
    "reactionsForComment": {
        [
           {
                "reaction": {
                    "emoji:"#",
                    "shortCode": "thumbsup",
                    "unicode": "U+1F44D"
                },
                "users": [
                    "arn:aws:iam::123456789012:user/Li_Juan",
                    "arn:aws:iam::123456789012:user/Mary_Major",
                    "arn:aws:iam::123456789012:user/Jorge_Souza"
                ]
            },
            {
                "reaction": {
                    "emoji": "#",
                    "shortCode": "thumbsdown",
                    "unicode": "U+1F44E"
                },
                "users": [
                    "arn:aws:iam::123456789012:user/Nikhil_Jayashankar"
                ]
            },
            {
                "reaction": {
                    "emoji": "#",
                    "shortCode": "confused",
                    "unicode": "U+1F615"
                },
```

```
                "users": [
                    "arn:aws:iam::123456789012:user/Saanvi_Sarkar"
                ]
            }
        ]
    }
}
```

에서 Git 태그 생성 AWS CodeCommit

Git 태그를 사용하여 다른 리포지토리 사용자가 중요성을 파악할 수 있도록 커밋에 레이블을 표시할 수 있습니다. CodeCommit 리포지토리에서 Git 태그를 만들려면, CodeCommit 리포지토리에 연결된 로컬 리포지토리에서 Git을 사용할 수 있습니다. 로컬 리포지토리에 Git 태그를 생성한 후에 git push --tags를 사용하여 CodeCommit 리포지토리로 푸시할 수 있습니다.

자세한 내용은 태그 세부 정보 보기 단원을 참조하십시오.

Git을 사용한 태그 생성

이 단계들을 따라 로컬 리포지토리에서 Git을 사용하여 CodeCommit 리포지토리에 Git 태그를 생성합니다.

이 단계들에서는 로컬 리포지토리를 이미 CodeCommit 리포지토리에 연결했다고 가정했습니다. 지침은 리포지토리에 연결 섹션을 참조하세요.

1. git tag ***new-tag-name commit-id*** 명령을 실행합니다. 여기서 *new-tag-name*은 새 Git 태그의 이름이고 *commit-id*는 Git 태그에 연결할 커밋의 ID입니다.

   예를 들어, 다음 명령을 실행하면 `beta`라는 Git 태그가 생성되고 이 태그는 커밋 ID `dc082f9a...af873b88`과 연결됩니다.

   ```
   git tag beta dc082f9a...af873b88
   ```

2. 새 Git 태그를 로컬 리포지토리에서 CodeCommit 리포지토리로 푸시하려면 git push ***remote-name new-tag-name*** 명령을 실행합니다. 여기서 *remote-name*은 CodeCommit 리포지토리의 이름이고 *new-tag-name*은 새 Git 태그의 이름입니다.

   예를 들어, `origin`이라는 CodeCommit 리포지토리에 `beta`라는 새 Git 태그를 푸시하려면 다음을 실행합니다.

```
git push origin beta
```

> **Note**
>
> 모든 새 Git 태그를 로컬 리포지토리에서 CodeCommit 리포지토리로 푸시하려면 git push --tags를 실행합니다.
> 로컬 리포지토리가 CodeCommit 리포지토리에 있는 모든 Git 태그로 업데이트되도록 하려면 git fetch를 실행한 다음 git fetch --tags를 실행합니다.

다른 옵션들에 대해서는 Git 설명서를 참조하세요.

에서 Git 태그 세부 정보 보기 AWS CodeCommit

Git에서 태그는 커밋과 같은 참조에 적용할 수 있는 레이블로, 다른 리포지토리 사용자에게 중요할 수 있는 정보를 표시합니다. 예를 들어 **beta** 태그를 사용하여 프로젝트의 베타 릴리스 포인트인 커밋에 태그를 지정할 수 있습니다. 자세한 내용은 Git을 사용한 태그 생성 단원을 참조하십시오. Git 태그는 리포지토리 태그와 다릅니다. 리포지토리 태그를 사용하는 방법에 대한 자세한 내용은 리포지토리에 태그 추가 단원을 참조하세요.

AWS CodeCommit 콘솔을 사용하여 각 Git 태그에서 참조하는 커밋의 날짜 및 커밋 메시지를 포함하여 리포지토리의 Git 태그에 대한 정보를 볼 수 있습니다. 콘솔에서 태그에서 참조되는 커밋과 리포지토리의 기본 브랜치 헤드를 비교할 수 있습니다. 다른 커밋과 마찬가지로 해당 Git 태그 지점에서도 코드를 볼 수 있습니다.

또한 터미널이나 명령줄에서 Git을 사용하여 로컬 리포지토리의 Git 태그에 대한 세부 정보를 볼 수 있습니다.

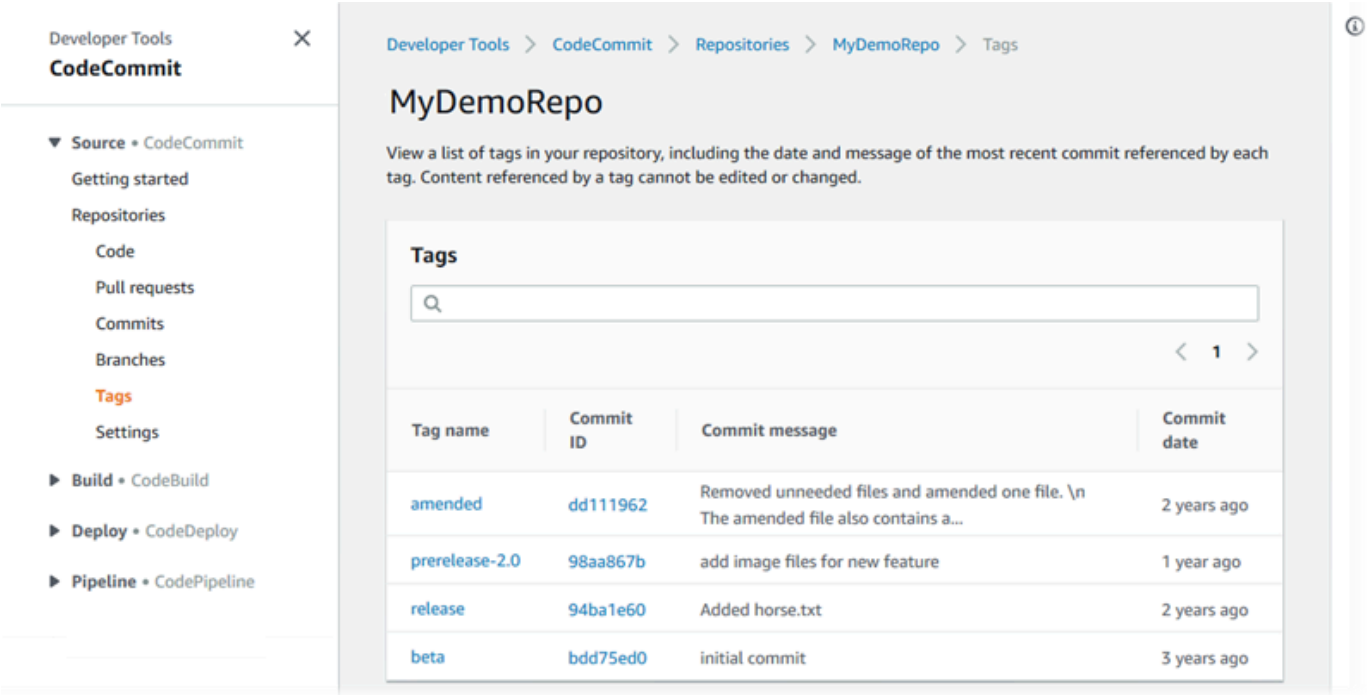
주제

- 태그 세부 정보 보기 (콘솔)
- Git 태그 세부 정보 보기 (Git)

태그 세부 정보 보기 (콘솔)

AWS CodeCommit 콘솔을 사용하여 리포지토리의 Git 태그 목록과 Git 태그에서 참조하는 커밋에 대한 세부 정보를 빠르게 볼 수 있습니다.

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서, 태그를 보려는 리포지토리의 이름을 선택합니다.
3. 탐색 창에서 Git 태그를 선택합니다.



4. 다음 중 하나를 수행합니다.
   - 해당 커밋에 있을 당시의 코드를 보려면 Git 태그 이름을 선택합니다.
   - 전체 커밋 메시지, 커미터 및 작성자 등 커밋에 대한 세부 정보를 보려면 약식 커밋 ID를 선택합니다.

Git 태그 세부 정보 보기 (Git)

Git을 사용하여 로컬 리포지토리에서 Git 태그에 대한 세부 정보를 보려면 다음 명령 중 하나를 실행합니다.

- git tag - Git 태그 이름의 목록을 봅니다.
- git show - 특정 Git 태그에 대한 정보를 봅니다.
- git ls-remote - CodeCommit 리포지토리의 Git 태그에 대한 정보를 봅니다.

> **Note**
> 로컬 리포지토리가 CodeCommit 리포지토리에 있는 모든 Git 태그로 업데이트되도록 하려면 git fetch를 실행한 다음 git fetch --tags를 실행합니다.

다음 단계들에서는 로컬 리포지토리를 이미 CodeCommit 리포지토리에 연결했다고 가정했습니다. 지침은 리포지토리에 연결 섹션을 참조하세요.

로컬 리포지토리의 Git 태그 목록을 보려면 다음을 수행합니다.

1. git tag 명령을 실행합니다.

```
git tag
```

2. 이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
beta
release
```

> **Note**
> 정의된 태그가 없는 경우 git tag가 아무 것도 반환하지 않습니다.

다른 옵션들에 대해서는 Git 설명서를 참조하세요.

로컬 리포지토리의 Git 태그에 대한 정보를 보려면 다음을 수행합니다.

1. git show ***tag-name*** 명령을 실행합니다. 예를 들어, Git 태그 `beta`에 대한 정보를 보려면 다음을 실행합니다.

```
git show beta
```

2. 이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
commit 317f8570...ad9e3c09
Author: John Doe <johndoe@example.com>
Date:   Tue Sep 23 13:49:51 2014 -0700
```

```
    Added horse.txt

diff --git a/horse.txt b/horse.txt
new file mode 100644
index 0000000..df42ff1
--- /dev/null
+++ b/horse.txt
@@ -0,0 +1 @@
+The horse (Equus ferus caballus) is one of two extant subspecies of Equus ferus
\ No newline at end of file
```

> **Note**
>
> Git 태그 정보 출력을 종료하려면 :q를 입력합니다.

다른 옵션들에 대해서는 Git 설명서를 참조하세요.

CodeCommit 리포지토리의 Git 태그에 대한 정보를 보려면

1. git ls-remote --tags 명령을 실행합니다.

```
git ls-remote --tags
```

2. 제대로 실행된 경우 이 명령은 CodeCommit 리포지토리에 Git 태그 목록을 출력으로 생성합니다.

```
129ce87a...70fbffba    refs/tags/beta
785de9bd...59b402d8    refs/tags/release
```

   정의된 Git 태그가 없는 경우 git ls-remote --tags가 빈 줄을 반환합니다.

다른 옵션들에 대해서는 Git 설명서를 참조하세요.

에서 Git 태그 삭제 AWS CodeCommit

CodeCommit 리포지토리에서 Git 태그를 삭제하려면, CodeCommit 리포지토리에 연결된 로컬 리포지토리에서 Git을 사용합니다.

Git을 사용하여 Git 태그 삭제

다음 단계에 따라 로컬 리포지토리에서 Git을 사용하여 CodeCommit 리포지토리의 Git 태그를 삭제합니다.

이러한 단계는 로컬 리포지토리를 이미 CodeCommit 리포지토리에 연결했다는 가정하에 작성되었습니다. 지침은 리포지토리에 연결 섹션을 참조하세요.

1. 로컬 리포지토리에서 Git 태그를 삭제하려면 git tag -d ***tag-name*** 명령을 실행합니다. 여기서 *tag-name*은 삭제할 Git 태그의 이름입니다.

   > **Tip**
   >
   > Git 태그 이름 목록을 가져오려면 git tag를 실행하세요.

   예를 들어, `beta`라는 로컬 리포지토리에서 Git 태그를 삭제하려면 다음을 실행합니다.

   ```
   git tag -d beta
   ```

2. CodeCommit 리포지토리에서 Git 태그를 삭제하려면 git push ***remote-name*** --delete ***tag-name*** 명령을 실행합니다. 여기서 *remote-name*은 로컬 리포지토리가 CodeCommit 리포지토리에 사용하는 별명이고, *tag-name*은 CodeCommit 리포지토리에서 삭제할 Git 태그의 이름입니다.

   > **Tip**
   >
   > CodeCommit 리포지토리 이름의 목록과 그 URL을 가져오려면 git remote -v 명령을 실행합니다.

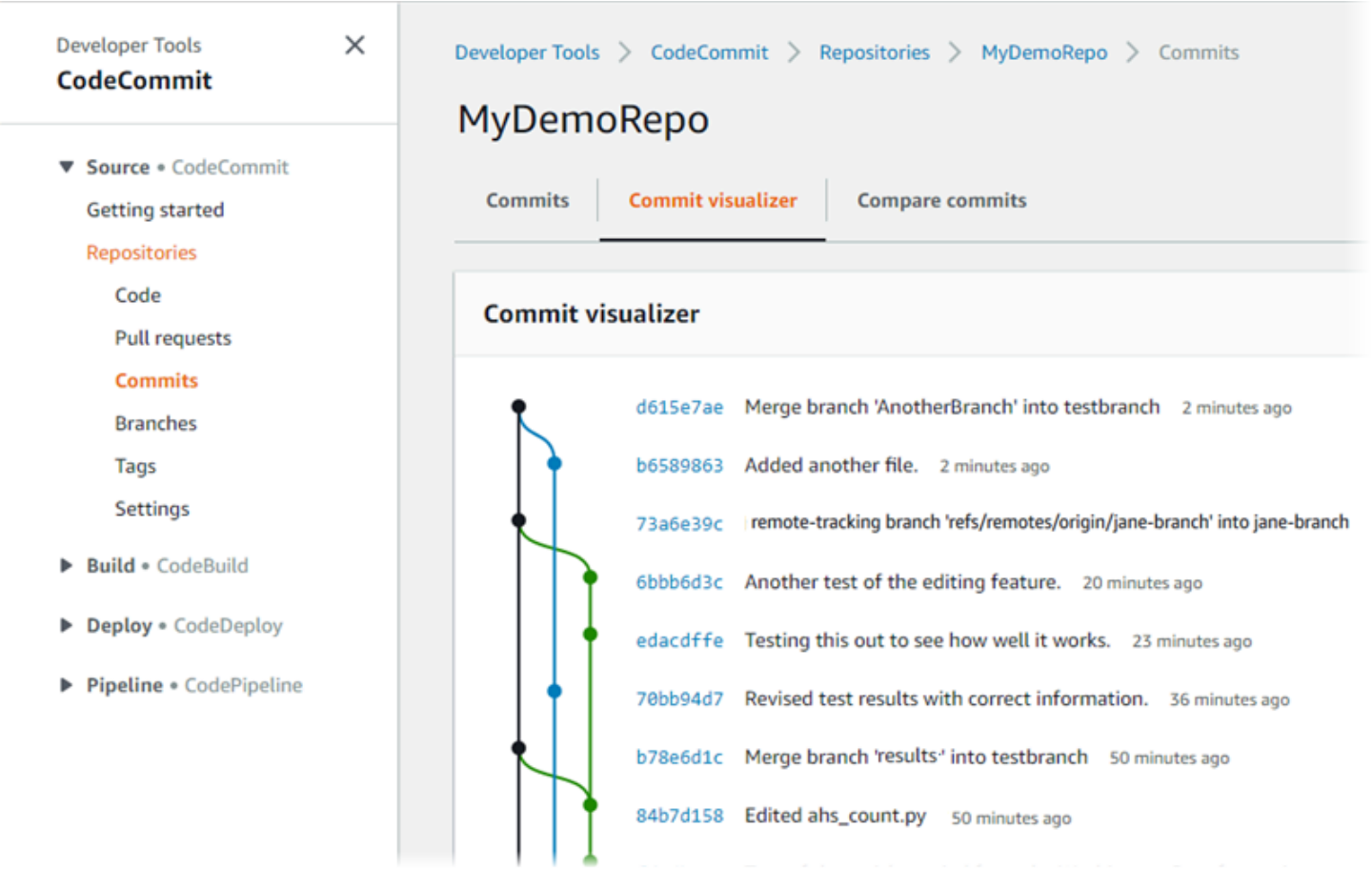
   예를 들어, `origin`이라는 CodeCommit 리포지토리에서 `beta`라는 Git 태그를 삭제하려면 다음을 실행합니다.

   ```
   git push origin --delete beta
   ```

AWS CodeCommit 리포지토리의 브랜치 작업

브랜치란 무엇인가요? Git에서 브랜치는 커밋에 대한 포인터 또는 참조입니다. 개발 시 작업을 체계적으로 정리할 수 있는 편리한 수단입니다. 브랜치를 사용하면 다른 브랜치의 작업에 영향을 주지 않으면서 파일의 새 버전이나 다른 버전에 대한 작업을 분리할 수 있습니다. 브랜치를 사용하면 새 기능을 개발하고 특정 커밋의 특정 프로젝트 버전을 저장하는 등의 작업을 수행할 수 있습니다. 첫 번째 커밋을 만들면 기본 브랜치가 자동으로 생성됩니다. 이 기본 브랜치는 사용자가 리포지토리를 복제할 때 로컬 리포지토리에서 기본 브랜치로 사용될 브랜치입니다. 기본 브랜치의 이름은 첫 커밋을 만드는 방법에 따라 달라집니다. CodeCommit 콘솔, AWS CLI또는 SDKs 중 하나를 사용하여 리포지토리에 첫 번째 파일을 추가하는 경우 해당 기본 브랜치의 이름이 기본입니다. 이 이름이 본 안내서의 예제에 사용된 기본 브랜치 이름입니다. Git 클라이언트를 사용하여 첫 번째 커밋을 푸시하는 경우에는 Git 클라이언트가 기본 브랜치로 지정하는 이름이 기본 브랜치의 이름입니다. 첫 브랜치의 이름으로 main을 사용하도록 Git 클라이언트 구성해 보세요.

CodeCommit에서는 리포지토리의 기본 브랜치를 변경할 수 있습니다. 브랜치를 생성 및 삭제하고 브랜치에 대한 세부 정보를 볼 수도 있습니다. 브랜치와 기본 브랜치(또는 임의의 두 브랜치) 간의 차이를 빠르게 비교할 수 있습니다. 리포지토리의 브랜치 및 병합 기록을 보려면, 다음 그림에 표시된 커밋 데이터 시각화 도우미를 사용하면 됩니다.



CodeCommit에서 리포지토리의 다른 측면으로 작업하는 방법에 대해 자세히 알아보려면 리포지토리 작업, 파일 작업하기, 풀 요청 작업, 커밋 작업, 사용자 기본 설정으로 작업 섹션을 참조하세요.

주제

- 에서 브랜치 생성 AWS CodeCommit
- 의 브랜치에 대한 푸시 및 병합 제한 AWS CodeCommit
- 에서 브랜치 세부 정보 보기 AWS CodeCommit
- AWS CodeCommit에서 브랜치 비교 및 병합
- 에서 브랜치 설정 변경 AWS CodeCommit
- 에서 브랜치 삭제 AWS CodeCommit

에서 브랜치 생성 AWS CodeCommit

CodeCommit 콘솔 또는를 사용하여 리포지토리 AWS CLI 에 대한 브랜치를 생성할 수 있습니다. 이렇게 하면 기본 브랜치의 작업에 영향을 주지 않으면서 파일의 새 버전 또는 다른 버전에 대한 작업을 쉽

게 분리할 수 있습니다. CodeCommit 콘솔에서 브랜치를 생성한 후 이 변경 내용을 로컬 리포지토리로 풀해야 합니다. 또는 브랜치를 로컬로 생성한 다음 CodeCommit 리포지토리에 연결된 로컬 리포지토리의 Git을 사용하여 해당 변경 내용을 푸시할 수도 있습니다.

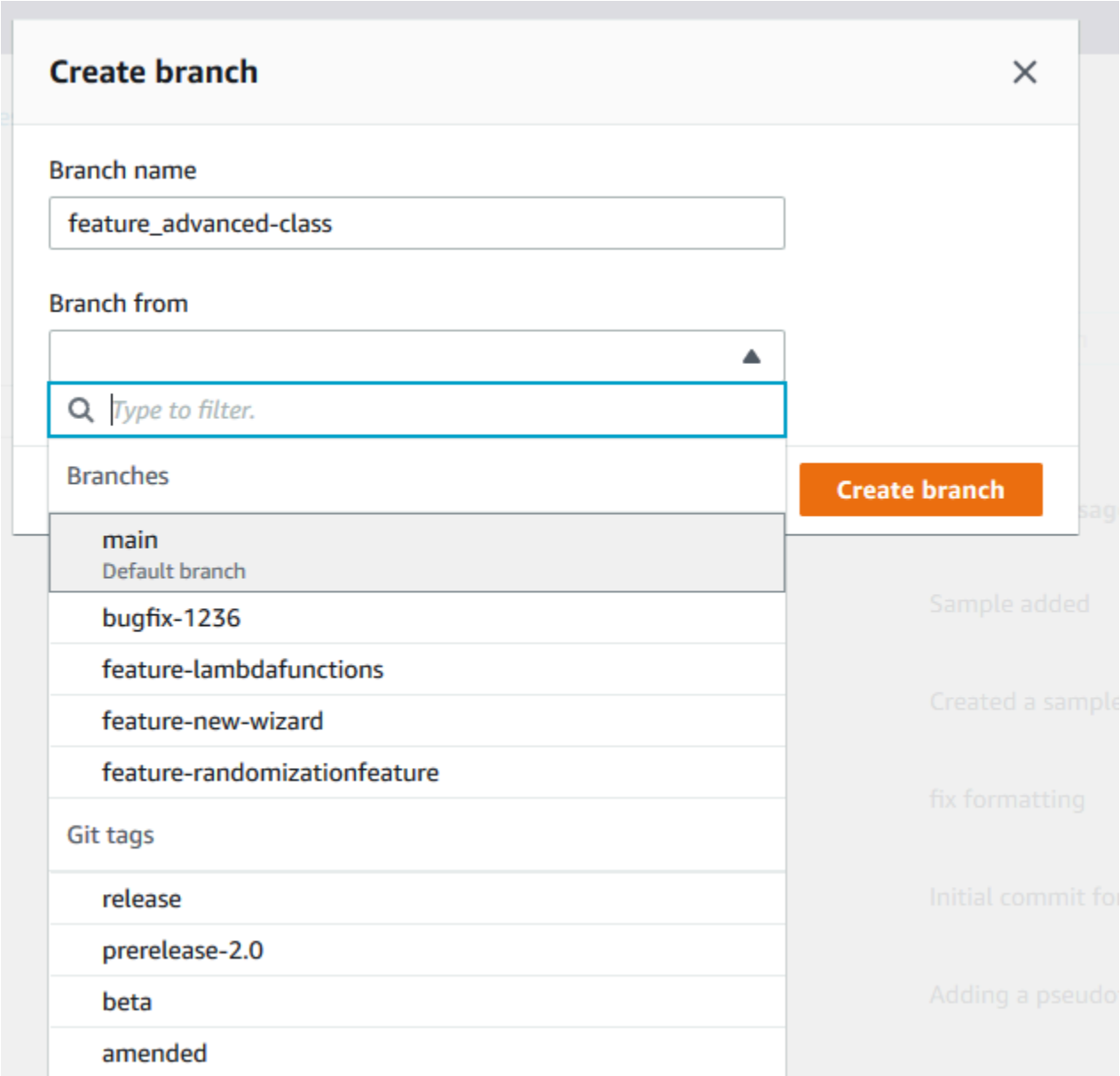
주제

- 브랜치 생성 (콘솔)
- 브랜치 생성 (Git)
- 브랜치 생성 (AWS CLI)

브랜치 생성 (콘솔)

CodeCommit 콘솔을 사용하여 CodeCommit 리포지토리에 브랜치를 생성할 수 있습니다. 사용자는 다음에 리포지토리에서 변경 내용을 풀할 때 새 브랜치를 볼 수 있습니다.

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서, 브랜치를 생성하려는 리포지토리의 이름을 선택합니다.
3. 탐색 창에서 Branches를 선택합니다.
4. 브랜치 생성을 선택합니다.



브랜치 이름에 브랜치의 이름을 입력합니다. 브랜치 위치에서 목록에 있는 브랜치 또는 태그를 선택하거나 커밋 ID를 붙여 넣습니다. 브랜치 생성을 선택합니다.

브랜치 생성 (Git)

이러한 단계에 따라 로컬 리포지토리에서 Git을 사용하여 로컬 리포지토리에 브랜치를 생성한 다음 이 브랜치를 CodeCommit 리포지토리로 푸시합니다.

이 단계들은 로컬 리포지토리를 이미 CodeCommit 리포지토리에 연결했다는 가정하에 작성되었습니다. 지침은 리포지토리에 연결 섹션을 참조하세요.

1. git checkout -b ***new-branch-name*** 명령을 실행하여 로컬 리포지토리에 새 브랜치를 생성합니다. 여기서 *new-branch-name*은 새 브랜치의 이름입니다.

   예를 들어 다음 명령은 로컬 리포지토리에 `MyNewBranch`라는 브랜치를 생성합니다.

   ```
   git checkout -b MyNewBranch
   ```

2. 새 브랜치를 로컬 리포지토리에서 CodeCommit 리포지토리로 푸시하려면 git push 명령을 실행하여 ***remote-name*** 및 ***new-branch-name***을 모두 지정합니다.

   예를 들어 로컬 리포지토리의 `MyNewBranch`라는 새 브랜치를 `origin`이라는 별명의 CodeCommit 리포지토리로 푸시하려면 다음을 수행합니다.

   ```
   git push origin MyNewBranch
   ```

> **Note**
>
> `-u` 옵션을 git push에 추가하면(예: git push -u origin main), 나중에 *remote-name*, *branch-name*을 사용하지 않고 git push 를 실행할 수 있습니다. 업스트림 추적 정보가 설정됩니다. 업스트림 추적 정보를 보려면 git remote show ***remote-name***을 실행합니다(예: git remote show origin).
> 모든 로컬 및 원격 추적 브랜치의 목록을 보려면 git branch --all을 실행합니다.
> CodeCommit 리포지토리의 브랜치에 연결된 로컬 리포지토리의 브랜치를 설정하려면 git checkout ***remote-branch-name***을 실행합니다.

다른 옵션들에 대해서는 Git 설명서를 참조하세요.

브랜치 생성 (AWS CLI)

CodeCommit에서 AWS CLI 명령을 사용하려면를 설치합니다 AWS CLI. 자세한 내용은 명령줄 참조 단원을 참조하십시오.

다음 단계에 따라 AWS CLI 를 사용하여 CodeCommit 리포지토리에 브랜치를 생성한 다음 해당 브랜치를 CodeCommit 리포지토리로 푸시합니다. 초기 커밋을 만들고 빈 리포지토리의 기본 브랜치 이름을 지정하는 단계들에 대해서는 AWS CLI를 사용하여 리포지토리의 첫 번째 커밋 생성을 참조하세요.

1. 다음을 지정하여 create-branch 명령을 실행합니다.

- 브랜치를 생성할 CodeCommit 리포지토리의 이름(--repository-name 옵션 사용).

> **Note**
>
> CodeCommit 리포지토리의 이름을 가져오려면 list-repositories 명령을 실행합니다.

- 새 브랜치의 이름(--branch-name 옵션 사용).
- 새 브랜치가 가리키는 커밋의 ID(--commit-id 옵션 사용).

예를 들어 `MyDemoRepo`라는 CodeCommit 리포지토리에서 커밋 ID `317f8570EXAMPLE`를 가리키는 `MyNewBranch`라는 브랜치를 생성하려면 다음을 수행합니다.

```
aws codecommit create-branch --repository-name MyDemoRepo --branch-name MyNewBranch
 --commit-id 317f8570EXAMPLE
```

이 명령은 오류가 있는 경우에만 출력을 생성합니다.

2. 로컬 리포지토리에서 사용 가능한 CodeCommit 리포지토리 브랜치의 목록을 새 원격 브랜치 이름으로 업데이트하려면 git remote update ***remote-name*** 명령을 실행합니다.

   예를 들어 `origin`라는 별명의 CodeCommit 리포지토리에 대하여 사용 가능한 브랜치의 목록을 업데이트하려면 다음을 수행합니다.

```
git remote update origin
```

> **Note**
>
> 또는 git fetch 명령을 실행할 수 있습니다. 또한 git branch --all 명령을 실행하여 모든 원격 브랜치를 볼 수도 있지만, 로컬 리포지토리의 목록을 업데이트할 때까지는 본인이 생성한 원격 브랜치가 목록에 표시되지 않습니다.
> 다른 옵션들에 대해서는 Git 설명서를 참조하세요.

3. CodeCommit 리포지토리의 브랜치에 연결된 로컬 리포지토리의 새 브랜치를 설정하려면 git checkout ***remote-branch-name***을 실행합니다.

> **Note**
>
> CodeCommit 리포지토리 이름의 목록과 그 URL을 가져오려면 git remote -v 명령을 실행합니다.

의 브랜치에 대한 푸시 및 병합 제한 AWS CodeCommit

기본적으로 리포지토리에 코드 푸시 권한을 충분히 갖고 있는 모든 CodeCommit 리포지토리 사용자는 그 리포지토리의 모든 브랜치에 기여할 수 있습니다. 콘솔을 사용하거나 명령줄이나 Git를 사용하는 등 어떤 방법으로 리포지토리에 브랜치를 추가하든 상관 없습니다. 단, 일부 리포지토리 사용자만 그 브랜치에 푸시 또는 병합할 수 있도록 브랜치를 구성해야 할 수 있습니다. 예를 들어 선임 개발자의 하위 집합만 그 브랜치에 변경 사항을 푸시 또는 병합할 수 있도록 프로덕션 코드에 사용하는 브랜치를 구성해야 할 수 있습니다. 다른 개발자도 여전히 이 브랜치에서 풀을 사용하고, 브랜치를 생성하며, 풀 요청을 생성할 수 있지만 변경 사항을 푸시하거나 병합할 수는 없습니다. IAM에서 하나 이상의 브랜치에 대하여 컨텍스트 키를 사용하는 조건 정책을 만들어 이 액세스를 구성할 수 있습니다.

> **Note**
>
> 이 주제의 일부 절차를 완료하려면 IAM 정책의 구성 및 적용 권한이 충분히 있는 관리자 사용자로 로그인해야 합니다. 자세한 내용은 IAM 관리자 및 그룹 만들기를 참조하세요.

주제

- 브랜치에 대한 푸시 및 병합을 제한하는 IAM 정책 구성
- IAM 그룹 또는 역할에 IAM 정책 적용
- 정책 테스트

브랜치에 대한 푸시 및 병합을 제한하는 IAM 정책 구성

브랜치에 커밋을 푸시하고 풀 요청을 병합하는 작업을 포함한 브랜치 업데이트 작업을 하지 못하도록 IAM에 정책을 만들 수 있습니다. 이를 위하여 조건에 해당하는 경우에만 Deny 문의 결과가 적용되도록 정책에서 조건문을 사용합니다. 어떤 작업을 허용하지 않는지는 Deny 문에 포함된 API가 결정합니다. Amazon Web Services 계정에서 리포지토리의 한 브랜치에만 적용하도록 하거나 모든 리포지토리에서 기준에 부합하는 모든 브랜치에 적용하도록 정책을 구성할 수 있습니다.

브랜치에 대한 조건 정책을 생성하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/ IAM 콘솔을 엽니다.
2. 탐색 창에서 Policies를 선택합니다.
3. 정책 생성을 선택합니다.
4. JSON을 선택하고 다음 정책 예시를 붙여 넣습니다. `Resource` 값을 액세스를 제한할 브랜치가 포함된 리포지토리의 ARN으로 대체합니다. `codecommit:References` 값을 액세스를 제한할 브랜치의 참조로 대체합니다. 예를 들어이 정책은 커밋 푸시, 브랜치 병합, 브랜치 삭제, 파일 삭제, 풀 요청 병합, 라는 브랜치 및 라는 리포지토리*prod*의 브*main*랜치에 파일 추가를 거부합니다. *MyDemoRepo*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "codecommit:GitPush",
                "codecommit:DeleteBranch",
                "codecommit:DeleteFile",
                "codecommit:PutFile",
                "codecommit:MergeBranchesByFastForward",
                "codecommit:MergeBranchesBySquash",
                "codecommit:MergeBranchesByThreeWay",
                "codecommit:MergePullRequestByFastForward",
                "codecommit:MergePullRequestBySquash",
                "codecommit:MergePullRequestByThreeWay"
            ],
            "Resource": "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
            "Condition": {
                "StringEqualsIfExists": {
                    "codecommit:References": [
                        "refs/heads/main",
                        "refs/heads/prod"
                     ]
                },
                "Null": {
                    "codecommit:References": "false"
                }
            }
```

```
        }
    ]
}
```

Git의 브랜치는 HEAD 커밋의 SHA-1 값에 대한 단순한 포인터(참조)인데, 이 때문에 이 조건은 `References`를 사용합니다. `Null` 문은 결과가 `Deny`이고 작업 중 하나가 `GitPush`인 모든 정책에 필요합니다. 이는 로컬 리포지토리에서 CodeCommit으로 변경 사항을 푸시할 때 Git과 `git-receive-pack`이 작업하는 방식 때문입니다.

> **Tip**
>
> Amazon Web Services 계정의 모든 리포지토리에서 main이라는 이름의 모든 브랜치에 적용되는 정책을 생성하려면, `Resource` 값을 리포지토리 ARN에서 별표(*)로 변경합니다.

5. 정책 검토를 선택합니다. 정책 문의 오류를 모두 수정하고 정책 생성을 계속합니다.
6. JSON이 검증되면 정책 생성 페이지가 표시됩니다. 요약 섹션의 경고는 이 정책이 권한을 허용하지 않음을 알려줍니다. 이는 예상된 동작입니다.
   - 이름에 **`DenyChangesToMain`** 등 이 정책의 이름을 입력합니다.
   - 설명에 정책의 목적에 대한 설명을 입력합니다. 이는 선택 사항이며, 권장 사항은 아닙니다.
   - 정책 생성을 선택합니다.

IAM 그룹 또는 역할에 IAM 정책 적용

지금까지 브랜치에 푸시 및 병합을 제한하는 정책을 만들었지만 IAM 사용자나 그룹, 역할에 적용하기 전까지 이 정책은 아무런 결과가 없습니다. 모범 사례로서 IAM 그룹 또는 역할에 정책을 적용해 보세요. 정책을 개별 IAM 사용자에게 적용해도 잘 확장되지 않습니다.

조건 정책을 그룹 또는 역할에 적용하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/ IAM 콘솔을 엽니다.
2. 정책을 IAM 그룹에 적용하려면 탐색 창에서 그룹을 선택합니다. 정책을 사용자가 수임한 역할에 적용하려면 역할을 선택합니다. 그룹 또는 역할의 이름을 선택합니다.
3. 권한 탭에서 정책 연결을 선택합니다.
4. 정책 목록에서 만들어 둔 조건 정책을 선택한 다음 정책 연결을 선택합니다.

자세한 내용은 IAM 정책 연결 및 분리를 참조하세요.

정책 테스트

그룹 또는 역할에 적용한 정책의 결과를 테스트하여 예상대로 작동하는지 확인해야 합니다. 방법은 다양합니다. 예를 들어 위에 보이는 것과 유사한 정책을 테스트하려면 다음과 같이 합니다.

- 정책을 적용한 IAM 그룹의 멤버이거나 그 정책이 적용된 역할을 수임한 IAM 사용자로 CodeCommit 콘솔에 로그인합니다. 콘솔에서 제한이 적용된 브랜치에 파일을 추가합니다. 이 브랜치에 파일을 저장하려고 하거나 업로드하려고 시도하면 오류 메시지가 떠야 합니다. 다른 브랜치에 파일을 추가합니다. 작업이 이제 성공합니다.
- 정책을 적용한 IAM 그룹의 멤버이거나 그 정책이 적용된 역할을 수임한 IAM 사용자로 CodeCommit 콘솔에 로그인합니다. 제한이 적용되는 브랜치에 병합할 풀 요청을 생성합니다. 풀 요청을 만들 수는 있지만 병합을 시도하면 오류가 떠야 합니다.
- 터미널 또는 명령줄에서 제한이 적용되는 브랜치에 커밋을 생성한 후 CodeCommit 리포지토리에 그 커밋을 푸시합니다. 오류 메시지가 떠야 합니다. 다른 브랜치에서 작업한 커밋과 푸시는 정상으로 작동해야 합니다.

에서 브랜치 세부 정보 보기 AWS CodeCommit

CodeCommit 콘솔을 사용하여 CodeCommit 리포지토리의 브랜치에 대한 세부 정보를 볼 수 있습니다. 브랜치에 마지막으로 커밋한 날짜, 커밋 메시지 등을 볼 수 있습니다. CodeCommit 리포지토리에 연결된 로컬 리포지토리에서 AWS CLI 또는 Git을 사용할 수도 있습니다.

주제

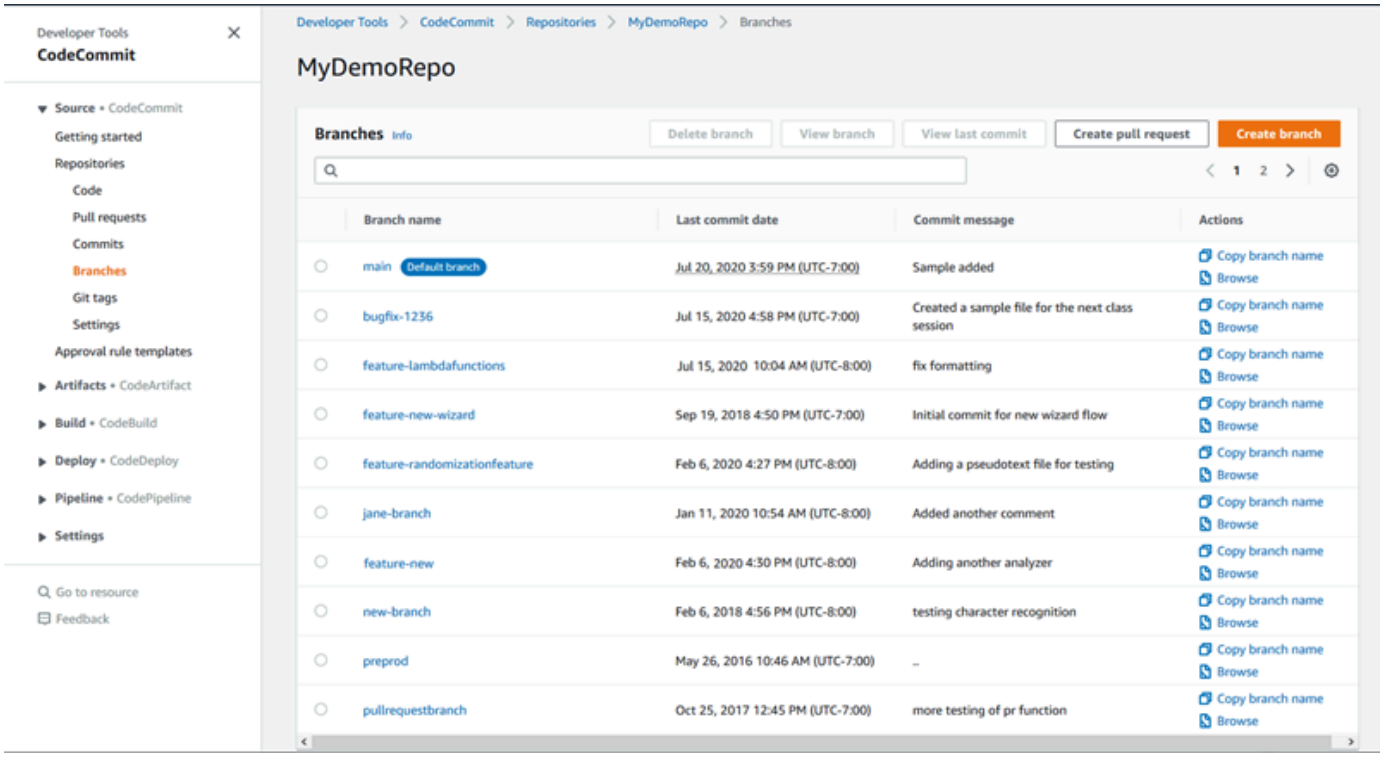
- 브랜치 세부 정보 보기 (콘솔)
- 브랜치 세부 정보 보기 (Git)
- 브랜치 세부 정보 보기 (AWS CLI)

브랜치 세부 정보 보기 (콘솔)

CodeCommit 콘솔을 사용하여 리포지토리의 브랜치 목록과 브랜치에 대한 세부 정보를 빠르게 볼 수 있습니다.

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서, 브랜치 세부 정보를 보려는 리포지토리의 이름을 선택합니다.

3. 탐색 창에서 브랜치를 선택합니다.



4. 리포지토리의 기본값으로 사용되는 브랜치의 이름은 기본 브랜치 옆에 표시됩니다. 브랜치에 대한 가장 최근 커밋에 대한 세부 정보를 보려면, 브랜치를 선택한 다음 마지막 커밋 보기를 선택합니다. 브랜치의 파일과 코드를 보려면 브랜치 이름을 선택합니다.

브랜치 세부 정보 보기 (Git)

로컬 리포지토리에서 Git을 사용하여 CodeCommit 리포지토리의 로컬 및 원격 추적 브랜치에 대한 세부 정보를 보려면 git branch 명령을 실행합니다.

다음 단계들은 로컬 리포지토리를 이미 CodeCommit 리포지토리에 연결했다는 가정하에 작성되었습니다. 지침은 리포지토리에 연결 섹션을 참조하세요.

1. --all 옵션을 지정하여 git branch 명령을 실행합니다.

```
git branch --all
```

2. 이 명령이 제대로 실행되면 다음과 비슷한 출력이 반환됩니다.

```
  MyNewBranch
* main
  remotes/origin/MyNewBranch
```

```
  remotes/origin/main
```

별표(*)는 현재 설정된 브랜치 옆에 표시됩니다. 그 이후의 항목은 원격 추적 참조입니다.

> **Tip**
>
> git branch는 로컬 브랜치를 보여 줍니다.
> git branch -r은 원격 브랜치를 보여 줍니다.
> git checkout ***existing-branch-name***은 지정된 브랜치 이름으로 전환되며, 그 후에 git branch가 즉시 실행되면 그것을 별표(*)로 표시합니다.
> git remote update ***remote-name***는 사용 가능한 CodeCommit 리포지토리 브랜치 목록으로 로컬 리포지토리를 업데이트합니다. (CodeCommit 리포지토리 이름의 목록과 그 URL을 가져오려면 git remote -v 명령을 실행합니다.)

다른 옵션들에 대해서는 Git 설명서를 참조하세요.

브랜치 세부 정보 보기 (AWS CLI)

CodeCommit에서 AWS CLI 명령을 사용하려면를 설치합니다 AWS CLI. 자세한 내용은 명령줄 참조 단원을 참조하십시오.

AWS CLI 를 사용하여 CodeCommit 리포지토리의 브랜치에 대한 세부 정보를 보려면 다음 명령 중 하나 이상을 실행합니다.

- 브랜치 이름 목록을 보려면 list-branches를 실행합니다.
- 특정 브랜치에 대한 정보를 보려면 get-branch를 실행합니다.

브랜치 이름 목록을 보려면

1. CodeCommit 리포지토리의 이름을 지정하여 list-branches 명령을 실행합니다(`--repository-name` 옵션 사용).

> **Tip**
>
> CodeCommit 리포지토리의 이름을 가져오려면 list-repositories 명령을 실행합니다.

예를 들어, MyDemoRepo라는 CodeCommit 리포지토리의 브랜치에 대한 세부 정보를 보려면 다음과 같이 합니다.

```
aws codecommit list-branches --repository-name MyDemoRepo
```

2. 성공하면 이 명령은 각 브랜치에 대한 항목이 있는 branchNameList 객체를 출력합니다.

   다음은 위 예제 명령의 출력 예입니다.

```
{
    "branches": [
        "MyNewBranch",
        "main"
    ]
}
```

커밋에 대한 정보를 보려면

1. 다음을 지정하여 get-branch 명령을 실행합니다.

   - 리포지토리 이름(--repository-name 옵션 사용).
   - 브랜치 이름(--branch-name 옵션 사용).

   예를 들어, MyDemoRepo라는 CodeCommit 리포지토리의 MyNewBranch라는 브랜치에 대한 정보를 보려면 다음과 같이 합니다.

```
aws codecommit get-branch --repository-name MyDemoRepo --branch-name MyNewBranch
```

2. 성공하면 이 명령은 브랜치의 이름과 이 브랜치에 대한 마지막 커밋의 ID를 출력합니다.

   다음은 위 예제 명령의 출력 예입니다.

```
{
    "branch": {
          "branchName": "MyNewBranch",
          "commitID": "317f8570EXAMPLE"
    }
```

```
}
```

AWS CodeCommit에서 브랜치 비교 및 병합

CodeCommit 콘솔을 사용하여 CodeCommit 리포지토리의 브랜치를 비교할 수 있습니다. 브랜치를 비교하면 브랜치와 기본 브랜치 간의 차이 또는 두 브랜치 간의 차이를 빠르게 확인할 수 있습니다.

주제

- 임의의 브랜치와 기본 브랜치 간의 비교
- 두 특정 브랜치 비교
- 두 브랜치 병합 (AWS CLI)

임의의 브랜치와 기본 브랜치 간의 비교

CodeCommit 콘솔을 사용하여 리포지토리의 기본 브랜치와 다른 브랜치 간의 차이를 빠르게 확인할 수 있습니다.

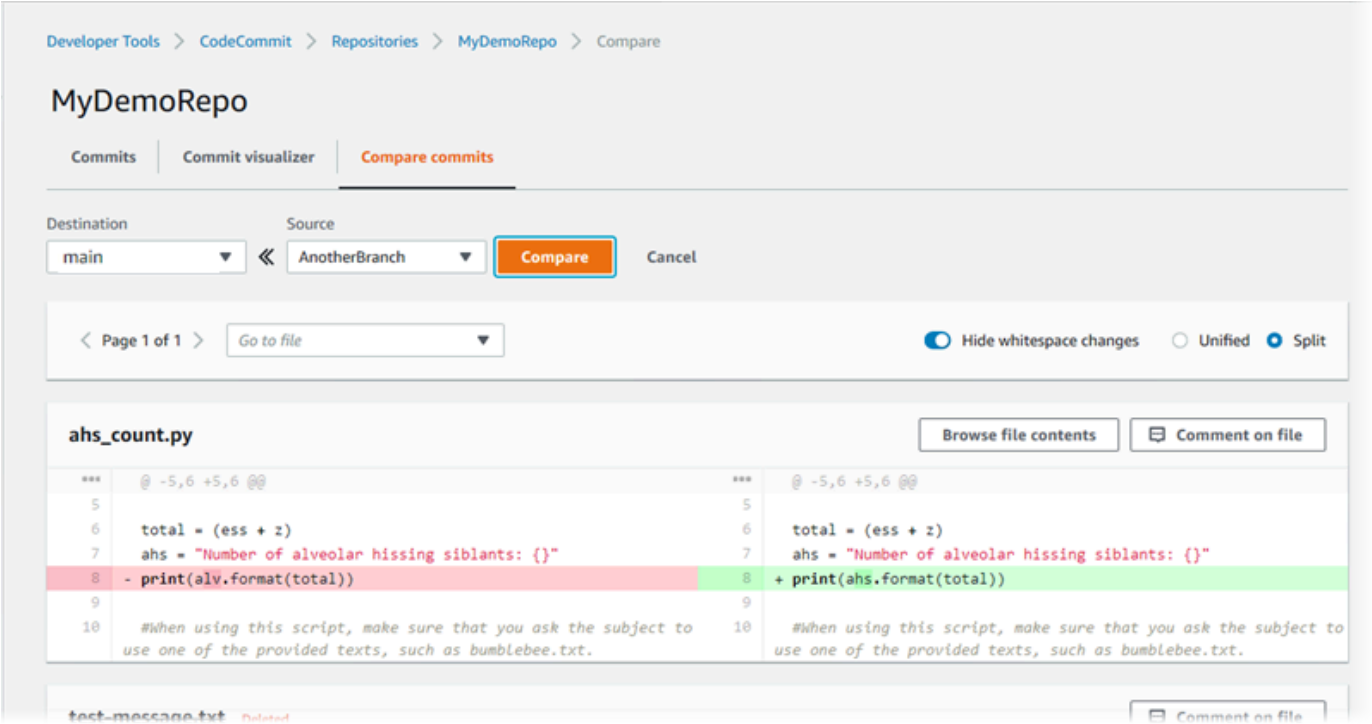
1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서, 브랜치를 비교하려는 리포지토리의 이름을 선택합니다.
3. 탐색 창에서 커밋을 선택한 다음 커밋 비교 탭을 선택합니다.
4. 대상 주소에서 기본 브랜치의 이름을 선택합니다. 소스에서 기본 브랜치와 비교할 브랜치를 선택합니다. 비교를 선택합니다.

두 특정 브랜치 비교

CodeCommit 콘솔을 사용하여 비교할 두 브랜치 간의 차이를 볼 수 있습니다.

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서, 브랜치를 비교하려는 리포지토리의 이름을 선택합니다.
3. 탐색 창에서 커밋을 선택한 다음 커밋 비교 탭을 선택합니다.
4. 대상 주소와 소스에서 비교할 두 브랜치를 선택한 다음 비교를 선택합니다. 구성된 파일 목록을 보려면 변경된 파일 목록을 확장합니다. 변경 내용은 나란히(분할 보기) 또는 일렬(통합 보기)로 표시할 수 있습니다.

> **Note**
>
> IAM 사용자로 로그인한 경우 코드 및 다른 콘솔 설정 보기에 대한 기본 설정을 구성하고 저장할 수 있습니다. 자세한 내용은 사용자 기본 설정으로 작업 단원을 참조하십시오.



두 브랜치 병합 (AWS CLI)

다음 명령 중 하나를 실행하여 사용 가능한 병합 전략 중 하나를 AWS CLI 사용하여 CodeCommit 리포지토리의 두 브랜치를 병합할 수 있습니다.

- 패스트 포워드 병합 전략을 사용하여 두 브랜치를 병합하려면 merge-branches-by-fast-forward 명령을 실행합니다.
- 스쿼시 병합 전략을 사용하여 두 브랜치를 병합하려면 merge-branches-by-squash 명령을 실행합니다.
- 3방향 병합 전략을 사용하여 두 브랜치를 병합하려면 merge-branches-by-three-way 명령을 실행합니다.

create-unreferenced-merge-commit 명령을 실행하여 병합을 테스트할 수도 있습니다. 자세한 내용은 풀 요청의 충돌 해결을 참조하세요.

> **Note**
>
> CodeCommit에서 AWS CLI 명령을 사용하려면를 설치합니다 AWS CLI. 자세한 내용은 명령줄 참조 단원을 참조하십시오.

AWS CLI 를 사용하여 CodeCommit 리포지토리의 두 브랜치를 병합하려면

1. 패스트 포워드 병합 전략을 사용하여 두 브랜치를 병합하려면 merge-branches-by-fast-forward 명령을 실행하고 다음을 지정합니다.

   - 병합할 변경 사항을 포함하는 소스 브랜치의 이름(--source-commit-specifier 옵션 사용).
   - 변경 사항을 병합할 대상 브랜치의 이름(--destination-commit-specifier 옵션 사용).
   - 리포지토리의 이름(--repository-name 옵션 사용).

   예를 들어, *bugfix-1234*라는 소스 브랜치를 *MyDemoRepo*라는 리포지토리의 *preprod*라는 대상 브랜치로 통합하려면 다음과 같이 합니다.

   ```
   aws codecommit merge-branches-by-fast-forward --source-commit-specifier bugfix-
   bug1234 --destination-commit-specifier preprod --repository-name MyDemoRepo
   ```

   이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

   ```
   {
       "commitId": "4f178133EXAMPLE",
       "treeId": "389765daEXAMPLE"
   }
   ```

2. 스쿼시 병합 전략을 사용하여 두 브랜치를 병합하려면 merge-branches-by-squash 명령을 실행하고 다음을 지정합니다.

   - 병합할 변경 사항을 포함하는 소스 브랜치의 이름(--source-commit-specifier 옵션 사용).
   - 변경 사항을 병합할 대상 브랜치의 이름(--destination-commit-specifier 옵션 사용).
   - 리포지토리의 이름(--repository-name 옵션 사용).

- 포함시킬 커밋 메시지(--commit-message 옵션 사용).
- 커밋에 사용할 이름(--name 옵션 사용).
- 커밋에 사용할 이메일 주소(--email 옵션 사용).

예를 들어, *bugfix-bug1234*라는 소스 브랜치를 *MyDemoRepo*라는 리포지토리의 *bugfix-quarterly*라는 대상 브랜치로 통합하려면 다음과 같이 합니다.

```
aws codecommit merge-branches-by-squash --source-commit-specifier bugfix-bug1234 --
destination-commit-specifier bugfix-quarterly --author-name "Maria Garcia" --email
 "maria_garcia@example.com" --commit-message "Merging in fix branches to prepare
 for a general patch." --repository-name MyDemoRepo
```

이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
{
    "commitId": "4f178133EXAMPLE",
    "treeId": "389765daEXAMPLE"
}
```

3. 3방향 병합 전략을 사용하여 두 브랜치를 병합하려면 merge-branches-by-three-way 명령을 실행하고 다음을 지정합니다.

- 병합할 변경 사항을 포함하는 소스 브랜치의 이름(--source-commit-specifier 옵션 사용).
- 변경 사항을 병합할 대상 브랜치의 이름(--destination-commit-specifier 옵션 사용).
- 리포지토리의 이름(--repository-name 옵션 사용).
- 포함시킬 커밋 메시지(--commit-message 옵션 사용).
- 커밋에 사용할 이름(--name 옵션 사용).
- 커밋에 사용할 이메일 주소(--email 옵션 사용).

예를 들어, *main*이라는 소스 브랜치를 *MyDemoRepo*라는 리포지토리의 *bugfix-1234*라는 대상 브랜치와 통합하려면 다음과 같이 합니다.

```
aws codecommit merge-branches-by-three-way --source-commit-specifier main --
destination-commit-specifier bugfix-bug1234 --author-name "Jorge Souza" --email
 "jorge_souza@example.com" --commit-message "Merging changes from main to bugfix
 branch before additional testing."  --repository-name MyDemoRepo
```

이 명령이 제대로 실행되면 다음과 비슷하게 출력됩니다.

```
{
    "commitId": "4f178133EXAMPLE",
    "treeId": "389765daEXAMPLE"
}
```

에서 브랜치 설정 변경 AWS CodeCommit

AWS CodeCommit 콘솔 또는에서 기본 브랜치로 사용할 브랜치를 변경할 수 있습니다 AWS CLI. 예를 들어, 기본 브랜치를 master로 설정하는 Git 클라이언트를 사용하여 첫 번째 커밋을 만든 경우, main이라는 브랜치를 생성한 다음, 새 브랜치가 리포지토리의 기본 브랜치로 설정되도록 브랜치 설정을 변경할 수 있습니다. 다른 브랜치 설정을 변경하려면 CodeCommit 리포지토리에 연결된 로컬 리포지토리의 Git을 사용할 수 있습니다.

주제

- 기본 브랜치 변경 (콘솔)
- 기본 브랜치 변경 (AWS CLI)

기본 브랜치 변경 (콘솔)

AWS CodeCommit 콘솔의 CodeCommit 리포지토리에서 기본 브랜치로 지정할 수 있습니다.

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서, 설정을 변경하려는 리포지토리의 이름을 선택합니다.
3. 탐색 창에서 설정을 선택합니다.
4. 기본 브랜치에서 브랜치 드롭다운 목록을 선택하고 다른 브랜치를 선택합니다. 저장을 선택합니다.

> **Tip**
>
> - 드롭다운 목록에 다른 브랜치가 보이지 않는다면, 브랜치를 추가로 만들지 않은 것입니다. 리포지토리에 브랜치가 하나뿐인 경우에는 리포지토리의 기본 브랜치를 변경할 수 없습니다. 자세한 내용은 에서 브랜치 생성 AWS CodeCommit 단원을 참조하십시오.

- 기본 브랜치 섹션 대신 알림 규칙 및 연결 항목이 표시된다면, 현재 콘솔의 일반 설정 메뉴에 있는 것입니다. 리포지토리의 설정 메뉴는 코드 및 풀 요청과 동일한 수준에 있는 리포지토리의 밑에 나열됩니다.

기본 브랜치 변경 (AWS CLI)

CodeCommit에서 AWS CLI 명령을 사용하려면를 설치합니다 AWS CLI. 자세한 내용은 명령줄 참조 단원을 참조하십시오.

AWS CLI 를 사용하여 CodeCommit 리포지토리에서 리포지토리의 브랜치 설정을 변경하려면 다음 명령을 실행합니다.

- update-default-branch - 기본 브랜치를 변경합니다.

기본 브랜치를 변경하려면

1. 다음을 지정하여 update-default-branch 명령을 실행합니다.

   - 기본 브랜치가 업데이트되는 CodeCommit 리포지토리의 이름(--repository-name 옵션 사용).

     > **Tip**
     > CodeCommit 리포지토리의 이름을 가져오려면 list-repositories 명령을 실행합니다.

   - 새 기본 브랜치의 이름(--default-branch-name 옵션 사용).

     > **Tip**
     > 브랜치 이름을 가져오려면 list-branches 명령을 실행합니다.

2. 예를 들어, `MyDemoRepo`라는 CodeCommit 리포지토리에서 기본 브랜치를 `MyNewBranch`로 변경하려면 다음과 같이 합니다.

   ```
   aws codecommit update-default-branch --repository-name MyDemoRepo --default-branch-name MyNewBranch
   ```

   이 명령은 오류가 있는 경우에만 출력을 생성합니다.

다른 옵션들에 대해서는 Git 설명서를 참조하세요.

에서 브랜치 삭제 AWS CodeCommit

CodeCommit 콘솔을 사용하여 리포지토리의 브랜치를 삭제할 수 있습니다. CodeCommit에서 브랜치를 삭제해도 로컬 리포지토리에서 해당 브랜치가 삭제되지 않으므로, 사용자는 다음에 변경 사항을 풀할 때까지 해당 브랜치의 복사본을 계속 보유할 수 있습니다. 로컬에서 브랜치를 삭제하고 변경 사항을 CodeCommit 리포지토리로 푸시하려면, CodeCommit 리포지토리에 연결된 로컬 리포지토리에서 Git을 사용합니다.

브랜치를 삭제해도 커밋은 삭제되지 않지만, 해당 브랜치의 커밋에 대한 참조는 모두 삭제됩니다. 리포지토리의 다른 브랜치에 병합되지 않은 커밋이 들어 있는 브랜치를 삭제할 경우, 전체 커밋 ID가 없으면 해당 커밋을 검색할 수 없습니다.

> **Note**
>
> 이 주제의 지침을 사용하여 리포지토리의 기본 브랜치를 삭제할 수는 없습니다. 기본 브랜치를 삭제하려면, 브랜치를 하나 만들고 이 새 브랜치를 기본 브랜치로 만든 다음 이전 브랜치를 삭제해야 합니다. 자세한 내용은 브랜치 생성 및 브랜치 설정 변경 섹션을 참조하세요.

주제

- 브랜치 삭제 (콘솔)
- 브랜치 삭제 (AWS CLI)
- 브랜치 삭제 (Git)

브랜치 삭제 (콘솔)

CodeCommit 콘솔을 사용하여 CodeCommit 리포지토리에서 브랜치를 삭제할 수 있습니다.

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서, 브랜치를 삭제하려는 리포지토리의 이름을 선택합니다.
3. 탐색 창에서 브랜치를 선택합니다.
4. 삭제하려는 브랜치의 이름을 찾아 브랜치 삭제를 선택한 다음 선택을 확인합니다.

브랜치 삭제 (AWS CLI)

브랜치가 AWS CLI 리포지토리의 기본 브랜치가 아닌 경우를 사용하여 CodeCommit 리포지토리에서 브랜치를 삭제할 수 있습니다. 설치 및 사용에 대한 자세한 내용은 섹션을 AWS CLI참조하세요명령줄 참조.

1. 터미널 또는 명령줄에서 다음을 지정하여 delete-branch 명령을 실행합니다.

   - 브랜치를 삭제할 CodeCommit 리포지토리의 이름(--repository-name 옵션 사용).

     > **Tip**
     > CodeCommit 리포지토리의 이름을 가져오려면 list-repositories 명령을 실행합니다.

   - 삭제할 브랜치의 이름(branch-name 옵션 사용).

     > **Tip**
     > 브랜치 이름을 가져오려면 list-branches 명령을 실행합니다.

2. 예를 들어, `MyDemoRepo`라는 CodeCommit 리포지토리에서 `MyNewBranch`라는 브랜치를 삭제하려면 다음과 같이 합니다.

   ```
   aws codecommit delete-branch --repository-name MyDemoRepo --branch-name MyNewBranch
   ```

   이 명령은 삭제된 브랜치의 이름과 브랜치의 헤드였던 커밋의 전체 커밋 ID를 포함하여 삭제된 브랜치에 대한 정보를 반환합니다. 예시:

   ```
   "deletedBranch": {
       "branchName": "MyNewBranch",
       "commitId": "317f8570EXAMPLE"
   }
   ```

브랜치 삭제 (Git)

다음 단계에 따라 로컬 리포지토리에서 Git을 사용하여 CodeCommit 리포지토리의 브랜치를 삭제합니다.

이 단계들은 로컬 리포지토리를 이미 CodeCommit 리포지토리에 연결했다는 가정하에 작성되었습니다. 지침은 리포지토리에 연결 섹션을 참조하세요.

1. 로컬 리포지토리에서 브랜치를 삭제하려면 git branch -D ***branch-name*** 명령을 실행합니다. 여기서 *branch-name*은 삭제할 브랜치의 이름입니다.

   > **Tip**
   > 브랜치 이름 목록을 가져오려면 git branch --all 명령을 실행합니다.

   예를 들어, `MyNewBranch`라는 로컬 리포지토리에서 브랜치를 삭제하려면 다음을 실행합니다.

   ```
   git branch -D MyNewBranch
   ```

2. CodeCommit 리포지토리에서 브랜치를 삭제하려면 git push ***remote-name*** --delete ***branch-name*** 명령을 실행합니다. 여기서 *remote-name*은 로컬 리포지토리가 CodeCommit 리포지토리에 사용하는 별명이고, *branch-name*은 CodeCommit 리포지토리에서 삭제할 브랜치의 이름입니다.

   > **Tip**
   > CodeCommit 리포지토리 이름의 목록과 그 URL을 가져오려면 git remote -v 명령을 실행합니다.

   예를 들어, `origin`라는 CodeCommit 리포지토리에서 `MyNewBranch`라는 브랜치를 삭제하려면 다음과 같이 합니다.

   ```
   git push origin --delete MyNewBranch
   ```

   > **Tip**
   > 이 명령은 삭제하려는 브랜치가 기본 브랜치인 경우 그 브랜치를 삭제하지 않습니다.

다른 옵션들에 대해서는 Git 설명서를 참조하세요.

사용자 기본 설정으로 작업

AWS CodeCommit 콘솔을 사용하여 일부 기본 설정을 구성할 수 있습니다. 예를 들어 코드 변경 내용을 인라인 또는 분할 보기로 볼 수 있도록 기본 설정을 변경할 수 있습니다. 이러한 설정 중 하나를 변경하면 AWS CodeCommit 콘솔은 브라우저에 콘솔을 사용할 때마다 선택 사항을 저장하고 적용하는 쿠키를 설정합니다. 이러한 기본 설정은 해당 브라우저를 사용하여 AWS CodeCommit 콘솔에 액세스할 때마다 모든 리전의 모든 리포지토리에 적용됩니다. 이러한 기본 설정은 리포지토리 또는 리전별로 국한되지 않습니다. 이는 AWS CLI, AWS CodeCommit API 또는와 상호 작용하는 기타 서비스와의 상호 작용에 영향을 미치지 않습니다 AWS CodeCommit.

> **Note**
>
> 사용자 기본 설정 쿠키는 브라우저별로 국한됩니다. 브라우저에서 쿠키를 지우면 기본 설정이 지워집니다. 마찬가지로 다른 브라우저를 사용하여 리포지토리에 액세스하는 경우, 해당 브라우저는 다른 브라우저의 쿠키에 액세스할 수 없습니다. 기본 설정은 유지되지 않습니다.

사용자 기본 설정은 다음과 같습니다.

- 코드에서 변경 내용을 볼 때 [Unified]/[Split] 보기 사용 여부와 공백 변경 내용의 표시 여부.
- 코드를 보기, 편집 또는 작성할 때, 코드 편집기 창에서 밝은 배경을 사용할지 어두운 배경을 사용할지 선택해야 합니다.

기본 설정 전용의 단일한 페이지는 없습니다. 그 대신, 코드 보기 방식 변경 등 기본 설정을 콘솔의 어디에서 변경하든 변경 내용이 저장되고 해당되는 모든 영역에 적용됩니다.

로 마이그레이션 AWS CodeCommit

Git 리포지토리를 CodeCommit 리포지토리로 마이그레이션하는 방법에는 여러 가지가 있습니다. 복제, 미러링, 브랜치의 전체 또는 일부를 마이그레이션하기 등 다양한 방법을 사용할 수 있습니다. 또한 컴퓨터에 있는 버전 관리 미사용의 로컬 콘텐츠를 CodeCommit으로 마이그레이션할 수 있습니다.

다음 주제에서는 리포지토리를 마이그레이션할 수 있는 몇 가지 방법을 보여 줍니다. 각 단계는 리포지토리의 유형, 스타일, 복잡성 그리고 마이그레이션할 대상 및 방법에 대한 결정 사항 등에 따라 달라질 수 있습니다. 매우 큰 리포지토리의 경우, 점진적 마이그레이션을 고려하는 것이 좋을 수도 있습니다.

> **Note**
>
> Perforce, Subversion, TFS 등과 같은 기타 버전 관리 시스템에서 CodeCommit으로 마이그레이션할 수 있지만, 먼저 Git으로 마이그레이션해야 합니다.
> 다른 옵션들에 대해서는 Git 설명서를 참조하세요.
> 또는 Scott Chacon과 Ben Straub이 쓴 Pro Git 책에서, Git으로 마이그레이션하기에 대한 정보를 검토할 수도 있습니다.

주제

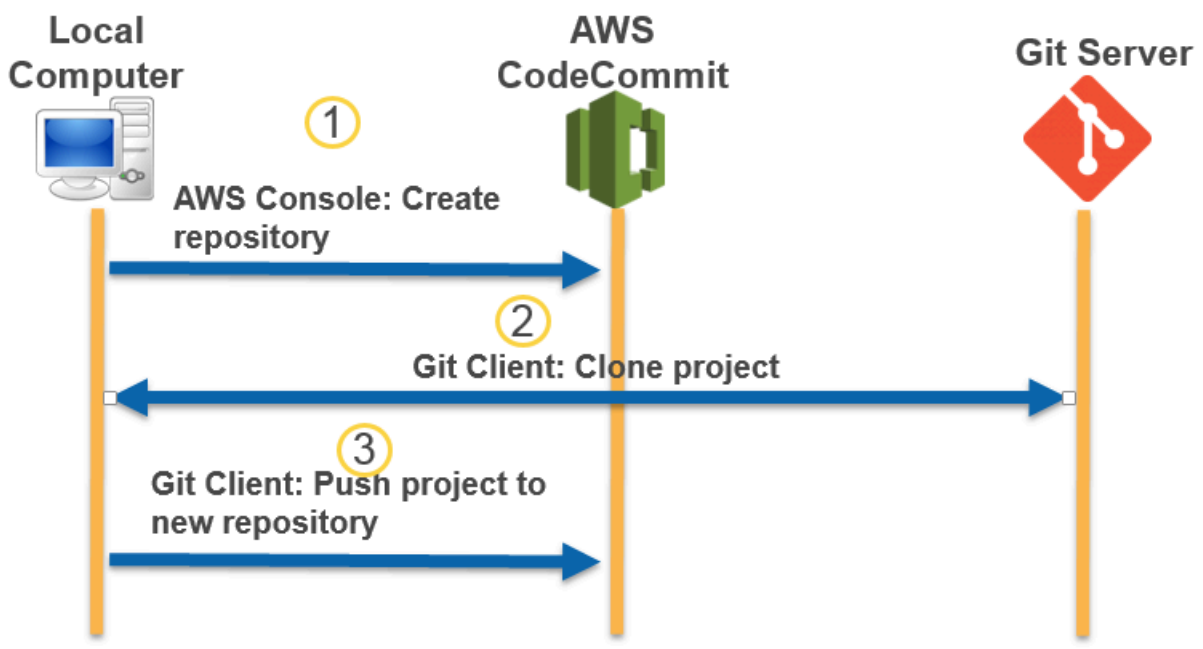
- Git 리포지토리를 로 마이그레이션 AWS CodeCommit
- 로컬 또는 버전이 지정되지 않은 콘텐츠를 로 마이그레이션 AWS CodeCommit
- 리포지토리를 증분으로 마이그레이션하기

Git 리포지토리를 로 마이그레이션 AWS CodeCommit

기존 Git 리포지토리를 CodeCommit 리포지토리로 마이그레이션할 수 있습니다. 이 주제에서는 또 다른 Git 리포지토리에 호스팅된 프로젝트를 CodeCommit으로 마이그레이션하는 절차를 안내합니다. 이 절차의 일부로써 다음을 수행합니다.

- CodeCommit에 필요한 초기 설정 완료.
- CodeCommit 리포지토리 생성.
- 리포지토리를 복제하여 CodeCommit으로 푸시.
- CodeCommit 리포지토리에서 파일 보기.

- 소속 팀과 CodeCommit 리포지토리 공유.



주제

- 0단계: CodeCommit 액세스에 필요한 설정
- 1단계: CodeCommit 리포지토리 생성
- 2단계: 리포지토리를 복제하여 CodeCommit 리포지토리로 푸시
- 3단계: CodeCommit에서 파일 보기
- 4단계: CodeCommit 리포지토리 공유

0단계: CodeCommit 액세스에 필요한 설정

리포지토리를 CodeCommit으로 마이그레이션하기 전에, 먼저 CodeCommit에 IAM 사용자를 생성 및 구성하고 액세스를 위해 로컬 컴퓨터를 구성해야 합니다. 또한 CodeCommit을 관리하기 위해 AWS CLI 를 설치해야 합니다. AWS CLI 가 없어도 대부분의 CodeCommit 작업을 수행할 수 있지만, 만약 있으면 명령줄 또는 터미널에서 Git으로 작업할 때 융통성을 발휘할 수 있습니다.

CodeCommit에 대한 설정을 이미 마친 경우에는 1단계: CodeCommit 리포지토리 생성로 건너뛸 수 있습니다.

CodeCommit에 액세스할 IAM 사용자를 생성 및 구성하려면

1. http://aws.amazon.com에서 가입을 선택하여 Amazon Web Services 계정을 만듭니다.
2. Amazon Web Services 계정에서 IAM 사용자를 생성하거나 기존 사용자를 사용합니다. 액세스 키 ID가 있고 해당 IAM 사용자와 연결된 비밀 액세스 키가 있는지 확인합니다. 자세한 내용은 Amazon Web Services 계정에서 IAM 사용자 생성하기를 참조하세요.

   > **Note**
   >
   > CodeCommit에는가 필요합니다 AWS Key Management Service. 기존 IAM 사용자를 사용하는 경우 CodeCommit에 필요한 AWS KMS 작업을 명시적으로 거부하는 정책이 사용자에게 연결되어 있지 않은지 확인합니다. 자세한 내용은 AWS KMS 및 암호화 단원을 참조하십시오.

3. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/://https://https://https://https://://https://https://://https://://https://://https://://https://://http
4. IAM 콘솔의 탐색 창에서 사용자를 선택한 다음 CodeCommit 액세스에 대해 구성할 IAM 사용자를 선택합니다.
5. 권한 탭에서 권한 추가를 선택합니다.
6. 권한 부여에서 기존 정책 직접 연결을 선택합니다.
7. 정책 목록에서, CodeCommit 액세스에 대한 AWSCodeCommitPowerUser 또는 다른 관리형 정책을 선택합니다. 자세한 내용은 AWS CodeCommit에 대한 관리형 정책 단원을 참조하십시오.

   연결할 정책을 선택한 후 다음: 검토를 선택하여 IAM 사용자에게 연결할 정책의 목록을 검토합니다. 목록이 올바르면 권한 추가를 선택합니다.

   CodeCommit 관리형 정책, 그리고 리포지토리에 대한 액세스를 다른 그룹 및 사용자와 공유하는 방법에 대해 자세히 알아보려면 리포지토리 공유, AWS CodeCommit에 대한 인증 및 액세스 제어 섹션을 참조하세요.

를 설치하고 구성하려면 AWS CLI

1. 로컬 시스템에서를 다운로드하여 설치합니다 AWS CLI. 이는 명령줄에서 CodeCommit과 상호 작용하기 위한 사전 조건입니다. AWS CLI 버전 2를 설치하는 것이 좋습니다. 의 최신 메이저 버전이며 모든 최신 기능을 AWS CLI 지원합니다. 에서 루트 계정, 페더레이션 액세스 또는 임시 자격 증명 사용을 AWS CLI 지원하는 유일한 버전의 입니다git-remote-codecommit.

자세한 내용은 AWS 명령줄 인터페이스 설정 단원을 참조하십시오.

> **Note**
>
> CodeCommit은 AWS CLI 버전 1.7.38 이상에서만 작동합니다. 가장 좋은 방법은를 설치하거나 사용 가능한 최신 버전으로 업그레이드 AWS CLI 하는 것입니다. 설치 AWS CLI 한 버전을 확인하려면 aws --version 명령을 실행합니다.
> 이전 버전의 AWS CLI 를 최신 버전으로 업그레이드하려면 설치를 참조하세요 AWS Command Line Interface.

2. 이 명령을 실행하여에 대한 CodeCommit 명령 AWS CLI 이 설치되어 있는지 확인합니다.

```
aws codecommit help
```

이 명령은 CodeCommit 명령 목록을 반환합니다.

3. 다음과 같이 configure 명령을 사용하여 프로파일 AWS CLI 로를 구성합니다.

```
aws configure
```

메시지가 표시되면 CodeCommit과 함께 사용할 IAM 사용자의 AWS 액세스 키와 AWS 보안 액세스 키를 지정합니다. 또한와 같이 리포지토리 AWS 리전 가 있는를 지정해야 합니다`us-east-2`. 기본 출력 형식을 묻는 메시지가 표시되면 `json`을 지정합니다. 예를 들어, IAM 사용자의 프로필을 구성하는 경우에는 다음과 같이 합니다.

```
AWS Access Key ID [None]: Type your IAM user AWS access key ID here, and then press
 Enter
AWS Secret Access Key [None]: Type your IAM user AWS secret access key here, and
 then press Enter
Default region name [None]: Type a supported region for CodeCommit here, and then
 press Enter
Default output format [None]: Type json here, and then press Enter
```

에서 사용할 프로필을 생성하고 구성하는 방법에 대한 자세한 내용은 다음을 AWS CLI참조하세요.

- 명명된 프로파일
- 에서 IAM 역할 사용 AWS CLI

- 설정 명령
- 교체 자격 증명을 사용하여 AWS CodeCommit 리포지토리에 연결

리포지토리 또는 다른의 리소스에 연결하려면 기본 리전 이름으로 AWS CLI 를 재구성 AWS 리전 해야 합니다. CodeCommit에서 지원되는 기본 리전 이름은 다음과 같습니다.

- us-east-2
- us-east-1
- eu-west-1
- us-west-2
- ap-northeast-1
- ap-southeast-1
- ap-southeast-2
- ap-southeast-3
- me-central-1
- eu-central-1
- ap-northeast-2
- sa-east-1
- us-west-1
- eu-west-2
- ap-south-1
- ap-south-1
- ca-central-1
- us-gov-west-1
- us-gov-east-1
- eu-north-1
- ap-east-1
- me-south-1
- cn-north-1
- cn-northwest-1

- eu-south-1

- ap-northeast-3
- af-south-1
- il-central-1

CodeCommit 및에 대한 자세한 내용은 섹션을 AWS 리전참조하세요리전 및 Git 연결 엔드포인트. IAM, 액세스 키, 비밀 키에 대한 자세한 내용을 확인하려면 보안 인증 정보 얻는 방법 및 IAM 사용자의 액세스 키 관리를 참조하세요. AWS CLI 및 프로파일에 대한 자세한 내용은 명명된 프로파일을 참조하세요.

그 다음에는 Git을 설치해야 합니다.

- Linux, macOS, Unix의 경우:

  CodeCommit 리포지토리에서 파일과 커밋 및 기타 정보를 사용하려면 로컬 시스템에 Git을 설치해야 합니다. CodeCommit은 Git 버전 1.7.9 이상을 지원합니다. Git 버전 2.28에서는 초기 커밋을 위한 브랜드 이름을 구성할 수 있습니다. 최신 버전의 Git를 사용하는 것이 좋습니다.

  Git 설치를 위해서는 Git 다운로드와 같은 웹 사이트를 권장합니다.

  > **Note**
  > Git은 계속 개선되며 정기적으로 업데이트되는 플랫폼입니다. 기능 변경에 따라 CodeCommit 사용 방식이 달라지는 경우가 있습니다. Git과 CodeCommit의 특정 버전에서 문제가 발생할 경우 문제 해결의 내용을 검토합니다.

- Windows의 경우:

  CodeCommit 리포지토리에서 파일과 커밋 및 기타 정보를 사용하려면 로컬 시스템에 Git을 설치해야 합니다. CodeCommit은 Git 버전 1.7.9 이상을 지원합니다. Git 버전 2.28에서는 초기 커밋을 위한 브랜드 이름을 구성할 수 있습니다. 최신 버전의 Git를 사용하는 것이 좋습니다.

  Git를 설치하려면 Git for Windows 등의 웹 사이트를 사용하는 것이 좋습니다. 이 링크를 사용하여 Git을 설치하는 경우 다음을 제외한 설치 기본 설정을 모두 수락할 수 있습니다.

  - PATH 환경 조정 단계에서 메시지가 표시될 경우 명령줄에서 Git를 사용하는 옵션을 선택합니다.
  - (선택 사항) CodeCommit에 대한 Git 자격 증명을 구성하는 AWS CLI 대신에 포함된 자격 증명 헬퍼와 함께 HTTPS를 사용하려면 추가 옵션 구성 페이지에서 Git 자격 증명 관리자 활성화 옵션이 선택 취소되어 있는지 확인합니다. IAM 사용자가 Git 보안 인증 정보를 구성할 경우 Git 보안 인증

관리자는 오직 CodeCommit과 호환됩니다. 자세한 내용은 Git 보안 인증 정보를 사용하는 HTTPS 사용자의 경우 및 Windows용 Git: Windows용 Git을 설치했지만 내 리포지토리에 액세스가 거부되었습니다(403). 단원을 참조하세요.

> **Note**
>
> Git은 계속 개선되며 정기적으로 업데이트되는 플랫폼입니다. 기능 변경에 따라 CodeCommit 사용 방식이 달라지는 경우가 있습니다. Git과 CodeCommit의 특정 버전에서 문제가 발생할 경우 문제 해결의 내용을 검토합니다.

CodeCommit은 HTTPS 및 SSH 인증을 모두 지원합니다. 설정을 완료하려면 CodeCommit에 대한 Git 자격 증명(HTTPS, 대부분의 사용자에게 권장), CodeCommit에 액세스할 때 사용할 SSH 키 페어(SSH), git-remote-codecommit (페더레이션 액세스를 사용하는 사용자에게 권장) 또는 AWS CLI (HTTPS)에 포함된 자격 증명 헬퍼를 구성해야 합니다.

- 지원되는 모든 운영 체제에서의 Git 보안 인증 정보에 대한 정보는 3단계: CodeCommit에 대한 HTTPS 접속을 위한 Git 보안 인증 정보 생성 단원을 참조하세요.
- Linux, macOS, Unix의 SSH에 대한 내용은 SSH와 Linux, macOS Unix: Git과 CodeCommit에 사용되는 퍼블릭 키와 프라이빗 키를 설정합니다. 섹션을 참조하세요.
- Windows에서의 SSH에 대한 정보는 3단계: Git과 CodeCommit에 사용되는 퍼블릭 키와 프라이빗 키 설정 단원을 참조하세요.
- git-remote-codecommit의 경우 git-remote-codecommit을 AWS CodeCommit 사용하여에 HTTPS 연결을 위한 설정 단계 섹션을 참조하세요.
- Linux, macOS, Unix의 보안 인증 도우미에 대한 내용은 보안 인증 도우미 설정(Linux, macOS, Unix)을 참조하세요.
- Windows에서의 보안 인증 도우미에 대한 정보는 보안 인증 도우미 설정(Windows)을 참조하세요.

1단계: CodeCommit 리포지토리 생성

이 섹션에서는 CodeCommit 콘솔을 사용하여 이 자습서의 나머지 부분에서 사용할 CodeCommit 리포지토리를 생성합니다. 를 사용하여 리포지토리 AWS CLI 를 생성하려면 섹션을 참조하세요리포지토리 생성 (AWS CLI).

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.

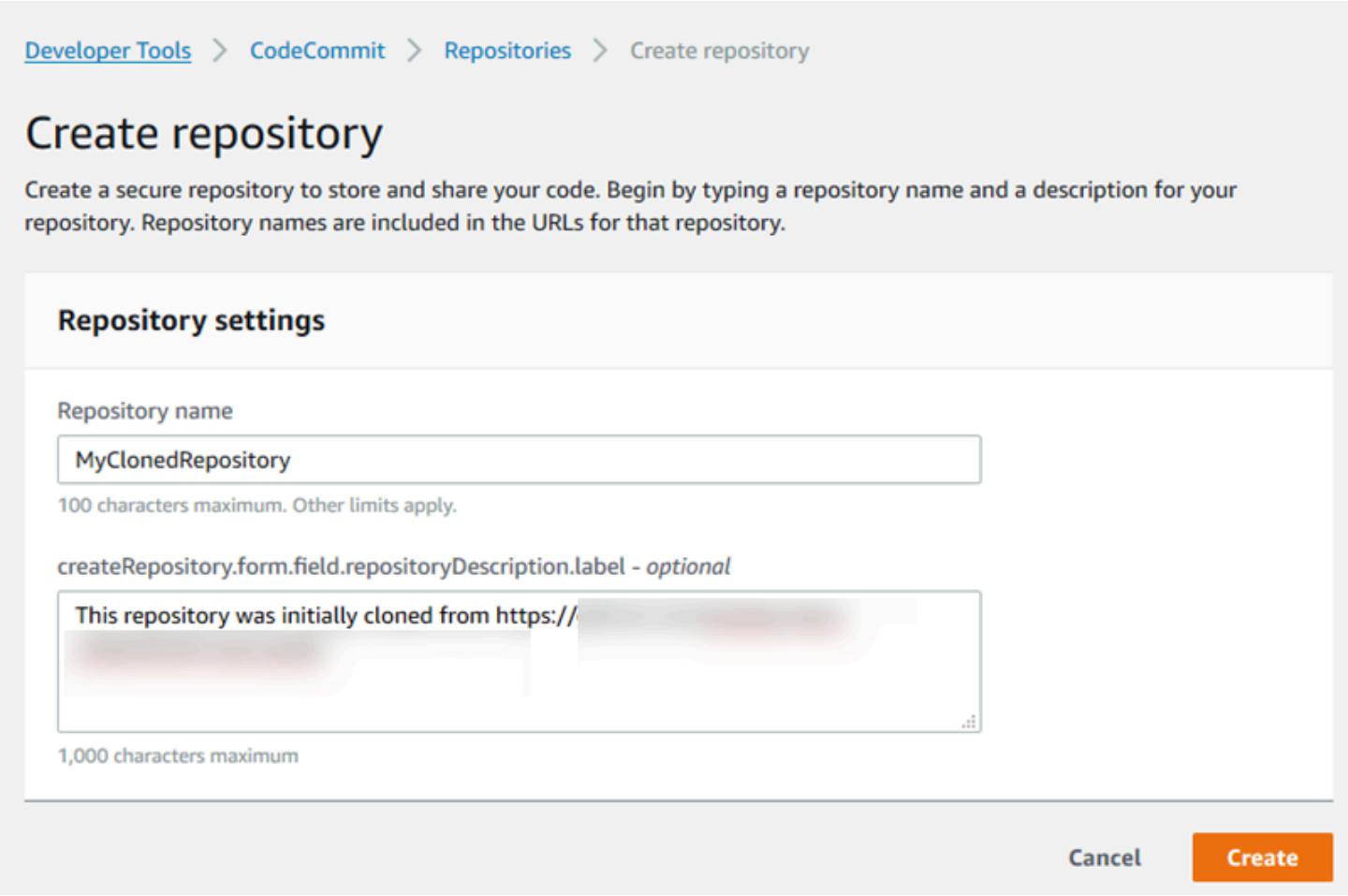
2. 리전 선택기에서 리포지토리를 생성할 AWS 리전 를 선택합니다. 자세한 내용은 리전 및 Git 연결 엔드포인트 단원을 참조하십시오.
3. 리포지토리 페이지에서 리포지토리 생성을 선택합니다.
4. 리포지토리 생성 페이지의 리포지토리 이름에 리포지토리 이름을 입력합니다.

> **Note**
>
> 리포지토리 이름은 대소문자를 구분합니다. 이 이름은 Amazon Web Services 계정이 속한 AWS 리전 에서 고유해야 합니다.

5. (선택 사항) 설명에 리포지토리에 대한 설명을 입력합니다. 그러면 사용자들이 리포지토리의 용도를 식별하는 데 도움이 됩니다.

> **Note**
>
> 설명 필드에는 콘솔의 마크다운이 표시되며, 모든 HTML 문자와 유효한 Unicode 문자를 모두 사용할 수 있습니다. `GetRepository` 또는 `BatchGetRepositories` API를 사용하는 애플리케이션 개발자인 경우 웹 브라우저에 리포지토리 설명 필드를 표시하려면 CodeCommit API 참조를 참조하세요.

6. (선택 사항) 태그 추가를 선택하여 하나 이상의 리포지토리 태그(AWS 리소스를 구성하고 관리하는 데 도움이 되는 사용자 지정 속성 레이블)를 리포지토리에 추가합니다. 자세한 내용은 에서 리포지토리 태그 지정 AWS CodeCommit 단원을 참조하십시오.
7. (선택 사항) 추가 구성을 확장하여이 리포지토리의 데이터를 암호화하고 해독하는 데 기본 AWS 관리형 키 또는 자체 고객 관리형 키를 사용할지 여부를 지정합니다. 자체 고객 관리형 키를 사용하기로 선택한 경우 리포지토리를 생성하는 AWS 리전 에서 사용할 수 있고 키가 활성 상태인지 확인해야 합니다. 자세한 내용은 AWS Key Management Service 및 AWS CodeCommit 리포지토리 암호화 단원을 참조하십시오.
8. (선택 사항) 이 리포지토리에 Java나 Python 코드가 들어가고 CodeGuru Reviewer로 해당 코드를 분석하려 한다면, Java 및 Python용 Amazon CodeGuru Reviewer 활성화를 선택합니다. CodeGuru Reviewer는 다양한 기계 학습 모델을 사용하여 풀 요청의 코드 결함을 찾아내고 개선점과 해결책을 제안합니다. 자세한 내용은 Amazon CodeGuru Reviewer 사용 안내서를 참조하십시오.
9. Create(생성)를 선택합니다.



리포지토리를 생성하면 생성된 리포지토리가 리포지토리 목록에 표시됩니다. URL 열에서 복사 아이콘을 선택한 후 CodeCommit에 접속할 때 사용할 프로토콜(SSH 또는 HTTPS)을 선택합니다. URL을 복사합니다.

예를 들어, 리포지토리 이름을 *MyClonedRepository*로 지정하고 미국 동부(오하이오) 리전에서 HTTPS로 Git 보안 인증 정보를 사용하는 경우, URL은 다음과 같습니다.

```
https://git-codecommit.us-east-2.amazonaws.com/MyClonedRepository
```

이 URL은 나중에 2단계: 리포지토리를 복제하여 CodeCommit 리포지토리로 푸시에서 필요합니다.

2단계: 리포지토리를 복제하여 CodeCommit 리포지토리로 푸시

이 단원에서는 Git 리포지토리를 로컬 컴퓨터에 복제하여 소위 말하는 로컬 리포지토리를 생성합니다. 그런 다음 로컬 리포지토리의 콘텐츠를 이전에 생성한 CodeCommit 리포지토리로 푸시합니다.

1. 로컬 컴퓨터의 터미널 또는 명령 프롬프트에서 `--mirror` 옵션과 함께 git clone 명령을 실행하여 원격 리포지토리의 사본을 새 폴더 *aws-codecommit-demo*로 복제합니다. 이것은 마이그레션 전용 베어 리포지토리입니다. CodeCommit의 마이그레이션된 리포지토리와 상호 작용하기 위한 로컬 리포지토리가 아닙니다. 나중에 CodeCommit으로의 마이그레이션이 완료된 후에 생성할 수 있습니다.

   다음 예제는 GitHub(*https://github.com/awslabs/aws-demo-php-simple-app.git*)에서 호스팅되는 데모 애플리케이션을 디렉터리 *aws-codecommit-demo*의 로컬 리포지토리로 복제합니다.

   ```
   git clone --mirror https://github.com/awslabs/aws-demo-php-simple-app.git aws-codecommit-demo
   ```

2. 디렉터리를 복제를 생성한 디렉터리로 변경합니다.

   ```
   cd aws-codecommit-demo
   ```

3. URL과 대상 CodeCommit 리포지토리의 이름, --all 옵션을 지정하여 git push 명령을 실행합니다. (여기서 URL은 1단계: CodeCommit 리포지토리 생성에서 복사해 놓은 것입니다.)

   예를 들어 리포지토리의 이름을 *MyClonedRepository*로 짓고 HTTPS를 사용하도록 설정된 경우, 다음 명령을 실행합니다.

   ```
   git push https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyClonedRepository
    --all
   ```

   > **Note**
   >
   > --all 옵션은 리포지토리의 브랜치만을 모두 푸시합니다. 태그와 같은 다른 참조는 푸시하지 않습니다. 태그를 푸시하려면 초기 푸시가 완료될 때까지 기다렸다가 이번에는 --tags 옵션을 사용하여 다음과 같이 다시 푸시합니다.
   >
   > ```
   > git push ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyClonedRepository --tags
   > ```
   >
   > 자세한 내용은 Git 웹 사이트의 Git push 단원을 참조하세요. 대용량 리포지토리를 푸시, 특히 모든 참조를 한 번에 푸시하는 방법(예: --mirror 옵션 사용)에 대한 자세한 내용은 리포지토리를 증분으로 마이그레이션하기 섹션을 참조하세요.

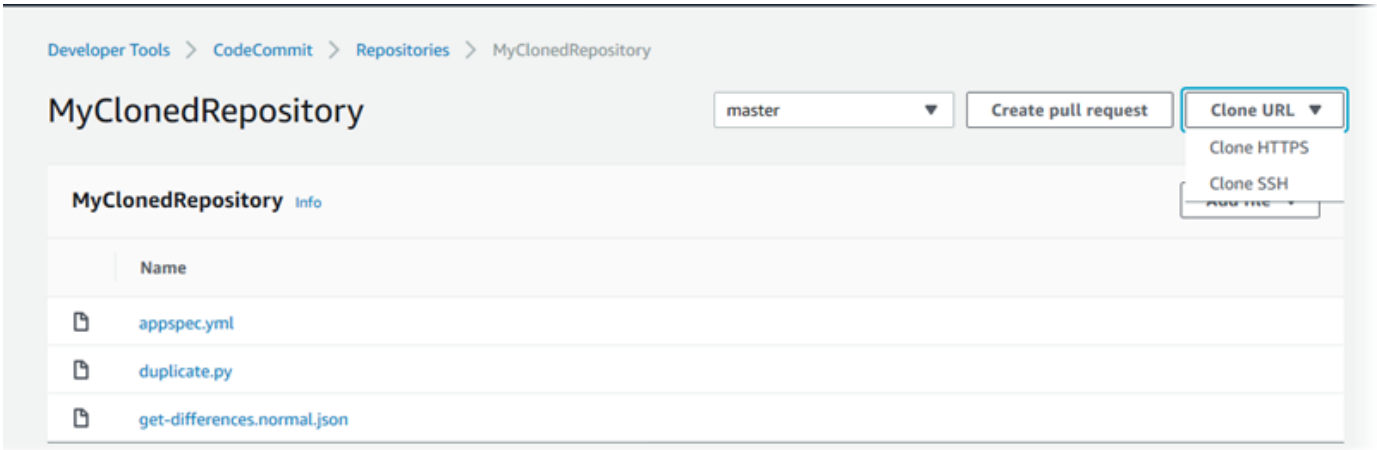
aws-codecommit-demo 폴더와 그 콘텐츠는 리포지토리를 CodeCommit으로 마이그레이션한 후에 삭제할 수 있습니다. CodeCommit에서 리포지토리 작업에 필요한 모든 올바른 참조를 사용하여 로컬 리포지토리를 생성하려면 `--mirror` 옵션 없이 `git clone` 명령을 실행합니다.

```
git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyClonedRepository
```

3단계: CodeCommit에서 파일 보기

디렉터리의 콘텐츠를 푸시한 후에는 CodeCommit 콘솔을 사용하여 해당 리포지토리의 모든 파일을 빠르게 열람할 수 있습니다.

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리에서 해당 리포지토리의 이름을 선택합니다(예: *MyClonedRepository*).
3. 해당 리포지토리에 있는 파일에서 브랜치, 복제 URL, 설정 등을 조회합니다.



4단계: CodeCommit 리포지토리 공유

CodeCommit에 리포지토리를 생성하면 엔드포인트가 두 개 생성되는데, 하나는 HTTPS 접속을 위한 것이고 다른 하나는 SSH 접속을 위한 것입니다. 둘 다 네트워크를 통한 안전한 접속이 가능합니다. 사용자는 둘 중 한 가지 프로토콜을 사용할 수 있습니다. 사용자에게 어떤 프로토콜을 권장하든 두 엔드포인트 모두 활성화된 상태를 유지합니다. 자신의 리포지토리를 다른 사람과 공유하기 전에, 먼저 다른 사용자가 자신의 리포지토리에 액세스하도록 허용하는 IAM 정책을 생성해야 합니다. 그러한 액세스 지침을 해당 사용자들에게 전달하세요.

리포지토리에 대한 고객 관리형 정책 생성

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/://https://https://https://https://://://https://://https://://https://://://https://://https://://https
2. 대시보드 탐색 영역에서 정책을 선택한 다음 정책 생성을 선택합니다.
3. 정책 생성 페이지에서 관리형 정책 가져오기를 선택합니다.
4. 관리형 정책 가져오기 페이지의 필터 정책에 **AWSCodeCommitPowerUser**를 입력합니다. 정책 이름 옆에 있는 버튼을 선택한 다음 가져오기를 선택합니다.
5. 정책 생성 페이지에서 JSON을 선택합니다. 다음과 같이 CodeCommit 작업을 위한 `Resource` 줄의 "*" 부분을 CodeCommit 리포지토리의 Amazon 리소스 이름(ARN)으로 교체합니다.

```
"Resource": [
 "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo"
 ]
```

> **Tip**
>
> CodeCommit 리포지토리의 ARN을 찾으려면, CodeCommit 콘솔로 이동하고 목록에서 리포지토리 이름을 선택한 다음 설정을 선택합니다. 자세한 내용은 리포지토리 세부 정보 보기 단원을 참조하십시오.

이 정책을 둘 이상의 리포지토리에 적용하려면, ARN을 지정하여 각 리포지토리를 리소스로 추가합니다. 다음과 같이 각 리소스 명령문 사이에 쉼표를 넣습니다.

```
"Resource": [
 "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
 "arn:aws:codecommit:us-east-2:111111111111:MyOtherDemoRepo"
 ]
```

편집이 완료되면 정책 검토를 선택합니다.

6. 정책 검토 페이지의 이름에 정책의 새 이름을 입력합니다(예: *AWSCodeCommitPowerUser-MyDemoRepo*). 선택 사항으로 이 정책에 대한 설명을 제공할 수 있습니다.
7. 정책 생성을 선택합니다.

자신의 리포지토리에 대한 액세스를 관리하려면, 리포지토리 사용자에 대한 IAM 그룹을 생성하고, 해당 그룹에 IAM 사용자를 추가한 다음, 이전 단계에서 생성한 고객 관리형 정책을 연결합니다. IAMUserSSHKeys 또는 IAMSelfManageServiceSpecificCredentials와 같이, 액세스에 필요한 기타 정책을 연결합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/://https://https://https://https://://https://://https://://https://://https://://https://://
2. 대시보드 탐색 영역에서 그룹을 선택한 다음 새 그룹 생성을 선택합니다.
3. 그룹 이름 설정 페이지의 그룹 이름에 그룹 이름(예: *MyDemoRepoGroup*)을 입력한 다음 다음 단계를 선택합니다. 그룹 이름에는 리포지토리 이름을 포함시키는 것이 좋습니다.

   > **Note**
   > 그룹 이름은 Amazon Web Services 계정 전체에서 고유해야 합니다.

4. 이전 섹션에서 생성한 고객 관리형 정책(예: AWSCodeCommitPowerUser-MyDemoRepo) 옆의 상자를 선택합니다.
5. 검토 페이지에서 그룹 생성을 선택합니다. IAM은 지정된 정책이 이미 연결된 상태로 이 그룹을 생성합니다. 해당 그룹이 Amazon Web Services 계정에 연결된 그룹 목록에 나타납니다.
6. 목록에서 그룹을 선택합니다.
7. 그룹 요약 페이지에서 사용자 탭을 선택한 다음 그룹에 사용자 추가를 선택합니다. Amazon Web Services 계정과 연결된 모든 사용자를 표시하는 목록에서, CodeCommit 리포지토리에 대한 액세스를 허용할 사용자 옆의 상자를 선택한 다음 사용자 추가를 선택합니다.

   > **Tip**
   > 검색 상자를 사용하여 이름별로 사용자를 빠르게 찾을 수 있습니다.

8. 사용자를 추가했으면 IAM 콘솔을 닫습니다.

직접 구성한 정책 그룹과 정책을 사용하여 CodeCommit에 액세스할 IAM 사용자를 생성한 후에, 해당 사용자에게 리포지토리에 접속하는 데 필요한 정보를 전송합니다.

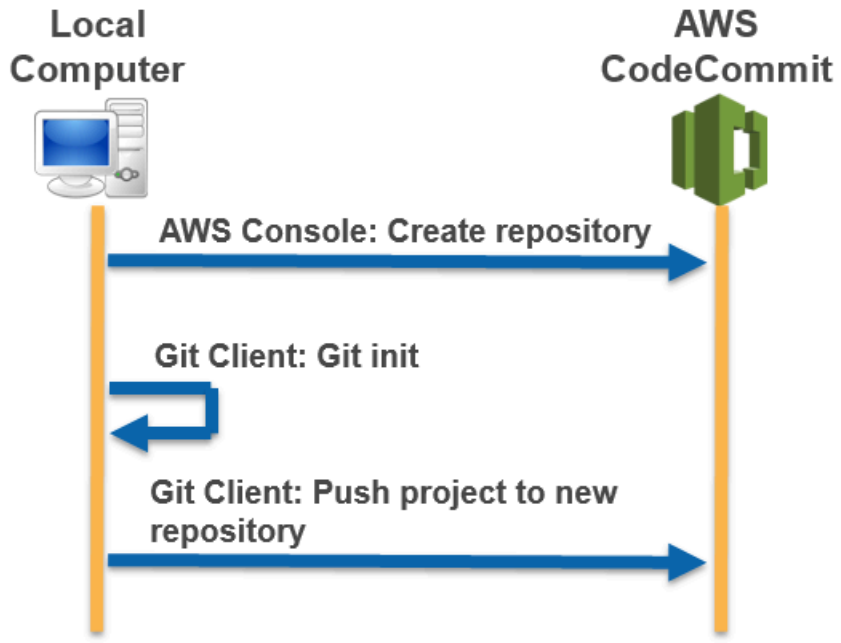
1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리전 선택기에서 리포지토리가 생성된 AWS 리전 를 선택합니다. 리포지토리는에 고유합니다 AWS 리전. 자세한 내용은 리전 및 Git 연결 엔드포인트 단원을 참조하십시오.

3. 리포지토리 페이지에서 공유할 리포지토리를 선택합니다.
4. URL 복제에서 사용자가 사용하게 할 프로토콜을 선택합니다. 그러면 연결 프로토콜에 대한 복제 URL이 복사됩니다.
5. 사용자에게 복제 URL을 설치, 프로필 AWS CLI구성 또는 Git 설치와 같은 다른 지침과 함께 전송합니다. 연결 프로토콜(예: HTTPS)에 대한 구성 정보를 포함해야 합니다.

로컬 또는 버전이 지정되지 않은 콘텐츠를 로 마이그레이션 AWS CodeCommit

이 주제에서는 컴퓨터에 있는 기존 프로젝트 또는 로컬 콘텐츠를 CodeCommit 리포지토리로 마이그레이션하는 방법을 안내합니다. 이 절차의 일부로써 다음을 수행합니다.

- CodeCommit에 필요한 초기 설정 완료.
- CodeCommit 리포지토리를 생성합니다.
- 로컬 폴더를 Git 버전 관리 하에 두고 이 폴더의 콘텐츠를 CodeCommit 리포지토리로 푸시합니다.
- CodeCommit 리포지토리에서 파일을 봅니다.
- 소속 팀과 CodeCommit 리포지토리를 공유합니다.



주제

- 0단계: CodeCommit 액세스에 필요한 설정
- 1단계: CodeCommit 리포지토리 생성
- 2단계: 로컬 콘텐츠를 CodeCommit 리포지토리로 마이그레이션
- 3단계: CodeCommit에서 파일 보기
- 4단계: CodeCommit 리포지토리 공유

0단계: CodeCommit 액세스에 필요한 설정

로컬 콘텐츠를 CodeCommit으로 마이그레이션하기 전에, 먼저 CodeCommit에 IAM 사용자를 생성 및 구성하고 액세스를 위해 로컬 컴퓨터를 구성해야 합니다. 또한 CodeCommit을 관리하기 위해 AWS CLI 를 설치해야 합니다. AWS CLI 가 없어도 대부분의 CodeCommit 작업을 수행할 수 있지만, 만약 있으면 Git으로 작업할 때 융통성을 발휘할 수 있습니다.

CodeCommit에 대한 설정을 이미 마친 경우에는 1단계: CodeCommit 리포지토리 생성로 건너뛸 수 있습니다.

CodeCommit에 액세스할 IAM 사용자를 생성 및 구성하려면

1. http://aws.amazon.com에서 가입을 선택하여 Amazon Web Services 계정을 만듭니다.
2. Amazon Web Services 계정에서 IAM 사용자를 생성하거나 기존 사용자를 사용합니다. 액세스 키 ID가 있고 해당 IAM 사용자와 연결된 비밀 액세스 키가 있는지 확인합니다. 자세한 내용은 Amazon Web Services 계정에서 IAM 사용자 생성하기를 참조하세요.

   > **Note**
   >
   > CodeCommit에는가 필요합니다 AWS Key Management Service. 기존 IAM 사용자를 사용하는 경우 CodeCommit에 필요한 AWS KMS 작업을 명시적으로 거부하는 정책이 사용자에게 연결되어 있지 않은지 확인합니다. 자세한 내용은 AWS KMS 및 암호화 단원을 참조하십시오.

3. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/://https://https://https://https://://https://://https://://https://://https://://https://://https://://
4. IAM 콘솔의 탐색 창에서 사용자를 선택한 다음 CodeCommit 액세스에 대해 구성할 IAM 사용자를 선택합니다.
5. 권한 탭에서 권한 추가를 선택합니다.

6. 권한 부여에서 기존 정책 직접 연결을 선택합니다.
7. 정책 목록에서, CodeCommit 액세스에 대한 AWSCodeCommitPowerUser 또는 다른 관리형 정책을 선택합니다. 자세한 내용은 AWS CodeCommit에 대한 관리형 정책 단원을 참조하십시오.

   연결할 정책을 선택한 후 다음: 검토를 선택하여 IAM 사용자에게 연결할 정책의 목록을 검토합니다. 목록이 올바르면 권한 추가를 선택합니다.

   CodeCommit 관리형 정책, 그리고 리포지토리에 대한 액세스를 다른 그룹 및 사용자와 공유하는 방법에 대해 자세히 알아보려면 리포지토리 공유, AWS CodeCommit에 대한 인증 및 액세스 제어 섹션을 참조하세요.

설치 및 구성 AWS CLI

1. 로컬 시스템에서를 다운로드하여 설치합니다 AWS CLI. 이는 명령줄에서 CodeCommit과 상호 작용하기 위한 사전 조건입니다. AWS CLI 버전 2를 설치하는 것이 좋습니다. 의 최신 메이저 버전이며 모든 최신 기능을 AWS CLI 지원합니다. 에서 루트 계정, 페더레이션 액세스 또는 임시 자격 증명 사용을 AWS CLI 지원하는 유일한 버전의 입니다git-remote-codecommit.

   자세한 내용은 AWS 명령줄 인터페이스 설정 단원을 참조하십시오.

   > **Note**
   >
   > CodeCommit은 AWS CLI 버전 1.7.38 이상에서만 작동합니다. 가장 좋은 방법은를 설치하거나 사용 가능한 최신 버전으로 업그레이드 AWS CLI 하는 것입니다. 설치 AWS CLI 한 버전을 확인하려면 aws --version 명령을 실행합니다.
   > 의 이전 버전을 AWS CLI 최신 버전으로 업그레이드하려면 설치를 참조하세요 AWS Command Line Interface.

2. 이 명령을 실행하여에 대한 CodeCommit 명령 AWS CLI 이 설치되어 있는지 확인합니다.

   ```
   aws codecommit help
   ```

   이 명령은 CodeCommit 명령 목록을 반환합니다.

3. 다음과 같이 configure 명령을 사용하여 프로파일 AWS CLI 로를 구성합니다.

   ```
   aws configure
   ```

메시지가 표시되면 CodeCommit과 함께 사용할 IAM 사용자의 AWS 액세스 키와 AWS 보안 액세스 키를 지정합니다. 또한와 같이 리포지토리 AWS 리전 가 있는를 지정해야 합니다`us-east-2`. 기본 출력 형식을 묻는 메시지가 표시되면 `json`을 지정합니다. 예를 들어, IAM 사용자의 프로필을 구성하는 경우에는 다음과 같이 합니다.

```
AWS Access Key ID [None]: Type your IAM user AWS access key ID here, and then press
 Enter
AWS Secret Access Key [None]: Type your IAM user AWS secret access key here, and
 then press Enter
Default region name [None]: Type a supported region for CodeCommit here, and then
 press Enter
Default output format [None]: Type json here, and then press Enter
```

에서 사용할 프로필을 생성하고 구성하는 방법에 대한 자세한 내용은 다음을 AWS CLI참조하세요.

- 명명된 프로파일
- 에서 IAM 역할 사용 AWS CLI
- 설정 명령
- 교체 자격 증명을 사용하여 AWS CodeCommit 리포지토리에 연결

리포지토리 또는 다른의 리소스에 연결하려면 기본 리전 이름으로 AWS CLI 를 재구성 AWS 리전 해야 합니다. CodeCommit에서 지원되는 기본 리전 이름은 다음과 같습니다.

- us-east-2
- us-east-1
- eu-west-1
- us-west-2
- ap-northeast-1
- ap-southeast-1
- ap-southeast-2
- ap-southeast-3
- me-central-1
- eu-central-1
- ap-northeast-2

- sa-east-1
- us-west-1
- eu-west-2
- ap-south-1
- ap-south-1
- ca-central-1
- us-gov-west-1
- us-gov-east-1
- eu-north-1
- ap-east-1
- me-south-1
- cn-north-1
- cn-northwest-1
- eu-south-1
- ap-northeast-3
- af-south-1
- il-central-1

CodeCommit 및에 대한 자세한 내용은 섹션을 AWS 리전참조하세요리전 및 Git 연결 엔드포인트. IAM, 액세스 키, 비밀 키에 대한 자세한 내용을 확인하려면 보안 인증 정보 얻는 방법 및 IAM 사용자의 액세스 키 관리를 참조하세요. AWS CLI 및 프로파일에 대한 자세한 내용은 명명된 프로파일을 참조하세요.

그 다음에는 Git을 설치해야 합니다.

- Linux, macOS, Unix의 경우:

  CodeCommit 리포지토리에서 파일과 커밋 및 기타 정보를 사용하려면 로컬 시스템에 Git을 설치해야 합니다. CodeCommit은 Git 버전 1.7.9 이상을 지원합니다. Git 버전 2.28에서는 초기 커밋을 위한 브랜드 이름을 구성할 수 있습니다. 최신 버전의 Git를 사용하는 것이 좋습니다.

  Git 설치를 위해서는 Git 다운로드와 같은 웹 사이트를 권장합니다.

> **Note**
>
> Git은 계속 개선되며 정기적으로 업데이트되는 플랫폼입니다. 기능 변경에 따라 CodeCommit 사용 방식이 달라지는 경우가 있습니다. Git과 CodeCommit의 특정 버전에서 문제가 발생할 경우 문제 해결의 내용을 검토합니다.

- Windows의 경우:

  CodeCommit 리포지토리에서 파일과 커밋 및 기타 정보를 사용하려면 로컬 시스템에 Git을 설치해야 합니다. CodeCommit은 Git 버전 1.7.9 이상을 지원합니다. Git 버전 2.28에서는 초기 커밋을 위한 브랜드 이름을 구성할 수 있습니다. 최신 버전의 Git를 사용하는 것이 좋습니다.

  Git를 설치하려면 Git for Windows 등의 웹 사이트를 사용하는 것이 좋습니다. 이 링크를 사용하여 Git을 설치하는 경우 다음을 제외한 설치 기본 설정을 모두 수락할 수 있습니다.

  - PATH 환경 조정 단계에서 메시지가 표시될 경우 명령줄에서 Git를 사용하는 옵션을 선택합니다.
  - (선택 사항) CodeCommit에 대한 Git 자격 증명을 구성하는 AWS CLI 대신에 포함된 자격 증명 헬퍼와 함께 HTTPS를 사용하려면 추가 옵션 구성 페이지에서 Git 자격 증명 관리자 활성화 옵션이 선택 취소되어 있는지 확인합니다. IAM 사용자가 Git 보안 인증 정보를 구성할 경우 Git 보안 인증 관리자는 오직 CodeCommit과 호환됩니다. 자세한 내용은 Git 보안 인증 정보를 사용하는 HTTPS 사용자의 경우 및 Windows용 Git: Windows용 Git을 설치했지만 내 리포지토리에 액세스가 거부되었습니다(403). 단원을 참조하세요.

> **Note**
>
> Git은 계속 개선되며 정기적으로 업데이트되는 플랫폼입니다. 기능 변경에 따라 CodeCommit 사용 방식이 달라지는 경우가 있습니다. Git과 CodeCommit의 특정 버전에서 문제가 발생할 경우 문제 해결의 내용을 검토합니다.

CodeCommit은 HTTPS 및 SSH 인증을 모두 지원합니다. 설정을 완료하려면, CodeCommit에 대한 Git 보안 인증 정보(HTTPS, 대부분의 사용자에게 권장), CodeCommit에 액세스할 때 사용할 SSH 키 페어(SSH), git-remote-codecommit(페더레이션 액세스를 사용하는 사용자에게 권장) 또는 AWS CLI에 포함된 보안 인증 도우미 등을 구성해야 합니다.

- 지원되는 모든 운영 체제에서의 Git 보안 인증 정보에 대한 정보는 3단계: CodeCommit에 대한 HTTPS 접속을 위한 Git 보안 인증 정보 생성 단원을 참조하세요.

- Linux, macOS, Unix의 SSH에 대한 내용은 SSH와 Linux, macOS Unix: Git과 CodeCommit에 사용되는 퍼블릭 키와 프라이빗 키를 설정합니다. 섹션을 참조하세요.
- Windows에서의 SSH에 대한 정보는 3단계: Git과 CodeCommit에 사용되는 퍼블릭 키와 프라이빗 키 설정 단원을 참조하세요.
- git-remote-codecommit의 경우 git-remote-codecommit을 AWS CodeCommit 사용하여에 HTTPS 연결을 위한 설정 단계 섹션을 참조하세요.
- Linux, macOS, Unix의 보안 인증 도우미에 대한 내용은 보안 인증 도우미 설정(Linux, macOS, Unix)을 참조하세요.
- Windows에서의 보안 인증 도우미에 대한 정보는 보안 인증 도우미 설정(Windows)을 참조하세요.

1단계: CodeCommit 리포지토리 생성

이 섹션에서는 CodeCommit 콘솔을 사용하여 이 자습서의 나머지 부분에서 사용할 CodeCommit 리포지토리를 생성합니다. 를 사용하여 리포지토리 AWS CLI 를 생성하려면 섹션을 참조하세요리포지토리 생성 (AWS CLI).

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리전 선택기에서 리포지토리를 생성할 AWS 리전 를 선택합니다. 자세한 내용은 리전 및 Git 연결 엔드포인트 단원을 참조하십시오.
3. 리포지토리 페이지에서 리포지토리 생성을 선택합니다.
4. 리포지토리 생성 페이지의 리포지토리 이름에 리포지토리 이름을 입력합니다.

   > **Note**
   > 리포지토리 이름은 대소문자를 구분합니다. 이 이름은 Amazon Web Services 계정이 속한 AWS 리전 에서 고유해야 합니다.

5. (선택 사항) 설명에 리포지토리에 대한 설명을 입력합니다. 그러면 사용자들이 리포지토리의 용도를 식별하는 데 도움이 됩니다.

   > **Note**
   > 설명 필드에는 콘솔의 마크다운이 표시되며, 모든 HTML 문자와 유효한 Unicode 문자를 모두 사용할 수 있습니다. `GetRepository` 또는 `BatchGetRepositories` API를 사

용하는 애플리케이션 개발자인 경우 웹 브라우저에 리포지토리 설명 필드를 표시하려면 CodeCommit API 참조를 참조하세요.

6. (선택 사항) 태그 추가를 선택하여 하나 이상의 리포지토리 태그(AWS 리소스를 구성하고 관리하는 데 도움이 되는 사용자 지정 속성 레이블)를 리포지토리에 추가합니다. 자세한 내용은 에서 리포지토리 태그 지정 AWS CodeCommit 단원을 참조하십시오.
7. (선택 사항) 추가 구성을 확장하여이 리포지토리의 데이터를 암호화하고 해독하는 데 기본 AWS 관리형 키 또는 자체 고객 관리형 키를 사용할지 여부를 지정합니다. 자체 고객 관리형 키를 사용하기로 선택한 경우 리포지토리를 생성하는 AWS 리전 에서 사용할 수 있고 키가 활성 상태인지 확인해야 합니다. 자세한 내용은 AWS Key Management Service 및 AWS CodeCommit 리포지토리 암호화 단원을 참조하십시오.
8. (선택 사항) 이 리포지토리에 Java나 Python 코드가 들어가고 CodeGuru Reviewer로 해당 코드를 분석하려 한다면, Java 및 Python용 Amazon CodeGuru Reviewer 활성화를 선택합니다. CodeGuru Reviewer는 다양한 기계 학습 모델을 사용하여 풀 요청의 코드 결함을 찾아내고 개선점과 해결책을 제안합니다. 자세한 내용은 Amazon CodeGuru Reviewer 사용 안내서를 참조하십시오.
9. 생성(Create)을 선택합니다.

리포지토리를 생성하면 생성된 리포지토리가 리포지토리 목록에 표시됩니다. URL 열에서 복사 아이콘을 선택한 후 CodeCommit에 접속할 때 사용할 프로토콜(HTTPS 또는 SSH)을 선택합니다. URL을 복사합니다.

예를 들어 리포지토리의 이름을 *MyFirstRepo*로 짓고 HTTPS를 사용하는 경우, URL은 다음과 같습니다.

```
https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyFirstRepo
```

이 URL은 나중에 2단계: 로컬 콘텐츠를 CodeCommit 리포지토리로 마이그레이션에서 필요합니다.

2단계: 로컬 콘텐츠를 CodeCommit 리포지토리로 마이그레이션

현재 CodeCommit 리포지토리가 있으므로 로컬 컴퓨터에서 로컬 Git 리포지토리로 변환할 디렉터리를 선택할 수 있습니다. git init 명령을 사용해 기존의 버전 관리 미사용 콘텐츠를 Git 리포지토리로 변환하거나, 파일 또는 콘텐츠가 아직 없는 경우에는 비어 있는 새 리포지토리를 초기화할 수 있습니다.

1. 로컬 컴퓨터의 터미널 또는 명령줄에서 디렉터리를 해당 리포지토리의 소스로 사용할 디렉터리로 변경합니다.

2. 다음 명령을 실행하여 **main**이라는 기본 브랜치를 사용하도록 Git을 구성합니다.

```
git config --local init.defaultBranch main
```

이 명령을 실행하여, 새로 생성한 모든 리포지토리의 기본 브랜치 이름을 **main**으로 설정할 수도 있습니다.

```
git config --global init.defaultBranch main
```

3. git init 명령을 실행하여 디렉터리에서 Git 버전 관리를 초기화합니다. 이렇게 하면 디렉터리의 루트에 버전 관리 추적을 가능케 하는 .git 하위 디렉터리가 생성됩니다. 또한 .git 폴더에는 리포지토리에 필요한 모든 메타데이터가 들어 있습니다.

```
git init
```

4. 다음 명령을 실행하여 초기화된 디렉터리의 상태를 확인합니다.

```
git status
```

버전 관리에 추가하고 싶은 파일을 추가합니다. 이 자습서에서는 `.` 지정자와 함께 `git add` 명령을 실행하여 이 디렉터리에 있는 모든 파일을 추가해 보겠습니다. 다른 옵션에 대해서는 해당 Git 문서를 참조하세요.

```
git add .
```

5. 커밋 메시지와 함께 추가된 파일에 대한 커밋을 생성합니다.

```
git commit -m "Initial commit"
```

6. 대상 CodeCommit 리포지토리의 URL과 이름, `--all` 옵션을 지정하여 git push 명령을 실행합니다. (여기서 URL은 1단계: CodeCommit 리포지토리 생성에서 복사해 놓은 것입니다.)

예를 들어 리포지토리의 이름을 *MyFirstRepo*로 짓고 HTTPS를 사용하도록 설정된 경우, 다음 명령을 실행합니다.

```
git push https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyFirstRepo --all
```

3단계: CodeCommit에서 파일 보기

디렉터리의 콘텐츠를 푸시한 후에는 CodeCommit 콘솔을 사용하여 리포지토리의 모든 파일을 빠르게 열람할 수 있습니다.

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리포지토리의 목록에서 해당 리포지토리의 이름을 선택합니다(예: *MyFirstRepository*).
3. 해당 리포지토리에 있는 파일에서 브랜치, 복제 URL, 설정 등을 조회합니다.

4단계: CodeCommit 리포지토리 공유

CodeCommit에 리포지토리를 생성하면 엔드포인트가 두 개 생성되는데, 하나는 HTTPS 접속을 위한 것이고 다른 하나는 SSH 접속을 위한 것입니다. 둘 다 네트워크를 통한 안전한 접속이 가능합니다. 사용자는 둘 중 한 가지 프로토콜을 사용할 수 있습니다. 사용자에게 어떤 프로토콜을 권장하든 두 엔드포인트 모두 활성화된 상태를 유지합니다. 자신의 리포지토리를 다른 사람과 공유하기 전에, 먼저 다른 사용자가 자신의 리포지토리에 액세스하도록 허용하는 IAM 정책을 생성해야 합니다. 그러한 액세스 지침을 해당 사용자들에게 전달하세요.

리포지토리에 대한 고객 관리형 정책 생성

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/://https://https://https://https://://https://://https://://https://https://https://https://://://https
2. 대시보드 탐색 영역에서 정책을 선택한 다음 정책 생성을 선택합니다.
3. 정책 생성 페이지에서 관리형 정책 가져오기를 선택합니다.
4. 관리형 정책 가져오기 페이지의 필터 정책에 **AWSCodeCommitPowerUser**를 입력합니다. 정책 이름 옆에 있는 버튼을 선택한 다음 가져오기를 선택합니다.
5. 정책 생성 페이지에서 JSON을 선택합니다. 다음과 같이 CodeCommit 작업을 위한 `Resource` 줄의 “*” 부분을 CodeCommit 리포지토리의 Amazon 리소스 이름(ARN)으로 교체합니다.

```
"Resource": [
 "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo"
 ]
```

> **Tip**
>
> CodeCommit 리포지토리의 ARN을 찾으려면, CodeCommit 콘솔로 이동하고 목록에서 리포지토리 이름을 선택한 다음 설정을 선택합니다. 자세한 내용은 리포지토리 세부 정보 보기 단원을 참조하십시오.

이 정책을 둘 이상의 리포지토리에 적용하려면, ARN을 지정하여 각 리포지토리를 리소스로 추가합니다. 다음과 같이 각 리소스 명령문 사이에 쉼표를 넣습니다.

```
"Resource": [
 "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
 "arn:aws:codecommit:us-east-2:111111111111:MyOtherDemoRepo"
 ]
```

편집이 완료되면 정책 검토를 선택합니다.

6. 정책 검토 페이지의 이름에 정책의 새 이름을 입력합니다(예: *AWSCodeCommitPowerUser-MyDemoRepo*). 선택 사항으로 이 정책에 대한 설명을 제공할 수 있습니다.
7. 정책 생성을 선택합니다.

자신의 리포지토리에 대한 액세스를 관리하려면, 리포지토리 사용자에 대한 IAM 그룹을 생성하고, 해당 그룹에 IAM 사용자를 추가한 다음, 이전 단계에서 생성한 고객 관리형 정책을 연결합니다. `IAMSelfManageServiceSpecificCredentials` 또는 `IAMUserSSHKeys`와 같이 액세스에 필요한 기타 정책을 연결합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/://https://https://https://://https://://https://://https://://https://://https://://https://
2. 대시보드 탐색 영역에서 그룹을 선택한 다음 새 그룹 생성을 선택합니다.
3. 그룹 이름 설정 페이지의 그룹 이름에 그룹 이름(예: *MyDemoRepoGroup*)을 입력한 다음 다음 단계를 선택합니다. 그룹 이름에는 리포지토리 이름을 포함시키는 것이 좋습니다.

> **Note**
>
> 그룹 이름은 Amazon Web Services 계정 전체에서 고유해야 합니다.

4. 이전 섹션에서 생성한 고객 관리형 정책(예: AWSCodeCommitPowerUser-MyDemoRepo) 옆의 상자를 선택합니다.
5. 검토 페이지에서 그룹 생성을 선택합니다. IAM은 지정된 정책이 이미 연결된 상태로 이 그룹을 생성합니다. 해당 그룹이 Amazon Web Services 계정에 연결된 그룹 목록에 나타납니다.
6. 목록에서 그룹을 선택합니다.
7. 그룹 요약 페이지에서 사용자 탭을 선택한 다음 그룹에 사용자 추가를 선택합니다. Amazon Web Services 계정과 연결된 모든 사용자를 표시하는 목록에서, CodeCommit 리포지토리에 대한 액세스를 허용할 사용자 옆의 상자를 선택한 다음 사용자 추가를 선택합니다.

   > **Tip**
   >
   > 검색 상자를 사용하여 이름별로 사용자를 빠르게 찾을 수 있습니다.

8. 사용자를 추가했으면 IAM 콘솔을 닫습니다.

직접 구성한 정책 그룹과 정책을 사용하여 CodeCommit에 액세스하는 데 사용할 IAM 사용자를 생성한 후에, 해당 사용자에게 리포지토리에 접속하는 데 필요한 정보를 전송합니다.

1. https://console.aws.amazon.com/codesuite/codecommit/home에서 CodeCommit 콘솔을 엽니다.
2. 리전 선택기에서 리포지토리가 생성된 AWS 리전 를 선택합니다. 리포지토리는에 고유합니다 AWS 리전. 자세한 내용은 리전 및 Git 연결 엔드포인트 단원을 참조하십시오.
3. 리포지토리 페이지에서 공유할 리포지토리를 선택합니다.
4. URL 복제에서 사용자가 사용하게 할 프로토콜을 선택합니다. 그러면 연결 프로토콜에 대한 복제 URL이 복사됩니다.
5. 사용자에게 복제 URL을 설치, 프로필 AWS CLI구성 또는 Git 설치와 같은 다른 지침과 함께 전송합니다. 연결 프로토콜(예: HTTPS)에 대한 구성 정보를 포함해야 합니다.

리포지토리를 증분으로 마이그레이션하기

로 마이그레이션할 때는 리포지토리를 증분 또는 청크로 푸시하여 간헐적인 네트워크 문제 또는 네트워크 성능 저하로 인해 전체 푸시가 실패할 가능성을 줄이는 것이 AWS CodeCommit좋습니다. 여기에 포함된 것과 같은 스크립트로 증분 푸시를 사용하면, 마이그레이션을 다시 시작하고 이전 시도에서 성공하지 못한 커밋만 푸시할 수 있습니다.

이 항목에 소개된 절차는 리포지토리를 증분으로 마이그레이션하고 그중 성공하지 못한 증분만 마이그레이션이 완료될 때까지 다시 푸시하는 스크립트를 만들고 실행하는 방법을 보여 줍니다.

이 지침들은 사용자가 설정 및 리포지토리 생성의 단계를 이미 완료했다는 가정하에 작성되었습니다.

주제

- 0단계: 증분 마이그레이션 여부 결정
- 1단계: 필수 구성 요소를 설치하고 CodeCommit 리포지토리를 원격으로 추가
- 2단계: 증분 마이그레이션에 사용할 스크립트 생성
- 3단계: 스크립트 실행 및 CodeCommit으로 증분 마이그레이션
- 부록: 샘플 스크립트 incremental-repo-migration.py

0단계: 증분 마이그레이션 여부 결정

리포지토리의 전체 크기와 증분 마이그레이션의 여부를 결정할 때는 몇 가지 요소를 고려해야 합니다. 이 중에 가장 눈에 띄는 요소는 리포지토리에 있는 아티팩트의 전체 크기입니다. 리포지토리의 누적 기록과 같은 요소도 크기에 영향을 미칠 수 있습니다. 수년간의 기록과 브랜치가 담긴 리포지토리는 각각의 자산이 크지 않더라도 전체적으로는 매우 클 수 있습니다. 이러한 리포지토리를 더 간단하고 효율적으로 마이그레이션하기 위해 취할 수 있는 전략은 여러 가지가 있습니다. 예를 들어, 개발 기간이 긴 리포지토리를 복제할 때 얕은 복제 전략을 사용하거나, 대용량 바이너리 파일의 경우 델타 압축을 비활성화할 수 있습니다. 또 Git 설명서를 참조하여 옵션을 분석할 수 있으며, 이 항목에 포함된 샘플 스크립트 `incremental-repo-migration.py`를 사용하여 리포지토리를 마이그레이션하기 위한 증분 푸시를 설정 및 구성할 수도 있습니다.

다음 조건 중 하나 이상에 해당하면 증분 푸시를 구성하는 것이 좋을 수도 있습니다.

- 마이그레이션하려는 리포지토리에 5년 이상의 기록이 담겨 있습니다.
- 인터넷 연결이 간헐적인 끊김, 패킷 손실, 느린 응답 속도, 기타 서비스 중단 등의 문제를 일으킵니다.
- 리포지토리의 전체 크기가 2GB를 넘으며 리포지토리 전체를 마이그레이션하려 합니다.
- 리포지토리에, 압축이 잘 되지 않는 대용량 아티팩트나 바이너리(예: 추적된 버전이 5개 이상인 대용량 이미지 파일)가 포함되어 있습니다.
- 이전에 CodeCommit으로 마이그레이션을 시도한 적이 있는데 “내부 서비스 오류” 메시지를 받았습니다.

상기 조건에 해당하지 않더라도 증분 푸시를 선택할 수 있습니다.

1단계: 필수 구성 요소를 설치하고 CodeCommit 리포지토리를 원격으로 추가

자체적인 필수 요소가 있는 고유의 사용자 지정 스크립트를 생성할 수 있습니다. 이 항목에 포함된 샘플을 사용하는 경우 다음을 수행해야 합니다.

- 필수 구성 요소를 설치합니다.
- 리포지토리를 로컬 컴퓨터에 복제하려면
- CodeCommit 리포지토리를, 마이그레이션하려는 리포지토리의 원격 리포지토리로 추가합니다.

incremental-repo-migration.py 실행 설정

1. 로컬 컴퓨터에 Python을 2.6 이상으로 설치합니다. 자세한 내용과 최신 버전은 Python 웹사이트를 참조하세요.
2. 동일한 컴퓨터에 GitPython을 설치합니다. GitPython은 Git 리포지토리와 상호작용하는 데 사용되는 Python 라이브러리입니다. 자세한 내용은 GitPython 설명서를 참조하세요.
3. git clone --mirror 명령을 사용하여, 마이그레이션하려는 리포지토리를 로컬 컴퓨터로 복제합니다. 터미널(Linux, macOS, Unix) 또는 명령 프롬프트(Windows)에서 git clone --mirror 명령을 사용하여 해당 리포지토리의 로컬 리포지토리를 생성합니다. 이때 로컬 리포지토리를 생성할 디렉터리도 생성합니다. 예를 들어, URL이 *https://example.com/my-repo/*이고 이름이 *MyMigrationRepo*인 Git 리포지토리를 *my-repo*라는 디렉터리에 복제하려면 다음과 같이 합니다.

   ```
   git clone --mirror https://example.com/my-repo/MyMigrationRepo.git my-repo
   ```

   다음과 비슷한 출력이 표시될 것입니다. 이는 리포지토리가 my-repo라는 이름의 베어 로컬 리포지토리에 복제되었음을 나타냅니다.

   ```
   Cloning into bare repository 'my-repo'...
   remote: Counting objects: 20, done.
   remote: Compressing objects: 100% (17/17), done.
   remote: Total 20 (delta 5), reused 15 (delta 3)
   Unpacking objects: 100% (20/20), done.
   Checking connectivity... done.
   ```

4. 방금 복제한 리포지토리의 로컬 리포지토리로 디렉터리를 변경합니다(예: *my-repo*). 해당 디렉터리에서 git remote add ***DefaultRemoteName RemoteRepositoryURL*** 명령을 사용하여 CodeCommit 리포지토리를 로컬 리포지토리의 원격 리포지토리로 추가합니다.

> **Note**
>
> 대규모 리포지토리를 푸시할 때는 HTTPS 대신 SSH를 사용하는 것이 좋습니다. 대규모 변경 사항, 대량의 변경 사항 또는 대규모 리포지토리 푸시할 때는 장시간 실행 중인 HTTPS 연결이 네트워킹 문제나 방화벽 설정으로 인해 조기에 종료되는 경우가 많습니다. SSH를 사용하는 CodeCommit 설정에 대해 자세히 알아보려면 Linux, macOS, Unix에서 SSH 연결 또는 Windows에서 SSH 연결 섹션을 참조하세요.

예를 들어, 다음 명령을 사용하여 MyDestinationRepo라는 CodeCommit 리포지토리의 SSH 엔드포인트를 `codecommit`라는 이름의 원격 리포지토리로 추가합니다.

```
git remote add codecommit ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/
MyDestinationRepo
```

> **Tip**
>
> 이는 복제이기 때문에, 기본 원격 이름인 (`origin`)은 이미 사용되고 있습니다. 다른 원격 이름을 사용해야 합니다. 예제에서는 `codecommit`을 사용하지만, 원하는 이름을 사용해도 됩니다. git remote show 명령을 사용하여 로컬 리포지토리에 설정된 원격 목록을 검토합니다.

5. git remote -v 명령을 사용하여 로컬 리포지토리의 페치 및 푸시 설정을 표시하고 올바르게 설정되었는지 확인합니다. 예시:

```
codecommit  ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDestinationRepo
 (fetch)
codecommit  ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDestinationRepo
 (push)
```

> **Tip**
> 다른 원격 리포지토리의 페치 및 푸시 항목(예: origin 항목)이 계속 표시되면, git remote set-url --delete 명령을 사용하여 제거합니다.

2단계: 증분 마이그레이션에 사용할 스크립트 생성

이 단계는 `incremental-repo-migration.py` 샘플 스크립트를 사용하고 있다는 가정하에 작성되었습니다.

1. 텍스트 편집기를 열고 샘플 스크립트의 내용을 빈 문서에 붙여 넣습니다.
2. 문서를 문서 디렉터리(로컬 리포지토리의 작업 디렉터리가 아님)에 저장하고 이름을 `incremental-repo-migration.py`으로 지정합니다. 선택한 디렉터리는 로컬 환경 또는 경로 변수에 구성된 디렉터리여야 합니다. 그래야 명령줄이나 터미널에서 Python 스크립트를 실행할 수 있습니다.

3단계: 스크립트 실행 및 CodeCommit으로 증분 마이그레이션

이제 `incremental-repo-migration.py` 스크립트를 만들었으니, 이를 사용하여 로컬 리포지토리를 CodeCommit 리포지토리로 증분 마이그레이션할 수 있습니다. 기본적으로 스크립트는 1,000개의 커밋을 일괄적으로 푸시하며, 자신이 실행되는 디렉터리의 Git 설정을 로컬 리포지토리와 원격 리포지토리의 설정으로 사용하려고 시도합니다. 필요한 경우 `incremental-repo-migration.py`에 포함된 옵션을 사용하여 다른 설정을 구성할 수 있습니다.

1. 터미널 또는 명령 프롬프트에서, 마이그레이션하려는 로컬 리포지토리로 디렉터리를 변경합니다.
2. 해당 디렉터리에서 다음 명령을 실행합니다.

```
python incremental-repo-migration.py
```

3. 스크립트가 실행되며, 터미널 또는 명령 프롬프트에 진행 상황이 표시됩니다. 일부 대형 리포지토리는 진행 상황이 느리게 표시됩니다. 단일 푸시가 세 번 실패하면 스크립트는 중단됩니다. 그런 다음 스크립트를 다시 실행하면 실패한 배치부터 다시 시작할 수 있습니다. 스크립트 재실행은 모든 푸시가 성공하고 마이그레이션이 완료될 때까지 계속할 수 있습니다.

> **Tip**
>
> `-l` 및 `-r` 옵션을 활용해 로컬 및 원격 설정을 지정하기만 하면 모든 디렉터리에서 `incremental-repo-migration.py`를 실행할 수 있습니다. 예를 들어, 임의의 디렉터리의 스크립트를 사용하여, /tmp/*my-repo*에 위치한 로컬 리포지토리를 *codecommit*이라는 원격 리포지토리로 마이그레이션하려면 다음과 같이 합니다.
>
> ```
> python incremental-repo-migration.py -l "/tmp/my-repo" -r "codecommit"
> ```
>
> `-b` 옵션을 사용하여, 증분 푸시할 때 사용되는 기본 배치 크기를 변경하는 것이 좋을 수 있습니다. 예를 들어, 자주 변경되는 대용량 바이너리 파일이 있는 리포지토리를 정기적으로 푸시하고 네트워크 대역폭이 제한된 장소에서 작업하는 경우, `-b` 옵션을 사용하여 배치 크기를 1,000이 아닌 500으로 변경하는 것이 좋을 수 있습니다. 예시:
>
> ```
> python incremental-repo-migration.py -b 500
> ```
>
> 이렇게 하면 로컬 리포지토리를 커밋 500개씩 묶어서 증분 푸시합니다. 리포지토리를 마이그레이션할 때 배치 크기를 다시 변경하려는 경우(예: 시도를 실패한 후에 배치 크기를 줄이기로 결정했을 때), `-b`로 배치 크기를 재설정하기 전에 `-c` 옵션으로 배치 태그를 제거해야 합니다.
>
> ```
> python incremental-repo-migration.py -c
> python incremental-repo-migration.py -b 250
> ```

> **Important**
>
> 실패 후 스크립트를 다시 실행하려면 이 `-c` 옵션은 사용하지 않습니다. `-c` 옵션은 커밋을 일괄 처리하는 데 사용된 태그를 제거합니다. 배치 크기를 변경하고 다시 시작하려는 경우나 스크립트를 더 이상 사용하지 않으려는 경우에만 `-c` 옵션을 사용합니다.

부록: 샘플 스크립트 `incremental-repo-migration.py`

편의를 위해, 리포지토리를 증분 푸시해 주는 샘플 Python 스크립트 `incremental-repo-migration.py`를 개발했습니다. 본 샘플은 오픈 소스이며 있는 그대로 제공됩니다.

```
# Copyright 2015 Amazon.com, Inc. or its affiliates. All Rights Reserved. Licensed
 under the Amazon Software License (the "License").
# You may not use this file except in compliance with the License. A copy of the
 License is located at
#    http://aws.amazon.com/asl/
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
 KIND, express or implied. See the License for
# the specific language governing permissions and limitations under the License.

#!/usr/bin/env python

import os
import sys
from optparse import OptionParser
from git import Repo, TagReference, RemoteProgress, GitCommandError


class PushProgressPrinter(RemoteProgress):
    def update(self, op_code, cur_count, max_count=None, message=""):
        op_id = op_code & self.OP_MASK
        stage_id = op_code & self.STAGE_MASK
        if op_id == self.WRITING and stage_id == self.BEGIN:
            print("\tObjects: %d" % max_count)


class RepositoryMigration:
    MAX_COMMITS_TOLERANCE_PERCENT = 0.05
    PUSH_RETRY_LIMIT = 3
    MIGRATION_TAG_PREFIX = "codecommit_migration_"

    def migrate_repository_in_parts(
        self, repo_dir, remote_name, commit_batch_size, clean
    ):
        self.next_tag_number = 0
        self.migration_tags = []
        self.walked_commits = set()
        self.local_repo = Repo(repo_dir)
        self.remote_name = remote_name
        self.max_commits_per_push = commit_batch_size
        self.max_commits_tolerance = (
            self.max_commits_per_push * self.MAX_COMMITS_TOLERANCE_PERCENT
        )
```

```
        try:
            self.remote_repo = self.local_repo.remote(remote_name)
            self.get_remote_migration_tags()
        except (ValueError, GitCommandError):
            print(
                "Could not contact the remote repository. The most common reasons for
this error are that the name of the remote repository is incorrect, or that you do not
have permissions to interact with that remote repository."
            )
            sys.exit(1)

        if clean:
            self.clean_up(clean_up_remote=True)
            return

        self.clean_up()

        print("Analyzing repository")
        head_commit = self.local_repo.head.commit
        sys.setrecursionlimit(max(sys.getrecursionlimit(), head_commit.count()))

        # tag commits on default branch
        leftover_commits = self.migrate_commit(head_commit)
        self.tag_commits([commit for (commit, commit_count) in leftover_commits])

        # tag commits on each branch
        for branch in self.local_repo.heads:
            leftover_commits = self.migrate_commit(branch.commit)
            self.tag_commits([commit for (commit, commit_count) in leftover_commits])

        # push the tags
        self.push_migration_tags()

        # push all branch references
        for branch in self.local_repo.heads:
            print("Pushing branch %s" % branch.name)
            self.do_push_with_retries(ref=branch.name)

        # push all tags
        print("Pushing tags")
        self.do_push_with_retries(push_tags=True)

        self.get_remote_migration_tags()
        self.clean_up(clean_up_remote=True)
```

```
        print("Migration to CodeCommit was successful")

    def migrate_commit(self, commit):
        if commit in self.walked_commits:
            return []

        pending_ancestor_pushes = []
        commit_count = 1

        if len(commit.parents) > 1:
            # This is a merge commit
            # Ensure that all parents are pushed first
            for parent_commit in commit.parents:
                pending_ancestor_pushes.extend(self.migrate_commit(parent_commit))
        elif len(commit.parents) == 1:
            # Split linear history into individual pushes
            next_ancestor, commits_to_next_ancestor = self.find_next_ancestor_for_push(
                commit.parents[0]
            )
            commit_count += commits_to_next_ancestor
            pending_ancestor_pushes.extend(self.migrate_commit(next_ancestor))

        self.walked_commits.add(commit)

        return self.stage_push(commit, commit_count, pending_ancestor_pushes)

    def find_next_ancestor_for_push(self, commit):
        commit_count = 0

        # Traverse linear history until we reach our commit limit, a merge commit, or
an initial commit
        while (
            len(commit.parents) == 1
            and commit_count < self.max_commits_per_push
            and commit not in self.walked_commits
        ):
            commit_count += 1
            self.walked_commits.add(commit)
            commit = commit.parents[0]

        return commit, commit_count

    def stage_push(self, commit, commit_count, pending_ancestor_pushes):
```

```
        # Determine whether we can roll up pending ancestor pushes into this push
        combined_commit_count = commit_count + sum(
            ancestor_commit_count
            for (ancestor, ancestor_commit_count) in pending_ancestor_pushes
        )

        if combined_commit_count < self.max_commits_per_push:
            # don't push anything, roll up all pending ancestor pushes into this
pending push
            return [(commit, combined_commit_count)]

        if combined_commit_count <= (
            self.max_commits_per_push + self.max_commits_tolerance
        ):
            # roll up everything into this commit and push
            self.tag_commits([commit])
            return []

        if commit_count >= self.max_commits_per_push:
            # need to push each pending ancestor and this commit
            self.tag_commits(
                [
                    ancestor
                    for (ancestor, ancestor_commit_count) in pending_ancestor_pushes
                ]
            )
            self.tag_commits([commit])
            return []

        # push each pending ancestor, but roll up this commit
        self.tag_commits(
            [ancestor for (ancestor, ancestor_commit_count) in pending_ancestor_pushes]
        )
        return [(commit, commit_count)]

    def tag_commits(self, commits):
        for commit in commits:
            self.next_tag_number += 1
            tag_name = self.MIGRATION_TAG_PREFIX + str(self.next_tag_number)

            if tag_name not in self.remote_migration_tags:
                tag = self.local_repo.create_tag(tag_name, ref=commit)
                self.migration_tags.append(tag)
            elif self.remote_migration_tags[tag_name] != str(commit):
```

```
                print(
                    "Migration tags on the remote do not match the local tags. Most
likely your batch size has changed since the last time you ran this script. Please run
this script with the --clean option, and try again."
                )
                sys.exit(1)

    def push_migration_tags(self):
        print("Will attempt to push %d tags" % len(self.migration_tags))
        self.migration_tags.sort(
            key=lambda tag: int(tag.name.replace(self.MIGRATION_TAG_PREFIX, ""))
        )
        for tag in self.migration_tags:
            print(
                "Pushing tag %s (out of %d tags), commit %s"
                % (tag.name, self.next_tag_number, str(tag.commit))
            )
            self.do_push_with_retries(ref=tag.name)

    def do_push_with_retries(self, ref=None, push_tags=False):
        for i in range(0, self.PUSH_RETRY_LIMIT):
            if i == 0:
                progress_printer = PushProgressPrinter()
            else:
                progress_printer = None

            try:
                if push_tags:
                    infos = self.remote_repo.push(tags=True, progress=progress_printer)
                elif ref is not None:
                    infos = self.remote_repo.push(
                        refspec=ref, progress=progress_printer
                    )
                else:
                    infos = self.remote_repo.push(progress=progress_printer)

                success = True
                if len(infos) == 0:
                    success = False
                else:
                    for info in infos:
                        if (
                            info.flags & info.UP_TO_DATE
                            or info.flags & info.NEW_TAG
```

```
                            or info.flags & info.NEW_HEAD
                        ):
                            continue
                        success = False
                        print(info.summary)

                if success:
                    return
            except GitCommandError as err:
                print(err)

        if push_tags:
            print("Pushing all tags failed after %d attempts" %
(self.PUSH_RETRY_LIMIT))
        elif ref is not None:
            print("Pushing %s failed after %d attempts" % (ref, self.PUSH_RETRY_LIMIT))
            print(
                "For more information about the cause of this error, run the following
command from the local repo: 'git push %s %s'"
                % (self.remote_name, ref)
            )
        else:
            print(
                "Pushing all branches failed after %d attempts"
                % (self.PUSH_RETRY_LIMIT)
            )
        sys.exit(1)

    def get_remote_migration_tags(self):
        remote_tags_output = self.local_repo.git.ls_remote(
            self.remote_name, tags=True
        ).split("\n")
        self.remote_migration_tags = dict(
            (tag.split()[1].replace("refs/tags/", ""), tag.split()[0])
            for tag in remote_tags_output
            if self.MIGRATION_TAG_PREFIX in tag
        )

    def clean_up(self, clean_up_remote=False):
        tags = [
            tag
            for tag in self.local_repo.tags
            if tag.name.startswith(self.MIGRATION_TAG_PREFIX)
        ]
```

```
        # delete the local tags
        TagReference.delete(self.local_repo, *tags)

        # delete the remote tags
        if clean_up_remote:
            tags_to_delete = [":" + tag_name for tag_name in
 self.remote_migration_tags]
            self.remote_repo.push(refspec=tags_to_delete)


parser = OptionParser()
parser.add_option(
    "-l",
    "--local",
    action="store",
    dest="localrepo",
    default=os.getcwd(),
    help="The path to the local repo. If this option is not specified, the script will
 attempt to use current directory by default. If it is not a local git repo, the script
 will fail.",
)
parser.add_option(
    "-r",
    "--remote",
    action="store",
    dest="remoterepo",
    default="codecommit",
    help="The name of the remote repository to be used as the push or migration
 destination. The remote must already be set in the local repo ('git remote add ...').
 If this option is not specified, the script will use 'codecommit' by default.",
)
parser.add_option(
    "-b",
    "--batch",
    action="store",
    dest="batchsize",
    default="1000",
    help="Specifies the commit batch size for pushes. If not explicitly set, the
 default is 1,000 commits.",
)
parser.add_option(
    "-c",
    "--clean",
```

```
    action="store_true",
    dest="clean",
    default=False,
    help="Remove the temporary tags created by migration from both the local repo
 and the remote repository. This option will not do any migration work, just cleanup.
 Cleanup is done automatically at the end of a successful migration, but not after a
 failure so that when you re-run the script, the tags from the prior run can be used to
 identify commit batches that were not pushed successfully.",
)

(options, args) = parser.parse_args()

migration = RepositoryMigration()
migration.migrate_repository_in_parts(
    options.localrepo, options.remoterepo, int(options.batchsize), options.clean
)
```

의 보안 AWS CodeCommit

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. 에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램 AWS 제공 범위 내 서비스규정 준수 프로그램 제공 범위 내 서비스를 AWS CodeCommit참조하세요.
- 클라우드의 보안 - 사용자의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 귀하는 귀사의 데이터 민감도, 귀사의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 CodeCommit 사용 시 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 주제에서는 보안 및 규정 준수 목표를 충족하도록 CodeCommit을 구성하는 방법을 보여 줍니다. 또한 CodeCommit 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법을 알아봅니다.

주제

- 의 데이터 보호 AWS CodeCommit
- 에 대한 자격 증명 및 액세스 관리 AWS CodeCommit
- 의 복원력 AWS CodeCommit
- 의 인프라 보안 AWS CodeCommit

의 데이터 보호 AWS CodeCommit

관리형 서비스인는 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 AWS 환경을 설계하려면 Security Pillar AWS Well-Architected Framework의 인프라 보호를 참조하세요.

AWS 에서 게시한 API 호출을 사용하여 네트워크를 통해에 액세스합니다. 고객은 다음을 지원해야 합니다.

- Transport Layer Security(TLS) TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군 Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 위탁자와 관련된 보안 암호 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 자격 증명을 생성하여 요청에 서명할 수 있습니다.

CodeCommit 리포지토리는 미사용 시 자동으로 암호화됩니다. 고객의 조치는 필요하지 않습니다. CodeCommit은 리포지토리의 전송 중 데이터도 암호화합니다. CodeCommit 리포지토리에서는 HTTPS 프로토콜이나 SSH 프로토콜을 사용할 수 있으며 양쪽을 모두 사용하는 것도 가능합니다. 자세한 내용은 에 대한 설정 AWS CodeCommit 단원을 참조하세요. 사용자는 CodeCommit 리포지토리에 대한 크로스 계정 액세스를 구성할 수도 있습니다.

주제

- AWS Key Management Service 및 AWS CodeCommit 리포지토리 암호화
- 교체 자격 증명을 사용하여 AWS CodeCommit 리포지토리에 연결

AWS Key Management Service 및 AWS CodeCommit 리포지토리 암호화

CodeCommit 리포지토리의 데이터는 전송 중이거나 미사용 중일 때 암호화됩니다. CodeCommit 리포지토리 내에 데이터가 푸시(예: git push 호출)될 때는 CodeCommit이 해당 데이터를 리포지토리 저장 시점에 암호화합니다. CodeCommit 리포지토리에서 데이터가 풀(예: git pull 호출)될 때는 CodeCommit이 해당 데이터를 해독한 다음 호출자에게 전송합니다. 푸시 또는 풀 요청과 연결된 IAM 사용자가에 의해 인증되었다고 가정합니다 AWS. 전송하거나 수신한 데이터는 HTTPS 또는 네트워크 프로토콜을 암호화한 SSH를 사용하여 전송됩니다.

AWS 관리형 키 또는 고객 관리형 키를 사용하여 리포지토리의 데이터를 암호화하고 해독할 수 있습니다. 고객 관리형 키와 AWS 관리형 키의 차이점에 대한 자세한 내용은 고객 관리형 키와 AWS 관리형 키를 참조하세요. 고객 관리형 키를 지정하지 않으면 CodeCommit은 리포지토리의 데이터를 암호화하고 복호화하는 데 AWS 관리형 키 를 사용합니다. 이 AWS 관리형 키 는 AWS 계정에서 자동으로 생성됩니다. Amazon Web Services 계정의 새에 CodeCommit AWS 리전 리포지토리를 처음 생성할 때 고객 관리형 키를 지정하지 않으면 CodeCommit은 동일한 AWS 리전 in AWS 관리형 키 ()에 AWS Key

Management Service (`aws/codecommit`키)를 생성합니다AWS KMS. 이 `aws/codecommit` 키는 오로지 CodeCommit에만 사용됩니다. 이 키는 Amazon Web Services 계정에 저장됩니다. 지정한 내용에 따라 CodeCommit은 고객 관리형 키 또는 AWS 관리형 키 를 사용하여 리포지토리의 데이터를 암호화하고 해독합니다.

> **Important**
>
> CodeCommit은 리포지토리의 데이터를 암호화하고 복호화하는 데 사용되는 AWS KMS 키에 대해 다음 AWS KMS 작업을 수행합니다. 를 사용하는 경우 AWS 관리형 키사용자는 이러한 작업에 대한 명시적 권한이 필요하지 않지만 `aws/codecommit` 키에 대해 이러한 작업을 거부하는 연결된 정책이 없어야 합니다. AWS 계정 ID가 해당 키의 정책 보안 주체로 설정된 고객 관리형 키를 사용하는 경우 이러한 권한을 로 명시적으로 설정해야 합니다`allow`. 특히 첫 번째 리포지토리를 생성하고 리포지토리의 키를 업데이트할 때를 사용하는 `deny` 경우 다음 권한 중 하나를 로 설정하지 않아야 하며 AWS 관리형 키, 정책 보안 주체와 함께 고객 관리형 키를 사용하는 `allow` 경우 로 설정해야 합니다.
>
> - `"kms:Encrypt"`
> - `"kms:Decrypt"`
> - `"kms:ReEncrypt"` (상황에 따라 `kms:ReEncryptFrom`와 `kms:ReEncryptTo` 또는 `kms:ReEncrypt*`가 거부로 설정되지 않아야 함)
> - `"kms:GenerateDataKey"`
> - `"kms:GenerateDataKeyWithoutPlaintext"`
> - `"kms:DescribeKey"`

자체 고객 관리형 키를 사용하려면 리포지토리 AWS 리전 가 있는에서 키를 사용할 수 있어야 합니다. CodeCommit은 단일 및 다중 리전 고객 관리형 키 사용을 모두 지원합니다. 모든 키 구성 요소 오리진 유형이 지원되지만 기본 KMS 옵션을 사용하는 것이 좋습니다. 외부 키 저장소 옵션을 사용하는 고객은 저장소 공급자로부터 지연이 발생할 수 있습니다. 또한 CodeCommit은 고객 관리형 키에 대해 다음과 같은 요구 사항을 가지고 있습니다:

- CodeCommit은 대칭 키 사용만 지원합니다.
- 키 사용 유형은 암호화 및 복호화로 설정해야 합니다.

고객 관리형 키 생성에 대한 자세한 내용은 개념 및 키 생성을 참조하세요.

CodeCommit에서 AWS 관리형 키 생성된에 대한 정보를 보려면 다음을 수행합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms://https://https://https://https://https://https://https://https://https:// AWS Key Management Service AWS KMS https://https://http
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 서비스 탐색 창에서 AWS 관리형 키를 선택합니다. 키를 검토할 AWS 리전 에 로그인했는지 확인합니다.
4. 암호화 키 목록에서 별칭이 aws/codecommit AWS 관리형 키 인를 선택합니다. 에 대한 기본 정보가 AWS 소유 키 표시됩니다.

이를 변경하거나 삭제할 수 없습니다 AWS 관리형 키.

암호화 알고리즘을 사용하여 리포지토리 데이터를 암호화하는 방법

CodeCommit은 데이터를 암호화하기 위해 두 가지 다른 접근 방식을 사용합니다. 6MB 미만의 개별 Git 객체는 데이터 무결성 검증을 제공하는 AES-GCM-256을 사용하여 암호화됩니다. 단일 Blob에 대해 6MB에서 최대 2GB 사이의 객체는 AES-CBC-256을 사용하여 암호화됩니다. CodeCommit은 항상 암호화 컨텍스트를 검증합니다.

암호화 컨텍스트

와 통합된 각 서비스는 암호화 및 복호화 작업 모두에 대한 암호화 컨텍스트를 AWS KMS 지정합니다. 암호화 컨텍스트는 데이터 무결성 확인을 위해 AWS KMS 에서 사용하는 추가적인 인증 정보입니다. 암호화 작업에 대해 지정하면 해독 작업에도 지정해야 합니다. 그렇지 않으면 암호화 해제가 실패합니다. CodeCommit은 암호화 컨텍스트에 CodeCommit 리포지토리 ID를 사용합니다. get-repository 명령이나 CodeCommit 콘솔을 사용하면 리포지토리 ID를 찾을 수 있습니다. AWS CloudTrail 로그에서 CodeCommit 리포지토리 ID를 검색하여 CodeCommit 리포지토리의 데이터를 암호화하거나 해독 AWS KMS 하기 위해 어떤 키에서 어떤 암호화 작업이 수행되었는지 파악합니다.

에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서를 AWS KMS참조하세요.

교체 자격 증명을 사용하여 AWS CodeCommit 리포지토리에 연결

사용자에게 IAM 사용자를 구성하거나 액세스 키 및 보안 키를 사용하지 않고도 AWS CodeCommit 리포지토리에 대한 액세스 권한을 부여할 수 있습니다. 페더레이션 보안 인증 정보 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 페더레이션 자격 증명이 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 관련 역할에 대한 자세한 내용은 IAM 사용 설명서의

Create a role for a third-party identity provider (federation)를 참조하세요. IAM Identity Center를 사용하는 경우, 권한 집합을 구성합니다. 인증 후 ID가 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 집합을 IAM의 역할과 연관짓습니다. 권한 집합에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 집합을 참조하세요. 또한 IAM 사용자가 별도의 Amazon Web Services 계정의 CodeCommit 리포지토리에 액세스할 수 있도록 역할 기반 액세스를 구성할 수도 있습니다(크로스 계정 액세스라고 하는 기술). 리포지토리에 대한 크로스 계정 액세스를 구성하는 연습은 역할을 사용하여 AWS CodeCommit 리포지토리에 대한 교차 계정 액세스 구성 단원을 참조하세요.

다음과 같은 방법으로 인증하려는(또는 인증해야 하는) 사용자에 대해 액세스를 구성할 수 있습니다.

- SAML(Security Assertion Markup Language)
- 멀티 팩터 인증(MFA)
- 연동
- Login with Amazon
- Amazon Cognito
- Facebook
- Google
- OpenID Connect(OIDC) 호환 ID 공급자

> **Note**
>
> 다음 정보는 CodeCommit 리포지토리에 연결하기 위해 git-remote-codecommit 또는 AWS CLI 보안 인증 도우미를 사용하는 경우에만 적용됩니다. CodeCommit에 대한 임시 액세스 또는 페더레이션 액세스에 권장되는 접근 방식은 git-remote-codecommit를 설정하는 것이므로, 이 항목에서는 해당 유틸리티를 사용하는 예제를 제공합니다. 자세한 내용은 git-remote-codecommit을 AWS CodeCommit 사용하여에 HTTPS 연결을 위한 설정 단계 단원을 참조하세요.
>
> SSH 또는 Git 보안 인증 정보와 HTTPS를 사용하여 CodeCommit 리포지토리를 교체 또는 임시 액세스 보안 인증 정보와 연결할 수 없습니다.

다음 요구 사항이 모두 충족될 경우에는 이 단계들을 완료할 필요가 없습니다.

- Amazon EC2 인스턴스에 로그인한 경우.
- 자격 AWS CLI 증명 헬퍼와 함께 Git 및 HTTPS를 사용하여 Amazon EC2 인스턴스에서 CodeCommit 리포지토리로 연결합니다.

- Amazon EC2 인스턴스에, AWS CLI 자격 증명 헬퍼를 사용하여 Linux, macOS 또는 Unix에서 HTTPS 연결 또는 자격 AWS CLI 증명 헬퍼를 사용하여 Windows에서 HTTPS 연결에 설명된 액세스 권한을 포함하는 IAM 인스턴스 프로파일이 연결되어 있는 경우.
- AWS CLI 자격 증명 헬퍼를 사용하여 Linux, macOS 또는 Unix에서 HTTPS 연결 또는 자격 AWS CLI 증명 헬퍼를 사용하여 Windows에서 HTTPS 연결에서 설명한 대로 Amazon EC2 인스턴스에서 Git 보안 인증 도우미를 올바르게 설치하여 구성한 경우.

이전 요구 사항을 충족하는 Amazon EC2 인스턴스가 임시 액세스 보안 인증 정보를 CodeCommit에 자동으로 전달해 주도록 이미 설정되어 있는 경우.

> **Note**
>
> Amazon EC2 인스턴스에서 git-remote-codecommit을 구성하고 사용할 수 있습니다.

사용자에게 CodeCommit 리포지토리에 대한 임시 액세스 권한을 제공하려면 다음 단계를 완료합니다.

1단계: 필수 구성 요소 완성

설정 단계를 완료하여 사용자에게 교체 보안 인증 정보를 사용하여 CodeCommit 리포지토리에 액세스할 수 있는 권한을 제공합니다.

- 크로스 계정 액세스의 경우에는 연습: IAM 역할을 사용하는 Amazon Web Services 계정 간 액세스 권한 위임 및 역할을 사용하여 AWS CodeCommit 리포지토리에 대한 교차 계정 액세스 구성 단원을 참조하세요.
- SAML 및 페더레이션의 경우 조직의 인증 시스템을 사용하여 AWS 리소스에 대한 액세스 권한 부여 및 AWS STS SAML 2.0 기반 페더레이션 정보를 참조하세요.
- MFA의 경우에는 AWS에서 다중 인증(MFA) 디바이스 사용하기 및 IAM 사용자를 위해 액세스 가능한 임시 보안 인증 정보 생성하기를 참조하세요.
- Login with Amazon, Amazon Cognito, Facebook, Google 또는 OIDC 호환 자격 증명 공급자의 경우 AWS STS 웹 자격 증명 페더레이션 정보를 참조하세요.

AWS CodeCommit에 대한 인증 및 액세스 제어의 정보를 활용하여 사용자에게 부여할 CodeCommit 권한을 지정합니다.

2단계: 역할 이름 또는 액세스 보안 인증 정보 가져오기

사용자가 역할을 맡아 리포지토리에 액세스하도록 하려면 해당 역할의 Amazon 리소스 이름(ARN)을 사용자에게 제공합니다. 그렇지 않으면 액세스 설정 방법에 따라 사용자는 다음 방법 중 하나를 사용하여 교체 보안 인증 정보를 얻을 수 있습니다.

- 교차 계정 액세스를 위해 the AWS CLI assume-role 명령을 호출하거나 AWS STS AssumeRole API를 호출합니다.
- SAML의 경우 the AWS CLI assume-role-with-saml 명령 또는 AWS STS AssumeRoleWithSAML API를 호출합니다.
- 페더레이션의 경우 the AWS CLI assume-role 또는 get-federation-token 명령 또는 AWS STS AssumeRole 또는 GetFederationToken APIs.
- MFA의 경우 the AWS CLI get-session-token 명령 또는 AWS STS GetSessionToken API를 호출합니다.
- Login with Amazon, Amazon Cognito, Facebook, Google 또는 OIDC 호환 자격 증명 공급자의 경우 AWS CLI assume-role-with-web-identity 명령 또는 AWS STS AssumeRoleWithWebIdentity API를 호출합니다.

3단계: git-remote-codecommit 설치 및 구성 AWS CLI

AWS CLI에서 git-remote-codecommit을 설치하고 프로필을 구성하여 액세스 보안 인증 정보를 사용하도록 로컬 컴퓨터를 구성해야 합니다.

1. 설정 의 지침에 따라 AWS CLI를 설정합니다. aws configure 명령을 사용하여 하나 이상의 프로필을 구성합니다. 교체 보안 인증 정보를 사용하여 CodeCommit 리포지토리에 연결할 때 사용할 명명된 프로필을 생성하는 것이 좋습니다.
2. 다음 방법 중 하나를 사용하여 자격 증명을 사용자의 AWS CLI 명명된 프로필과 연결할 수 있습니다.
   - CodeCommit에 액세스할 역할을 수임하는 경우 해당 역할을 수임하는 데 필요한 정보를 사용하여 명명된 프로파일을 구성합니다. 예를 들어, Amazon Web Services 계정 111111111111에서 *CodeCommitAccess*라는 역할을 수임하려는 경우 다른 AWS 리소스로 작업할 때 사용할 기본 프로파일과 해당 역할을 수임할 때 사용할 명명된 프로파일을 구성할 수 있습니다. 다음 명령은 *CodeCommitAccess*라는 역할을 맡은 *CodeAccess*라는 프로필을 생성합니다. 사용자 이름 *Maria_Garcia*는 세션과 연결되며 기본 프로필은 자격 AWS 증명의 소스로 설정됩니다.

```
aws configure set role_arn arn:aws:iam::111111111111:role/CodeCommitAccess --
profile CodeAccess
aws configure set source_profile default --profile CodeAccess
aws configure set role_session_name "Maria_Garcia" --profile CodeAccess
```

변경 사항을 확인하려면 `~/.aws/config` 파일(Linux용) 또는 `%UserProfile%.aws\config` 파일(Windows용)을 수동으로 보거나 편집하고 명명된 프로필 아래의 정보를 검토합니다. 예를 들어 파일은 다음과 비슷할 것입니다.

```
[default]
region = us-east-1
output = json

[profile CodeAccess]
source_profile = default
role_session_name = Maria_Garcia
role_arn = arn:aws:iam::111111111111:role/CodeCommitAccess
```

명명된 프로필을 구성한 후에는 명명된 프로필을 사용하여 git-remote-codecommit 유틸리티로 CodeCommit 리포지토리를 복제할 수 있습니다. 예를 들어, *MyDemoRepo*라는 리포지토리를 복제하려면 다음과 같이 합니다.

```
git clone codecommit://CodeAccess@MyDemoRepo
```

- 웹 자격 증명 페더레이션 및 OpenID Connect(OIDC)를 사용하는 경우 임시 자격 증명을 새로 고치기 위해 사용자를 대신하여 AWS Security Token Service (AWS STS) `AssumeRoleWithWebIdentity` API를 호출하는 명명된 프로파일을 구성합니다. aws configure set 명령을 사용하거나 `~/.aws/credentials` 파일(Linux의 경우) 또는 `%UserProfile%.aws\credentials` 파일(Windows의 경우)을 수동으로 편집하여 필요한 설정 값이 있는 AWS CLI 명명된 프로파일을 추가합니다. 예를 들어, *CodeCommitAccess* 역할을 수임하고 웹 자격 증명 토큰 파일 ~/*my-credentials/my-token-file*을 사용하는 프로필을 생성하려면 다음과 같이 합니다.

```
[CodeCommitWebIdentity]
role_arn = arn:aws:iam::111111111111:role/CodeCommitAccess
web_identity_token_file=~/my-credentials/my-token-file
role_session_name = Maria_Garcia
```

자세한 내용은 AWS Command Line Interface 사용 설명서에서 AWS Command Line Interface구성하기 및 AWS CLI에서 IAM 역할 사용하기를 참조하세요.

4단계: CodeCommit 리포지토리 액세스

사용자가 리포지토리에 연결의 지침에 따라 CodeCommit 리포지토리에 연결한 경우, git-remote-codecommit에서 제공한 확장된 기능과 Git을 사용해 git clone, git push, git pull을 직접 호출하여, 액세스 권한이 있는 CodeCommit 리포지토리를 복제하고 푸시하고 풀할 수 있습니다. 예를 들어 리포지토리를 복제하려면 다음과 같이 합니다.

```
git clone codecommit://CodeAccess@MyDemoRepo
```

Git 커밋, 푸시 및 풀 명령은 일반 Git 구문을 사용합니다.

사용자가를 사용하고 교체 액세스 자격 증명과 연결된 AWS CLI 명명된 프로파일을 AWS CLI 지정하면 해당 프로파일로 범위가 지정된 결과가 반환됩니다.

에 대한 자격 증명 및 액세스 관리 AWS CodeCommit

AWS Identity and Access Management (IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어할 수 AWS 서비스 있도록 도와주는 입니다. IAM 관리자는 누가 인증(로그인) 및 권한(권한 있음)을 받아 CodeCommit 리소스를 사용할 수 있는 있는지를 제어합니다. IAM은 추가 비용 없이 사용할 수 AWS 서비스 있는 입니다.

주제

- 대상
- ID를 통한 인증
- 정책을 사용하여 액세스 관리
- AWS CodeCommit에 대한 인증 및 액세스 제어
- AWS CodeCommit 에서 IAM을 사용하는 방법
- CodeCommit 리소스 기반 정책
- CodeCommit 태그 기반 권한 부여
- CodeCommit IAM 역할

- AWS CodeCommit 자격 증명 기반 정책 예제
- 자격 AWS CodeCommit 증명 및 액세스 문제 해결

대상

AWS Identity and Access Management (IAM)를 사용하는 방법은 CodeCommit에서 수행하는 작업에 따라 다릅니다.

서비스 사용자 - CodeCommit 서비스를 사용하여 작업을 수행하는 경우 필요한 보안 인증 정보와 권한을 관리자가 제공합니다. 더 많은 CodeCommit 기능을 사용하여 작업을 수행하게 되면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방법을 이해하면 관리자에게 올바른 권한을 요청하는 데 도움이 됩니다. CodeCommit의 기능에 액세스할 수 없는 경우 자격 AWS CodeCommit 증명 및 액세스 문제 해결 섹션을 참조하세요.

서비스 관리자 – 회사 내 CodeCommit 리소스를 책임지고 있는 경우 CodeCommit에 대한 전체 액세스 권한을 가지고 있을 것입니다. 서비스 관리자는 서비스 사용자가 액세스해야 하는 CodeCommit 기능과 리소스를 결정합니다. 그런 다음 IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해하세요. 회사에서 CodeCommit에 IAM을 어떻게 사용할 수 있는지 자세히 알아보려면 AWS CodeCommit 에서 IAM을 사용하는 방법 섹션을 참조하세요.

IAM 관리자 - IAM 관리자인 경우 CodeCommit에 대한 액세스 권한 관리 정책을 작성하는 방법에 대해 자세히 알아보고 싶을 수 있습니다. IAM에서 사용할 수 있는 CodeCommit ID 기반 정책 예제를 보려면 AWS CodeCommit 자격 증명 기반 정책 예제 섹션을 참조하세요.

ID를 통한 인증

인증은 자격 증명 AWS 으로에 로그인하는 방법입니다. IAM 사용자 또는 AWS 계정 루트 사용자 IAM 역할을 수임하여 로 인증(로그인 AWS)되어야 합니다.

자격 증명 소스를 통해 제공된 자격 증명을 사용하여 페더레이션 자격 증명 AWS 으로에 로그인할 수 있습니다. AWS IAM Identity Center (IAM Identity Center) 사용자, 회사의 Single Sign-On 인증 및 Google 또는 Facebook 자격 증명이 페더레이션 자격 증명의 예입니다. 페더레이션형 ID로 로그인할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 페더레이션을 사용하여 AWS 에 액세스하면 간접적으로 역할을 수임하게 됩니다.

사용자 유형에 따라 AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. 로그인에 대한 자세한 내용은 AWS 로그인 사용 설명서의에 로그인하는 방법을 AWS참조하세요. AWS 계정

AWS 프로그래밍 방식으로에 액세스하는 경우는 자격 증명을 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트(SDK)와 명령줄 인터페이스(CLI)를 AWS 제공합니다. AWS 도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. 권장 방법을 사용하여 요청에 직접 서명하는 자세한 방법은 IAM 사용 설명서에서 API 요청용AWS Signature Version 4를 참조하세요.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어는 다중 인증(MFA)을 사용하여 계정의 보안을 강화할 것을 AWS 권장합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 다중 인증 및 IAM 사용 설명서에서 IAM의AWS 다중 인증을 참조하세요.

AWS 계정 루트 사용자

를 생성할 때 계정의 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 하나의 로그인 자격 증명으로 AWS 계정시작합니다. 이 자격 증명을 AWS 계정 루트 사용자라고 하며 계정을 생성하는 데 사용한 이메일 주소와 암호로 로그인하여 액세스합니다. 일상적인 작업에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명을 보호하고 루트 사용자만 수행할 수 있는 작업을 수행하는 데 사용합니다. 루트 사용자로 로그인해야 하는 전체 작업 목록은 IAM 사용 설명서의 루트 사용자 자격 증명이 필요한 작업을 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 사용자 또는 애플리케이션에 대한 특정 권한이 AWS 계정 있는 내의 자격 증명입니다. 가능하면 암호 및 액세스 키와 같은 장기 자격 증명이 있는 IAM 사용자를 생성하는 대신 임시 자격 증명을 사용하는 것이 좋습니다. 하지만 IAM 사용자의 장기 자격 증명이 필요한 특정 사용 사례가 있는 경우, 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 장기 보안 인증이 필요한 사용 사례의 경우, 정기적으로 액세스 키 교체를 참조하세요.

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

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 장기 자격 증명을 가지고 있지만, 역할은 임시 보안 인증만 제공합니다. 자세한 내용은 IAM 사용 설명서에서 IAM 사용자 사용 사례를 참조하세요.

IAM 역할

IAM 역할은 특정 권한이 AWS 계정 있는 내 자격 증명입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. 에서 IAM 역할을 일시적으로 AWS Management Console수임하려면 사용자에서

IAM 역할(콘솔)로 전환할 수 있습니다. 또는 AWS API 작업을 호출하거나 사용자 지정 URL을 AWS CLI 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 내용은 IAM 사용 설명서의 역할 수임 방법을 참조하세요.

임시 보안 인증이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 페더레이션 사용자 액세스 - 페더레이션 ID에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 페더레이션 ID가 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 관련 역할에 대한 자세한 내용은 IAM 사용 설명서의 Create a role for a third-party identity provider (federation)를 참조하세요. IAM Identity Center를 사용하는 경우, 권한 집합을 구성합니다. 인증 후 ID가 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 집합을 IAM의 역할과 연관짓습니다. 권한 집합에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 집합을 참조하세요.
- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 IAM 역할을 수임하여 특정 작업에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 교차 계정 액세스 - IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 에서는 (역할을 프록시로 사용하는 대신) 정책을 리소스에 직접 연결할 AWS 서비스 수 있습니다. 교차 계정 액세스에 대한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 교차 계정 리소스 액세스를 참조하세요.
- 교차 서비스 액세스 - 일부는 다른의 기능을 AWS 서비스 사용합니다 AWS 서비스. 예를 들어, 서비스에서 호출하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는 직접적으로 호출하는 위탁자의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.
  - 전달 액세스 세션(FAS) - IAM 사용자 또는 역할을 사용하여에서 작업을 수행하는 경우 AWS보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우, 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는를 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대한 요청과 AWS 서비스함께 사용합니다. FAS 요청은 서비스가 완료하려면 다른 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 수신하는 경우에만 이루어집니다. 이 경우, 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.
  - 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하기 위해 맡는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 Create a role to delegate permissions to an AWS 서비스를 참조하세요.
  - 서비스 연결 역할 - 서비스 연결 역할은에 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은에

나타나 AWS 계정 며 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

- Amazon EC2에서 실행되는 애플리케이션 - IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 AWS CLI 또는 AWS API 요청을 수행하는 애플리케이션의 임시 자격 증명을 관리할 수 있습니다. 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS 역할을 할당하고 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로파일을 생성합니다. 인스턴스 프로필에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증을 얻을 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하세요.

정책을 사용하여 액세스 관리

정책을 AWS 생성하고 자격 증명 또는 리소스에 연결하여 AWS 에서 액세스를 제어합니다. 정책은 자격 증명 또는 리소스와 연결 AWS 될 때 권한을 정의하는의 객체입니다.는 보안 주체(사용자, 루트 사용자 또는 역할 세션)가 요청할 때 이러한 정책을 AWS 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는 지를 결정합니다. 대부분의 정책은에 JSON 문서 AWS 로 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 정보는 IAM 사용 설명서의 JSON 정책 개요를 참조하세요.

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

기본적으로, 사용자 및 역할에는 어떠한 권한도 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 수임할 수 있습니다.

IAM 정책은 작업을 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, `iam:GetRole` 작업을 허용하는 정책이 있다고 가정합니다. 해당 정책이 있는 사용자는 AWS Management Console AWS CLI, 또는 API에서 역할 정보를 가져올 수 있습니다 AWS .

ID 기반 정책

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.

ID 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은의 여러 사용자, 그룹 및 역할에 연결할 수 있는

독립 실행형 정책입니다 AWS 계정. 관리형 정책에는 AWS 관리형 정책 및 고객 관리형 정책이 포함됩니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책 및 인라인 정책 중에서 선택을 참조하세요.

리소스 기반 정책

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 위탁자가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 위탁자를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는이 포함될 수 있습니다 AWS 서비스.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS 관리형 정책을 사용할 수 없습니다.

액세스 제어 목록(ACL)

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 S3 AWS WAF및 Amazon VPC는 ACLs. ACL에 관한 자세한 내용은 Amazon Simple Storage Service 개발자 가이드의 액세스 제어 목록(ACL) 개요를 참조하세요.

기타 정책 타입

AWS 는 덜 일반적인 추가 정책 유형을 지원합니다. 이러한 정책 타입은 더 일반적인 정책 유형에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 – 권한 경계는 ID 기반 정책에 따라 IAM 엔터티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 객체의 ID 기반 정책과 그 권한 경계의 교집합입니다. `Principal` 필드에서 사용자나 역할을 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 정보는 IAM 사용 설명서의 IAM 엔티티에 대한 권한 경계를 참조하세요.
- 서비스 제어 정책(SCPs) - SCPs는의 조직 또는 조직 단위(OU)에 대한 최대 권한을 지정하는 JSON 정책입니다 AWS Organizations. AWS Organizations 는 기업이 소유한 여러 AWS 계정 을 그룹화하고 중앙에서 관리하기 위한 서비스입니다. 조직에서 모든 기능을 활성화할 경우, 서비스 제어 정

책(SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 각각을 포함하여 멤버 계정의 엔터티에 대한 권한을 제한합니다 AWS 계정 루트 사용자. 조직 및 SCP에 대한 자세한 내용은 AWS Organizations 사용 설명서에서 Service control policies을 참조하세요.

- 리소스 제어 정책(RCP) - RCP는 소유한 각 리소스에 연결된 IAM 정책을 업데이트하지 않고 계정의 리소스에 대해 사용 가능한 최대 권한을 설정하는 데 사용할 수 있는 JSON 정책입니다. RCP는 멤버 계정의 리소스에 대한 권한을 제한하며 조직에 속하는지 여부에 AWS 계정 루트 사용자관계없이를 포함한 자격 증명의 유효 권한에 영향을 미칠 수 있습니다. RCP를 AWS 서비스 지원하는 목록을 포함하여 조직 및 RCPs에 대한 자세한 내용은 AWS Organizations 사용 설명서의 리소스 제어 정책(RCPs)을 참조하세요.
- 세션 정책 – 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 ID 기반 정책의 교차와 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 정보는 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 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.

AWS CodeCommit에 대한 인증 및 액세스 제어

에 액세스하려면 자격 증명이 AWS CodeCommit 필요합니다. 이러한 자격 증명에는 Git 자격 증명 또는 Git 연결을 만드는 데 사용하는 SSH 퍼블릭 키를 관리하는 데 사용하는 CodeCommit 리포지토리 및 IAM 사용자와 같은 AWS 리소스에 액세스할 수 있는 권한이 있어야 합니다. 다음 섹션에서는 AWS Identity and Access Management (IAM) 및 CodeCommit을 사용하여 리소스에 대한 액세스를 보호할 수 있는 방법에 대해 세부 정보를 제공합니다.

- 인증
- 액세스 제어

인증

CodeCommit 리포지토리는 Git 기반이며 Git 보안 인증 정보를 비롯하여 기본적인 Git 기능을 지원하므로, CodeCommit 작업 시 IAM 사용자를 사용하는 것이 좋습니다. 다른 자격 증명 유형을 사용하여 CodeCommit에 액세스할 수도 있지만, 다른 자격 증명 유형은 아래 설명과 같이 제한적일 수 있습니다.

자격 증명 유형:

- IAM 사용자 – IAM 사용자는 특정 사용자 지정 권한을 보유한 Amazon Web Services 계정 내의 자격 증명입니다. 예를 들어, IAM 사용자에는 CodeCommit 리포지토리에 액세스하는 데 필요한 Git 보안 인증 정보를 생성 및 관리할 권한이 있을 수 있습니다. CodeCommit 작업용으로 권장되는 사용자 유형입니다. IAM 사용자 이름과 암호를 사용하여 AWS Management Console, AWS 토론 포럼 또는 AWS Support 센터와 같은 보안 AWS 웹 페이지에 로그인할 수 있습니다.

  Git 보안 인증 정보를 생성하거나 SSH 퍼블릭 키를 IAM 사용자와 연결하거나 git-remote-codecommit를 설치 및 구성할 수 있습니다. 이 방법들은 CodeCommit 리포지토리 작업용으로 Git을 설정하는 가장 쉬운 방법입니다. Git 보안 인증 정보를 사용하여 IAM에서 정적 사용자 이름 및 암호를 생성합니다. 그런 다음 Git 사용자 이름 및 암호 인증을 지원하는 Git 및 타사 도구에서 HTTPS 연결용으로 이러한 보안 인증 정보를 사용합니다. SSH 연결을 사용하여 Git 및 CodeCommit에서 SSH 인증에 사용하는 퍼블릭 및 프라이빗 키 파일을 로컬 시스템에서 생성합니다. 퍼블릭 키를 IAM 사용자와 연결하면 로컬 컴퓨터에 프라이빗 키를 저장합니다. git-remote-codecommit는 Git 자체를 확장하며 사용자에 대한 Git 보안 인증 정보를 설정할 필요가 없습니다.

  또한 각 사용자마다 액세스 키를 생성할 수도 있습니다. AWS SDKs 중 하나를 통해 또는 AWS Command Line Interface (AWS CLI)를 사용하여 프로그래밍 방식으로 AWS 서비스에 액세스할 때 액세스 키를 사용합니다. SDK 및 CLI 도구는 액세스 키를 사용하여 요청에 암호화 방식으로 서명합니다. AWS 도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. CodeCommit은 인바운드 API 요청을 인증하기 위한 프로토콜인 서명 버전 4를 지원합니다. 요청 인증에 대한 자세한 내용은 https://docs.aws.amazon.com/general/latest/gr/signature-version-4.html의AWS 일반 참조서명 버전 4 서명 프로세스를 참조하세요.

- Amazon Web Services 계정 루트 사용자 - 가입 AWS할 때 Amazon Web Services 계정과 연결된 이메일 주소와 암호를 제공합니다. 이 두 가지가 루트 보안 인증 정보이며 모든 AWS 리소스에 대한 전체 액세스 권한을 제공합니다. 일부 CodeCommit 기능은 루트 계정 사용자에게 사용할 수 없습니다. 또한 루트 계정에서 Git을 사용하는 유일한 방법은에 포함된 자격 AWS 증명 도우미를 설치 및 구성git-remote-codecommit(권장)하거나 구성하는 것입니다 AWS CLI. Git 보안 인증 정보나 SSH 퍼블릭-프라이빗 키 페어는 루트 계정 사용자와 함께 사용할 수 없습니다. 이러한 이유로 CodeCommit과 상호 작용 시 루트 계정 사용자를 사용하지 않는 것이 좋습니다.

> **Important**
>
> 보안상 관리자 사용자, 즉 AWS 계정에 대한 전체 권한이 있는 IAM 사용자를 만들 때만 루트 보안 인증 정보를 사용하는 것이 좋습니다. 그런 다음 이 관리자 사용자를 사용하여 제한

된 권한이 있는 다른 IAM 사용자 및 역할을 만들 수 있습니다. 자세한 내용은 IAM 사용 설명서에서 IAM 모범 사례 및 관리자 사용자 및 그룹 생성을 참조하세요.

- IAM Identity Center 및 IAM Identity Center의 사용자 - AWS IAM Identity Center 는의 기능을 확장 AWS Identity and Access Management 하여 사용자 관리와 AWS 계정 및 클라우드 애플리케이션에 대한 사용자 액세스를 통합하는 중앙 위치를 제공합니다. AWS를 사용하는 대부분의 사용자에게 모범 사례로 권장되지만, IAM Identity Center는 현재 Git 보안 인증 정보 또는 SSH 키 페어에 대한 메커니즘을 제공하지 않습니다. 이 사용자들은 git-remote-codecommit을 설치하고 구성하여 CodeCommit 리포지토리를 로컬로 복제할 수 있지만, 모든 IDE (통합 개발 환경)가 git-remote-codecommit을 사용한 복제, 푸시 또는 풀링을 지원하는 것은 아닙니다.

  가장 좋은 방법은 관리자 액세스가 필요한 사용자를 포함한 인간 사용자가 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 사용하여 AWS 서비스 에 액세스하도록 요구하는 것입니다.

  페더레이션 자격 증명은 엔터프라이즈 사용자 디렉터리, 웹 자격 증명 공급자, AWS Directory Service, Identity Center 디렉터리 또는 자격 증명 소스를 통해 제공된 자격 증명을 사용하여 AWS 서비스 에 액세스하는 모든 사용자의 사용자입니다. 페더레이션 자격 증명이 액세스할 때 역할을 AWS 계정수임하고 역할은 임시 자격 증명을 제공합니다.

  중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center을(를) 사용하는 것이 좋습니다. IAM Identity Center에서 사용자 및 그룹을 생성하거나 모든 및 애플리케이션에서 사용할 수 있도록 자체 ID 소스의 사용자 AWS 계정 및 그룹 집합에 연결하고 동기화할 수 있습니다. IAM Identity Center에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 IAM Identity Center란 무엇인가요?를 참조하세요.

- IAM 역할 – IAM 사용자와 마찬가지로, IAM 역할은 특정 권한을 부여하기 위해 계정에서 생성할 수 있는 IAM 자격 증명입니다.

  IAM 역할은 특정 권한이 AWS 계정 있는 내 자격 증명입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. 에서 IAM 역할을 일시적으로 AWS Management Console수임하려면 사용자에서 IAM 역할(콘솔)로 전환할 수 있습니다. 또는 AWS API 작업을 호출하거나 사용자 지정 URL을 AWS CLI 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 내용은 IAM 사용 설명서의 역할 수임 방법을 참조하세요.

  임시 보안 인증이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

  - 페더레이션 사용자 액세스 - 페더레이션 ID에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 페더레이션 ID가 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 관련 역할에 대한 자세한 내용은 IAM 사용 설명서의 Create a role for a third-party identity provider (federation)를 참조하세요. IAM Identity Center를 사용하는 경우, 권한 집합을 구

성합니다. 인증 후 ID가 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 집합을 IAM의 역할과 연관짓습니다. 권한 집합에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 집합을 참조하세요.

- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 IAM 역할을 수임하여 특정 작업에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 교차 계정 액세스 - IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 에서는 (역할을 프록시로 사용하는 대신) 정책을 리소스에 직접 연결할 AWS 서비스수 있습니다. 교차 계정 액세스에 대한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 교차 계정 리소스 액세스를 참조하세요.
- 교차 서비스 액세스 - 일부는 다른의 기능을 AWS 서비스 사용합니다 AWS 서비스. 예를 들어, 서비스에서 호출하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는 직접적으로 호출하는 위탁자의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.
  - 전달 액세스 세션(FAS) - IAM 사용자 또는 역할을 사용하여에서 작업을 수행하는 경우 AWS보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우, 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는를 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대한 요청과 AWS 서비스함께 사용합니다. FAS 요청은 서비스가 완료하려면 다른 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 수신하는 경우에만 이루어집니다. 이 경우, 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.
  - 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하기 위해 맡는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 Create a role to delegate permissions to an AWS 서비스를 참조하세요.
  - 서비스 연결 역할 - 서비스 연결 역할은에 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은에 나타나 AWS 계정 며 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.
- Amazon EC2에서 실행되는 애플리케이션 - IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 AWS CLI 또는 AWS API 요청을 수행하는 애플리케이션의 임시 자격 증명을 관리할 수 있습니다. 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS 역할을 할당하고 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로파일을 생성합니다. 인스턴스 프로필에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는

프로그램이 임시 보안 인증을 얻을 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하세요.

> **Note**
>
> Git 보안 인증 정보나 SSH 퍼블릭 키 페어는 페더레이션 사용자와 함께 사용할 수 없습니다. 또한 페더레이션 사용자에는 사용자 기본 설정을 사용할 수 없습니다. 페더레이션 액세스를 사용하여 연결을 설정하는 방법에 대한 자세한 내용은 git-remote-codecommit을 AWS CodeCommit 사용하여에 HTTPS 연결을 위한 설정 단계 단원을 참조하세요.

액세스 제어

요청을 인증하는 데 유효한 보안 인증 정보가 있더라도 권한이 없다면 CodeCommit 리소스를 생성하거나 액세스할 수 없습니다. 예를 들면 리포지토리 보기, 코드 푸시, Git 보안 인증 정보 생성 및 관리 등의 작업을 수행할 수 있는 권한이 있어야 합니다.

다음 단원에서는 CodeCommit에 대한 권한을 관리하는 방법에 대해 설명합니다. 먼저 개요를 읽어 보면 도움이 됩니다.

- CodeCommit 리소스에 대한 액세스 권한 관리 개요
- CodeCommit에 대한 자격 증명 기반 정책(IAM 정책) 사용
- CodeCommit 권한 참조

CodeCommit 리소스에 대한 액세스 권한 관리 개요

모든 AWS 리소스는 Amazon Web Services 계정이 소유합니다. 리소스를 생성하고 액세스할 수 있는 권한은 권한 정책에서 관리합니다. 계정 관리자는 IAM 자격 증명(사용자, 그룹 및 역할)에 권한 정책을 연결할 수 있습니다. 와 같은 일부 서비스는 리소스에 권한 정책 연결 AWS Lambda도 지원합니다.

> **Note**
>
> 계정 관리자 또는 관리자 사용자는 관리자 권한이 있는 사용자입니다. 자세한 설명은 IAM 사용자 가이드의 IAM 모범 사례 섹션을 참조하십시오.

권한을 부여하려면 권한을 부여 받을 사용자, 권한 대상이 되는 리소스, 해당 리소스에 허용되는 특정 작업을 결정합니다.

주제

- CodeCommit 리소스 및 작업
- 리소스 소유권 이해
- 리소스 액세스 관리
- CodeCommit에서 리소스 범위 지정
- 정책 요소 지정: 리소스, 작업, 효과 및 보안 주체
- 정책에서 조건 지정

CodeCommit 리소스 및 작업

CodeCommit에서는 리포지토리가 기본 리소스입니다. 각 리소스에는 관련된 고유 Amazon 리소스 이름(ARN)이 있습니다. 정책에서 Amazon 리소스 이름(ARN)을 사용하여 정책이 적용되는 리소스를 식별합니다. ARN에 대한 자세한 내용은 Amazon Web Services 일반 참조에서 Amazon 리소스 이름(ARN) 및 AWS 서비스 네임스페이스를 참조하세요. CodeCommit은 현재 하위 리소스라고도 하는 다른 리소스 유형을 지원하지 않습니다.

다음 표에서는 CodeCommit 리소스를 지정하는 방법에 대해 설명합니다.

리소스 유형	ARN 형식
리포지토리	arn:aws:codecommit:*region*:*account-id* :*repository-name*
모든 CodeCommit 리포지토리	arn:aws:codecommit:*
지정된 계정에서 소유한 모든 CodeCommit 리포지토리 AWS 리전	arn:aws:codecommit:*region*:*account-id* :*

> **Note**
> 대부분의 AWS 서비스는 ARNs의 콜론(:) 또는 슬래시(/)를 동일한 문자로 처리합니다. 하지만 CodeCommit에서는 리소스 패턴 및 규칙에 정확히 일치하는 항목이 있어야 합니다. 이벤트 패턴을 만들 때 리소스에서 ARN 구문이 일치하도록 정확한 문자를 사용해야 합니다.

예를 들어 명령문에서 다음과 같이 ARN을 사용하여 특정 리포지토리(*MyDemoRepo*)를 나타낼 수 있습니다.

```
"Resource": "arn:aws:codecommit:us-west-2:111111111111:MyDemoRepo"
```

특정 계정에 속하는 모든 리포지토리를 지정하려면 다음과 같이 와일드카드 문자(*)를 사용합니다.

```
"Resource": "arn:aws:codecommit:us-west-2:111111111111:*"
```

모든 리소스를 지정해야 하거나 특정 API 작업이 ARN을 지원하지 않는 경우 다음과 같이 `Resource` 요소에 와일드카드 문자(*)를 사용합니다.

```
"Resource": "*"
```

와일드카드 문자(*)를 사용하여 리포지토리 이름의 일부와 일치하는 모든 리소스를 지정할 수도 있습니다. 예를 들어, 다음 ARN은 이름이 `MyDemo`로 시작하며 `us-east-2` AWS 리전의 Amazon Web Services 계정 111111111111에 등록된 CodeCommit 리포지토리를 지정합니다.

```
arn:aws:codecommit:us-east-2:111111111111:MyDemo*
```

CodeCommit 리소스로 작동하는 사용 가능한 작업의 목록은 CodeCommit 권한 참조 섹션을 참조하세요.

리소스 소유권 이해

Amazon Web Services 계정은 리소스 생성자와 상관없이 계정에서 생성된 리소스를 소유합니다. 인라인 정책을 포함하면 해당 정책의 권한이 잘못된 보안 주체 엔터티에 실수로 추가되는 일을 예방할 수 있습니다. 다음의 예제에서는 이러한 작동 방법을 설명합니다.

- Amazon Web Services 계정에서 IAM 사용자를 생성하고 CodeCommit 리소스를 생성할 수 있는 권한을 해당 사용자에게 부여하면 해당 사용자는 CodeCommit 리소스를 생성할 수 있습니다. 하지만 해당 사용자가 속한 Amazon Web Services 계정이 CodeCommit 리소스를 소유합니다.
- Amazon Web Services 계정의 루트 계정 보안 인증 정보를 사용하여 규칙을 생성하면, Amazon Web Services 계정이 CodeCommit 리소스의 소유자가 됩니다.
- Amazon Web Services 계정에서 CodeCommit 리소스를 생성할 권한이 있는 IAM 역할을 만드는 경우, 해당 역할을 담당할 수 있는 사람은 누구나 CodeCommit 리소스를 생성할 수 있습니다. 이 경우 역할이 속한 Amazon Web Services 계정이 CodeCommit 리소스를 소유합니다.

리소스 액세스 관리

AWS 리소스에 대한 액세스를 관리하려면 권한 정책을 사용합니다. 권한 정책은 누가 무엇에 액세스할 수 있는지를 나타냅니다. 다음 단원에서는 권한 정책을 만드는 데 옵션에 대해 설명합니다.

> **Note**
>
> 이 섹션에서는 CodeCommit의 맥락에서 IAM을 사용하는 방법에 대해 설명하며, IAM 서비스에 대한 자세한 정보는 다루지 않습니다. IAM에 대한 자세한 내용은 IAM 사용 설명서의 IAM이란 무엇입니까? 단원을 참조하세요. IAM 정책 구문과 설명에 대한 자세한 내용은 IAM 사용 설명서의 IAM 정책 참조 섹션을 참조하세요.

자격 증명 기반 정책(IAM 정책)이라고 하는 IAM 자격 증명에 연결된 권한 정책입니다. 리소스에 연결된 권한 정책을 리소스 기반 정책이라고 합니다. 현재 CodeCommit은 자격 증명 기반 정책(IAM 정책)만 지원합니다.

주제

- 자격 증명 기반 정책(IAM 정책)
- 리소스 기반 정책

자격 증명 기반 정책(IAM 정책)

AWS 리소스에 대한 액세스를 관리하려면 IAM 자격 증명에 권한 정책을 연결합니다. CodeCommit에서 자격 증명 기반 정책을 사용하여 리포지토리에 대한 액세스 권한을 관리합니다. 예를 들어, 다음을 수행할 수 있습니다.

- 계정 내 사용자 또는 그룹에 권한 정책 연결 – CodeCommit 콘솔에서 CodeCommit 리소스를 볼 수 있는 사용자 권한을 부여하려면 자격 증명 권한 정책을 사용자 또는 사용자가 속하는 그룹에 연결합니다.
- 권한 정책을 역할에 연결(크로스 계정 권한을 부여하기 위해) – 위임(크로스 계정 액세스를 부여하려는 경우 등). 위임하려면 리소스에 속하는 계정(트러스트 계정)과 해당 리소스에 액세스해야 하는 사용자를 포함하는 계정(신뢰 계정) 간의 신뢰를 구축해야 합니다. 권한 정책은 역할 사용자에게 리소스에 대해 의도한 작업을 수행하는 데 필요한 권한을 부여합니다. 신뢰 정책은 어느 신뢰받는 계정이 사용자에게 해당 역할을 위임할 권한을 부여하도록 허용할지 지정합니다. 자세한 내용은 IAM 용어 및 개념을 참조하세요.

크로스 계정 권한을 부여하려면 자격 증명 기반 권한 정책을 IAM 역할에 연결합니다. 예를 들어 계정 A의 관리자는 다음과 같이 다른 Amazon Web Services 계정(예: 계정 B) 또는 AWS 서비스에 교차 계정 권한을 부여하는 역할을 생성할 수 있습니다.

1. 계정 A 관리자는 IAM 역할을 생성하고 계정 A의 리소스에 대한 권한을 부여하는 역할에 권한 정책을 연결합니다.
2. 계정 A 관리자는 계정 B를 역할을 수임할 보안 주체로 식별하는 역할에 신뢰 정책을 연결합니다.
3. 그런 다음 계정 B 관리자는 계정 B의 사용자에게 역할을 수임할 권한을 위임할 수 있습니다. 이렇게 하면 계정 B의 사용자가 계정 A에서 리소스를 생성하거나 액세스할 수 있습니다. 역할을 수임할 수 있는 AWS 서비스 권한을 부여하려는 경우 신뢰 정책의 보안 주체가 서비스 보안 주체일 AWS 수도 있습니다. 자세한 내용은 IAM 용어 및 개념을 참조하세요.

IAM을 사용하여 권한을 위임하는 방법에 대한 자세한 내용은 IAM 사용자 설명서의 액세스 관리를 참조하세요.

다음 예제 정책에서는 사용자가 *MyDemoRepo*라는 리포지토리에서 브랜치를 생성하도록 허용합니다.

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codecommit:CreateBranch"
      ],
      "Resource" : "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo"
    }
  ]
}
```

계정의 사용자가 액세스할 수 있는 호출 및 리소스를 제한하려면 특정 IAM 정책을 생성하고 나서 IAM 사용자에 이러한 정책을 연결합니다. IAM 역할 생성 방법 및 CodeCommit용 IAM 정책 명령문 예제를 살펴보는 방법에 대한 자세한 내용은 고객 관리형 자격 증명 정책 예 단원을 참조하세요.

리소스 기반 정책

Amazon S3와 같은 일부 서비스는 리소스 기반 권한 정책도 지원합니다. 예를 들어, 리소스 기반 정책을 S3 버킷에 연결하여 해당 버킷에 대한 액세스 권한을 관리할 수 있습니다. CodeCommit은 리소스

기반 정책을 지원하지 않지만, 태그를 사용하여 리소스를 식별한 다음 IAM 정책에서 사용할 수 있습니다. 태그 기반 정책의 예제는 자격 증명 기반 정책(IAM 정책) 단원을 참조하세요.

CodeCommit에서 리소스 범위 지정

CodeCommit에서는 CodeCommit 리소스 및 작업에 설명된 대로 자격 증명 기반 정책 및 권한의 범위를 리소스로 지정할 수 있습니다. 하지만 `ListRepositories` 권한의 범위를 리소스 하나로 지정할 수는 없습니다. 대신에 모든 리소스로 범위를 지정할 수 있습니다(* 와일드카드 사용). 그렇지 않으면 작업이 실패합니다.

다른 모든 CodeCommit 권한은 범위를 리소스로 지정할 수 있습니다.

정책 요소 지정: 리소스, 작업, 효과 및 보안 주체

사용자의 리소스 액세스를 허용 또는 거부하는 정책을 만들거나 사용자가 해당 리소스에서 특정 작업을 수행하도록 허용 또는 거부하는 정책을 만들 수 있습니다. CodeCommit은 사용자가 CodeCommit 콘솔, SDKs, AWS CLI를 통해 또는 해당 API를 직접 호출하여 서비스를 사용하는 방법을 정의하는 일련의 퍼블릭 APIs 작업을 정의합니다. 이러한 API 작업에 대한 권한을 부여하기 위해 CodeCommit에서는 정책에서 지정할 수 있는 작업을 정의합니다.

일부 API 작업은 두 가지 이상의 작업에 대한 권한이 필요합니다. 리소스 및 API 작업에 대한 자세한 설명은 CodeCommit 리소스 및 작업 및 CodeCommit 권한 참조 섹션을 참조하십시오.

다음은 정책의 기본 요소입니다:

- 리소스 – Amazon 리소스 이름(ARN)을 사용하여 정책을 적용할 리소스를 식별합니다. 자세한 내용은 CodeCommit 리소스 및 작업 단원을 참조하십시오.
- 작업 – 작업 키워드를 사용하여 허용 또는 거부할 리소스 작업을 식별합니다. 예를 들면 지정한 `Effect`에 따라 `codecommit:GetBranch` 권한은 사용자에게 `GetBranch` 작업(CodeCommit 리포지토리에서 브랜치에 대한 세부 정보를 가져옴)을 수행하도록 허용하거나 거부합니다.
- 효과 – 사용자가 특정 작업을 요청할 때 발생되는 결과(허용 또는 거부)를 지정합니다. 명시적으로 리소스에 대한 액세스 권한을 부여(허용)하지 않는 경우, 액세스는 묵시적으로 거부됩니다. 다른 정책에서 액세스 권한을 부여하는 경우라도 사용자가 해당 리소스에 액세스할 수 없도록 하기 위해 리소스에 대한 권한을 명시적으로 거부할 수도 있습니다.
- 보안 주체 – 자격 증명 기반 정책(IAM 정책)에서 CodeCommit가 해당 자격 증명이 연결된 사용자를 지원하는 유일한 정책 유형은 암시적 보안 주체입니다.

IAM 정책 구문에 대한 자세한 내용은 IAM 사용 설명서의 IAM 정책 참조 섹션을 참조하세요.

모든 CodeCommit API 작업과 해당 작업이 적용되는 리소스를 보여주는 표는 CodeCommit 권한 참조 섹션을 참조하세요.

정책에서 조건 지정

권한을 부여할 때 IAM에 대한 액세스 정책 언어를 사용하여 정책이 적용될 조건을 지정합니다. 예를 들어, 특정 날짜 이후에만 정책을 적용할 수 있습니다. 정책 언어에서의 조건 지정에 관한 자세한 내용은 IAM 사용 설명서의 조건 및 정책 문법을 참조하세요.

조건을 표시하려면 미리 정의된 조건 키를 사용합니다. CodeCommit에만 해당되는 특정한 조건 키는 없습니다. 그러나 필요에 따라 사용할 수 있는 AWS전체 조건 키가 있습니다. AWS전체 키의 전체 목록은 IAM 사용 설명서의 조건에 사용 가능한 키를 참조하세요.

CodeCommit에 대한 자격 증명 기반 정책(IAM 정책) 사용

다음 자격 증명 기반 정책의 예는 계정 관리자가 권한 정책을 IAM 자격 증명(즉, 사용자, 그룹 및 역할)에 연결하고 이 과정을 통해 CodeCommit 리소스에서 작업을 수행할 권한을 부여할 수 있는 방법을 보여줍니다.

> **Important**
>
> CodeCommit 리소스에 대한 액세스 관리를 위해 제공되는 기본 개념과 옵션 설명에 대한 소개 주제 부분을 먼저 읽어 보세요. 자세한 내용은 CodeCommit 리소스에 대한 액세스 권한 관리 개요 단원을 참조하십시오.

주제

- CodeCommit 콘솔 사용에 필요한 권한
- 콘솔에서 리소스 보기
- AWS CodeCommit에 대한 관리형 정책
- 고객 관리형 정책 예

다음은 자격 증명 기반 권한 정책의 예입니다.

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

```
      "Effect" : "Allow",
      "Action" : [
        "codecommit:BatchGetRepositories"
      ],
      "Resource" : [
        "arn:aws:codecommit:us-east-2:111111111111:MyDestinationRepo",
        "arn:aws:codecommit:us-east-2:111111111111:MyDemo*"
      ]
    }
  ]
}
```

이 정책에는 사용자가 **us-east-2** 리전의 MyDemo라는 이름으로 시작하는 모든 CodeCommit 리포지토리와 MyDestinationRepo라는 이름의 CodeCommit 리포지토리에 대한 정보를 가져오도록 허용하는 명령문이 있습니다.

CodeCommit 콘솔 사용에 필요한 권한

각 CodeCommit API 작업에 필요한 권한을 설정하는 방법과 CodeCommit 작업에 대한 자세한 내용은 CodeCommit 권한 참조 단원을 참조하세요.

사용자에게 CodeCommit 콘솔을 사용하도록 허용하려는 경우 관리자는 CodeCommit 작업에 대한 권한을 부여해야 합니다. 예를 들어 AWSCodeCommitPowerUser 관리형 정책 또는 이와 동등한 정책을 사용자나 그룹에 연결할 수 있습니다.

자격 증명 기반 정책에 의해 사용자에게 부여된 권한 외에도 CodeCommit에는 AWS Key Management Service (AWS KMS) 작업에 대한 권한이 필요합니다. IAM 사용자는 이러한 작업에 대한 명시적인 Allow 권한이 필요하지 않지만 다음 권한을 Deny로 설정하는 어떠한 정책에도 연결되면 안 됩니다.

```
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt",
        "kms:GenerateDataKey",
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:DescribeKey"
```

암호화 및 CodeCommit에 대한 자세한 내용은 AWS KMS 및 암호화 단원을 참조하세요.

콘솔에서 리소스 보기

CodeCommit 콘솔에는 로그인한에서 Amazon Web Services 계정의 리포지토리 목록을 표시할 수 AWS 리전 있는 ListRepositories 권한이 필요합니다. 또한 콘솔에는 리소스의 대/소문자를 구분

하지 않고 빠르게 검색할 수 있는 Go to resource(리소스로 이동) 기능이 포함되어 있습니다. 이 검색은 로그인 AWS 리전 한의 Amazon Web Services 계정에서 수행됩니다. 다음 서비스에 표시되는 리소스는 다음과 같습니다.

- AWS CodeBuild: 빌드 프로젝트
- AWS CodeCommit: 리포지토리
- AWS CodeDeploy: 애플리케이션
- AWS CodePipeline: 파이프라인

모든 서비스의 리소스에서 이 검색을 수행하려면 다음 권한이 있어야 합니다.

- CodeBuild: `ListProjects`
- CodeCommit: `ListRepositories`
- CodeDeploy: `ListApplications`
- CodePipeline: `ListPipelines`

해당 서비스에 대한 권한이 없는 경우 서비스의 리소스에 대한 결과가 반환되지 않습니다. 리소스를 볼 수 있는 권한이 있더라도 해당 리소스 보기에 대한 명시적 `Deny`가 있으면 특정 리소스가 반환되지 않습니다.

AWS CodeCommit에 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. 이 정책은 일반적인 사용 사례를 다루며 사용자의 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 제공합

니다. 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 는에서 생성하고 관리하는 독립 실행형 IAM 정책을 제공하여 많은 일반적인 사용 사례를 처리합니다 AWS. 이러한 AWS 관리형 정책은 일반적인 사용 사례에 필요한 권한을 부여합니다. CodeCommit에 대한 관리형 정책은 IAM, Amazon SNS 및 Amazon CloudWatch Events 같은 다른 서비스에서 작업을 수행할 수 있는 권한도 제공합니다. 해당 정책이 부여된 사용자의 책임에 필요하기 때문입니다. 예를 들어, AWSCodeCommitFullAccess 정책은 관리 수준의 사용자 정책으로, 이 정책을 통해 사용자가 리포지토리에 대한 CloudWatch Events 규칙(이름에 `codecommit`이라는 접두사가 붙은 규칙)과, 리포지토리 관련 이벤트에 대한 알림을 제공할 Amazon SNS 주제(이름에 `codecommit`이라는 접두사가 붙은 주제)를 만들고 관리할 수 있으며, CodeCommit의 관리자 리포지토리를 관리할 수 있습니다.

계정의 사용자에게 연결할 수 있는 다음 AWS 관리형 정책은 CodeCommit에 고유합니다.

주제

- AWS 관리형 정책: AWSCodeCommitFullAccess
- AWS 관리형 정책: AWSCodeCommitPowerUser
- AWS 관리형 정책: AWSCodeCommitReadOnly
- CodeCommit 관리형 정책 및 알림
- AWS CodeCommit 관리형 정책 및 Amazon CodeGuru Reviewer
- AWS 관리형 정책에 대한 CodeCommit 업데이트

AWS 관리형 정책: AWSCodeCommitFullAccess

`AWSCodeCommitFullAccess` 정책을 IAM 보안 인증에 연결할 수 있습니다. 이 정책은 CodeCommit에 대한 모든 액세스 권한을 부여합니다. 이 정책은 리포지토리 삭제 기능을 비롯하여 Amazon Web Services 계정 내 CodeCommit 리포지토리 및 관련 리소스에 대한 모든 제어 권한을 부여할 관리 수준의 사용자에게만 적용됩니다.

AWSCodeCommitFullAccess 정책에는 다음과 같은 정책 명령문이 포함됩니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
```

```
            "codecommit:*"
          ],
          "Resource": "*"
        },
        {
          "Sid": "CloudWatchEventsCodeCommitRulesAccess",
          "Effect": "Allow",
          "Action": [
            "events:DeleteRule",
            "events:DescribeRule",
            "events:DisableRule",
            "events:EnableRule",
            "events:PutRule",
            "events:PutTargets",
            "events:RemoveTargets",
            "events:ListTargetsByRule"
          ],
          "Resource": "arn:aws:events:*:*:rule/codecommit*"
        },
        {
          "Sid": "SNSTopicAndSubscriptionAccess",
          "Effect": "Allow",
          "Action": [
            "sns:CreateTopic",
            "sns:DeleteTopic",
            "sns:Subscribe",
            "sns:Unsubscribe",
            "sns:SetTopicAttributes"
          ],
          "Resource": "arn:aws:sns:*:*:codecommit*"
        },
        {
          "Sid": "SNSTopicAndSubscriptionReadAccess",
          "Effect": "Allow",
          "Action": [
            "sns:ListTopics",
            "sns:ListSubscriptionsByTopic",
            "sns:GetTopicAttributes"
          ],
          "Resource": "*"
        },
        {
          "Sid": "LambdaReadOnlyListAccess",
          "Effect": "Allow",
```

```
            "Action": [
              "lambda:ListFunctions"
            ],
            "Resource": "*"
          },
          {
            "Sid": "IAMReadOnlyListAccess",
            "Effect": "Allow",
            "Action": [
              "iam:ListUsers"
            ],
            "Resource": "*"
          },
          {
            "Sid": "IAMReadOnlyConsoleAccess",
            "Effect": "Allow",
            "Action": [
              "iam:ListAccessKeys",
              "iam:ListSSHPublicKeys",
              "iam:ListServiceSpecificCredentials"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
          },
          {
            "Sid": "IAMUserSSHKeys",
            "Effect": "Allow",
            "Action": [
              "iam:DeleteSSHPublicKey",
              "iam:GetSSHPublicKey",
              "iam:ListSSHPublicKeys",
              "iam:UpdateSSHPublicKey",
              "iam:UploadSSHPublicKey"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
          },
          {
            "Sid": "IAMSelfManageServiceSpecificCredentials",
            "Effect": "Allow",
            "Action": [
              "iam:CreateServiceSpecificCredential",
              "iam:UpdateServiceSpecificCredential",
              "iam:DeleteServiceSpecificCredential",
              "iam:ResetServiceSpecificCredential"
            ],
```

```
            "Resource": "arn:aws:iam::*:user/${aws:username}"
          },
          {
            "Sid": "CodeStarNotificationsReadWriteAccess",
            "Effect": "Allow",
            "Action": [
              "codestar-notifications:CreateNotificationRule",
              "codestar-notifications:DescribeNotificationRule",
              "codestar-notifications:UpdateNotificationRule",
              "codestar-notifications:DeleteNotificationRule",
              "codestar-notifications:Subscribe",
              "codestar-notifications:Unsubscribe"
            ],
            "Resource": "*",
            "Condition": {
              "StringLike": {
                "codestar-notifications:NotificationsForResource": "arn:aws:codecommit:*"
              }
            }
          },
          {
            "Sid": "CodeStarNotificationsListAccess",
            "Effect": "Allow",
            "Action": [
              "codestar-notifications:ListNotificationRules",
              "codestar-notifications:ListTargets",
              "codestar-notifications:ListTagsforResource",
              "codestar-notifications:ListEventTypes"
            ],
            "Resource": "*"
          },
          {
            "Sid": "CodeStarNotificationsSNSTopicCreateAccess",
            "Effect": "Allow",
            "Action": [
              "sns:CreateTopic",
              "sns:SetTopicAttributes"
            ],
            "Resource": "arn:aws:sns:*:*:codestar-notifications*"
          },
          {
            "Sid": "AmazonCodeGuruReviewerFullAccess",
            "Effect": "Allow",
            "Action": [
```

```
            "codeguru-reviewer:AssociateRepository",
            "codeguru-reviewer:DescribeRepositoryAssociation",
            "codeguru-reviewer:ListRepositoryAssociations",
            "codeguru-reviewer:DisassociateRepository",
            "codeguru-reviewer:DescribeCodeReview",
            "codeguru-reviewer:ListCodeReviews"
          ],
          "Resource": "*"
        },
        {
          "Sid": "AmazonCodeGuruReviewerSLRCreation",
          "Action": "iam:CreateServiceLinkedRole",
          "Effect": "Allow",
          "Resource": "arn:aws:iam::*:role/aws-service-role/codeguru-
reviewer.amazonaws.com/AWSServiceRoleForAmazonCodeGuruReviewer",
          "Condition": {
            "StringLike": {
              "iam:AWSServiceName": "codeguru-reviewer.amazonaws.com"
            }
          }
        },
        {
          "Sid": "CloudWatchEventsManagedRules",
          "Effect": "Allow",
          "Action": [
            "events:PutRule",
            "events:PutTargets",
            "events:DeleteRule",
            "events:RemoveTargets"
          ],
          "Resource": "*",
          "Condition": {
            "StringEquals": {
              "events:ManagedBy": "codeguru-reviewer.amazonaws.com"
            }
          }
        },
        {
          "Sid": "CodeStarNotificationsChatbotAccess",
          "Effect": "Allow",
          "Action": [
            "chatbot:DescribeSlackChannelConfigurations",
            "chatbot:ListMicrosoftTeamsChannelConfigurations"
          ],
```

```
            "Resource": "*"
        },
        {
            "Sid": "CodeStarConnectionsReadOnlyAccess",
            "Effect": "Allow",
            "Action": [
                "codestar-connections:ListConnections",
                "codestar-connections:GetConnection"
            ],
            "Resource": "arn:aws:codestar-connections:*:*:connection/*"
        }
    ]
}
```

AWS 관리형 정책: AWSCodeCommitPowerUser

`AWSCodeCommitPowerUser` 정책을 IAM 보안 인증에 연결할 수 있습니다. 이 정책은 사용자가 CodeCommit 및 리포지토리 관련 리소스의 모든 기능에 액세스할 수 있도록 허용합니다. 단, 사용자가 Amazon CloudWatch Events와 같은 다른 AWS 서비스에서 CodeCommit 리포지토리를 삭제하거나 리포지토리 관련 리소스를 생성하거나 삭제하는 것은 허용하지 않습니다. 이 정책은 대부분의 사용자에게 적용하는 것이 좋습니다.

AWSCodeCommitPowerUser 정책에는 다음과 같은 정책 명령문이 포함됩니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecommit:AssociateApprovalRuleTemplateWithRepository",
        "codecommit:BatchAssociateApprovalRuleTemplateWithRepositories",
        "codecommit:BatchDisassociateApprovalRuleTemplateFromRepositories",
        "codecommit:BatchGet*",
        "codecommit:BatchDescribe*",
        "codecommit:Create*",
        "codecommit:DeleteBranch",
        "codecommit:DeleteFile",
        "codecommit:Describe*",
        "codecommit:DisassociateApprovalRuleTemplateFromRepository",
        "codecommit:EvaluatePullRequestApprovalRules",
        "codecommit:Get*",
        "codecommit:List*",
```

```
            "codecommit:Merge*",
            "codecommit:OverridePullRequestApprovalRules",
            "codecommit:Put*",
            "codecommit:Post*",
            "codecommit:TagResource",
            "codecommit:Test*",
            "codecommit:UntagResource",
            "codecommit:Update*",
            "codecommit:GitPull",
            "codecommit:GitPush"
          ],
          "Resource": "*"
        },
        {
          "Sid": "CloudWatchEventsCodeCommitRulesAccess",
          "Effect": "Allow",
          "Action": [
            "events:DeleteRule",
            "events:DescribeRule",
            "events:DisableRule",
            "events:EnableRule",
            "events:PutRule",
            "events:PutTargets",
            "events:RemoveTargets",
            "events:ListTargetsByRule"
          ],
          "Resource": "arn:aws:events:*:*:rule/codecommit*"
        },
        {
          "Sid": "SNSTopicAndSubscriptionAccess",
          "Effect": "Allow",
          "Action": [
            "sns:Subscribe",
            "sns:Unsubscribe"
          ],
          "Resource": "arn:aws:sns:*:*:codecommit*"
        },
        {
          "Sid": "SNSTopicAndSubscriptionReadAccess",
          "Effect": "Allow",
          "Action": [
            "sns:ListTopics",
            "sns:ListSubscriptionsByTopic",
            "sns:GetTopicAttributes"
```

```
            ],
            "Resource": "*"
        },
        {
            "Sid": "LambdaReadOnlyListAccess",
            "Effect": "Allow",
            "Action": [
                "lambda:ListFunctions"
            ],
            "Resource": "*"
        },
        {
            "Sid": "IAMReadOnlyListAccess",
            "Effect": "Allow",
            "Action": [
                "iam:ListUsers"
            ],
            "Resource": "*"
        },
        {
            "Sid": "IAMReadOnlyConsoleAccess",
            "Effect": "Allow",
            "Action": [
                "iam:ListAccessKeys",
                "iam:ListSSHPublicKeys",
                "iam:ListServiceSpecificCredentials"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "IAMUserSSHKeys",
            "Effect": "Allow",
            "Action": [
                "iam:DeleteSSHPublicKey",
                "iam:GetSSHPublicKey",
                "iam:ListSSHPublicKeys",
                "iam:UpdateSSHPublicKey",
                "iam:UploadSSHPublicKey"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "IAMSelfManageServiceSpecificCredentials",
            "Effect": "Allow",
```

```
            "Action": [
              "iam:CreateServiceSpecificCredential",
              "iam:UpdateServiceSpecificCredential",
              "iam:DeleteServiceSpecificCredential",
              "iam:ResetServiceSpecificCredential"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
          },
          {
            "Sid": "CodeStarNotificationsReadWriteAccess",
            "Effect": "Allow",
            "Action": [
              "codestar-notifications:CreateNotificationRule",
              "codestar-notifications:DescribeNotificationRule",
              "codestar-notifications:UpdateNotificationRule",
              "codestar-notifications:Subscribe",
              "codestar-notifications:Unsubscribe"
            ],
            "Resource": "*",
            "Condition": {
              "StringLike": {
                "codestar-notifications:NotificationsForResource": "arn:aws:codecommit:*"
              }
            }
          },
          {
            "Sid": "CodeStarNotificationsListAccess",
            "Effect": "Allow",
            "Action": [
              "codestar-notifications:ListNotificationRules",
              "codestar-notifications:ListTargets",
              "codestar-notifications:ListTagsforResource",
              "codestar-notifications:ListEventTypes"
            ],
            "Resource": "*"
          },
          {
            "Sid": "AmazonCodeGuruReviewerFullAccess",
            "Effect": "Allow",
            "Action": [
              "codeguru-reviewer:AssociateRepository",
              "codeguru-reviewer:DescribeRepositoryAssociation",
              "codeguru-reviewer:ListRepositoryAssociations",
              "codeguru-reviewer:DisassociateRepository",
```

```
            "codeguru-reviewer:DescribeCodeReview",
            "codeguru-reviewer:ListCodeReviews"
          ],
          "Resource": "*"
        },
        {
          "Sid": "AmazonCodeGuruReviewerSLRCreation",
          "Action": "iam:CreateServiceLinkedRole",
          "Effect": "Allow",
          "Resource": "arn:aws:iam::*:role/aws-service-role/codeguru-
reviewer.amazonaws.com/AWSServiceRoleForAmazonCodeGuruReviewer",
          "Condition": {
            "StringLike": {
              "iam:AWSServiceName": "codeguru-reviewer.amazonaws.com"
            }
          }
        },
        {
          "Sid": "CloudWatchEventsManagedRules",
          "Effect": "Allow",
          "Action": [
            "events:PutRule",
            "events:PutTargets",
            "events:DeleteRule",
            "events:RemoveTargets"
          ],
          "Resource": "*",
          "Condition": {
            "StringEquals": {
              "events:ManagedBy": "codeguru-reviewer.amazonaws.com"
            }
          }
        },
        {
          "Sid": "CodeStarNotificationsChatbotAccess",
          "Effect": "Allow",
          "Action": [
            "chatbot:DescribeSlackChannelConfigurations",
            "chatbot:ListMicrosoftTeamsChannelConfigurations"
          ],
          "Resource": "*"
        },
        {
            "Sid": "CodeStarConnectionsReadOnlyAccess",
```

```
            "Effect": "Allow",
            "Action": [
                "codestar-connections:ListConnections",
                "codestar-connections:GetConnection"
            ],
            "Resource": "arn:aws:codestar-connections:*:*:connection/*"
        }
    ]
}
```

AWS 관리형 정책: AWSCodeCommitReadOnly

`AWSCodeCommitReadOnly` 정책을 IAM 보안 인증에 연결할 수 있습니다. 이 정책은 다른 AWS 서비스의 CodeCommit 및 리포지토리 관련 리소스에 대한 읽기 전용 액세스 권한과 자체 CodeCommit 관련 리소스(예: 리포지토리에 액세스할 때 IAM 사용자가 사용할 수 있는 Git 자격 증명 및 SSH 키)를 생성하고 관리하는 기능을 부여합니다. 이 정책은 리포지토리 내용 읽기 기능은 부여하되 내용에 대한 변경을 허용하지 않으려는 사용자에게 적용합니다.

AWSCodeCommitReadOnly 정책에는 다음과 같은 정책 명령문이 포함됩니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "codecommit:BatchGet*",
            "codecommit:BatchDescribe*",
            "codecommit:Describe*",
            "codecommit:EvaluatePullRequestApprovalRules",
            "codecommit:Get*",
            "codecommit:List*",
            "codecommit:GitPull"
         ],
         "Resource":"*"
      },
      {
         "Sid":"CloudWatchEventsCodeCommitRulesReadOnlyAccess",
         "Effect":"Allow",
         "Action":[
            "events:DescribeRule",
            "events:ListTargetsByRule"
         ],
```

```
            "Resource":"arn:aws:events:*:*:rule/codecommit*"
        },
        {
            "Sid":"SNSSubscriptionAccess",
            "Effect":"Allow",
            "Action":[
                "sns:ListTopics",
                "sns:ListSubscriptionsByTopic",
                "sns:GetTopicAttributes"
            ],
            "Resource":"*"
        },
        {
            "Sid":"LambdaReadOnlyListAccess",
            "Effect":"Allow",
            "Action":[
                "lambda:ListFunctions"
            ],
            "Resource":"*"
        },
        {
            "Sid":"IAMReadOnlyListAccess",
            "Effect":"Allow",
            "Action":[
                "iam:ListUsers"
            ],
            "Resource":"*"
        },
        {
            "Sid":"IAMReadOnlyConsoleAccess",
            "Effect":"Allow",
            "Action":[
                "iam:ListAccessKeys",
                "iam:ListSSHPublicKeys",
                "iam:ListServiceSpecificCredentials",
                "iam:GetSSHPublicKey"
            ],
            "Resource":"arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid":"CodeStarNotificationsReadOnlyAccess",
            "Effect":"Allow",
            "Action":[
                "codestar-notifications:DescribeNotificationRule"
```

```
            ],
            "Resource":"*",
            "Condition":{
               "StringLike":{
                  "codestar-
notifications:NotificationsForResource":"arn:aws:codecommit:*"
               }
            }
         },
         {
            "Sid":"CodeStarNotificationsListAccess",
            "Effect":"Allow",
            "Action":[
               "codestar-notifications:ListNotificationRules",
               "codestar-notifications:ListEventTypes",
               "codestar-notifications:ListTargets"
            ],
            "Resource":"*"
         },
         {
            "Sid": "AmazonCodeGuruReviewerReadOnlyAccess",
            "Effect": "Allow",
            "Action": [
               "codeguru-reviewer:DescribeRepositoryAssociation",
               "codeguru-reviewer:ListRepositoryAssociations",
               "codeguru-reviewer:DescribeCodeReview",
               "codeguru-reviewer:ListCodeReviews"
            ],
            "Resource": "*"
         },
         {
           "Sid": "CodeStarConnectionsReadOnlyAccess",
           "Effect": "Allow",
           "Action": [
               "codestar-connections:ListConnections",
               "codestar-connections:GetConnection"
           ],
           "Resource": "arn:aws:codestar-connections:*:*:connection/*"
        }
    ]
}
```

CodeCommit 관리형 정책 및 알림

AWS CodeCommit 는 사용자에게 리포지토리에 대한 중요한 변경 사항을 알릴 수 있는 알림을 지원합니다. CodeCommit에 대한 관리형 정책에는 알림 기능에 대한 정책 명령문이 포함되어 있습니다. 자세한 내용은 알림이란 무엇입니까?를 참조하세요.

전체 액세스 관리형 정책의 알림과 관련된 권한

`AWSCodeCommitFullAccess` 관리형 정책에는 알림에 대한 전체 액세스를 허용하는 다음 설명이 포함되어 있습니다. 이 관리형 정책이 적용된 사용자는 알림을 위한 Amazon SNS 주제를 생성 및 관리하고, 사용자를 주제에 구독 및 구독 취소하고, 알림 규칙의 대상으로 선택할 주제를 나열하고, Slack용으로 구성된 채팅 애플리케이션 클라이언트에서 Amazon Q Developer를 나열할 수도 있습니다.

```
    {
        "Sid": "CodeStarNotificationsReadWriteAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:CreateNotificationRule",
            "codestar-notifications:DescribeNotificationRule",
            "codestar-notifications:UpdateNotificationRule",
            "codestar-notifications:DeleteNotificationRule",
            "codestar-notifications:Subscribe",
            "codestar-notifications:Unsubscribe"
        ],
        "Resource": "*",
        "Condition" : {
            "StringLike" : {"codestar-notifications:NotificationsForResource" :
 "arn:aws:codecommit:*"}
        }
    },
    {
        "Sid": "CodeStarNotificationsListAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:ListNotificationRules",
            "codestar-notifications:ListTargets",
            "codestar-notifications:ListTagsforResource,"
            "codestar-notifications:ListEventTypes"
        ],
        "Resource": "*"
    },
    {
        "Sid": "CodeStarNotificationsSNSTopicCreateAccess",
```

```
        "Effect": "Allow",
        "Action": [
            "sns:CreateTopic",
            "sns:SetTopicAttributes"
        ],
        "Resource": "arn:aws:sns:*:*:codestar-notifications*"
    },
    {
        "Sid": "CodeStarNotificationsChatbotAccess",
        "Effect": "Allow",
        "Action": [
            "chatbot:DescribeSlackChannelConfigurations",
            "chatbot:ListMicrosoftTeamsChannelConfigurations"
          ],
       "Resource": "*"
    }
```

읽기 전용 관리형 정책의 알림과 관련된 권한

`AWSCodeCommitReadOnlyAccess` 관리형 정책에는 알림에 대한 읽기 전용 액세스를 허용하는 다음 설명이 포함되어 있습니다. 이 관리형 정책이 적용된 사용자는 리소스에 대한 알림을 볼 수 있지만 리소스를 생성, 관리 또는 구독할 수는 없습니다.

```
    {
        "Sid": "CodeStarNotificationsPowerUserAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:DescribeNotificationRule"
        ],
        "Resource": "*",
        "Condition" : {
            "StringLike" : {"codestar-notifications:NotificationsForResource" :
 "arn:aws:codecommit:*"}
        }
    },
    {
        "Sid": "CodeStarNotificationsListAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:ListNotificationRules",
            "codestar-notifications:ListEventTypes",
            "codestar-notifications:ListTargets"
        ],
```

```
        "Resource": "*"
    }
```

다른 관리형 정책의 알림과 관련된 권한

`AWSCodeCommitPowerUser` 관리형 정책에는 사용자가 알림을 생성, 편집 및 구독할 수 있도록 허용하는 다음 설명이 포함되어 있습니다. 사용자는 알림 규칙을 삭제하거나 리소스에 대한 태그를 관리할 수는 없습니다.

```
    {
        "Sid": "CodeStarNotificationsReadWriteAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:CreateNotificationRule",
            "codestar-notifications:DescribeNotificationRule",
            "codestar-notifications:UpdateNotificationRule",
            "codestar-notifications:DeleteNotificationRule",
            "codestar-notifications:Subscribe",
            "codestar-notifications:Unsubscribe"
        ],
        "Resource": "*",
        "Condition" : {
            "StringLike" : {"codestar-notifications:NotificationsForResource" :
 "arn:aws:codecommit*"}
        }
    },
    {
        "Sid": "CodeStarNotificationsListAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:ListNotificationRules",
            "codestar-notifications:ListTargets",
            "codestar-notifications:ListTagsforResource",
            "codestar-notifications:ListEventTypes"
        ],
        "Resource": "*"
    },
    {
        "Sid": "SNSTopicListAccess",
        "Effect": "Allow",
        "Action": [
            "sns:ListTopics"
        ],
```

```
        "Resource": "*"
    },
    {
        "Sid": "CodeStarNotificationsChatbotAccess",
        "Effect": "Allow",
        "Action": [
            "chatbot:DescribeSlackChannelConfigurations",
            "chatbot:ListMicrosoftTeamsChannelConfigurations"
          ],
       "Resource": "*"
    }
```

IAM 및 알림에 대한 자세한 내용은 AWS CodeStar 알림의 Identity and Access Management를 참조하세요.

AWS CodeCommit 관리형 정책 및 Amazon CodeGuru Reviewer

CodeCommit은 프로그램 분석 및 기계 학습을 사용하여 Java 또는 Python 코드에서 일반적인 문제를 감지하고 권장 수정 사항을 제공하는 자동화된 코드 검토 서비스인 Amazon CodeGuru Reviewer를 지원합니다. CodeCommit용 관리형 정책에는 CodeGuru Reviewer 기능 관련 정책 명령문이 포함되어 있습니다. 자세한 내용은 Amazon CodeGuru Reviewer는 무엇입니까를 참조하세요.

AWSCodeCommitFullAccess의 CodeGuru Reviewer와 관련된 권한

`AWSCodeCommitFullAccess` 관리형 정책에는 CodeGuru Reviewer가 CodeCommit 리포지토리와 연결 및 연결 해제할 수 있도록 허용하는 다음 명령문이 포함되어 있습니다. 이 관리형 정책이 적용된 사용자는 CodeCommit 리포지토리와 CodeGuru Reviewer 간의 연결 상태를 볼 수도 있고 풀 요청에 대한 리뷰 작업의 상태를 볼 수도 있습니다.

```
    {
      "Sid": "AmazonCodeGuruReviewerFullAccess",
      "Effect": "Allow",
      "Action": [
        "codeguru-reviewer:AssociateRepository",
        "codeguru-reviewer:DescribeRepositoryAssociation",
        "codeguru-reviewer:ListRepositoryAssociations",
        "codeguru-reviewer:DisassociateRepository",
        "codeguru-reviewer:DescribeCodeReview",
        "codeguru-reviewer:ListCodeReviews"
      ],
      "Resource": "*"
    },
```

```
    {
      "Sid": "AmazonCodeGuruReviewerSLRCreation",
      "Action": "iam:CreateServiceLinkedRole",
      "Effect": "Allow",
      "Resource": "arn:aws:iam::*:role/aws-service-role/codeguru-
reviewer.amazonaws.com/AWSServiceRoleForAmazonCodeGuruReviewer",
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": "codeguru-reviewer.amazonaws.com"
        }
      }
    },
    {
      "Sid": "CloudWatchEventsManagedRules",
      "Effect": "Allow",
      "Action": [
        "events:PutRule",
        "events:PutTargets",
        "events:DeleteRule",
        "events:RemoveTargets"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "events:ManagedBy": "codeguru-reviewer.amazonaws.com"
        }
      }
    }
```

AWSCodeCommitPowerUser의 CodeGuru Reviewer와 관련된 권한

`AWSCodeCommitPowerUser` 관리형 정책에는 사용자가 리포지토리를 CodeGuru Reviewer와 연결 및 해제하고 연결 상태를 확인하고 풀 요청에 대한 리뷰 작업의 상태를 볼 수 있는 다음과 같은 명령문이 포함되어 있습니다.

```
    {
      "Sid": "AmazonCodeGuruReviewerFullAccess",
      "Effect": "Allow",
      "Action": [
        "codeguru-reviewer:AssociateRepository",
        "codeguru-reviewer:DescribeRepositoryAssociation",
        "codeguru-reviewer:ListRepositoryAssociations",
        "codeguru-reviewer:DisassociateRepository",
```

```
        "codeguru-reviewer:DescribeCodeReview",
        "codeguru-reviewer:ListCodeReviews"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AmazonCodeGuruReviewerSLRCreation",
      "Action": "iam:CreateServiceLinkedRole",
      "Effect": "Allow",
      "Resource": "arn:aws:iam::*:role/aws-service-role/codeguru-
reviewer.amazonaws.com/AWSServiceRoleForAmazonCodeGuruReviewer",
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": "codeguru-reviewer.amazonaws.com"
        }
      }
    },
    {
      "Sid": "CloudWatchEventsManagedRules",
      "Effect": "Allow",
      "Action": [
        "events:PutRule",
        "events:PutTargets",
        "events:DeleteRule",
        "events:RemoveTargets"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "events:ManagedBy": "codeguru-reviewer.amazonaws.com"
        }
      }
    }
```

AWSCodeCommitReadOnly의 CodeGuru Reviewer와 관련된 권한

`AWSCodeCommitReadOnlyAccess` 관리형 정책에는 사용자가 CodeGuru Reviewer 연결 상태에 대해 읽기 전용으로 액세스하고 풀 요청에 대한 리뷰 작업 상태를 볼 수 있는 다음과 같은 명령문이 포함되어 있습니다. 이 관리형 정책이 적용된 사용자는 리포지토리를 연결하거나 연결 해제할 수 없습니다.

```
     {
      "Sid": "AmazonCodeGuruReviewerReadOnlyAccess",
      "Effect": "Allow",
```

```
        "Action": [
              "codeguru-reviewer:DescribeRepositoryAssociation",
              "codeguru-reviewer:ListRepositoryAssociations",
              "codeguru-reviewer:DescribeCodeReview",
              "codeguru-reviewer:ListCodeReviews"
        ],
        "Resource": "*"
    }
```

Amazon CodeGuru Reviewer 서비스 연결 역할

리포지토리를 CodeGuru Reviewer와 연결하면, CodeGuru Reviewer가 풀 요청에서 Java 또는 Python 코드의 문제를 감지하고 권장 수정 사항을 제공할 수 있도록 서비스 링크 역할이 생성됩니다. 서비스 링크 역할의 이름은 AWSServiceRoleForAmazonCodeGuruReviewer입니다. 자세한 내용은 Amazon CodeGuru Reviewer에 대한 서비스 연결 역할 사용을 참조하세요.

자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책을 참조하세요.

AWS 관리형 정책에 대한 CodeCommit 업데이트

이 서비스가 이러한 변경 사항을 추적하기 시작한 이후부터 CodeCommit의 AWS 관리형 정책 업데이트에 대한 세부 정보를 봅니다. 이 페이지의 변경 사항에 대한 자동 알림을 받아보려면 AWS CodeCommit 사용 설명서 문서 기록에서 RSS 피드를 구독하세요.

변경 사항	설명	날짜
AWS 관리형 정책: AWSCodeCommitFullAccess 및 AWS 관리형 정책: AWSCodeCommitPowerUser – 기존 정책에 대한 업데이트	CodeCommit은 채팅 애플리케이션에서 Amazon Q Developer를 사용하여 추가 알림 유형을 지원하는 권한을 이러한 정책에 추가했습니다. AWSCodeCommitPowerUser 및 AWSCodeCommitFullAccess 정책이 `chatbot:ListMicrosoftTeamsChannelConfigurations` 권한을 추가하도록 변경되었습니다.	2023년 5월 16일

변경 사항	설명	날짜
AWS 관리형 정책: AWSCodeCommitReadOnly - 기존 정책 업데이트	CodeCommit이 정책에서 중복 권한을 제거했습니다. AWSCodeCommitReadOnly가 중복 권한 `"iam:ListAccessKeys"` 를 제거하도록 변경되었습니다.	2021년 8월 18일
CodeCommit이 변경 내용 추적 시작	CodeCommit이 AWS 관리형 정책에 대한 변경 사항 추적을 시작했습니다.	2021년 8월 18일

고객 관리형 정책 예

CodeCommit 작업 및 리소스에 대한 권한을 허용하는 고유의 사용자 지정 IAM 정책을 생성할 수 있습니다. 해당 권한이 필요한 IAM 사용자 또는 그룹에 이러한 사용자 지정 정책을 연결할 수 있습니다. CodeCommit과 다른 AWS 서비스 간의 통합을 위한 사용자 지정 IAM 정책을 직접 생성할 수도 있습니다.

주제

- 고객 관리형 자격 증명 정책 예
- 고객 관리형 통합 정책 예

고객 관리형 자격 증명 정책 예

다음 예제 IAM 정책은 다양한 CodeCommit 작업에 대한 권한을 부여합니다. 이러한 정책을 사용하여 IAM 사용자 및 역할에 대한 CodeCommit 액세스를 제한할 수 있습니다. 이러한 정책은 CodeCommit 콘솔, API, AWS SDKs AWS CLI.

> **Note**
> 모든 예에서는 미국 서부(오레곤) 리전(us-west-2)을 사용하며 가상의 계정 ID를 포함합니다.

예제

- 예제 1: 사용자가 단일에서 CodeCommit 작업을 수행하도록 허용 AWS 리전
- 예제 2: 사용자에게 단일 리포지토리에서 Git을 사용하도록 허용
- 예제 3: 지정 IP 주소 범위에서 연결된 사용자에게 리포지토리에 대한 액세스 허용
- 예시 4: 모든 브랜치에서 작업 허용 또는 거부
- 예제 5: 태그에 따라 리포지토리에 대한 작업 거부 또는 허용

예제 1: 사용자가 단일에서 CodeCommit 작업을 수행하도록 허용 AWS 리전

다음 권한 정책은 와일드카드 문자(`"codecommit:*"`)를 사용하여 사용자에게 다른 AWS 리전이 아닌 us-east-2 리전의 모든 CodeCommit 작업을 수행할 수 있도록 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codecommit:*",
            "Resource": "arn:aws:codecommit:us-east-2:111111111111:*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestedRegion": "us-east-2"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "codecommit:ListRepositories",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestedRegion": "us-east-2"
                }
            }
        }
    ]
}
```

예제 2: 사용자에게 단일 리포지토리에서 Git을 사용하도록 허용

CodeCommit에서 `GitPull` IAM 정책 권한은 git fetch, git clone 등을 포함해 CodeCommit으로부터 데이터가 검색되는 Git 클라이언트 명령에 적용됩니다. 마찬가지로, `GitPush` IAM 정책 권한은 데이

터가 CodeCommit으로 전송되는 모든 Git 클라이언트 명령에 적용됩니다. 예를 들어, `GitPush` IAM 정책 권한이 `Allow`로 설정되면 사용자는 Git 프로토콜을 사용하여 브랜치 삭제를 푸시할 수 있습니다. 이 푸시는 IAM 사용자에 대한 `DeleteBranch` 작업에 적용되는 권한에 영향을 받지 않습니다. `DeleteBranch` 권한은 콘솔, , SDKs 및 API AWS CLI를 사용하여 수행되는 작업에 적용되지만 Git 프로토콜에는 적용되지 않습니다.

다음 예제는 지정된 사용자가 `MyDemoRepo`라는 CodeCommit 리포지토리에 대해 풀하고 푸시할 수 있도록 허용합니다.

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codecommit:GitPull",
        "codecommit:GitPush"
      ],
      "Resource" : "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo"
    }
  ]
}
```

예제 3: 지정 IP 주소 범위에서 연결된 사용자에게 리포지토리에 대한 액세스 허용

IP 주소가 특정 IP 주소 범위 안에 있는 경우 사용자에게 CodeCommit 리포지토리 연결만 허용하는 정책을 생성할 수 있습니다. 여기에는 서로 동등하게 타당한 두 가지 접근 방법이 있습니다. 하나는 사용자의 IP 주소가 특정 블록 안에 들지 않는 경우 CodeCommit 작업을 허용하지 않는 `Deny` 정책을 생성하는 것이며, 또 다른 하나는 사용자의 IP 주소가 특정 블록 안에 드는 경우 CodeCommit 작업을 허용하는 `Allow` 정책을 생성하는 것입니다.

특정 IP 주소 범위에 들지 않는 모든 사용자에 대한 액세스를 거부하는 `Deny` 정책을 생성할 수 있습니다. 예를 들어, AWSCodeCommitPowerUser 관리형 정책과 고객 관리형 정책을 리포지토리에 액세스해야 하는 모든 사용자에게 연결할 수 있습니다. 다음 예제 정책에서는 IP 주소가 지정된 IP 주소 블록 203.0.113.0/16에 들지 않는 사용자에 대해 모든 CodeCommit 권한을 거부합니다.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Deny",
```

```
            "Action": [
               "codecommit:*"
            ],
            "Resource": "*",
            "Condition": {
               "NotIpAddress": {
                  "aws:SourceIp": [
                     "203.0.113.0/16"
                  ]
               }
            }
         }
      ]
}
```

다음 예제 정책은 지정된 사용자에게 MyDemoRepo라는 CodeCommit 리포지토리에 액세스하도록 허용하되, IP 주소가 지정된 주소 블록 203.0.113.0/16 안에 드는 경우에만 AWSCodeCommitPowerUser 관리형 정책의 상응하는 권한을 허용합니다.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "codecommit:BatchGetRepositories",
            "codecommit:CreateBranch",
            "codecommit:CreateRepository",
            "codecommit:Get*",
            "codecommit:GitPull",
            "codecommit:GitPush",
            "codecommit:List*",
            "codecommit:Put*",
            "codecommit:Post*",
            "codecommit:Merge*",
            "codecommit:TagResource",
            "codecommit:Test*",
            "codecommit:UntagResource",
            "codecommit:Update*"
         ],
         "Resource": "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
         "Condition": {
            "IpAddress": {
```

```
                "aws:SourceIp": [
                   "203.0.113.0/16"
                ]
            }
         }
      }
   ]
}
```

예시 4: 모든 브랜치에서 작업 허용 또는 거부

하나 이상의 브랜치의 지정 작업에 사용자 권한을 거부하는 정책을 생성할 수 있습니다. 또는 리포지토리의 다른 브랜치에 없을 작업을 하나 이상의 브랜치에서 허용하는 정책을 만들 수 있습니다. 이런 정책은 해당 관리형(사전에 정의된) 정책과 함께 사용할 수 있습니다. 자세한 내용은 의 브랜치에 대한 푸시 및 병합 제한 AWS CodeCommit 단원을 참조하십시오.

예를 들어 main이라는 브랜치에 대하여 삭제는 물론 변경 권한을 거부하는 `Deny` 정책을 *`MyDemoRepo`*라는 리포지토리에 만들 수 있습니다. 이 정책은 AWSCodeCommitPowerUser 관리형 정책과 함께 사용할 수 있습니다. 이 두 정책이 적용된 사용자는 AWSCodeCommitPowerUser에서 허용하는 대로 브랜치를 생성 및 삭제하고 풀 요청 및 기타 모든 작업을 생성할 수 있지만, main이라는 브랜치에 변경 사항을 푸시하거나 CodeCommit 콘솔의 main 브랜치에 파일을 추가 또는 편집하거나 브랜치 또는 풀 요청을 main 브랜치에 병합할 수는 없습니다. `Deny`가 `GitPush`에 적용되므로 정책에 `Null` 문을 포함해야 사용자 로컬 리포지토리에서 푸시할 때 최초의 `GitPush` 호출이 유효성 분석의 대상이 됩니다.

> **Tip**
>
> Amazon Web Services 계정의 모든 리포지토리에서 main이라는 이름의 모든 브랜치에 적용되는 정책을 생성하려면, `Resource`에 대해 리포지토리 ARN 대신 별표(`*`)를 지정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "codecommit:GitPush",
                "codecommit:DeleteBranch",
```

```
                "codecommit:PutFile",
                "codecommit:Merge*"
            ],
            "Resource": "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
            "Condition": {
                "StringEqualsIfExists": {
                    "codecommit:References": [
                        "refs/heads/main"
                    ]
                },
                "Null": {
                    "codecommit:References": "false"
                }
            }
        }
    ]
}
```

다음 정책 예시에서는 Amazon Web Services 계정의 모든 리포지토리에서 main이라는 브랜치에 대한 사용자의 변경 권한을 허용합니다. 다른 브랜치에 대한 변경은 허용하지 않습니다. 이 정책을 AWSCodeCommitReadOnly 관리형 정책과 함께 사용하여 main 브랜치 내 리포지토리에 푸시를 자동화할 수 있습니다. 결과가 `Allow`이기 때문에, 이 정책 예시는 AWSCodeCommitPowerUser 같은 관리형 정책과 함께 작동하지 않습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codecommit:GitPush",
                "codecommit:Merge*"
            ],
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "codecommit:References": [
                        "refs/heads/main"
                    ]
                }
            }
        }
    ]
```

```
}
```

예제 5: 태그에 따라 리포지토리에 대한 작업 거부 또는 허용

해당 리포지토리와 연결된 AWS 태그를 기반으로 리포지토리에 대한 작업을 허용하거나 거부하는 정책을 생성한 다음 IAM 사용자를 관리하기 위해 구성한 IAM 그룹에 해당 정책을 적용할 수 있습니다. 예를 들어 AWS , 태그 키 상태 및 키 값이 보안 암호인 모든 리포지토리에 대한 모든 CodeCommit 작업을 거부하는 정책을 생성한 다음 일반 개발자(*###*)를 위해 생성한 IAM 그룹에 해당 정책을 적용할 수 있습니다. 그런 다음 태그 지정된 리포지토리에 대해 작업하는 개발자가 일반 *Developers* 그룹의 구성원이 되지 않고 그 대신 제한 정책이 적용되지 않는 다른 IAM 그룹(SecretDevelopers)에 속하게 해야 합니다.

다음 예제에서는 Status 키와 Secret 키 값으로 태그된 리포지토리에 대해 모든 CodeCommit 작업을 거부합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codecommit:Associate*",
        "codecommit:Batch*",
        "codecommit:CancelUploadArchive",
        "codecommit:CreateBranch",
        "codecommit:CreateCommit",
        "codecommit:CreatePullRequest*",
        "codecommit:CreateRepository",
        "codecommit:CreateUnreferencedMergeCommit",
        "codecommit:DeleteBranch",
        "codecommit:DeleteCommentContent",
        "codecommit:DeleteFile",
        "codecommit:DeletePullRequest*",
        "codecommit:DeleteRepository",
        "codecommit:Describe*",
        "codecommit:DisassociateApprovalRuleTemplateFromRepository",
        "codecommit:EvaluatePullRequestApprovalRules",
        "codecommit:GetBlob",
        "codecommit:GetBranch",
        "codecommit:GetComment*",
        "codecommit:GetCommit",
```

```
        "codecommit:GetDifferences*",
        "codecommit:GetFile",
        "codecommit:GetFolder",
        "codecommit:GetMerge*",
        "codecommit:GetObjectIdentifier",
        "codecommit:GetPullRequest*",
        "codecommit:GetReferences",
        "codecommit:GetRepository*",
        "codecommit:GetTree",
        "codecommit:GetUploadArchiveStatus",
        "codecommit:Git*",
        "codecommit:ListAssociatedApprovalRuleTemplatesForRepository",
        "codecommit:ListBranches",
        "codecommit:ListPullRequests",
        "codecommit:ListTagsForResource",
        "codecommit:Merge*",
        "codecommit:OverridePullRequestApprovalRules",
        "codecommit:Post*",
        "codecommit:Put*",
        "codecommit:TagResource",
        "codecommit:TestRepositoryTriggers",
        "codecommit:UntagResource",
        "codecommit:UpdateComment",
        "codecommit:UpdateDefaultBranch",
        "codecommit:UpdatePullRequest*",
        "codecommit:UpdateRepository*",
        "codecommit:UploadArchive"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Status": "Secret"
        }
      }
    }
  ]
}
```

리소스로서 모든 리포지토리보다는 특정 리포지토리를 지정하여 이 전략을 더욱 구체화할 수 있습니다. 또한 특정 태그로 태그 지정되지 않은 모든 리포지토리에 대한 CodeCommit 작업을 허용하는 정책을 생성할 수도 있습니다. 예를 들어, 다음 정책은 CodeCommit 작업에 대해 AWSCodeCommitPowerUser 권한과 동등한 권한을 허용합니다. 단, 지정된 태그가 태그되지 않은 리포지토리에 대한 CodeCommit 작업만을 허용합니다.

> **Note**
>
> 이 정책 예제에는 CodeCommit에 대한 작업만 포함되어 있습니다. 여기에는 AWSCodeCommitPowerUser 관리형 정책에 포함된 다른 AWS 서비스에 대한 작업은 포함되지 않습니다. 자세한 내용은 단원을 참조하십시오AWS 관리형 정책: AWSCodeCommitPowerUser.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecommit:Associate*",
        "codecommit:Batch*",
        "codecommit:CancelUploadArchive",
        "codecommit:CreateBranch",
        "codecommit:CreateCommit",
        "codecommit:CreatePullRequest*",
        "codecommit:CreateRepository",
        "codecommit:CreateUnreferencedMergeCommit",
        "codecommit:DeleteBranch",
        "codecommit:DeleteCommentContent",
        "codecommit:DeleteFile",
        "codecommit:DeletePullRequest*",
        "codecommit:Describe*",
        "codecommit:DisassociateApprovalRuleTemplateFromRepository",
        "codecommit:EvaluatePullRequestApprovalRules",
        "codecommit:GetBlob",
        "codecommit:GetBranch",
        "codecommit:GetComment*",
        "codecommit:GetCommit",
        "codecommit:GetDifferences*",
        "codecommit:GetFile",
        "codecommit:GetFolder",
        "codecommit:GetMerge*",
        "codecommit:GetObjectIdentifier",
        "codecommit:GetPullRequest*",
        "codecommit:GetReferences",
        "codecommit:GetRepository*",
        "codecommit:GetTree",
```

```
        "codecommit:GetUploadArchiveStatus",
        "codecommit:Git*",
        "codecommit:ListAssociatedApprovalRuleTemplatesForRepository",
        "codecommit:ListBranches",
        "codecommit:ListPullRequests",
        "codecommit:ListTagsForResource",
        "codecommit:Merge*",
        "codecommit:OverridePullRequestApprovalRules",
        "codecommit:Post*",
        "codecommit:Put*",
        "codecommit:TagResource",
        "codecommit:TestRepositoryTriggers",
        "codecommit:UntagResource",
        "codecommit:UpdateComment",
        "codecommit:UpdateDefaultBranch",
        "codecommit:UpdatePullRequest*",
        "codecommit:UpdateRepository*",
        "codecommit:UploadArchive"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceTag/Status": "Secret",
          "aws:ResourceTag/Team": "Saanvi"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "codecommit:CreateApprovalRuleTemplate",
        "codecommit:GetApprovalRuleTemplate",
        "codecommit:ListApprovalRuleTemplates",
        "codecommit:ListRepositories",
        "codecommit:ListRepositoriesForApprovalRuleTemplate",
        "codecommit:UpdateApprovalRuleTemplateContent",
        "codecommit:UpdateApprovalRuleTemplateDescription",
        "codecommit:UpdateApprovalRuleTemplateName"
      ],
      "Resource": "*"
    }
  ]
}
```

고객 관리형 통합 정책 예

이 섹션에서는 CodeCommit과 다른 AWS 서비스 간의 통합에 대한 권한을 부여하는 고객 관리형 사용자 정책의 예를 제공합니다. CodeCommit 리포지토리에 대한 크로스 계정 액세스를 허용하는 특정한 정책 예제는 역할을 사용하여 AWS CodeCommit 리포지토리에 대한 교차 계정 액세스 구성 단원을 참조하세요.

> **Note**
>
> 모든 예제 AWS 리전 에서는가 필요할 때 미국 서부(오레곤) 리전(us-west-2)을 사용하며 가상의 계정 IDs 포함합니다.

예제

- 예제 1: Amazon SNS 주제에 대한 크로스 계정 액세스 권한을 허용하는 정책 생성
- 예제 2: Amazon Simple Notification Service(SNS) 주제 정책을 생성하여 Amazon CloudWatch Events가 주제에 대한 CodeCommit 이벤트를 게시하도록 허용합니다.
- 예제 3: CodeCommit 트리거와 AWS Lambda 통합하기 위한 정책 생성

예제 1: Amazon SNS 주제에 대한 크로스 계정 액세스 권한을 허용하는 정책 생성

코드 푸시 또는 기타 이벤트가 Amazon Simple Notification Service(SNS)로부터 알림 전송과 같은 작업을 트리거하도록 CodeCommit 리포지토리를 구성할 수 있습니다. CodeCommit 리포지토리를 생성하는 데 사용한 것과 동일한 계정으로 Amazon SNS 주제를 생성할 경우 추가 IAM 정책 또는 권한을 구성할 필요가 없습니다. 주제를 생성한 다음 리포지토리 트리거를 생성할 수 있습니다. 자세한 내용은 Amazon SNS 주제에 대한 트리거 생성 단원을 참조하십시오.

하지만 다른 Amazon Web Services 계정에서 Amazon SNS 주제를 사용하도록 트리거를 구성하려면, CodeCommit이 해당 주제를 게시하도록 허용하는 정책으로 해당 주제를 먼저 구성해야 합니다. 다른 계정에서 Amazon SNS 콘솔을 열고, 목록에서 주제를 선택하고, 기타 주제 작업에서 주제 정책 편집을 선택합니다. 고급 탭에서 CodeCommit이 해당 주제에 게시하도록 허용하게 주제의 정책을 수정합니다. 예를 들어 해당 정책이 기본 정책일 경우 다음과 같이 정책을 수정하여 ### ### ###를 리포지토리, Amazon SNS 주제 및 계정과 일치하도록 변경합니다.

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
```

```
    {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "sns:Subscribe",
        "sns:ListSubscriptionsByTopic",
        "sns:DeleteTopic",
        "sns:GetTopicAttributes",
        "sns:Publish",
        "sns:RemovePermission",
        "sns:AddPermission",        "sns:SetTopicAttributes"
      ],
      "Resource": "arn:aws:sns:us-east-2:111111111111:NotMySNSTopic",
      "Condition": {
        "StringEquals": {
          "AWS:SourceOwner": "111111111111"
        }
      }
     },
     {
      "Sid": "CodeCommit-Policy_ID",
      "Effect": "Allow",
      "Principal": {
        "Service": "codecommit.amazonaws.com"
      },
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:111111111111:NotMySNSTopic",
      "Condition": {
        "StringEquals": {
          "AWS:SourceArn": "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
          "AWS:SourceAccount": "111111111111"
        }
      }
    }
  ]
}
```

예제 2: Amazon Simple Notification Service(SNS) 주제 정책을 생성하여 Amazon CloudWatch Events가 주제에 대한 CodeCommit 이벤트를 게시하도록 허용합니다.

CodeCommit 이벤트를 비롯한 이벤트가 발생하면, Amazon SNS 주제에 게시하도록 CloudWatch Events를 구성할 수 있습니다. 그렇게 하려면 해당 주제에 대한 정책을 생성하거나 다음과 유사한 주제에 대한 기존 정책을 수정하여 이벤트를 Amazon SNS 주제에 게시할 수 있는 권한이 CloudWatch Events에 있어야 합니다.

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic",
      "Condition": {
        "StringEquals": {
          "AWS:SourceOwner": "123456789012"
        }
      }
    },
    {
      "Sid": "Allow_Publish_Events",
      "Effect": "Allow",
      "Principal": {
        "Service": "events.amazonaws.com"
      },
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
    }
  ]
}
```

CodeCommit 및 CloudWatch 이벤트에 대한 자세한 내용은 지원되는 서비스의 CloudWatch 이벤트의 이벤트 예제를 참조하세요. IAM 및 정책 언어에 대한 자세한 내용은 IAM JSON 정책 언어의 문법을 참조하세요.

예제 3: CodeCommit 트리거와 AWS Lambda 통합하기 위한 정책 생성

코드 푸시나 기타 이벤트가 AWS Lambda의 함수 호출과 같은 작업을 트리거하도록 CodeCommit 리포지토리를 구성할 수 있습니다. 자세한 내용은 Lambda 함수를 위한 트리거 생성 단원을 참조하십시오. 이 정보는 트리거와 관련되며 CloudWatch Events와는 관련되지 않습니다.

트리거가 Amazon SNS 주제를 사용하여 Lambda 함수를 간접 호출하는 대신 직접 Lambda 함수를 실행하기를 원한다면, 그리고 Lambda 콘솔에서 트리거를 구성하지 않으려면, 함수의 리소스 기반 정책에 다음과 비슷한 명령문을 포함시켜야 합니다.

```
{
  "Statement":{
     "StatementId":"Id-1",
     "Action":"lambda:InvokeFunction",
     "Principal":"codecommit.amazonaws.com",
     "SourceArn":"arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
     "SourceAccount":"111111111111"
  }
}
```

Lambda 함수를 호출하는 CodeCommit 트리거를 수동으로 구성할 경우, Lambda AddPermission 명령을 사용하여 CodeCommit이 함수를 간접 호출할 수 있는 권한도 부여해야 합니다. 해당 예제는 CodeCommit이 Lambda 함수를 실행할 수 있게 하려면의 기존 Lambda 함수를 위한 트리거 생성 단원을 참조하세요.

Lambda 함수의 리소스 정책에 대해 자세히 알아보려면 AWS Lambda 개발자 안내서에서 AddPermission 및 풀/푸시 이벤트 모델을 참조하세요.

CodeCommit 권한 참조

다음 표에는 각 CodeCommit API 작업, 권한을 부여할 수 있는 해당 작업, 권한을 부여하는 데 사용할 수 있는 리소스 ARN 형식이 나열되어 있습니다. CodeCommit API는 해당 API에서 허용하는 작업의 범위에 근거하여 표로 그룹화됩니다. IAM 자격 증명에 연결할 수 있는 액세스 제어 및 쓰기 권한 정책(자격 증명 기반 정책)을 설정할 때는 다음 표를 참조하세요.

권한 정책 생성 시 정책의 `Action` 필드에 작업을 지정합니다. 와일드카드(*) 사용 여부와 상관없이 정책의 `Resource` 필드에 리소스 값으로 ARN을 지정합니다.

CodeCommit 정책에서 조건을 표현하려면 AWS차원 조건 키를 사용합니다. AWS전체 키의 전체 목록은 IAM 사용 설명서의 사용 가능한 키를 참조하세요. IAM 정책의 CodeCommit에 대한 작업, 리소스,

조건 키 등에 대해 자세히 알아보려면 AWS CodeCommit에 대한 작업, 리소스 및 조건 키를 참조하세요.

> **Note**
>
> 작업을 지정하려면 `codecommit:` 접두사 다음에 API 작업 이름을 사용합니다(예: `codecommit:GetRepository` 또는 `codecommit:CreateRepository`).

와일드카드 사용

여러 작업이나 리소스를 지정하려면 ARN에서 와일드카드(*)를 사용합니다. 예를 들어, `codecommit:*`은 모든 CodeCommit 작업을 지정하고, `codecommit:Get*`는 `Get`이라는 단어로 시작하는 모든 CodeCommit 작업을 지정합니다. 다음 정책은 이름이 `MyDemo`로 시작되는 모든 리포지토리에 대한 액세스 권한을 부여합니다.

```
arn:aws:codecommit:us-west-2:111111111111:MyDemo*
```

다음 표에 나열된 *`repository-name`* 리소스에만 와일드카드를 사용할 수 있습니다. *`region`* 또는 *`account-id`* 리소스에는 와일드카드를 사용할 수 없습니다. 와일드카드에 대한 자세한 내용은 IAM 사용 설명서에서 IAM 식별자를 참조하세요.

주제

- Git 클라이언트 명령을 위한 필수 권한
- 브랜치 작업 권한
- 병합 작업 권한
- 풀 요청에 대한 작업 권한
- 승인 규칙 템플릿에 대한 작업 권한
- 개별 파일에 대한 작업 권한
- 주석 작업 권한
- 커밋된 코드 대한 작업 권한
- 리포지토리에 대한 작업 권한
- 태그에 대한 작업 권한
- 트리거에 대한 작업 권한
- CodePipeline 통합에 대한 작업 권한

Git 클라이언트 명령을 위한 필수 권한

CodeCommit에서 `GitPull` IAM 정책 권한은 git fetch, git clone 등을 포함해 CodeCommit으로부터 데이터가 검색되는 Git 클라이언트 명령에 적용됩니다. 마찬가지로, `GitPush` IAM 정책 권한은 데이터가 CodeCommit으로 전송되는 모든 Git 클라이언트 명령에 적용됩니다. 예를 들어, `GitPush` IAM 정책 권한이 `Allow`로 설정되면 사용자는 Git 프로토콜을 사용하여 브랜치 삭제를 푸시할 수 있습니다. 이 푸시는 IAM 사용자에 대한 `DeleteBranch` 작업에 적용되는 권한에 영향을 받지 않습니다. `DeleteBranch` 권한은 콘솔, , SDKs 및 API AWS CLI를 사용하여 수행되는 작업에 적용되지만 Git 프로토콜에는 적용되지 않습니다.

`GitPull`과 `GitPush`는 IAM 정책 권한입니다. API 작업이 아닙니다.

Git 클라이언트 명령을 위한 CodeCommit 작업 필수 권한

GitPull

작업: `codecommit:GitPull`

CodeCommit 리포지토리에서 로컬 리포지토리로 정보를 풀하는 데 필요합니다. IAM 정책 권한일 뿐, API 작업이 아닙니다.

리소스: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

GitPush

작업: `codecommit:Git Push`

로컬 리포지토리에서 CodeCommit 리포지토리로 정보를 푸시하는 데 필요합니다. IAM 정책 권한일 뿐, API 작업이 아닙니다.

리소스: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

브랜치 작업 권한

다음 권한은 CodeCommit 리포지토리에서 브랜치에 대한 작업을 허용하거나 거부합니다. 이러한 권한은 CodeCommit 콘솔 및 CodeCommit API를 사용하여 수행되는 작업과 AWS CLI를 사용하여 수행되는 명령에만 적용됩니다. 이러한 권한은 Git 프로토콜을 사용하여 수행할 수 있는 비슷한 작업에는 적용되지 않습니다. 예를 들면 git show-branch -r 명령은 Git 프로토콜을 사용하여 리포지토리와 관련 커밋에 대한 원격 브랜치 목록을 표시합니다. CodeCommit `ListBranches` 작업 관련 권한에 영향을 받지 않습니다.

브랜치에 대한 정책에 대해 자세히 알아보려면 의 브랜치에 대한 푸시 및 병합 제한 AWS CodeCommit 및 고객 관리형 정책 예 단원을 참조하세요.

브랜치에 대한 CodeCommit API 작업 및 작업 필수 권한

CreateBranch

작업: `codecommit:CreateBranch`

CodeCommit 리포지토리에서 브랜치를 생성하는 데 필요합니다.

리소스: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

DeleteBranch

작업: `codecommit:DeleteBranch`

CodeCommit 리포지토리에서 브랜치를 삭제하는 데 필요합니다.

리소스: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

GetBranch

작업: `codecommit:GetBranch`

CodeCommit 리포지토리의 브랜치에 대한 세부 정보를 가져오는 데 필요합니다.

리소스: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

ListBranches

작업: `codecommit:ListBranches`

CodeCommit 리포지토리에서 브랜치 목록을 가져오는 데 필요합니다.

리소스: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

MergeBranchesByFastForward

작업: `codecommit:MergeBranchesByFastForward`

CodeCommit 리포지토리의 fast-forward merge를 사용하여 2개 브랜치를 병합하는 데 필요합니다.

리소스: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

MergeBranchesBySquash

작업: `codecommit:ListBranches`

CodeCommit 리포지토리의 squash 병합 전략을 사용하여 2개 브랜치를 병합하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

MergeBranchesByThreeWay

작업: `codecommit:ListBranches`

CodeCommit 리포지토리의 3방향 병합 전략을 사용하여 2개 브랜치를 병합하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

UpdateDefaultBranch

작업: `codecommit:UpdateDefaultBranch`

CodeCommit 리포지토리에서 기본 브랜치를 변경하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

병합 작업 권한

다음 권한은 CodeCommit 리포지토리에서 병합에 대한 작업을 허용하거나 거부합니다. 이러한 권한은 CodeCommit 콘솔 및 CodeCommit API를 사용하여 수행되는 작업과 AWS CLI를 사용하여 수행되는 명령에 적용됩니다. 이러한 권한은 Git 프로토콜을 사용하여 수행할 수 있는 비슷한 작업에는 적용되지 않습니다. 브랜치 관련 권한은 브랜치 작업 권한 단원을 참조하세요. 풀 요청 관련 권한은 풀 요청에 대한 작업 권한 단원을 참조하세요.

병합 명령 작업에 대한 CodeCommit API 작업 및 필수 권한

BatchDescribeMergeConflicts

작업: `codecommit:BatchDescribeMergeConflicts`

CodeCommit 리포지토리에서 커밋 간 병합의 충돌에 대한 정보를 반환하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

CreateUnreferencedMergeCommit

작업: `codecommit:CreateUnreferencedMergeCommit`

상호 비교 및 잠재적 충돌을 식별하기 위한 목적으로 CodeCommit 리포지토리의 2개 브랜치 또는 커밋 간의 참조되지 않은 커밋을 생성하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

DescribeMergeConflicts

작업: `codecommit:DescribeMergeConflicts`

CodeCommit 리포지토리의 잠재적 병합에서 파일의 기반, 소스, 대상 버전 간 병합 충돌에 대한 정보를 반환하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

GetMergeCommit

작업: `codecommit:GetMergeCommit`

CodeCommit 리포지토리의 소스와 대상 커밋 간의 병합에 대한 정보를 반환하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

GetMergeOptions

작업: `codecommit:GetMergeOptions`

CodeCommit 리포지토리의 2개 브랜치 또는 커밋 지정자 간의 사용 가능한 병합 옵션에 대한 정보를 반환하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

풀 요청에 대한 작업 권한

다음 권한은 CodeCommit 리포지토리에서 풀 요청에 대한 작업을 허용하거나 거부합니다. 이러한 권한은 CodeCommit 콘솔 및 CodeCommit API를 사용하여 수행되는 작업과 AWS CLI를 사용하여 수행되는 명령에 적용됩니다. 이러한 권한은 Git 프로토콜을 사용하여 수행할 수 있는 비슷한 작업에는 적용되지 않습니다. 주석 관련 권한은 주석 작업 권한 단원을 참조하세요.

풀 요청에 대한 CodeCommit API 작업 및 작업 필수 권한

BatchGetPullRequests

작업: `codecommit:BatchGetPullRequests`

CodeCommit 리포지토리에서 하나 이상의 풀 요청에 관한 정보를 반환하는 데 필요합니다. 이것은 IAM 정책 권한일 뿐, 직접 호출할 수 있는 API 작업은 아닙니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

CreatePullRequest

작업: `codecommit:CreatePullRequest`

CodeCommit 리포지토리에서 풀 요청을 생성하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

CreatePullRequestApprovalRule

작업: `codecommit:CreatePullRequestApprovalRule`

CodeCommit 리포지토리에서 풀 요청에 대한 승인 규칙을 생성하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

DeletePullRequestApprovalRule

작업: `codecommit:DeletePullRequestApprovalRule`

CodeCommit 리포지토리에서 풀 요청에 대한 승인 규칙을 삭제하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

DescribePullRequestEvents

작업: `codecommit:DescribePullRequestEvents`

CodeCommit 리포지토리에서 하나 이상의 풀 요청 이벤트에 관한 정보를 반환하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

EvaluatePullRequestApprovalRules

작업: `codecommit:EvaluatePullRequestApprovalRules`

풀 요청이 CodeCommit 리포지토리의 연결된 승인 규칙에 지정된 모든 조건을 충족하는지 여부를 평가하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

GetCommentsForPullRequest

작업: `codecommit:GetCommentsForPullRequest`

풀 요청에 대한 주석을 반환하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

GetCommitsFromMergeBase

작업: `codecommit:GetCommitsFromMergeBase`

잠재적 병합 컨텍스트에서 커밋 간의 차이에 대한 정보를 반환하기 위해 필요합니다. 이것은 IAM 정책 권한일 뿐, 직접 호출할 수 있는 API 작업은 아닙니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

GetMergeConflicts

작업: `codecommit:GetMergeConflicts`

풀 요청의 소스 브랜치와 대상 브랜치 간의 병합 충돌에 대한 정보를 반환하기 위해 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

GetPullRequest

작업: `codecommit:GetPullRequest`

CodeCommit 리포지토리에서 풀 요청에 관한 정보를 반환하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

GetPullRequestApprovalStates

작업: `codecommit:GetPullRequestApprovalStates`

지정된 풀 요청의 승인 상태에 대한 정보를 반환하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

GetPullRequestOverrideState

작업: `codecommit:GetPullRequestOverrideState`

풀 요청에 대한 승인 규칙을 무시(재정의)했는지 여부에 대한 정보를 반환하고, 재정의된 경우 풀 요청에 대한 규칙 및 요구 사항을 재정의한 사용자 또는 자격 증명의 Amazon 리소스 이름(ARN)을 반환하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

ListPullRequests

작업: `codecommit:ListPullRequests`

리포지토리의 풀 요청을 나열하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

MergePullRequestByFastForward

작업: `codecommit:MergePullRequestByFastForward`

풀 요청을 닫고 fast-forward merge 전략을 사용하여 소스 브랜치를 풀 요청의 대상 브랜치에 병합하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

MergePullRequestBySquash

작업: `codecommit:MergePullRequestBySquash`

풀 요청을 닫고 squash 병합 전략을 사용하여 소스 브랜치를 풀 요청의 대상 브랜치에 병합하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

MergePullRequestByThreeWay

작업: `codecommit:MergePullRequestByThreeWay`

풀 요청을 닫고 3방향 병합 전략을 사용하여 소스 브랜치를 풀 요청의 대상 브랜치에 병합하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

OverridePullRequestApprovalRules

작업: `codecommit:OverridePullRequestApprovalRules`

CodeCommit 리포지토리에서 풀 요청에 대한 모든 승인 규칙 요구 사항을 구분하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

PostCommentForPullRequest

작업: `codecommit:PostCommentForPullRequest`

CodeCommit 리포지토리에서 풀 요청에 대한 주석을 게시하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

UpdatePullRequestApprovalRuleContent

작업: `codecommit:UpdatePullRequestApprovalRuleContent`

CodeCommit 리포지토리에서 풀 요청에 대한 승인 규칙의 구조를 변경하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

UpdatePullRequestApprovalState

작업: `codecommit:UpdatePullRequestApprovalState`

CodeCommit 리포지토리에서 풀 요청의 승인 상태를 업데이트하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

UpdatePullRequestDescription

작업: `codecommit:UpdatePullRequestDescription`

CodeCommit 리포지토리에서 풀 요청의 설명을 변경하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

UpdatePullRequestStatus

작업: `codecommit:UpdatePullRequestStatus`

CodeCommit 리포지토리에서 풀 요청의 상태를 변경하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

UpdatePullRequestTitle

작업: `codecommit:UpdatePullRequestTitle`

CodeCommit 리포지토리에서 풀 요청의 제목을 변경하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

승인 규칙 템플릿에 대한 작업 권한

다음 권한은 CodeCommit 리포지토리에서 승인 규칙 템플릿에 대한 작업을 허용하거나 거부합니다. 이러한 권한은 CodeCommit 콘솔 및 CodeCommit API를 사용하여 수행되는 작업과 AWS CLI를 사용하여 수행되는 명령에만 적용됩니다. 이러한 권한은 Git 프로토콜을 사용하여 수행할 수 있는 비슷한 작업에는 적용되지 않습니다. 풀 요청 관련 권한은 풀 요청에 대한 작업 권한 단원을 참조하세요.

승인 규칙 템플릿에 대한 CodeCommit API 작업 및 작업 필수 권한

AssociateApprovalRuleTemplateWithRepository

작업: `codecommit:AssociateApprovalRuleTemplateWithRepository`

템플릿을 Amazon Web Services 계정의 지정된 리포지토리와 연결하는 데 필요합니다. 연결된 템플릿은 지정된 리포지토리에서 생성되는 모든 풀 요청에 대해 템플릿 조건과 일치하는 승인 규칙을 자동으로 생성합니다.

리소스: *

BatchAssociateApprovalRuleTemplateWithRepositories

작업: `codecommit:BatchAssociateApprovalRuleTemplateWithRepositories`

템플릿을 Amazon Web Services 계정에서 지정된 하나 이상의 리포지토리와 연결하는 데 필요합니다.

리소스: *

BatchDisassociateApprovalRuleTemplateFromRepositories

작업: `codecommit:BatchDisassociateApprovalRuleTemplateFromRepositories`

Amazon Web Services 계정에서 지정된 하나 이상의 리포지토리에서 템플릿을 연결 해제하는 데 필요합니다.

리소스: *

CreateApprovalRuleTemplate

작업: `codecommit:CreateApprovalRuleTemplate`

Amazon Web Services 계정에 있는 하나 이상의 리포지토리와 연결할 수 있는 승인 규칙에 대한 템플릿을 만드는 데 필요합니다.

리소스: *

DeleteApprovalRuleTemplate

작업: `codecommit:DeleteApprovalRuleTemplate`

AWS 계정에서 승인 규칙 템플릿을 삭제하는 데 필요합니다.

리소스: *

DisassociateApprovalRuleTemplateFromRepository

작업: `codecommit:DisassociateApprovalRuleTemplateFromRepository`

Amazon Web Service 계정의 리포지토리에서 지정된 템플릿을 연결 해제하는 데 필요합니다. 템플릿으로 이미 생성된 풀 요청에 대한 승인 규칙은 제거되지 않습니다.

리소스: *

GetApprovalRuleTemplate

작업: `codecommit:GetApprovalRuleTemplate`

Amazon Web Services 계정의 승인 규칙 템플릿에 대한 정보를 반환하는 데 필요합니다.

리소스: *

ListApprovalRuleTemplates

작업: `codecommit:ListApprovalRuleTemplates`

Amazon Web Services 계정의 승인 규칙 템플릿을 나열하는 데 필요합니다.

리소스: *

ListAssociatedApprovalRuleTemplatesForRepository

작업: `codecommit:ListAssociatedApprovalRuleTemplatesForRepository`

Amazon Web Services 계정의 지정된 리포지토리와 연결된 모든 승인 규칙 템플릿을 나열하는 데 필요합니다.

리소스: *

ListRepositoriesForApprovalRuleTemplate

작업: `codecommit:ListRepositoriesForApprovalRuleTemplate`

Amazon Web Services 계정의 지정된 승인 규칙 템플릿과 연결된 모든 리포지토리를 나열하는 데 필요합니다.

리소스: *

UpdateApprovalRuleTemplateContent

작업: `codecommit:UpdateApprovalRuleTemplateContent`

Amazon Web Services 계정에서 승인 규칙 템플릿의 내용을 업데이트하는 데 필요합니다.

리소스: *

UpdateApprovalRuleTemplateDescription

작업: `codecommit:UpdateApprovalRuleTemplateDescription`

Amazon Web Services 계정에서 승인 규칙 템플릿에 대한 설명을 업데이트하는 데 필요합니다.

리소스: *

UpdateApprovalRuleTemplateName

작업: `codecommit:UpdateApprovalRuleTemplateName`

AWS 계정에서 승인 규칙 템플릿의 이름을 업데이트하는 데 필요합니다.

리소스: *

개별 파일에 대한 작업 권한

다음 권한은 CodeCommit 리포지토리에서 개별 파일에 대한 작업을 허용하거나 거부합니다. 이러한 권한은 CodeCommit 콘솔 및 CodeCommit API를 사용하여 수행되는 작업과 AWS CLI를 사용하여 수행되는 명령에만 적용됩니다. 이러한 권한은 Git 프로토콜을 사용하여 수행할 수 있는 비슷한 작업에는 적용되지 않습니다. 예를 들어 `git push` 명령은 Git 프로토콜을 사용하여 CodeCommit 리포지토리에 새로 생성된 파일과 변경된 파일을 푸시합니다. CodeCommit `PutFile` 작업에 대한 어떤 권한에도 영향을 받지 않습니다.

개별 파일에 대한 CodeCommit API 작업 및 작업 필수 권한

DeleteFile

작업: `codecommit:DeleteFile`

CodeCommit 콘솔에서 CodeCommit 리포지토리의 지정된 브랜치에서 지정된 파일을 삭제할 때 필요합니다.

리소스: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

GetBlob

작업: `codecommit:GetBlob`

CodeCommit 콘솔에서 CodeCommit 리포지토리 내 개별 파일의 인코딩된 콘텐츠를 보는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

GetFile

작업: `codecommit:GetFile`

CodeCommit 콘솔에서 CodeCommit 리포지토리 내 지정된 파일의 인코딩된 콘텐츠와 메타데이터를 보는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

GetFolder

작업: `codecommit:GetFolder`

CodeCommit 콘솔에서 CodeCommit 리포지토리의 지정된 폴더의 콘텐츠를 보는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

PutFile

작업: `codecommit:PutFile`

CodeCommit 콘솔, CodeCommit API 또는 AWS CLI에서 CodeCommit 리포지토리에 새로 생성되거나 수정된 파일을 추가하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

주석 작업 권한

다음 권한은 CodeCommit 리포지토리에서 주석에 대한 작업을 허용하거나 거부합니다. 이러한 권한은 CodeCommit 콘솔 및 CodeCommit API를 사용하여 수행되는 작업과 AWS CLI를 사용하여 수행되는 명령에 적용됩니다. 풀 요청의 주석 관련 권한은 풀 요청에 대한 작업 권한 단원을 참조하세요.

리포지토리에 대한 CodeCommit API 작업 및 작업 필수 권한

DeleteCommentContent

작업: `codecommit:DeleteCommentContent`

리포지토리에서 변경 사항, 파일 또는 커밋에 대한 주석 내용을 삭제하는 데 필요합니다. 주석은 삭제할 수 없지만 주석의 내용은 이 권한을 가진 사용자가 제거할 수 있습니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

GetComment

작업: `codecommit:GetComment`

CodeCommit 리포지토리의 변경 사항, 파일 또는 커밋에 대해 작성된 주석에 관한 정보를 반환하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

GetCommentReactions

작업: `codecommit:GetCommentReactions`

CodeCommit 리포지토리의 변경 사항, 파일 또는 커밋에 대해 작성된 주석의 이모티콘 반응에 대한 정보를 반환하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

GetCommentsForComparedCommit

작업: `codecommit:GetCommentsForComparedCommit`

CodeCommit 리포지토리의 두 커밋 간의 비교에 대해 작성된 주석에 관한 정보를 반환하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

PostCommentForComparedCommit

작업: `codecommit:PostCommentForComparedCommit`

CodeCommit 리포지토리의 두 커밋 간 비교에 대해 주석을 작성하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

PostCommentReply

작업: `codecommit:PostCommentReply`

CodeCommit 리포지토리에서 커밋 간 비교 또는 풀 요청에 대한 주석에 댓글을 작성하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

PutCommentReaction

작업: `codecommit:PutCommentReaction`

CodeCommit 리포지토리의 커밋 또는 풀 요청에 대한 주석에 이모티콘으로 댓글을 작성하는 데 필요합니다.

리소스: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

UpdateComment

작업: `codecommit:UpdateComment`

커밋 간 비교 또는 풀 요청에 대한 주석에 댓글을 편집하는 데 필요합니다. 주석은 주석 작성자만 편집할 수 있습니다.

리소스: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

커밋된 코드 대한 작업 권한

다음 권한은 CodeCommit 리포지토리로 커밋된 코드에 대한 작업을 허용하거나 거부합니다. 이러한 권한은 CodeCommit 콘솔 및 CodeCommit API를 사용하여 수행되는 작업과 AWS CLI를 사용하여 수행되는 명령에 적용됩니다. 이러한 권한은 Git 프로토콜을 사용하여 수행할 수 있는 비슷한 작업에는 적용되지 않습니다. 예를 들어 git commit 명령은 Git 프로토콜을 사용하여 리포지토리에 브랜치에 대한 커밋을 생성합니다. CodeCommit `CreateCommit` 작업에 대한 어떤 권한에도 영향을 받지 않습니다.

이러한 권한 중 일부를 명시적으로 거부하면 CodeCommit 콘솔에서 예상치 못한 결과가 발생할 수도 있습니다. 예를 들어 `GetTree`를 `Deny`로 설정할 경우 사용자가 콘솔에서 리포지토리의 콘텐츠를 탐색할 수 없지만 사용자가 리포지토리 내 파일의 콘텐츠를 보는 것을 차단하지는 않습니다(예: 전송된 경우, 이메일 내 파일 링크). `GetBlob`을 `Deny`로 설정할 경우 사용자가 파일의 콘텐츠를 볼 수 없지만 사용자가 리포지토리의 구조를 검색하는 것을 차단하지는 않습니다. `GetCommit`을 `Deny`로 설정할 경우 사용자가 커밋에 대한 세부 정보를 검색할 수 없습니다. `GetObjectIdentifier`를 `Deny`로 설정할 경우 코드 검색 기능이 대부분 차단됩니다. 정책에서 세 작업을 모두 `Deny`로 설정하면 해당 정책이 연결된 사용자는 CodeCommit 콘솔에서 코드를 검색할 수 없습니다.

커밋된 코드에 대한 CodeCommit API 작업 및 작입 필수 권한

BatchGetCommits

작업: `codecommit:BatchGetCommits`

CodeCommit 리포지토리에서 하나 이상의 커밋에 관한 정보를 반환하는 데 필요합니다. 이것은 IAM 정책 권한일 뿐, 직접 호출할 수 있는 API 작업은 아닙니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

CreateCommit

작업: `codecommit:CreateCommit`

커밋을 생성하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

GetCommit

작업: `codecommit:GetCommit`

커밋에 대한 정보를 반환하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

GetCommitHistory

작업: `codecommit:GetCommitHistory`

리포지토리에서 커밋의 이력에 대한 정보를 반환하는 데 필요합니다. 이것은 IAM 정책 권한일 뿐, 직접 호출할 수 있는 API 작업은 아닙니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

GetDifferences

작업: `codecommit:GetDifferences`

커밋 지정자(예: 브랜치, 태그, HEAD, 커밋 ID 또는 기타 정규화된 참조)의 차이에 대한 정보를 반환하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

GetObjectIdentifier

작업: `codecommit:GetObjectIdentifier`

BLOB, 트리 및 커밋을 해당 식별자로 확인하는 데 필요합니다. 이것은 IAM 정책 권한일 뿐, 직접 호출할 수 있는 API 작업은 아닙니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

GetReferences

작업: `codecommit:GetReferences`

브랜치, 태그와 같은 참조를 모두 반환하는 데 필요합니다. 이것은 IAM 정책 권한일 뿐, 직접 호출할 수 있는 API 작업은 아닙니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

GetTree

작업: `codecommit:GetTree`

CodeCommit 콘솔에서 CodeCommit 리포지토리의 지정된 트리의 콘텐츠를 보는 데 필요합니다. 이것은 IAM 정책 권한일 뿐, 직접 호출할 수 있는 API 작업은 아닙니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

리포지토리에 대한 작업 권한

다음 권한은 CodeCommit 리포지토리에 대한 작업을 허용하거나 거부합니다. 이러한 권한은 CodeCommit 콘솔 및 CodeCommit API를 사용하여 수행되는 작업과 AWS CLI를 사용하여 수행되는 명령에 적용됩니다. 이러한 권한은 Git 프로토콜을 사용하여 수행할 수 있는 비슷한 작업에는 적용되지 않습니다.

리포지토리에 대한 CodeCommit API 작업 및 작업 필수 권한

BatchGetRepositories

작업: `codecommit:BatchGetRepositories`

Amazon Web Services 계정에 있는 여러 CodeCommit 리포지토리에 대한 정보를 가져오는 데 필요합니다. `Resource`에는 사용자가 정보를 가져오도록 허용된(또는 거부된) 모든 CodeCommit 리포지토리의 이름을 지정해야 합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

CreateRepository

작업: `codecommit:CreateRepository`

CodeCommit 리포지토리를 생성하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

DeleteRepository

작업: `codecommit:DeleteRepository`

CodeCommit 리포지토리를 삭제하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

GetRepository

작업: `codecommit:GetRepository`

단일 CodeCommit 리포지토리에 대한 정보를 가져오는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

ListRepositories

작업: `codecommit:ListRepositories`

Amazon Web Services 계정 내 여러 CodeCommit 리포지토리의 이름 및 시스템 ID 목록을 가져오는 데 필요합니다. 이 작업의 `Resource` 에 대해 허용된 유일한 값은 모든 리포지토리(`*`)입니다.

리소스: `*`

UpdateRepositoryDescription

작업: `codecommit:UpdateRepositoryDescription`

CodeCommit 리포지토리의 설명을 변경하는 데 필요합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

UpdateRepositoryName

작업: `codecommit:UpdateRepositoryName`

CodeCommit 리포지토리의 이름을 변경하는 데 필요합니다. `Resource`에는 변경이 허용된 CodeCommit 리포지토리와 새 리포지토리 이름을 모두 지정해야 합니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

태그에 대한 작업 권한

다음 권한은 CodeCommit 리소스의 AWS 태그에 대한 작업을 허용하거나 거부합니다.

태그에 대한 CodeCommit API 작업 및 작업 필수 권한

ListTagsForResource

작업: `codecommit:ListTagsForResource`

CodeCommit의 리소스에 구성된 AWS 태그에 대한 정보를 반환하는 데 필요합니다.

리소스: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

TagResource

작업: `codecommit:TagResource`

리포지토리의 AWS 태그를 추가하거나 편집하는 데 필요합니다.

리소스: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

UntagResource

작업: `codecommit:UntagResource`

CodeCommit의 리소스에서 AWS 태그를 제거하는 데 필요합니다.

리소스: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

트리거에 대한 작업 권한

다음 권한은 CodeCommit 리포지토리용 트리거에 대한 작업을 허용하거나 거부합니다.

트리거에 대한 CodeCommit API 작업 및 작업 필수 권한

GetRepositoryTriggers

작업: `codecommit:GetRepositoryTriggers`

리포지토리용으로 구성된 트리거에 대한 정보를 반환하는 데 필요합니다.

리소스: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

PutRepositoryTriggers

작업: `codecommit:PutRepositoryTriggers`

리포지토리 트리거를 생성, 편집 또는 삭제하는 데 필요합니다.

리소스: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`

TestRepositoryTriggers

작업: `codecommit:TestRepositoryTriggers`

트리거에 대해 구성된 주제 또는 함수로 데이터를 전송하여 리포지토리 트리거의 기능을 테스트하는 데 필요합니다.

리소스: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`

CodePipeline 통합에 대한 작업 권한

CodePipeline에서 파이프라인 소스 작업에 CodeCommit 리포지토리를 사용하려면 다음 표에 나열된 모든 권한을 CodePipeline의 서비스 역할에 부여해야 합니다. 서비스 역할에서 이러한 권한이 설정되지 않거나 **Deny**로 설정될 경우 리포지토리가 변경될 경우 파이프라인이 자동으로 실행되지 않고 변경 사항을 수동으로 릴리스할 수 없습니다.

CodePipeline 통합에 대한 CodeCommit API 작업 및 작업 필수 권한

GetBranch

작업: `codecommit:GetBranch`

CodeCommit 리포지토리의 브랜치에 대한 세부 정보를 가져오는 데 필요합니다.

리소스: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`

GetCommit

작업: `codecommit:GetCommit`

커밋에 대한 정보를 반환하는 데 필요합니다.

리소스: `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`

UploadArchive

작업: `codecommit:UploadArchive`

CodePipeline 서비스 역할이 리포지토리 변경 사항을 파이프라인으로 업로드하도록 허용하는 데 필요합니다. 이것은 IAM 정책 권한일 뿐, 직접 호출할 수 있는 API 작업은 아닙니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

GetUploadArchiveStatus

작업: `codecommit:GetUploadArchiveStatus`

아카이브의 업로드 상태(진행 중, 완료, 취소 또는 오류 발생)를 결정하는 데 필요합니다. 이것은 IAM 정책 권한일 뿐, 직접 호출할 수 있는 API 작업은 아닙니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

CancelUploadArchive

작업: `codecommit:CancelUploadArchive`

파이프라인으로 아카이브 업로드를 취소하는 데 필요합니다. 이것은 IAM 정책 권한일 뿐, 직접 호출할 수 있는 API 작업은 아닙니다.

리소스: `arn:aws:codecommit:region:account-id:repository-name`

AWS CodeCommit 에서 IAM을 사용하는 방법

IAM을 사용하여 CodeCommit에 대한 액세스를 관리하려면 먼저 어떤 IAM 기능을 CodeCommit에 사용할 수 있는지를 이해해야 합니다. CodeCommit 및 기타 AWS 서비스에서 IAM을 사용하는 방법을 전체적으로 알아보려면 IAM 사용 설명서의 AWS IAM으로 작업하는 서비스를 참조하세요.

주제

- 조건 키
- 예시

조건 키

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` 요소(또는 `Condition` 블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. `Condition` 요소는 옵션입니다. 같거나 작음과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다.

한 문에서 여러 `Condition` 요소를 지정하거나 단일 `Condition` 요소에서 여러 키를 지정하는 경우, AWS 는 논리적 `AND` 작업을 사용하여 평가합니다. 단일 조건 키에 여러 값을 지정하는 경우는 논리적 `OR` 작업을 사용하여 조건을 AWS 평가합니다. 문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 예를 들어, IAM 사용자에게 IAM 사용자 이름으로 태그가 지정된 경우에만 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM 정책 요소: 변수 및 태그를 참조하세요.

AWS 는 전역 조건 키와 서비스별 조건 키를 지원합니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

CodeCommit에서는 자체 조건 키 집합을 정의하고 일부 전역 조건 키 사용도 지원합니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

일부 CodeCommit 작업은 `codecommit:References` 조건 키를 지원합니다. 이 키를 사용하는 정책 예제는 예시 4: 모든 브랜치에서 작업 허용 또는 거부 단원을 참조하세요.

CodeCommit 조건 키 목록을 보려면 IAM 사용자 설명서에서 AWS CodeCommit에 대한 조건 키를 참조하세요. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 에서 정의한 작업을 AWS CodeCommit 참조하세요.

예시

CodeCommit 자격 증명 기반 정책에 관한 예제를 보려면 AWS CodeCommit 자격 증명 기반 정책 예제 섹션을 참조하세요.

CodeCommit 리소스 기반 정책

CodeCommit은 리소스 기반 정책을 지원하지 않습니다.

CodeCommit 태그 기반 권한 부여

태그를 CodeCommit 리소스에 연결하거나 요청을 통해 태그를 CodeCommit에 전달할 수 있습니다. 태그에 근거하여 액세스를 제어하려면 `codecommit:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* 또는 `aws:TagKeys` 조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다. CodeCommit 리소스 태깅에 대한 자세한 내용은 예제 5: 태그에 따라 리포지토리에 대한 작업 거부 또는 허용 섹션을 참조하세요. 태그 지정 전략에 대한 자세한 내용은 AWS 리소스 태그 지정을 참조하세요.

CodeCommit은 세션 태그를 기반으로 하는 정책도 지원합니다. 자세한 내용은 세션 태그를 참조하세요.

CodeCommit에서 태그를 사용하여 ID 정보 제공

CodeCommit에서는 세션 태그의 사용을 지원합니다. 세션 태그는 IAM 역할을 맡거나, 임시 자격 증명을 사용하거나, AWS Security Token Service (AWS STS)에서 사용자를 연동할 때 전달하는 키-값 페어 속성입니다. IAM 사용자에 태그를 연결할 수도 있습니다. 이러한 태그에 제공된 정보를 통해 누가 변경을 수행했는지 또는 이벤트를 유발했는지 보다 쉽게 파악할 수 있습니다. CodeCommit은 CodeCommit 이벤트에서 다음과 같은 키 이름을 가진 태그의 값을 포함합니다.

키 이름	값
`displayName`	사용자를 표시하고 사용자와 연결하는 데 사용할 사람이 읽을 수 있는 이름(예: Mary Major 또는 Saanvi Sarkar)입니다.
`emailAddress`	사용자에 대해 표시하고 사용자와 연결할 이메일 주소(예: mary_major@example.com 또는 saanvi_sarkar@example.com)입니다.

이 정보가 제공되면 CodeCommit은 Amazon EventBridge 및 Amazon CloudWatch Events로 전송되는 이벤트에 해당 정보를 포함합니다. 자세한 내용은 Amazon EventBridge 및 Amazon CloudWatch Events의 CodeCommit 이벤트 모니터링 단원을 참조하십시오.

세션 태깅을 사용하려면 역할의 정책에 `sts:TagSession` 권한이 `Allow`로 설정되어 있어야 합니다. 연동 액세스를 사용하는 경우 설정 과정에서 표시 이름 및 이메일 태그 정보를 구성할 수 있습니다. 예를 들어 Azure Active Directory를 사용하는 경우 다음과 같은 클레임 정보를 제공할 수 있습니다.

클레임 이름	값
`https://aws.amazon.com/SAML/Attributes/PrincipalTag:displayNam e`	`user.displayname`
`https://aws.amazon.com/SAML/Attributes/PrincipalTag:emailAddress`	`user.mail`

를 사용하여 displayName 및 AWS CLI 에 대한 세션 태그를 전달할 수 emailAddress 있습니다AssumeRole. 예를 들어, *Developer*라는 역할을 맡고 *Mary Major*라는 이름을 연결하려는 사용자는 다음과 유사한 assume-role 명령을 사용할 수 있습니다.

```
aws sts assume-role \
--role-arn arn:aws:iam::123456789012:role/Developer \
--role-session-name Mary-Major \
--tags Key=displayName,Value="Mary Major"
 Key=emailAddress,Value="mary_major@example.com" \
--external-id Example987
```

자세한 내용은 AssumeRole을 참조하세요.

AssumeRoleWithSAML 작업을 사용하여 displayName 및 emailAddress 태그를 포함하는 임시 자격 증명 세트를 반환할 수 있습니다. CodeCommit 리포지토리에 액세스할 때 이러한 태그를 사용할 수 있습니다. 이를 위해서는 회사 또는 그룹이 이미 타사 SAML 솔루션을 통합해야 합니다 AWS. 이 경우 SAML 속성을 세션 태그로 전달할 수 있습니다. 예를 들어 *Saanvi Sarkar*라는 사용자의 표시 이름 및 이메일 주소에 대한 자격 증명 속성을 세션 태그로 전달하려면 다음을 실행합니다.

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:displayName">
  <AttributeValue>Saanvi Sarkar</AttributeValue>
</Attribute>
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:emailAddress">
  <AttributeValue>saanvi_sarkar@example.com</AttributeValue>
</Attribute>
```

자세한 내용은 AssumeRoleWithSAML을 사용하여 세션 태그 전달을 참조하세요.

AssumeRoleWithIdentity 작업을 사용하여 displayName 및 emailAddress 태그를 포함하는 임시 자격 증명 세트를 반환할 수 있습니다. CodeCommit 리포지토리에 액세스할 때 이러한 태그를 사용할 수 있습니다. OIDC(OpenID Connect)에서 세션 태그를 전달하려면 JWT(JSON 웹 토큰)에 세션 태그를 포함해야 합니다. 예를 들어, *Li Juan*이라는 사용자의 displayName 및 emailAddress 세션 태그가 포함된 AssumeRoleWithWebIdentity 직접 호출에 사용된 디코딩된 JWP 토큰은 다음과 같습니다.

```
{
    "sub": "lijuan",
    "aud": "ac_oic_client",
    "jti": "ZYUCeREXAMPLE",
    "iss": "https://xyz.com",
```

```
    "iat": 1566583294,
    "exp": 1566583354,
    "auth_time": 1566583292,
    "https://aws.amazon.com/tags": {
        "principal_tags": {
            "displayName": ["Li Juan"],
            "emailAddress": ["li_juan@example.com"],
        },
        "transitive_tag_keys": [
            "displayName",
            "emailAddress"
        ]
    }
}
```

자세한 내용은 AssumeRoleWithWebIdentity를 사용하여 세션 태그 전달을 참조하십시오.

`GetFederationToken` 작업을 사용하여 `displayName` 및 `emailAddress` 태그를 포함하는 임시 자격 증명 세트를 반환할 수 있습니다. CodeCommit 리포지토리에 액세스할 때 이러한 태그를 사용할 수 있습니다. 예를 들어 AWS CLI 를 사용하여 `displayName` 및 `emailAddress` 태그가 포함된 페더레이션 토큰을 가져오려면

```
aws sts get-federation-token \
--name my-federated-user \
--tags key=displayName,value="Nikhil Jayashankar"
 key=emailAddress,value=nikhil_jayashankar@example.com
```

자세한 내용은 GetFederationToken을 사용하여 세션 태그 전달을 참조하세요.

CodeCommit IAM 역할

IAM 역할은 특정 권한이 있는 Amazon Web Services 계정 내의 엔터티입니다.

CodeCommit에서 임시 보안 인증 정보 사용

임시 보안 인증을 사용하여 페더레이션을 통해 로그인하거나, IAM 역할을 맡거나, 교차 계정 역할을 맡을 수 있습니다. AssumeRole 또는 GetFederationToken과 같은 AWS STS API 작업을 호출하여 임시 보안 자격 증명을 얻습니다.

CodeCommit는 임시 보안 인증 정보 사용을 지원합니다. 자세한 내용은 교체 자격 증명을 사용하여 AWS CodeCommit 리포지토리에 연결 단원을 참조하십시오.

서비스 연결 역할

서비스 연결 역할을 사용하면 AWS 서비스가 다른 서비스의 리소스에 액세스하여 사용자를 대신하여 작업을 완료할 수 있습니다. 서비스 연결 역할은 IAM 계정에 나타나고 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집할 수 없습니다.

CodeCommit은 서비스 연결 역할을 사용하지 않습니다.

서비스 역할

이 기능을 사용하면 서비스가 사용자를 대신하여 서비스 역할을 수임할 수 있습니다. 이 역할을 사용하면 서비스가 다른 서비스의 리소스에 액세스해 사용자를 대신해 작업을 완료할 수 있습니다. 서비스 역할은 IAM 계정에 나타나고, 해당 계정이 소유합니다. 즉, IAM 관리자가 이 역할에 대한 권한을 변경할 수 있습니다. 그러나 권한을 변경하면 서비스의 기능이 손상될 수 있습니다.

CodeCommit은 서비스 역할을 사용하지 않습니다.

AWS CodeCommit 자격 증명 기반 정책 예제

기본적으로 IAM 사용자 및 역할은 CodeCommit 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 또한 AWS Management Console AWS CLI또는 AWS API를 사용하여 작업을 수행할 수 없습니다. IAM 관리자는 지정된 리소스에서 특정 API 작업을 수행할 수 있는 권한을 사용자와 역할에게 부여하는 IAM 정책을 생성해야 합니다. 그런 다음 관리자는 해당 권한이 필요한 IAM 사용자 또는 그룹에 이러한 정책을 연결해야 합니다.

정책 예제는 다음을 참조하세요.

- 예제 1: 사용자가 단일에서 CodeCommit 작업을 수행하도록 허용 AWS 리전
- 예제 2: 사용자에게 단일 리포지토리에서 Git을 사용하도록 허용
- 예제 3: 지정 IP 주소 범위에서 연결된 사용자에게 리포지토리에 대한 액세스 허용
- 예시 4: 모든 브랜치에서 작업 허용 또는 거부
- 예제 5: 태그에 따라 리포지토리에 대한 작업 거부 또는 허용
- 역할을 사용하여 AWS CodeCommit 리포지토리에 대한 교차 계정 액세스 구성

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 JSON 탭에서 정책 생성을 참조하세요.

주제

- 정책 모범 사례
- CodeCommit 콘솔 사용
- 사용자가 자신이 권한을 볼 수 있도록 허용
- 태그를 기반으로 CodeCommit 리포지토리 보기

정책 모범 사례

ID 기반 정책에 따라 계정에서 사용자가 CodeCommit 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 따릅니다.

- 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 정보는 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 AWS CloudFormation. 자세한 정보는 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 직접 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.

CodeCommit 콘솔 사용

AWS CodeCommit 콘솔에 액세스하려면 최소 권한 집합이 있어야 합니다. 이러한 권한은 Amazon Web Services 계정에서 CodeCommit 리소스에 대한 세부 정보를 나열하고 볼 수 있도록 허용해야 합니다. 최소 필수 권한보다 더 제한적인 보안 인증 기반 정책을 만들면 콘솔이 해당 정책에 연결된 개체(IAM 사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

이러한 엔터티가 CodeCommit 콘솔을 계속 사용할 수 있도록 하려면 다음 AWS 관리형 정책도 엔터티에 연결합니다. 자세한 내용은 IAM 사용 설명서의 사용자에게 권한 추가를 참조하세요.

자세한 내용은 CodeCommit에 대한 자격 증명 기반 정책(IAM 정책) 사용 단원을 참조하십시오.

AWS CLI 또는 AWS API만 호출하는 사용자에게는 최소 콘솔 권한을 허용할 필요가 없습니다. 그 대신, 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.

사용자가 자신이 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 포함되어 있습니다.

```
{
    "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": "*"
        }
    ]
}
```

태그를 기반으로 CodeCommit ##### 보기

자격 증명 기반 정책의 조건을 사용하여 태그를 기반으로 CodeCommit 리소스에 대한 액세스를 제어할 수 있습니다. 이 작업을 수행하는 방법을 보여 주는 정책 예제는 예제 5: 태그에 따라 리포지토리에 대한 작업 거부 또는 허용 단원을 참조하세요.

자세한 정보는 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.

자격 AWS CodeCommit 증명 및 액세스 문제 해결

다음 정보를 사용하여 CodeCommit 및 IAM 작업 시 발생할 수 있는 공통적인 문제를 진단하고 수정할 수 있습니다.

주제

- CodeCommit에서 작업을 수행할 권한이 없음
- iam:PassRole을 수행할 권한이 없음
- 액세스 키를 보아야 합니다.
- 관리자이며 다른 사용자가 CodeCommit에 액세스할 수 있게 허용하기를 원함
- 내 Amazon Web Services 계정 외부의 사람이 내 CodeCommit 리소스에 액세스하도록 허용하려고 함

CodeCommit에서 작업을 수행할 권한이 없음

에서 작업을 수행할 수 있는 권한이 없다는 AWS Management Console 메시지가 표시되면 관리자에게 문의하여 도움을 받아야 합니다. 관리자는 로그인 보안 인증 정보를 제공한 사람입니다.

자세한 내용은 CodeCommit 콘솔 사용에 필요한 권한 단원을 참조하세요.

iam:PassRole을 수행할 권한이 없음

`iam:PassRole` 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 CodeCommit에 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 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예제 오류는 `marymajor`라는 IAM 사용자가 콘솔을 사용하여 CodeCommit에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 수 있는 권한을 가지고 있지 않습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우, Mary가 `iam:PassRole` 작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 보안 인증 정보를 제공한 사람입니다.

액세스 키를 보아야 합니다.

IAM 사용자 액세스 키를 생성한 후에는 언제든지 액세스 키 ID를 볼 수 있습니다. 하지만 보안 액세스 키는 다시 볼 수 없습니다. 보안 액세스 키를 잃어버린 경우 새로운 액세스 키 페어를 생성해야 합니다.

액세스 키는 액세스 키 ID(예: `AKIAIOSFODNN7EXAMPLE`)와 보안 액세스 키(예: `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`)의 두 가지 부분으로 구성됩니다. 사용자 이름 및 암호와 같이 액세스 키 ID와 보안 액세스 키를 함께 사용하여 요청을 인증해야 합니다. 사용자 이름과 암호를 관리하는 것처럼 안전하게 액세스 키를 관리합니다.

> **Important**
>
> 정식 사용자 ID를 찾는 데 도움이 되더라도 액세스 키를 타사에 제공하지 마시기 바랍니다. 이렇게 하면 누군가에게에 대한 영구 액세스 권한을 부여할 수 있습니다 AWS 계정.

액세스 키 페어를 생성할 때는 액세스 키 ID와 보안 액세스 키를 안전한 위치에 저장하라는 메시지가 나타납니다. 보안 액세스 키는 생성할 때만 사용할 수 있습니다. 하지만 보안 액세스 키를 잃어버린 경

우 새로운 액세스 키를 IAM 사용자에게 추가해야 합니다. 최대 두 개의 액세스 키를 가질 수 있습니다. 이미 두 개가 있는 경우 새로 생성하려면 먼저 키 페어 하나를 삭제해야 합니다. 지침을 보려면 IAM 사용 설명서의 액세스 키 관리를 참조하십시오.

관리자이며 다른 사용자가 CodeCommit에 액세스할 수 있게 허용하기를 원함

다른 사용자가 CodeCommit에 액세스하도록 허용하려면 액세스 권한이 필요한 사용자 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 CodeCommit에서 올바른 권한을 부여하는 정책을 해당 엔터티에 연결해야 합니다. 권한이 부여되면 사용자 또는 애플리케이션 개발자에게 자격 증명을 제공합니다. 이들은 이 자격 증명을 사용하여 AWS에 액세스합니다. IAM 사용자, 그룹, 정책, 권한 생성에 대한 자세한 내용은 IAM 사용 설명서의 IAM ID 및 IAM 정책과 권한을 참조하세요.

내 Amazon Web Services 계정 외부의 사람이 내 CodeCommit 리소스에 액세스하도록 허용하려고 함

자세한 내용은 역할을 사용하여 AWS CodeCommit 리포지토리에 대한 교차 계정 액세스 구성 단원을 참조하십시오.

의 복원력 AWS CodeCommit

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다.는 지연 시간이 짧고 처리량이 높으며 중복성이 높은 네트워킹과 연결된 물리적으로 분리되고 격리된 여러 가용 영역을 AWS 리전 제공합니다. 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.

CodeCommit 리포지토리 또는 CodeCommit 승인 규칙 템플릿은 생성된 AWS 리전 에 있습니다. 자세한 내용은 에 대한 리전 및 Git 연결 엔드포인트 AWS CodeCommit 단원을 참조하십시오. 리포지토리의 복원성을 위해 사용자는 한 번에 두 개의 리포지토리로 푸시하도록 Git 클라이언트를 구성할 수 있습니다. 자세한 내용은 추가 Git 리포지토리로 커밋 푸시 단원을 참조하십시오.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

의 인프라 보안 AWS CodeCommit

관리형 서비스인는 Amazon Web Services: 보안 프로세스 개요 백서에 설명된 AWS 글로벌 네트워크 보안 절차로 AWS CodeCommit 보호됩니다. https://d0.awsstatic.com/whitepapers/Security/AWS_Security_Whitepaper.pdf

AWS 에서 게시한 API 호출을 사용하여 네트워크를 통해 CodeCommit에 액세스합니다. 클라이언트가 전송 계층 보안(TLS) 1.0 이상을 지원해야 합니다. TLS 1.2 이상을 권장합니다. 클라이언트는 Ephemeral Diffie-Hellman(DHE) 또는 Elliptic Curve Ephemeral Diffie-Hellman(ECDHE)과 같은 PFS(전달 완전 보안, Perfect Forward Secrecy)가 포함된 암호 제품군도 지원해야 합니다. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

요청은 액세스 키 ID 및 IAM 보안 주체와 관련된 보안 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 자격 증명을 생성하여 요청에 서명할 수 있습니다.

모든 네트워크 위치에서 이러한 API 작업을 직접 호출할 수 있지만, CodeCommit은 소스 IP 주소를 기반으로 한 제한을 지원합니다. CodeCommit 정책을 사용하여 특정 Amazon Virtual Private Cloud(VPC) 엔드포인트 또는 특정 VPC에서 액세스를 제어할 수도 있습니다. 이렇게 하면 네트워크의 특정 VPC에서만 지정된 CodeCommit 리소스에 대한 AWS 네트워크 액세스가 효과적으로 격리됩니다.

자세한 내용은 다음 자료를 참조하세요.

- 예제 1: 사용자가 단일에서 CodeCommit 작업을 수행하도록 허용 AWS 리전
- 예제 3: 지정 IP 주소 범위에서 연결된 사용자에게 리포지토리에 대한 액세스 허용
- 인터페이스 VPC 엔드포인트 AWS CodeCommit 와 함께 사용

모니터링 AWS CodeCommit

모니터링은 CodeCommit 및 기타 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 데 중요한 부분입니다.는 CodeCommit을 모니터링하고, 이상이 있을 때 보고하고, 적절한 경우 자동 조치를 취할 수 있도록 다음 모니터링 도구를 AWS 제공합니다.

- Amazon EventBridge를 사용하여 AWS 서비스를 자동화하고 애플리케이션 가용성 문제 또는 리소스 변경과 같은 시스템 이벤트에 자동으로 대응할 수 있습니다. AWS 서비스의 이벤트는 거의 실시간으로 EventBridge로 전송됩니다. 원하는 이벤트만 표시하도록 간단한 규칙을 작성한 후 규칙과 일치하는 이벤트 발생 시 실행할 자동화 작업을 지정할 수 있습니다. 자세한 내용은 Amazon EventBridge 사용 설명서 및 Amazon EventBridge 및 Amazon CloudWatch Events의 CodeCommit 이벤트 모니터링 섹션을 참조하세요.
- Amazon CloudWatch Events는 AWS 리소스의 변경 사항을 설명하는 시스템 이벤트의 스트림을 거의 실시간으로 제공합니다. CloudWatch Events를 사용하면 특정 이벤트를 감시하고 이러한 이벤트가 발생할 때 다른 AWS 서비스에서 자동화된 작업을 트리거하는 규칙을 작성할 수 있으므로 자동화된 이벤트 기반 컴퓨팅이 가능합니다. 자세한 내용은 Amazon CloudWatch Events 사용 설명서 및 Amazon EventBridge 및 Amazon CloudWatch Events의 CodeCommit 이벤트 모니터링 섹션을 참조하세요.
- Amazon CloudWatch Logs는 CloudTrail 또는 기타 소스의 로그 파일을 모니터링, 저장, 액세스하는 데 사용할 수 있습니다. CloudWatch Logs는 로그 파일의 정보를 모니터링하고 특정 임곗값에 도달하면 사용자에게 알릴 수 있습니다. 또한 매우 내구력 있는 스토리지에 로그 데이터를 저장할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서를 참조하세요.
- AWS CloudTrail 는 Amazon Web Services 계정에 의해 또는 Amazon Web Services 계정을 대신하여 수행된 API 호출 및 관련 이벤트를 캡처하고 사용자가 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 어떤 사용자 및 계정이 AWS를 호출했는지 어떤 소스 IP 주소에 호출이 이루어졌는지 언제 호출이 발생했는지 확인할 수 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서와 를 사용하여 AWS CodeCommit API 호출 로깅 AWS CloudTrail를 참조하세요.

Amazon EventBridge 및 Amazon CloudWatch Events의 CodeCommit 이벤트 모니터링

자체 애플리케이션, software-as-a-service(SaaS) 애플리케이션 및 AWS 서비스의 실시간 데이터 스트림을 제공하는 EventBridge에서 AWS CodeCommit 이벤트를 모니터링할 수 있습니다. EventBridge는 해당 데이터를 AWS Lambda 및 Amazon Simple Notification Service와 같은 대상으로 라우팅합니다.

이러한 이벤트는 AWS 리소스의 변경 사항을 설명하는 시스템 이벤트의 스트림을 거의 실시간으로 제공하는 Amazon CloudWatch Events에 나타나는 이벤트와 동일합니다.

다음은 CodeCommit 이벤트의 예제입니다.

> **Note**
>
> CodeCommit에서는 이벤트의 세션 태그에 포함된 `displayName` 및 `emailAddress` 정보를 제공할 수 있습니다(해당 정보를 사용할 수 있는 경우). 자세한 내용은 세션 태그 및 CodeCommit에서 태그를 사용하여 ID 정보 제공 섹션을 참조하세요.

주제

- referenceCreated event
- referenceUpdated event
- referenceDeleted 이벤트
- unreferencedMergeCommitCreated 이벤트
- commentOnCommitCreated 이벤트
- commentOnCommitUpdated 이벤트
- commentOnPullRequestCreated 이벤트
- commentOnPullRequestUpdated 이벤트
- pullRequestCreated 이벤트
- pullRequestSourceBranchUpdated 이벤트
- pullRequestStatusChanged 이벤트
- pullRequestMergeStatusUpdated 이벤트
- approvalRuleTemplateCreated 이벤트
- approvalRuleTemplateUpdated 이벤트
- approvalRuleTemplateDeleted 이벤트
- approvalRuleTemplateAssociatedWithRepository 이벤트
- approvalRuleTemplateDisassociatedWithRepository 이벤트
- approvalRuleTemplateBatchAssociatedWithRepositories 이벤트
- approvalRuleTemplateBatchDisassociatedFromRepositories 이벤트
- pullRequestApprovalRuleCreated 이벤트

- pullRequestApprovalRuleDeleted 이벤트
- pullRequestApprovalRuleOverridden 이벤트
- pullRequestApprovalStateChanged 이벤트
- pullRequestApprovalRuleUpdated 이벤트
- reactionCreated 이벤트
- reactionUpdated 이벤트

referenceCreated event

이 예제 이벤트에서는 myBranch라는 브랜치가 MyDemoRepo라는 리포지토리에 생성되었습니다.

```
{
   "version": "0",
   "id": "01234567-EXAMPLE",
   "detail-type": "CodeCommit Repository State Change",
   "source": "aws.codecommit",
   "account": "123456789012",
   "time": "2019-06-12T10:23:43Z",
   "region": "us-east-2",
   "resources": [
     "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
   ],
   "detail": {
     "event": "referenceCreated",
     "repositoryName": "MyDemoRepo",
     "repositoryId": "12345678-1234-5678-abcd-12345678abcd",
     "referenceType": "branch",
     "referenceName": "myBranch",
     "referenceFullName": "refs/heads/myBranch",
     "commitId": "3e5983DESTINATION"
   }
 }
```

referenceUpdated event

이 예제 이벤트에서는 myBranch라는 브랜치가 MyDemoRepo라는 리포지토리에서 병합을 통해 업데이트되었습니다.

```
{
```

```
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodeCommit Repository State Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-06-12T10:23:43Z",
    "region": "us-east-2",
    "resources": [
      "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
      "event": "referenceUpdated",
      "repositoryName": "MyDemoRepo",
      "repositoryId": "12345678-1234-5678-abcd-12345678abcd",
      "referenceType": "branch",
      "referenceName": "myBranch",
      "referenceFullName": "refs/heads/myBranch",
      "commitId": "7f0103fMERGE",
      "oldCommitId": "3e5983DESTINATION",
      "baseCommitId": "3e5a9bf1BASE",
      "sourceCommitId": "26a8f2SOURCE",
      "destinationCommitId": "3e5983DESTINATION",
      "mergeOption": "THREE_WAY_MERGE",
      "conflictDetailsLevel": "LINE_LEVEL",
      "conflictResolutionStrategy": "AUTOMERGE"
    }
}
```

referenceDeleted 이벤트

이 예제 이벤트에서는 `myBranch`라는 브랜치가 `MyDemoRepo`라는 리포지토리에서 삭제되었습니다.

```
{
  "version": "0",
  "id": "01234567-EXAMPLE",
  "detail-type": "CodeCommit Repository State Change",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-06-12T10:23:43Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
```

```
  "detail": {
    "event": "referenceDeleted",
    "repositoryName": "MyDemoRepo",
    "repositoryId": "12345678-1234-5678-abcd-12345678abcd",
    "referenceType": "branch",
    "referenceName": "myBranch",
    "referenceFullName": "refs/heads/myBranch",
    "oldCommitId": "26a8f2EXAMPLE"
  }
}
```

unreferencedMergeCommitCreated 이벤트

이 예제 이벤트에서는 참조되지 않은 병합 커밋이 MyDemoRepo라는 리포지토리에서 생성되었습니다.

```
{
  "version": "0",
  "id": "01234567-EXAMPLE",
  "detail-type": "CodeCommit Repository State Change",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-06-12T10:23:43Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
  "detail": {
    "event": "unreferencedMergeCommitCreated",
    "repositoryName": "MyDemoRepo",
    "repositoryId": "12345678-1234-5678-abcd-12345678abcd",
    "commitId": "7f0103fMERGE",
    "baseCommitId": "3e5a9bf1BASE",
    "sourceCommitId": "26a8f2SOURCE",
    "destinationCommitId": "3e5983DESTINATION",
    "mergeOption": "SQUASH_MERGE",
    "conflictDetailsLevel": "LINE_LEVEL",
    "conflictResolutionStrategy": "AUTOMERGE"
  }
}
```

commentOnCommitCreated 이벤트

이 예제 이벤트에서는 Mary_Major라는 페더레이션 사용자가 커밋에 설명을 덧붙였습니다. 이 예제에서는 페더레이션 자격 증명 공급자가 displayName 및 emailAddress에 대한 세션 태그를 구성했습니다. 해당 정보는 이벤트에 포함됩니다.

```
{
  "version": "0",
  "id": "e9dce2e9-EXAMPLE",
  "detail-type": "CodeCommit Comment on Commit",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-09-29T20:20:39Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
  "detail": {
    "beforeCommitId": "3c5dEXAMPLE",
    "repositoryId": "7dd1EXAMPLE...",
    "inReplyTo": "695bEXAMPLE...",
    "notificationBody": "A comment event occurred in the following repository:
 MyDemoRepo. The display name for the user is Mary Major. The email address for
 the user is mary_major@example.com. The user arn:aws:sts::123456789012:federated-
user/Mary_Major made a comment. The comment was made on the following comment ID:
 463bEXAMPLE.... For more information, go to the AWS CodeCommit console at https://us-
east-2.console.aws.amazon.com/codecommit/home?region=us-east-2#/repository/MyDemoRepo/
compare/3c5dEXAMPLE...f4d5EXAMPLE#463bEXAMPLE....",
    "commentId": "463bEXAMPLE...",
    "afterCommitId": "f4d5EXAMPLE",
    "event": "commentOnCommitCreated",
    "repositoryName": "MyDemoRepo",
    "callerUserArn": "arn:aws:sts::123456789012:federated-user/Mary_Major",
    "displayName": "Mary Major",
    "emailAddress": "mary_major@example.com"
  }
}
```

commentOnCommitUpdated 이벤트

이 예제 이벤트에서는 세션 이름이 `Mary_Major`이고 `Admin` 역할을 맡은 사용자가 커밋에 대한 설명을 편집했습니다. 이 예제에서는 `displayName` 및 `emailAddress`에 대해 구성된 세션 태그가 역할에 포함되었습니다. 해당 정보는 이벤트에 포함됩니다.

```
{
  "version": "0",
  "id": "98377d67-EXAMPLE",
  "detail-type": "CodeCommit Comment on Commit",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-02-09T07:15:16Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
  "detail": {
    "afterCommitId": "53812581",
    "beforeCommitId": "03314446",
    "callerUserArn": "arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major",
    "commentId": "a7e5471e-EXAMPLE",
    "event": "commentOnCommitUpdated",
    "inReplyTo": "bdb07d47-EXAMPLE",
    "notificationBody": "A comment event occurred in the following AWS
 CodeCommit repository: MyDemoRepo. The display name for the user is Mary
 Major. The email address for the user is mary_major@example.com. The user
 arn:aws:sts::123456789012:federated-user/Mary_Major updated a comment or
 replied to a comment. The comment was made on the following comment ID:
 bdb07d47-6fe9-47b0-a839-b93cc743b2ac:468cd1cb-2dfb-4f68-9636-8de52431d1d6.
 For more information, go to the AWS CodeCommit console https://us-
east-2.console.aws.amazon.com/codesuite/codecommit/repositories/MyDemoRepo/
compare/03314446461784295899698230967095822517 68/.../53812581502937833614716802771360172913 82?
region\u003dus-east-2",
    "repositoryId": "12345678-1234-1234-1234-123456789012",
    "repositoryName": "MyDemoRepo",
    "displayName": "Mary Major",
    "emailAddress": "mary_major@example.com"
  }
}
```

commentOnPullRequestCreated 이벤트

이 예제 이벤트에서는 Saanvi_Sarkar라는 페더레이션 사용자가 풀 요청에 대한 설명을 덧붙였습니다. 이 예제에서는 페더레이션 자격 증명 공급자가 displayName 및 emailAddress에 대한 세션 태그를 구성했습니다. 해당 정보는 이벤트에 포함됩니다.

```
{
  "version": "0",
  "id": "98377d67-EXAMPLE",
  "detail-type": "CodeCommit Comment on Pull Request",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-02-09T07:15:16Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
  "detail": {
    "beforeCommitId": "3c5dEXAMPLE",
    "repositoryId": "7dd1EXAMPLE...",
    "inReplyTo": "695bEXAMPLE...",
    "notificationBody": "A comment event occurred in the following AWS
 CodeCommit repository: MyDemoRepo. The display name for the user is Saanvi
 Sarkar. The email address for the user is saanvi_sarkar@example.com. The user
 arn:aws:sts::123456789012:federated-user/Saanvi_Sarkar made a comment. The comment
 was made on the following Pull Request: 201. For more information, go to the AWS
 CodeCommit console https://us-east-2.console.aws.amazon.com/codecommit/home?region=us-
east-2#/repository/MyDemoRepo/pull-request/201/activity#3276EXAMPLE...",
    "commentId": "463bEXAMPLE...",
    "afterCommitId": "f4d5EXAMPLE",
    "event": "commentOnPullRequestCreated",
    "repositoryName": "MyDemoRepo",
    "callerUserArn": "arn:aws:sts::123456789012:federated-user/Saanvi_Sarkar",
    "pullRequestId": "201",
    "displayName": "Saanvi Sarkar",
    "emailAddress": "saanvi_sarkar@example.com"
  }
}
```

commentOnPullRequestUpdated 이벤트

이 예제 이벤트에서는 Saanvi_Sarkar라는 페더레이션 사용자가 풀 요청에 대한 설명을 편집했습니다. 이 예제에서는 페더레이션 자격 증명 공급자가 displayName 및 emailAddress에 대한 세션 태그를 구성했습니다. 해당 정보는 이벤트에 포함됩니다.

```
{
  "version": "0",
  "id": "98377d67-EXAMPLE",
  "detail-type": "CodeCommit Comment on Pull Request",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-02-09T07:15:16Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
  "detail": {
    "afterCommitId": "96814774EXAMPLE",
    "beforeCommitId": "6031971EXAMPLE",
    "callerUserArn": "arn:aws:sts::123456789012:federated-user/Saanvi_Sarkar",
    "commentId": "40cb52f0-EXAMPLE",
    "event": "commentOnPullRequestUpdated",
    "inReplyTo": "1285e713-EXAMPLE",
    "notificationBody": "A comment event occurred in the following AWS
 CodeCommit repository: MyDemoRepo. The display name for the user is Saanvi
 Sarkar. The email address for the user is saanvi_sarkar@example.com. The user
 arn:aws:sts::123456789012:federated-user/Saanvi_Sarkar updated a comment or
 replied to a comment. The comment was made on the following Pull Request:
 1. For more information, go to the AWS CodeCommit console https://us-
east-2.console.aws.amazon.com/codesuite/codecommit/repositories/MyDemoRepo/pull-
requests/1/activity#40cb52f0-aac7-4c43-b771-601eff02EXAMPLE",
    "pullRequestId": "1",
    "repositoryId": "12345678-1234-1234-1234-123456789012",
    "repositoryName": "MyDemoRepo"
  }
}
```

pullRequestCreated 이벤트

이 예제 이벤트에서는 세션 이름이 `Mary_Major`이고 `Admin` 역할을 맡은 사용자가 `MyDemoRepo`라는 리포지토리에서 풀 요청을 생성했습니다. 제공된 세션 태그 정보가 없으므로 이벤트에 정보가 포함되지 않습니다.


```
{
  "version": "0",
  "id": "98377d67-EXAMPLE",
  "detail-type": "CodeCommit Pull Request State Change",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-02-09T07:15:16Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
  "detail": {
    "author": "arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major",
    "callerUserArn": "arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major",
    "creationDate": "Tue Feb 9 2019 10:18:42 PDT ",
    "description": "An example description.",
    "destinationCommit": "12241970EXAMPLE",
    "destinationReference": "refs/heads/main",
    "event": "pullRequestCreated",
    "isMerged": "False",
    "lastModifiedDate": "Tue Feb 9 2019 10:18:42 PDT",
    "notificationBody": "A pull request event occurred in the following AWS CodeCommit
 repository: MyDemoRepo. User: arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major.
 Event: Created. The pull request was created with the following information: Pull
 Request ID as 1 and title as My Example Pull Request. For more information, go to the
 AWS CodeCommit console https://us-east-2.console.aws.amazon.com/codesuite/codecommit/
repositories/MyDemoRepo/pull-requests/1",
    "pullRequestId": "1",
    "pullRequestStatus": "Open",
    "repositoryNames": ["MyDemoRepo"],
    "revisionId": "bdc0cb9bEXAMPLE",
    "sourceCommit": "2774290EXAMPLE",
    "sourceReference": "refs/heads/test-branch",
    "title": "My Example Pull Request"
  }
}
```

pullRequestSourceBranchUpdated 이벤트

이 예제 이벤트에서는 세션 이름이 `Mary_Major`이고 `Admin` 역할을 맡은 사용자가 ID가 1인 풀 요청에 대해 `test-branch`라는 소스 브랜치를 업데이트했습니다.

```
{
  "version": "0",
  "id": "98377d67-EXAMPLE",
  "detail-type": "CodeCommit Pull Request State Change",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-02-09T07:15:16Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
  "detail": {
    "author": "arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major",
    "callerUserArn": "arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major",
    "creationDate": "Tue Feb 9 2019 10:18:42 PDT",
    "description": "An example description.",
    "destinationCommit": "7644990EXAMPLE",
    "destinationReference": "refs/heads/main",
    "event": "pullRequestSourceBranchUpdated",
    "isMerged": "False",
    "lastModifiedDate": "Tue Feb 9 2019 10:18:42 PDT",
    "notificationBody": "A pull request event occurred in the following AWS
 CodeCommit repository: MyDemoRepo. User: arn:aws:sts::123456789012:assumed-role/
Admin/Mary_Major. Event: Updated.  The user updated the following pull request:
 1. The pull request was updated with one or more commits to the source branch:
 test-branch. For more information, go to the AWS CodeCommit console https://us-
east-2.console.aws.amazon.com/codesuite/codecommit/repositories/MyDemoRepo/pull-
requests/1?region\u003dus-east-2",
    "pullRequestId": "1",
    "pullRequestStatus": "Open",
    "repositoryNames": ["MyDemoRepo"],
    "revisionId": "bdc0cb9b4EXAMPLE",
    "sourceCommit": "64875001EXAMPLE",
    "sourceReference": "refs/heads/test-branch",
    "title": "My Example Pull Request"
  }
}
```

pullRequestStatusChanged 이벤트

이 예제 이벤트에서는 세션 이름이 `Mary_Major`이고 `Admin` 역할을 맡은 사용자가 ID가 1인 풀 요청을 닫았습니다. 풀 요청이 병합되지 않았습니다.

```
{
  "version": "0",
  "id": "98377d67-EXAMPLE",
  "detail-type": "CodeCommit Pull Request State Change",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-02-09T07:15:16Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
  "detail": {
    "author": "arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major",
    "callerUserArn": "arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major",
    "creationDate": "Tue Jun 18 10:34:20 PDT 2019",
    "description": "An example description.",
    "destinationCommit": "95149731EXAMPLE",
    "destinationReference": "refs/heads/main",
    "event": "pullRequestStatusChanged",
    "isMerged": "False",
    "lastModifiedDate": "Tue Jun 18 10:34:20 PDT 2019",
    "notificationBody": "A pull request event occurred in the following AWS CodeCommit
 repository: MyDemoRepo. arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major
 updated the following PullRequest 1. The pull request status has been updated. The
 status is closed. For more information, go to the AWS CodeCommit console https://
us-east-2.console.aws.amazon.com/codesuite/codecommit/repositories/MyDemoRepo/pull-
requests/1?region\u003dus-east-2",
    "pullRequestId": "1",
    "pullRequestStatus": "Closed",
    "repositoryNames": ["MyDemoRepo"],
    "revisionId": "bdc0cb9bEXAMPLE",
    "sourceCommit": "4409936EXAMPLE",
    "sourceReference": "refs/heads/test-branch",
    "title": "My Example Pull Request"
  }
}
```

pullRequestMergeStatusUpdated 이벤트

이 예제 이벤트에서는 세션 이름이 `Mary_Major`이고 `Admin` 역할을 맡은 사용자가 ID가 1인 풀 요청을 병합했습니다.

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "CodeCommit Pull Request State Change",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-06-12T10:23:43Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
  "detail": {
    "author": "arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major",
    "callerUserArn": "arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major",
    "creationDate": "Mon Mar 11 14:42:31 PDT 2019",
    "description": "An example description.",
    "destinationCommit": "4376719EXAMPLE",
    "destinationReference": "refs/heads/main",
    "event": "pullRequestMergeStatusUpdated",
    "isMerged": "True",
    "lastModifiedDate": "Mon Mar 11 14:42:31 PDT 2019",
    "mergeOption": "FAST_FORWARD_MERGE",
    "notificationBody": "A pull request event occurred in the following AWS CodeCommit
 repository: MyDemoRepo. arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major
 updated the following PullRequest 1. The pull request merge status has been updated.
 The status is merged. For more information, go to the AWS CodeCommit console https://
us-east-2.console.aws.amazon.com/codesuite/codecommit/repositories/MyDemoRepo/pull-
requests/1?region\u003dus-east-2",
    "pullRequestId": "1",
    "pullRequestStatus": "Closed",
    "repositoryNames": ["MyDemoRepo"],
    "revisionId": "bdc0cb9beEXAMPLE",
    "sourceCommit": "0701696EXAMPLE",
    "sourceReference": "refs/heads/test-branch",
    "title": "My Example Pull Request"
  }
}
```

approvalRuleTemplateCreated 이벤트

이 예제 이벤트에서는 IAM 사용자 이름이 `Mary_Major`인 사용자가 `2-approvers-required-for-main`라는 승인 규칙 템플릿을 생성했습니다.

```
{
    "version": "0",
    "id": "f7702227-EXAMPLE",
    "detail-type": "CodeCommit Approval Rule Template Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:02:27Z",
    "region": "us-east-2",
    "resources": [],
    "detail": {
        "approvalRuleTemplateContentSha256": "f742eebbEXAMPLE",
        "approvalRuleTemplateId": "d7385967-EXAMPLE",
        "approvalRuleTemplateName": "2-approvers-required-for-main",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:02:14 UTC 2019",
        "event": "approvalRuleTemplateCreated",
        "lastModifiedDate": "Wed Nov 06 19:02:14 UTC 2019",
        "notificationBody": "A approval rule template event occurred in the following
 AWS CodeCommit account: 123456789012. User: arn:aws:iam::123456789012:user/Mary_Major.
  Additional information: An approval rule template with the following name has been
 created: 2-approvers-required-for-main. The ID of the created template is: d7385967-
EXAMPLE. For more information, go to the AWS CodeCommit console.",
        "repositories": {}
    }
}
```

approvalRuleTemplateUpdated 이벤트

이 예제 이벤트에서는 IAM 사용자 이름이 `Mary_Major`인 사용자가 `2-approvers-required-for-main`라는 승인 규칙 템플릿을 편집했습니다. 승인 규칙 템플릿은 리포지토리와 연결되어 있지 않습니다.

```
{
  "version": "0",
  "id": "66403118-EXAMPLE",
  "detail-type": "CodeCommit Approval Rule Template Change",
```

```
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-11-12T23:03:30Z",
  "region": "us-east-2",
  "resources": [

  ],
  "detail": {
    "approvalRuleTemplateContentSha256": "f742eebbEXAMPLE",
    "approvalRuleTemplateId": "c9d2b844-EXAMPLE",
    "approvalRuleTemplateName": "2-approvers-required-for-main",
    "callerUserArn": "arn:aws:iam::123456789012:user\Mary_Major",
    "creationDate": "Tue Nov 12 23:03:06 UTC 2019",
    "event": "approvalRuleTemplateDeleted",
    "lastModifiedDate": "Tue Nov 12 23:03:20 UTC 2019",
    "notificationBody": "A approval rule template event occurred in the following AWS
 CodeCommit account: 123456789012. User: arn:aws:iam::123456789012:user\Mary_Major.
 Additional information: An approval rule template with the following name has been
 deleted: 2-approvers-required-for-main. The ID of the updated template is: c9d2b844-
EXAMPLE. For more information, go to the AWS CodeCommit console.",
    "repositories": {}
  }
}
```

approvalRuleTemplateDeleted 이벤트

이 예제 이벤트에서는 IAM 사용자 이름이 `Mary_Major`인 사용자가 `2-approvers-required-for-main`라는 승인 규칙 템플릿을 삭제했습니다. 승인 규칙 템플릿은 리포지토리와 연결되어 있지 않습니다.

```
{
  "version": "0",
  "id": "66403118-EXAMPLE",
  "detail-type": "CodeCommit Approval Rule Template Change",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-11-12T23:03:30Z",
  "region": "us-east-2",
  "resources": [],
  "detail": {
    "approvalRuleTemplateContentSha256": "4f3de6632EXAMPLE",
    "approvalRuleTemplateId": "c9d2b844-EXAMPLE",
    "approvalRuleTemplateName": "2-approvers-required-for-main",
```

```
    "callerUserArn": "arn:aws:iam::123456789012:user\Mary_Major",
    "creationDate": "Tue Nov 12 23:03:06 UTC 2019",
    "event": "approvalRuleTemplateUpdated",
    "lastModifiedDate": "Tue Nov 12 23:03:20 UTC 2019",
    "notificationBody": "A approval rule template event occurred in the following AWS
 CodeCommit account: 123456789012. User: arn:aws:iam::123456789012:user\Mary_Major.
 Additional information: An approval rule template with the following name has
 been updated: 2-approvers-required-for-main. The ID of the updated template is:
 c9d2b844-EXAMPLE. The after rule template content SHA256 is 4f3de663EXAMPLE. For more
 information, go to the AWS CodeCommit console.",
    "repositories": {}
  }
}
```

approvalRuleTemplateAssociatedWithRepository 이벤트

이 예제 이벤트에서는 IAM 사용자 이름이 `Mary_Major`인 사용자가 `2-approvers-required-for-main`라는 승인 규칙 템플릿을 `MyDemoRepo`라는 리포지토리에 연결했습니다.

```
{
    "version": "0",
    "id": "ea1c6d73-EXAMPLE",
    "detail-type": "CodeCommit Approval Rule Template Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:02:27Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "approvalRuleTemplateContentSha256": "f742eebbEXAMPLE",
        "approvalRuleTemplateId": "d7385967-EXAMPLE",
        "approvalRuleTemplateName": "2-approvers-required-for-main",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:02:14 UTC 2019",
        "event": "approvalRuleTemplateAssociatedWithRepository",
        "lastModifiedDate": "Wed Nov 06 19:02:14 UTC 2019",
        "notificationBody": "A approval rule template event occurred in the following
 AWS CodeCommit account: 123456789012. User: arn:aws:iam::123456789012:user/Mary_Major.
 Additional information: An approval rule template has been associated with the
 following repository: [MyDemoRepo]. For more information, go to the AWS CodeCommit
 console.",
```

```
        "repositories": {
            "MyDemoRepo": "92ca7bf2-d878-49ed-a994-336a6cc7c574"
        }
    }
}
```

approvalRuleTemplateDisassociatedWithRepository 이벤트

이 예제 이벤트에서는 IAM 사용자 이름이 `Mary_Major`인 사용자가 `2-approvers-required-for-main`라는 승인 규칙 템플릿을 `MyDemoRepo`라는 리포지토리에서 연결 해제했습니다.

```
{
    "version": "0",
    "id": "ea1c6d73-EXAMPLE",
    "detail-type": "CodeCommit Approval Rule Template Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:02:27Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "approvalRuleTemplateContentSha256": "f742eebbEXAMPLE",
        "approvalRuleTemplateId": "d7385967-EXAMPLE",
        "approvalRuleTemplateName": "2-approvers-required-for-main",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:02:14 UTC 2019",
        "event": "approvalRuleTemplateDisassociatedFromRepository",
        "lastModifiedDate": "Wed Nov 06 19:02:14 UTC 2019",
        "notificationBody": "A approval rule template event occurred in the following
 AWS CodeCommit account: 123456789012. User: arn:aws:iam::123456789012:user/Mary_Major.
 Additional information: An approval rule template has been disassociated from the
 following repository: [MyDemoRepo]. For more information, go to the AWS CodeCommit
 console.",
        "repositories": {
            "MyDemoRepo": "92ca7bf2-d878-49ed-a994-336a6cc7c574"
        }
    }
}
```

approvalRuleTemplateBatchAssociatedWithRepositories 이벤트

이 예제 이벤트에서는 IAM 사용자 이름이 `Mary_Major`인 사용자가 `2-approvers-required-for-main`라는 승인 규칙 템플릿을 `MyDemoRepo`라는 리포지토리 및 `MyTestRepo`라는 리포지토리에 일괄적으로 연결했습니다.

```
{
    "version": "0",
    "id": "0f861e5b-EXAMPLE",
    "detail-type": "CodeCommit Approval Rule Template Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-12T23:39:09Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "approvalRuleTemplateContentSha256": "f742eebbEXAMPLE",
        "approvalRuleTemplateId": "c71c1fe0-EXAMPLE",
        "approvalRuleTemplateName": "2-approvers-required-for-main",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Tue Nov 12 23:38:57 UTC 2019",
        "event": "batchAssociateApprovalRuleTemplateWithRepositories",
        "lastModifiedDate": "Tue Nov 12 23:38:57 UTC 2019",
        "notificationBody": "A approval rule template event occurred in the following
 AWS CodeCommit account: 123456789012. User: arn:aws:iam::123456789012:user\Mary_Major.
 Additional information: An approval rule template has been batch associated with the
 following repository names: [MyDemoRepo, MyTestRepo]. For more information, go to the
 AWS CodeCommit console.",
        "repositories": {
            "MyDemoRepo": "MyTestRepo"
        }
    }
}
```

approvalRuleTemplateBatchDisassociatedFromRepositories 이벤트

이 예제 이벤트에서는 IAM 사용자 이름이 `Mary_Major`인 사용자가 `2-approvers-required-for-main`라는 승인 규칙 템플릿을 `MyDemoRepo`라는 리포지토리 및 `MyTestRepo`라는 리포지토리에서 일괄적으로 연결 해제했습니다.

```
{
    "version": "0",
    "id": "e08fc996-EXAMPLE",
    "detail-type": "CodeCommit Approval Rule Template Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-12T23:39:09Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "approvalRuleTemplateContentSha256": "f742eebbEXAMPLE",
        "approvalRuleTemplateId": "c71c1fe0-ff91-4db4-9a45-a86a7b6c474f",
        "approvalRuleTemplateName": "2-approvers-required-for-main",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Tue Nov 12 23:38:57 UTC 2019",
        "event": "batchDisassociateApprovalRuleTemplateFromRepositories",
        "lastModifiedDate": "Tue Nov 12 23:38:57 UTC 2019",
        "notificationBody": "A approval rule template event occurred in the following
 AWS CodeCommit account: 123456789012. User: arn:aws:iam::123456789012:user/Mary_Major.
 Additional information: An approval rule template has been batch disassociated from
 the following repository names: [MyDemoRepo, MyTestRepo]. For more information, go to
 the AWS CodeCommit console.",
        "repositories": {
            "MyDemoRepo": "MyTestRepo"
        }
    }
}
```

pullRequestApprovalRuleCreated 이벤트

이 예제 이벤트에서는 IAM 사용자 이름이 `Mary_Major`인 사용자가 ID가 227인 풀 요청에 대해 `1-approver-needed`라는 승인 규칙을 생성했습니다.

```
{
    "version": "0",
    "id": "ad860f12-EXAMPLE",
    "detail-type": "CodeCommit Pull Request State Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:12:19Z",
```

```
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "approvalRuleContentSha256": "f742eebbEXAMPLE",
        "approvalRuleId": "0a9b5dfc-EXAMPLE",
        "approvalRuleName": "1-approver-needed",
        "author": "arn:aws:iam::123456789012:user/Mary_Major",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:10:58 UTC 2019",
        "description": "An An example description.",
        "destinationCommit": "194fdf00EXAMPLE",
        "destinationReference": "refs/heads/main",
        "event": "pullRequestApprovalRuleCreated",
        "isMerged": "False",
        "lastModifiedDate": "Wed Nov 06 19:10:58 UTC 2019",
        "notificationBody": "A pull request event occurred in the following AWS
 CodeCommit repository: MyDemoRepo. User: arn:aws:iam::123456789012:user/Mary_Major.
 Event: Updated. Pull request: 227. Additional information: An approval rule has been
 created with the following name: 1-approver-needed. For more information, go to the
 AWS CodeCommit console https://us-east-2.console.aws.amazon.com/codesuite/codecommit/
repositories/MyDemoRepo/pull-requests/227?region=us-east-2",
        "pullRequestId": "227",
        "pullRequestStatus": "Open",
        "repositoryNames": [
            "MyDemoRepo"
        ],
        "revisionId": "3b8cecab3EXAMPLE",
        "sourceCommit": "29964a17EXAMPLE",
        "sourceReference": "refs/heads/test-branch",
        "title": "My example pull request"
    }
}
```

pullRequestApprovalRuleDeleted 이벤트

이 예제 이벤트에서는 IAM 사용자 이름이 `Mary_Major`인 사용자가 ID가 227인 풀 요청에 대해 `1-approver-needed`라는 승인 규칙을 삭제했습니다. `Saanvi_Sarkar` 이름을 가진 IAM 사용자가 원래 승인 규칙을 작성했습니다.

```
{
    "version": "0",
```

```
    "id": "c1c3509d-EXAMPLE",
    "detail-type": "CodeCommit Pull Request State Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:12:19Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "approvalRuleContentSha256": "f742eebbEXAMPLE",
        "approvalRuleId": "0a9b5dfc-EXAMPLE",
        "approvalRuleName": "1-approver-needed",
        "author": "arn:aws:iam::123456789012:user/Saanvi_Sarkar",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:10:58 UTC 2019",
        "description": "An An example description.",
        "destinationCommit": "194fdf00EXAMPLE",
        "destinationReference": "refs/heads/main",
        "event": "pullRequestApprovalRuleDeleted",
        "isMerged": "False",
        "lastModifiedDate": "Wed Nov 06 19:10:58 UTC 2019",
        "notificationBody": "A pull request event occurred in the following AWS
 CodeCommit repository: MyDemoRepo. User: arn:aws:iam::123456789012:user/Mary_Major.
 Event: Created. Pull request: 227. Additional information: An approval rule has been
 deleted: 1-approver-needed was deleted. For more information, go to the AWS CodeCommit
 console https://us-east-2.console.aws.amazon.com/codesuite/codecommit/repositories/
MyDemoRepo/pull-requests/227?region=us-east-2",
        "pullRequestId": "227",
        "pullRequestStatus": "Open",
        "repositoryNames": [
            "MyDemoRepo"
        ],
        "revisionId": "3b8cecabEXAMPLE",
        "sourceCommit": "29964a17EXAMPLE",
        "sourceReference": "refs/heads/test-branch",
        "title": "My example pull request"
    }
}
```

pullRequestApprovalRuleOverridden 이벤트

이 예제 이벤트에서는 IAM 사용자 이름이 `Mary_Major`인 사용자가 풀 요청에 대한 승인 규칙 요구 사항을 무시(OVERRIDE)했습니다. IAM 사용자 이름이 `Li_Juan`인 사용자가 풀 요청을 작성했습니다.

```
{
    "version": "0",
    "id": "52d2cb73-EXAMPLE",
    "detail-type": "CodeCommit Pull Request State Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:12:19Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "author": "arn:aws:iam::123456789012:user/Li_Juan",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:10:58 UTC 2019",
        "description": "An An example description.",
        "destinationCommit": "194fdf00EXAMPLE",
        "destinationReference": "refs/heads/main",
        "event": "pullRequestApprovalRuleOverridden",
        "isMerged": "False",
        "lastModifiedDate": "Wed Nov 06 19:10:58 UTC 2019",
        "notificationBody": "A pull request event occurred in the following AWS
 CodeCommit repository: MyDemoRepo. User: arn:aws:iam::123456789012:user/Mary_Major.
 Event: Updated. Pull request name: 227. Additional information: An override
 event has occurred for the approval rules for this pull request. Override status:
 OVERRIDE. For more information, go to the AWS CodeCommit console https://us-
east-2.console.aws.amazon.com/codesuite/codecommit/repositories/MyDemoRepo/pull-
requests/227?region=us-east-2",
        "overrideStatus": "OVERRIDE",
        "pullRequestId": "227",
        "pullRequestStatus": "Open",
        "repositoryNames": [
            "MyDemoRepo"
        ],
        "revisionId": "3b8cecabEXAMPLE",
        "sourceCommit": "29964a17EXAMPLE",
        "sourceReference": "refs/heads/test-branch",
        "title": "My example pull request"
```

```
    }
}
```

이 예제 이벤트에서는 풀 요청에 대한 승인 규칙 요구 사항이 복구되었습니다(REVOKE).

```
{
    "version": "0",
    "id": "2895482d-13eb-b783-270d-76588e6029fa",
    "detail-type": "CodeCommit Pull Request State Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:12:19Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "author": "arn:aws:iam::123456789012:user/Li_Juan",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:10:58 UTC 2019",
        "description": "An An example description.",
        "destinationCommit": "194fdf00EXAMPLE",
        "destinationReference": "refs/heads/main",
        "event": "pullRequestApprovalRuleOverridden",
        "isMerged": "False",
        "lastModifiedDate": "Wed Nov 06 19:10:58 UTC 2019",
        "notificationBody": "A pull request event occurred in the following
 AWS CodeCommit repository: MyDemoRepo. User: arn:aws:iam::123456789012:user/
Mary_Major. Event: Updated. Pull request name: 227. Additional information: An
 override event has occurred for the approval rules for this pull request. Override
 status: REVOKE. For more information, go to the AWS CodeCommit console https://
us-east-2.console.aws.amazon.com/codesuite/codecommit/repositories/MyDemoRepo/pull-
requests/227?region=us-east-2",
        "overrideStatus": "REVOKE",
        "pullRequestId": "227",
        "pullRequestStatus": "Open",
        "repositoryNames": [
            "MyDemoRepo"
        ],
        "revisionId": "3b8cecabEXAMPLE",
        "sourceCommit": "29964a17EXAMPLE",
        "sourceReference": "refs/heads/test-branch",
        "title": "My example pull request"
```

```
    }
}
```

pullRequestApprovalStateChanged 이벤트

이 예제 이벤트에서는 IAM 사용자 이름이 `Mary_Major`인 사용자가 풀 요청을 승인했습니다.

```
{
    "version": "0",
    "id": "53e5d7e9-986c-1ebf-9d8b-ebef5596da0e",
    "detail-type": "CodeCommit Pull Request State Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:12:19Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "approvalStatus": "APPROVE",
        "author": "arn:aws:iam::123456789012:user/Li_Juan",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:10:58 UTC 2019",
        "description": "An An example description.",
        "destinationCommit": "194fdf00EXAMPLE",
        "destinationReference": "refs/heads/main",
        "event": "pullRequestApprovalStateChanged",
        "isMerged": "False",
        "lastModifiedDate": "Wed Nov 06 19:10:58 UTC 2019",
        "notificationBody": "A pull request event occurred in the following
 AWS CodeCommit repository: MyDemoRepo. User: arn:aws:iam::123456789012:user/
Mary_Major. Event: Updated. Pull request name: 227. Additional information:
 A user has changed their approval state for the pull request. State change:
 APPROVE. For more information, go to the AWS CodeCommit console https://us-
east-2.console.aws.amazon.com/codesuite/codecommit/repositories/MyDemoRepo/pull-
requests/227?region=us-east-2",
        "pullRequestId": "227",
        "pullRequestStatus": "Open",
        "repositoryNames": [
            "MyDemoRepo"
        ],
        "revisionId": "3b8cecabEXAMPLE",
        "sourceCommit": "29964a17EXAMPLE",
```

```
        "sourceReference": "refs/heads/test-branch",
        "title": "My example pull request"
    }
}
```

이 예제 이벤트에서는 IAM 사용자 이름이 `Mary_Major`인 사용자가 풀 요청에 대한 승인을 취소했습니다.

```
{
    "version": "0",
    "id": "25e183d7-d01a-4e07-2bd9-b2d56ebecc81",
    "detail-type": "CodeCommit Pull Request State Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:12:19Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "approvalStatus": "REVOKE",
        "author": "arn:aws:iam::123456789012:user/Li_Juan",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:10:58 UTC 2019",
        "description": "An An example description.",
        "destinationCommit": "194fdf00EXAMPLE",
        "destinationReference": "refs/heads/main",
        "event": "pullRequestApprovalStateChanged",
        "isMerged": "False",
        "lastModifiedDate": "Wed Nov 06 19:10:58 UTC 2019",
        "notificationBody": "A pull request event occurred in the following AWS
 CodeCommit repository: MyDemoRepo. User: arn:aws:iam::123456789012:user/Mary_Major.
 Event: Updated. Pull request name: 227. Additional information: A user has changed
 their approval state for the pull request. State change: REVOKE. For more information,
 go to the AWS CodeCommit console https://us-east-2.console.aws.amazon.com/codesuite/
codecommit/repositories/MyDemoRepo/pull-requests/227?region=us-east-2",
        "pullRequestId": "227",
        "pullRequestStatus": "Open",
        "repositoryNames": [
            "MyDemoRepo"
        ],
        "revisionId": "3b8cecabEXAMPLE",
        "sourceCommit": "29964a17EXAMPLE",
```

```
        "sourceReference": "refs/heads/test-branch",
        "title": "My example pull request"
    }
}
```

pullRequestApprovalRuleUpdated 이벤트

이 예제 이벤트에서는 IAM 사용자 이름이 `Mary_Major`인 사용자가 풀 요청에 대한 승인 규칙을 편집했습니다. 해당 사용자가 풀 요청을 작성한 사용자이기도 합니다.

```
{
    "version": "0",
    "id": "21b1c819-2889-3528-1cb8-3861aacf9d42",
    "detail-type": "CodeCommit Pull Request State Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:12:19Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "approvalRuleContentSha256": "f742eebbEXAMPLE",
        "approvalRuleId": "0a9b5dfc-EXAMPLE",
        "approvalRuleName": "1-approver-needed",
        "author": "arn:aws:iam::123456789012:user/Mary_Major",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:10:58 UTC 2019",
        "description": "An example description.",
        "destinationCommit": "194fdf00EXAMPLE",
        "destinationReference": "refs/heads/main",
        "event": "pullRequestApprovalRuleUpdated",
        "isMerged": "False",
        "lastModifiedDate": "Wed Nov 06 19:10:58 UTC 2019",
        "notificationBody": "A pull request event occurred in the following
 AWS CodeCommit repository: MyDemoRepo. User: arn:aws:iam::123456789012:user/
Mary_Major. Event: Updated. Pull request name: 227. The content of an approval
 rule has been updated for the pull request. The name of the updated rule is: 1-
approver-needed. For more information, go to the AWS CodeCommit console https://
us-east-2.console.aws.amazon.com/codesuite/codecommit/repositories/MyDemoRepo/pull-
requests/227?region=us-east-2",
        "pullRequestId": "227",
        "pullRequestStatus": "Open",
```

```
        "repositoryNames": [
            "MyDemoRepo"
        ],
        "revisionId": "3b8cecab3EXAMPLE",
        "sourceCommit": "29964a17EXAMPLE",
        "sourceReference": "refs/heads/test-branch",
        "title": "My example pull request"
    }
}
```

reactionCreated 이벤트

이 예제 이벤트에서는 IAM 사용자 이름이 `Mary_Major`인 사용자가 주석에 대한 반응을 추가했습니다.

```
{
   "version":"0",
   "id":"59fcccd8-217a-32ce-2b05-561ed68a1c42",
   "detail-type":"CodeCommit Comment Reaction Change",
   "source":"aws.codecommit",
   "account":"123456789012",
   "time":"2020-04-14T00:49:03Z",
   "region":"us-east-2",
   "resources":[
      "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
   ],
   "detail":{
      "callerUserArn":"arn:aws:iam::123456789012:user/Mary_Major",
      "commentId":"28930161-EXAMPLE",
      "event":"commentReactionCreated",
      "notificationBody":"A comment reaction event occurred in the following AWS
 CodeCommit Repository: MyDemoRepo. The user: arn:aws:iam::123456789012:user/Mary_Major
 made a comment reaction # to the comment with comment ID: 28930161-EXAMPLE",
      "reactionEmojis":["#"],
      "reactionShortcodes":[":thumbsdown:"],
      "reactionUnicodes":["U+1F44E"],
      "repositoryId":"12345678-1234-5678-abcd-12345678abcd",
      "repositoryName":"MyDemoRepo"
   }
}
```

reactionUpdated 이벤트

이 예제 이벤트에서는 IAM 사용자 이름이 `Mary_Major`인 사용자가 주석에 대한 반응을 업데이트했습니다. 사용자는 자신의 반응만 업데이트할 수 있습니다.

```
{
   "version":"0",
   "id":"0844ed99-a53f-3bdb-6048-4de315516889",
   "detail-type":"CodeCommit Comment Reaction Change",
   "source":"aws.codecommit",
   "account":"123456789012",
   "time":"2020-04-22T23:19:42Z",
   "region":"us-east-2",
   "resources":[
      "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
   ],
   "detail":{
      "callerUserArn":"arn:aws:iam::123456789012:user/Mary_Major",
      "commentId":"28930161-EXAMPLE",
      "event":"commentReactionUpdated",
      "notificationBody":"A comment reaction event occurred in the following AWS
 CodeCommit Repository: MyDemoRepo. The user: arn:aws:iam::123456789012:user/Mary_Major
 updated a reaction :smile: to the comment with comment ID: 28930161-EXAMPLE",
      "reactionEmojis":[
         "#"
      ],
      "reactionShortcodes":[
         ":smile:"
      ],
      "reactionUnicodes":[
         "U+1F604"
      ],
      "repositoryId":"12345678-1234-5678-abcd-12345678abcd",
      "repositoryName":"MyDemoRepo"
   }
}
```

를 사용하여 AWS CodeCommit API 호출 로깅 AWS CloudTrail

CodeCommit은 CodeCommit에서 사용자 AWS CloudTrail, 역할 또는 서비스가 수행한 작업에 대한 레코드를 제공하는 AWS 서비스와 통합됩니다. CloudTrail은 CodeCommit에 대한 모든 API 호출을 이벤

트로 캡쳐합니다. 여기에는 CodeCommit 콘솔과 Git 클라이언트의 호출, CodeCommit API에 대한 코드 호출 등이 포함됩니다. 추적을 생성하면 CodeCommit 이벤트를 포함한 CloudTrail 이벤트를 지속적으로 Amazon S3 버킷에 배포할 수 있습니다. 추적을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 CodeCommit에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

CloudTrail의 CodeCommit 정보

CloudTrail은 계정 생성 시 Amazon Web Services 계정에서 활성화됩니다. CodeCommit에서 활동이 발생하면 해당 활동이 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. Amazon Web Services 계정에서 최신 이벤트를 확인, 검색 및 다운로드할 수 있습니다. 자세한 정보는 CloudTrail 이벤트 기록을 사용하여 이벤트 보기를 참조하세요.

CodeCommit에 대한 이벤트를 포함하여 Amazon Web Services 계정에 이벤트를 지속적으로 기록하려면 추적을 생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 트레일을 생성하면 기본적으로 모든 리전에 트레일이 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.

- 트레일 생성 개요
- CloudTrail 지원 서비스 및 통합
- CloudTrail에서 Amazon SNS 알림 구성
- 여러 리전으로부터 CloudTrail 로그 파일 받기 및 여러 계정으로부터 CloudTrail 로그 파일 받기

Amazon Web Services 계정에서 CloudTrail 로깅이 활성화되면 CodeCommit 작업에 대한 API 호출이 CloudTrail 로그 파일에서 추적되며, 여기서 다른 AWS 서비스 레코드와 함께 기록됩니다. CloudTrail은 기간 및 파일 크기를 기준으로 새 파일을 만들고 기록하는 시점을 결정합니다.

모든 CodeCommit 작업은 CloudTrail에 의해 기록됩니다. 여기에는 현재 AWS CodeCommit API 참조에 문서화되어 있지 않지만 그 대신 액세스 권한으로 참조되고 CodeCommit 권한 참조에 문서화된 작업들(예:`GetObjectIdentifier`)이 포함됩니다. 예를 들어, `ListRepositories` (AWS CLI, `aws codecommit list-repositories`), `CreateRepository` (`aws codecommit create-repository`) 및 `PutRepositoryTriggers` (`aws codecommit put-repository-triggers`)

작업에 대한 호출은 CloudTrail 로그 파일에 항목과 `GitPull` 및에 대한 Git 클라이언트 호출을 생성합니다`GitPush`. 또한 CodeCommit 리포지토리가 CodePipeline의 파이프라인 소스로 구성된 경우, CodeCommit 액세스 권한 작업(예: CodePipeline의 `UploadArchive`)에 대한 호출들을 볼 수 있습니다. CodeCommit은 AWS Key Management Service 를 사용하여 리포지토리를 암호화하고 복호화하므로 CloudTrail 로그 AWS KMS 에서 CodeCommit에서 `Encrypt` 및 로 호출하는 `Decrypt` 작업도 볼 수 있습니다.

모든 로그 항목은 누가 요청을 생성했는가에 대한 정보가 들어 있습니다. 로그 항목의 사용자 신원 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 IAM 사용자 자격 증명으로 했는지 여부
- 요청이 역할 또는 페더레이션 사용자에 대한 임시 보안 인증 정보로 생성되었는지 아니면 수임된 역할에 의해 생성되었는지 여부
- 요청이 다른 AWS 서비스에 의해 이루어졌는지 여부

자세한 설명은 CloudTrail userIdentity 요소를 참조하세요.

원하는 기간만큼 Amazon S3 버킷에 로그 파일을 저장할 수 있습니다. 그러나 Amazon S3 수명 주기 규칙을 정의하여 자동으로 로그 파일을 보관하거나 삭제할 수도 있습니다. 기본적으로 로그 파일은 Amazon S3 서버 측 암호화(SSE)를 통해 암호화합니다.

CodeCommit 로그 파일 항목 이해

CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 각 항목은 여러 개의 JSON 형식 이벤트를 표시합니다. 로그 이벤트는 모든 소스의 단일 요청을 나타내며 요청된 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보를 포함합니다. 로그 항목은 퍼블릭 API 직접 호출의 주문 스택 트레이스가 아니기 때문에 특정 순서로 표시되지 않습니다.

> **Note**
>
> 이 예제는 가독성을 높이도록 형식이 지정되었습니다. CloudTrail 로그 파일에서는 모든 항목 및 이벤트가 한 줄로 연결되어 있습니다. 이 예제는 또한 단일 CodeCommit 항목으로 제한되었습니다. 실제 CloudTrail 로그 파일에는 여러 AWS 서비스의 항목 및 이벤트가 기록됩니다.

목차

- 예: CodeCommit 리포지토리를 나열하는 것에 대한 로그 항목

- 예: CodeCommit 리포지토리를 생성하는 것에 대한 로그 항목
- 예: CodeCommit 리포지토리의 Git 풀 호출에 대한 로그 항목
- 예: CodeCommit 리포지토리로 성공적으로 푸시하는 것에 대한 로그 항목

예: CodeCommit 리포지토리를 나열하는 것에 대한 로그 항목

다음은 `ListRepositories` 작업을 설명하는 CloudTrail 로그 항목을 보여 주는 예제입니다.

> **Note**
>
> `ListRepositories`가 리포지토리 목록을 반환하지만, 변경할 수 없는 응답은 CloudTrail 로그에 기록되지 않습니다. 따라서 `responseElements`는 로그 파일에서 `null`과 같이 표시됩니다.

```
{
  "eventVersion":"1.05",
  "userIdentity": {
    "type":"IAMUser",
    "principalId":"AIDACKCEVSQ6C2EXAMPLE",
    "arn":"arn:aws:iam::444455556666:user/Mary_Major",
    "accountId":"444455556666",
    "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
    "userName":"Mary_Major"
    },
  "eventTime":"2016-12-14T17:57:36Z",
  "eventSource":"codecommit.amazonaws.com",
  "eventName":"ListRepositories",
  "awsRegion":"us-east-1",
  "sourceIPAddress":"203.0.113.12",
  "userAgent":"aws-cli/1.10.53 Python/2.7.9 Windows/8 botocore/1.4.43",
  "requestParameters":null,
  "responseElements":null,
  "requestID":"cb8c167e-EXAMPLE",
  "eventID":"e3c6f4ce-EXAMPLE",
  "readOnly":true,
  "eventType":"AwsApiCall",
  "apiVersion":"2015-04-13",
  "recipientAccountId":"444455556666"
}
```

예: CodeCommit 리포지토리를 생성하는 것에 대한 로그 항목

다음은 미국 동부(오하이오)리전의 `CreateRepository` 작업을 설명하는 CloudTrail 로그 항목을 보여 주는 예제입니다.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::444455556666:user/Mary_Major",
    "accountId": "444455556666",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName":"Mary_Major"
  },
  "eventTime": "2016-12-14T18:19:15Z",
  "eventSource": "codecommit.amazonaws.com",
  "eventName": "CreateRepository",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "203.0.113.12",
  "userAgent": "aws-cli/1.10.53 Python/2.7.9 Windows/8 botocore/1.4.43",
  "requestParameters": {
    "repositoryDescription": "Creating a demonstration repository.",
    "repositoryName": "MyDemoRepo"
  },
  "responseElements": {
    "repositoryMetadata": {
      "arn": "arn:aws:codecommit:us-east-2:111122223333:MyDemoRepo",
      "creationDate": "Dec 14, 2016 6:19:14 PM",
      "repositoryId": "8afe792d-EXAMPLE",
      "cloneUrlSsh": "ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/
MyDemoRepo",
      "repositoryName": "MyDemoRepo",
      "accountId": "111122223333",
      "cloneUrlHttp": "https://git-codecommit.us-east-2.amazonaws.com/v1/repos/
MyDemoRepo",
      "repositoryDescription": "Creating a demonstration repository.",
      "lastModifiedDate": "Dec 14, 2016 6:19:14 PM"
    }
  },
  "requestID": "d148de46-EXAMPLE",
  "eventID": "740f179d-EXAMPLE",
  "readOnly": false,
```

```
  "resources": [
    {
      "ARN": "arn:aws:codecommit:us-east-2:111122223333:MyDemoRepo",
      "accountId": "111122223333",
      "type": "AWS::CodeCommit::Repository"
    }
  ],
  "eventType": "AwsApiCall",
  "apiVersion": "2015-04-13",
  "recipientAccountId": "111122223333"
}
```

예: CodeCommit 리포지토리의 Git 풀 호출에 대한 로그 항목

다음은 로컬 리포지토리가 이미 최신 상태인 `GitPull` 작업을 설명하는 CloudTrail 로그 항목을 보여주는 예제입니다.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::444455556666:user/Mary_Major",
    "accountId": "444455556666",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName":"Mary_Major"
    },
  "eventTime": "2016-12-14T18:19:15Z",
  "eventSource": "codecommit.amazonaws.com",
  "eventName": "GitPull",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "203.0.113.12",
  "userAgent": "git/2.11.0.windows.1",
  "requestParameters": null,
  "responseElements": null,
  "additionalEventData": {
    "protocol": "HTTP",
    "dataTransferred": false,
    "repositoryName": "MyDemoRepo",
    "repositoryId": "8afe792d-EXAMPLE",
    },
  "requestID": "d148de46-EXAMPLE",
  "eventID": "740f179d-EXAMPLE",
```

```
  "readOnly": true,
  "resources": [
    {
      "ARN": "arn:aws:codecommit:us-east-2:111122223333:MyDemoRepo",
      "accountId": "111122223333",
      "type": "AWS::CodeCommit::Repository"
      }
    ],
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

다음은 로컬 리포지토리가 최신 상태가 아니어서 데이터가 CodeCommit 리포지토리에서 로컬 리포지토리로 전송되는 `GitPull` 작업을 설명하는 CloudTrail 로그 항목을 보여 주는 예제입니다.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::444455556666:user/Mary_Major",
    "accountId": "444455556666",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName":"Mary_Major"
    },
  "eventTime": "2016-12-14T18:19:15Z",
  "eventSource": "codecommit.amazonaws.com",
  "eventName": "GitPull",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "203.0.113.12",
  "userAgent": "git/2.10.1",
  "requestParameters": null,
  "responseElements": null,
  "additionalEventData": {
    "protocol": "HTTP",
    "capabilities": [
      "multi_ack_detailed",
      "side-band-64k",
      "thin-pack"
      ],
    "dataTransferred": true,
    "repositoryName": "MyDemoRepo",
    "repositoryId": "8afe792d-EXAMPLE",
```

```
    "shallow": false
    },
  "requestID": "d148de46-EXAMPLE",
  "eventID": "740f179d-EXAMPLE",
  "readOnly": true,
  "resources": [
    {
      "ARN": "arn:aws:codecommit:us-east-2:111122223333:MyDemoRepo",
      "accountId": "111122223333",
      "type": "AWS::CodeCommit::Repository"
      }
    ],
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

예: CodeCommit 리포지토리로 성공적으로 푸시하는 것에 대한 로그 항목

다음은 성공적인 `GitPush` 작업을 설명하는 CloudTrail 로그 항목을 보여 주는 예제입니다. `GitPush` 작업은 성공한 푸시에 대한 로그 항목에 두 번 나타납니다.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::444455556666:user/Mary_Major",
    "accountId": "444455556666",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName":"Mary_Major"
    },
  "eventTime": "2016-12-14T18:19:15Z",
  "eventSource": "codecommit.amazonaws.com",
  "eventName": "GitPush",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "203.0.113.12",
  "userAgent": "git/2.10.1",
  "requestParameters": null,
  "responseElements": null,
  "additionalEventData": {
    "protocol": "HTTP",
    "dataTransferred": false,
    "repositoryName": "MyDemoRepo",
```

```
    "repositoryId": "8afe792d-EXAMPLE",
    },
  "requestID": "d148de46-EXAMPLE",
  "eventID": "740f179d-EXAMPLE",
  "readOnly": false,
  "resources": [
    {
      "ARN": "arn:aws:codecommit:us-east-2:111122223333:MyDemoRepo",
      "accountId": "111122223333",
      "type": "AWS::CodeCommit::Repository"
      }
    ],
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
},
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::444455556666:user/Mary_Major",
    "accountId": "444455556666",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName":"Mary_Major"
    },
  "eventTime": "2016-12-14T18:19:15Z",
  "eventSource": "codecommit.amazonaws.com",
  "eventName": "GitPush",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "203.0.113.12",
  "userAgent": "git/2.10.1",
  "requestParameters": {
    "references": [
      {
        "commit": "100644EXAMPLE",
        "ref": "refs/heads/main"
        }
      ]
    },
  "responseElements": null,
  "additionalEventData": {
    "protocol": "HTTP",
    "capabilities": [
      "report-status",
```

```
      "side-band-64k"
      ],
    "dataTransferred": true,
    "repositoryName": "MyDemoRepo",
    "repositoryId": "8afe792d-EXAMPLE",
    },
  "requestID": "d148de46-EXAMPLE",
  "eventID": "740f179d-EXAMPLE",
  "readOnly": false,
  "resources": [
    {
      "ARN": "arn:aws:codecommit:us-east-2:111122223333:MyDemoRepo",
      "accountId": "111122223333",
      "type": "AWS::CodeCommit::Repository"
      }
    ],
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

를 사용하여 CodeCommit 리소스 생성 AWS CloudFormation

AWS CodeCommit 는 AWS 리소스 및 인프라를 생성하고 관리하는 데 소요되는 시간을 줄일 수 있도록 리소스를 모델링하고 설정하는 데 도움이 되는 AWS CloudFormation서비스인와 통합됩니다. 원하는 모든 AWS 리소스(예: 리포지토리)를 설명하는 템플릿을 생성하면가 해당 리소스를 AWS CloudFormation 프로비저닝하고 구성합니다.

를 사용하면 템플릿을 재사용하여 CodeCommit 리소스를 일관되고 반복적으로 설정할 AWS CloudFormation수 있습니다. 리소스를 한 번 설명한 다음 여러 AWS 계정 및 리전에서 동일한 리소스를 반복적으로 프로비저닝합니다.

CodeCommit 및 AWS CloudFormation 템플릿

CodeCommit 및 관련 서비스에 대한 리소스를 프로비저닝하고 구성하려면 AWS CloudFormation 템플릿을 이해해야 합니다. 템플릿은 JSON 또는 YAML로 서식 지정된 텍스트 파일입니다. 이러한 템플릿은 AWS CloudFormation 스택에서 프로비저닝하려는 리소스를 설명합니다. JSON 또는 YAML에 익숙하지 않은 경우 AWS CloudFormation Designer를 사용하여 AWS CloudFormation 템플릿을 시작할 수 있습니다. 자세한 내용은 AWS CloudFormation 사용 설명서에서 AWS CloudFormation Designer이란 무엇입니까?를 참조하세요.

CodeCommit은 콘솔 또는 명령줄에서 리포지토리를 생성하는 AWS CloudFormation 것과 달리를 사용하여 리포지토리 AWS CloudFormation 를 생성하고 Amazon S3 버킷의 지정된 .zip 파일에서 새로 생성된 리포지토리에 코드를 자동으로 커밋할 수 있습니다. 리포지토리를 위한 JSON 및 YAML 템플릿에 관한 예시를 비롯해 자세한 내용을 살펴보려면 AWS: :CodeCommit::리포지토리를 참조하세요.

를 사용하여 CodeCommit 리포지토리를 생성할 때 AWS:CodeCommit::Repository Code에서 속성을 구성하여 아카이브가 20MB 미만인 한 생성 프로세스의 일부로 해당 리포지토리에 코드를 커밋할 AWS CloudFormation수 있습니다. 사용자는 코드가 저장되는 Amazon S3 버킷을 지정할 수 있으며, 선택적으로 BranchName 속성을 사용하여 해당 코드의 초기 커밋 시에 생성될 기본 브랜치의 이름을 지정할 수 있습니다. 이러한 속성은 초기 리포지토리 생성 시에만 사용되며 스택 업데이트 시에는 무시됩니다. 초기 커밋을 만든 후에는 이러한 속성을 사용하여 리포지토리에 추가 커밋을 만들거나 기본 브랜치의 이름을 변경할 수 없습니다.

> **Note**
>
> 2021년 1월 19일에 CodeCommit의 기본 브랜치 이름을 마스터에서 기본으로 AWS 변경했습니다. 이 이름 변경은 CodeCommit 콘솔, CodeCommit APIs, AWS SDKs 및를 사용하여 리포

지토리에 대한 초기 커밋을 생성할 때 CodeCommit의 기본 동작에 영향을 줍니다 AWS CLI. 생성 과정에서 코드의 초기 커밋 AWS CDK 을 사용하여 AWS CloudFormation 또는 로 생성된 리포지토리는 2021년 3월 4일부터이 변경 사항과 일치합니다. 이 변경 사항은 기존 리포지토리나 브랜치에 영향을 주지 않습니다. 로컬 Git 클라이언트를 사용하여 초기 커밋을 생성하는 고객은 해당 Git 클라이언트의 구성을 따르는 기본 브랜치 이름을 갖게 됩니다. 자세한 내용은 브렌치로 작업하기, 커밋 생성, 브랜치 설정 변경을 참조하세요.

사용자는 리포지토리, AWS CodeBuild 빌드 프로젝트, AWS CodeDeploy 애플리케이션, AWS CodePipeline 파이프라인 등에 관한 알림 규칙과 같은 관련 리소스를 생성하는 템플릿을 생성할 수도 있습니다.

템플릿 예제

다음 예제에서는 *MyDemoRepo*라는 CodeCommit 리포지토리를 생성합니다. 새로 생성된 리포지토리는 *MySourceCodeBucket*이라는 Amazon S3 버킷에 저장된 코드로 채워지고 해당 리포지토리의 기본 브랜치인 *development*라는 브랜치에 배치됩니다.

> **Note**
> 새 리포지토리에 커밋될 콘텐츠의 ZIP 파일을 포함하는 Amazon S3 버킷의 이름은 Amazon Web Services 계정에서 ARN이나 버킷의 이름을 사용해 지정할 수 있습니다. Amazon S3 객체 키는 Amazon S3 개발자 안내서에 정의되어 있습니다.

JSON:

```
{
    "MyRepo": {
        "Type": "AWS::CodeCommit::Repository",
        "Properties": {
            "RepositoryName": "MyDemoRepo",
            "RepositoryDescription": "This is a repository for my project with code
 from MySourceCodeBucket.",
            "Code": {
                "BranchName": "development",
                "S3": {
                    "Bucket": "MySourceCodeBucket",
                    "Key": "MyKey",
```

```
                    "ObjectVersion": "1"
                }
            }
        }
    }
}
```

YAML:

```
MyRepo:
  Type: AWS::CodeCommit::Repository
  Properties:
    RepositoryName: MyDemoRepo
    RepositoryDescription: This is a repository for my project with code from
 MySourceCodeBucket.
    Code:
      BranchName: development
      S3:
        Bucket: MySourceCodeBucket,
        Key: MyKey,
        ObjectVersion: 1
```

더 많은 예제를 보려면 AWS::CodeCommit::리포지토리를 참조하세요.

AWS CloudFormation, CodeCommit 및 AWS 클라우드 개발 키트 (AWS CDK)

생성 시 AWS CDK 사용 AWS CloudFormation 기능을 사용하여 생성된 리포지토리. AWS CloudFormation 템플릿이 CodeCommit 리소스와 작동하는 방식을 이해하면 AWS CDK 코드를 생성하고 관리하는 데 도움이 될 수 있습니다. 에 대한 자세한 내용은 AWS CDKAWS 클라우드 개발 키트 (AWS CDK) 개발자 안내서 및 AWS CDK API 참조를 참조하세요.

다음 AWS CDK Typescript 예제에서는 *MyDemoRepo*라는 CodeCommit 리포지토리를 생성합니다. 새로 생성된 리포지토리는 *MySourceCodeBucket*이라는 Amazon S3 버킷에 저장된 코드로 채워지고 해당 리포지토리의 기본 브랜치인 *development*라는 브랜치에 배치됩니다.

```
import * as cdk from '@aws-cdk/core';
import codecommit = require('@aws-cdk/aws-codecommit');
export class CdkCodecommitStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
```

```
    super(scope, id, props);
    // The code creates a CodeCommit repository with a default branch name development
      new codecommit.CfnRepository(this, 'MyRepoResource', {
            repositoryName: "MyDemoRepo",
            code: {
              "branchName": "development",
              "s3": {
                "bucket": "MySourceCodeBucket",
                "key": "MyKey"
              }
            },
        }
     );
  }
}
```

에 대해 자세히 알아보기 AWS CloudFormation

에 대해 자세히 알아보려면 다음 리소스를 AWS CloudFormation참조하세요.

- AWS CloudFormation
- AWS CloudFormation 사용 설명서
- AWS CloudFormation 명령줄 인터페이스 사용 설명서

문제 해결 AWS CodeCommit

다음은 AWS CodeCommit에서 일반적으로 발생하는 문제를 해결하는 데 유용한 정보입니다.

주제

- 에 대한 Git 자격 증명 및 HTTPS 연결 문제 해결 AWS CodeCommit
- git-remote-codecommit 및 문제 해결 AWS CodeCommit
- AWS CodeCommit과의 SSH 연결 문제 해결
- AWS CodeCommit에 대한 보안 인증 도우미 및 HTTPS 연결 문제 해결
- Git 클라이언트 및 문제 해결 AWS CodeCommit
- 액세스 오류 및 문제 해결 AWS CodeCommit
- 구성 오류 및 문제 해결 AWS CodeCommit
- 콘솔 오류 및 문제 해결 AWS CodeCommit
- 트리거 및 문제 해결 AWS CodeCommit
- 디버깅 활성화

에 대한 Git 자격 증명 및 HTTPS 연결 문제 해결 AWS CodeCommit

다음 정보는 Git 자격 증명 및 HTTPS를 사용하여 AWS CodeCommit 리포지토리에 연결할 때 발생하는 일반적인 문제를 해결하는 데 도움이 될 수 있습니다.

주제

- 에 대한 Git 자격 증명 AWS CodeCommit: 터미널 또는 명령줄에서 CodeCommit 리포지토리에 연결할 때 자격 증명 프롬프트가 계속 표시됨
- 에 대한 Git 자격 증명 AWS CodeCommit: Git 자격 증명을 설정했지만 시스템이 이를 사용하고 있지 않습니다.

에 대한 Git 자격 증명 AWS CodeCommit: 터미널 또는 명령줄에서 CodeCommit 리포지토리에 연결할 때 자격 증명 프롬프트가 계속 표시됨

문제: 터미널이나 명령줄에서 CodeCommit 리포지토리에 대해 푸시하거나 풀하거나 상호 작용하려 할 때, 사용자 이름과 암호를 입력하라는 메시지가 표시되며 IAM 사용자의 Git 보안 인증 정보를 제공해야 합니다.

가능한 해결 방법: 이 오류의 가장 일반적인 원인은 로컬 컴퓨터에 보안 인증 정보 관리를 지원하지 않는 운영 체제가 실행되고 있거나, 보안 인증 정보 관리 유틸리티가 설치되어 있지 않거나, IAM 사용자의 Git 보안 인증 정보가 이러한 보안 인증 정보 관리 시스템에 저장되어 있지 않기 때문입니다. 운영 체제 및 로컬 환경에 따라 사용자는 보안 인증 정보 관리자를 설치하거나, 운영 체제에 포함된 보안 인증 정보 관리자를 구성하거나, 보안 인증 정보 스토리지를 사용하도록 로컬 환경을 사용자 지정해야 할 수 있습니다. 예를 들어, 컴퓨터에서 macOS를 실행하는 경우에는 Keychain Access 유틸리티를 사용하여 보안 인증 정보를 저장할 수 있습니다. 컴퓨터에서 Windows를 실행하는 경우에는 Windows용 Git과 함께 설치된 Git 보안 인증 정보 관리자를 사용할 수 있습니다. 자세한 내용을 알아보려면 Git 설명서에서 Git 보안 인증 정보를 사용하는 HTTPS 사용자의 경우 섹션 및 보안 인증 정보 스토리지를 참조하세요.

에 대한 Git 자격 증명 AWS CodeCommit: Git 자격 증명을 설정했지만 시스템이 이를 사용하고 있지 않습니다.

문제: Git 클라이언트로 CodeCommit을 사용하려 하면 클라이언트가 IAM 사용자의 Git 보안 인증 정보를 사용하지 않는 것 같습니다.

가능한 해결 방법: 이 오류의 가장 일반적인 원인은 컴퓨터가 AWS CLI에 포함된 보안 인증 도우미를 사용하도록 설정되어 있기 때문입니다. .gitconfig 파일에 다음과 유사한 구성 섹션이 있는지 확인하고 제거합니다.

```
[credential "https://git-codecommit.*.amazonaws.com"]
    helper = !aws codecommit credential-helper $@
    UseHttpPath = true
```

파일을 저장한 다음 새 명령줄 또는 터미널 세션을 연 후 연결을 다시 시도합니다.

또한 컴퓨터에 보안 인증 도우미 또는 관리자가 여러 개 설치되어 있을 수 있으며, 시스템이 다른 구성을 기본값으로 사용하고 있을 수도 있습니다. 보안 인증 도우미가 기본값으로 사용되도록 재설정하려면 git config 명령을 실행할 때 --global 또는 --local 대신 --system 옵션을 사용할 수 있습니다.

자세한 내용을 알아보려면 Git 설명서에서 Git 보안 인증 정보를 사용하는 HTTPS 사용자의 경우 섹션 및 보안 인증 정보 스토리지를 참조하세요.

git-remote-codecommit 및 문제 해결 AWS CodeCommit

다음 정보는 AWS CodeCommit 리포지토리 연결 시 발생하는 git-remote-codecommit 문제를 해결하는 데 도움이 될 수 있습니다.

주제

- 오류가 표시됩니다: git: 'remote-codecommit'은 git 명령어가 아닙니다
- 오류가 표시됩니다: fatal: 'codecommit'의 원격 도우미를 찾을 수 없습니다.
- 복제 오류: IDE에서 CodeCommit 리포지토리를 복제할 수 없습니다
- 푸시 또는 풀 오류: IDE에서 CodeCommit 리포지토리로 커밋을 푸시하거나 풀할 수 없습니다
- 설치 오류: git-remote-codecommitexternally-managed-environment 오류가 표시됩니다.

오류가 표시됩니다: git: 'remote-codecommit'은 git 명령어가 아닙니다

문제: git-remote-codecommit을 사용하려고 하면, git-remote-codecommit은 git 명령어가 아니라는 오류가 나타납니다. ‘git --help’”를 참조하세요.

가능한 해결 방법: 이 오류가 발생하는 가장 일반적인 이유는 git-remote-codecommit 실행 파일을 PATH에 추가하지 않았거나 문자열에 구문 오류가 포함되어 있기 때문입니다. 이는 git과 remote-codecommit 사이에 하이픈이 없거나 git-remote-codecommit 앞에 git이 추가로 배치되는 경우 발생할 수 있습니다.

로컬 설치를 지원되지 않는 버전으로 업데이트한 경우에도이 오류가 발생할 수 있습니다. 지원되는 Python 버전에 대한 자세한 내용은 git-remote-codecommit을 참조하세요.

git-remote-codecommit 설정 및 사용에 대한 자세한 내용은 git-remote-codecommit을 AWS CodeCommit 사용하여에 HTTPS 연결을 위한 설정 단계 단원을 참조하세요.

오류가 표시됩니다: fatal: 'codecommit'의 원격 도우미를 찾을 수 없습니다.

문제: git-remote-codecommit을 사용하려고 하면 “fatal: ‘codecommit’의 원격 도우미를 찾을 수 없습니다”라는 오류 메시지가 표시됩니다.

가능한 해결 방법: 이 오류 메시지가 나타나는 가장 일반적인 이유는 다음과 같습니다.

- git-remote-codecommit에 대한 설정이 완료되지 않았습니다
- 경로에 없거나 `Path` 환경 변수의 일부로 구성되지 않은 위치에 git-remote-codecommit을 설치했습니다
- Python이 경로에 없거나 `Path` 환경 변수의 일부로 구성되지 않았습니다
- git-remote-codecommit 설치가 완료된 이후 다시 시작된 것이 아닌 터미널 또는 명령줄 창을 사용하고 있습니다

git-remote-codecommit 설정 및 사용에 대한 자세한 내용은 git-remote-codecommit을 AWS CodeCommit 사용하여에 HTTPS 연결을 위한 설정 단계 단원을 참조하세요.

복제 오류: IDE에서 CodeCommit 리포지토리를 복제할 수 없습니다

문제: IDE에서 CodeCommit 리포지토리를 복제하려고 하면 엔드포인트 또는 URL이 유효하지 않다는 오류가 표시됩니다.

가능한 해결 방법: 모든 IDE가 복제 중에 git-remote-codecommit에서 사용하는 URL을 지원하는 것은 아닙니다. 터미널이나 명령줄에서 로컬로 리포지토리를 복제한 다음 해당 로컬 리포지토리를 IDE에 추가합니다. 자세한 내용은 3단계: CodeCommit 콘솔 연결 및 리포지토리 복제 단원을 참조하십시오.

푸시 또는 풀 오류: IDE에서 CodeCommit 리포지토리로 커밋을 푸시하거나 풀할 수 없습니다

문제: IDE에서 코드를 푸시하거나 풀하려고 시도할 때 연결 오류가 발생합니다.

가능한 해결 방법: 이 오류의 가장 일반적인 이유는 IDE가 git-remote-codecommit와 같은 Git 원격 도우미와 호환되지 않는다는 것입니다. IDE 기능을 사용하여 코드를 커밋, 푸시 및 풀하는 대신 Git 명령을 사용하여 명령줄이나 터미널에서 로컬 리포지토리를 수동으로 업데이트하십시오.

원격 도우미 및 Git에 대한 자세한 내용은 Git 설명서를 참조하십시오.

설치 오류: git-remote-codecommitexternally-managed-environment 오류가 표시됩니다.

문제: `pip install git-remote-codecommit` 명령을 실행하려고 하면 환경이 외부에서 관리됨을 나타내는 오류가 표시됩니다.

가능한 해결 방법:이 오류의 가장 일반적인 이유는 EXTERNALLY-MANAGED 마커 파일을 정의하는 Python의 배포(디스트로)를 실행하기 때문입니다. 이에 대한 가장 좋은 해결책은 가상 환경을 생성하고 사용하는 것입니다.

외부 관리형 환경 및 Python에 대한 자세한 내용은 Python 설명서의 외부 관리형 환경 및 pip 및 venv를 사용하여 가상 환경에 패키지 설치를 참조하세요.

AWS CodeCommit과의 SSH 연결 문제 해결

다음 정보는 SSH를 사용하여 CodeCommit 리포지토리에 연결 시 일반적인 문제를 해결하는 데 도움이 됩니다.

주제

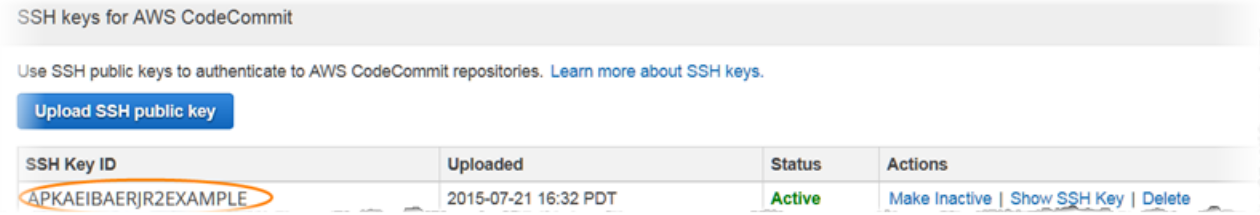
- 액세스 오류: 퍼블릭 키가 IAM에 성공적으로 업로드되었지만 Linux, macOS, Unix 시스템에서 연결이 실패했습니다
- 액세스 오류: 퍼블릭 키가 IAM에 성공적으로 업로드되고 SSH가 성공적으로 테스트되었지만 Windows 시스템에서 연결이 실패했습니다
- 인증 문제: CodeCommit 리포지토리에 연결할 때 호스트의 인증을 설정할 수 없습니다
- IAM 오류: 퍼블릭 키를 IAM에 추가하려고 시도할 때 '잘못된 형식' 오류가 나타납니다
- SSH 보안 인증 정보를 사용하여 여러 Amazon Web Services 계정의 CodeCommit 리포지토리에 액세스해야 합니다
- Windows에서의 Git: SSH를 사용하여 연결을 시도할 때 Bash 에뮬레이터 또는 명령줄이 중지됩니다
- 일부 Linux 배포판에 공개 키 형식을 지정해야 합니다
- 액세스 오류: CodeCommit 리포지토리에 연결할 때 SSH 퍼블릭 키가 거부됩니다

액세스 오류: 퍼블릭 키가 IAM에 성공적으로 업로드되었지만 Linux, macOS, Unix 시스템에서 연결이 실패했습니다

문제: SSH 엔드포인트에 연결하여 CodeCommit 리포지토리와 통신을 시도할 때, 그리고 연결을 테스트하거나 리포지토리를 복제할 때 연결이 실패하거나 거부됩니다.

가능한 해결 방법:: IAM의 퍼블릭 키에 할당된 SSH 키 ID가 연결 시도에 연결되지 않았을 수 있습니다. config 파일을 구성하지 않았거나, config 파일에 대한 액세스 권한이 없거나 다른 설정이 config 파일을 성공적으로 읽지 못하도록 하고 있거나, 잘못된 키 ID를 입력했거나, 키 ID 대신에 IAM 사용자의 ID를 제공했을 수 있습니다.

SSH 키 ID는 IAM 사용자 프로필의 IAM 콘솔에서 찾을 수 있습니다.



> **Note**
>
> SSH 키 ID를 두 개 이상 업로드한 경우, 키가 업로드 날짜가 아니라 키 ID를 기준으로 영문자순으로 나열됩니다. 올바른 업로드 날짜와 연결된 키 ID를 복사했는지 확인합니다.

다음 명령으로 연결 테스트를 시도합니다.

```
ssh Your-SSH-Key-ID@git-codecommit.us-east-2.amazonaws.com
```

연결 확인 후 성공 메시지가 나타나면 SSH 키 ID가 유효한 것입니다. config 파일을 편집하여 IAM의 퍼블릭 키로 연결 시도에 연결합니다. config 파일을 편집하지 않으려면 SSH 키 ID를 사용하여 리포지토리에 모든 연결 시도를 시작할 수 있습니다. 예를 들어, config 파일을 수정하지 않고 *MyDemoRepo* 리포지토리를 복제하여 연결 시도에 연결하려면 다음 명령을 실행합니다.

```
git clone ssh://Your-SSH-Key-ID@git-codecommit.us-east-2.amazonaws.com/v1/
repos/MyDemoRepo my-demo-repo
```

자세한 내용은 Linux, macOS, Unix에서 SSH 연결 단원을 참조하십시오.

액세스 오류: 퍼블릭 키가 IAM에 성공적으로 업로드되고 SSH가 성공적으로 테스트되었지만 Windows 시스템에서 연결이 실패했습니다

문제: SSH 엔드포인트를 사용하여 CodeCommit 리포지토리에 대해 복제나 통신을 시도할 때 `No supported authentication methods available` 문구가 포함된 오류 메시지가 나타납니다.

수정 방법: 이 오류가 발생하는 가장 일반적인 이유는 SSH 사용을 시도할 때 Windows에게 다른 프로그램을 사용하도록 지시하는 Windows 시스템 환경 설정 변수 집합이 있기 때문입니다. 예를 들어, GIT_SSH 변수가 도구(plink.exe)의 PuTTY 설정 중 하나를 가리키도록 설정했을 수 있습니다. 이는 레거시 구성이거나 컴퓨터에 설치된 하나 이상의 기타 프로그램에 필요한 것일 수 있습니다. 이 환경 변수가 필요하지 않다고 확신하면 시스템 속성을 열어 제거할 수 있습니다.

이 문제를 해결하려면 Bash 에뮬레이터를 연 다음 SSH 연결을 다시 시도하세요. 이때 `GIT_SSH_COMMAND="SSH"`를 접두사로 포함합니다. 예를 들어, SSH를 사용하는 리포지토리를 복제하려면 다음 명령을 실행합니다.

```
GIT_SSH_COMMAND="ssh" git clone ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/
MyDemoRepo my-demo-repo
```

Windows 명령줄에서 SSH를 통해 연결할 때 Windows 버전이 연결 문자열의 일부로 SSH 키 ID를 포함하도록 요구하는 경우 유사한 문제가 발생할 수 있습니다. 이번에는 IAM에서 복사된 SSH 키 ID를 명령의 일부로 포함하여 연결을 다시 시도하세요. 예시:

```
git clone ssh://Your-SSH-Key-ID@git-codecommit.us-east-2.amazonaws.com/v1/repos/
MyDemoRepo my-demo-repo
```

인증 문제: CodeCommit 리포지토리에 연결할 때 호스트의 인증을 설정할 수 없습니다

문제: SSH 엔드포인트를 사용하여 CodeCommit 리포지토리와 통신을 시도할 때 `The authenticity of host 'host-name' can't be established.` 문구가 포함된 경고 메시지가 나타납니다.

수정 방법: 자격 증명이 올바르게 설정되지 않았을 수 있습니다. Linux, macOS, Unix에서 SSH 연결 또는 Windows에서 SSH 연결의 지침을 따릅니다.

이 단계를 따랐는데도 문제가 지속되는 경우 누군가 중간자 공격을 시도하고 있는 것일 수 있습니다. 다음 메시지가 나타나면 no를 입력하고 Enter 키를 누릅니다.

```
Are you sure you want to continue connecting (yes/no)?
```

CodeCommit 연결을 진행하기 전에 연결을 위한 지문과 퍼블릭 키가 SSH 설정 항목에 기록된 것과 일치하는지 확인합니다.

CodeCommit의 퍼블릭 지문

Server	암호화 해시 유형	지문
git-codecommit.us-east-2.amazonaws.com	MD5	`a9:6d:03:ed:08:42:21:be:06:e1:e0:2a:d1:75:31:5e`
git-codecommit.us-east-2.amazonaws.com	SHA256	`3lBlW2g5xn/NA2Ck6dyeJIrQOWvn7n8UEs56fG6ZIzQ`
git-codecommit.us-east-1.amazonaws.com	MD5	`a6:9c:7d:bc:35:f5:d4:5f:8b:ba:6f:c8:bc:d4:83:84`
git-codecommit.us-east-1.amazonaws.com	SHA256	`eLMY1j0DKA4uvDZcl/KgtIayZANwX6t8+8isPtotBoY`

Server	암호화 해시 유형	지문
git-codecommit.us-west-2.amazonaws.com	MD5	a8:68:53:e3:99:ac:6e:d7:04:7e:f7:92:95:77:a9:77
git-codecommit.us-west-2.amazonaws.com	SHA256	0pJx9SQpkbPUAHwy58UVIq0IHcyo1fwCpOOuVgcAWPo
git-codecommit.eu-west-1.amazonaws.com	MD5	93:42:36:ea:22:1f:f1:0f:20:02:4a:79:ff:ea:12:1d
git-codecommit.eu-west-1.amazonaws.com	SHA256	tKjRkOL8dmJyTmSbeSdN1S8F/f0iql3RlvqgTOP1UyQ
git-codecommit.ap-northeast-1.amazonaws.com	MD5	8e:a3:f0:80:98:48:1c:5c:6f:59:db:a7:8f:6e:c6:cb
git-codecommit.ap-northeast-1.amazonaws.com	SHA256	Xk/WeYD/K/bnBybzhiuu4dWpBJtXPf7E30jHU7se4Ow
git-codecommit.ap-southeast-1.amazonaws.com	MD5	65:e5:27:c3:09:68:0d:8e:b7:6d:94:25:80:3e:93:cf
git-codecommit.ap-southeast-1.amazonaws.com	SHA256	ZIsVa7OVzxrTIf+Rk4UbhPv6Es22mSB3uTBojfPXIno
git-codecommit.ap-southeast-2.amazonaws.com	MD5	7b:d2:c1:24:e6:91:a5:7b:fa:c1:0c:35:95:87:da:a0

Server	암호화 해시 유형	지문
git-codecommit.ap-southeast-2.amazonaws.com	SHA256	nYp+gHas80HY3DqbP4yanCDFhqDVjseefVbHEXqH2Ec
git-codecommit.ap-southeast-3.amazonaws.com	MD5	64:d9:e0:53:19:4f:a8:91:9a:c3:53:22:a6:a8:ed:a6
git-codecommit.ap-southeast-3.amazonaws.com	SHA256	ATdkGSFhpqIu7RqUVT/1RZo6MLxxxUW9NoDVMbAc/6g
git-codecommit.me-central-1.amazonaws.com	MD5	bd:fa:e2:f9:05:84:d6:39:6f:bc:d6:8d:fe:de:61:76
git-codecommit.me-central-1.amazonaws.com	SHA256	grceUDWubo4MzG1NoaKZKUfrgPvfN3ijliOnQr1lTZA
git-codecommit.eu-central-1.amazonaws.com	MD5	74:5a:e8:02:fc:b2:9c:06:10:b4:78:84:65:94:22:2d
git-codecommit.eu-central-1.amazonaws.com	SHA256	MwGrkiEki8QkkBtlAgXbYt0hoZYBnZF62VY5RzGJEUY
git-codecommit.ap-northeast-2.amazonaws.com	MD5	9f:68:48:9b:5f:fc:96:69:39:45:58:87:95:b3:69:ed
git-codecommit.ap-northeast-2.amazonaws.com	SHA256	eegAPQrWY9YsYo9ZHIKOmxetfXBHzAZd8Eya53Qcwko

Server	암호화 해시 유형	지문
git-codecommit.sa-east-1.amazonaws.com	MD5	`74:99:9d:ff:2b:ef:63:c6:4b:b4:6a:7f:62:c5:4b:51`
git-codecommit.sa-east-1.amazonaws.com	SHA256	`kW+VKB0jpRaG/ZbXkgbtMQbKgEDK7JnISV3SVoyCmzU`
git-codecommit.us-west-1.amazonaws.com	MD5	`3b:76:18:83:13:2c:f8:eb:e9:a3:d0:51:10:32:e7:d1`
git-codecommit.us-west-1.amazonaws.com	SHA256	`gzauWTWXDK2u5KuMMi5vbKTmfyerdIwgSbzYBODLpzg`
git-codecommit.eu-west-2.amazonaws.com	MD5	`a5:65:a6:b1:84:02:b1:95:43:f9:0e:de:dd:ed:61:d3`
git-codecommit.eu-west-2.amazonaws.com	SHA256	`r0Rwz5k/IHp/QyrRnfiM9j02D5UEqMbtFNTuDG2hNbs`
git-codecommit.ap-south-1.amazonaws.com	MD5	`da:41:1e:07:3b:9e:76:a0:c5:1e:64:88:03:69:86:21`
git-codecommit.ap-south-1.amazonaws.com	SHA256	`hUKwnTj7+Xpx4Kddb6p45j4RazIJ4IhAMD8k29itOfE`
git-codecommit.ap-south-2.amazonaws.com	MD5	`bc:cc:9f:15:f8:f3:58:a2:68:65:21:e2:23:71:8d:ce`

Server	암호화 해시 유형	지문
git-codecommit.ap-south-2.amazonaws.com	SHA256	`Xe0CyZEOvgR5Xa2YUGqf+jn8/Ut7l7nX/CmslSFNEig`
git-codecommit.ca-central-1.amazonaws.com	MD5	`9f:7c:a2:2f:8c:b5:74:fd:ab:b7:e1:fd:af:46:ed:23`
git-codecommit.ca-central-1.amazonaws.com	SHA256	`Qz5puafQdANVprLlj6r0Qyh4lCNsF6ob61dGcPtFS7w`
git-codecommit.eu-west-3.amazonaws.com	MD5	`1b:7f:97:dd:d7:76:8a:32:2c:bd:2c:7b:33:74:6a:76`
git-codecommit.eu-west-3.amazonaws.com	SHA256	`uw7c2FL564jVoFgtc+ikzILnKBsZz7t9+CFdSJjKbLI`
git-codecommit.us-gov-west-1.amazonaws.com	MD5	`9f:6c:19:3b:88:cd:e8:88:1b:9c:98:6a:95:31:8a:69`
git-codecommit.us-gov-west-1.amazonaws.com	SHA256	`djXQoSIFcg8vHe0KVH1xW/gOF9X37tWTqu4Hkng75x4`
git-codecommit.us-gov-east-1.amazonaws.com	MD5	`00:8d:b5:55:6f:05:78:05:ed:ea:cb:3f:e6:f0:62:f2`
git-codecommit.us-gov-east-1.amazonaws.com	SHA256	`fVb+R0z7qW7minenW+rUpAABRCRBTCzmETAJEQrg98`

Server	암호화 해시 유형	지문
git-codecommit.eu-north-1.amazonaws.com	MD5	8e:53:d8:59:35:88:82:fd:73:4b:60:8a:50:70:38:f4
git-codecommit.eu-north-1.amazonaws.com	SHA256	b6KSK7xKq+V8jl7iuAcjqXsG7zkqoUZZmmhYYFBq1wQ
git-codecommit.me-south-1.amazonaws.com	MD5	0e:39:28:56:d5:41:e6:8d:fa:81:45:37:fb:f3:cd:f7
git-codecommit.me-south-1.amazonaws.com	SHA256	O+NToCGgjrHekiBuOl0ad7ROGEsz+DBLXOd/c9wc0JU
git-codecommit.ap-east-1.amazonaws.com	MD5	a8:00:3d:24:52:9d:61:0e:f6:e3:88:c8:96:01:1c:fe
git-codecommit.ap-east-1.amazonaws.com	SHA256	LafadYwUYW8hONoTRpojbjNs9IRnbEwHtezD3aAIBX0
git-codecommit.cn-north-1.amazonaws.com .cn	MD5	11:7e:2d:74:9e:3b:94:a2:69:14:75:6f:5e:22:3b:b3
git-codecommit.cn-north-1.amazonaws.com .cn	SHA256	IYUXxH2OpTDsyYMLIp+JY8CTLS4UX+ZC5JVZXPRaxc8
git-codecommit.cn-northwest-1.amazonaws.com .cn	MD5	2e:a7:fb:4c:33:ac:6c:f9:aa:f2:bc:fb:0a:7b:1e:b6

Server	암호화 해시 유형	지문
git-codecommit.cn-northwest-1.amazonaws.com .cn	SHA256	wqjd6eHd0+mOBx+dCNuL0omUoCNjaDtZiEpWj5TmCfQ
git-codecommit.eu-south-1.amazonaws.com	MD5	b9:f6:5d:e2:48:92:3f:a9:37:1e:c4:d0:32:0e:fb:11
git-codecommit.eu-south-1.amazonaws.com	SHA256	lyXrWbCg3uQmJrl1XxB/ASR7ugW1Ysf5yzYOJbudHsI
git-codecommit.ap-northeast-3.amazonaws.com	MD5	25:17:40:da:b9:d4:18:c3:b6:b3:fb:ed:1c:20:fe:29
git-codecommit.ap-northeast-3.amazonaws.com	SHA256	2B815B9F0AvwLnRxSVxUz4kDYmtEQUGGdQYP8OQLXhA
git-codecommit.af-south-1.amazonaws.com	MD5	21:a0:ba:d7:c1:d1:b5:39:98:8d:4d:7c:96:f5:ca:29
git-codecommit.af-south-1.amazonaws.com	SHA256	C34ji3x/cnsDZjUpyNGXdE5pjHYimqJrQZ3leTgqJHM
git-codecommit.il-central-1.amazonaws.com	MD5	04:74:89:16:98:7a:61:b1:69:46:42:3c:d1:b4:ac:a9
git-codecommit.il-central-1.amazonaws.com	SHA256	uFxhp51kUWhleTLeYbxQVYm4RnNLNZ5Dbdm1cgdSl/8

IAM 오류: 퍼블릭 키를 IAM에 추가하려고 시도할 때 '잘못된 형식' 오류가 나타납니다

문제: IAM에서 CodeCommit과 함께 SSH를 사용하도록 설정할 때 퍼블릭 키를 추가하려고 시도하면 `Invalid format` 문구가 포함된 오류 메시지가 나타납니다.

가능한 해결 방법: IAM은 퍼블릭 키가 ssh-rsa 형식 또는 PEM 형식으로 인코딩되어야 합니다. IAM은 OpenSSH 형식의 공개 키만 허용하며, IAM 사용 설명서의 CodeCommit과 함께 SSH 키 사용에 명시된 것과 같은 추가 요구 사항들이 있습니다. 다른 형식으로 퍼블릭 키를 제공하거나 키에 필요한 비트 수가 없는 경우 이 오류가 나타납니다.

- SSH 퍼블릭 키를 복사할 때 운영 체제에서 줄 바꿈이 발생했을 수도 있습니다. IAM에 추가한 퍼블릭 키에 줄 바꿈이 없는지 확인합니다.
- 일부 Windows 운영 체제에서는 OpenSSH 형식을 사용하지 않습니다. 키 페어를 생성하고 IAM에서 요구하는 OpenSSH 형식을 복사하려면 the section called “3단계: Git과 CodeCommit에 사용되는 퍼블릭 키와 프라이빗 키 설정” 섹션을 참조하세요.

IAM의 SSH 키에 필요한 요구 사항에 대해 자세히 알아보려면 IAM 사용 설명서에서 CodeCommit에 SSH 키 사용을 참조하세요.

SSH 보안 인증 정보를 사용하여 여러 Amazon Web Services 계정의 CodeCommit 리포지토리에 액세스해야 합니다

문제: 둘 이상의 Amazon Web Services 계정에서 CodeCommit 리포지토리에 대한 SSH 액세스를 설정하고 싶습니다.

가능한 해결 방법: Amazon Web Services 계정별로 고유한 SSH 공개/개인 키 페어를 생성하고 각 키로 IAM을 구성할 수 있습니다. 그런 다음 퍼블릭 키와 연결된 각 IAM 사용자 ID에 대한 정보로 ~/.ssh/config 파일을 구성할 수 있습니다. 예시:

```
Host codecommit-1
    Hostname git-codecommit.us-east-1.amazonaws.com
    User SSH-KEY-ID-1 # This is the SSH Key ID you copied from IAM in Amazon Web
 Services account 1 (for example, APKAEIBAERJR2EXAMPLE1).
    IdentityFile ~/.ssh/codecommit_rsa # This is the path to the associated public key
 file, such as id_rsa.  We advise creating CodeCommit specific _rsa files.

Host codecommit-2
```

```
    Hostname git-codecommit.us-east-1.amazonaws.com
    User SSH-KEY-ID-2 # This is the SSH Key ID you copied from IAM in Amazon Web
 Services account 2 (for example, APKAEIBAERJR2EXAMPLE2).
    IdentityFile ~/.ssh/codecommit_2_rsa # This is the path to the other associated
 public key file.  We advise creating CodeCommit specific _rsa files.
```

이 구성에서는 'git-codecommit.us-east-1.amazonaws.com'을 'codecommit-2'로 바꿀 수 있습니다. 예를 들어, 두 번째 Amazon Web Services 계정의 리포지토리를 복제하려면 다음과 같이 합니다.

```
git clone ssh://codecommit-2/v1/repos/YourRepositoryName
```

리포지토리에 원격 설정을 하려면 git remote add를 실행합니다. 예시:

```
git remote add origin ssh://codecommit-2/v1/repos/YourRepositoryName
```

더 많은 예시를 보려면 이 포럼의 게시물 및 GitHub에 대한 이 기고문을 참조하세요.

Windows에서의 Git: SSH를 사용하여 연결을 시도할 때 Bash 에뮬레이터 또는 명령줄이 중지됩니다

문제: Windows에 대한 SSH 액세스를 구성하고 명령줄이나 터미널에서 연결을 확인한 후 명령 프롬프트 또는 Bash 에뮬레이터에서 git pull, git push 또는 git clone과 같은 명령을 사용할 때 레지스트리에서 서버의 호스트 키를 캐시할 수 없다는 메시지가 나타나고 캐시에서 키를 저장할 프롬프트가 중지(y/n/return 입력이 수락되지 않음)됩니다.

수정 방법: 이 오류가 발생하는 가장 일반적인 원인은 Git 환경이 인증에 필요한 OpenSSH가 아닌 다른 방식(예: PuTTY)을 사용하도록 구성되었기 때문입니다. 이는 일부 구성에서 키의 캐싱 문제를 유발하는 것으로 알려져 있습니다. 이 문제를 수정하려면 다음 중 한 가지를 시도합니다.

- Bash 에뮬레이터를 열고 `GIT_SSH_COMMAND="ssh"` 파라미터를 Git 명령 앞에 추가합니다. 예를 들어 리포지토리로 푸시를 시도하는 경우, git push를 입력하는 대신 다음을 입력합니다.

  ```
  GIT_SSH_COMMAND="ssh" git push
  ```

- PuTTY를 설치한 경우에는 PuTTY를 열고 호스트 이름(또는 IP 주소)에, 도달하려는 엔드포인트(예: git-codecommit.us-east-2.amazonaws.com)를 입력합니다. 열기를 선택합니다. PuTTY 보안 알림을 통해 메시지가 나타나면 예를 선택하여 키를 영구적으로 캐싱합니다.
- `GIT_SSH` 환경 변수의 이름은 더 이상 사용하지 않으면 바꾸거나 삭제합니다. 새 명령 프롬프트 또는 Bash 에뮬레이터 세션을 연 다음 명령을 다시 시도합니다.

다른 해결 방법은 Stack Overflow의 Git clone/pull continually freezing at Store key in cache를 참조하세요.

일부 Linux 배포판에 공개 키 형식을 지정해야 합니다

문제: 공개-개인 키 페어를 구성하려고 하면 오류가 발생합니다.

가능한 해결 방법: 일부 Linux 배포판에서는 허용되는 공개 키 유형을 명시하는 `~/.ssh/config` 파일에 추가 구성 행이 필요합니다. 자세한 내용은 `PubkeyAcceptedKeyTypes` 배포에 대한 설명서를 참조하세요.

액세스 오류: CodeCommit 리포지토리에 연결할 때 SSH 퍼블릭 키가 거부됩니다

문제: SSH 엔드포인트를 사용하여 CodeCommit 리포지토리와 통신을 시도할 때, `Error: public key denied` 문구를 포함한 오류 메시지가 나타납니다.

수정 방법: SSH 연결을 위한 설정을 완료하지 않은 것이 이 오류의 가장 일반적인 원인입니다. 퍼블릭 또는 프라이빗 SSH 키 페어를 구성한 다음 퍼블릭 키를 IAM 사용자와 연결합니다. SSH 구성에 대한 자세한 내용은 Linux, macOS, Unix에서 SSH 연결 및 Windows에서 SSH 연결 단원을 참조하세요.

AWS CodeCommit에 대한 보안 인증 도우미 및 HTTPS 연결 문제 해결

다음 정보는 AWS CLI 및 HTTPS에 포함된 자격 증명 헬퍼를 사용하여 CodeCommit 리포지토리에 연결할 때 발생하는 일반적인 문제를 해결하는 데 도움이 될 수 있습니다.

> **Note**
>
> 보안 인증 도우미는 페더레이션 액세스, ID 공급자 또는 임시 보안 인증 정보를 사용하여 CodeCommit에 연결하는 데 지원되는 방법이지만 git-remote-codecommit 유틸리티를 설치하고 사용하는 것이 좋습니다. 자세한 내용은 git-remote-codecommit을 AWS CodeCommit 사용하여에 HTTPS 연결을 위한 설정 단계 단원을 참조하십시오.

주제

- 보안 인증 도우미를 구성하는 git config 명령을 실행할 때 오류가 발생합니다.

- 리포지토리를 복제하려고 할 때 사용자 이름을 읽는 동안 오류가 표시됩니다.
- 보안 인증 도우미를 사용할 때 Windows에서 명령을 찾을 수 없음 오류가 발생합니다.
- CodeCommit 리포지토리에 연결할 때 사용자 이름을 묻는 메시지가 표시됩니다.
- macOS용 Git: 보안 인증 도우미를 성공적으로 구성했지만 현재 내 리포지토리에 액세스가 거부되었습니다(403).
- Windows용 Git: Windows용 Git을 설치했지만 내 리포지토리에 액세스가 거부되었습니다(403).

보안 인증 도우미를 구성하는 `git config` 명령을 실행할 때 오류가 발생합니다.

문제: git config 명령을 실행하여 CodeCommit 리포지토리와 통신하도록 보안 인증 도우미를 구성하려고 하면, 인수가 너무 적다는 오류가 표시되거나 Git config 명령 및 구문을 제안하는 사용 프롬프트가 표시됩니다.

가능한 해결 방법: 이 오류가 발생하는 가장 일반적인 이유는 Windows 운영 체제의 명령어에 작은따옴표가 사용되거나 Linux, macOS, Unix 운영 체제의 명령어에 큰따옴표가 사용되기 때문입니다. 정확한 구문은 다음과 같습니다.

- Windows: `git config --global credential.helper "!aws codecommit credential-helper $@"`
- Linux, macOS, Unix: `git config --global credential.helper '!aws codecommit credential-helper $@'`

리포지토리를 복제하려고 할 때 사용자 이름을 읽는 동안 오류가 표시됩니다.

문제: 자격 증명 도우미를 사용하여 리포지토리를 복제하려고 하면 시스템이 리포지토리 URL의 사용자 이름을 읽을 수 없다는 오류가 표시됩니다.

수정 방법:이 오류의 가장 일반적인 이유는 컴퓨터에서 .gitconfig 파일이 올바르게 생성되거나 구성되지 않았기 때문입니다. .gitconfig 파일을 열고 자격 증명 헬퍼가 올바르게 설정되었는지 확인합니다. Linux, macOS 또는 Unix를 실행하는 컴퓨터에 있는 경우의 값이 시스템에 맞게 올바르게 `$HOME` 설정되어 있는지도 확인합니다.

보안 인증 도우미를 사용할 때 Windows에서 명령을 찾을 수 없음 오류가 발생합니다.

문제: AWS CLI를 업데이트한 후 CodeCommit 리포지토리에 대한 자격 증명 헬퍼 연결이 로 실패합니다`aws codecommit credential-helper $@ get: aws: command not found`.

원인:이 오류의 가장 일반적인 이유는 AWS CLI 버전이 Python 3을 사용하는 버전으로 업데이트되었기 때문입니다. MSI 패키지에 알려진 문제가 있습니다. 영향받는 버전 중 하나가 있는지 확인하려면, 명령줄을 열고 `aws --version` 명령을 실행합니다.

결과 Python 버전이 3으로 시작한다면, 영향을 받은 버전이 있는 것입니다. 예시:

```
aws-cli/1.16.62 Python/3.6.2 Darwin/16.7.0 botocore/1.12.52
```

수정 방법: 다음 중 하나를 수해하여 이러한 문제를 해결할 수 있습니다.

- MSI 대신 Python 및 pip를 사용하여 Windows AWS CLI 에를 설치하고 구성합니다. 자세한 내용은 Windows AWS CLI 에 Python, pip 및를 설치하기를 참조하세요.
- `.gitconfig` 파일을 수동으로 편집함으로써 `[credential]` 섹션을 변경하여 로컬 컴퓨터에서 `aws.cmd`를 명시적으로 가리킵니다. 예시:

  ```
  [credential]
      helper = !"\C:\\Program Files\\Amazon\\AWSCLI\\bin\\aws.cmd\" codecommit
   credential-helper $@
      UseHttpPath = true
  ```

- git config 명령의 실행을 통해 `.gitconfig` 파일을 업데이트하여 `aws.cmd`를 명시적으로 참조하고 PATH 환경 변수를 수동으로 업데이트하여 필요한 명령에 대한 경로를 포함시킵니다. 예시:

  ```
  git config --global credential.helper "!aws.cmd codecommit credential-helper $@"
  git config --global credential.UseHttpPath true
  ```

CodeCommit 리포지토리에 연결할 때 사용자 이름을 묻는 메시지가 표시됩니다.

문제: 보안 인증 도우미를 사용해 CodeCommit 리포지토리와 통신을 시도할 때, 사용자 이름을 묻는 메시지가 나타납니다.

수정 방법: AWS 프로필을 구성하거나 사용 중인 프로필이 CodeCommit으로 작업하도록 구성한 프로필인지 확인합니다. 설정에 대한 자세한 내용은 AWS CLI 자격 증명 헬퍼를 사용하여 Linux, macOS 또는 Unix의 AWS CodeCommit 리포지토리에 HTTPS 연결을 위한 설정 단계 또는 AWS CLI 자격 증명 헬퍼를 사용하여 Windows의 AWS CodeCommit 리포지토리에 HTTPS 연결을 위한 설정 단계 단원을 참조하세요. IAM, 액세스 키, 보안 키에 등에 대한 자세한 내용은 IAM 사용자의 액세스 키 관리 및 보안 인증 정보를 얻는 방법을 참조하세요.

macOS용 Git: 보안 인증 도우미를 성공적으로 구성했지만 현재 내 리포지토리에 액세스가 거부되었습니다(403).

문제: macOS에서 보안 인증 도우미가 예상대로 보안 인증 정보를 사용 또는 액세스하지 않은 것 같습니다. 이는 다음과 같은 두 가지 문제로 인해 발생할 수 있습니다.

- AWS CLI 는 리포지토리가 있는와 AWS 리전 다른에 대해 구성됩니다.
- Keychain Access 유틸리티가 만료된 보안 인증 정보를 저장했습니다.

수정 방법: AWS CLI 가 올바른 리전에 대해 구성되어 있는지 확인하려면 aws configure 명령을 실행하고 표시된 정보를 검토합니다. CodeCommit 리포지토리가에 표시된 것과 AWS 리전 다른에 AWS CLI있는 경우 aws configure 명령을 실행하고 값을 해당 리전에 적합한 값으로 변경해야 합니다. 자세한 내용은 1단계: CodeCommit에 대한 초기 구성 단원을 참조하십시오.

OS X 및 macOS에 출시된 Git의 기본 버전은 Keychain Access 유틸리티를 사용하여 생성된 보안 인증 정보를 저장합니다. 보안상의 이유로 CodeCommit 리포지토리에 액세스하기 위해 생성된 암호는 임시 암호이므로, 키체인에 저장된 보안 인증 정보는 약 15분 후에 작동을 멈춥니다. CodeCommit으로 Git 액세스만 하는 경우 다음을 시도합니다.

1. 터미널에서 git config 명령을 실행하여 Keychain Access 유틸리티가 정의된 Git 구성 파일(`gitconfig`)을 찾습니다. 로컬 시스템과 기본 설정에 따라서 `gitconfig` 파일을 하나 이상 가질 수 있습니다.

   ```
   git config -l --show-origin | grep credential
   ```

   이 명령의 출력에서 다음과 유사한 결과를 검색합니다.

   ```
   file:/path/to/gitconfig  credential.helper=osxkeychain
   ```

   이 줄의 시작 부분에 나열된 파일이 편집 대상인 Git 구성 파일입니다.

2. Git 구성 파일을 편집하려면 평문 편집기를 사용하거나 다음 명령을 실행합니다.

```
nano /usr/local/git/etc/gitconfig
```

3. 다음 전략 중 하나를 사용하여 구성을 수정합니다.

   - `helper = osxkeychain`을 포함하는 보안 인증 정보 섹션을 주석 처리하거나 삭제합니다. 예시:

     ```
     # helper = osxkeychain
     ```

   - 컨텍스트가 포함되도록 `aws credential helper` 및 `osxkeychain` 보안 인증 도우미를 모두 업데이트합니다. 예를 들어, `osxkeychain`이 GitHub를 인증하는 데 사용되는 경우에는 다음과 같습니다.

     ```
     [credential "https://git-codecommit.us-east-1.amazonaws\.com"]
       helper = !aws --profile CodeCommitProfile codecommit credential-helper $@
       UseHttpPath = true
     [credential "https://github.com"]
       helper = osxkeychain
     ```

     이 구성에서 Git은 원격 호스트가 '`https://github.com`'과 일치하면 `osxkeychain` 도우미를 사용하고 원격 호스트가 '`https://git-codecommit\.us-east-1\.amazonaws.com`'과 일치하면 보안 인증 도우미를 사용합니다.

   - 보안 인증 도우미 앞에 빈 문자열 도우미를 포함시킵니다. 예를 들어 CLI에서 *`CodeCommitProfile`*이라는 프로필을 사용할 때 `osxkeychain`헬퍼를 사용하지 않으려면

     ```
     [credential]
       helper =
       helper = !aws --profile CodeCommitProfile codecommit credential-helper $@
       UseHttpPath = true
     ```

     > **Tip**
     >
     > 모든 프로필을 제외하려는 경우 CodeCommit과 일치하지 않도록 빈 문자열 헬퍼 라인 뒤에 있는 줄을 구성할 수도 있습니다.
     >
     > ```
     > helper = !aws codecommit credential-helper $@
     > ```

Keychain Access 유틸리티를 계속 이용해 다른 Git 리포지토리의 보안 인증 정보를 캐시하고 싶다면, 명령줄을 코멘트 아웃하는 대신 헤더를 수정해야 합니다. 예를 들어 캐시된 보안 인증 정보를 GitHub에서 허용하고 싶다면, 헤더를 다음과 같이 수정해야 합니다.

```
[credential "https://github.com"]
   helper = osxkeychain
```

Git으로 다른 리포지토리에 액세스한다면, CodeCommit 리포지토리에 필요한 보안 인증 정보를 제공하지 않도록 Keychain Access 유틸리티를 구성할 수 있습니다. Keychain Access 유틸리티를 구성하려면 다음을 수행합니다.

1. Keychain Access 유틸리티를 실행합니다. (Finder를 사용하여 이 유틸리티를 찾을 수 있습니다.)
2. *us-east-2*를 검색`git-codecommit.`*us-east-2*`.amazonaws.com`하고 리포지토리 AWS 리전 가 있는 로 바꿉니다. 행을 강조 표시하고 마우스 오른쪽 버튼을 클릭하여 컨텍스트 메뉴를 연 다음 정보 가져오기를 선택합니다.
3. 액세스 제어 탭을 선택합니다.
4. 액세스 허용 전 확인에서 `git-credential-osxkeychain`을 선택한 후 마이너스 기호를 선택하여 목록에서 제거합니다.

> **Note**
>
> 목록에서 `git-credential-osxkeychain`을 제거하면 Git 명령을 실행할 때마다 대화 상자가 나타납니다. 거부를 선택하여 계속 진행합니다. 팝업이 너무 많이 나타나는 경우 다음과 같이 몇 가지 대체 옵션이 있습니다.
>
> - HTTPS를 통한 보안 인증 도우미 대신 SSH나 Git 보안 인증 정보를 사용하여 CodeCommit에 연결합니다. 자세한 내용은 Linux, macOS, Unix에서 SSH 연결 및 Git 보안 인증 정보를 사용하는 HTTPS 사용자를 위한 설정 단원을 참조하세요.
> - Keychain Access 유틸리티 내, `git-codecommit.us-east-2.amazonaws.com`에 대한 액세스 제어 탭에서 모든 애플리케이션이 이 항목에 액세스하도록 허용(이 항목에 대한 액세스는 제한되지 않음) 옵션을 선택합니다. 이렇게 하면 팝업이 표시되지 않지만 자격 증명은 결국 만료되며(평균 약 15분 소요) 그 다음에는 403 오류 메시지가 표시됩니다. 이 경우 기능을 복원하려면 키체인 항목을 삭제해야 합니다.
> - 키 체인을 사용하도록 기본 설정되어 있지 않은 Git 버전을 설치합니다.

- Keychain 항목을 삭제하기 위한 스크립팅 솔루션을 고려합니다. 커뮤니티에서 생성한 스크립트 솔루션의 샘플을 열람하려면, 제품 및 서비스 통합 섹션에서 OS X 인증서 저장소에 있는 캐시된 보안 인증 정보를 주기적으로 삭제하는 Mac OS X 스크립트를 참조하세요.

Git이 Keychain Access 유틸리티를 아예 사용하지 못하도록 하려면 보안 인증 도우미로 'osxkeychain'을 사용하지 않도록 Git을 구성할 수 있습니다. 예를 들어 터미널을 열고 `git config --system credential.helper` 명령을 실행해 `osxkeychain`이 반환되면 Keychain Access 유틸리티를 사용하도록 Git가 설정됩니다. 다음 명령을 실행하여 이를 변경할 수 있습니다.

```
git config --system --unset credential.helper
```

주의하세요. `--system` 옵션을 사용하여 이 명령을 실행하면 모든 사용자에 대해 시스템 전체의 Git 동작이 변경되며, 이에 따라 CodeCommit 외에 다른 리포지토리 서비스를 사용 중인 경우에는 다른 리포지토리에 의도치 않은 결과가 초래될 수 있습니다. 또한 이 방법은 `sudo` 사용이 필요할 수 있으며, 이 변경을 적용하기에 충분한 권한이 계정에 없을 수 있다는 점에 유의하세요. `git config --system credential.helper` 명령을 다시 실행하여 명령이 성공적으로 적용되었는지 확인하십시오. 자세한 내용은 Git 사용자 지정 - Git 구성 및 스택 오버플로우에 대한 이 기사를 참조하세요.

Windows용 Git: Windows용 Git을 설치했지만 내 리포지토리에 액세스가 거부되었습니다(403).

문제: Windows에서 보안 인증 도우미가 필요한 보안 인증 정보를 사용 또는 액세스하지 않은 것 같습니다. 이는 다음과 같은 다른 문제로 인해 발생할 수 있습니다.

- AWS CLI 는 리포지토리가 있는와 AWS 리전 다른에 대해 구성됩니다.
- 기본적으로 Git for Windows는 자격 증명 헬퍼를 사용하는 CodeCommit 연결과 호환되지 않는 Git AWS Credential Manager 유틸리티를 설치합니다. 설치하면 자격 증명 헬퍼가와 함께 설치되고 CodeCommit에 연결하도록 AWS CLI 구성되었더라도 리포지토리에 대한 연결이 실패합니다.
- Windows용 Git의 일부 버전은 RFC 2617 및 RFC 4559를 완전히 준수하지 않을 수 있으며, 잠재적으로 AWS CLI와 함께 포함된 Git 보안 인증 정보 및 보안 인증 도우미 모두에 대해 문제를 발생시킬 수 있습니다. 자세한 내용은 Version 2.11.0(3) does not ask for username/password를 참조하세요.

수정 방법:

- 에 포함된 자격 증명 헬퍼를 사용하려는 경우 자격 증명 헬퍼를 사용하는 대신 HTTPS를 통해 Git 자격 증명으로 연결하는 것이 AWS CLI좋습니다. IAM 사용자에게 구성된 Git 보안 인증 정보는 AWS CodeCommit의 보안 인증 도우미와 달리 Windows용 Git 보안 인증 정보 관리자와 호환 가능합니다. 자세한 내용은 Git 보안 인증 정보를 사용하는 HTTPS 사용자의 경우 단원을 참조하십시오.

  자격 증명 헬퍼를 사용하려면 AWS CLI 가 올바른에 대해 구성되어 있는지 확인하고 aws configure 명령을 AWS 리전실행하고 표시된 정보를 검토합니다. CodeCommit 리포지토리가에 표시된 것과 AWS 리전 다른에 AWS CLI있는 경우 aws configure 명령을 실행하고 값을 해당 리전에 적합한 값으로 변경해야 합니다. 자세한 내용은 1단계: CodeCommit에 대한 초기 구성 단원을 참조하십시오.

- 가능한 경우 Windows용 Git을 제거하고 다시 설치합니다. Windows용 Git을 설치할 때 Git Credential Manager 유틸리티 설치 옵션의 확인란 선택을 취소합니다. 이 보안 인증 정보 관리자는 AWS CodeCommit의 보안 인증 도우미와 호환되지 않습니다. Git 보안 인증 정보 관리자 또는 다른 보안 인증 정보 관리 유틸리티를 설치하고 제거하지 않으려는 경우, 다음과 같이 `.gitconfig` 파일을 수정하고 CodeCommit의 보안 인증 정보 관리를 추가할 수 있습니다.

  1. 제어판을 열고 보안 인증 정보 관리자를 선택한 다음 저장된 모든 CodeCommit의 보안 인증 정보를 제거합니다.
  2. 메모장과 같은 평문 편집기에서 `.gitconfig` 파일을 엽니다.

     > **Note**
     > 여러 Git 프로필을 사용하는 경우 로컬 및 글로벌 `.gitconfig` 파일을 둘 다 가질 수 있습니다. 적합한 파일을 편집해야 합니다.

  3. 다음 섹션을 `.gitconfig` 파일에 추가합니다.

     ```
     [credential "https://git-codecommit.*.amazonaws.com"]
         helper = !aws codecommit credential-helper $@
         UseHttpPath = true
     ```

  4. 파일을 저장한 다음 새 명령줄 세션을 연 후 연결을 다시 시도합니다.

  CodeCommit 리포지토리에 연결할 AWS CodeCommit 때에 자격 증명 헬퍼를 사용하고 GitHub 리포지토리와 같은 다른 호스팅 리포지토리에 연결할 때 다른 자격 증명 관리 시스템을 사용하려면이 접근 방식을 사용할 수도 있습니다.

  보안 인증 도우미가 기본값으로 사용되도록 재설정하려면 git config 명령을 실행할 때 --global 또는 --local 대신 --system 옵션을 사용할 수 있습니다.

- Windows 컴퓨터에서 Git 보안 인증 정보를 사용하는 경우 연결 문자열의 일부로 Git 보안 인증 정보 사용자 이름을 포함하여 RFC 미준수 문제를 해결할 수 있습니다. 예를 들어, 이 문제를 해결하고 미국 동부(오하이오) 리전에서 *MyDemorepo*라는 리포지토리를 복제하려면 다음과 같이 합니다.

```
git clone https://Your-Git-Credential-Username@git-codecommit.us-
east-2.amazonaws.com/v1/repos/MyDemoRepo my-demo-repo
```

> **Note**
>
> Git 보안 인증 정보 사용자 이름에 @ 문자가 있는 경우, 이 방법은 효과가 없습니다. 문자에 대해 URL 인코딩(URL 이스케이프 또는 퍼센트 인코딩이라고도 함)을 수행해야 합니다.

Git 클라이언트 및 문제 해결 AWS CodeCommit

다음 정보는 AWS CodeCommit 리포지토리에서 Git을 사용할 때 발생하는 일반적인 문제를 해결하는 데 도움이 됩니다. HTTPS 또는 SSH 사용 시 Git 클라이언트와 관련해 발생하는 문제를 해결하려면 Git 보안 인증 정보(HTTPS) 문제 해결, SSH 연결 문제 해결, 보안 인증 도우미(HTTPS) 문제 해결 섹션을 함께 참조하세요.

주제

- Git 오류: RPC failed; result=56, HTTP code = 200 fatal: 원격 엔드가 예기치 않게 끊겼습니다
- Git 오류: 참조 업데이트 명령이 너무 많습니다
- Git 오류: 일부 Git 버전에서 HTTPS를 통한 푸시가 손상되었습니다
- Git 오류: 'gnutls_handshake()가 실패했습니다'
- Git 오류: Git이 CodeCommit 리포지토리를 찾을 수 없거나 리포지토리에 액세스할 권한이 없습니다
- Windows의 Git: 지원되는 인증 방법이 없습니다 (퍼블릭키)

Git 오류: RPC failed; result=56, HTTP code = 200 fatal: 원격 엔드가 예기치 않게 끊겼습니다

문제:대규모 변경 사항, 대량의 변경 사항 또는 대규모 리포지토리 푸시할 때는 장시간 실행 중인 HTTPS 연결이 네트워킹 문제나 방화벽 설정으로 인해 조기에 종료되는 경우가 많습니다.

가능한 해결 방법: SSH를 대신 사용하여 푸시하거나, 대규모 리포지토리를 마이그레이션하는 경우에는 리포지토리를 증분으로 마이그레이션하기 섹션의 다음 단계를 따릅니다. 또한 개별 파일의 크기 제한을 초과하지 않도록 합니다. 자세한 내용은 할당량 단원을 참조하십시오.

Git 오류: 참조 업데이트 명령이 너무 많습니다

문제: 푸시당 최대 참조 업데이트 수는 4,000개입니다. 이 오류는 푸시에 4,000개 이상의 참조 업데이트가 포함되면 나타납니다.

가능한 해결 방법: `git push --all` 및 `git push --tags`를 사용하여 브랜치와 태그를 개별적으로 푸시합니다. 태그가 너무 많으면 태그를 여러 번으로 나누어 푸시합니다. 자세한 내용은 할당량 단원을 참조하십시오.

Git 오류: 일부 Git 버전에서 HTTPS를 통한 푸시가 손상되었습니다

문제: curl을 7.41.0으로 업데이트하면서 발생하는 문제로 인해 SSPI 기반 다이제스트 인증이 실패합니다. 영향을 받는 것으로 알려진 Git 버전 중에는 1.9.5.msysgit.1이 포함되어 있습니다. Windows용 Git의 일부 버전은 RFC 2617 및 RFC 4559를 완전히 준수하지 않을 수 있으며, AWS CLI와 함께 포함된 Git 보안 인증 또는 보안 인증 도우미를 사용하여 HTTPS 연결에 관한 문제를 일으킬 가능성이 있습니다.

가능한 해결 방법: 사용 중인 Git 버전의 알려진 문제를 확인하거나, 이전 또는 이후 버전을 사용합니다. mysysgit에 대한 자세한 내용은 GitHub 포럼에서 HTTPS를 통한 푸시가 손상되었습니다를 참조하세요. Windows용 Git의 버전 문제에 대해 자세히 알아보려면 버전 2.11.0(3)이 사용자 이름과 암호를 묻지 않습니다를 참조하세요.

Git 오류: 'gnutls_handshake()가 실패했습니다'

문제: Linux에서 Git을 사용하여 CodeCommit 리포지토리와 통신을 시도할 때 `error: gnutls_handshake() failed` 문구가 포함된 오류 메시지가 나타납니다.

가능한 해결 방법: OpenSSL에 대해 Git을 컴파일합니다. 한 가지 방안으로, Ubuntu에 질문하기 포럼에서 HTTPS 서버에 연결 시 '오류: nutls_handshake() 실패'를 참조하세요.

또는 HTTPS 대신 SSH를 사용하여 CodeCommit 리포지토리와 통신할 수 있습니다.

Git 오류: Git이 CodeCommit 리포지토리를 찾을 수 없거나 리포지토리에 액세스할 권한이 없습니다

문제: 연결 문자열에 후행 슬래시가 있으면 연결 시도가 실패할 수 있습니다.

가능한 해결 방법: 리포지토리의 이름과 연결 문자열을 올바르게 입력했는지, 그리고 후행 슬래시가 없는지 확인합니다. 자세한 내용은 리포지토리에 연결 단원을 참조하십시오.

Windows의 Git: 지원되는 인증 방법이 없습니다 (퍼블릭키)

문제: Windows용 SSH 액세스를 구성한 후, git pull, git push, git clone와 같은 명령어를 사용하려고 하면 액세스 거부 오류가 표시됩니다.

가능한 해결 방법: 이 오류의 가장 일반적인 원인은 GIT_SSH 환경 변수가 컴퓨터에 존재하고 PuTTY와 같은 다른 연결 유틸리티를 지원하도록 구성되어 있기 때문입니다. 이 문제를 수정하려면 다음 중 한 가지를 시도합니다.

- Bash 에뮬레이터를 열고 `GIT_SSH_COMMAND="ssh"` 파라미터를 Git 명령 앞에 추가합니다. 예를 들어 리포지토리의 복제를 시도하는 경우에는 git clone ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-demo-repo 대신 다음을 실행합니다.

  ```
  GIT_SSH_COMMAND="ssh" git clone ssh://git-codecommit.us-east-2.amazonaws.com/v1/
  repos/MyDemoRepo my-demo-repo
  ```

- `GIT_SSH` 환경 변수의 이름은 더 이상 사용하지 않으면 바꾸거나 삭제합니다. 새 명령 프롬프트 또는 Bash 에뮬레이터 세션을 연 다음 명령을 다시 시도합니다.

SSH를 사용할 때 Windows에서 발생하는 Git 문제를 해결하는 방법에 대해 자세히 알아보려면 SSH 연결 문제 해결 섹션을 참조하세요.

액세스 오류 및 문제 해결 AWS CodeCommit

다음 정보는 AWS CodeCommit 리포지토리 연결 시 발생하는 액세스 오류를 해결하는 데 도움이 될 수 있습니다.

주제

- 액세스 오류: Windows에서 CodeCommit 리포지토리에 연결할 때 사용자 이름과 암호를 묻는 메시지가 나타납니다.
- 액세스 오류: CodeCommit 리포지토리에 연결할 때 퍼블릭 키가 거부됩니다.
- 액세스 오류: CodeCommit 리포지토리에 연결할 때 “Rate Exceeded” 또는 “429” 메시지가 표시됩니다.

액세스 오류: Windows에서 CodeCommit 리포지토리에 연결할 때 사용자 이름과 암호를 묻는 메시지가 나타납니다.

문제: Git을 사용하여 CodeCommit 리포지토리와 통신을 시도할 때 사용자 이름과 암호를 묻는 대화 상자가 나타납니다.

수정 방법: 이것은 Windows의 기본 제공 보안 인증 정보 관리 시스템일 수 있습니다. 구성에 따라 다음 중 하나를 수행합니다.

- Git 보안 인증 정보와 함께 HTTPS를 사용하려는 경우 Git 보안 인증 정보가 시스템에 아직 저장되어 있지 않습니다. Git 보안 인정 정보를 제공하고 계속 진행합니다. 다시 메시지가 표시되지 않습니다. 자세한 내용은 Git 보안 인증 정보를 사용하는 HTTPS 사용자의 경우 단원을 참조하십시오.
- HTTPS를 자격 증명 도우미와 함께 사용하는 경우 Windows 자격 증명 관리 시스템과 호환 AWS CodeCommit되지 않습니다. 취소를 선택합니다.

  또한 Windows용 Git를 설치할 때 Git 보안 인증 정보 관리자를 설치했음을 표시한 것일 수 있습니다. Git 보안 인증 정보 관리자는 AWS CLI에 포함된 CodeCommit용 보안 인증 도우미와 호환되지 않습니다. Git 보안 인증 정보 관리자를 제거하는 것을 고려하세요. CodeCommit용 보안 인증 도우미를 사용하는 대신 git-remote-codecommit를 설치하고 구성할 수도 있습니다.

  자세한 내용은 git-remote-codecommit을 AWS CodeCommit 사용하여에 HTTPS 연결을 위한 설정 단계, 자격 AWS CLI 증명 헬퍼를 사용하여 Windows에서 HTTPS 연결, Windows용 Git: Windows용 Git을 설치했지만 내 리포지토리에 액세스가 거부되었습니다(403). 섹션을 참조하세요.

액세스 오류: CodeCommit 리포지토리에 연결할 때 퍼블릭 키가 거부됩니다.

문제: SSH 엔드포인트를 사용하여 CodeCommit 리포지토리와 통신을 시도할 때 `Error: public key denied` 문구가 포함된 오류 메시지가 나타납니다.

수정 방법: SSH 연결을 위한 설정을 완료하지 않은 것이 이 오류의 가장 일반적인 원인입니다. 퍼블릭 또는 프라이빗 SSH 키 페어를 구성한 다음 퍼블릭 키를 IAM 사용자와 연결합니다. SSH 구성에 대한 자세한 내용은 Linux, macOS, Unix에서 SSH 연결 및 Windows에서 SSH 연결 섹션을 참조하세요.

액세스 오류: CodeCommit 리포지토리에 연결할 때 “Rate Exceeded” 또는 “429” 메시지가 표시됩니다.

문제: CodeCommit 리포지토리와 통신을 시도할 때 “Rate Exceeded” 또는 오류 코드 “429”가 포함된 메시지가 표시됩니다. 통신 속도가 상당히 느려지거나 통신에 실패합니다.

원인: CodeCommit에 대한 모든 호출 AWS Management Console은 애플리케이션, , AWS CLI Git 클라이언트 또는의 호출에 관계없이 초당 최대 요청 수와 전체 활성 요청이 적용됩니다. 어떤 에서든 Amazon Web Services 계정에 허용되는 최대 요청 속도를 초과할 수 없습니다 AWS 리전. 최대 요청 빈도를 넘길 경우, 오류가 수신되며 그 이상의 호출은 Amazon Web Services 계정에 대해 일시적으로 병목 현상을 일으킵니다. 제한이 존재하는 동안에는 CodeCommit에 대한 연결 속도가 느려지거나 연결이 실패할 수 있습니다.

가능한 해결 방법: CodeCommit에 대한 연결 수 또는 호출 수를 낮추거나 요청을 분산시키는 조치를 취합니다. 고려해야 할 몇 가지 접근 방식:

- 요청에, 특히 정기적인 폴링 요청에 지터 실행

  CodeCommit을 정기적으로 폴링하는 애플리케이션이 여러 Amazon EC2 인스턴스에서 실행되는 경우, 1초 단위의 시간 내에 서로 다른 Amazon EC2 인스턴스가 폴링하지 않도록 지터(무작위적인 양의 지연)를 도입합니다. 1분의 시간에 걸쳐 폴링 메커니즘을 균등하게 배포하는 데에는 0초~59초 사이의 임의 숫자가 좋습니다.
- 폴링보다는 이벤트 기반 아키텍처 사용

  이벤트가 발생할 때 호출만 생성되도록, 폴링이 아닌 이벤트 기간 아키텍처를 사용합니다. AWS CodeCommit 이벤트에 대해 CloudWatch Events 알림을 사용하여 워크플로를 트리거하는 것을 고려하세요.
- 오류 재시도 실행과 API 및 자동화된 Git 작업에 대한 지수 백오프

  오류 재시도 및 지수 백오프를 통해 호출 속도를 제한할 수 있습니다. 각 AWS SDK는 자동 재시도 로직과 지수 백오프 알고리즘을 구현합니다. 자동화된 Git 푸시 및 Git 가져오기에 대해서는 자체적인 재시도 로직을 실행해야 할 수 있습니다. 자세한 내용은 오류 재시도 및 지수 백오프를 AWS 참조하세요.
- AWS 지원 센터에 CodeCommit 서비스 할당량 증가 요청

  서비스 제한 증가 혜택을 누리려면 오류 재시도 또는 지수 백오프의 실행 등 여기에 제시된 제안을 이미 따랐음을 확인해야 합니다. 요청 시 제한 문제의 영향을 받는 AWS 리전 Amazon Web Services 계정 및 기간도 제공해야 합니다.

구성 오류 및 문제 해결 AWS CodeCommit

다음 정보는 AWS CodeCommit 리포지토리에 연결할 때 발생할 수 있는 구성 오류를 해결하는 데 도움이 될 수 있습니다.

주제

- 구성 오류: macOS에서 AWS CLI 자격 증명을 구성할 수 없음

구성 오류: macOS에서 AWS CLI 자격 증명을 구성할 수 없음

문제:를 실행`aws configure`하여를 구성하면 `ConfigParseError` 메시지가 AWS CLI표시됩니다.

가능한 해결 방법: 이 오류의 가장 일반적인 원인은 보안 인증 정보 파일이 이미 존재하기 때문입니다. ~/.aws로 이동하여 `credentials`라는 이름의 파일을 찾습니다. 해당 파일을 개명하거나 삭제한 다음 aws configure를 다시 실행합니다.

콘솔 오류 및 문제 해결 AWS CodeCommit

다음 정보는 AWS CodeCommit 리포지토리 사용 시 발생하는 콘솔 오류를 해결하는 데 도움이 될 수 있습니다.

주제

- 액세스 오류: 콘솔 또는에서 CodeCommit 리포지토리에 대한 암호화 키 액세스가 거부됨 AWS CLI
- 암호화 오류: 리포지토리를 해독할 수 없습니다.
- 콘솔 오류: 콘솔에서 CodeCommit 리포지토리의 코드를 검색할 수 없습니다.
- 디스플레이 오류: 파일을 열람하거나 파일 간 비교를 볼 수 없습니다.

액세스 오류: 콘솔 또는에서 CodeCommit 리포지토리에 대한 암호화 키 액세스가 거부됨 AWS CLI

문제: 콘솔 또는에서 CodeCommit에 액세스하려고 하면 `EncryptionKeyAccessDeniedException` 또는 구문이 포함된 AWS CLI오류 메시지가 나타납니다`User is not authorized for the KMS default key for CodeCommit 'aws/codecommit' in your account.`

가능한 해결 방법:이 오류의 가장 일반적인 원인은 Amazon Web Services 계정이 CodeCommit에 AWS Key Management Service필요한 구독되지 않았기 때문입니다. AWS KMS 콘솔을 열고 AWS 관리형 키를 선택한 다음 지금 시작하기를 선택합니다. 현재 AWS Key Management Service 서비스를 구독하고 있지 않다는 메시지가 표시되면 해당 페이지의 지침에 따라 구독합니다. CodeCommit 및에 대한 자세한 내용은 섹션을 AWS Key Management Service참조하세요AWS KMS 및 암호화.

암호화 오류: 리포지토리를 해독할 수 없습니다.

문제: 콘솔 또는에서 CodeCommit 리포지토리에 액세스하려고 하면 문구가 포함된 AWS CLI오류 메시지가 나타납니다`Repository can't be decrypted`.

가능한 수정 사항:이 오류의 가장 일반적인 원인은이 리포지토리의 데이터를 암호화하고 복호화하는 데 사용되는 AWS KMS 키가 활성 상태가 아니거나 삭제 보류 중이기 때문입니다. CodeCommit에는의 활성 AWS 관리형 키 또는 고객 관리형 키 AWS Key Management Service가 필요합니다. AWS KMS 콘솔을 열고 AWS 관리형 키 또는 고객 관리형 키를 선택한 다음 리포지토리에 사용되는 키가 리포지토리가 AWS 리전 있는에 있고 해당 상태가 활성인지 확인합니다. CodeCommit 및에 대한 자세한 내용은 섹션을 AWS Key Management Service참조하세요AWS KMS 및 암호화.

> **Important**
>
> 리포지토리의 데이터를 암호화하고 해독하는 데 사용된 키가 영구적으로 삭제되었거나 다른 방법으로 액세스할 수 없는 경우 해당 키로 암호화된 리포지토리의 데이터에 액세스할 수 없습니다.

콘솔 오류: 콘솔에서 CodeCommit 리포지토리의 코드를 검색할 수 없습니다.

문제: 콘솔에서 리포지토리의 콘텐츠를 검색하려고 하면 액세스를 거부하는 오류 메시지가 나타납니다.

가능한 해결 방법: 이 오류의 가장 일반적인 원인은 Amazon Web Services 계정에 적용된 IAM 정책이 CodeCommit 콘솔에서 코드를 검색하는 데 필요한 권한을 하나 이상 거부하기 때문입니다. CodeCommit 액세스 권한 및 검색에 대한 자세한 내용은 AWS CodeCommit에 대한 인증 및 액세스 제어 섹션을 참조하세요.

디스플레이 오류: 파일을 열람하거나 파일 간 비교를 볼 수 없습니다.

문제: CodeCommit 콘솔에서 파일을 열람하거나 한 파일의 두 버전 간 비교를 보려 하면 파일 또는 차이점 내용이 너무 커서 표시할 수 없다는 오류가 나타납니다.

가능한 해결 방법: 이 오류의 가장 일반적인 원인은 파일이 너무 커서 표시할 수 없거나, 파일의 한 줄에 대한 문자 제한을 초과하는 줄이 하나 이상 포함되어 있거나, 파일의 두 버전 간의 차이점 내용이 줄 제한을 초과하기 때문입니다. 자세한 내용은 할당량 단원을 참조하십시오. 파일 또는 파일 버전 간의 차이를 보려면 원하는 IDE에서 로컬로 파일을 열거나, Git diff 도구를 사용하거나, git diff 명령을 실행할 수 있습니다.

트리거 및 문제 해결 AWS CodeCommit

다음은 AWS CodeCommit에서 트리거로 발생하는 문제를 해결하는 데 유용한 정보입니다.

주제

- 트리거 오류: 리포지토리 트리거가 예상대로 실행되지 않습니다

트리거 오류: 리포지토리 트리거가 예상대로 실행되지 않습니다

문제: 리포지토리에 구성된 하나 이상의 트리거가 실행되지 않는 것처럼 보이거나 예상대로 실행되지 않습니다.

가능한 해결 방법: 트리거의 대상이 AWS Lambda 함수인 경우에는 CodeCommit에서 액세스할 수 있도록 함수의 리소스 정책을 구성했는지 확인합니다. 자세한 내용은 예제 3: CodeCommit 트리거와 AWS Lambda 통합하기 위한 정책 생성 단원을 참조하십시오.

또는 트리거를 편집한 다음, 트리거를 작동하고 싶은 이벤트가 선택되었는지, 그리고 트리거의 작업 대상이 되는 브랜치 중에 작업에 대한 응답을 확인하려는 브랜치가 포함되어 있는지 확인합니다. 트리거 설정을 모든 리포지토리 이벤트 및 모든 브랜치로 변경한 다음 트리거를 테스트합니다. 자세한 내용은 리포지토리 트리거 편집 단원을 참조하십시오.

디버깅 활성화

문제: 디버깅을 활성화하여 Git 명령 실행 방법 및 내 리포지토리에 대한 자세한 정보를 알고 싶습니다.

수정 방법: 다음을 수행해 보세요.

1. 터미널 또는 명령 프롬프트에서 로컬 시스템에 다음 명령을 실행한 후 Git 명령을 실행하십시오.

   Linux, macOS, Unix의 경우:

   ```
   export GIT_TRACE_PACKET=1
   export GIT_TRACE=1
   export GIT_CURL_VERBOSE=1
   ```

   Windows의 경우:

   ```
   set GIT_TRACE_PACKET=1
   set GIT_TRACE=1
   ```

```
set GIT_CURL_VERBOSE=1
```

> **Note**
>
> `GIT_CURL_VERBOSE` 설정은 HTTPS 연결에만 유용합니다. SSH는 `libcurl` 라이브러리를 사용하지 않습니다.

2. Git 리포지토리에 대한 자세한 내용은 최신 버전의 git-sizer를 설치하는 것이 좋습니다. 운영 체제 및 환경에 적합한 유틸리티를 설치하기 위한 지침을 따릅니다. 설치되면 명령줄 또는 터미널에서 디렉터리를 로컬 리포지토리로 변경한 다음 다음 명령을 실행합니다.

```
git-sizer --verbose
```

> **Tip**
>
> 특히 시간이 지남에 따라 문제를 해결할 때 다른 사람과 쉽게 공유할 수 있도록 명령의 출력을 파일에 저장하는 것이 좋습니다.

AWS CodeCommit 참조

다음 참조 주제는 CodeCommit, Git AWS 리전, 서비스 제한 등을 더 잘 이해하는 데 도움이 될 수 있습니다.

주제

- 에 대한 리전 및 Git 연결 엔드포인트 AWS CodeCommit
- 인터페이스 VPC 엔드포인트 AWS CodeCommit 와 함께 사용
- 의 할당량 AWS CodeCommit
- AWS CodeCommit 명령줄 참조
- 기본 Git 명령

에 대한 리전 및 Git 연결 엔드포인트 AWS CodeCommit

각 CodeCommit 리포지토리는와 연결됩니다 AWS 리전. CodeCommit은 서비스에 요청을 하기 위한 지역별 엔드포인트를 제공합니다. 또한 CodeCommit은 CodeCommit을 사용할 수 있는 모든 리전에서 SSH 및 HTTPS 프로토콜 모두에 대한 Git 연결 엔드포인트를 제공합니다.

이 안내서의 모든 예제는 미국 동부(오하이오)의 Git과 동일한 엔드포인트 URL(`git-codecommit.us-east-2.amazonaws.com`)을 사용합니다. 그러나 Git을 사용하고 연결을 구성할 때는 AWS 리전 리포지토리를 호스팅하는 과 일치하는 Git 연결 엔드포인트를 선택해야 합니다. 예를 들어, 미국 동부(버지니아 북부)에서 리포지토리에 연결을 만드는 경우 엔드포인트 URL인 `git-codecommit.us-east-1.amazonaws.com`을 사용합니다. 이는 API 호출에도 마찬가지입니다. AWS CLI 또는 SDKs를 사용하여 CodeCommit 리포지토리에 연결할 때는 리포지토리에 올바른 리전 엔드포인트를 사용해야 합니다.

주제

- CodeCommit AWS 리전 지원
- Git 연결 엔드포인트
- CodeCommit의 서버 지문

CodeCommit AWS 리전 지원

AWS 리전다음에서 CodeCommit 리포지토리를 생성하고 사용할 수 있습니다.

- 미국 동부(오하이오)
- 미국 동부(버지니아 북부)
- 미국 서부(캘리포니아 북부)
- 미국 서부(오리건)
- 유럽(아일랜드)
- 유럽(런던)
- 유럽(파리)
- 유럽(프랑크푸르트)
- 유럽(스톡홀름)
- 유럽(밀라노)
- 아프리카(케이프타운)
- 이스라엘(텔아비브)
- 아시아 태평양(도쿄)
- 아시아 태평양(싱가포르)
- 아시아 태평양(시드니)
- 아시아 태평양(자카르타)
- 중동(UAE)
- 아시아 태평양(서울)
- 아시아 태평양(오사카)
- 아시아 태평양(뭄바이)
- 아시아 태평양(하이데라바드)
- 아시아 태평양(홍콩)
- 남아메리카(상파울루)
- 중동(바레인)
- 캐나다(중부)
- 중국(베이징)
- 중국(닝샤)
- AWS GovCloud(미국 서부)
- AWS GovCloud(미국 동부)

CodeCommit이 일부 리전에서 Federal Information Processing Standard(FIPS) Publication 140-2 정부 표준에 대한 지원을 추가했습니다. FIPS 및 FIPS 엔드포인트에 대한 자세한 내용은 Federal Information Processing Standard(FIPS) 140-2 개요를 참조하세요. FIPS를 지원하는 Git 연결 엔드포인트에 대해서는 Git 연결 엔드포인트 단원을 참조하세요.

CodeCommit에 대한 리전 엔드포인트 AWS CLI, 서비스 및 API 호출에 대한 자세한 내용은 AWS CodeCommit 엔드포인트 및 할당량을 참조하세요.

Git 연결 엔드포인트

CodeCommit 리포지토리에 Git 연결을 구성하는 경우 다음 URL을 사용하세요.

에 대한 Git 연결 엔드포인트 AWS CodeCommit

지역명	지역	엔드포인트 URL	프로토콜
미국 동부(오하이오)	us-east-2	https://git-codecommit.us-east-2.amazonaws.com	HTTPS
미국 동부(오하이오)	us-east-2	ssh://git-codecommit.us-east-2.amazonaws.com	SSH
미국 동부(오하이오)	us-east-2	https://git-codecommit-fips.us-east-2.amazonaws.com	HTTPS
미국 동부(버지니아 북부)	us-east-1	https://git-codecommit.us-east-1.amazonaws.com	HTTPS
미국 동부(버지니아 북부)	us-east-1	ssh://git-codecommit.us-east-1.amazonaws.com	SSH
미국 동부(버지니아 북부)	us-east-1	https://git-codecommit-fips.us-east-1.amazonaws.com	HTTPS

지역명	지역	엔드포인트 URL	프로토콜
미국 서부(오레곤)	us-west-2	https://git-codecommit.us-west-2.amazonaws.com	HTTPS
미국 서부(오레곤)	us-west-2	ssh://git-codecommit.us-west-2.amazonaws.com	SSH
미국 서부(오리건)	us-west-2	https://git-codecommit-fips.us-west-2.amazonaws.com	HTTPS
미국 서부(캘리포니아 북부)	us-west-1	https://git-codecommit.us-west-1.amazonaws.com	HTTPS
미국 서부(캘리포니아 북부)	us-west-1	ssh://git-codecommit.us-west-1.amazonaws.com	SSH
미국 서부(캘리포니아 북부)	us-west-1	https://git-codecommit-fips.us-west-1.amazonaws.com	HTTPS
유럽(아일랜드)	eu-west-1	https://git-codecommit.eu-west-1.amazonaws.com	HTTPS
유럽(아일랜드)	eu-west-1	ssh://git-codecommit.eu-west-1.amazonaws.com	SSH
아시아 태평양(도쿄)	ap-northeast-1	https://git-codecommit.ap-northeast-1.amazonaws.com	HTTPS

지역명	지역	엔드포인트 URL	프로토콜
아시아 태평양(도쿄)	ap-northeast-1	ssh://git-codecommit.ap-northeast-1.amazonaws.com	SSH
아시아 태평양(싱가포르)	ap-southeast-1	https://git-codecommit.ap-southeast-1.amazonaws.com	HTTPS
아시아 태평양(싱가포르)	ap-southeast-1	ssh://git-codecommit.ap-southeast-1.amazonaws.com	SSH
아시아 태평양(시드니)	ap-southeast-2	https://git-codecommit.ap-southeast-2.amazonaws.com	HTTPS
아시아 태평양(시드니)	ap-southeast-2	ssh://git-codecommit.ap-southeast-2.amazonaws.com	SSH
아시아 태평양(자카르타)	ap-southeast-3	https://git-codecommit.ap-southeast-3.amazonaws.com	HTTPS
아시아 태평양(자카르타)	ap-southeast-3	ssh://git-codecommit.ap-southeast-3.amazonaws.com	SSH
중동(UAE)	me-central-1	https://git-codecommit.me-central-1.amazonaws.com	HTTPS
중동(UAE)	me-central-1	ssh://git-codecommit.me-central-1.amazonaws.com	SSH

지역명	지역	엔드포인트 URL	프로토콜
유럽(프랑크푸르트)	eu-central-1	https://git-codecommit.eu-central-1.amazonaws.com	HTTPS
유럽(프랑크푸르트)	eu-central-1	ssh://git-codecommit.eu-central-1.amazonaws.com	SSH
아시아 태평양(서울)	ap-northeast-2	https://git-codecommit.ap-northeast-2.amazonaws.com	HTTPS
아시아 태평양(서울)	ap-northeast-2	ssh://git-codecommit.ap-northeast-2.amazonaws.com	SSH
남아메리카(상파울루)	sa-east-1	https://git-codecommit.sa-east-1.amazonaws.com	HTTPS
남아메리카(상파울루)	sa-east-1	ssh://git-codecommit.sa-east-1.amazonaws.com	SSH
유럽(런던)	eu-west-2	https://git-codecommit.eu-west-2.amazonaws.com	HTTPS
유럽(런던)	eu-west-2	ssh://git-codecommit.eu-west-2.amazonaws.com	SSH
아시아 태평양(뭄바이)	ap-south-1	https://git-codecommit.ap-south-1.amazonaws.com	HTTPS

지역명	지역	엔드포인트 URL	프로토콜
아시아 태평양(뭄바이)	ap-south-1	ssh://git-codecommit.ap-south-1.amazonaws.com	SSH
아시아 태평양(하이데라바드)	ap-south-2	https://git-codecommit.ap-south-2.amazonaws.com	HTTPS
아시아 태평양(하이데라바드)	ap-south-2	ssh://git-codecommit.ap-south-2.amazonaws.com	SSH
캐나다(중부)	ca-central-1	https://git-codecommit.ca-central-1.amazonaws.com	HTTPS
캐나다(중부)	ca-central-1	ssh://git-codecommit.ca-central-1.amazonaws.com	SSH
캐나다(중부)	ca-central-1	https://git-codecommit-fips.ca-central-1.amazonaws.com	HTTPS
유럽(파리)	eu-west-3	https://git-codecommit.eu-west-3.amazonaws.com	HTTPS
유럽(파리)	eu-west-3	ssh://git-codecommit.eu-west-3.amazonaws.com	SSH
AWS GovCloud(미국 서부)	us-gov-west-1	https://git-codecommit.us-gov-west-1.amazonaws.com	HTTPS

지역명	지역	엔드포인트 URL	프로토콜
AWS GovCloud(미국 서부)	us-gov-west-1	ssh://git-codecommit.us-gov-west-1.amazonaws.com	SSH
AWS GovCloud(미국 서부)	us-gov-west-1	https://git-codecommit-fips.us-gov-west-1.amazonaws.com	HTTPS
AWS GovCloud(미국 동부)	us-gov-east-1	https://git-codecommit.us-gov-east-1.amazonaws.com	HTTPS
AWS GovCloud(미국 동부)	us-gov-east-1	ssh://git-codecommit.us-gov-east-1.amazonaws.com	SSH
AWS GovCloud(미국 동부)	us-gov-east-1	https://git-codecommit-fips.us-gov-east-1.amazonaws.com	HTTPS
유럽(스톡홀름)	eu-north-1	https://git-codecommit.eu-north-1.amazonaws.com	HTTPS
유럽(스톡홀름)	eu-north-1	ssh://git-codecommit.eu-north-1.amazonaws.com	SSH
중동(바레인)	me-south-1	https://git-codecommit.me-south-1.amazonaws.com	HTTPS
중동(바레인)	me-south-1	ssh://git-codecommit.me-south-1.amazonaws.com	SSH

지역명	지역	엔드포인트 URL	프로토콜
아시아 태평양(홍콩)	ap-east-1	https://git-codecommit.ap-east-1.amazonaws.com	HTTPS
아시아 태평양(홍콩)	ap-east-1	ssh://git-codecommit.ap-east-1.amazonaws.com	SSH
중국(베이징)	cn-north-1	https://git-codecommit.cn-north-1.amazonaws.com.cn	HTTPS
중국(베이징)	cn-north-1	ssh://git-codecommit.cn-north-1.amazonaws.com.cn	SSH
중국(닝샤)	cn-northwest-1	https://git-codecommit.cn-northwest-1.amazonaws.com.cn	HTTPS
중국(닝샤)	cn-northwest-1	ssh://git-codecommit.cn-northwest-1.amazonaws.com.cn	SSH
유럽(밀라노)	eu-south-1	https://git-codecommit.eu-south-1.amazonaws.com	HTTPS
유럽(밀라노)	eu-south-1	ssh://git-codecommit.eu-south-1.amazonaws.com	SSH
아시아 태평양(오사카)	ap-northeast-3	https://git-codecommit.ap-northeast-3.amazonaws.com	HTTPS

지역명	지역	엔드포인트 URL	프로토콜
아시아 태평양(오사카)	ap-northeast-3	ssh://git-codecommit.ap-northeast-3.amazonaws.com	SSH
아프리카(케이프타운)	af-south-1	https://git-codecommit.af-south-1.amazonaws.com	HTTPS
아프리카(케이프타운)	af-south-1	ssh://git-codecommit.af-south-1.amazonaws.com	SSH
이스라엘(텔아비브)	il-central-1	https://git-codecommit.il-central-1.amazonaws.com	HTTPS
이스라엘(텔아비브)	il-central-1	ssh://git-codecommit.il-central-1.amazonaws.com	SSH

CodeCommit의 서버 지문

다음 표에는 CodeCommit의 Git 연결 엔드포인트을 위한 퍼블릭 지문이 나열되어 있습니다. 이러한 서버 지문은 엔드포인트를 알려진 호스트 파일에 추가하기 위한 확인 프로세스의 일부로서 표시됩니다.

CodeCommit의 퍼블릭 지문

Server	암호화 해시 유형	지문
git-codecommit.us-east-2.amazonaws.com	MD5	`a9:6d:03:ed:08:42:21:be:06:e1:e0:2a:d1:75:31:5e`
git-codecommit.us-east-2.amazonaws.com	SHA256	`3lBlW2g5xn/NA2Ck6dyeJIrQOWvn7n8UEs56fG6ZIzQ`

Server	암호화 해시 유형	지문
git-codecommit.us-east-1.amazonaws.com	MD5	a6:9c:7d:bc:35:f5:d4:5f:8b:ba:6f:c8:bc:d4:83:84
git-codecommit.us-east-1.amazonaws.com	SHA256	eLMY1j0DKA4uvDZcl/KgtIayZANwX6t8+8isPtotBoY
git-codecommit.us-west-2.amazonaws.com	MD5	a8:68:53:e3:99:ac:6e:d7:04:7e:f7:92:95:77:a9:77
git-codecommit.us-west-2.amazonaws.com	SHA256	0pJx9SQpkbPUAHwy58UVIq0IHcyo1fwCpOOuVgcAWPo
git-codecommit.eu-west-1.amazonaws.com	MD5	93:42:36:ea:22:1f:f1:0f:20:02:4a:79:ff:ea:12:1d
git-codecommit.eu-west-1.amazonaws.com	SHA256	tKjRkOL8dmJyTmSbeSdN1S8F/f0iql3RlvqgTOP1UyQ
git-codecommit.ap-northeast-1.amazonaws.com	MD5	8e:a3:f0:80:98:48:1c:5c:6f:59:db:a7:8f:6e:c6:cb
git-codecommit.ap-northeast-1.amazonaws.com	SHA256	Xk/WeYD/K/bnBybzhiuu4dWpBJtXPf7E30jHU7se4Ow
git-codecommit.ap-southeast-1.amazonaws.com	MD5	65:e5:27:c3:09:68:0d:8e:b7:6d:94:25:80:3e:93:cf

Server	암호화 해시 유형	지문
git-codecommit.ap-southeast-1.amazonaws.com	SHA256	`ZIsVa7OVzxrTIf+Rk4UbhPv6Es22mSB3uTBojfPXIno`
git-codecommit.ap-southeast-2.amazonaws.com	MD5	`7b:d2:c1:24:e6:91:a5:7b:fa:c1:0c:35:95:87:da:a0`
git-codecommit.ap-southeast-2.amazonaws.com	SHA256	`nYp+gHas80HY3DqbP4yanCDFhqDVjseefVbHEXqH2Ec`
git-codecommit.ap-southeast-3.amazonaws.com	MD5	`64:d9:e0:53:19:4f:a8:91:9a:c3:53:22:a6:a8:ed:a6`
git-codecommit.ap-southeast-3.amazonaws.com	SHA256	`ATdkGSFhpqIu7RqUVT/1RZo6MLxxxUW9NoDVMbAc/6g`
git-codecommit.me-central-1.amazonaws.com	MD5	`bd:fa:e2:f9:05:84:d6:39:6f:bc:d6:8d:fe:de:61:76`
git-codecommit.me-central-1.amazonaws.com	SHA256	`grceUDWubo4MzG1NoaKZKUfrgPvfN3ijliOnQr1lTZA`
git-codecommit.eu-central-1.amazonaws.com	MD5	`74:5a:e8:02:fc:b2:9c:06:10:b4:78:84:65:94:22:2d`
git-codecommit.eu-central-1.amazonaws.com	SHA256	`MwGrkiEki8QkkBtlAgXbYt0hoZYBnZF62VY5RzGJEUY`

Server	암호화 해시 유형	지문
git-codecommit.ap-northeast-2.amazonaws.com	MD5	`9f:68:48:9b:5f:fc:96:69:39:45:58:87:95:b3:69:ed`
git-codecommit.ap-northeast-2.amazonaws.com	SHA256	`eegAPQrWY9YsYo9ZHIKOmxetfXBHzAZd8Eya53Qcwko`
git-codecommit.sa-east-1.amazonaws.com	MD5	`74:99:9d:ff:2b:ef:63:c6:4b:b4:6a:7f:62:c5:4b:51`
git-codecommit.sa-east-1.amazonaws.com	SHA256	`kW+VKB0jpRaG/ZbXkgbtMQbKgEDK7JnISV3SVoyCmzU`
git-codecommit.us-west-1.amazonaws.com	MD5	`3b:76:18:83:13:2c:f8:eb:e9:a3:d0:51:10:32:e7:d1`
git-codecommit.us-west-1.amazonaws.com	SHA256	`gzauWTWXDK2u5KuMMi5vbKTmfyerdIwgSbzYBODLpzg`
git-codecommit.eu-west-2.amazonaws.com	MD5	`a5:65:a6:b1:84:02:b1:95:43:f9:0e:de:dd:ed:61:d3`
git-codecommit.eu-west-2.amazonaws.com	SHA256	`r0Rwz5k/IHp/QyrRnfiM9j02D5UEqMbtFNTuDG2hNbs`
git-codecommit.ap-south-1.amazonaws.com	MD5	`da:41:1e:07:3b:9e:76:a0:c5:1e:64:88:03:69:86:21`

Server	암호화 해시 유형	지문
git-codecommit.ap-south-1.amazonaws.com	SHA256	hUKwnTj7+Xpx4Kddb6p45j4RazIJ4IhAMD8k29itOfE
git-codecommit.ap-south-2.amazonaws.com	MD5	bc:cc:9f:15:f8:f3:58:a2:68:65:21:e2:23:71:8d:ce
git-codecommit.ap-south-2.amazonaws.com	SHA256	Xe0CyZEOvgR5Xa2YUGqf+jn8/Ut7l7nX/CmslSFNEig
git-codecommit.ca-central-1.amazonaws.com	MD5	9f:7c:a2:2f:8c:b5:74:fd:ab:b7:e1:fd:af:46:ed:23
git-codecommit.ca-central-1.amazonaws.com	SHA256	Qz5puafQdANVprLlj6r0Qyh4lCNsF6ob61dGcPtFS7w
git-codecommit.eu-west-3.amazonaws.com	MD5	1b:7f:97:dd:d7:76:8a:32:2c:bd:2c:7b:33:74:6a:76
git-codecommit.eu-west-3.amazonaws.com	SHA256	uw7c2FL564jVoFgtc+ikzILnKBsZz7t9+CFdSJjKbLI
git-codecommit.us-gov-west-1.amazonaws.com	MD5	9f:6c:19:3b:88:cd:e8:88:1b:9c:98:6a:95:31:8a:69
git-codecommit.us-gov-west-1.amazonaws.com	SHA256	djXQoSIFcg8vHe0KVH1xW/gOF9X37tWTqu4Hkng75x4

Server	암호화 해시 유형	지문
git-codecommit.us-gov-east-1.amazonaws.com	MD5	`00:8d:b5:55:6f:05:78:05:ed:ea:cb:3f:e6:f0:62:f2`
git-codecommit.us-gov-east-1.amazonaws.com	SHA256	`fVb+R0z7qW7minenW+rUpAABRCRBTCzmETAJEQrg98`
git-codecommit.eu-north-1.amazonaws.com	MD5	`8e:53:d8:59:35:88:82:fd:73:4b:60:8a:50:70:38:f4`
git-codecommit.eu-north-1.amazonaws.com	SHA256	`b6KSK7xKq+V8jl7iuAcjqXsG7zkqoUZZmmhYYFBq1wQ`
git-codecommit.me-south-1.amazonaws.com	MD5	`0e:39:28:56:d5:41:e6:8d:fa:81:45:37:fb:f3:cd:f7`
git-codecommit.me-south-1.amazonaws.com	SHA256	`O+NToCGgjrHekiBuOl0ad7ROGEsz+DBLXOd/c9wc0JU`
git-codecommit.ap-east-1.amazonaws.com	MD5	`a8:00:3d:24:52:9d:61:0e:f6:e3:88:c8:96:01:1c:fe`
git-codecommit.ap-east-1.amazonaws.com	SHA256	`LafadYwUYW8hONoTRpojbjNs9IRnbEwHtezD3aAIBX0`
git-codecommit.cn-north-1.amazonaws.com .cn	MD5	`11:7e:2d:74:9e:3b:94:a2:69:14:75:6f:5e:22:3b:b3`

Server	암호화 해시 유형	지문
git-codecommit.cn-north-1.amazonaws.com .cn	SHA256	IYUXxH2OpTDsyYMLIp+JY8CTLS4UX+ZC5JVZXPRaxc8
git-codecommit.cn-northwest-1.amazonaws.com .cn	MD5	2e:a7:fb:4c:33:ac:6c:f9:aa:f2:bc:fb:0a:7b:1e:b6
git-codecommit.cn-northwest-1.amazonaws.com .cn	SHA256	wqjd6eHd0+mOBx+dCNuL0omUoCNjaDtZiEpWj5TmCfQ
git-codecommit.eu-south-1.amazonaws.com	MD5	b9:f6:5d:e2:48:92:3f:a9:37:1e:c4:d0:32:0e:fb:11
git-codecommit.eu-south-1.amazonaws.com	SHA256	lyXrWbCg3uQmJrl1XxB/ASR7ugW1Ysf5yzYOJbudHsI
git-codecommit.ap-northeast-3.amazonaws.com	MD5	25:17:40:da:b9:d4:18:c3:b6:b3:fb:ed:1c:20:fe:29
git-codecommit.ap-northeast-3.amazonaws.com	SHA256	2B815B9F0AvwLnRxSVxUz4kDYmtEQUGGdQYP8OQLXhA
git-codecommit.af-south-1.amazonaws.com	MD5	21:a0:ba:d7:c1:d1:b5:39:98:8d:4d:7c:96:f5:ca:29
git-codecommit.af-south-1.amazonaws.com	SHA256	C34ji3x/cnsDZjUpyNGXdE5pjHYimqJrQZ3leTgqJHM

Server	암호화 해시 유형	지문
git-codecommit.il-central-1.amazonaws.com	MD5	`04:74:89:16:98:7a:61:b1:69:46:42:3c:d1:b4:ac:a9`
git-codecommit.il-central-1.amazonaws.com	SHA256	`uFxhp51kUWhleTLeYbxQVYm4RnNLNZ5Dbdm1cgdSl/8`

인터페이스 VPC 엔드포인트 AWS CodeCommit 와 함께 사용

Amazon Virtual Private Cloud(VPC)를 사용하여 AWS 리소스를 호스팅하는 경우 VPC와 CodeCommit 간에 프라이빗 연결을 설정할 수 있습니다. 이 연결을 사용하면 CodeCommit이 퍼블릭 인터넷을 통하지 않고 VPC의 리소스와 통신하게 할 수 있습니다.

Amazon VPC는 정의한 가상 네트워크에서 AWS 리소스를 시작하는 데 사용할 수 있는 AWS 서비스입니다. VPC를 사용하여 IP 주소 범위, 서브넷, 라우팅 테이블, 네트워크 게이트웨이 등의 네트워크 설정을 제어할 수 있습니다. VPC 엔드포인트를 사용하면 VPC와 AWS 서비스 간의 라우팅이 AWS 네트워크에서 처리되며 IAM 정책을 사용하여 서비스 리소스에 대한 액세스를 제어할 수 있습니다.

VPC를 CodeCommit에 연결하려면 CodeCommit에 대해 인터페이스 VPC 엔드포인트를 정의합니다. 인터페이스 엔드포인트는 지원되는 AWS 서비스로 향하는 트래픽의 진입점 역할을 하는 프라이빗 IP 주소가 있는 탄력적 네트워크 인터페이스입니다. 이 엔드포인트를 이용하면 인터넷 게이트웨이나 Network Address Translation(NAT) 인스턴스 또는 VPN 연결 없이도 CodeCommit에 안정적이고 확장 가능하게 연결됩니다. 자세한 내용은 Amazon VPC 사용 설명서의 Amazon VPC란 무엇입니까를 참조하세요.

> **Note**
>
> VPC 지원을 제공하고 CodeCommit과 통합되는와 같은 다른 AWS 서비스는 해당 통합에 Amazon VPC 엔드포인트 사용을 지원하지 않을 AWS CodePipeline수 있습니다. 예를 들어 CodePipeline과 CodeCommit 간의 트래픽은 VPC 서브넷 범위로 제한할 수 없습니다. 통합을 지원하는 서비스(예: AWS Cloud9)에는 AWS Systems Manager와 같은 추가 서비스가 필요할 수 있습니다.

인터페이스 VPC 엔드포인트는 프라이빗 IP 주소가 있는 탄력적 네트워크 인터페이스를 사용하여 AWS 서비스 간 프라이빗 통신을 지원하는 AWS 기술인 AWS PrivateLink로 구동됩니다. 자세한 내용은 AWS PrivateLink 섹션을 참조하세요.

다음은 Amazon VPC 사용자를 위한 단계들입니다. 자세한 내용은 Amazon VPC 사용 설명서의 시작하기를 참조하세요.

가용성

CodeCommit은 현재 AWS 리전다음에서 VPC 엔드포인트를 지원합니다.

- 미국 동부(오하이오)
- 미국 동부(버지니아 북부)
- 미국 서부(캘리포니아 북부)
- 미국 서부(오리건)
- 유럽(아일랜드)
- 유럽(런던)
- 유럽(파리)
- 유럽(프랑크푸르트)
- 유럽(스톡홀름)
- 유럽(밀라노)
- 아프리카(케이프타운)
- 이스라엘(텔아비브)
- 아시아 태평양(도쿄)
- 아시아 태평양(싱가포르)
- 아시아 태평양(시드니)
- 아시아 태평양(자카르타)
- 중동(UAE)
- 아시아 태평양(서울)
- 아시아 태평양(오사카)
- 아시아 태평양(뭄바이)
- 아시아 태평양(하이데라바드)
- 아시아 태평양(홍콩)

- 남아메리카(상파울루)
- 중동(바레인)
- 캐나다(중부)
- 중국(베이징)
- 중국(닝샤)
- AWS GovCloud(미국 서부)
- AWS GovCloud(미국 동부)

CodeCommit의 VPC 엔드포인트 생성

VPC로 CodeCommit을 사용하려면, CodeCommit을 위한 인터페이스 VPC 엔드포인트를 생성합니다. CodeCommit은 Git 작업과 CodeCommit API 작업을 위해 별도의 엔드포인트가 필요합니다. 비즈니스 요구 사항에 따라 VPC 엔드포인트를 두 개 이상 생성해야 할 수 있습니다. CodeCommit을 위한 VPC 엔드포인트를 생성하려면, AWS 서비스를 선택하고, 서비스 이름의 다음 옵션 중에서 선택합니다.

- com.amazonaws.***region***.git-codecommit: CodeCommit 리포지토리로 Git 작업을 하기 위한 VPC 엔드포인트를 생성하려는 경우 이 옵션을 선택합니다. 예를 들어 사용자가 CodeCommit 리포지토리와 상호 작용할 때 Git 클라이언트와 `git pull`, `git commit`, `git push` 등의 명령을 사용려는 경우 이 옵션을 선택합니다.
- com.amazonaws.***region***.git-codecommit-fips: CodeCommit 리포지토리로 미국 연방 정보 처리 표준(FIPS) Publication 140-2 US 정부 표준을 준수하는 Git 작업을 위한 VPC 엔드포인트를 생성하려는 경우 이 옵션을 선택합니다.

> **Note**
>
> 일부 AWS 리전에서는 Git용 FIPS 엔드포인트를 사용할 수 없습니다. 자세한 내용은 Git 연결 엔드포인트 단원을 참조하십시오.

- com.amazonaws.***region***.codecommit: CodeCommit API 작업을 위한 VPC 엔드포인트를 생성하려는 경우 이 옵션을 선택합니다. 예를 들어 사용자가 AWS CLI, CodeCommit API 또는 AWS SDKs를 사용하여 , `CreateRepository` `ListRepositories`및와 같은 작업을 위해 CodeCommit과 상호 작용하는 경우이 옵션을 선택합니다`PutFile`.
- com.amazonaws.***region***.codecommit-fips: 미국 연방 정보 처리 표준(FIPS) Publication 140-2 US 정부 표준을 준수하는 CodeCommit API 작업을 위한 VPC 엔드포인트를 생성하려는 경우 이 옵션을 선택합니다.

> **Note**
>
> 일부 AWS 리전에서는 FIPS 엔드포인트를 사용할 수 없습니다. 자세한 내용은 FIPS(Federal Information Processing Standard) 140-2 개요 AWS CodeCommit 의 항목을 참조하세요. https://aws.amazon.com/compliance/fips/

CodeCommit을 위한 VPC 엔드포인트 정책 생성

CodeCommit의 Amazon VPC 엔드포인트에 대한 정책을 생성할 수 있습니다. 이 정책에서는 다음을 지정할 수 있습니다.

- 작업을 수행할 수 있는 위탁자.
- 수행할 수 있는 작업입니다.
- 수행되는 작업을 가질 수 있는 리소스입니다.

예를 들어 한 회사가 리포지토리에 대한 액세스를 VPC의 네트워크 주소 범위로 제한하고자 할 수 있습니다. 이 종류의 정책 예는 예제 3: 지정 IP 주소 범위에서 연결된 사용자에게 리포지토리에 대한 액세스 허용 에서 볼 수 있습니다. 이 회사는 미국 동부(오하이오) 리전에 대해 `com.amazonaws.us-east-2.codecommit`과 `com-amazonaws.us-east-2.git-codecommit-fips`라는 두 Git VPC 엔드포인트를 구성했습니다. 이 회사는 FIPS를 준수하는 엔드포인트에서만 *MyDemoRepo*라는 CodeCommit 리포지토리로 코드를 푸시할 수 있도록 허용합니다. 그리고 이를 강제하기 위해 Git 푸시 작업을 명확히 거부하는 `com.amazonaws.us-east-2.codecommit` 엔드포인트에 대해 다음과 비슷한 정책을 구성합니다.

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "codecommit:GitPush",
            "Effect": "Deny",
            "Resource": "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo",
            "Principal": "*"
```

```
        }
    ]
}
```

> **Important**
>
> 전역 조건 키`aws:VpcSourceIp`는 `git push` 명령에 대한 IAM 정책에서 CodeCommit 리포지토리가 지원되지 않습니다.

자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트 생성을 참조하세요.

의 할당량 AWS CodeCommit

다음 표에서는 CodeCommit의 할당량에 대해 설명합니다. 변경 가능한 제한에 대한 자세한 내용은 AWS CodeCommit 엔드포인트 및 할당량을 참조하세요. 할당량 증가 요청에 대한 자세한 내용은 AWS 서비스 할당량을 참조하세요. Git 및 기타 소프트웨어의 필수 버전에 대한 자세한 내용은 CodeCommit, Git 및 기타 구성 요소에 대한 호환성 단원을 참조하세요.

승인 규칙 및 승인 규칙 템플릿 이름	문자, 숫자, 마침표, 공백, 밑줄, 대시를 조합하여 사용할 수 있으며 길이는 1~100자 사이여야 합니다. 이름은 대/소문자를 구분합니다. 이름은 .git으로 끝날 수 없으며 다음 문자를 포함할 수 없습니다. ! ? @ # $ % ^ & * ( ) + = { } [ ] | \ / > < ~ ` ' " ; :
승인 규칙 내용 길이	3000자
승인 규칙 템플릿 설명 길이	1000자
승인 규칙 템플릿 대상 참조	100
승인 규칙 템플릿	의 1000 AWS 리전
풀 요청에 대한 승인 규칙	최대 30개까지 가능합니다. 이 중 최대 25개까지 승인 규칙 템플릿에서 생성할 수 있습니다.
승인 규칙 템플릿에서 생성된 풀 요청에 대한 승인 규칙	25

풀 요청에 대한 승인	200
승인 풀의 승인자	50
브랜치 이름	1~256자 사이의 문자 조합은 무엇이든 허용됩니다. 단, 16진수 문자 40자로 이루어진 브랜치 이름은 허용하지 않습니다. 브랜치 이름에는 다음과 같은 제한이 있습니다. • 슬래시(/) 또는 마침표(.)로 시작하거나 끝날 수 없음 • 단일 문자 @로 구성될 수 없음 • 두 개 이상의 연속 마침표(..), 슬래시(//) 또는 다음 문자 조합은 포함할 수 없음: @{ • 공백 또는 허용되지 않는 문자(? ^ * [\ ~ :)의 사용 브랜치 이름은 참조입니다. 브랜치 이름에 대한 대부분의 제한 사항은 Git 참조 표준에 근거합니다. 자세한 내용은 Git Internals 및 git-check-ref-format을 참조하세요.
주석 길이	최대 10,240자입니다.
에서 CodeCommit 리포지토리를 초기화하기 위한 압축된 아카이브 크기 AWS CloudFormation	압축 시 최대 20GB.
트리거에 대한 사용자 지정 데이터	1,000자로 제한되는 문자열 필드입니다. 동적 파라미터를 전달하는 데 사용할 수 없습니다.

콘솔에 표시	파일 또는 파일간 비교 내용은 다음과 같은 경우 콘솔에서 열람이 되지 않을 수 있습니다. • 파일이 2MB를 초과합니다. • 파일의 한 행에 25,000자가 넘게 포함되어 있습니다. • 비교하는 두 파일 간의 차이가 총 6,500행을 초과합니다.
콘솔에서 수행된 커밋의 이메일 주소	1~256자의 허용되는 문자 조합이면 됩니다. 이메일 주소가 유효하지 않습니다.

파일 경로	1~4,096자의 허용되는 문자 조합이면 됩니다. 파일 경로는 파일과 파일의 정확한 위치를 지정하는 명확한 이름이어야 합니다. 파일 경로는 깊이가 디렉터리 20개를 초과할 수 없습니다. 또한 파일 경로는 다음과 같아서는 안 됩니다.
	• 빈 문자열 포함
	• 상대적 파일 경로
	• 후행 슬래시나 백슬래시로 끝나는
	`/./`
	`/../`
	`//`
	• 문자 조합 포함
	
	파일 이름 및 경로는 정규화된 이름과 경로여야합니다. 로컬 컴퓨터의 파일 이름과 경로는 해당 운영 체제에 대한 표준을 따라야 합니다. CodeCommit 리포지토리의 파일에 경로를 지정할 때는 Amazon Linux에 대한 표준을 사용합니다.

파일 크기	CodeCommit 콘솔, API 또는 AWS CLI의 사용 시 개별 파일의 경우 최대 6MB.
Git BLOB 크기	최대 2GB **Note** 단일 커밋의 모든 파일 수 또는 총 파일 크기에 대한 제한은 없습니다. 단, 메타데이터는 6MB를 초과할 수 없고 단일 BLOB은 2GB를 초과할 수 없습니다.
커밋 시각화 도우미의 브랜치에 대한 그래프 표시	페이지당 35개. 단일 페이지에서 브랜치가 35개를 넘으면 그래프가 표시되지 않습니다.
커밋에 대한 메타데이터	CodeCommit 콘솔, API 또는 AWS CLI사용 시 커밋에 대한 통합 메타데이터(예: 작성자 정보, 날짜, 상위 커밋 목록 및 커밋 메시지 등의 조합)의 용량은 최대 20MB입니다. **Note** 단일 커밋의 모든 파일 수 또는 총 파일 크기에 대한 제한은 없습니다. 단, 메타데이터는 6MB를 초과할 수 없고, 개별 파일은 6MB를 초과할 수 없으며, 단일 BLOB은 2GB를 초과할 수 없습니다.
커밋에 포함된 파일 수	최대 100개.
열린 풀 요청 수	최 1,000개.
단일 푸시의 참조 수	생성, 삭제, 업데이트를 포함하여 최대 4,000개. 리포지토리의 최대 참조 수는 제한이 없습니다.

리포지토리 수	Amazon Web Services 계정 하나당 최대 5,000개. 이 제한은 변경할 수 있습니다. 자세한 내용은 AWS CodeCommit 엔드포인트 및 할당량 및 AWS 서비스 할당량을 참조하세요.
리포지토리의 트리거 수	최대 10개

리전	CodeCommit은 AWS 리전다음에서 사용할 수 있습니다. • 미국 동부(오하이오) • 미국 동부(버지니아 북부) • 미국 서부(캘리포니아 북부) • 미국 서부(오리건) • 유럽(아일랜드) • 유럽(런던) • 유럽(파리) • 유럽(프랑크푸르트) • 유럽(스톡홀름) • 유럽(밀라노) • 아프리카(케이프타운) • 이스라엘(텔아비브) • 아시아 태평양(도쿄) • 아시아 태평양(싱가포르) • 아시아 태평양(시드니) • 아시아 태평양(자카르타) • 중동(UAE) • 아시아 태평양(서울) • 아시아 태평양(오사카) • 아시아 태평양(뭄바이) • 아시아 태평양(하이데라바드) • 아시아 태평양(홍콩) • 남아메리카(상파울루) • 중동(바레인) • 캐나다(중부) • 중국(베이징) • 중국(닝샤) • AWS GovCloud(미국 서부)

	• AWS GovCloud(미국 동부) 자세한 내용은 리전 및 Git 연결 엔드포인트 단원을 참조하십시오.
리포지토리 설명	0~1,000자의 문자 조합이면 됩니다. 리포지토리 설명은 선택 사항입니다.
리포지토리 이름	문자, 숫자, 마침표, 밑줄, 대시를 조합하여 사용할 수 있으며 길이는 1~100자 사이여야 합니다. 이름은 대/소문자를 구분합니다. 리포지토리 이름은 .git으로 끝날 수 없으며 다음 문자를 포함할 수 없습니다. ! ? @ # $ % ^ & * ( ) + = { } [ ] | \ / > < ~ ` ' " ; :
리포지토리 태그 키 이름	유니코드 문자, 숫자, 공백 및 UTF-8 형식의 허용되는 문자 조합(1 - 128 문자) 허용되는 문자는 + - = . _ : / @입니다. 태그 키 이름은 고유해야 하며 각 키는 하나의 값만 가질 수 있습니다. 태그는 다음을 허용하지 않습니다. • aws:로 시작할 수 없음 • 공백으로만 이루어짐 • 공백으로 끝남 • 이모티콘 또는 다음 문자를 포함할 수 없음: ? ^ * [\ ~ ! # $ % & * ( ) > < | " ' ` [ ] { } ;

리포지토리 태그 값	유니코드 문자, 숫자, 공백 및 UTF-8 형식의 허용되는 문자 조합(1 - 256 문자) 허용되는 문자는 `+ - = . _ : / @`입니다. 키는 하나의 값만 가질 수 있지만 많은 키가 동일한 값을 가질 수 있습니다. 태그는 다음을 허용하지 않습니다. • 공백으로만 이루어짐 • 공백으로 끝남 • 이모티콘 또는 다음 문자를 포함할 수 없음: `? ^ * [ \ ~ ! # $ % & * ( ) > < | " ' ` [ ] { } ;`
리포지토리 태그	태그는 대/소문자를 구분합니다. 리소스당 최대 50개입니다. 16진수 문자 40자로 이루어진 태그 이름은 허용되지 않습니다.
트리거 이름	문자, 숫자, 마침표, 밑줄, 대시를 조합하여 사용할 수 있으며 길이는 1~100자 사이여야 합니다. 트리거 이름에 공백 또는 쉼표가 포함될 수 없습니다.
콘솔에서 수행된 커밋의 사용자 이름	1~1,024자의 허용되는 문자 조합이면 됩니다.

AWS CodeCommit 명령줄 참조

이 참조는 AWS CLI사용 방법을 파악하는 데 도움이 됩니다.

를 설치하고 구성하려면 AWS CLI

1. 로컬 시스템에서를 다운로드하여 설치합니다 AWS CLI. 이는 명령줄에서 CodeCommit과 상호 작용하기 위한 사전 조건입니다. AWS CLI 버전 2를 설치하는 것이 좋습니다. 의 최신 메이저 버전이며 모든 최신 기능을 AWS CLI 지원합니다. 에서 루트 계정, 페더레이션 액세스 또는 임시 자격 증명 사용을 AWS CLI 지원하는 유일한 버전의 입니다git-remote-codecommit.

   자세한 내용은 AWS 명령줄 인터페이스 설정 단원을 참조하십시오.

> **Note**
>
> CodeCommit은 AWS CLI 버전 1.7.38 이상에서만 작동합니다. 가장 좋은 방법은를 설치하거나 사용 가능한 최신 버전으로 업그레이드 AWS CLI 하는 것입니다. 설치 AWS CLI 한 버전을 확인하려면 aws --version 명령을 실행합니다.
> 의 이전 버전을 AWS CLI 최신 버전으로 업그레이드하려면 설치를 참조하세요 AWS Command Line Interface.

2. 이 명령을 실행하여에 대한 CodeCommit 명령 AWS CLI 이 설치되어 있는지 확인합니다.

```
aws codecommit help
```

이 명령은 CodeCommit 명령 목록을 반환합니다.

3. 다음과 같이 configure 명령을 사용하여 프로파일 AWS CLI 로를 구성합니다.

```
aws configure
```

메시지가 표시되면 CodeCommit과 함께 사용할 IAM 사용자의 AWS 액세스 키와 AWS 보안 액세스 키를 지정합니다. 또한와 같이 리포지토리 AWS 리전 가 있는를 지정해야 합니다`us-east-2`. 기본 출력 형식을 묻는 메시지가 표시되면 `json`을 지정합니다. 예를 들어, IAM 사용자의 프로필을 구성하는 경우에는 다음과 같이 합니다.

```
AWS Access Key ID [None]: Type your IAM user AWS access key ID here, and then press
 Enter
AWS Secret Access Key [None]: Type your IAM user AWS secret access key here, and
 then press Enter
Default region name [None]: Type a supported region for CodeCommit here, and then
 press Enter
Default output format [None]: Type json here, and then press Enter
```

에서 사용할 프로필을 생성하고 구성하는 방법에 대한 자세한 내용은 다음을 AWS CLI참조하세요.

- 명명된 프로파일
- 에서 IAM 역할 사용 AWS CLI
- 설정 명령
- 교체 자격 증명을 사용하여 AWS CodeCommit 리포지토리에 연결

리포지토리 또는 다른의 리소스에 연결하려면 기본 리전 이름으로 AWS CLI 를 재구성 AWS 리전 해야 합니다. CodeCommit에서 지원되는 기본 리전 이름은 다음과 같습니다.

- us-east-2
- us-east-1
- eu-west-1
- us-west-2
- ap-northeast-1
- ap-southeast-1
- ap-southeast-2
- ap-southeast-3
- me-central-1
- eu-central-1
- ap-northeast-2
- sa-east-1
- us-west-1
- eu-west-2
- ap-south-1
- ap-south-1
- ca-central-1
- us-gov-west-1
- us-gov-east-1
- eu-north-1
- ap-east-1
- me-south-1
- cn-north-1
- cn-northwest-1
- eu-south-1
- ap-northeast-3
- af-south-1

- il-central-1

CodeCommit 및에 대한 자세한 내용은 섹션을 AWS 리전참조하세요리전 및 Git 연결 엔드포인트. IAM, 액세스 키, 비밀 키에 대한 자세한 내용을 확인하려면 보안 인증 정보 얻는 방법 및 IAM 사용자의 액세스 키 관리를 참조하세요. AWS CLI 및 프로파일에 대한 자세한 내용은 명명된 프로파일을 참조하세요.

이용 가능한 모든 CodeCommit 명령의 목록을 보려면 다음 명령을 실행합니다.

```
aws codecommit help
```

CodeCommit 명령에 대한 정보를 보려면 다음 명령을 실행합니다. 여기서 *command-name*은 명령의 이름입니다(예: create-repository).

```
aws codecommit command-name help
```

AWS CLI의 명령에 대한 설명과 사용 예를 보려면 다음을 참조하세요.

- associate-approval-rule-template-with-repository
- batch-associate-approval-rule-template-with-repositories
- batch-disassociate-approval-rule-template-from-repositories
- batch-describe-merge-conflicts
- batch-get-commits
- batch-get-repositories
- create-approval-rule-template
- create-branch
- create-commit
- create-pull-request
- create-pull-request-approval-rule
- create-repository
- create-unreferenced-merge-commit
- delete-approval-rule-template
- delete-branch

- delete-comment-content
- delete-file
- delete-repository
- describe-merge-conflicts
- delete-pull-request-approval-rule
- describe-pull-request-events
- disassociate-pull-request-approval-rule-template-from-repository
- evaluate-pull-request-approval-rules
- get-approval-rule-template
- get-blob
- get-branch
- get-comment
- get-comment-reactions
- get-comments-for-compared-commit
- get-comments-for-pull-request
- get-commit
- get-differences
- get-merge-commit
- get-merge-conflicts
- get-merge-options
- get-pull-request
- get-pull-request-approval-states
- get-pull-request-override-state
- get-repository
- get-repository-triggers
- list-approval-rule-templates
- list-associated-approval-rule-templates-for-repository
- list-branches
- list-pull-requests
- list-repositories

- list-repositories-for-approval-rule-template
- list-tags-for-resource
- merge-branches-by-fast-forward
- merge-branches-by-squash
- merge-branches-by-three-way
- merge-pull-request-by-fast-forward
- merge-pull-request-by-squash
- merge-pull-request-by-three-way
- override-pull-request-approval-rules
- post-comment-for-compared-commit
- post-comment-for-pull-request
- post-comment-reply
- put-comment-reaction
- put-file
- put-repository-triggers
- tag-resource
- test-repository-triggers
- untag-resource
- update-approval-rule-template-content
- update-approval-rule-template-description
- update-approval-rule-template-name
- update-comment
- update-default-branch
- update-pull-request-approval-rule-content
- update-pull-request-approval-state
- update-pull-request-description
- update-pull-request-status
- update-pull-request-title
- update-repository-description
- update-repository-name

기본 Git 명령

Git을 사용하여 로컬 리포지토리에서, 그리고 로컬 리포지토리에 연결된 CodeCommit 리포지토리에서 작업할 수 있습니다.

다음은 자주 사용되는 Git 명령의 몇 가지 기본 예제입니다.

다른 옵션들에 대해서는 Git 설명서를 참조하세요.

주제

- 구성 변수
- 원격 리포지토리
- 커밋
- 브랜치
- Tags

구성 변수

모든 구성 변수를 나열합니다.	`git config --list`
로컬 구성 변수만 나열합니다.	`git config --local -l`
시스템 구성 변수만 나열합니다.	`git config --system -l`
글로벌 구성 변수만 나열합니다.	`git config --global -l`
지정된 구성 파일에 구성 변수를 설정합니다.	`git config [--local | --global | --system] variable-name variable-value`
기본 브랜치가 아직 없는 리포지토리에 대해 첫 커밋을 작성할 때 모든 로컬 리포지토리의 기본 브랜치 이름을 main으로 설정합니다.	git config --global init.defaultBranch main
구성 파일을 직접 편집합니다. 특정 구성 파일의 위치를 찾는 데에도 사용할 수 있습니다. 편집 모드를 종료하려면, 보통 `:q`(변경 내용을 저장하	`git config [--local | --global | --system] --edit`

지 않고 종료) 또는 `:wq`(변경 내용을 저장한 다음 종료)를 입력한 다음 Enter 키를 누릅니다.	

원격 리포지토리

CodeCommit 리포지토리에 연결할 준비를 하기 위해 로컬 리포지토리를 초기화합니다.	`git init`
로컬 리포지토리와 원격 리포지토리(예: CodeCommit 리포지토리) 간의 연결을 설정하는 데 사용할 수 있습니다. 로컬 리포지토리가 CodeCommit 리포지토리에 대해 지정한 별명과 CodeCommit 리포지토리에 지정된 URL을 사용합니다.	`git remote add` *`remote-name`* *`remote-url`*
로컬 컴퓨터에 있는 현재 폴더의 지정된 하위 폴더에서 지정된 URL에 CodeCommit 리포지토리의 복사본을 만들어 로컬 리포지토리를 생성합니다. 이 명령은 또한 복제된 CodeCommit 리포지토리의 각 브랜치에 대한 원격 추적 브랜치를 생성하며, 복제된 CodeCommit 리포지토리의 현재 기본 브랜치에서 파생된 초기 브랜치를 생성하고 체크아웃합니다.	`git clone` *`remote-url`* *`local-subfolder-name`*
로컬 리포지토리가 CodeCommit 리포지토리에 대해 사용하는 별명을 표시합니다.	`git remote`
로컬 리포지토리가 CodeCommit 리포지토리를 대상으로 가져오고 푸시하는 데 사용하는 별명과 URL을 표시합니다.	`git remote -v`
로컬 리포지토리에서 CodeCommit 리포지토리로 최종 커밋을 푸시합니다. 로컬 리포지토리가 CodeCommit 리포지토리와 지정 브랜치에 대해 지정한 별명을 사용합니다. 또한 푸시하는 동안	`git push -u` *`remote-name`* *`branch-name`*

로컬 리포지토리에 대한 업스트림 추적 정보를 설정합니다.	
업스트림 추적 정보가 설정된 후 로컬 리포지토리에서 CodeCommit 리포지토리로 최종 커밋을 푸시합니다.	`git push`
CodeCommit 리포지토리에서 로컬 리포지토리로 최종 커밋을 풀합니다. 로컬 리포지토리가 CodeCommit 리포지토리와 지정 브랜치에 대해 지정한 별명을 사용합니다.	`git pull` *`remote-name`* *`branch-name`*
업스트림 추적 정보가 설정된 후 CodeCommit 리포지토리에서 로컬 리포지토리로 최종 커밋을 푸시합니다.	`git pull`
CodeCommit 리포지토리에서 로컬 리포지토리의 연결을 끊습니다. 로컬 리포지토리가 CodeCommit 리포지토리에 대해 지정한 별명을 사용합니다.	`git remote rm` *`remote-name`*

커밋

로컬 리포지토리의 보류 중인 커밋에 추가되거나 추가되지 않은 내용을 표시합니다.	`git status`
로컬 리포지토리의 보류 중인 커밋에 추가되거나 추가되지 않은 내용을 간결한 형식으로 보여줍니다. (M= 수정됨, A = 추가됨, D = 삭제됨 등)	`git status -sb`
보류 중인 커밋과 로컬 리포지토리의 최신 커밋 사이의 변경 사항을 표시합니다.	`git diff HEAD`
로컬 리포지토리의 보류 중인 커밋에 특정 파일을 추가합니다.	`git add [`*`file-name-1 file-name-2 file-name-N`* `|` *`file-pattern`* `]`

새 파일, 수정 파일, 삭제된 파일을 모두 로컬 리포지토리의 보류 중인 커밋에 추가합니다.	`git add`
로컬 리포지토리의 보류 중인 커밋을 마무리하기 시작합니다. 커밋 메시지를 제공하는 편집기가 표시됩니다. 메시지가 입력되면 보류 중인 커밋이 완료됩니다.	`git commit`
커밋 메시지를 동시에 지정하는 것을 포함하여 로컬 리포지토리의 보류 중인 커밋을 마무리합니다.	`git commit -m "Some meaningful commit comment"`
로컬 리포지토리의 최근 커밋을 나열합니다.	`git log`
로컬 리포지토리의 최근 커밋을 그래프 형식으로 나열합니다.	`git log --graph`
로컬 리포지토리의 최근 커밋을 미리 정의된 요약 형식으로 나열합니다.	`git log --pretty=oneline`
로컬 리포지토리의 최근 커밋을 미리 정의된 요약 형식으로 그래프와 함께 나열합니다.	`git log --graph --pretty=oneline`
로컬 리포지토리의 최근 커밋을 사용자 지정 형식으로 그래프와 함께 나열합니다. (자세한 옵션은 Git 설명서의 기록 이해: 커밋을 참조하세요.)	`git log --graph --pretty=format:"%H (%h) : %cn : %ar : %s"`

브랜치

로컬 리포지토리의 모든 브랜치를 나열하며, 현재 브랜치 옆에 별표(*)를 표시합니다.	`git branch`
CodeCommit 리포지토리의 모든 기존 브랜치에 대한 정보를 로컬 리포지토리로 풀합니다.	`git fetch`

로컬 리포지토리의 브랜치와 로컬 리포지토리의 원격 추적 브랜치를 모두 나열합니다.	`git branch -a`
로컬 리포지토리의 원격 추적 브랜치만 나열합니다.	`git branch -r`
지정된 브랜치 이름을 사용하여 로컬 리포지토리에 새 브랜치를 생성합니다.	`git branch new-branch-name`
지정된 브랜치 이름을 사용하여 로컬 리포지토리의 다른 브랜치로 전환합니다.	`git checkout other-branch-name`
지정된 브랜치 이름을 사용하여 로컬 리포지토리에 새 브랜치를 만든 다음 해당 브랜치로 전환합니다.	`git checkout -b new-branch-name`
로컬 리포지토리가 CodeCommit 리포지토리에 대해 지정한 닉네임과 지정된 브랜치 이름을 사용하여 로컬 리포지토리에서 CodeCommit 리포지토리로 새 브랜치를 푸시합니다. 또한 푸시하는 동안 로컬 리포지토리의 브랜치에 대한 업스트림 추적 정보를 설정합니다.	`git push -u remote-name new-branch-name`
지정된 브랜치 이름을 사용하여 로컬 리포지토리에 새 브랜치를 생성합니다. 로컬 리포지토리가 CodeCommit 리포지토리에 대해 지정한 닉네임과 지정된 브랜치 이름을 사용하여 로컬 리포지토리의 새 브랜치를 CodeCommit 리포지토리의 기존 브랜치로 연결합니다.	`git branch --track new-branch-name remote-name /remote-branch-name`
로컬 리포지토리의 다른 브랜치에서 변경한 내용을 로컬 리포지토리의 현재 브랜치에 병합합니다.	`git merge from-other-branch-name`
병합되지 않은 작업이 포함되어 있지 않는 한 로컬 리포지토리에서 브랜치를 삭제합니다.	`git branch -d branch-name`

로컬 리포지토리가 CodeCommit 리포지토리에 대해 지정한 닉네임과 지정된 브랜치 이름을 사용하여 CodeCommit 리포지토리의 새 브랜치를 삭제합니다. (콜론(:) 사용에 주의하세요.)	`git push remote-name :branch-name`

Tags

로컬 리포지토리의 모든 태그를 나열합니다.	`git tag`
CodeCommit 리포지토리의 모든 태그를 로컬 리포지토리로 풀합니다.	`git fetch --tags`
로컬 리포지토리의 특정 태그에 대한 정보를 표시합니다.	`git show tag-name`
로컬 리포지토리에 “경량” 태그를 생성합니다.	`git tag tag-name commit-id-to-point-tag-at`
로컬 리포지토리가 CodeCommit 리포지토리에 대해 지정한 닉네임과 지정된 브랜치 이름을 사용하여 로컬 리포지토리의 특정 태그를 CodeCommit 리포지토리로 푸시합니다.	`git push remote-name tag-name`
로컬 리포지토리가 CodeCommit 리포지토리에 대해 지정한 닉네임을 사용하여 로컬 리포지토리의 모든 태그를 CodeCommit 리포지토리로 푸시합니다.	`git push remote-name --tags`
로컬 리포지토리에서 태그를 삭제합니다.	`git tag -d tag-name`
로컬 리포지토리가 CodeCommit 리포지토리에 대해 지정한 닉네임과 지정된 브랜치 이름을 사용하여 CodeCommit 리포지토리에서 태그를 삭제합니다. (콜론(:) 사용에 주의하세요.)	`git push remote-name :tag-name`

AWS CodeCommit 사용 설명서 문서 기록

다음 표에서는 CodeCommit 설명서의 주요 변경 사항에 대해 설명합니다. 이 설명서에 대한 업데이트 알림을 받으려면 RSS 피드를 구독하면 됩니다.

- API 버전: 2015-04-13

변경 사항	설명	날짜
AWS CodeCommit 는 더 이상 신규 고객이 사용할 수 없습니다.	AWS CodeCommit 는 더 이상 신규 고객이 사용할 수 없습니다. AWS CodeCommit 의 기존 고객은 정상적으로 서비스를 계속 이용할 수 있습니다. 자세히 알아보기	2024년 7월 25일
자격 증명 도우미에 대한 문제 해결 정보 업데이트	불완전하거나 잘못 구성된 .gitconfig 파일로 인한 오류 문제 해결에 대한 정보가 추가되었습니다.	2024년 7월 9일
에 대한 문제 해결 정보 업데이트 `git-remote-codecommit`	외부 관리형 환경의 오류 문제 해결에 대한 추가 정보가 추가되었습니다.	2024년 7월 9일
에 대한 참조 제거 AWS Toolkit for Eclipse	AWS Toolkit for Eclipse 수명이 종료되어 더 이상 지원되지 않습니다. 이 도구 키트에 대한 참조가 제거되었습니다.	2024년 7월 9일
이제 CodeCommit이 고객 관리형 키 사용을 지원함	이제 고객 관리형 키 또는를 사용하여 AWS 관리형 키 리포지토리의 데이터를 암호화하고 해독할 수 있습니다. 자세한 내용은 AWS KMS 및 암호화, 리포지토리 생성 및 리포지토리 설정 변경을 참조하세요.	2023년 12월 21일

CodeCommit을 이스라엘(텔아비브)에서 사용 가능	CodeCommit은 이제 이스라엘(텔아비브)에서 사용할 수 있습니다. 자세한 내용은 리전 및 Git 연결 엔드포인트를 참조하세요.	2023년 8월 28일
CodeCommit의 관리형 정책에 대한 변경 사항	AWSCodeCommitPowerUser와 AWSCodeCommitFullAccess 정책이 추가 권한으로 업데이트되었습니다. 자세한 내용은 AWS 관리형 정책에 대한 CodeCommit 업데이트를 참조하세요.	2023년 5월 16일
CodeCommit은 이제 세 가지 추가 AWS 리전	CodeCommit은 이제 AWS 리전아시아 태평양(자카르타), 중동(UAE) 및 아시아 태평양(하이데라바드)의 세 가지 추가에서 사용할 수 있습니다. 자세한 내용은 리전 및 Git 연결 엔드포인트를 참조하세요.	2023년 2월 28일
CodeCommit을 아프리카(케이프타운)에서 사용 가능	CodeCommit은 이제 AWS 리전아프리카(케이프타운)라는 추가에서 사용할 수 있습니다. 자세한 내용은 리전 및 Git 연결 엔드포인트를 참조하세요.	2021년 9월 15일
CodeCommit의 관리형 정책에 대한 변경 사항	이제 CodeCommit의 AWS 관리형 정책 업데이트에 대한 세부 정보를 사용할 수 있습니다. 자세한 내용은 AWS 관리형 정책에 대한 CodeCommit 업데이트를 참조하세요.	2021년 8월 18일

CodeCommit을 아시아 태평양 (오사카)에서 사용 가능	이제 AWS 리전아시아 태평양 (오사카)에서 CodeCommit을 추가로 사용할 수 있습니다. 자세한 내용은 리전 및 Git 연결 엔드포인트를 참조하세요.	2021년 4월 14일
AWS CloudFormation 및는 CodeCommit의 기본 브랜치에 대한 이름 지정 동작을 AWS 클라우드 개발 키트 (AWS CDK) 변경합니다.	코드의 초기 커밋과 AWS CDK 함께 AWS CloudFormation 또는를 사용하여 생성된 리포지토리는 이제 기본 브랜치 이름인 main을 사용합니다. 이 변경 사항은 기존 리포지토리나 브랜치에 영향을 주지 않습니다. 로컬 Git 클라이언트를 사용하여 초기 커밋을 생성하는 고객은 해당 Git 클라이언트의 구성을 따르는 기본 브랜치 이름을 갖게 됩니다. 자세한 내용은 를 사용하여 CodeCommit 리소스 생성을 AWS CloudFormation 참조하세요.	2021년 3월 4일
CodeCommit이 기본 브랜치에 대한 이름 지정 동작 변경	2021년 1월 19일부로, CodeCommit 리포지토리에 대한 초기 커밋으로 생성되는 기본 브랜치 이름은 main입니다. 이 변경 사항은 기존 리포지토리 또는 브랜치에 영향을 주지 않습니다. 로컬 Git 클라이언트를 사용하여 초기 커밋을 생성하는 고객은 해당 Git 클라이언트의 구성을 따르는 기본 브랜치 이름을 갖게 됩니다. 자세한 내용은 브랜치로 작업하기, 커밋 생성, 브랜치 설정 변경을 참조하세요.	2021년 1월 19일

CodeCommit을 유럽(밀라노)에서 사용 가능	이제 AWS 리전유럽(밀라노)에서 CodeCommit을 추가로 사용할 수 있습니다. 자세한 내용은 리전 및 Git 연결 엔드포인트를 참조하세요.	2020년 9월 16일
CodeCommit이 댓글의 이모티콘 반응에 대한 지원 추가	CodeCommit은 이제 이모티콘으로 다른 사용자의 댓글에 반응할 수 있도록 지원합니다. 자세한 내용은 Comment on a Commit 및 Review a Pull Request를 참조하세요.	2020년 6월 24일
CodeCommit을 중국(베이징), 중국(닝샤)에서 사용 가능	CodeCommit은 이제 중국(베이징)과 중국(닝샤) AWS 리전의 두 가지 추가에서 사용할 수 있습니다. 자세한 내용은 리전 및 Git 연결 엔드포인트를 참조하세요.	2020년 4월 23일

CodeCommit이 git-remote-codecommit에 대한 지원 추가	CodeCommit은 Git을 수정하는 유틸리티인 git-remote-codecommit을 사용하는 HTTPS를 통해 CodeCommit 리포지토리에 대한 연결을 지원합니다. 이는 CodeCommit 리포지토리에 대한 페더레이션 액세스 연결 또는 임시 액세스 연결에 권장되는 접근 방식입니다. git-remote-codecommit을 IAM 사용자로 사용할 수도 있습니다. git-remote-codecommit은 사용자에 대한 Git 보안 인증 정보를 설정할 필요가 없습니다. 자세한 내용은 git-remote-codecommit을 AWS CodeCommit 사용하여에 대한 HTTPS 연결을 위한 설정 단계를 참조하세요.	2020년 3월 4일
CodeCommit이 세션 태그 지원	CodeCommit에서는 세션 태그의 사용을 지원합니다. 세션 태그는 IAM 역할을 맡거나, 임시 보안 인증 정보를 사용하거나, AWS Security Token Service (AWS STS)에서 사용자를 연동할 때 전달하는 키-값 페어 속성입니다. 이러한 태그에 제공된 정보를 통해 누가 변경을 수행했는지 또는 이벤트를 유발했는지 보다 쉽게 파악할 수 있습니다. 자세한 내용은 CodeCommit 모니터링 및 CodeCommit에서 ID 정보 제공을 위한 태그 사용을 참조하세요.	2019년 12월 19일

CodeCommit을 아시아 태평양(홍콩)에서 사용 가능	CodeCommit은 이제 아시아 태평양(홍콩)에서 사용할 수 있습니다. Git 연결 엔드포인트에 대한 자세한 내용은 리전을 참조하세요.	2019년 12월 11일
CodeCommit이 Amazon CodeGuru Reviewer 지원	CodeCommit은 프로그램 분석 및 기계 학습을 사용하여 Java 또는 Python 코드에서 일반적인 문제를 감지하고 권장 수정 사항을 제공하는 자동화된 코드 검토 서비스인 Amazon CodeGuru Reviewer를 지원합니다. 자세한 내용은 리포지토리를 Amazon CodeGuru Reviewer와 연결 및 연결 해제하기, 풀 요청으로 작업하기를 참조하세요.	2019년 12월 3일
CodeCommit이 승인 규칙 지원	이제 승인 규칙을 사용하여 리포지토리 사이에서 개발 워크플로를 사용자 지정하여 여러 브랜치에서 풀 요청에 대한 적절한 수준의 승인 및 제어 권한을 보유하도록 할 수 있습니다. 자세한 내용은 승인 규칙 템플릿 작업 및 풀 요청 작업을 참조하세요.	2019년 11월 20일
CodeCommit이 알림 규칙 지원	이제 알림 규칙을 사용하여 사용자에게 리포지토리의 중요 변경 사항을 알릴 수 있습니다. 이 기능은 2019년 11월 5일 이전에 생성된 알림을 대체합니다. 자세한 내용은 알림 규칙 생성을 참조하세요.	2019년 11월 5일

CodeCommit을 중동(바레인)에서 사용 가능	CodeCommit은 이제 중동(바레인)에서 사용할 수 있습니다. Git 연결 엔드포인트에 대한 자세한 내용은 리전을 참조하세요.	2019년 10월 30일
CodeCommit이 여러 커밋에 대한 정보를 검색하기 위한 지원 추가	AWS CLI에서 batch-get-commits 명령을 사용하여 여러 커밋에 대한 정보를 가져올 수 있습니다. 자세한 내용은 커밋 세부 정보 보기를 참조하세요.	2019년 8월 15일
CodeCommit을 유럽(스톡홀름)에서 사용 가능	CodeCommit은 이제 유럽(스톡홀름)에서 사용할 수 있습니다. Git 연결 엔드포인트에 대한 자세한 내용은 리전을 참조하세요.	2019년 7월 31일
CodeCommit이 CodeCommit 콘솔에서 리포지토리에 태그 지정을 하기 위한 지원 추가	이제 CodeCommit 콘솔에서 AWS 리소스를 관리하는 데 도움이 되는 리포지토리의 태그를 추가, 관리 및 제거할 수 있습니다. 자세한 내용은 리포지토리 태그 지정을 참조하세요.	2019년 7월 2일
CodeCommit이 추가 Git 병합 전략에 대한 지원 추가	이제 CodeCommit에서 풀 요청 병합 시 Git 병합 전략 중에서 선택할 수 있습니다. 또한 CodeCommit 콘솔 내 병합 충돌을 해결할 수 있습니다. 자세한 내용은 풀 요청 작업을 참조하세요.	2019년 6월 10일

CodeCommit은 AWS GovCloud(미국 동부)에서 사용할 수 있습니다.	이제 CodeCommit in AWS GovCloud(미국 동부)를 사용할 수 있습니다. Git 연결 엔드포인트에 대한 자세한 내용은 리전을 참조하세요.	2019년 5월 31일
CodeCommit이 리포지토리 태그 지정에 대한 지원 추가	이제 리포지토리에 대한 태그를 추가, 관리 및 제거하여 AWS 리소스를 관리할 수 있습니다. 자세한 내용은 리포지토리 태그 지정을 참조하세요.	2019년 5월 30일
콘솔에서 리소스 찾기	이제 리포지토리, 빌드 프로젝트, 배포 애플리케이션 및 파이프라인과 같은 리소스를 신속하게 검색할 수 있습니다. 리소스로 이동을 선택하거나 / 키를 누른 후 리소스 이름을 입력합니다. 자세한 내용은 CodeCommit 자습서를 참조하세요.	2019년 5월 14일
CodeCommit은 AWS GovCloud(미국 서부)에서 사용할 수 있습니다.	이제 CodeCommit in AWS GovCloud(미국 서부)를 사용할 수 있습니다. Git 연결 엔드포인트에 대한 자세한 내용은 리전을 참조하세요.	2019년 4월 18일
CodeCommit이 Amazon VPC 엔드포인트에 대한 지원 추가	이제 VPC와 CodeCommit 간에 프라이빗 연결을 설정할 수 있습니다. 자세한 내용은 인터페이스 VPC 엔드포인트와 함께 CodeCommit 사용하기를 참조하세요.	2019년 3월 7일

CodeCommit이 새 API 추가	CodeCommit이 커밋 생성을 위한 API를 추가했습니다. 자세한 내용은 커밋 생성을 참조하세요.	2019년 2월 20일
내용 업데이트	이 안내서의 내용에는 마이너 수정과 추가적인 문제 해결 지침이 업데이트되어 있습니다.	2019년 1월 2일
내용 업데이트	이 안내서의 내용이 새로운 CodeCommit 콘솔 환경을 지원하도록 업데이트되었습니다.	2018년 10월 30일
CodeCommit 및 Federal Information Processing Standard(FIPS)	CodeCommit이 일부 리전에서 Federal Information Processing Standard(FIPS) Publication 140-2 정부 표준에 대한 지원을 추가했습니다. FIPS 및 FIPS 엔드포인트에 대한 자세한 내용은 Federal Information Processing Standard(FIPS) 140-2 개요를 참조하세요. Git 연결 엔드포인트에 대한 자세한 내용은 리전을 참조하세요.	2018년 10월 25일
CodeCommit이 세 가지 API 추가	CodeCommit이 파일 작업을 지원하기 위해 세 가지 API를 추가했습니다. Git 연결 엔드포인트에 대한 자세한 내용은 개별 파일 작업을 위한 권한 및 AWS CodeCommit API 참조를 참조하세요.	2018년 9월 27일
RSS 피드를 통해 CodeCommit 설명서 기록 알림 이용 가능	이제 RSS 피드를 구독하면 CodeCommit 설명서의 업데이트에 대한 알림을 받을 수 있습니다.	2018년 6월 29일

이전 업데이트

다음 표에서는 2018년 6월 29일 이전에 설명서에서 변경된 중요 사항에 대해 설명합니다.

변경 사항	설명	변경 날짜
새 주제	브랜치에 대한 푸시 및 병합 제한 주제가 추가되었습니다. CodeCommit 권한 참조 주제가 업데이트되었습니다.	2018년 5월 16일
새로운 섹션	AWS CodeCommit 리포지토리의 파일 작업 섹션이 추가되었습니다. CodeCommit 권한 참조 및 시작하기 AWS CodeCommit 주제가 업데이트되었습니다.	2018년 2월 21일
새 주제	역할을 사용하여 AWS CodeCommit 리포지토리에 대한 교차 계정 액세스 구성 주제가 추가되었습니다.	2018년 2월 21일
새 주제	AWS Cloud9 와 통합 AWS CodeCommit 주제가 추가되었습니다. 제품 및 서비스 통합 주제가에 대한 정보로 업데이트되었습니다 AWS Cloud9.	2017년 12월 1일
새로운 섹션	AWS CodeCommit 리포지토리에서 풀 요청 작업 섹션이 추가되었습니다. 풀 요청 및 주석 달기를 위한 권한에 대한 정보를 포함하도록 AWS CodeCommit에 대한 인증 및 액세스 제어 섹션이 업데이트되었습니다. 업데이트된 관리형 정책 명령문도 포함합니다.	2017년 11월 20일
업데이트된 주제	제품 및 서비스 통합 주제가 특정 고객에 대한 링크를 포함하도록 업데이트되었습니다. 여기서 특정 고객은 CodeCommit 리포지토리의 변경에 따라 Amazon CloudWatch Events를 사용하여 파이프라인을 시작하기 위해 기존 파이프라인을 업데이트하려는 고객을 말합니다.	2017년 10월 11일
새로운 주제	AWS CodeCommit에 대한 인증 및 액세스 제어 섹션이 추가되었습니다. 액세스 권한 참조 주제를 대체합니다.	2017년 9월 11일
업데이트된 주제	트리거 구성의 변경 내용을 반영하여 리포지토리 트리거 관리 단원을 업데이트했습니다. 탐색 모음의 변경 내용을 반영하여 설명서 전체의 주제와 이미지를 업데이트했습니다.	2017년 8월 29일

변경 사항	설명	변경 날짜
새 주제	사용자 기본 설정으로 작업 주제가 추가되었습니다. 태그 세부 정보 보기 주제가 업데이트되었습니다. 제품 및 서비스 통합 주제가 Amazon CloudWatch Events와의 통합에 대한 정보를 포함하도록 업데이트되었습니다.	2017년 8월 3일
새로운 주제	Eclipse를 AWS CodeCommit 및 Visual Studio를와 통합 AWS CodeCommit 주제와 통합이 추가되었습니다.	2017년 6월 29일
업데이트된 주제	CodeCommit은 이제 새로 추가된 두 개의 리전인 아시아 태평양(뭄바이), 캐나다(중부)에서 사용할 수 있습니다. 리전 및 Git 연결 엔드포인트 주제가 업데이트되었습니다.	2017년 6월 29일
업데이트된 주제	CodeCommit은 이제 새로 추가된 네 개의 리전인 아시아 태평양(서울), 남아메리카(상파울루), 미국 서부(캘리포니아 북부), 유럽(런던)에서 사용할 수 있습니다. 리전 및 Git 연결 엔드포인트 주제가 업데이트되었습니다.	2017년 6월 6일
업데이트된 주제	CodeCommit은 이제 새로 추가된 네 개의 리전인 아시아 태평양(도쿄), 아시아 태평양(싱가포르), 아시아 태평양(시드니), 유럽(프랑크푸르트)에서 사용할 수 있습니다. 리전 및 Git 연결 엔드포인트 주제가 CodeCommit의 Git 연결 엔드포인트 및 지원 리전에 대한 정보를 제공하도록 업데이트되었습니다.	2017년 5월 25일
새 주제	브랜치 비교 및 병합 주제가 추가되었습니다. 브랜치 작업 섹션의 내용이 CodeCommit 콘솔을 사용한 리포지토리의 브랜치 작업에 대한 정보가 포함되도록 업데이트되었습니다.	2017년 5월 18일
새 주제	커밋 비교 주제에는 커밋 비교에 대한 정보가 추가되었습니다. 리포지토리, 커밋 및 브랜치 작업에 대한 사용 설명서의 구조가 업데이트되었습니다.	2017년 3월 28일

변경 사항	설명	변경 날짜
업데이트된 주제	커밋 세부 정보 보기 주제에는 콘솔에서 커밋과 그 상위 항목 간의 차이점을 보는 방법과 get-differences 명령을 통해 AWS CLI를 사용하여 커밋 간의 차이점을 보는 방법에 대한 정보가 업데이트되었습니다.	2017년 1월 24일
새 주제	를 사용하여 AWS CodeCommit API 호출 로깅 AWS CloudTrail 주제에는 AWS CloudFormation을 사용하여 CodeCommit에 연결하는 로깅 정보가 추가되었습니다.	2017년 1월 11일
새 주제	Git 보안 인증 정보를 사용하는 HTTPS 사용자의 경우 주제에는 HTTPS를 통한 Git 보안 인증 정보를 사용하여 CodeCommit에 연결하는 설정 정보가 추가되었습니다.	2016년 12월 22일
업데이트된 주제	통합에 대한 정보를 포함하도록 제품 및 서비스 통합 주제가 업데이트되었습니다 AWS CodeBuild.	2016년 5월 12일
업데이트된 주제	CodeCommit은 이제 다른 리전인 유럽(아일랜드)에서 사용할 수 있습니다. 리전 및 Git 연결 엔드포인트 주제가 CodeCommit의 Git 연결 엔드포인트 및 지원 리전에 대한 정보를 제공하도록 업데이트되었습니다.	2016년 11월 16일
업데이트된 주제	CodeCommit은 이제 다른 리전인 미국 서부(오레곤)에서 사용할 수 있습니다. 리전 및 Git 연결 엔드포인트 주제가 CodeCommit의 Git 연결 엔드포인트 및 지원 리전에 대한 정보를 제공하도록 업데이트되었습니다.	2016년 11월 14일
새 주제	Lambda 함수를 위한 트리거 생성 주제가 Lambda 함수 생성의 일부로 CodeCommit 트리거를 생성하는 기능이 반영되도록 업데이트되었습니다. 이렇게 단순화된 프로세스는 트리거 생성을 간소화하고, CodeCommit이 Lambda 함수를 간접 호출하는 데 필요한 권한을 사용하여 트리거를 자동으로 구성합니다. 기존 Lambda 함수를 위한 트리거 생성 주제가 CodeCommit 콘솔에서 기존 Lambda 함수용 트리거 생성에 대한 정보를 포함하도록 추가되었습니다.	2016년 10월 19일

변경 사항	설명	변경 날짜
새 주제	CodeCommit은 이제 다른 지역인 미국 동부(오하이오)에서 사용할 수 있습니다. 리전 및 Git 연결 엔드포인트 주제가 CodeCommit의 Git 연결 엔드포인트 및 지원 리전에 대한 정보를 제공하도록 추가되었습니다.	2016년 10월 17일
주제 업데이트	통합에 대한 정보를 포함하도록 제품 및 서비스 통합 주제가 업데이트되었습니다 AWS Elastic Beanstalk.	2016년 10월 13일
주제 업데이트	통합에 대한 정보를 포함하도록 제품 및 서비스 통합 주제가 업데이트되었습니다 AWS CloudFormation.	2016년 10월 6일
주제 업데이트	Windows에서 SSH 연결 주제에는 PuTTY 도구 대신 Windows에서 SSH 연결용 Bash 에뮬레이터 사용에 대한 지침을 제공하도록 개정되었습니다.	2016년 9월 29일
주제 업데이트	커밋 세부 정보 보기 및 CodeCommit 시작하기 주제가 CodeCommit 콘솔에서 커밋 시각화에 대한 정보를 포함하도록 업데이트되었습니다. 할당량주제에는 단일 푸시에서 허용되는 참조 수의 증가로 업데이트되었습니다.	2016년 9월 14일
주제 업데이트	커밋 세부 정보 보기 및 CodeCommit 시작하기 주제가 CodeCommit 콘솔에서 커밋 이력 보기에 대한 정보를 포함하도록 업데이트되었습니다.	2016년 7월 28일
새로운 주제	Git 리포지토리를 로 마이그레이션 AWS CodeCommit 및 로컬 또는 버전이 지정되지 않은 콘텐츠를 로 마이그레이션 AWS CodeCommit 주제가 추가되었습니다.	2016년 6월 29일
주제 업데이트	문제 해결 및 자격 AWS CLI 증명 헬퍼를 사용하여 Windows에서 HTTPS 연결 주제에 대해 일부 업데이트가 진행되었습니다.	2016년 6월 22일
주제 업데이트	제품 및 서비스 통합 및 액세스 권한 참조 주제가 CodePipeline과의 통합 정보를 포함하도록 업데이트되었습니다.	2016년 4월 18일

변경 사항	설명	변경 날짜
새로운 주제	리포지토리 트리거 관리 섹션이 추가되었습니다. 새로운 주제에는 정책 및 코드 샘플을 포함한 트리거 작성, 편집 및 삭제 방법에 대한 예제가 포함되었습니다.	2016년 3월 7일
새 주제	제품 및 서비스 통합 주제가 추가되었습니다. 문제 해결에 대한 일부 내용이 업데이트되었습니다.	2016년 3월 7일
주제 업데이트	MD5 서버 지문 외에도 CodeCommit용 SHA256 서버 지문이 Linux, macOS, Unix에서 SSH 연결 및 Windows에서 SSH 연결에 추가되었습니다.	2015년 12월 9일
새 주제	리포지토리에서 파일 검색 주제가 추가되었습니다. 새로운 사안이 문제 해결에 추가되었습니다. 사용 설명서 전체에서 일부 개선 및 수정이 이루어졌습니다.	2015년 10월 5일
새 주제	를 사용하지 않는 SSH 사용자의 경우 AWS CLI 주제가 추가되었습니다. 설정 섹션의 주제가 간소화되었습니다. 사용자가 운영 체제 및 기본 프로토콜에 따라 수행할 단계를 결정하는 데 도움이 되는 지침이 제공되었습니다.	2015년 8월 5일
주제 업데이트	SSH와 Linux, macOS Unix: Git과 CodeCommit에 사용되는 퍼블릭 키와 프라이빗 키를 설정합니다. 및 3단계: Git과 CodeCommit에 사용되는 퍼블릭 키와 프라이빗 키 설정의 SSH 키 ID 단계에 명확성 및 예제가 추가되었습니다.	2015년 7월 24일
주제 업데이트	3단계: Git과 CodeCommit에 사용되는 퍼블릭 키와 프라이빗 키 설정의 단계가 IAM의 문제와 퍼블릭 키 파일을 저장하는 문제를 해결하도록 업데이트되었습니다.	2015년 7월 22일
주제 업데이트	탐색 지원으로 문제 해결이 업데이트되었습니다. 자격 증명 키 체인 문제에 대한 추가 문제 해결 정보가 추가되었습니다.	2015년 7월 20일
주제 업데이트	AWS Key Management Service 권한에 대한 자세한 내용은 AWS KMS 및 암호화 및 액세스 권한 참조 주제에 추가되었습니다.	2015년 7월 17일

변경 사항	설명	변경 날짜
주제 업데이트	AWS Key Management Service문제 해결에 대한 정보가 실린 또 다른 섹션이 문제 해결에 추가되었습니다.	2015년 7월 10일
초기 릴리스	CodeCommit 사용자 설명서의 최초 릴리스입니다.	2015년 7월 9일

AWS 용어집

최신 AWS 용어는 AWS 용어집 참조의 AWS 용어집을 참조하세요.