aws

사용 설명서

AWS CodeBuild



API 버전 2016-10-06

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

AWS CodeBuild: 사용 설명서

Copyright © 2026 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

AWS CodeBuild란 무엇인가요? 1
에 가입 AWS 계정 1
CodeBuild 실행 방법 1
CodeBuild 요금 3
CodeBuild는 어떻게 시작할 수 있나요? 3
개념 3
CodeBuild 작동 방식 3
다음 단계 5
시작하기 6
콘솔을 사용하여 시작하기 6
1단계: 소스 코드 생성 7
2단계: buildspec 파일 생성 10
3단계: 두 개의 S3 버킷 생성 12
4단계: 소스 코드 및 buildspec 파일 업로드 13
5단계: 빌드 프로젝트 생성 14
6단계: 빌드 실행 16
7단계: 요약된 빌드 정보 보기 16
8단계: 자세한 빌드 정보 보기 17
9단계: 빌드 출력 아티팩트 가져오기 18
10단계: S3 버킷 삭제 19
마무리 19
AWS CLI사용 시작하기 20
1단계: 소스 코드 생성 21
2단계: buildspec 파일 생성 23
3단계: 두 개의 S3 버킷 생성 26
4단계: 소스 코드 및 buildspec 파일 업로드 26
5단계: 빌드 프로젝트 생성 27
6단계: 빌드 실행 31
7단계: 요약된 빌드 정보 보기 33
8단계: 자세한 빌드 정보 보기 36
9단계: 빌드 출력 아티팩트 가져오기 38
10단계: S3 버킷 삭제 39
마무리 40
사용 사례 기반 샘플 41

교차 서비스 샘플 42
Amazon ECR 샘플 43
Amazon EFS 샘플 49
AWS CodePipeline 샘플 55
AWS Config 샘플 66
빌드 알림 샘플 67
빌드 배지 샘플 82
활성화된 빌드 배지를 사용하여 빌드 프로젝트 생성 83
AWS CodeBuild 빌드 배지 액세스 86
CodeBuild 빌드 배지 게시 86
CodeBuild 배지 상태 86
테스트 보고서 샘플 87
테스트 보고서 샘플 실행 87
CodeBuild용 Docker 샘플 94
도커 사용자 지정 이미지 샘플 94
Docker 이미지 빌드 서버 샘플 97
Windows Docker 빌드 샘플 100
'Amazon ECR에 Docker 이미지 게시' 샘플 102
AWS Secrets Manager 샘플이 포함된 프라이빗 레지스트리 111
빌드 출력을 S3 버킷에서 호스팅 115
다중 입력 및 출력 샘플 118
여러 입력 및 출력으로 빌드 프로젝트 생성 118
소스 없이 빌드 프로젝트 생성 121
buildspec 파일 샘플의 런타임 버전 122
buildspec 파일의 런타임 버전 업데이트 123
런타임 2개 지정 127
소스 버전 샘플 131
커밋 ID를 사용하여 GitHub 리포지토리 버전 지정 132
참조 및 커밋 ID를 사용하여 GitHub 리포지토리 버전 지정 134
타사 소스 리포지토리 샘플 135
Bitbucket 샘플 실행 135
GitHub Enterprise Server 샘플 실행 140
GitHub pull 요청 및 웹후크 필터 샘플 실행 146
자습서: 인증서 스토리지에 S3를 사용하여 CodeBuild에서 Fastlane을 사용한 Apple 코드 서명 150

자습서: 인증서 스토리지에 GitHub를 사용하여 CodeBuild에서 Fastlane을 사용한 Apple 코드 서명 156
빌드 시 아티팩트 이름 설정 162
Windows 샘플 실행 165
Windows 샘플 실행 165
디렉터리 구조 166
F# 및 .NET Framework 166
Visual Basic 및 .NET Framework 167
파일 167
F# 및 .NET Framework 167
Visual Basic 및 .NET Framework 172
빌드 계획 186
buildspec 참조 189
buildspec 파일 이름 및 스토리지 위치 189
buildspec 구문 190
버전 193
run-as 193
env 193
proxy 198
단계 199
보고서 203
artifacts 205
cache 210
buildspec 예제 212
buildspec 버전 215
배치 buildspec 참조 216
일괄 216
batch/build-graph 217
batch/build-list 219
batch/build-matrix 222
batch/build-fanout 223
빌드 환경 참조 226
CodeBuild가 제공하는 도커 이미지 227
현재 Docker 이미지 목록 가져오기 227
EC2 컴퓨팅 이미지 227
Lambda 컴퓨팅 이미지 230

더 이상 사용되지 않는 CodeBuild 이미지 ... 236
사용 가능한 런타임 ... 237
실행 시간 버전 ... 261
빌드 환경 컴퓨팅 모드 및 유형 ... 265
컴퓨팅 정보 ... 265
예약 용량 환경 유형 정보 ... 266
온디맨드 환경 유형 정보 ... 329
빌드 환경의 쉘 및 명령 ... 341
빌드 환경의 환경 변수 ... 342
빌드 환경의 배경 작업 ... 347
빌드 프로젝트 ... 349
빌드 프로젝트 생성 ... 349
사전 조건 ... 350
빌드 프로젝트 만들기(콘솔) ... 350
빌드 프로젝트 생성(AWS CLI) ... 370
빌드 프로젝트(AWS SDKs) 생성 ... 389
빌드 프로젝트 생성(CloudFormation) ... 389
알림 규칙 생성 ... 389
빌드 프로젝트 설정 변경 ... 392
빌드 프로젝트 설정 변경(콘솔) ... 392
빌드 프로젝트 설정 변경(AWS CLI) ... 414
빌드 프로젝트의 설정 변경(AWS SDKs) ... 415
다중 액세스 토큰 ... 415
1단계: Secrets Manager 보안 암호 또는 AWS CodeConnections 연결 생성 ... 416
2단계: Secrets Manager 보안 암호에 대한 IAM 역할 액세스 권한을 CodeBuild 프로젝트에 부여 ... 416
3단계: Secrets Manager 또는 AWS CodeConnections 토큰 구성 ... 418
추가 설정 옵션 ... 423
에서 빌드 프로젝트 삭제 ... 426
빌드 프로젝트 삭제(콘솔) ... 426
빌드 프로젝트 삭제(AWS CLI) ... 427
빌드 프로젝트(AWS SDKs) 삭제 ... 427
퍼블릭 빌드 프로젝트 URL 가져오기 ... 427
빌드 프로젝트 공유 ... 428
프로젝트 공유 ... 429
관련 서비스 ... 432

공유 프로젝트 액세스 432
공유 프로젝트의 공유 해제 432
공유 프로젝트 식별 433
공유 프로젝트 권한 433
빌드 프로젝트 태그 지정 433
프로젝트에 태그 추가 434
프로젝트의 태그 보기 436
프로젝트의 태그 편집 437
프로젝트에서 태그 제거 438
실행기 사용 439
GitHub Actions 439
GitLab 실행기 462
Buildkite 실행기 476
웹후크 사용 498
webhook 사용 모범 사례 498
Bitbucket Webhook 이벤트 499
GitHub 글로벌 및 조직 웹후크 513
GitHub 수동 웹후크 519
GitHub Webhook 이벤트 520
GitLab 그룹 웹후크 537
GitLab 수동 웹후크 542
GitLab 웹후크 이벤트 543
Buildkite 수동 웹후크 558
풀 요청 설명 승인 559
빌드 프로젝트 세부 정보 보기 565
빌드 프로젝트 세부 정보 보기(콘솔) 565
빌드 프로젝트 세부 정보 보기(AWS CLI) 566
빌드 프로젝트의 세부 정보(AWS SDKs) 보기 568
빌드 프로젝트 이름 보기 568
빌드 프로젝트 이름 목록 보기(콘솔) 568
빌드 프로젝트 이름 목록 보기(AWS CLI) 569
빌드 프로젝트 이름(AWS SDKs) 목록 보기 570
빌드 571
빌드를 수동으로 실행 572
로컬에서 빌드 실행 572
빌드 실행(콘솔) 576

빌드 실행(AWS CLI) ... 577
배치 빌드 실행(AWS CLI) ... 583
빌드 실행 자동 시작(AWS CLI) ... 584
빌드 실행 자동 중지(AWS CLI) ... 585
빌드 실행(AWS SDKs) ... 586
Lambda 컴퓨팅에서 빌드 실행 ... 586
AWS Lambda에서 실행되는 큐레이팅된 런타임 환경 도커 이미지에는 어떤 도구와 런타임이 포함되나요? ... 587
큐레이팅된 이미지에 필요한 도구가 포함되어 있지 않으면 어떻게 해야 하나요? ... 587
CodeBuild에서 AWS Lambda 컴퓨팅을 지원하는 리전은 어디입니까? ... 588
AWS Lambda 컴퓨팅의 제한 사항 ... 588
CodeBuild Lambda Java와 AWS SAM 함께를 사용하여 Lambda 함수 배포 ... 588
CodeBuild Lambda Node.js를 사용하여 단일 페이지 React 앱 생성 ... 592
CodeBuild Lambda Python으로 Lambda 함수 구성 업데이트 ... 595
예약 용량 플릿에서 빌드 실행 ... 599
예약 용량 플릿 생성 ... 600
모범 사례 ... 602
예약 용량 플릿을 여러 CodeBuild 프로젝트에서 공유할 수 있습니까? ... 602
속성 기반 컴퓨팅은 어떻게 작동하나요? ... 603
플릿에 Amazon EC2 인스턴스를 수동으로 지정할 수 있나요? ... 603
예약 용량 플릿을 지원하는 리전은 어디입니까? ... 603
예약 용량 macOS 플릿을 구성하려면 어떻게 해야 합니까? ... 604
예약 용량 플릿에 대한 사용자 지정 Amazon Machine Image(AMI)를 구성하려면 어떻게 해야 하나요? ... 605
예약 용량 플릿의 제한 ... 606
예약 용량 플릿 속성 ... 607
예약 용량 샘플 ... 611
배치 빌드 실행 ... 613
보안 역할 ... 613
배치 빌드 유형 ... 614
배치 보고서 모드 ... 618
추가 정보 ... 618
병렬 테스트 실행 ... 618
에서 지원 AWS CodeBuild ... 619
배치 빌드에서 병렬 테스트 실행 활성화 ... 622
`codebuild-tests-run` CLI 명령 사용 ... 623

codebuild-glob-search CLI 명령 사용 625
테스트 분할 정보 627
개별 빌드 보고서 자동 병합 627
샘플의 병렬 테스트 실행 629
캐시 빌드 640
Amazon S3 캐싱 640
로컬 캐싱 647
로컬 캐시 지정 648
빌드 디버그 651
CodeBuild 샌드박스를 사용하여 빌드 디버그 651
Session Manager를 사용하여 빌드 디버그 651
CodeBuild 샌드박스를 사용하여 빌드 디버그 651
Session Manager를 사용하여 빌드 디버그 681
빌드 삭제 685
빌드 삭제(AWS CLI) 686
빌드 삭제(AWS SDKs) 687
수동으로 빌드 재시도 687
빌드를 수동으로 재시도(콘솔) 687
빌드를 수동으로 재시도(AWS CLI) 688
빌드 수동 재시AWS SDKs) 688
자동으로 빌드 재시도 688
빌드 자동 재시도(콘솔) 689
빌드 자동 재시도(AWS CLI) 689
빌드 자동 재시AWS SDKs) 690
빌드 중지 690
빌드 중지(콘솔) 690
빌드 중지(AWS CLI) 691
빌드 중지(AWS SDKs) 691
배치 빌드 중지 691
배치 빌드 중지(콘솔) 692
배치 빌드 중지(AWS CLI) 692
배치 빌드(AWS SDKs) 중지 693
빌드를 자동으로 트리거 693
빌드 트리거 생성 693
빌드 트리거 편집 696
빌드 세부 정보 보기 699

빌드 세부 정보 보기(콘솔) ... 699
빌드 세부 정보 보기(AWS CLI) ... 699
빌드 세부 정보(AWS SDKs) 보기 ... 700
빌드 단계 진행 ... 700
빌드 ID 보기 ... 701
빌드 ID 목록 보기(콘솔) ... 701
빌드 ID 목록 보기(AWS CLI) ... 701
배치 빌드 ID 목록 보기(AWS CLI) ... 703
빌드 IDs(AWS SDKs) 목록 보기 ... 704
빌드 프로젝트의 빌드 ID 보기 ... 704
빌드 프로젝트의 빌드 ID 목록 보기(콘솔) ... 704
빌드 프로젝트의 빌드 ID 목록 보기(AWS CLI) ... 705
빌드 프로젝트의 배치 빌드 ID 목록 보기(AWS CLI) ... 706
빌드 프로젝트(AWS SDKs)의 빌드 IDs 목록 보기 ... 708
테스트 보고서 ... 709
테스트 보고서 생성 ... 710
코드 범위 보고서 생성 ... 711
코드 범위 보고서 생성 ... 712
보고서 자동 검색 ... 713
콘솔을 사용하여 보고서 자동 검색 구성 ... 714
프로젝트 환경 변수를 사용하여 보고서 자동 검색 구성 ... 715
보고서 그룹 ... 715
보고서 그룹 만들기 ... 716
보고서 그룹 이름 지정 ... 721
보고서 그룹 공유 ... 722
테스트 파일 지정 ... 728
테스트 명령 지정 ... 728
보고서 그룹 태그 지정 ... 729
보고서 그룹 업데이트 ... 734
테스트 프레임워크 ... 737
Jasmine 설정 ... 738
Jest 설정 ... 740
pytest 설정 ... 742
RSpec 설정 ... 743
테스트 보고서 보기 ... 743
빌드에 대한 테스트 보고서 보기 ... 744

보고서 그룹에 대한 테스트 보고서 보기 744
AWS 계정에서 테스트 보고서 보기 744
테스트 보고서 권한 745
테스트 보고서의 IAM 역할 745
테스트 보고 작업에 대한 권한 747
테스트 보고 권한 예제 747
테스트 보고서 상태 748
VPC 지원 750
사용 사례 750
VPC 모범 사례 751
VPC의 제한 사항 751
CodeBuild 프로젝트에서 Amazon VPC 액세스 허용 752
VPC 설정 문제 해결 753
VPC 엔드포인트 사용 753
VPC 엔드포인트를 생성하기 전에 754
CodeBuild용 VPC 엔드포인트 생성 754
CodeBuild에 대한 VPC 엔드포인트 정책 생성 755
CodeBuild 관리형 프록시 서버 사용 756
예약 용량 플릿에 대한 관리형 프록시 구성 756
CodeBuild 예약 용량 플릿 실행 758
프록시 서버 사용 758
프록시 서버에서 CodeBuild를 실행하기 위해 필요한 구성 요소 설정 759
명시적 프록시 서버에서 CodeBuild 실행 761
투명 프록시 서버에서 CodeBuild 실행 765
프록시 서버에서 패키지 관리자 및 기타 도구 실행 767
CloudFormation VPC 템플릿 769
로깅 및 모니터링 775
CodeBuild API 직접 호출 로깅 775
CloudTrail의 AWS CodeBuild 정보 775
AWS CodeBuild 로그 파일 항목 정보 776
빌드 모니터링 778
CloudWatch Metrics 779
CloudWatch 리소스 사용률 지표 782
CloudWatch 차원 783
CloudWatch 경보 784
CodeBuild 지표 보기 784

CodeBuild 리소스 사용률 지표 보기 786
CloudWatch에서 CodeBuild 경보 생성 790
보안 792
데이터 보호 792
데이터 암호화 794
키 관리 795
트래픽 개인 정보 보호 795
ID 및 액세스 관리 796
액세스 관리 개요 796
자격 증명 기반 정책 사용 800
AWS CodeBuild 권한 참조 828
태그를 사용하여 AWS CodeBuild 리소스에 대한 액세스 제어 835
콘솔에서 리소스 보기 839
규정 준수 확인 840
복원력 840
인프라 보안 840
소스 공급자 액세스 841
Secrets Manager 보안 암호에 토큰 생성 및 저장 841
GitHub 및 GitHub Enterprise Server 액세스 844
Bitbucket 액세스 855
GitLab 액세스 865
교차 서비스 혼동된 대리인 방지 872
고급 주제 874
사용자가 CodeBuild와 상호 작용하도록 허용 874
CodeBuild가 다른 AWS 서비스와 상호 작용하도록 허용 881
빌드 출력 암호화 889
를 사용하여 CodeBuild와 상호 작용 AWS CLI 891
명령줄 참조 891
AWS SDKs 및 도구 참조 893
AWS SDKs 및 도구 AWS CodeBuild 893
AWS SDKs 작업 894
CodeBuild 엔드포인트 지정 895
AWS CodeBuild 엔드포인트 지정(AWS CLI) 895
AWS CodeBuild 엔드포인트 지정(AWS SDK) 896
CodePipeline에서 CodeBuild 사용 898
사전 조건 899

파이프라인 생성(콘솔) ... 901
파이프라인 생성(AWS CLI) ... 905
빌드 작업 추가 ... 909
테스트 작업 추가 ... 913
Codecov에서 CodeBuild 사용 ... 916
Codecov를 빌드 프로젝트와 통합 ... 917
Jenkins에서 CodeBuild 사용 ... 920
Jenkins 설정 ... 920
플러그인 설치 ... 921
플러그인 사용 ... 921
서버리스 앱에서 CodeBuild 사용 ... 923
관련 리소스 ... 924
타사 작업 ... 924
1) 기본 도커 이미지 - windowsservercore ... 924
2) Windows 기반 도커 이미지 - choco ... 925
3) Windows 기반 도커 이미지 – git --버전 2.16.2 ... 926
4) Windows 기반 도커 이미지 - microsoft-build-tools --버전 15.0.26320.2 ... 926
5) Windows 기반 도커 이미지 - nuget.commandline --버전 4.5.1 ... 930
7) Windows 기반 도커 이미지 - netfx-4.6.2-devpack ... 930
8) Windows 기반 도커 이미지 - visualfsharptools, v 4.0 ... 931
9) Windows 기반 도커 이미지 - netfx-pcl-reference-assemblies-4.6 ... 932
10) Windows 기반 도커 이미지 - visualcppbuildtools v 14.0.25420.1 ... 935
11) Windows 기반 도커 이미지 - microsoft-windows-netfx3-ondemand-package.cab ... 938
12) Windows 기반 도커 이미지 - dotnet-sdk ... 939
CodeBuild 조건 키를 IAM 서비스 역할 변수로 사용 ... 940
AWS CodeBuild 조건 키 ... 941
프로젝트 및 플릿에 VPC 연결 설정 적용 ... 941
프로젝트 buildspec에 대한 무단 수정 방지 ... 942
빌드의 컴퓨팅 유형 제한 ... 943
제어 환경 변수 설정 ... 943
조건 키 이름에 변수 사용 ... 945
API 요청에 속성이 있는지 확인 ... 945
코드 예제 ... 947
기본 사항 ... 947
작업 ... 947
퍼블릭 빌드 API ... 964

퍼블릭 빌드 작업 964
DescribeBuildBatchesForPublicProject 965
DescribeBuildsForPublicProject 968
GetCloudWatchLogsForPublicBuild 973
GetPresignedUrlsForPublicBuild 976
GetPublicBuild 979
GetPublicBuildBatch 983
GetPublicProject 988
퍼블릭 빌드 데이터 형식 991
BuildBatchForDescribeBuildBatchesPublic 993
PublicBuild 995
PublicBuildArtifacts 999
PublicBuildBatch 1000
PublicBuildGroup 1004
PublicBuildSummary 1006
PublicLogsStatus 1008
PublicProject 1009
PublicProjectArtifacts 1012
PublicProjectBuildBatchConfig 1013
PublicProjectEnvironment 1014
PublicProjectSource 1016
PublicWebhook 1018
S3Downloadable 1020
문제 해결 1022
Apache Maven 빌드가 잘못된 리포지토리의 아티팩트를 참조함 1023
기본적으로 루트로 실행되는 빌드 명령 1025
파일 이름에 미국 영어 이외의 문자가 있으면 빌드가 실패할 수 있음 1025
Amazon EC2 Parameter Store에서 파라미터를 가져올 때 빌드가 실패할 수 있음 1026
CodeBuild 콘솔에서 브랜치 필터에 액세스할 수 없음 1027
빌드 성공 또는 실패 여부를 볼 수 없음 1027
빌드 상태가 소스 공급자에게 보고되지 않음 1027
Windows Server Core 2019 플랫폼의 기본 이미지를 찾고 선택할 수 없습니다. 1028
buildspec 파일의 앞에 있는 명령을 나중에 있는 명령에서 인식하지 못함 1028
오류: 캐시를 다운로드하려고 할 때 “Access denied”라는 메시지가 표시됨 1029
오류: 사용자 지정 빌드 이미지를 사용할 때
"BUILD_CONTAINER_UNABLE_TO_PULL_IMAGE"라는 메시지가 표시됨 1029

오류: “빌드를 완료하기 전에 빌드 컨테이너가 종료된 것으로 나타났습니다. 빌드 컨테이너가 메모리가 부족하거나 도커 이미지가 지원되지 않기 때문에 종료되었습니다. ErrorCode: 500" 1030
오류: 빌드 실행 시 "도커 데몬에 연결할 수 없음" 1031
오류: 빌드 프로젝트를 생성 또는 업데이트할 때 "CodeBuild is not authorized to perform: sts:AssumeRole"오류가 표시됨 1032
오류: "GetBucketAcl 호출 중 오류 발생: 버킷 소유자가 변경되었거나 해당 서비스 역할에 s3:GetBucketAcl이라는 이름의 권한이 없습니다" 1033
오류: 빌드를 실행할 때 "Failed to upload artifacts: Invalid arn"이라는 메시지가 표시됨 1033
오류: "Git Clone Failed: unable to access 'your-repository-URL': SSL certificate problem: self signed certificate" 1033
오류: 빌드를 실행할 때 "The bucket you are attempting to access must be addressed using the specified endpoint..."라는 메시지가 표시됨 1034
오류: "이 빌드 이미지는 하나 이상의 런타임 버전을 선택해야 합니다." 1034
오류: 빌드 대기열의 빌드가 실패할 때 "QUEUED: INSUFFICIENT_SUBNET"이라는 메시지가 표시됨 1035
오류: "Unable to download cache: RequestError: Send request failed caused by: x509: Failed to load system roots and no roots provided" 1036
오류: "Unable to download certificate from S3. AccessDenied" 1036
오류: "Unable to locate credentials" 1037
프록시 서버에서 CodeBuild를 실행할 때 RequestError 시간 초과 오류 1038
빌드 이미지에 있어야 하는 Bourne 쉘(sh) 1039
경고: 빌드 실행 시 “런타임 설치를 건너뜁니다. 런타임 버전 선택은 이 빌드 이미지에서 지원되지 않습니다.”가 표시됨 1040
오류: “JobWorker ID를 확인할 수 없음” 1040
빌드를 시작하지 못함 1040
로컬로 캐시된 빌드의 GitHub 메타데이터에 액세스 1040
AccessDenied: 보고서 그룹의 버킷 소유자가 S3 버킷 소유자와 일치하지 않습니다. 1041
오류:를 사용하여 CodeBuild 프로젝트를 생성할 때 "자격 증명에 필요한 권한 범위가 하나 이상 없음" AWS CodeConnections 1041
오류: Ubuntu 설치 명령을 사용하여 빌드 시 '죄송합니다. 터미널이 전혀 요청되지 않았습니다. 입력을 가져올 수 없습니다' 1042
Secrets Manager OAuth 사용 후 인증 오류와 함께 Bitbucket 빌드 실패 1043
할당량 1044
Service Quotas 1044
기타 제한 1049
빌드 프로젝트 1049

빌드 1050
컴퓨팅 플릿 1050
Reports 1051
Tags 1052
문서 기록 1054
이전 업데이트 1075
..... mlxxxvi

AWS CodeBuild란 무엇인가요?

AWS CodeBuild 는 클라우드의 완전 관리형 빌드 서비스입니다. CodeBuild는 소스 코드를 컴파일하고 단위 테스트를 실행하며 배포 준비가 완료된 아티팩트를 생성합니다. CodeBuild에서는 자체 빌드 서버를 프로비저닝, 관리 및 확장할 필요가 없습니다. 이 서비스는 Apache Maven, Gradle 등과 같은 널리 사용되는 프로그래밍 언어 및 빌드 도구에 맞게 사전 패키지된 빌드 환경을 제공합니다. CodeBuild에서 빌드 환경을 사용자 지정하여 사용자 고유의 빌드 도구를 사용할 수도 있습니다. CodeBuild는 최대 빌드 요청 수에 맞게 자동으로 확장합니다.

CodeBuild는 다음과 같은 이점을 제공합니다.

- 완전 관리형 - CodeBuild에서는 빌드 서버를 직접 설정하고, 패치 및 업데이트를 적용하고, 관리할 필요가 없습니다.
- 온디맨드 – CodeBuild는 빌드 요구 사항을 충족하기 위해 요구에 따라 크기가 조정됩니다. 사용한 빌드 시간만큼만 요금을 지불합니다.
- 즉시 사용 가능 – CodeBuild는 널리 사용되는 프로그래밍 언어에 맞게 사전 구성된 빌드 환경을 제공합니다. 빌드 스크립트를 선택하여 시작하기만 하면 됩니다.

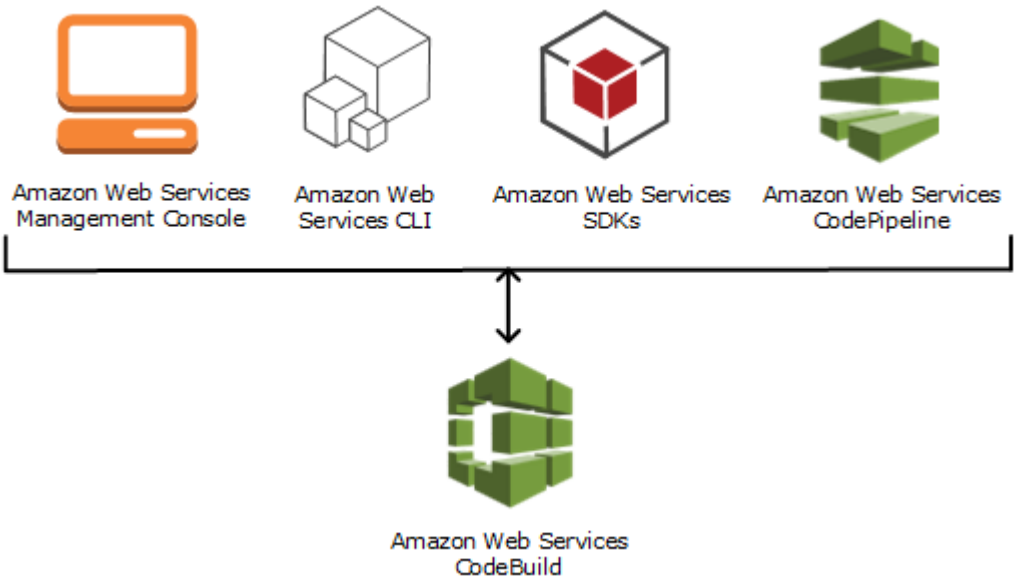
자세한 내용은 AWS CodeBuild 섹션을 참조하세요.

에 가입 AWS 계정

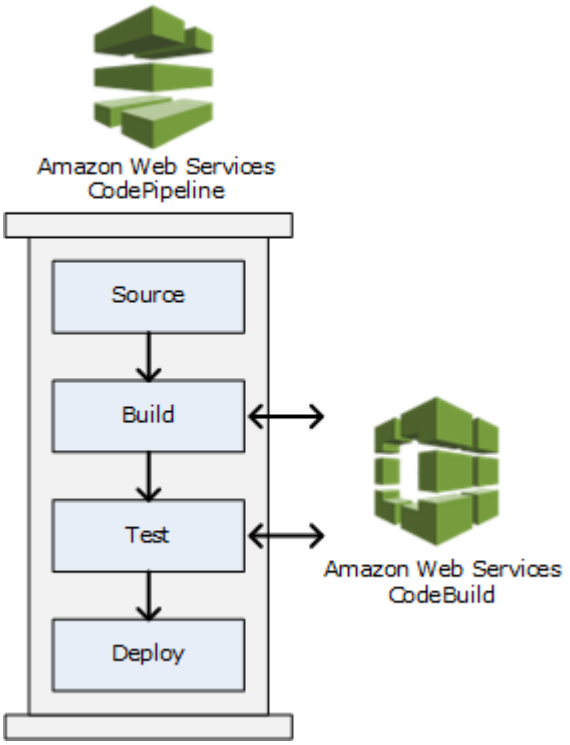
시작하려면이 AWS필요합니다 AWS 계정. 두 가지 방법으로 가입할 수 있습니다 AWS. 계정 구성 및 콘텐츠 저장 위치를 완벽하게 제어하거나 HIPAA 또는 FedRAMP와 같은 규제 대상 워크로드가 있는 경우 가입 AWS (고급)을 사용합니다. 새 사용자이고 이미 소유한 로그인을 사용하여에 로그인 AWS 하고에서 사전 구성된 기본값으로 AWS 환경을 생성하려면 가입을 AWS AWS 리전사용합니다.

CodeBuild 실행 방법

AWS CodeBuild 또는 AWS CodePipeline 콘솔을 사용하여 CodeBuild를 실행할 수 있습니다. AWS Command Line Interface (AWS CLI) 또는SDK를 사용하여 CodeBuild 실행을 자동화할 수도 있습니다. AWS SDKs



다음 다이어그램에서 볼 수 있듯이 CodeBuild를 파이프라인의 빌드 또는 테스트 단계에 빌드 또는 테스트 작업으로 추가할 수 있습니다 AWS CodePipeline. AWS CodePipeline 는 코드 릴리스에 필요한 단계를 모델링, 시각화 및 자동화하는 데 사용할 수 있는 지속적 제공 서비스입니다. 여기에는 코드 빌드도 포함됩니다. 파이프라인은 코드 변경 사항이 릴리스 프로세스를 통과하는 방식을 설명하는 워크플로우 구성입니다.



CodePipeline을 사용하여 파이프라인을 생성한 다음, CodeBuild 빌드나 테스트 작업을 추가하려면 CodePipeline에서 CodeBuild 사용 섹션을 참조하세요. CodePipeline에 대한 자세한 내용을 알아보려면 AWS CodePipeline 사용 설명서를 참조하세요.

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

CodeBuild 요금

자세한 내용은 CodeBuild 요금을 참조하세요.

CodeBuild는 어떻게 시작할 수 있나요?

다음 단계를 수행하는 것이 좋습니다.

1. 개념의 정보를 읽고 CodeBuild에 대해 자세히 알아보세요.
2. 콘솔을 사용하여 시작하기의 지침에 따라 예제 시나리오에서 CodeBuild를 실험해 보세요.
3. 빌드 계획의 지침에 따라 자체 시나리오에서 CodeBuild를 사용하세요.

AWS CodeBuild 개념

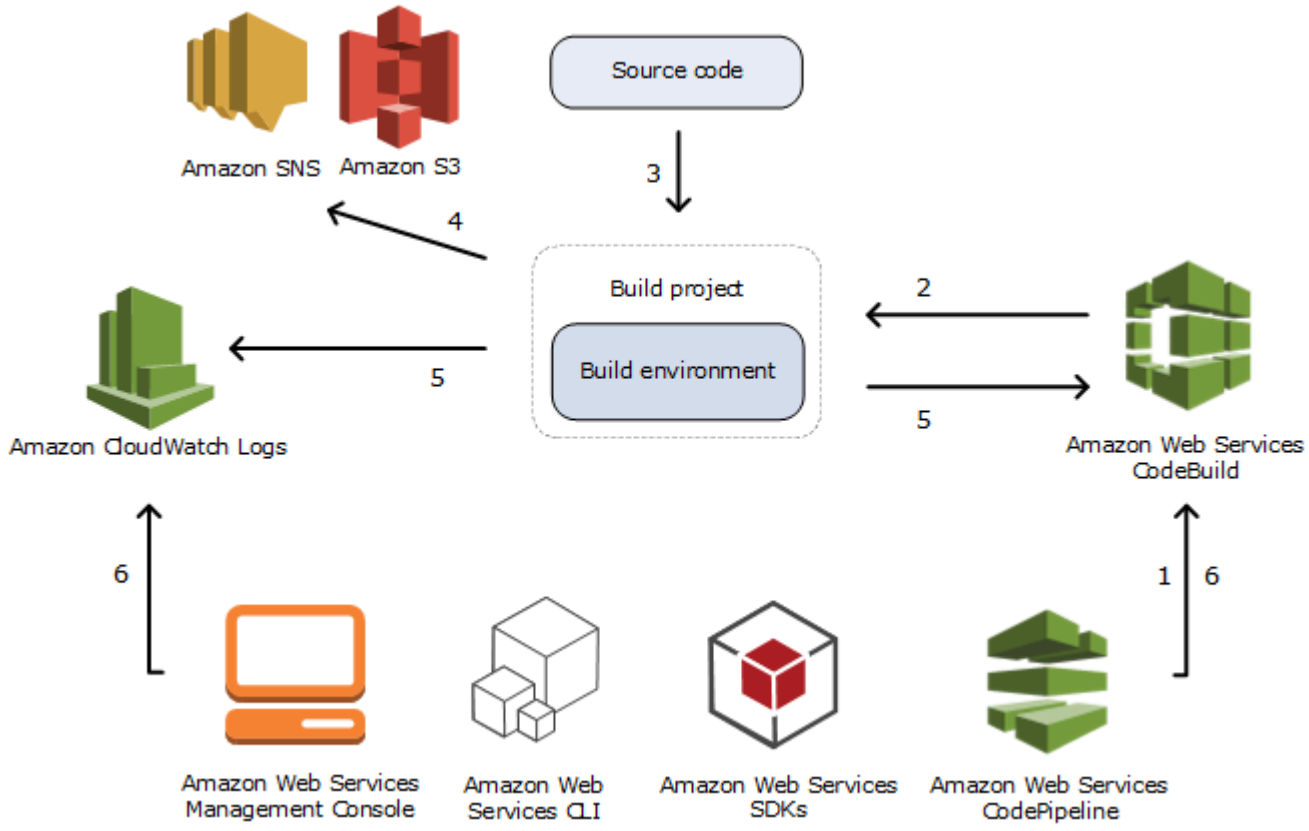
다음은 CodeBuild 작동 방식을 이해하는 데 필요한 중요한 개념입니다.

주제

- CodeBuild 작동 방식
- 다음 단계

CodeBuild 작동 방식

다음 다이어그램은 CodeBuild를 사용하여 빌드를 실행할 때 나타나는 현상을 보여 줍니다.



1. 입력으로 빌드 프로젝트와 함께 CodeBuild를 제공해야 합니다. 이 빌드 프로젝트에는 소스 코드를 가져올 위치, 사용할 빌드 환경, 실행할 빌드 명령 및 빌드 출력을 저장할 위치를 비롯하여 빌드 실행 방법에 대한 정보가 포함되어 있습니다. 빌드 환경은 CodeBuild가 빌드를 실행하는 데 사용하는 운영 체제, 프로그래밍 언어 런타임 및 도구의 조합을 나타냅니다. 자세한 내용은 다음을 참조하세요.

   - 빌드 프로젝트 생성
   - 빌드 환경 참조

2. CodeBuild가 빌드 프로젝트를 사용하여 빌드 환경을 생성합니다.
3. CodeBuild가 빌드 환경에 소스 코드를 다운로드한 다음, 빌드 프로젝트에 정의되거나 소스 코드에 직접 포함된 빌드 사양(buildspec)을 사용합니다. buildspec은 CodeBuild가 빌드를 실행하는 데 사용하는 YAML 형식의 빌드 명령 및 관련 설정의 모음입니다. 자세한 내용은 buildspec 참조 단원을 참조하십시오.
4. 빌드 출력이 있으면 빌드 환경에서 출력을 S3 버킷에 업로드합니다. 빌드 환경에서 사용자가 buildspec에 지정한 작업을 수행할 수도 있습니다(예: Amazon SNS 주제에 빌드 알림 전송). 예제는 빌드 알림 샘플 섹션을 참조하세요.

5. 빌드가 실행되는 동안 빌드 환경이 정보를 CodeBuild 및 Amazon CloudWatch Logs에 전송합니다.
6. 빌드가 실행되는 동안 AWS CodeBuild 콘솔 AWS CLI또는 AWS SDKs를 사용하여 CodeBuild에서 요약된 빌드 정보를 가져오고 Amazon CloudWatch Logs에서 자세한 빌드 정보를 가져올 수 있습니다. AWS CodePipeline 를 사용하여 빌드를 실행하는 경우 CodePipeline에서 제한된 빌드 정보를 가져올 수 있습니다.

다음 단계

이제에 대해 자세히 알아보았으므로 다음 단계를 AWS CodeBuild수행하는 것이 좋습니다.

1. 콘솔을 사용하여 시작하기의 지침에 따라 예제 시나리오에서 CodeBuild를 실험해 보세요.
2. 빌드 계획의 지침에 따라 자체 시나리오에서 CodeBuild를 사용하세요.

CodeBuild 시작하기

다음 자습서에서는 AWS CodeBuild 를 사용하여 샘플 소스 코드 입력 파일 컬렉션을 배포 가능한 소스 코드 버전으로 빌드합니다.

두 자습서 모두 입력과 결과가 동일하지만 하나는 AWS CodeBuild 콘솔을 사용하고 다른 하나는를 사용합니다 AWS CLI.

> **Important**
>
> AWS 루트 계정을 사용하여이 자습서를 완료하지 않는 것이 좋습니다.

주제

- 콘솔 AWS CodeBuild 사용 시작하기
- AWS CodeBuild 사용 시작하기 AWS CLI

콘솔 AWS CodeBuild 사용 시작하기

이 자습서에서는 AWS CodeBuild 를 사용하여 샘플 소스 코드 입력 파일(빌드 입력 아티팩트 또는 빌드 입력) 모음을 배포 가능한 버전의 소스 코드(빌드 출력 아티팩트 또는 빌드 출력)로 빌드합니다. 특히, CodeBuild가 일반적인 빌드 도구인 Apache Maven을 사용하여 Java 클래스 파일 세트를 Java Archive(JAR) 파일에 빌드하도록 명령을 지정합니다. 이 자습서는 Apache Maven이나 Java에 익숙하지 않아도 완료할 수 있습니다.

CodeBuild 콘솔, AWS CodePipeline또는 SDK를 통해 CodeBuild로 작업 AWS CLI할 AWS 수 있습니다. SDKs 이 자습서는 CodeBuild 콘솔을 사용하는 방법을 설명합니다. CodePipeline 사용에 대한 자세한 내용은 CodePipeline에서 CodeBuild 사용 섹션을 참조하세요.

> **Important**
>
> 이 자습서의 단계에서는 AWS 계정에 요금이 부과될 수 있는 리소스(예: S3 버킷)를 생성해야 합니다. 여기에는 CodeBuild와 Amazon S3 및 CloudWatch Logs와 관련된 AWS 리소스 AWS KMS및 작업에 대해 발생할 수 있는 요금이 포함됩니다. 자세한 내용은 AWS CodeBuild 요금, Amazon S3 요금, AWS Key Management Service 요금 및 Amazon CloudWatch 요금을 참조하세요.

주제

- 1단계: 소스 코드 생성
- 2단계: buildspec 파일 생성
- 3단계: 두 개의 S3 버킷 생성
- 4단계: 소스 코드 및 buildspec 파일 업로드
- 5단계: 빌드 프로젝트 생성
- 6단계: 빌드 실행
- 7단계: 요약된 빌드 정보 보기
- 8단계: 자세한 빌드 정보 보기
- 9단계: 빌드 출력 아티팩트 가져오기
- 10단계: S3 버킷 삭제
- 마무리

1단계: 소스 코드 생성

(콘솔 AWS CodeBuild 사용 시작하기의 일부)

이 단계에서는 CodeBuild가 출력 버킷을 빌드하도록 할 소스 코드를 생성합니다. 이 소스 코드는 두 개의 Java 클래스 파일 및 Apache Maven Project Object Model(POM) 파일로 구성됩니다.

1. 로컬 컴퓨터나 인스턴스의 빈 디렉터리에 다음 디렉터리 구조를 생성합니다.

```
(root directory name)
    `-- src
         |-- main
         |     `-- java
         `-- test
               `-- java
```

2. 원하는 텍스트 편집기를 사용하여 다음 파일을 생성하고 이름을 `MessageUtil.java`로 지정한 다음 이를 `src/main/java` 디렉터리에 저장합니다.

```
public class MessageUtil {
  private String message;

  public MessageUtil(String message) {
    this.message = message;
```

```
  }

  public String printMessage() {
    System.out.println(message);
    return message;
  }

  public String salutationMessage() {
    message = "Hi!" + message;
    System.out.println(message);
    return message;
  }
}
```

이 클래스 파일은 자신에게 전달되는 문자열을 출력으로 생성합니다. `MessageUtil` 생성자는 문자열을 설정합니다. `printMessage` 메서드는 출력을 생성합니다. `salutationMessage` 메서드는 `Hi!` 다음에 문자열을 출력합니다.

3. 다음 파일을 생성하고 이름을 `TestMessageUtil.java`로 지정한 다음 이를 `/src/test/java` 디렉터리에 저장합니다.

```
import org.junit.Test;
import org.junit.Ignore;
import static org.junit.Assert.assertEquals;

public class TestMessageUtil {

  String message = "Robert";
  MessageUtil messageUtil = new MessageUtil(message);

  @Test
  public void testPrintMessage() {
    System.out.println("Inside testPrintMessage()");
    assertEquals(message,messageUtil.printMessage());
  }

  @Test
  public void testSalutationMessage() {
    System.out.println("Inside testSalutationMessage()");
    message = "Hi!" + "Robert";
    assertEquals(message,messageUtil.salutationMessage());
  }
```

```
}
```

이 클래스 파일은 `MessageUtil` 클래스의 `message` 변수를 `Robert`로 설정합니다. 그런 다음 테스트를 수행하여 `Robert` 및 `Hi!Robert` 문자열이 출력에 나타나는지 여부를 확인하여 `message` 변수가 성공적으로 설정되었는지를 확인합니다.

4. 다음 파일을 생성하고 이름을 `pom.xml`로 지정한 다음 이를 루트(최상위) 디렉터리에 저장합니다.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/
maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.example</groupId>
  <artifactId>messageUtil</artifactId>
  <version>1.0</version>
  <packaging>jar</packaging>
  <name>Message Utility Java Sample App</name>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.0</version>
      </plugin>
    </plugins>
  </build>
</project>
```

Apache Maven은 이 파일의 지침을 따라 `MessageUtil.java` 및 `TestMessageUtil.java` 파일을 `messageUtil-1.0.jar`이라는 파일로 변환한 다음 지정된 테스트를 실행합니다.

이때 다음과 같이 디렉터리 구조가 나타나야 합니다.

```
(root directory name)
    |-- pom.xml
    `-- src
         |-- main
         |     `-- java
         |           `-- MessageUtil.java
         `-- test
               `-- java
                     `-- TestMessageUtil.java
```

2단계: buildspec 파일 생성

(이전 단계: 1단계: 소스 코드 생성)

이 단계에서는 빌드 사양 파일을 생성합니다. buildspec은 CodeBuild가 빌드를 실행하는 데 사용하는 YAML 형식의 빌드 명령 및 관련 설정의 모음입니다. 빌드 사양 없이는 CodeBuild가 빌드 입력을 빌드 출력으로 성공적으로 변환할 수 없으며, 빌드 환경의 빌드 출력 아티팩트를 찾아 출력 버킷에 업로드할 수도 없습니다.

다음 파일을 생성하고 이름을 `buildspec.yml`로 지정한 다음 이를 루트(최상위) 디렉터리에 저장합니다.

```
version: 0.2

phases:
  install:
    runtime-versions:
      java: corretto11
  pre_build:
    commands:
      - echo Nothing to do in the pre_build phase...
  build:
    commands:
      - echo Build started on `date`
      - mvn install
  post_build:
    commands:
      - echo Build completed on `date`
artifacts:
  files:
    - target/messageUtil-1.0.jar
```

> **Important**
>
> 빌드 사양 선언은 올바른 YAML이어야 하므로 빌드 사양 선언의 공백이 중요합니다. 빌드 사양 선언의 공백 수가 YAML과 맞지 않으면 빌드가 즉시 실패할 수 있습니다. YAML 검사기를 사용하여 빌드 사양 선언이 올바른 YAML인지 여부를 테스트할 수 있습니다.

> **Note**
>
> 소스 코드에 빌드 사양 파일을 포함하는 대신, 빌드 프로젝트를 생성할 때 빌드 명령을 별도로 선언할 수 있습니다. 이 방법은 매번 소스 코드 리포지토리를 업데이트하지 않아도 되도록 여러 개의 빌드 명령이 있는 소스 코드를 빌드하려는 경우 유용합니다. 자세한 내용은 buildspec 구문 단원을 참조하십시오.

이 빌드 사양 선언에서:

- `version`은 사용 중인 빌드 사양 표준의 버전을 나타냅니다. 이 빌드 사양 선언은 최신 버전인 `0.2`을 사용합니다.
- `phases`는 CodeBuild가 명령을 실행하도록 지시할 수 있는 빌드 단계를 나타냅니다. 여기에서는 이 빌드 단계가 `install`, `pre_build`, `build` 및 `post_build`로 나열되어 있습니다. 이 빌드 단계 이름의 철자는 변경할 수 없으며 추가로 빌드 단계 이름을 생성할 수도 없습니다.

  이 예제에서는 `build` 단계 중에 CodeBuild가 `mvn install` 명령을 실행합니다. 이 명령은 Apache Maven이 Java 클래스 파일을 컴파일 및 테스트하고 컴파일된 Java 클래스 파일을 빌드 출력 결과물에 패키지하도록 지시합니다. 그리고 몇 가지 `echo` 명령을 각 빌드 단계에 추가하여 이 연습을 마치게 됩니다. 이 자습서의 뒷부분에서 자세한 빌드 정보를 확인할 때 이러한 `echo` 명령의 출력을 확인하면 CodeBuild가 명령을 실행하는 방법 및 명령 실행 순서를 이해하는 데 많은 도움이 됩니다. (이 예에는 모든 빌드 단계가 포함되어 있지만, 해당 단계에서 아무 명령도 실행하지 않으려면 빌드 단계를 포함하지 않아도 됩니다.) 빌드 단계마다, CodeBuild는 처음부터 끝까지 한 번에 하나씩 나열된 순서로 지정된 각 명령을 실행합니다.
- `artifacts`는 CodeBuild가 출력 버킷에 업로드하는 빌드 출력 아티팩트 세트를 나타냅니다. `files`는 빌드 출력에 포함할 파일을 나타냅니다. CodeBuild는 빌드 환경에 `target` 상대 디렉터리에 있는 단일 `messageUtil-1.0.jar` 파일을 업로드합니다. 파일 이름 `messageUtil-1.0.jar` 및 디렉터리 이름 `target`은 Apache Maven이 이 예제에서만 빌드 출력 결과물을 생성 및 저장하는 방식에 따라 달라집니다. 사용자 자체 빌드에서는 이러한 파일 이름과 디렉터리가 다릅니다.

자세한 내용은 buildspec 참조 단원을 참조하십시오.

이때 다음과 같이 디렉터리 구조가 나타나야 합니다.

```
(root directory name)
    |-- pom.xml
    |-- buildspec.yml
    `-- src
         |-- main
         |     `-- java
         |           `-- MessageUtil.java
         `-- test
               `-- java
                     `-- TestMessageUtil.java
```

3단계: 두 개의 S3 버킷 생성

(이전 단계: 2단계: buildspec 파일 생성)

한 개의 버킷을 사용하여 이 실습을 수행할 수도 있지만, 두 개의 버킷을 사용해야 빌드 입력이 들어오는 위치 및 빌드 출력이 나가는 위치를 쉽게 확인할 수 있습니다.

- 이러한 버킷 중 하나(입력 버킷)는 빌드 입력을 저장합니다. 이 자습서에서이 입력 버킷의 이름은 입니다. `codebuild-`*`region-ID`*`-`*`account-ID`*`-input-bucket`여기서 *`region-ID`*는 버킷의 AWS 리전이고 *`account-ID`*는 AWS 계정 ID입니다.
- 다른 버킷(출력 버킷)은 빌드 출력을 저장합니다. 이 자습서에서 이 출력 버킷의 이름은 `codebuild-`*`region-ID`*`-`*`account-ID`*`-output-bucket`입니다.

이러한 버킷에 대해 다른 이름을 선택한 경우 이 자습서 전체에서 해당 이름을 사용해야 합니다.

이 두 버킷은 빌드와 동일한 AWS 리전에 있어야 합니다. 예를 들어, CodeBuild가 미국 동부(오하이오) 리전에서 빌드를 실행하도록 명령을 지정하는 경우, 이러한 버킷도 미국 동부(오하이오) 리전에 있어야 합니다.

자세한 내용은 Amazon Simple Storage Service 사용 설명서에서 버킷 생성을 참조하세요.

> **Note**
>
> CodeBuild는 CodeCommit, GitHub 및 Bitbucket 리포지토리에 저장된 빌드 입력도 지원하지만 이 자습서에서는 그에 대한 사용 방법을 다루지 않습니다. 자세한 내용은 빌드 계획 단원을 참조하십시오.

4단계: 소스 코드 및 buildspec 파일 업로드

(이전 단계: 3단계: 두 개의 S3 버킷 생성)

이 단계에서는 소스 코드 및 빌드 사양 파일을 입력 버킷에 추가합니다.

사용하는 운영 체제의 zip 유틸리티를 사용하여 `MessageUtil.java`, `TestMessageUtil.java`, `pom.xml` 및 `buildspec.yml`을 포함하는 `MessageUtil.zip`이라는 파일을 생성합니다.

`MessageUtil.zip` 파일의 디렉터리 구조가 다음과 같이 나타나야 합니다.

```
MessageUtil.zip
    |-- pom.xml
    |-- buildspec.yml
    `-- src
         |-- main
         |     `-- java
         |           `-- MessageUtil.java
         `-- test
               `-- java
                     `-- TestMessageUtil.java
```

> **Important**
>
> *(root directory name)* 디렉터리는 포함하지 말고, *(root directory name)* 디렉터리 안에 있는 디렉터리 및 파일만 포함하십시오.

`MessageUtil.zip` 파일을 `codebuild-`*region-ID*`-`*account-ID*`-input-bucket`이라는 입력 버킷에 업로드합니다.

> **Important**
>
> CodeCommit, GitHub 및 Bitbucket 리포지토리의 경우 일반적으로 `buildspec.yml`이라는 빌드 사양 파일을 각 리포지토리의 루트(최상위)에 저장하거나 빌드 사양 선언을 빌드 프로젝트 정의의 일부로 포함해야 합니다. 리포지토리의 소스 코드 및 빌드 사양 파일을 포함하는 ZIP 파일은 생성하지 마십시오.
> S3 버킷에만 저장된 빌드 입력의 경우, 일반적으로 루트(최상위)에 소스 코드 및 `buildspec.yml`이라는 빌드 사양 파일을 포함하는 ZIP 파일을 생성하거나 빌드 사양 선언을 빌드 프로젝트 정의의 일부로 포함해야 합니다.
> 빌드 사양 파일에 다른 이름을 사용하거나 루트가 아닌 위치에서 빌드 사양을 참조하려면 빌드 사양 재정의를 빌드 프로젝트 정의의 일부로 지정할 수 있습니다. 자세한 내용은 buildspec 파일 이름 및 스토리지 위치 단원을 참조하십시오.

5단계: 빌드 프로젝트 생성

(이전 단계: 4단계: 소스 코드 및 buildspec 파일 업로드)

이 단계에서는가 빌드를 실행하는 데 AWS CodeBuild 사용하는 빌드 프로젝트를 생성합니다. 이 빌드 프로젝트에는 소스 코드를 가져올 위치, 사용할 빌드 환경, 실행할 빌드 명령 및 빌드 출력을 저장할 위치를 비롯하여 빌드 실행 방법에 대한 정보가 포함되어 있습니다. 빌드 환경은 CodeBuild가 빌드를 실행하는 데 사용하는 운영 체제, 프로그래밍 언어 런타임 및 도구의 조합을 나타냅니다. 빌드 환경은 도커 이미지로 표현됩니다. 자세한 정보는 Docker Docs 웹 사이트에서 Docker 개요 주제를 참조하십시오.

이 빌드 환경의 경우 사용자가 CodeBuild가 JDK(Java Development Kit) 버전 및 Apache Maven을 포함하는 도커 이미지를 사용하도록 지시합니다.

빌드 프로젝트를 생성하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. AWS 리전 선택기를 사용하여 CodeBuild가 지원되는 AWS 리전을 선택합니다. 자세한 내용은 Amazon Web Services 일반 참조의 AWS CodeBuild 엔드포인트 및 할당량을 참조하세요.
3. CodeBuild 정보 페이지가 나타나면 빌드 프로젝트 생성을 선택합니다. 그렇지 않을 경우, 탐색 창에서 빌드를 확장한 후 빌드 프로젝트를 선택하고 빌드 프로젝트 생성을 선택합니다.

4. 빌드 프로젝트 만들기 페이지의 프로젝트 구성에서 프로젝트 이름에 이 빌드 프로젝트의 이름(이 예에서는 `codebuild-demo-project`)을 입력합니다. 빌드 프로젝트 이름은 각 AWS 계정에서 고유해야 합니다. 다른 이름을 사용하는 경우 이 자습서 전체에서 해당 이름을 사용해야 합니다.

   > **Note**
   >
   > 빌드 프로젝트 생성 페이지에 You are not authorized to perform this operation.와 같은 오류 메시지가 표시될 수 있습니다. 이는 빌드 프로젝트를 생성할 권한이 없는 사용자 AWS Management Console 로에 로그인했기 때문일 가능성이 높습니다. 이 문제를 해결하려면 에서 로그아웃 AWS Management Console한 다음 다음 IAM 엔터티 중 하나에 속하는 자격 증명으로 다시 로그인합니다.
   >
   > - AWS 계정의 관리자 사용자입니다. 자세한 내용은 사용 설명서의 첫 번째 AWS 계정 루트 사용자 및 그룹 생성을 참조하세요.
   > - 해당 사용자 또는 해당 사용자가 속한 IAM 그룹에 연결된 `AWSCodeBuildAdminAccess`, `AmazonS3ReadOnlyAccess`, 및 `IAMFullAccess` 관리형 정책이 있는 AWS 계정의 사용자입니다. AWS 계정에 이러한 권한이 있는 사용자 또는 그룹이 없고 사용자 또는 그룹에 이러한 권한을 추가할 수 없는 경우 AWS 계정 관리자에게 문의하여 도움을 받으세요. 자세한 내용은 AWS 에 대한 관리형(미리 정의된) 정책 AWS CodeBuild 단원을 참조하십시오.
   >
   > 두 옵션 모두 이 자습서를 완료할 수 있도록 빌드 프로젝트를 생성할 수 있는 관리자 권한이 포함되어 있습니다. 작업을 완료하는 데 필요한 최소 권한을 항상 사용하는 것이 좋습니다. 자세한 내용은 AWS CodeBuild 권한 참조 단원을 참조하십시오.

5. 소스의 소스 공급자에서 Amazon S3를 선택합니다.
6. 버킷에서 codebuild-***region-ID*-*account-ID***-input-bucket을 선택합니다.
7. S3 object key(S3 객체 키)에 **`MessageUtil.zip`**을 입력합니다.
8. 환경의 환경 이미지에서 관리형 이미지를 선택된 상태로 둡니다.
9. 운영 체제에서 Amazon Linux를 선택합니다.
10. 런타임에서 표준을 선택합니다.
11. 이미지의 경우 aws/codebuild/amazonlinux-x86_64-standard:corretto11을 선택합니다.
12. 서비스 역할에서 새 서비스 역할을 선택된 상태로 두고, 역할 이름도 변경하지 않고 그대로 둡니다.
13. Buildspec(빌드 사양)에서 Use a buildspec file(빌드 사양 파일 사용)을 선택된 상태로 둡니다.

14. 아티팩트에서 유형으로 Amazon S3를 선택합니다.
15. 버킷 이름에서 codebuild-***region-ID*-*account-ID***-output-bucket을 선택합니다.
16. 이름 및 경로는 비워 둡니다.
17. 빌드 프로젝트 생성을 선택합니다.

6단계: 빌드 실행

(이전 단계: 5단계: 빌드 프로젝트 생성)

이 단계에서는에 빌드 프로젝트의 설정으로 빌드를 실행 AWS CodeBuild 하도록 지시합니다.

빌드를 실행하려면

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 프로젝트 빌드를 선택합니다.
3. 빌드 프로젝트 목록에서 codebuild-demo-project를 선택한 후 빌드 시작을 선택합니다. 빌드가 즉시 시작됩니다.

7단계: 요약된 빌드 정보 보기

(이전 단계: 6단계: 빌드 실행)

이 단계에서는 빌드 상태에 대한 요약 정보를 확인합니다.

요약된 빌드 정보를 보려면

1. codebuild-demo-project:***<build-ID>*** 페이지가 표시되지 않으면, 탐색 모음에서 빌드 이력을 선택합니다. 그런 다음, 빌드 프로젝트 목록에서 프로젝트에 대해 codebuild-demo-project에 대한 빌드 실행 링크를 선택합니다. 일치하는 링크가 하나만 있어야 합니다. (이전에 이 자습서를 완료한 경우 완료됨 열에서 가장 최근 값이 포함된 링크를 선택합니다.)
2. 빌드 상태 페이지의 단계 세부 정보에는 다음과 같은 빌드 단계가 표시되고 상태 열에 성공이 표시되어야 합니다.

   - SUBMITTED
   - 대기됨
   - PROVISIONING

- DOWNLOAD_SOURCE
- INSTALL
- PRE_BUILD
- BUILD
- POST_BUILD
- UPLOAD_ARTIFACTS
- FINALIZING
- COMPLETED

빌드 상태에, 성공이 표시되어야 합니다.

대신 진행 중이 표시되면 새로 고침 단추를 선택합니다.

3. 각 빌드 단계 옆에 있는 기간 값은 빌드 단계가 지속되는 기간을 나타냅니다. 종료 시간 값은 빌드 단계가 종료되었음을 나타냅니다.

8단계: 자세한 빌드 정보 보기

(이전 단계: 7단계: 요약된 빌드 정보 보기)

이 단계에서는 CloudWatch Logs의 빌드에 대한 자세한 정보를 확인합니다.

> **Note**
>
> 중요한 정보를 보호하기 위해 CodeBuild 로그에 다음 항목이 숨겨져 있습니다.
>
> - AWS 액세스 키 IDs. 자세한 내용은AWS Identity and Access Management 사용 설명서에서 IAM 사용자의 액세스 키 관리를 참조하세요.
> - 파라미터 스토어를 사용하여 지정된 문자열입니다. 자세한 내용은 Amazon EC2 Systems Manager 사용 설명서의 Systems Manager Parameter Store 및 Systems Manager Parameter Store 콘솔 연습을 참조하세요.
> - 를 사용하여 지정된 문자열입니다 AWS Secrets Manager. 자세한 내용은 키 관리 단원을 참조하십시오.

자세한 빌드 정보를 보려면

1. 이전 단계의 빌드 세부 정보 페이지가 계속 표시된 상태에서 [Build logs]에 빌드 로그의 마지막 10,000행이 표시되어 있습니다. CloudWatch Logs에 전체 빌드 로그를 표시하려면 전체 로그 보기 링크를 선택합니다.
2. CloudWatch Logs 로그 스트림에서 로그 이벤트를 찾아 볼 수 있습니다. 기본적으로 가장 최근의 로그 이벤트 세트만 표시됩니다. 이전의 로그 이벤트를 보려면 목록의 처음으로 스크롤합니다.
3. 이 자습서에서는 대부분의 로그 이벤트에 CodeBuild가 빌드 종속성 파일을 빌드 환경에 다운로드 및 설치하는 작업에 대한 자세한 정보가 들어 있는데, 대부분의 사용자에게 필요하지 않은 정보입니다. [Filter events]를 사용하면 표시되는 정보를 줄일 수 있습니다. 예를 들어, 필터 이벤트에 `"[INFO]"`를 입력하면 `[INFO]`를 포함하는 이벤트만 표시됩니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서의 필터 및 패턴 구문을 참조하세요.

9단계: 빌드 출력 아티팩트 가져오기

(이전 단계: 8단계: 자세한 빌드 정보 보기)

이 단계에서는 CodeBuild가 빌드하고 출력 버킷에 업로드한 `messageUtil-1.0.jar` 파일을 가져옵니다.

CodeBuild 콘솔 또는 Amazon S3 콘솔을 사용하여 이 단계를 완료할 수 있습니다.

빌드 출력 아티팩트를 가져오려면(AWS CodeBuild 콘솔)

1. CodeBuild 콘솔이 여전히 열려 있고 이전 단계의 빌드 세부 정보 페이지가 계속 표시된 상태에서 빌드 세부 정보 탭을 선택하고 아티팩트 섹션으로 스크롤합니다.

   > **Note**
   >
   > 빌드 세부 정보 페이지가 표시되지 않으면, 탐색 모음에서 빌드 이력을 선택한 다음, 빌드 실행 링크를 선택합니다.

2. Amazon S3 폴더로 연결되는 링크는 아티팩트 업로드 위치 아래에 있습니다. 이 링크를 클릭하면 `messageUtil-1.0.jar` 빌드 출력 아티팩트 파일을 찾는 Amazon S3의 폴더가 열립니다.

빌드 출력 아티팩트를 가져오려면(Amazon S3 콘솔)

1. https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.

2. codebuild-*region-ID*-*account-ID*-output-bucket를 엽니다.
3. codebuild-demo-project 폴더를 엽니다.
4. target 폴더를 엽니다. 이 폴더에서 messageUtil-1.0.jar 빌드 출력 결과물 파일을 찾을 수 있습니다.

10단계: S3 버킷 삭제

(이전 단계: 9단계: 빌드 출력 아티팩트 가져오기)

AWS 계정에 지속적인 요금이 부과되지 않도록이 자습서에서 사용되는 입력 및 출력 버킷을 삭제할 수 있습니다. 지침을 보려면 Amazon Simple Storage Service 사용 설명서에서 버킷 삭제 또는 비우기를 참조하세요.

IAM 사용자 또는 관리자 IAM 사용자를 통해 이러한 버킷을 삭제하는 경우 사용자에게 추가 액세스 권한이 있어야 합니다. 사용자의 기존 액세스 정책에 마커 사이의 다음 명령문(*### BEGIN ADDING STATEMENT HERE ###* 및 *### END ADDING STATEMENTS HERE ###*)을 추가합니다.

이 명령문의 줄임표(...)는 간결하게 나타내기 위해 사용됩니다. 기존 액세스 정책의 어떤 명령문도 제거하지 마십시오. 이러한 줄임표는 정책에 입력하지 않아야 합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:DeleteBucket",
        "s3:DeleteObject"
      ],
      "Resource": "*"
    }
  ]
}
```

마무리

이 자습서에서는 AWS CodeBuild 를 사용하여 Java 클래스 파일 세트를 JAR 파일로 빌드했습니다. 그리고 빌드 결과를 확인했습니다.

이제 자신의 시나리오에서 CodeBuild를 사용해 볼 수 있습니다. 빌드 계획의 지침을 따르세요. 아직 시도해 볼 준비가 되지 않은 것 같으면 샘플 몇 가지를 더 빌드해 볼 수 있습니다. 자세한 내용은 CodeBuild용 사용 사례 기반 샘플 단원을 참조하십시오.

AWS CodeBuild 사용 시작하기 AWS CLI

이 자습서에서는 AWS CodeBuild 를 사용하여 샘플 소스 코드 입력 파일(빌드 입력 아티팩트 또는 빌드 입력이라고 함) 모음을 배포 가능한 버전의 소스 코드(빌드 출력 아티팩트 또는 빌드 출력이라고 함)로 빌드합니다. 특히, CodeBuild가 일반적인 빌드 도구인 Apache Maven을 사용하여 Java 클래스 파일 세트를 Java Archive(JAR) 파일에 빌드하도록 명령을 지정합니다. 이 자습서는 Apache Maven이나 Java에 익숙하지 않아도 완료할 수 있습니다.

CodeBuild 콘솔, AWS CodePipeline또는 SDK를 통해 CodeBuild로 작업 AWS CLI할 AWS 수 있습니다. SDKs 이 자습서에서는 AWS CLI에서 CodeBuild를 사용하는 방법을 보여 줍니다. CodePipeline 사용에 대한 자세한 내용은 CodePipeline에서 CodeBuild 사용 섹션을 참조하세요.

> **Important**
>
> 이 자습서의 단계에서는 AWS 계정에 요금이 부과될 수 있는 리소스(예: S3 버킷)를 생성해야 합니다. 여기에는 CodeBuild와 Amazon S3 및 CloudWatch Logs와 관련된 AWS 리소스 AWS KMS및 작업에 대해 발생할 수 있는 요금이 포함됩니다. 자세한 내용은 CodeBuild 요금, Amazon S3 요금, AWS Key Management Service 요금 및 Amazon CloudWatch 요금을 참조하세요.

주제

- 1단계: 소스 코드 생성
- 2단계: buildspec 파일 생성
- 3단계: 두 개의 S3 버킷 생성
- 4단계: 소스 코드 및 buildspec 파일 업로드
- 5단계: 빌드 프로젝트 생성
- 6단계: 빌드 실행
- 7단계: 요약된 빌드 정보 보기
- 8단계: 자세한 빌드 정보 보기

- 9단계: 빌드 출력 아티팩트 가져오기
- 10단계: S3 버킷 삭제
- 마무리

1단계: 소스 코드 생성

(AWS CodeBuild 사용 시작하기 AWS CLI의 일부)

이 단계에서는 CodeBuild가 출력 버킷을 빌드하도록 할 소스 코드를 생성합니다. 이 소스 코드는 두 개의 Java 클래스 파일 및 Apache Maven Project Object Model(POM) 파일로 구성됩니다.

1. 로컬 컴퓨터나 인스턴스의 빈 디렉터리에 다음 디렉터리 구조를 생성합니다.

```
(root directory name)
    `-- src
         |-- main
         |     `-- java
         `-- test
               `-- java
```

2. 원하는 텍스트 편집기를 사용하여 다음 파일을 생성하고 이름을 `MessageUtil.java`로 지정한 다음 이를 `src/main/java` 디렉터리에 저장합니다.

```
public class MessageUtil {
  private String message;

  public MessageUtil(String message) {
    this.message = message;
  }

  public String printMessage() {
    System.out.println(message);
    return message;
  }

  public String salutationMessage() {
    message = "Hi!" + message;
    System.out.println(message);
    return message;
  }
}
```

이 클래스 파일은 자신에게 전달되는 문자열을 출력으로 생성합니다. `MessageUtil` 생성자는 문자열을 설정합니다. `printMessage` 메서드는 출력을 생성합니다. `salutationMessage` 메서드는 `Hi!` 다음에 문자열을 출력합니다.

3. 다음 파일을 생성하고 이름을 `TestMessageUtil.java`로 지정한 다음 이를 `/src/test/java` 디렉터리에 저장합니다.

```
import org.junit.Test;
import org.junit.Ignore;
import static org.junit.Assert.assertEquals;

public class TestMessageUtil {

  String message = "Robert";
  MessageUtil messageUtil = new MessageUtil(message);

  @Test
  public void testPrintMessage() {
    System.out.println("Inside testPrintMessage()");
    assertEquals(message,messageUtil.printMessage());
  }

  @Test
  public void testSalutationMessage() {
    System.out.println("Inside testSalutationMessage()");
    message = "Hi!" + "Robert";
    assertEquals(message,messageUtil.salutationMessage());
  }
}
```

이 클래스 파일은 `MessageUtil` 클래스의 `message` 변수를 `Robert`로 설정합니다. 그런 다음 테스트를 수행하여 `Robert` 및 `Hi!Robert` 문자열이 출력에 나타나는지 여부를 확인하여 `message` 변수가 성공적으로 설정되었는지를 확인합니다.

4. 다음 파일을 생성하고 이름을 `pom.xml`로 지정한 다음 이를 루트(최상위) 디렉터리에 저장합니다.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/
maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.example</groupId>
```

```
  <artifactId>messageUtil</artifactId>
  <version>1.0</version>
  <packaging>jar</packaging>
  <name>Message Utility Java Sample App</name>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.0</version>
      </plugin>
    </plugins>
  </build>
</project>
```

Apache Maven은 이 파일의 지침을 따라 `MessageUtil.java` 및 `TestMessageUtil.java` 파일을 `messageUtil-1.0.jar`이라는 파일로 변환한 다음 지정된 테스트를 실행합니다.

이때 다음과 같이 디렉터리 구조가 나타나야 합니다.

```
(root directory name)
    |-- pom.xml
    `-- src
         |-- main
         |     `-- java
         |           `-- MessageUtil.java
         `-- test
               `-- java
                     `-- TestMessageUtil.java
```

2단계: buildspec 파일 생성

(이전 단계: 1단계: 소스 코드 생성)

이 단계에서는 빌드 사양 파일을 생성합니다. buildspec은 CodeBuild가 빌드를 실행하는 데 사용하는 YAML 형식의 빌드 명령 및 관련 설정의 모음입니다. 빌드 사양 없이는 CodeBuild가 빌드 입력을 빌드 출력으로 성공적으로 변환할 수 없으며, 빌드 환경의 빌드 출력 아티팩트를 찾아 출력 버킷에 업로드할 수도 없습니다.

다음 파일을 생성하고 이름을 `buildspec.yml`로 지정한 다음 이를 루트(최상위) 디렉터리에 저장합니다.

```
version: 0.2

phases:
  install:
    runtime-versions:
      java: corretto11
  pre_build:
    commands:
      - echo Nothing to do in the pre_build phase...
  build:
    commands:
      - echo Build started on `date`
      - mvn install
  post_build:
    commands:
      - echo Build completed on `date`
artifacts:
  files:
    - target/messageUtil-1.0.jar
```

Important

빌드 사양 선언은 올바른 YAML이어야 하므로 빌드 사양 선언의 공백이 중요합니다. 빌드 사양 선언의 공백 수가 YAML과 맞지 않으면 빌드가 즉시 실패할 수 있습니다. YAML 검사기를 사용하여 빌드 사양 선언이 올바른 YAML인지 여부를 테스트할 수 있습니다.

Note

소스 코드에 빌드 사양 파일을 포함하는 대신, 빌드 프로젝트를 생성할 때 빌드 명령을 별도로 선언할 수 있습니다. 이 방법은 매번 소스 코드 리포지토리를 업데이트하지 않아도 되도록 여

러 개의 빌드 명령이 있는 소스 코드를 빌드하려는 경우 유용합니다. 자세한 내용은 buildspec 구문 단원을 참조하십시오.

이 빌드 사양 선언에서:

- `version`은 사용 중인 빌드 사양 표준의 버전을 나타냅니다. 이 빌드 사양 선언은 최신 버전인 `0.2`을 사용합니다.
- `phases`는 CodeBuild가 명령을 실행하도록 지시할 수 있는 빌드 단계를 나타냅니다. 여기에서는 이 빌드 단계가 `install`, `pre_build`, `build` 및 `post_build`로 나열되어 있습니다. 이 빌드 단계 이름의 철자는 변경할 수 없으며 추가로 빌드 단계 이름을 생성할 수도 없습니다.

  이 예제에서는 `build` 단계 중에 CodeBuild가 `mvn install` 명령을 실행합니다. 이 명령은 Apache Maven이 Java 클래스 파일을 컴파일 및 테스트하고 컴파일된 Java 클래스 파일을 빌드 출력 결과물에 패키지하도록 지시합니다. 그리고 몇 가지 `echo` 명령을 각 빌드 단계에 추가하여 이 연습을 마치게 됩니다. 이 자습서의 뒷부분에서 자세한 빌드 정보를 확인할 때 이러한 `echo` 명령의 출력을 확인하면 CodeBuild가 명령을 실행하는 방법 및 명령 실행 순서를 이해하는 데 많은 도움이 됩니다. (이 예에는 모든 빌드 단계가 포함되어 있지만, 해당 단계에서 아무 명령도 실행하지 않으려면 빌드 단계를 포함하지 않아도 됩니다.) 빌드 단계마다, CodeBuild는 처음부터 끝까지 한 번에 하나씩 나열된 순서로 지정된 각 명령을 실행합니다.
- `artifacts`는 CodeBuild가 출력 버킷에 업로드하는 빌드 출력 아티팩트 세트를 나타냅니다. `files`는 빌드 출력에 포함할 파일을 나타냅니다. CodeBuild는 빌드 환경에 `target` 상대 디렉터리에 있는 단일 `messageUtil-1.0.jar` 파일을 업로드합니다. 파일 이름 `messageUtil-1.0.jar` 및 디렉터리 이름 `target`은 Apache Maven이 이 예제에서만 빌드 출력 결과물을 생성 및 저장하는 방식에 따라 달라집니다. 사용자 자체 빌드에서는 이러한 파일 이름과 디렉터리가 다릅니다.

자세한 내용은 buildspec 참조 단원을 참조하십시오.

이때 다음과 같이 디렉터리 구조가 나타나야 합니다.

```
(root directory name)
    |-- pom.xml
    |-- buildspec.yml
    `-- src
         |-- main
         |     `-- java
         |           `-- MessageUtil.java
         `-- test
```

```
                `-- java
                      `-- TestMessageUtil.java
```

3단계: 두 개의 S3 버킷 생성

(이전 단계: 2단계: buildspec 파일 생성)

한 개의 버킷을 사용하여 이 실습을 수행할 수도 있지만, 두 개의 버킷을 사용해야 빌드 입력이 들어오는 위치 및 빌드 출력이 나가는 위치를 쉽게 확인할 수 있습니다.

- 이러한 버킷 중 하나(입력 버킷)는 빌드 입력을 저장합니다. 이 자습서에서이 입력 버킷의 이름은 입니다. `codebuild-`*`region-ID`*`-`*`account-ID`*`-input-bucket`여기서 *`region-ID`*는 버킷의 AWS 리전이고 *`account-ID`*는 AWS 계정 ID입니다.
- 다른 버킷(출력 버킷)은 빌드 출력을 저장합니다. 이 자습서에서 이 출력 버킷의 이름은 `codebuild-`*`region-ID`*`-`*`account-ID`*`-output-bucket`입니다.

이러한 버킷에 대해 다른 이름을 선택한 경우 이 자습서 전체에서 해당 이름을 사용해야 합니다.

이 두 버킷은 빌드와 동일한 AWS 리전에 있어야 합니다. 예를 들어, CodeBuild가 미국 동부(오하이오) 리전에서 빌드를 실행하도록 명령을 지정하는 경우, 이러한 버킷도 미국 동부(오하이오) 리전에 있어야 합니다.

자세한 내용은 Amazon Simple Storage Service 사용 설명서에서 버킷 생성을 참조하세요.

> **Note**
>
> CodeBuild는 CodeCommit, GitHub 및 Bitbucket 리포지토리에 저장된 빌드 입력도 지원하지만 이 자습서에서는 그에 대한 사용 방법을 다루지 않습니다. 자세한 내용은 빌드 계획 단원을 참조하십시오.

4단계: 소스 코드 및 buildspec 파일 업로드

(이전 단계: 3단계: 두 개의 S3 버킷 생성)

이 단계에서는 소스 코드 및 빌드 사양 파일을 입력 버킷에 추가합니다.

사용하는 운영 체제의 zip 유틸리티를 사용하여 `MessageUtil.java`, `TestMessageUtil.java`, `pom.xml` 및 `buildspec.yml`을 포함하는 `MessageUtil.zip`이라는 파일을 생성합니다.

`MessageUtil.zip` 파일의 디렉터리 구조가 다음과 같이 나타나야 합니다.

```
MessageUtil.zip
    |-- pom.xml
    |-- buildspec.yml
    `-- src
         |-- main
         |     `-- java
         |           `-- MessageUtil.java
         `-- test
               `-- java
                     `-- TestMessageUtil.java
```

> **Important**
>
> *(root directory name)* 디렉터리는 포함하지 말고, *(root directory name)* 디렉터리 안에 있는 디렉터리 및 파일만 포함하십시오.

`MessageUtil.zip` 파일을 `codebuild-`*region-ID*`-`*account-ID*`-input-bucket`이라는 입력 버킷에 업로드합니다.

> **Important**
>
> CodeCommit, GitHub 및 Bitbucket 리포지토리의 경우 일반적으로 `buildspec.yml`이라는 빌드 사양 파일을 각 리포지토리의 루트(최상위)에 저장하거나 빌드 사양 선언을 빌드 프로젝트 정의의 일부로 포함해야 합니다. 리포지토리의 소스 코드 및 빌드 사양 파일을 포함하는 ZIP 파일은 생성하지 마십시오.
>
> S3 버킷에만 저장된 빌드 입력의 경우, 일반적으로 루트(최상위)에 소스 코드 및 `buildspec.yml`이라는 빌드 사양 파일을 포함하는 ZIP 파일을 생성하거나 빌드 사양 선언을 빌드 프로젝트 정의의 일부로 포함해야 합니다.
>
> 빌드 사양 파일에 다른 이름을 사용하거나 루트가 아닌 위치에서 빌드 사양을 참조하려면 빌드 사양 재정의를 빌드 프로젝트 정의의 일부로 지정할 수 있습니다. 자세한 내용은 buildspec 파일 이름 및 스토리지 위치 단원을 참조하십시오.

5단계: 빌드 프로젝트 생성

(이전 단계: 4단계: 소스 코드 및 buildspec 파일 업로드)

이 단계에서는가 빌드를 실행하는 데 AWS CodeBuild 사용하는 빌드 프로젝트를 생성합니다. 이 빌드 프로젝트에는 소스 코드를 가져올 위치, 사용할 빌드 환경, 실행할 빌드 명령 및 빌드 출력을 저장할 위치를 비롯하여 빌드 실행 방법에 대한 정보가 포함되어 있습니다. 빌드 환경은 CodeBuild가 빌드를 실행하는 데 사용하는 운영 체제, 프로그래밍 언어 런타임 및 도구의 조합을 나타냅니다. 빌드 환경은 도커 이미지로 표현됩니다. 자세한 정보는 Docker Docs 웹 사이트에서 Docker 개요 주제를 참조하십시오.

이 빌드 환경의 경우 사용자가 CodeBuild가 JDK(Java Development Kit) 버전 및 Apache Maven을 포함하는 도커 이미지를 사용하도록 지시합니다.

빌드 프로젝트를 생성하려면

1. AWS CLI 를 사용하여 create-project 명령을 실행합니다.

   ```
   aws codebuild create-project --generate-cli-skeleton
   ```

   JSON 형식 데이터가 출력에 표시됩니다. 가 설치된 로컬 컴퓨터 또는 인스턴스`create-project.json`의 위치에 있는 라는 파일에 데이터를 복사 AWS CLI 합니다. 다른 파일 이름을 사용하기로 선택하는 경우 이 자습서 전체에서 해당 이름을 사용해야 합니다.

   복사된 데이터를 다음 형식으로 수정한 다음 결과를 저장합니다.

   ```
   {
     "name": "codebuild-demo-project",
     "source": {
       "type": "S3",
       "location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"
     },
     "artifacts": {
       "type": "S3",
       "location": "codebuild-region-ID-account-ID-output-bucket"
     },
     "environment": {
       "type": "LINUX_CONTAINER",
       "image": "aws/codebuild/standard:5.0",
       "computeType": "BUILD_GENERAL1_SMALL"
     },
     "serviceRole": "serviceIAMRole"
   }
   ```

*serviceIAMRole*을 CodeBuild 서비스 역할의 Amazon 리소스 이름(ARN)으로 바꿉니다(예: arn:aws:iam::*account-ID*:role/*role-name*). 파일을 만들려면 CodeBuild가 다른 AWS 서비스와 상호 작용하도록 허용 섹션을 참조하세요.

이 데이터에서:

- name은 이 빌드 프로젝트에 대한 필수 식별자를 나타냅니다(이 예에서는 codebuild-demo-project). 빌드 프로젝트 이름은 계정에 있는 모든 빌드 프로젝트에서 고유해야 합니다.
- source의 type은 소스 코드의 리포지토리 유형을 나타내는 필수 값입니다(이 예에서는 Amazon S3 버킷의 경우 S3).
- source의 location은 소스 코드의 경로를 나타냅니다(이 예에서는 입력 버킷 이름과 그 뒤의 ZIP 파일 이름).
- artifacts의 type은 빌드 출력 아티팩트의 리포지토리 유형을 나타내는 필수 값입니다(이 예에서는 Amazon S3 버킷의 경우 S3).
- artifacts의 location은 앞에서 생성하거나 지정한 출력 버킷의 이름을 나타냅니다(이 예에서는 codebuild-*region-ID*-*account-ID*-output-bucket).
- environment의 type은 빌드 환경의 유형을 나타내는 필수 값입니다(이 예에서는 LINUX_CONTAINER).
- environment의 image는 도커 이미지 리포지토리 유형에서 지정한 대로 이 빌드 프로젝트에서 사용할 도커 이미지 이름 및 태그 조합을 나타내는 필수 값입니다(이 예에서는 CodeBuild 도커 이미지 리포지토리의 도커 이미지의 경우 aws/codebuild/standard:5.0). aws/codebuild/standard는 도커 이미지의 이름입니다. 5.0은 도커 이미지의 태그입니다.

  자신의 시나리오에서 사용할 수 있는 더 많은 도커 이미지를 찾으려면 빌드 환경 참조 단원을 참조하십시오.
- environment의 computeType은 CodeBuild가 사용할 컴퓨팅 리소스를 나타내는 필수 값입니다(이 예에서는 BUILD_GENERAL1_SMALL).

> **Note**
>
> 원래의 JSON 형식 데이터에서 사용 가능한 다른 값으로, description, buildspec, auth (type 및 resource 포함), path, namespaceType, name(artifacts의 경우), packaging, environmentVariables(name 및 value 포함), timeoutInMinutes, encryptionKey 및 tags(key 및 value 포함) 등이 있으며 선택 사항입니다. 이러한 값

은 이 자습서에서 사용되지 않으므로 여기서는 다루지 않습니다. 자세한 내용은 빌드 프로젝트 생성(AWS CLI) 단원을 참조하십시오.

2. 방금 저장한 파일이 들어 있는 디렉터리로 전환한 다음, create-project 명령을 다시 실행합니다.

```
aws codebuild create-project --cli-input-json file://create-project.json
```

이 명령이 제대로 실행되면 다음과 비슷한 데이터가 출력에 표시됩니다.

```
{
  "project": {
    "name": "codebuild-demo-project",
    "serviceRole": "serviceIAMRole",
    "tags": [],
    "artifacts": {
      "packaging": "NONE",
      "type": "S3",
      "location": "codebuild-region-ID-account-ID-output-bucket",
      "name": "message-util.zip"
    },
    "lastModified": 1472661575.244,
    "timeoutInMinutes": 60,
    "created": 1472661575.244,
    "environment": {
      "computeType": "BUILD_GENERAL1_SMALL",
      "image": "aws/codebuild/standard:5.0",
      "type": "LINUX_CONTAINER",
      "environmentVariables": []
    },
    "source": {
      "type": "S3",
      "location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"
    },
    "encryptionKey": "arn:aws:kms:region-ID:account-ID:alias/aws/s3",
    "arn": "arn:aws:codebuild:region-ID:account-ID:project/codebuild-demo-project"
  }
}
```

- project는 이 빌드 프로젝트에 대한 정보를 나타냅니다.
  - tags는 선언된 태그를 나타냅니다.

- `packaging`은 빌드 출력 결과물이 출력 버킷에 저장되는 방식을 나타냅니다. `NONE`은 폴더가 출력 버킷 내부에 생성됨을 의미합니다. 빌드 출력 결과물이 해당 폴더에 저장됩니다.
- `lastModified`는 빌드 프로젝트에 대한 정보가 마지막으로 변경된 시간을 Unix 시간 형식으로 나타냅니다.
- `timeoutInMinutes`는 빌드가 완료되지 않는 경우 CodeBuild가 해당 빌드를 중지할 시간(분)을 나타냅니다. (기본값은 60분입니다.)
- `created`는 빌드 프로젝트가 생성된 시간을 Unix 시간 형식으로 나타냅니다.
- `environmentVariables`는 선언된 환경 변수로, 빌드 중에 CodeBuild가 사용할 수 있는 환경 변수를 나타냅니다.
- `encryptionKey`는 CodeBuild가 빌드 출력 아티팩트를 암호화하는 데 사용하는 고객 관리형 키의 ARN을 나타냅니다.
- `arn`은 빌드 프로젝트의 ARN을 나타냅니다.

> **Note**
>
> create-project 명령을 실행하면 User: ***user-ARN*** is not authorized to perform: codebuild:CreateProject와 유사한 오류 메시지가 출력될 수 있습니다. 이는 CodeBuild를 사용하여 빌드 프로젝트를 생성할 수 있는 충분한 권한이 없는 사용자의 자격 증명 AWS CLI 으로를 구성했기 때문일 가능성이 높습니다. 이 문제를 해결하려면 IAM 엔터티 중 하나에 속한 보안 인증을 사용하여 AWS CLI 를 구성하세요.
>
> - AWS 계정의 관리자 사용자입니다. 자세한 내용은 사용 설명서의 첫 번째 AWS 계정 루트 사용자 및 그룹 생성을 참조하세요.
> - 해당 사용자 또는 해당 사용자가 속한 IAM 그룹에 연결된 `AWSCodeBuildAdminAccess`, `AmazonS3ReadOnlyAccess`, 및 `IAMFullAccess` 관리형 정책이 있는 AWS 계정의 사용자입니다. AWS 계정에 이러한 권한이 있는 사용자 또는 그룹이 없고 사용자 또는 그룹에 이러한 권한을 추가할 수 없는 경우 AWS 계정 관리자에게 문의하여 도움을 받으세요. 자세한 내용은 AWS 에 대한 관리형(미리 정의된) 정책 AWS CodeBuild 단원을 참조하십시오.

6단계: 빌드 실행

(이전 단계: 5단계: 빌드 프로젝트 생성)

이 단계에서는에 빌드 프로젝트의 설정으로 빌드를 실행 AWS CodeBuild 하도록 지시합니다.

빌드를 실행하려면

1. AWS CLI 를 사용하여 start-build 명령을 실행합니다.

```
aws codebuild start-build --project-name project-name
```

*project-name*을 이전 단계의 빌드 프로젝트 이름으로 바꿉니다(예: codebuild-demo-project).

2. 이 명령이 제대로 실행되면 다음과 비슷한 데이터가 출력에 표시됩니다.

```
{
  "build": {
    "buildComplete": false,
    "initiator": "user-name",
    "artifacts": {
      "location": "arn:aws:s3:::codebuild-region-ID-account-ID-output-bucket/
message-util.zip"
    },
    "projectName": "codebuild-demo-project",
    "timeoutInMinutes": 60,
    "buildStatus": "IN_PROGRESS",
    "environment": {
      "computeType": "BUILD_GENERAL1_SMALL",
      "image": "aws/codebuild/standard:5.0",
      "type": "LINUX_CONTAINER",
      "environmentVariables": []
    },
    "source": {
      "type": "S3",
      "location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"
    },
    "currentPhase": "SUBMITTED",
    "startTime": 1472848787.882,
    "id": "codebuild-demo-project:0cfbb6ec-3db9-4e8c-992b-1ab28EXAMPLE",
    "arn": "arn:aws:codebuild:region-ID:account-ID:build/codebuild-demo-
project:0cfbb6ec-3db9-4e8c-992b-1ab28EXAMPLE"
  }
}
```

- build는 이 빌드에 대한 정보를 나타냅니다.
  - buildComplete는 빌드 완료 여부를 나타냅니다(true). 그렇지 않을 경우 false입니다.

- `initiator`는 빌드를 시작한 엔터티를 나타냅니다.
- `artifacts`는 빌드 출력에 대한 정보를 나타냅니다(위치 포함).
- `projectName`은 빌드 프로젝트의 이름을 나타냅니다.
- `buildStatus`는 start-build 명령이 실행되었을 당시의 빌드 상태를 나타냅니다.
- `currentPhase`는 start-build 명령이 실행되었을 당시의 빌드 단계를 나타냅니다.
- `startTime`은 빌드 프로세스가 시작된 시간을 Unix 시간 형식으로 나타냅니다.
- `id`는 빌드의 ID를 나타냅니다.
- `arn`은 빌드의 ARN을 나타냅니다.

[id] 값을 기록해 둡니다. 이 정보는 다음 단계에서 필요합니다.

7단계: 요약된 빌드 정보 보기

(이전 단계: 6단계: 빌드 실행)

이 단계에서는 빌드 상태에 대한 요약 정보를 확인합니다.

요약된 빌드 정보를 보려면

- AWS CLI 를 사용하여 batch-get-builds 명령을 실행합니다.

```
aws codebuild batch-get-builds --ids id
```

*id*를 이전 단계의 출력에 표시된 `id` 값으로 바꿉니다.

이 명령이 제대로 실행되면 다음과 비슷한 데이터가 출력에 표시됩니다.

```
{
  "buildsNotFound": [],
  "builds": [
    {
      "buildComplete": true,
      "phases": [
        {
          "phaseStatus": "SUCCEEDED",
          "endTime": 1472848788.525,
          "phaseType": "SUBMITTED",
          "durationInSeconds": 0,
```

```
          "startTime": 1472848787.882
        },
        ... The full list of build phases has been omitted for brevity ...
        {
          "phaseType": "COMPLETED",
          "startTime": 1472848878.079
        }
      ],
      "logs": {
        "groupName": "/aws/codebuild/codebuild-demo-project",
        "deepLink": "https://console.aws.amazon.com/cloudwatch/home?region=region-
ID#logEvent:group=/aws/codebuild/codebuild-demo-project;stream=38ca1c4a-e9ca-4dbc-
bef1-d52bfEXAMPLE",
        "streamName": "38ca1c4a-e9ca-4dbc-bef1-d52bfEXAMPLE"
      },
      "artifacts": {
        "md5sum": "MD5-hash",
        "location": "arn:aws:s3:::codebuild-region-ID-account-ID-output-bucket/
message-util.zip",
        "sha256sum": "SHA-256-hash"
      },
      "projectName": "codebuild-demo-project",
      "timeoutInMinutes": 60,
      "initiator": "user-name",
      "buildStatus": "SUCCEEDED",
      "environment": {
        "computeType": "BUILD_GENERAL1_SMALL",
        "image": "aws/codebuild/standard:5.0",
        "type": "LINUX_CONTAINER",
        "environmentVariables": []
      },
      "source": {
        "type": "S3",
        "location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"
      },
      "currentPhase": "COMPLETED",
      "startTime": 1472848787.882,
      "endTime": 1472848878.079,
      "id": "codebuild-demo-project:38ca1c4a-e9ca-4dbc-bef1-d52bfEXAMPLE",
      "arn": "arn:aws:codebuild:region-ID:account-ID:build/codebuild-demo-
project:38ca1c4a-e9ca-4dbc-bef1-d52bfEXAMPLE"
    }
  ]
```

```
}
```

- `buildsNotFound`는 정보를 찾을 수 없는 빌드의 빌드 ID를 나타냅니다. 이 예에서는 비어 있어야 합니다.
- `builds`는 정보를 찾을 수 있는 각 빌드에 대한 정보를 나타냅니다. 이 예에서는 한 빌드에 대한 정보만 출력에 표시됩니다.
  - `phases`는 빌드 프로세스 동안 CodeBuild가 실행하는 빌드 단계 세트를 나타냅니다. 각 빌드 단계에 대한 정보는 `startTime`, `endTime` 및 `durationInSeconds`(빌드 단계가 시작 및 종료된 시간은 Unix 형식으로, 지속된 기간은 초로 표시)뿐만 아니라 `phaseType`(`SUBMITTED`, `PROVISIONING`, `DOWNLOAD_SOURCE`, `INSTALL`, `PRE_BUILD`, `BUILD`, `POST_BUILD`, `UPLOAD_ARTIFACTS`, `FINALIZING` 또는 `COMPLETED` 등) 및 `phaseStatus`(`SUCCEEDED`, `FAILED`, `FAULT`, `TIMED_OUT`, `IN_PROGRESS` 또는 `STOPPED`)가 개별적으로 나열됩니다. batch-get-builds 명령을 처음으로 실행하면 단계가 많이 표시되지 않거나 아예 하나도 표시되지 않을 수 있습니다. 동일한 빌드 ID로 batch-get-builds 명령을 계속하여 실행하면 더 많은 빌드 단계가 출력에 표시됩니다.
  - `logs`는 빌드 로그에 대한 정보를 Amazon CloudWatch Logs에 나타냅니다.
  - `md5sum` 및 `sha256sum`은 빌드 출력 결과물의 MD5 및 SHA-256 해시를 나타냅니다. 이러한 값은 빌드 프로젝트의 `packaging` 값이 `ZIP`으로 설정되어 있는 경우에만 출력에 표시됩니다. (이 자습서에서는 이 값을 설정하지 않음) 이러한 해시를 체크섬 도구와 함께 사용하면 파일 무결성 및 신뢰성을 확인할 수 있습니다.

    > **Note**
    >
    > Amazon S3 콘솔을 사용해서도 이러한 해시를 확인할 수 있습니다. 빌드 출력 결과물 옆의 확인란을 선택하고 작업, 속성을 차례로 선택합니다. [Properties] 창에서 [Metadata]를 확장하고 [x-amz-meta-codebuild-content-md5] 및 [x-amz-meta-codebuild-content-sha256]의 값을 확인합니다. (Amazon S3 콘솔에서 빌드 출력 아티팩트의 ETag 값을 MD5 또는 SHA-256 해시로 해석하지 않아야 합니다.)
    > AWS SDKs를 사용하여 이러한 해시를 가져오는 경우 값의 이름은 `codebuild-content-md5` 및 입니다`codebuild-content-sha256`.

  - `endTime`은 빌드 프로세스가 종료된 시간을 Unix 시간 형식으로 나타냅니다.

> **Note**
>
> Amazon S3 메타데이터에는 Amazon S3에 아티팩트를 게시하는 CodeBuild 빌드의 `buildArn`이 포함된 `x-amz-meta-codebuild-buildarn`라는 CodeBuild 헤더가 있습니다. 알림에 대한 소스 추적을 허용하고 아티팩트가 생성된 빌드를 참조할 수 있도록 하기 위해 `buildArn`이 추가되었습니다.

8단계: 자세한 빌드 정보 보기

(이전 단계: 7단계: 요약된 빌드 정보 보기)

이 단계에서는 CloudWatch Logs의 빌드에 대한 자세한 정보를 확인합니다.

> **Note**
>
> 중요한 정보를 보호하기 위해 CodeBuild 로그에 다음 항목이 숨겨져 있습니다.
>
> - AWS 액세스 키 IDs. 자세한 내용은AWS Identity and Access Management 사용 설명서에서 IAM 사용자의 액세스 키 관리를 참조하세요.
> - 파라미터 스토어를 사용하여 지정된 문자열입니다. 자세한 내용은 Amazon EC2 Systems Manager 사용 설명서의 Systems Manager Parameter Store 및 Systems Manager Parameter Store 콘솔 연습을 참조하세요.
> - 를 사용하여 지정된 문자열입니다 AWS Secrets Manager. 자세한 내용은 키 관리 단원을 참조하십시오.

자세한 빌드 정보를 보려면

1. 웹 브라우저를 사용하여 이전 단계의 출력에 표시된 `deepLink` 위치로 이동합니다(예: `https://console.aws.amazon.com/cloudwatch/home?region=region-ID#logEvent:group=/aws/codebuild/codebuild-demo-project;stream=38ca1c4a-e9ca-4dbc-bef1-d52bfEXAMPLE`).
2. CloudWatch Logs 로그 스트림에서 로그 이벤트를 찾아 볼 수 있습니다. 기본적으로 가장 최근의 로그 이벤트 세트만 표시됩니다. 이전의 로그 이벤트를 보려면 목록의 처음으로 스크롤합니다.

3. 이 자습서에서는 대부분의 로그 이벤트에 CodeBuild가 빌드 종속성 파일을 빌드 환경에 다운로드 및 설치하는 작업에 대한 자세한 정보가 들어 있는데, 대부분의 사용자에게 필요하지 않은 정보입니다. [Filter events]를 사용하면 표시되는 정보를 줄일 수 있습니다. 예를 들어, 필터 이벤트에 "[INFO]"를 입력하면 [INFO]를 포함하는 이벤트만 표시됩니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서의 필터 및 패턴 구문을 참조하세요.

CloudWatch Logs 로그 스트림의 다음 부분이 이 자습서와 관련이 있습니다.

```
...
[Container] 2016/04/15 17:49:42 Entering phase PRE_BUILD
[Container] 2016/04/15 17:49:42 Running command echo Entering pre_build phase...
[Container] 2016/04/15 17:49:42 Entering pre_build phase...
[Container] 2016/04/15 17:49:42 Phase complete: PRE_BUILD Success: true
[Container] 2016/04/15 17:49:42 Entering phase BUILD
[Container] 2016/04/15 17:49:42 Running command echo Entering build phase...
[Container] 2016/04/15 17:49:42 Entering build phase...
[Container] 2016/04/15 17:49:42 Running command mvn install
[Container] 2016/04/15 17:49:44 [INFO] Scanning for projects...
[Container] 2016/04/15 17:49:44 [INFO]
[Container] 2016/04/15 17:49:44 [INFO]
 ------------------------------------------------------------------------
[Container] 2016/04/15 17:49:44 [INFO] Building Message Utility Java Sample App 1.0
[Container] 2016/04/15 17:49:44 [INFO]
 ------------------------------------------------------------------------
...
[Container] 2016/04/15 17:49:55
 -------------------------------------------------------
[Container] 2016/04/15 17:49:55  T E S T S
[Container] 2016/04/15 17:49:55
 -------------------------------------------------------
[Container] 2016/04/15 17:49:55 Running TestMessageUtil
[Container] 2016/04/15 17:49:55 Inside testSalutationMessage()
[Container] 2016/04/15 17:49:55 Hi!Robert
[Container] 2016/04/15 17:49:55 Inside testPrintMessage()
[Container] 2016/04/15 17:49:55 Robert
[Container] 2016/04/15 17:49:55 Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time
 elapsed: 0.018 sec
[Container] 2016/04/15 17:49:55
[Container] 2016/04/15 17:49:55 Results :
[Container] 2016/04/15 17:49:55
[Container] 2016/04/15 17:49:55 Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
...
```

```
[Container] 2016/04/15 17:49:56 [INFO]
 ------------------------------------------------------------------------
[Container] 2016/04/15 17:49:56 [INFO] BUILD SUCCESS
[Container] 2016/04/15 17:49:56 [INFO]
 ------------------------------------------------------------------------
[Container] 2016/04/15 17:49:56 [INFO] Total time: 11.845 s
[Container] 2016/04/15 17:49:56 [INFO] Finished at: 2016-04-15T17:49:56+00:00
[Container] 2016/04/15 17:49:56 [INFO] Final Memory: 18M/216M
[Container] 2016/04/15 17:49:56 [INFO]
 ------------------------------------------------------------------------
[Container] 2016/04/15 17:49:56 Phase complete: BUILD Success: true
[Container] 2016/04/15 17:49:56 Entering phase POST_BUILD
[Container] 2016/04/15 17:49:56 Running command echo Entering post_build phase...
[Container] 2016/04/15 17:49:56 Entering post_build phase...
[Container] 2016/04/15 17:49:56 Phase complete: POST_BUILD Success: true
[Container] 2016/04/15 17:49:57 Preparing to copy artifacts
[Container] 2016/04/15 17:49:57 Assembling file list
[Container] 2016/04/15 17:49:57 Expanding target/messageUtil-1.0.jar
[Container] 2016/04/15 17:49:57 Found target/messageUtil-1.0.jar
[Container] 2016/04/15 17:49:57 Creating zip artifact
```

이 예에서는 CodeBuild가 사전 빌드, 빌드 및 사후 빌드의 빌드 단계를 성공적으로 완료했습니다. 또한 단위 테스트를 실행하고 `messageUtil-1.0.jar` 파일을 성공적으로 빌드했습니다.

9단계: 빌드 출력 아티팩트 가져오기

(이전 단계: 8단계: 자세한 빌드 정보 보기)

이 단계에서는 CodeBuild가 빌드하고 출력 버킷에 업로드한 `messageUtil-1.0.jar` 파일을 가져옵니다.

CodeBuild 콘솔 또는 Amazon S3 콘솔을 사용하여 이 단계를 완료할 수 있습니다.

빌드 출력 아티팩트를 가져오려면(AWS CodeBuild 콘솔)

1. CodeBuild 콘솔이 여전히 열려 있고 이전 단계의 빌드 세부 정보 페이지가 계속 표시된 상태에서 빌드 세부 정보 탭을 선택하고 아티팩트 섹션으로 스크롤합니다.

> **Note**
>
> 빌드 세부 정보 페이지가 표시되지 않으면, 탐색 모음에서 빌드 이력을 선택한 다음, 빌드 실행 링크를 선택합니다.

2. Amazon S3 폴더로 연결되는 링크는 아티팩트 업로드 위치 아래에 있습니다. 이 링크를 클릭하면 `messageUtil-1.0.jar` 빌드 출력 아티팩트 파일을 찾는 Amazon S3의 폴더가 열립니다.

빌드 출력 아티팩트를 가져오려면(Amazon S3 콘솔)

1. https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. `codebuild-`*`region-ID`*`-`*`account-ID`*`-output-bucket`를 엽니다.
3. `codebuild-demo-project` 폴더를 엽니다.
4. `target` 폴더를 엽니다. 이 폴더에서 `messageUtil-1.0.jar` 빌드 출력 결과물 파일을 찾을 수 있습니다.

10단계: S3 버킷 삭제

(이전 단계: 9단계: 빌드 출력 아티팩트 가져오기)

AWS 계정에 지속적인 요금이 부과되지 않도록이 자습서에서 사용되는 입력 및 출력 버킷을 삭제할 수 있습니다. 지침을 보려면 Amazon Simple Storage Service 사용 설명서에서 버킷 삭제 또는 비우기를 참조하세요.

IAM 사용자 또는 관리자 IAM 사용자를 통해 이러한 버킷을 삭제하는 경우 사용자에게 추가 액세스 권한이 있어야 합니다. 사용자의 기존 액세스 정책에 마커 사이의 다음 명령문(*`### BEGIN ADDING STATEMENT HERE ###`* 및 *`### END ADDING STATEMENTS HERE ###`*)을 추가합니다.

이 명령문의 줄임표(...)는 간결하게 나타내기 위해 사용됩니다. 기존 액세스 정책의 어떤 명령문도 제거하지 마십시오. 이러한 줄임표는 정책에 입력하지 않아야 합니다.

JSON

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

```
      "Effect": "Allow",
      "Action": [
        "s3:DeleteBucket",
        "s3:DeleteObject"
      ],
      "Resource": "*"
    }
  ]
}
```

마무리

이 자습서에서는 AWS CodeBuild 를 사용하여 Java 클래스 파일 세트를 JAR 파일로 빌드했습니다. 그리고 빌드 결과를 확인했습니다.

이제 자신의 시나리오에서 CodeBuild를 사용해 볼 수 있습니다. 빌드 계획의 지침을 따르세요. 아직 시도해 볼 준비가 되지 않은 것 같으면 샘플 몇 가지를 더 빌드해 볼 수 있습니다. 자세한 내용은 CodeBuild용 사용 사례 기반 샘플 단원을 참조하십시오.

CodeBuild용 사용 사례 기반 샘플

이러한 사용 사례 기반 샘플을 사용하여 AWS CodeBuild다음을 실험할 수 있습니다.

교차 서비스 샘플

실험할 교차 서비스 샘플 목록입니다 AWS CodeBuild.

빌드 배지 샘플

빌드 배지를 사용한 CodeBuild 설정 방법을 보여 줍니다.

테스트 보고서 샘플

AWS CLI 를 사용하여 테스트 보고서의 결과를 생성, 실행 및 봅니다.

CodeBuild용 Docker 샘플

사용자 지정 Docker 이미지를 사용하고, Amazon ECR의 리포지토리에 Docker 이미지를 게시하고, 프라이빗 레지스트리에서 Docker 이미지를 사용하는 방법을 보여줍니다.

빌드 출력을 S3 버킷에서 호스팅

S3 버킷에서 암호화되지 않은 빌드 아티팩트를 사용해 정적 웹사이트를 생성하는 방법을 보여줍니다.

다중 입력 및 출력 샘플

빌드 프로젝트에서 다중 입력 소스와 다중 출력 아티팩트를 사용하는 방법을 보여 줍니다.

샘플의 병렬 테스트 실행

`codebuild-tests-run` CLI 명령을 사용하여 병렬 실행 환경 간에 테스트를 분할하고 실행하는 방법을 보여줍니다.

buildspec 파일 샘플의 런타임 버전

buildspec 파일에서 런타임과 그 버전을 지정하는 방법을 보여줍니다.

소스 버전 샘플

CodeBuild 빌드 프로젝트에서 소스의 특정 버전을 사용하는 방법을 보여 줍니다.

CodeBuild용 타사 소스 리포지토리 샘플

CodeBuild를 사용하여 웹후크를 통해 BitBucket, GitHub Enterprise Server 및 GitHub pull 요청을 생성하는 방법을 보여줍니다.

의미 체계 버전 관리를 사용하여 빌드 시 아티팩트 이름 설정

빌드 시 의미 체계 버전 관리를 사용해 아티팩트 이름을 생성하는 방법을 보여 줍니다.

CodeBuild에 대한 교차 서비스 샘플

이러한 교차 서비스 샘플을 사용하여 AWS CodeBuild다음을 실험할 수 있습니다.

Amazon ECR 샘플

Amazon ECR 리포지토리의 도커 이미지를 사용하여 Apache Maven을 사용하여 단일 JAR 파일을 생성합니다. 샘플 지침은 Docker 이미지를 생성하여 Amazon ECR에 푸시하고, Go 프로젝트를 생성하고, 프로젝트를 빌드하고, 프로젝트를 실행하고, CodeBuild가 Amazon ECR에 연결할 수 있는 권한을 설정하는 방법을 보여줍니다.

Amazon EFS 샘플

CodeBuild 프로젝트가 Amazon EFS 파일 시스템에 마운트 및 빌드하도록 buildspec 파일을 구성하는 방법을 보여 줍니다. 샘플 지침은 Amazon VPC를 생성하고, Amazon VPC에서 파일 시스템을 생성하고, Amazon VPC를 사용하는 프로젝트를 생성 및 빌드한 다음 생성된 프로젝트 파일 및 변수를 검토하는 방법을 보여줍니다.

AWS CodePipeline 샘플

를 AWS CodePipeline 사용하여 배치 빌드와 여러 입력 소스 및 여러 출력 아티팩트가 포함된 빌드를 생성하는 방법을 보여줍니다. 이 섹션에는 별도의 아티팩트와 결합된 아티팩트를 사용하여 배치 빌드를 생성하는 파이프라인 구조를 보여주는 예제 JSON 파일이 포함되어 있습니다. 여러 입력 소스와 여러 출력 아티팩트가 있는 파이프라인 구조를 보여주는 추가 JSON 샘플이 제공됩니다.

AWS Config 샘플

설정 방법을 보여줍니다 AWS Config. 추적되는 CodeBuild 리소스를 나열하고 CodeBuild 프로젝트를 조회하는 방법을 설명합니다 AWS Config. 샘플 지침은와 통합하기 위한 사전 조건 AWS Config, 설정 단계 AWS Config, CodeBuild 프로젝트 및 데이터를 조회하는 단계를 보여줍니다 AWS Config.

빌드 알림 샘플

Apache Maven을 사용하여 단일 JAR 파일을 생성합니다. Amazon SNS 주제 구독자에게 빌드 알림을 보냅니다. 샘플 지침은 CodeBuild가 Amazon SNS 및 CloudWatch와 통신할 수 있도록 권한을 설정하는 방법, Amazon SNS에서 CodeBuild 주제를 생성하고 식별하는 방법, 주제에 대한 수신자를 구독하는 방법, CloudWatch에서 규칙을 설정하는 방법을 보여줍니다.

CodeBuild용 Amazon ECR 샘플

이 샘플에서는 Amazon Elastic Container Registry(Amazon ECR) 이미지 리포지토리에 있는 도커 이미지를 사용하여 샘플 Go 프로젝트를 빌드합니다.

> **Important**
>
> 이 샘플을 실행하면 AWS 계정에 요금이 부과될 수 있습니다. 여기에는 Amazon S3, , AWS KMS CloudWatch Logs AWS CodeBuild 및 Amazon ECR과 관련된 AWS 리소스 및 작업에 대한 및 요금이 포함됩니다. 자세한 내용은 CodeBuild 요금, Amazon S3 요금, AWS Key Management Service 요금, Amazon CloudWatch 요금, Amazon Elastic Container Registry 요금을 참조하세요.

주제

- Amazon ECR 샘플 실행

Amazon ECR 샘플 실행

다음 지침에 따라 CodeBuild용 Amazon ECR 샘플을 실행합니다.

이 샘플을 실행하려면

1. Docker 이미지를 생성하고 Amazon ECR의 이미지 리포지토리에 푸시하려면, 'Amazon ECR에 Docker 이미지 게시' 샘플의 'Amazon ECR에 Docker 이미지 게시' 샘플 실행 섹션에 있는 단계를 수행하세요.
2. Go 프로젝트 만들기:
   a. 이 주제의 Go 프로젝트 구조 및 Go 프로젝트 파일 섹션에 설명된 대로 파일을 생성한 다음 S3 입력 버킷 또는 AWS CodeCommit, GitHub 또는 Bitbucket 리포지토리에 업로드합니다.

      > **Important**
      >
      > *(root directory name)*은 업로드하지 말고, *(root directory name)* 안에 있는 파일만 업로드하십시오.

S3 입력 버킷을 사용하고 있는 경우, 파일을 포함하는 ZIP 파일을 생성한 다음, 이를 입력 버킷에 업로드하십시오. *(root directory name)*을 ZIP 파일에 추가하지 말고, *(root directory name)* 안에 있는 파일만 추가하십시오.

b. 빌드 프로젝트를 생성하고, 빌드를 실행하고, 관련 빌드 정보를 확인합니다.

AWS CLI 를 사용하여 빌드 프로젝트를 생성하는 경우 `create-project` 명령에 대한 JSON 형식의 입력이 이와 비슷할 수 있습니다. (자리 표시자는 사용자의 값으로 바꾸십시오.)

```
{
  "name": "sample-go-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-bucket/GoSample.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket",
    "packaging": "ZIP",
    "name": "GoOutputArtifact.zip"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/standard:5.0",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

c. 빌드 출력 아티팩트를 가져오려면 S3 출력 버킷을 엽니다.

d. *GoOutputArtifact*`.zip` 파일을 로컬 컴퓨터나 인스턴스에 다운로드한 다음 파일의 내용을 추출합니다. 추출한 내용에서 `hello` 파일을 가져옵니다.

3. 다음 중 하나가 true인 경우가 도커 이미지를 빌드 환경으로 가져올 AWS CodeBuild 수 있도록 Amazon ECR의 이미지 리포지토리에 권한을 추가해야 합니다.

- 프로젝트에서는 CodeBuild 보안 인증을 사용하여 Amazon ECR 이미지를 끌어옵니다. 이는 `ProjectEnvironment`의 `imagePullCredentialsType` 속성에 `CODEBUILD` 값으로 표시됩니다.

- 프로젝트에서는 교차 계정 Amazon ECR 이미지를 사용합니다. 이 경우에는 프로젝트에서 서비스 역할을 사용하여 Amazon ECR 이미지를 끌어와야 합니다. 이 동작을 활성화하려면 `ProjectEnvironment`의 `imagePullCredentialsType` 속성을 `SERVICE_ROLE`로 설정합니다.

1. Amazon ECR 콘솔(https://console.aws.amazon.com/ecr/)을 엽니다.
2. 리포지토리 이름 목록에서 생성했거나 선택한 리포지토리의 이름을 선택합니다.
3. 탐색 창에서 권한을 선택하고, 편집을 선택한 다음 설명문 추가를 선택합니다.
4. Statement name(설명문 이름)에 식별자(예: **`CodeBuildAccess`**)를 입력합니다.
5. 효과에 대해 허용이 선택된 채로 둡니다. 이는 다른 AWS 계정으로의 액세스를 허용하겠다는 의미입니다.
6. 보안 주체에 대해 다음 중 하나를 실시합니다.
   - 프로젝트에서 CodeBuild 보안 인증을 사용하여 Amazon ECR 이미지를 끌어오는 경우 서비스 보안 주체에 **`codebuild.amazonaws.com`**을 입력합니다.
   - 프로젝트에서 교차 계정 Amazon ECR 이미지를 사용할 경우에는 AWS 계정 ID에 액세스 권한을 부여할 AWS 계정의 ID를 입력합니다.
7. All IAM entities(모든 IAM 엔터티) 목록을 건너뜁니다.
8. 작업은 가져오기 전용 작업인 ecr:GetDownloadUrlForLayer, ecr:BatchGetImage 및 ecr:BatchCheckLayerAvailability를 선택합니다.
9. 조건에 다음을 추가합니다.

```
{
   "StringEquals":{
      "aws:SourceAccount":"<AWS-account-ID>",
      "aws:SourceArn":"arn:aws:codebuild:<region>:<AWS-account-
ID>:project/<project-name>"
   }
}
```

10. 저장을 선택합니다.

이 정책이 Permissions(권한)에 표시됩니다. 보안 주체는 이 절차의 3단계에서 보안 주체에 입력한 내용입니다.

- 프로젝트에서 CodeBuild 보안 인증을 사용하여 Amazon ECR 이미지를 끌어오는 경우 서비스 보안 주체 아래에 "codebuild.amazonaws.com"이 나타납니다.

- 프로젝트에서 교차 계정 Amazon ECR 이미지를 사용하는 경우 액세스 권한을 부여하려는 AWS 계정의 ID가 AWS 계정 IDs 아래에 나타납니다.

  다음 샘플 정책은 CodeBuild 보안 인증과 교차 계정 Amazon ECR 이미지를 모두 사용합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "CodeBuildAccessPrincipal",
            "Effect": "Allow",
            "Action": [
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "ecr:BatchCheckLayerAvailability"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceArn": "arn:aws:codebuild:us-
east-1:111122223333:project/MyProject",
                    "aws:SourceAccount": "111122223333"
                }
            }
        },
        {
            "Sid": "CodeBuildAccessCrossAccount",
            "Effect": "Allow",
            "Action": [
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "ecr:BatchCheckLayerAvailability"
            ],
            "Resource": "*"
        }
    ]
}
```

- 프로젝트에서 CodeBuild 보안 인증을 사용하고 CodeBuild 프로젝트에서 Amazon ECR 리포지토리에 대한 퍼블릭 액세스 권한을 갖도록 하려면 `Condition` 키를 생략하고 다음 샘플 정책을 추가할 수 있습니다.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "CodeBuildAccessPrincipal",
            "Effect": "Allow",
            "Resource": [
                "arn:aws:codecommit:us-
east-2:111122223333:MySharedDemoRepo"
            ],
            "Action": [
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "ecr:BatchCheckLayerAvailability"
            ]
        },
        {
            "Sid": "CodeBuildAccessCrossAccount",
            "Effect": "Allow",
            "Resource": [
                "arn:aws:codecommit:us-
east-2:111122223333:MySharedDemoRepo"
            ],
            "Action": [
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "ecr:BatchCheckLayerAvailability"
            ]
        }
    ]
}
```

4. 빌드 프로젝트를 생성하고, 빌드를 실행하고, 빌드 정보를 확인합니다.

   AWS CLI 를 사용하여 빌드 프로젝트를 생성하는 경우 `create-project` 명령에 대한 JSON 형식의 입력이 이와 비슷할 수 있습니다. (자리 표시자는 사용자의 값으로 바꾸십시오.)

```
{
  "name": "amazon-ecr-sample-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-bucket/GoSample.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket",
    "packaging": "ZIP",
    "name": "GoOutputArtifact.zip"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "account-ID.dkr.ecr.region-ID.amazonaws.com/your-Amazon-ECR-repo-
name:tag",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

5. 빌드 출력 아티팩트를 가져오려면 S3 출력 버킷을 엽니다.
6. *GoOutputArtifact*.zip 파일을 로컬 컴퓨터나 인스턴스에 다운로드한 다음 *GoOutputArtifact*.zip 파일의 내용을 추출합니다. 추출한 내용에서 hello 파일을 가져옵니다.

Go 프로젝트 구조

이 샘플에서는 다음 디렉터리 구조를 가정합니다.

```
(root directory name)
### buildspec.yml
### hello.go
```

Go 프로젝트 파일

이 샘플은 다음 파일을 사용합니다.

buildspec.yml(*(root directory name)*에 있음)

```
version: 0.2

phases:
  install:
   runtime-versions:
     golang: 1.13
  build:
    commands:
      - echo Build started on `date`
      - echo Compiling the Go code
      - go build hello.go
  post_build:
    commands:
      - echo Build completed on `date`
artifacts:
  files:
    - hello
```

hello.go(*(root directory name)*에 있음)

```
package main
import "fmt"

func main() {
  fmt.Println("hello world")
  fmt.Println("1+1 =", 1+1)
  fmt.Println("7.0/3.0 =", 7.0/3.0)
  fmt.Println(true && false)
  fmt.Println(true || false)
  fmt.Println(!true)
}
```

용 Amazon Elastic File System 샘플 AWS CodeBuild

Amazon EC2 인스턴스에 대한 확장 가능한 공유 파일 서비스인 Amazon Elastic File System에서 AWS CodeBuild 빌드를 생성할 수 있습니다. Amazon EC2 Amazon EFS를 통한 스토리지 용량은 탄력적이므로 파일이 추가 및 제거될 때 확장되거나 축소됩니다. 또한 파일 시스템을 만들고 구성할 수 있는 간편한 웹 서비스 인터페이스를 제공합니다. 이 서비스는 모든 파일 스토리지 인프라를 관리하므로, 파일 시스템 구성을 배포하거나 패치를 적용하거나 유지 보수하는 데 신경을 쓸 필요가 없습니다. 자세한 내용은 Amazon Elastic File System 사용 설명서에서 Amazon Elastic File System이란?을 참조하세요.

이 예제는 Amazon EFS 파일 시스템에 Java 애플리케이션을 마운트 및 빌드하도록 CodeBuild 프로젝트를 구성하는 방법을 보여 줍니다. 시작하기 전에 S3 입력 버킷 또는 AWS CodeCommit GitHub, GitHub Enterprise Server 또는 Bitbucket 리포지토리에 업로드되는 Java 애플리케이션을 빌드할 준비가 되어 있어야 합니다.

파일 시스템에 전송되는 데이터는 암호화됩니다. 다른 이미지를 사용하여 전송 중인 데이터를 암호화하려면 전송 중 데이터 암호화를 참조하십시오.

주제

- Amazon Elastic File System과 AWS CodeBuild 함께 사용
- Amazon EFS 통합 문제 해결

Amazon Elastic File System과 AWS CodeBuild 함께 사용

이 샘플에서는 Amazon EFS를와 함께 사용하는 데 필요한 4가지 상위 수준 단계를 다룹니다 AWS CodeBuild. 스크립트는 다음과 같습니다.

1. AWS 계정에서 Virtual Private Cloud(VPC)를 생성합니다.
2. 이 VPC를 사용하는 파일 시스템을 생성합니다.
3. VPC를 사용하는 CodeBuild 프로젝트를 생성하고 빌드합니다. CodeBuild 프로젝트는 다음을 사용하여 파일 시스템을 식별합니다.
   - 고유한 파일 시스템 식별자. 식별자는 빌드 프로젝트에서 파일 시스템을 지정할 때 선택합니다.
   - 파일 시스템 ID. Amazon EFS 콘솔에서 파일 시스템을 볼 때 ID가 표시됩니다.
   - 탑재 지점. 파일 시스템을 탑재하는 Docker 컨테이너에 있는 디렉터리입니다.
   - 탑재 옵션. 여기에는 파일 시스템을 탑재하는 방법에 대한 세부 정보가 포함됩니다.
4. 빌드 프로젝트를 검토하여 올바른 프로젝트 파일과 변수가 생성되었는지 확인합니다.

> **Note**
>
> Amazon EFS에서 생성한 파일 시스템은 Linux 플랫폼에서만 지원됩니다.

주제

- 1단계:를 사용하여 VPC 생성 CloudFormation

- 2단계: VPC에서 Amazon Elastic File System 파일 시스템 생성
- 3단계: Amazon EFS에서 사용할 CodeBuild 프로젝트 생성
- 4단계: 빌드 프로젝트 검토

1단계:를 사용하여 VPC 생성 CloudFormation

CloudFormation 템플릿을 사용하여 VPC를 생성합니다.

1. 의 지침에 따라 CloudFormation VPC 템플릿를 CloudFormation 사용하여 VPC를 생성합니다.

   > **Note**
   >
   > 이 CloudFormation 템플릿으로 생성된 VPC에는 프라이빗 서브넷 2개와 퍼블릭 서브넷 2개가 있습니다. AWS CodeBuild 을 사용하여 Amazon EFS에서 생성한 파일 시스템을 탑재할 때는 프라이빗 서브넷만 사용해야 합니다. 퍼블릭 서브넷 중 하나를 사용할 경우 빌드가 실패합니다.

2. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/vpc/ Amazon VPC 콘솔을 엽니다.
3. 생성한 VPC를 선택합니다 CloudFormation.
4. 설명 탭에 표시된 VPC의 이름과 ID를 기록해 둡니다. VPC 이름과 ID 모두 이 예제 뒷부분에서 AWS CodeBuild 프로젝트를 만들 때 필요합니다.

2단계: VPC에서 Amazon Elastic File System 파일 시스템 생성

이 예제를 위해 앞에서 만든 VPC를 사용하여 간단한 Amazon EFS 파일 시스템을 생성합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/efs/ Amazon EFS 콘솔을 엽니다.
2. 파일 시스템 생성을 선택합니다.
3. 이 예제 앞부분에서 기록해 둔 VPC 이름을 VPC에서 선택합니다.
4. 가용 영역은 선택한 서브넷과 연결된 채로 둡니다.
5. 다음 단계를 선택합니다.
6. 태그 추가에서 기본 이름 키의 값에 Amazon EFS 파일 시스템의 이름을 입력합니다.
7. 기본 성능 및 처리량 모드로 선택한 Bursting(버스팅) 및 범용 모드를 그대로 두고 다음 단계를 선택합니다.

8. 클라이언트 액세스 구성에서 다음 단계를 선택합니다.
9. 파일 시스템 생성을 선택합니다.
10. (선택 사항) 전송 중 데이터를 암호화하는 정책을 Amazon EFS 파일 시스템에 추가하는 것이 좋습니다. Amazon EFS 콘솔에서 파일 시스템 정책을 선택하고 편집을 선택한 다음, 모든 클라이언트에 전송 중 암호화 적용이라는 레이블이 붙은 상자를 선택하고 저장을 선택합니다.

3단계: Amazon EFS에서 사용할 CodeBuild 프로젝트 생성

이 샘플의 앞부분에서 생성한 VPC를 사용하는 AWS CodeBuild 프로젝트를 생성합니다. 빌드가 실행되면 이전에 생성된 Amazon EFS 파일 시스템을 탑재합니다. 그런 다음 Java 애플리케이션에서 생성된 .jar 파일을 파일 시스템의 탑재 지점 디렉터리에 저장합니다.

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 빌드 프로젝트를 선택한 후 빌드 프로젝트 생성을 선택합니다.
3. 프로젝트 이름에 프로젝트의 이름을 입력합니다.
4. 소스 공급자에서 빌드하려는 Java 애플리케이션을 포함하는 리포지토리를 선택합니다.
5. CodeBuild가 애플리케이션을 찾기 위해 사용하는 리포지토리 URL 등과 같은 정보를 입력합니다. 옵션은 소스 공급자마다 다릅니다. 자세한 내용은 Choose source provider 단원을 참조하십시오.
6. 환경 이미지에서 관리형 이미지를 선택합니다.
7. 운영 체제에서 Amazon Linux 2를 선택합니다.
8. 런타임에서 표준을 선택합니다.
9. 이미지에서 aws/codebuild/amazonlinux-x86_64-standard:4.0을 선택합니다.
10. 환경 유형에서 Linux를 선택합니다.
11. 서비스 역할에서 새 서비스 역할을 선택합니다. 역할 이름에 CodeBuild가 생성할 역할 이름을 입력합니다.
12. 추가 구성을 확장합니다.
13. Docker 이미지를 빌드하거나 빌드에서 승격된 권한을 얻으려는 경우 이 플래그 활성화를 선택합니다.

> **Note**
>
> 기본적으로 비 VPC 빌드에는 Docker 데몬이 활성화됩니다. VPC 빌드에 Docker 컨테이너를 사용하려면 Docker Docs 웹 사이트의 런타임 권한 및 Linux 기능을 참조하고 권한 부여 모드를 활성화합니다. 또한 Windows는 권한 모드를 지원하지 않습니다.

14. VPC에서 VPC ID를 선택합니다.
15. 서브넷에서 VPC에 연결된 프라이빗 서브넷을 한 개 이상 선택합니다. Amazon EFS 파일 시스템을 마운트하는 빌드에 프라이빗 서브넷을 사용해야 합니다. 퍼블릭 서브넷을 사용할 경우 빌드가 실패합니다.
16. 보안 그룹에서 기본 보안 그룹을 선택합니다.
17. 파일 시스템에 다음 정보를 입력합니다.

    - 식별자에 고유한 파일 시스템 식별자를 입력합니다. 길이는 129자 미만이어야 하고 영숫자 문자와 밑줄만 사용할 수 있습니다. CodeBuild는 이 식별자를 사용하여 탄력적 파일 시스템을 식별하는 환경 변수를 생성합니다. 환경 변수 형식은 `CODEBUILD_<file_system_identifier>`(대문자)입니다. 예를 들어 `my_efs`을 입력하면 환경 변수는 `CODEBUILD_MY_EFS`입니다.
    - ID에서 파일 시스템 ID를 선택합니다.
    - (선택 사항) 파일 시스템에 디렉터리를 입력합니다. CodeBuild는 이 디렉터리를 탑재합니다. 디렉터리 경로를 비워 두면 CodeBuild에서 전체 파일 시스템을 탑재합니다. 경로는 파일 시스템의 루트와 관련이 있습니다.
    - 탑재 지점에 파일 시스템을 탑재하는 빌드 컨테이너의 디렉터리 절대 경로를 입력합니다. 이 디렉터리가 없으면 CodeBuild에서 빌드 중에 생성됩니다.
    - (선택 사항) 탑재 옵션을 입력합니다. 탑재 옵션을 비워 두면 CodeBuild에서는 기본 탑재 옵션을 사용합니다.

      ```
      nfsvers=4.1
      rsize=1048576
      wsize=1048576
      hard
      timeo=600
      retrans=2
      ```

      자세한 내용은 Amazon Elastic File System 사용 설명서의 권장되는 NFS 탑재 옵션을 참조하세요.
18. 빌드 사양에서 빌드 명령 삽입을 선택한 후 편집기로 전환을 선택합니다.
19. 편집기에 다음 buildspec 명령을 입력합니다. 17단계에서 입력한 식별자로 `<file_system_identifier>`를 바꿉니다. 대문자(예: `CODEBUILD_MY_EFS`)를 사용합니다.

    ```
    version: 0.2
    phases:
    ```

```
  install:
    runtime-versions:
      java: corretto11
  build:
    commands:
      - mvn compile -Dgpg.skip=true -Dmaven.repo.local=
$CODEBUILD_<file_system_identifier>
```

20. 기타 모든 설정에 대해 기본값을 사용하고 빌드 프로젝트 생성을 선택합니다. 빌드가 완료되면 프로젝트의 콘솔 페이지가 표시됩니다.
21. 빌드 시작를 선택합니다.

4단계: 빌드 프로젝트 검토

AWS CodeBuild 프로젝트가 빌드된 후:

- Amazon EFS 파일 시스템에 빌드된 Java 애플리케이션에서 생성된 .jar 파일을 탑재 지점 디렉터리에 저장합니다.
- 파일 시스템을 식별하는 환경 변수는 프로젝트를 만들 때 입력한 파일 시스템 식별자를 사용하여 생성됩니다.

자세한 내용은 Amazon Elastic File System 사용 설명서에서 파일 시스템 탑재를 참조하세요.

Amazon EFS 통합 문제 해결

CodeBuild로 Amazon EFS를 설정할 때 발생할 수 있는 오류는 다음과 같습니다.

주제

- CLIENT_ERROR: '127.0.0.1:/' 탑재에 실패했습니다. 권한이 거부되었습니다.
- CLIENT_ERROR: '127.0.0.1:/' 탑재에 실패했습니다. 피어에서 연결을 재설정했습니다.
- VPC_CLIENT_ERROR: 예상치 못한 EC2 오류: UnauthorizedOperation

CLIENT_ERROR: '127.0.0.1:/' 탑재에 실패했습니다. 권한이 거부되었습니다.

CodeBuild로 Amazon EFS를 탑재하는 데는 IAM 인증이 지원되지 않습니다. 사용자 지정 Amazon EFS 파일 시스템 정책을 사용하는 경우 모든 IAM 보안 주체에 읽기 및 쓰기 액세스 권한을 부여해야 합니다. 예제:

```
"Principal": {
  "AWS": "*"
}
```

CLIENT_ERROR: '127.0.0.1:/' 탑재에 실패했습니다. 피어에서 연결을 재설정했습니다.

이 오류의 가능한 두 가지 원인은 다음과 같습니다.

- CodeBuild VPC 서브넷은 Amazon EFS 탑재 대상과 다른 가용 영역에 있습니다. Amazon EFS 탑재 대상과 동일한 가용 영역에 VPC 서브넷을 추가하여 이 문제를 해결할 수 있습니다.
- 보안 그룹에는 Amazon EFS와 통신할 권한이 없습니다. VPC(VPC의 기본 CIDR 블록 추가) 또는 보안 그룹 자체에서 들어오는 모든 트래픽을 허용하는 인바운드 규칙을 추가하여 이 문제를 해결할 수 있습니다.

VPC_CLIENT_ERROR: 예상치 못한 EC2 오류: UnauthorizedOperation

이 오류는 CodeBuild 프로젝트의 VPC 구성에 있는 모든 서브넷이 퍼블릭 서브넷일 때 발생합니다. 네트워크 연결을 보장하려면 VPC에 프라이빗 서브넷이 하나 이상 있어야 합니다.

AWS CodePipeline CodeBuild용 샘플

이 섹션에서는 CodePipeline과 CodeBuild 간의 샘플 통합에 대해 설명합니다.

샘플	설명
CodePipeline/CodeBuild 통합 및 배치 빌드 샘플	이 샘플은를 AWS CodePipeline 사용하여 배치 빌드를 사용하는 빌드 프로젝트를 생성하는 방법을 보여줍니다.
CodePipeline/CodeBuild와 다중 입력 소스 및 출력 아티팩트 통합 샘플	이 샘플은를 AWS CodePipeline 사용하여 여러 입력 소스를 사용하여 여러 출력 아티팩트를 생성하는 빌드 프로젝트를 생성하는 방법을 보여줍니다.

CodePipeline/CodeBuild 통합 및 배치 빌드 샘플

AWS CodeBuild 는 배치 빌드를 지원합니다. 다음 샘플은를 AWS CodePipeline 사용하여 배치 빌드를 사용하는 빌드 프로젝트를 생성하는 방법을 보여줍니다.

파이프라인의 구조를 정의하는 JSON 형식 파일을 사용한 다음와 함께 사용하여 파이프라인 AWS CLI을 생성할 수 있습니다. 자세한 내용은AWS CodePipeline 사용 설명서의 AWS CodePipeline 파이프라인 구조 참조를 참조하세요.

개별 아티팩트를 사용한 배치 빌드

다음 JSON 파일을 별도의 아티팩트가 있는 배치 빌드를 생성하는 파이프라인 구조의 예로 사용하세요. CodePipeline에서 배치 빌드를 활성화하려면 `configuration` 객체의 `BatchEnabled` 파라미터를 `true`로 설정합니다.

```
{
  "pipeline": {
    "roleArn": "arn:aws:iam::account-id:role/my-AWS-CodePipeline-service-role-name",
    "stages": [
      {
        "name": "Source",
        "actions": [
          {
            "inputArtifacts": [],
            "name": "Source1",
            "actionTypeId": {
              "category": "Source",
              "owner": "AWS",
              "version": "1",
              "provider": "S3"
            },
            "outputArtifacts": [
              {
                "name": "source1"
              }
            ],
            "configuration": {
              "S3Bucket": "<my-input-bucket-name>",
              "S3ObjectKey": "my-source-code-file-name.zip"
            },
            "runOrder": 1
          },
          {
            "inputArtifacts": [],
            "name": "Source2",
            "actionTypeId": {
              "category": "Source",
              "owner": "AWS",
```

```
            "version": "1",
            "provider": "S3"
          },
          "outputArtifacts": [
            {
              "name": "source2"
            }
          ],
          "configuration": {
            "S3Bucket": "<my-other-input-bucket-name>",
            "S3ObjectKey": "my-other-source-code-file-name.zip"
          },
          "runOrder": 1
        }
      ]
    },
    {
      "name": "Build",
      "actions": [
        {
          "inputArtifacts": [
            {
              "name": "source1"
            },
            {
              "name": "source2"
            }
          ],
          "name": "Build",
          "actionTypeId": {
            "category": "Build",
            "owner": "AWS",
            "version": "1",
            "provider": "CodeBuild"
          },
          "outputArtifacts": [
            {
              "name": "build1"
            },
            {
              "name": "build1_artifact1"
            },
            {
              "name": "build1_artifact2"
```

```
              },
              {
                "name": "build2_artifact1"
              },
              {
                "name": "build2_artifact2"
              }
            ],
            "configuration": {
              "ProjectName": "my-build-project-name",
              "PrimarySource": "source1",
              "BatchEnabled": "true"
            },
            "runOrder": 1
          }
        ]
      }
    ],
    "artifactStore": {
      "type": "S3",
      "location": "<AWS-CodePipeline-internal-bucket-name>"
    },
    "name": "my-pipeline-name",
    "version": 1
  }
}
```

다음은 이 파이프라인 구성과 함께 작동하는 CodeBuild buildspec 파일의 예입니다.

```
version: 0.2
batch:
  build-list:
    - identifier: build1
      env:
        compute-type: BUILD_GENERAL1_SMALL
    - identifier: build2
      env:
        compute-type: BUILD_GENERAL1_MEDIUM

phases:
  build:
    commands:
      - echo 'file' > output_file
```

```
artifacts:
  files:
    - output_file
  secondary-artifacts:
    artifact1:
      files:
        - output_file
    artifact2:
      files:
        - output_file
```

파이프라인의 JSON 파일에서 지정하는 출력 아티팩트의 이름은 buildspec 파일에서 정의하는 빌드 및 아티팩트의 식별자와 일치해야 합니다. 구문은 기본 아티팩트의 경우 *buildIdentifier*이고 보조 아티팩트의 경우 *buildIdentifier_artifactIdentifier*입니다.

예를 들어 출력 아티팩트 이름 build1의 경우 CodeBuild는 build1의 기본 아티팩트를 build1의 위치에 업로드합니다. 출력 이름 build1_artifact1의 경우 CodeBuild는 build1의 보조 아티팩트 artifact1을 build1_artifact1 위치에 업로드하는 방식으로 진행됩니다. 출력 위치를 하나만 지정하는 경우 이름은 *buildIdentifier*입니다.

JSON 파일을 생성하였으면 이제 파이프라인을 만들 수 있습니다. AWS CLI 를 사용하여 create-pipeline 명령을 실행하고 파일을 --cli-input-json 파라미터에 전달합니다. 자세한 내용은AWS CodePipeline 사용 설명서의 파이프라인 생성(CLI)을 참조하세요.

결합된 아티팩트를 사용한 배치 빌드

다음 JSON 파일을 결합된 아티팩트가 있는 배치 빌드를 생성하는 파이프라인 구조의 예로 사용하세요. CodePipeline에서 배치 빌드를 활성화하려면 configuration 객체의 BatchEnabled 파라미터를 true로 설정합니다. 빌드 아티팩트를 같은 위치에 결합하려면 configuration 객체의 CombineArtifacts 파라미터를 true로 설정합니다.

```
{
 "pipeline": {
  "roleArn": "arn:aws:iam::account-id:role/my-AWS-CodePipeline-service-role-name",
  "stages": [
    {
      "name": "Source",
      "actions": [
        {
          "inputArtifacts": [],
          "name": "Source1",
```

```
          "actionTypeId": {
            "category": "Source",
            "owner": "AWS",
            "version": "1",
            "provider": "S3"
          },
          "outputArtifacts": [
            {
              "name": "source1"
            }
          ],
          "configuration": {
            "S3Bucket": "<my-input-bucket-name>",
            "S3ObjectKey": "my-source-code-file-name.zip"
          },
          "runOrder": 1
        },
        {
          "inputArtifacts": [],
          "name": "Source2",
          "actionTypeId": {
            "category": "Source",
            "owner": "AWS",
            "version": "1",
            "provider": "S3"
          },
          "outputArtifacts": [
            {
              "name": "source2"
            }
          ],
          "configuration": {
            "S3Bucket": "<my-other-input-bucket-name>",
            "S3ObjectKey": "my-other-source-code-file-name.zip"
          },
          "runOrder": 1
        }
      ]
    },
    {
      "name": "Build",
      "actions": [
        {
          "inputArtifacts": [
```

```
            {
              "name": "source1"
            },
            {
              "name": "source2"
            }
          ],
          "name": "Build",
          "actionTypeId": {
            "category": "Build",
            "owner": "AWS",
            "version": "1",
            "provider": "CodeBuild"
          },
          "outputArtifacts": [
            {
              "name": "output1 "
            }
          ],
          "configuration": {
            "ProjectName": "my-build-project-name",
            "PrimarySource": "source1",
             "BatchEnabled": "true",
             "CombineArtifacts": "true"
          },
          "runOrder": 1
        }
      ]
    }
  ],
  "artifactStore": {
    "type": "S3",
    "location": "<AWS-CodePipeline-internal-bucket-name>"
  },
  "name": "my-pipeline-name",
  "version": 1
 }
}
```

다음은 이 파이프라인 구성과 함께 작동하는 CodeBuild buildspec 파일의 예입니다.

```
version: 0.2
batch:
```

```
  build-list:
    - identifier: build1
      env:
        compute-type: BUILD_GENERAL1_SMALL
    - identifier: build2
      env:
        compute-type: BUILD_GENERAL1_MEDIUM

phases:
  build:
    commands:
      - echo 'file' > output_file

artifacts:
  files:
    - output_file
```

배치 빌드에 대해 결합된 아티팩트가 활성화된 경우 출력은 하나만 허용됩니다. CodeBuild는 모든 빌드의 기본 아티팩트를 하나의 ZIP 파일로 결합합니다.

JSON 파일을 생성하였으면 이제 파이프라인을 만들 수 있습니다. AWS CLI 를 사용하여 create-pipeline 명령을 실행하고 파일을 `--cli-input-json` 파라미터에 전달합니다. 자세한 내용은AWS CodePipeline 사용 설명서의 파이프라인 생성(CLI)을 참조하세요.

CodePipeline/CodeBuild와 다중 입력 소스 및 출력 아티팩트 통합 샘플

AWS CodeBuild 프로젝트는 둘 이상의 입력 소스를 사용할 수 있습니다. 이에 따라 출력 아티팩트도 다수를 생성할 수 있습니다. 이 샘플은를 AWS CodePipeline 사용하여 여러 입력 소스를 사용하여 여러 출력 아티팩트를 생성하는 빌드 프로젝트를 생성하는 방법을 보여줍니다. 자세한 내용은 다중 입력 소스 및 출력 아티팩트 샘플 단원을 참조하십시오.

파이프라인의 구조를 정의하는 JSON 형식 파일을 사용한 다음와 함께 사용하여 파이프라인 AWS CLI 을 생성할 수 있습니다. 다음 JSON 파일을 입력 소스 1개 이상과 출력 아티팩트 1개 이상을 사용해 빌드를 생성하는 파이프라인 구조의 예로 사용합니다. 이번 샘플 후반에서 이 파일이 다중 입력 및 출력을 어떻게 지정하는지 확인할 수 있습니다. 자세한 내용은AWS CodePipeline 사용 설명서의 CodePipeline 파이프라인 구조 참조를 참조하세요.

```
{
 "pipeline": {
  "roleArn": "arn:aws:iam::account-id:role/my-AWS-CodePipeline-service-role-name",
  "stages": [
    {
```

```
      "name": "Source",
      "actions": [
        {
          "inputArtifacts": [],
          "name": "Source1",
          "actionTypeId": {
            "category": "Source",
            "owner": "AWS",
            "version": "1",
            "provider": "S3"
          },
          "outputArtifacts": [
            {
              "name": "source1"
            }
          ],
          "configuration": {
            "S3Bucket": "my-input-bucket-name",
            "S3ObjectKey": "my-source-code-file-name.zip"
          },
          "runOrder": 1
        },
        {
          "inputArtifacts": [],
          "name": "Source2",
          "actionTypeId": {
            "category": "Source",
            "owner": "AWS",
            "version": "1",
            "provider": "S3"
          },
          "outputArtifacts": [
            {
              "name": "source2"
            }
          ],
          "configuration": {
            "S3Bucket": "my-other-input-bucket-name",
            "S3ObjectKey": "my-other-source-code-file-name.zip"
          },
          "runOrder": 1
        }
      ]
    },
```

```
    {
      "name": "Build",
      "actions": [
        {
          "inputArtifacts": [
            {
              "name": "source1"
            },
            {
              "name": "source2"
            }
          ],
          "name": "Build",
          "actionTypeId": {
            "category": "Build",
            "owner": "AWS",
            "version": "1",
            "provider": "AWS CodeBuild"
          },
          "outputArtifacts": [
            {
              "name": "artifact1"
            },
            {
              "name": "artifact2"
            }
          ],
          "configuration": {
            "ProjectName": "my-build-project-name",
            "PrimarySource": "source1"
          },
          "runOrder": 1
        }
      ]
    }
  ],
  "artifactStore": {
    "type": "S3",
    "location": "AWS-CodePipeline-internal-bucket-name"
  },
  "name": "my-pipeline-name",
  "version": 1
 }
}
```

위 JSON 파일에서,

- 입력 소스 중 하나는 `PrimarySource`로 지정되어야 합니다. 이 소스는 CodeBuild가 buildspec 파일을 찾아서 실행하는 디렉터리를 말합니다. 키워드 `PrimarySource`는 JSON 파일 CodeBuild 단계의 `configuration` 섹션에서 기본 소스를 지정하는 데 사용됩니다.
- 각 입력 소스는 자체 디렉터리에 설치됩니다. 이 디렉터리는 기본 소스의 경우 기본 제공 환경 변수 `$CODEBUILD_SRC_DIR`에, 기타 모든 소스의 경우 `$CODEBUILD_SRC_DIR_yourInputArtifactName`에 저장됩니다. 위 샘플의 파이프라인에서는 2개의 입력 소스 디렉터리가 `$CODEBUILD_SRC_DIR`과 `$CODEBUILD_SRC_DIR_source2`입니다. 자세한 내용은 빌드 환경의 환경 변수 단원을 참조하십시오.
- 파이프라인의 JSON 파일에서 지정하는 출력 아티팩트의 이름은 buildspec 파일에서 정의하는 보조 아티팩트의 이름과 일치해야 합니다. 이 파이프라인에서 사용하는 buildspec 파일은 다음과 같습니다. 자세한 내용은 buildspec 구문 단원을 참조하십시오.

```
version: 0.2

phases:
  build:
    commands:
      - touch source1_file
      - cd $CODEBUILD_SRC_DIR_source2
      - touch source2_file

artifacts:
  files:
    - '**/*'
  secondary-artifacts:
    artifact1:
      base-directory: $CODEBUILD_SRC_DIR
      files:
        - source1_file
    artifact2:
      base-directory: $CODEBUILD_SRC_DIR_source2
      files:
        - source2_file
```

JSON 파일을 생성하였으면 이제 파이프라인을 만들 수 있습니다. AWS CLI 를 사용하여 create-pipeline 명령을 실행하고 파일을 `--cli-input-json` 파라미터에 전달합니다. 자세한 내용은AWS CodePipeline 사용 설명서의 파이프라인 생성(CLI)을 참조하세요.

AWS Config CodeBuild를 사용한 샘플

AWS Config 는 AWS 리소스 인벤토리와 이러한 리소스에 대한 구성 변경 기록을 제공합니다.는 AWS Config 이제를 AWS 리소스 AWS CodeBuild 로 지원하므로 서비스가 CodeBuild 프로젝트를 추적할 수 있습니다. 에 대한 자세한 내용은 란 무엇입니까 AWS Config?를 AWS Config참조하십시오. AWS Config 개발자 안내서의 .

AWS Config 콘솔의 리소스 인벤토리 페이지에서 CodeBuild 리소스에 대한 다음 정보를 볼 수 있습니다.

- CodeBuild 구성 변경의 타임라인
- 각 CodeBuild 프로젝트의 구성 세부 정보
- 다른 AWS 리소스와의 관계.
- CodeBuild 프로젝트의 변경 사항 목록

주제

- 에서 CodeBuild 사용 AWS Config
- 3단계: AWS Config 콘솔에서 AWS CodeBuild 데이터 보기

에서 CodeBuild 사용 AWS Config

이 주제의 절차에서는 CodeBuild 프로젝트를 설정하고 AWS Config 조회하는 방법을 보여줍니다.

주제

- 사전 조건
- 1단계: 설정 AWS Config
- 2단계: AWS CodeBuild 프로젝트 조회

사전 조건

AWS CodeBuild 프로젝트를 생성합니다. 지침은 빌드 프로젝트 생성 섹션을 참조하세요.

1단계: 설정 AWS Config

- AWS Config 설정(콘솔)
- AWS Config 설정(AWS CLI)

> **Note**
>
> 설정을 완료한 후 AWS Config 콘솔에서 AWS CodeBuild 프로젝트를 보려면 최대 10분이 걸릴 수 있습니다.

2단계: AWS CodeBuild 프로젝트 조회

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/config AWS Config 콘솔을 엽니다.
2. 리소스 인벤토리 페이지의 리소스 유형에서 AWS CodeBuild 프로젝트를 선택합니다. 아래로 스크롤하여 CodeBuild 프로젝트 확인란을 선택합니다.
3. Look up(조회)을 선택합니다.
4. CodeBuild 프로젝트 목록을 추가한 후 Config 타임라인 열에서 CodeBuild 프로젝트 이름 링크를 선택합니다.

3단계: AWS Config 콘솔에서 AWS CodeBuild 데이터 보기

리소스 인벤토리 페이지에서 리소스를 조회할 때 AWS Config 타임라인을 선택하여 CodeBuild 프로젝트에 대한 세부 정보를 볼 수 있습니다. 리소스에 대한 세부 정보 페이지에서는 구성, 관계, 리소스 변경 사항 수에 대한 정보를 확인할 수 있습니다.

페이지 상단에 있는 블록을 타임라인이라고 합니다. 타임라인은 기록이 생성된 날짜와 시간을 표시합니다.

자세한 내용은 AWS Config 개발자 안내서의 AWS Config 콘솔에서 구성 세부 정보 보기를 참조하세요.

CodeBuild의 빌드 알림 샘플

Amazon CloudWatch Events에는에 대한 지원이 내장되어 있습니다 AWS CodeBuild. CloudWatch Events는 AWS 리소스의 변경 사항을 설명하는 시스템 이벤트 스트림입니다. CloudWatch Events

를 사용하여 관심 있는 이벤트를 수행할 자동 작업과 연결하는 선언적 규칙을 작성합니다. 이 샘플은 Amazon CloudWatch Events와 Amazon Simple Notification Service(SNS)를 사용하여 빌드가 성공하거나, 실패하거나, 한 빌드 단계에서 다른 빌드 단계로 진행하거나, 이러한 이벤트가 조합될 때마다 구독자에게 빌드 알림을 보냅니다.

> **Important**
>
> 이 샘플을 실행하면 AWS 계정에 요금이 부과될 수 있습니다. 여기에는 CodeBuild 및 Amazon CloudWatch 및 Amazon SNS와 관련된 AWS 리소스 및 작업에 대해 발생할 수 있는 요금이 포함됩니다. 자세한 내용은 CodeBuild 요금, Amazon CloudWatch 요금 및 Amazon SNS 요금을 참조하세요.

주제

- 빌드 알림 샘플 실행
- 빌드 알림 입력 형식 참조

빌드 알림 샘플 실행

다음 절차에 따라 빌드 알림 샘플을 실행합니다.

이 샘플을 실행하려면

1. Amazon SNS에서 이 샘플에 사용할 주제를 설정하고 구독한 경우 4단계로 건너뜁니다. 그렇지 않으면 AWS 루트 계정 또는 관리자 사용자 대신 IAM 사용자를 사용하여 Amazon SNS로 작업하는 경우 사용자(또는 사용자가 연결된 IAM 그룹)에 다음 문(*### BEGIN ADDING STATEMENT HERE ###*과 *### END ADDING STATEMENT HERE ###* 사이)을 추가합니다. AWS 루트 계정을 사용하는 것은 권장되지 않습니다. 이 명령문을 사용하면 Amazon SNS의 주제로의 알림 전송을 보고, 생성하고, 구독하고, 테스트할 수 있습니다. 간결하게 나타내고 명령문 추가 위치를 알 수 있도록 줄임표(`...`)가 사용되었습니다. 어떤 명령문도 제거하지 않아야 하며, 이러한 줄임표는 기존 정책에 입력하지 않아야 합니다.

   JSON

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

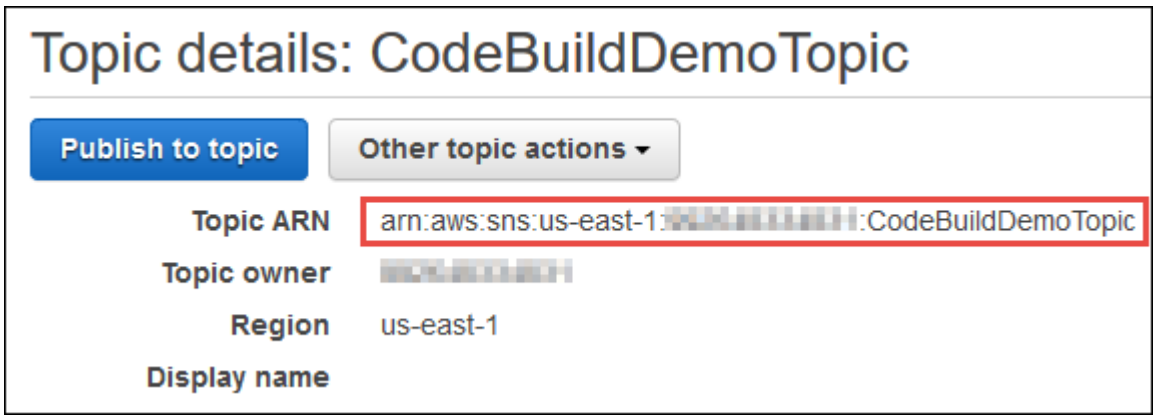
```
        {
            "Effect": "Allow",
            "Action": [
                "sns:CreateTopic",
                "sns:GetTopicAttributes",
                "sns:List*",
                "sns:Publish",
                "sns:SetTopicAttributes",
                "sns:Subscribe"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 이 정책을 수정하는 IAM 엔터티에는 정책을 수정하는 IAM의 권한이 있어야 합니다. 자세한 내용은 고객 관리형 정책 편집 또는 IAM 사용 설명서의 인라인 정책 작업(콘솔)에서 “그룹, 사용자 또는 역할에 대한 인라인 정책을 편집 또는 삭제하려면” 섹션을 참조하세요.

2. Amazon SNS에서 주제를 생성하거나 식별합니다. CloudWatch Events를 AWS CodeBuild 사용하여 Amazon SNS를 통해이 주제에 빌드 알림을 보냅니다.

   주제를 생성하려면 다음과 같이 합니다.

   1. https://console.aws.amazon.com/sns Amazon SNS 콘솔을 엽니다.
   2. 주제 생성을 선택합니다.
   3. 새로운 주제 생성의 주제 이름에 주제 이름(예: **CodeBuildDemoTopic**)을 입력합니다. (다른 이름을 선택하는 경우 이 샘플 전체에서 해당 이름으로 바꿉니다.)
   4. 주제 생성을 선택합니다.
   5. 주제 세부 정보: CodeBuildDemoTopic 페이지에서 주제 ARN 값을 복사합니다. 다음 단계에서 이 값을 사용합니다.

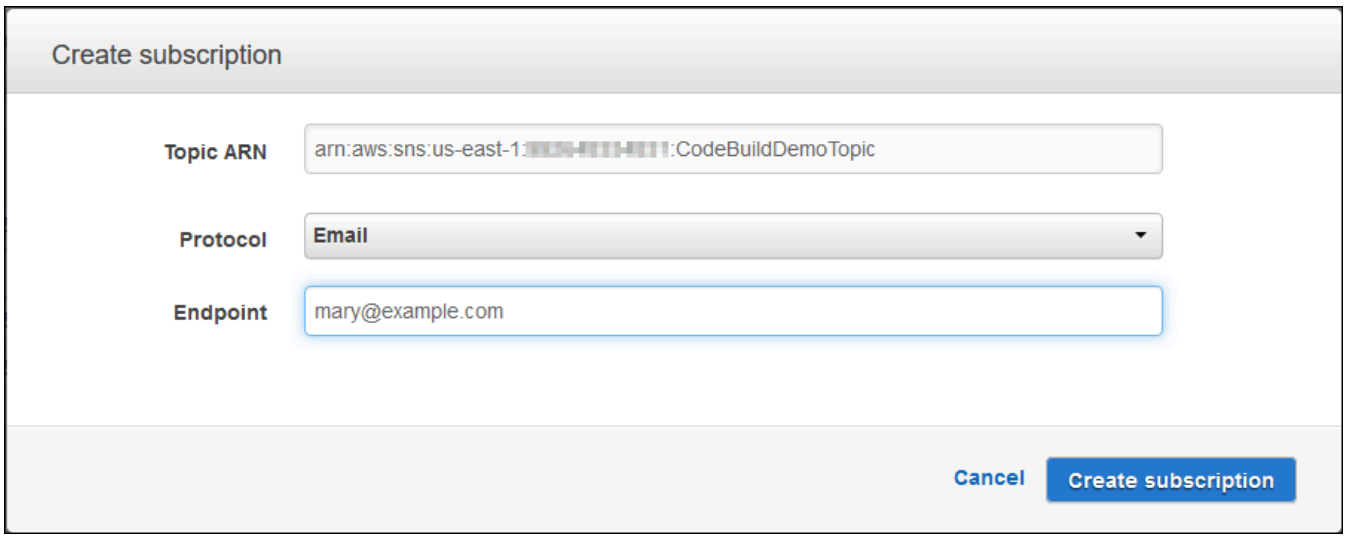


자세한 내용은 Amazon SNS 개발자 안내서의 주제 생성을 참조하세요.

3. 한 명 이상의 수신자가 주제를 구독하여 이메일 알림을 수신하게 합니다.

   수신자가 주제를 구독하게 하려면 다음과 같이 합니다.

   1. 이전 단계에서 Amazon SNS 콘솔을 연 상태에서 탐색 창에서 구독을 선택한 다음, 구독 생성을 선택합니다.
   2. 구독 생성의 주제 ARN에 이전 단계에서 복사한 주제 ARN을 붙여 넣습니다.
   3. 프로토콜에서 이메일을 선택합니다.
   4. 엔드포인트에 수신자의 전체 이메일 주소를 입력합니다.



   5. 구독 생성을 선택합니다.

6. Amazon SNS가 수신자에게 구독 확인 이메일을 보냅니다. 수신자는 알림을 수신하려면 구독 확인 이메일에서 구독 확인을 선택해야 합니다. 수신자가 링크를 클릭한 후 구독에 성공하면 Amazon SNS가 해당 수신자의 웹 브라우저에 확인 메시지를 표시합니다.

자세한 내용은 Amazon SNS 개발자 가이드의 주제 구독을 참조하세요.

4. AWS 루트 계정 또는 관리자 사용자 대신 사용자를 사용하여 CloudWatch Events를 사용하는 경우 사용자(또는 사용자가 연결된 IAM 그룹)에 다음 문(*### BEGIN ADDING STATEMENT HERE ###*과 *### END ADDING STATEMENT HERE ###* 사이)을 추가합니다. AWS 루트 계정을 사용하는 것은 권장되지 않습니다. 이 명령문은 사용자가 CloudWatch Events에서 작업할 수 있도록 하는 데 사용됩니다. 간결하게 나타내고 명령문 추가 위치를 알 수 있도록 줄임표(`...`)가 사용되었습니다. 어떤 명령문도 제거하지 않아야 하며, 이러한 줄임표는 기존 정책에 입력하지 않아야 합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "events:*",
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::*:role/Service*"
        }
    ]
}
```

> **Note**
>
> 이 정책을 수정하는 IAM 엔터티에는 정책을 수정하는 IAM의 권한이 있어야 합니다. 자세한 내용은 고객 관리형 정책 편집 또는 IAM 사용 설명서의 인라인 정책 작업(콘솔)에서 “그룹, 사용자 또는 역할에 대한 인라인 정책을 편집 또는 삭제하려면” 섹션을 참조하세요.

5. CloudWatch Events에서 규칙을 생성합니다. 이 작업을 수행하려면 https://console.aws.amazon.com/cloudwatch에서 CloudWatch 콘솔을 엽니다.
6. 탐색 창의 이벤트에서 규칙을 선택한 다음 규칙 생성을 선택합니다.
7. 1단계: 규칙 생성 페이지에서 이벤트 패턴 및 서비스별 이벤트와 일치시킬 이벤트 패턴 빌드가 선택된 상태여야 합니다.
8. 서비스 이름에서 CodeBuild를 선택합니다. 이벤트 유형에서 모든 이벤트가 이미 선택된 상태여야 합니다.
9. 이벤트 패턴 미리 보기에 다음 코드가 표시되어야 합니다.

```
{
  "source": [
    "aws.codebuild"
  ]
}
```

10. 편집을 선택하여 이벤트 패턴 미리 보기를 다음 두 가지 규칙 패턴 중 하나로 교체합니다.

    이 첫 번째 규칙 패턴은 빌드가 시작되거나 완료될 때 AWS CodeBuild에 지정된 빌드 프로젝트에 대해 이벤트를 트리거합니다.

```
{
  "source": [
    "aws.codebuild"
  ],
  "detail-type": [
    "CodeBuild Build State Change"
  ],
  "detail": {
    "build-status": [
      "IN_PROGRESS",
      "SUCCEEDED",
      "FAILED",
      "STOPPED"
    ],
    "project-name": [
      "my-demo-project-1",
      "my-demo-project-2"
    ]
  }
}
```

위의 규칙에서 다음과 같이 코드를 변경하세요.

- 빌드가 시작되거나 완료될 때 이벤트를 트리거하려면 `build-status` 어레이에 표시된 모든 값을 그대로 두거나 `build-status` 어레이를 모두 제거합니다.
- 빌드가 완료될 때만 이벤트를 트리거하려면 `build-status` 배열에서 `IN_PROGRESS`를 제거합니다.
- 빌드가 시작될 때만 이벤트를 트리거하려면 `build-status` 배열에서 `IN_PROGRESS`를 제외한 모든 값을 제거합니다.
- 모든 빌드 프로젝트에 대해 이벤트를 트리거하려면 `project-name` 배열을 모두 제거합니다.
- 개별 빌드 프로젝트에 대해서만 이벤트를 트리거하려면 `project-name` 배열에 각 빌드 프로젝트의 이름을 지정합니다.

이 두 번째 규칙 패턴은 AWS CodeBuild에 지정된 빌드 프로젝트에 대해 빌드가 한 빌드 단계에서 다른 빌드 단계로 이동할 때마다 이벤트를 트리거합니다.

```
{
  "source": [
    "aws.codebuild"
  ],
  "detail-type": [
    "CodeBuild Build Phase Change"
  ],
  "detail": {
    "completed-phase": [
      "SUBMITTED",
      "PROVISIONING",
      "DOWNLOAD_SOURCE",
      "INSTALL",
      "PRE_BUILD",
      "BUILD",
      "POST_BUILD",
      "UPLOAD_ARTIFACTS",
      "FINALIZING"
    ],
    "completed-phase-status": [
      "TIMED_OUT",
      "STOPPED",
      "FAILED",
      "SUCCEEDED",
```

```
      "FAULT",
      "CLIENT_ERROR"
    ],
    "project-name": [
      "my-demo-project-1",
      "my-demo-project-2"
    ]
  }
}
```

위의 규칙에서 다음과 같이 코드를 변경하세요.

- 모든 빌드 단계 변경(각 빌드에 대해 최대 9개의 알림을 보낼 수 있음)에 대해 이벤트를 트리거하려면 `completed-phase` 어레이에 표시된 모든 값을 그대로 두거나 `completed-phase` 어레이를 모두 제거합니다.
- 개별 빌드 단계 변경에 대해서만 이벤트를 트리거하려면 이벤트를 트리거하지 않으려는 `completed-phase` 배열의 각 빌드 단계 이름을 제거합니다.
- 모든 빌드 단계 상태 변경에 대한 이벤트를 트리거하려면 `completed-phase-status` 배열에 표시된 모든 값을 그대로 두거나 `completed-phase-status` 배열을 모두 제거합니다.
- 개별 빌드 단계 상태 변경에 대해서만 이벤트를 트리거하려면 이벤트를 트리거하지 않으려는 `completed-phase-status` 배열의 각 빌드 단계 이름을 제거합니다.
- 모든 빌드 프로젝트에 대한 이벤트를 트리거하려면 `project-name` 배열을 제거합니다.
- 개별 빌드 프로젝트에 대한 이벤트만 트리거하려면 `project-name` 배열에 각 빌드 프로젝트의 이름을 지정합니다.

이벤트 패턴에 대한 자세한 내용은 Amazon EventBridge 사용 설명서에서 이벤트 패턴을 참조하세요.

이벤트 패턴을 사용한 필터링에 대한 자세한 내용은 Amazon EventBridge 사용 설명서에서 이벤트 패턴을 사용한 콘텐츠 기반 필터링을 참조하세요.

> **Note**
>
> 빌드 상태 변경 및 빌드 단계 변경에 대한 이벤트를 트리거하려는 경우, 빌드 상태 변경에 대한 규칙 및 빌드 단계 변경을 위한 규칙이라는 두 가지 별도의 규칙을 생성해야 합니다. 두 가지 규칙을 단일 규칙으로 결합하려고 하면 결합된 해당 규칙으로 인해 예기치 않은 결과가 발생하거나 작업이 모두 중단될 수 있습니다.

코드 교체를 완료하면 저장을 선택합니다.

11. 대상(Targets)에서 대상 추가(Add target)를 선택합니다.
12. 대상 목록에서 SNS 주제를 선택합니다.
13. 주제에서 이전에 식별했거나 생성한 주제를 선택합니다.
14. 입력 구성을 확장한 후 입력 변환기를 선택합니다.
15. 입력 경로 상자에 다음 입력 경로 중 하나를 입력합니다.

    `CodeBuild Build State Change`의 `detail-type` 값을 사용하는 규칙에 대해 다음을 입력합니다.

    ```
    {"build-id":"$.detail.build-id","project-name":"$.detail.project-name","build-status":"$.detail.build-status"}
    ```

    `CodeBuild Build Phase Change`의 `detail-type` 값을 사용하는 규칙에 대해 다음을 입력합니다.

    ```
    {"build-id":"$.detail.build-id","project-name":"$.detail.project-name","completed-phase":"$.detail.completed-phase","completed-phase-status":"$.detail.completed-phase-status"}
    ```

    다른 유형의 정보를 보려면 빌드 알림 입력 형식 참조 섹션을 참조하세요.
16. 입력 템플릿 상자에 다음 입력 템플릿 중 하나를 입력합니다.

    `CodeBuild Build State Change`의 `detail-type` 값을 사용하는 규칙에 대해 다음을 입력합니다.

    ```
    "Build '<build-id>' for build project '<project-name>' has reached the build status of '<build-status>'."
    ```

    `CodeBuild Build Phase Change`의 `detail-type` 값을 사용하는 규칙에 대해 다음을 입력합니다.

    ```
    "Build '<build-id>' for build project '<project-name>' has completed the build phase of '<completed-phase>' with a status of '<completed-phase-status>'."
    ```
17. 세부 정보 구성을 선택합니다.

18. 2단계: 규칙 세부 정보 구성 페이지에 이름 및 선택적인 설명을 입력합니다. 상태에 대해 사용을 선택한 상태로 둡니다.
19. 규칙 생성을 선택합니다.
20. 빌드 프로젝트를 생성하고, 빌드를 실행하고, 빌드 정보를 확인합니다.
21. CodeBuild에서 현재 빌드 알림을 성공적으로 보내고 있는지 확인합니다. 예를 들어 빌드 알림 이메일이 현재 받은 편지함에 있는지 확인합니다.

규칙의 동작을 변경하려면 CloudWatch 콘솔에서 변경하려는 규칙을 선택한 후 작업, 편집을 차례로 선택합니다. 규칙을 변경하고 구성 세부 정보를 선택한 후 규칙 업데이트를 선택합니다.

규칙을 사용하여 빌드 알림을 보내는 것을 중지하려면 CloudWatch 콘솔에서 사용을 중지하려는 규칙을 선택한 후 작업, 비활성을 차례로 선택합니다.

규칙을 모두 삭제하려면 CloudWatch 콘솔에서 삭제하려는 규칙을 선택한 후 작업, 삭제를 차례로 선택합니다.

빌드 알림 입력 형식 참조

CloudWatch는 JSON 형식으로 알림을 제공합니다.

빌드 상태 변경 알림은 다음 형식을 사용합니다.

```
{
  "version": "0",
  "id": "c030038d-8c4d-6141-9545-00ff7b7153EX",
  "detail-type": "CodeBuild Build State Change",
  "source": "aws.codebuild",
  "account": "123456789012",
  "time": "2017-09-01T16:14:28Z",
  "region": "us-west-2",
  "resources":[
    "arn:aws:codebuild:us-west-2:123456789012:build/my-sample-project:8745a7a9-
c340-456a-9166-edf953571bEX"
  ],
  "detail":{
    "build-status": "SUCCEEDED",
    "project-name": "my-sample-project",
    "build-id": "arn:aws:codebuild:us-west-2:123456789012:build/my-sample-
project:8745a7a9-c340-456a-9166-edf953571bEX",
    "additional-information": {
      "artifact": {
```

```
        "md5sum": "da9c44c8a9a3cd4b443126e823168fEX",
        "sha256sum":
 "6ccc2ae1df9d155ba83c597051611c42d60e09c6329dcb14a312cecc0a8e39EX",
        "location": "arn:aws:s3:::codebuild-123456789012-output-bucket/my-output-
artifact.zip"
      },
      "environment": {
        "image": "aws/codebuild/standard:5.0",
        "privileged-mode": false,
        "compute-type": "BUILD_GENERAL1_SMALL",
        "type": "LINUX_CONTAINER",
        "environment-variables": []
      },
      "timeout-in-minutes": 60,
      "build-complete": true,
      "initiator": "MyCodeBuildDemoUser",
      "build-start-time": "Sep 1, 2017 4:12:29 PM",
      "source": {
        "location": "codebuild-123456789012-input-bucket/my-input-artifact.zip",
        "type": "S3"
      },
      "logs": {
        "group-name": "/aws/codebuild/my-sample-project",
        "stream-name": "8745a7a9-c340-456a-9166-edf953571bEX",
        "deep-link": "https://console.aws.amazon.com/cloudwatch/home?region=us-
west-2#logEvent:group=/aws/codebuild/my-sample-project;stream=8745a7a9-c340-456a-9166-
edf953571bEX"
      },
      "phases": [
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:12:29 PM",
          "end-time": "Sep 1, 2017 4:12:29 PM",
          "duration-in-seconds": 0,
          "phase-type": "SUBMITTED",
          "phase-status": "SUCCEEDED"
        },
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:12:29 PM",
          "end-time": "Sep 1, 2017 4:13:05 PM",
          "duration-in-seconds": 36,
          "phase-type": "PROVISIONING",
          "phase-status": "SUCCEEDED"
```

```
        },
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:13:05 PM",
          "end-time": "Sep 1, 2017 4:13:10 PM",
          "duration-in-seconds": 4,
          "phase-type": "DOWNLOAD_SOURCE",
          "phase-status": "SUCCEEDED"
        },
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:13:10 PM",
          "end-time": "Sep 1, 2017 4:13:10 PM",
          "duration-in-seconds": 0,
          "phase-type": "INSTALL",
          "phase-status": "SUCCEEDED"
        },
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:13:10 PM",
          "end-time": "Sep 1, 2017 4:13:10 PM",
          "duration-in-seconds": 0,
          "phase-type": "PRE_BUILD",
          "phase-status": "SUCCEEDED"
        },
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:13:10 PM",
          "end-time": "Sep 1, 2017 4:14:21 PM",
          "duration-in-seconds": 70,
          "phase-type": "BUILD",
          "phase-status": "SUCCEEDED"
        },
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:14:21 PM",
          "end-time": "Sep 1, 2017 4:14:21 PM",
          "duration-in-seconds": 0,
          "phase-type": "POST_BUILD",
          "phase-status": "SUCCEEDED"
        },
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:14:21 PM",
```

```
          "end-time": "Sep 1, 2017 4:14:21 PM",
          "duration-in-seconds": 0,
          "phase-type": "UPLOAD_ARTIFACTS",
          "phase-status": "SUCCEEDED"
        },
         {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:14:21 PM",
          "end-time": "Sep 1, 2017 4:14:26 PM",
          "duration-in-seconds": 4,
          "phase-type": "FINALIZING",
          "phase-status": "SUCCEEDED"
        },
        {
          "start-time": "Sep 1, 2017 4:14:26 PM",
          "phase-type": "COMPLETED"
        }
      ]
    },
    "current-phase": "COMPLETED",
    "current-phase-context": "[]",
    "version": "1"
  }
}
```

빌드 단계 변경 알림은 다음 형식을 사용합니다.

```
{
  "version": "0",
  "id": "43ddc2bd-af76-9ca5-2dc7-b695e15adeEX",
  "detail-type": "CodeBuild Build Phase Change",
  "source": "aws.codebuild",
  "account": "123456789012",
  "time": "2017-09-01T16:14:21Z",
  "region": "us-west-2",
  "resources":[
    "arn:aws:codebuild:us-west-2:123456789012:build/my-sample-project:8745a7a9-
c340-456a-9166-edf953571bEX"
  ],
  "detail":{
    "completed-phase": "COMPLETED",
    "project-name": "my-sample-project",
```

```
    "build-id": "arn:aws:codebuild:us-west-2:123456789012:build/my-sample-
project:8745a7a9-c340-456a-9166-edf953571bEX",
    "completed-phase-context": "[]",
    "additional-information": {
      "artifact": {
        "md5sum": "da9c44c8a9a3cd4b443126e823168fEX",
        "sha256sum":
 "6ccc2ae1df9d155ba83c597051611c42d60e09c6329dcb14a312cecc0a8e39EX",
        "location": "arn:aws:s3:::codebuild-123456789012-output-bucket/my-output-
artifact.zip"
      },
      "environment": {
        "image": "aws/codebuild/standard:5.0",
        "privileged-mode": false,
        "compute-type": "BUILD_GENERAL1_SMALL",
        "type": "LINUX_CONTAINER",
        "environment-variables": []
      },
      "timeout-in-minutes": 60,
      "build-complete": true,
      "initiator": "MyCodeBuildDemoUser",
      "build-start-time": "Sep 1, 2017 4:12:29 PM",
      "source": {
        "location": "codebuild-123456789012-input-bucket/my-input-artifact.zip",
        "type": "S3"
      },
      "logs": {
        "group-name": "/aws/codebuild/my-sample-project",
        "stream-name": "8745a7a9-c340-456a-9166-edf953571bEX",
        "deep-link": "https://console.aws.amazon.com/cloudwatch/home?region=us-
west-2#logEvent:group=/aws/codebuild/my-sample-project;stream=8745a7a9-c340-456a-9166-
edf953571bEX"
      },
      "phases": [
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:12:29 PM",
          "end-time": "Sep 1, 2017 4:12:29 PM",
          "duration-in-seconds": 0,
          "phase-type": "SUBMITTED",
          "phase-status": "SUCCEEDED"
        },
        {
          "phase-context": [],
```

```
      "start-time": "Sep 1, 2017 4:12:29 PM",
      "end-time": "Sep 1, 2017 4:13:05 PM",
      "duration-in-seconds": 36,
      "phase-type": "PROVISIONING",
      "phase-status": "SUCCEEDED"
    },
    {
      "phase-context": [],
      "start-time": "Sep 1, 2017 4:13:05 PM",
      "end-time": "Sep 1, 2017 4:13:10 PM",
      "duration-in-seconds": 4,
      "phase-type": "DOWNLOAD_SOURCE",
      "phase-status": "SUCCEEDED"
    },
    {
      "phase-context": [],
      "start-time": "Sep 1, 2017 4:13:10 PM",
      "end-time": "Sep 1, 2017 4:13:10 PM",
      "duration-in-seconds": 0,
      "phase-type": "INSTALL",
      "phase-status": "SUCCEEDED"
    },
    {
      "phase-context": [],
      "start-time": "Sep 1, 2017 4:13:10 PM",
      "end-time": "Sep 1, 2017 4:13:10 PM",
      "duration-in-seconds": 0,
      "phase-type": "PRE_BUILD",
      "phase-status": "SUCCEEDED"
    },
    {
      "phase-context": [],
      "start-time": "Sep 1, 2017 4:13:10 PM",
      "end-time": "Sep 1, 2017 4:14:21 PM",
      "duration-in-seconds": 70,
      "phase-type": "BUILD",
      "phase-status": "SUCCEEDED"
    },
    {
      "phase-context": [],
      "start-time": "Sep 1, 2017 4:14:21 PM",
      "end-time": "Sep 1, 2017 4:14:21 PM",
      "duration-in-seconds": 0,
      "phase-type": "POST_BUILD",
```

```
          "phase-status": "SUCCEEDED"
        },
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:14:21 PM",
          "end-time": "Sep 1, 2017 4:14:21 PM",
          "duration-in-seconds": 0,
          "phase-type": "UPLOAD_ARTIFACTS",
          "phase-status": "SUCCEEDED"
        },
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:14:21 PM",
          "end-time": "Sep 1, 2017 4:14:26 PM",
          "duration-in-seconds": 4,
          "phase-type": "FINALIZING",
          "phase-status": "SUCCEEDED"
        },
        {
          "start-time": "Sep 1, 2017 4:14:26 PM",
          "phase-type": "COMPLETED"
        }
      ]
    },
    "completed-phase-status": "SUCCEEDED",
    "completed-phase-duration-seconds": 4,
    "version": "1",
    "completed-phase-start": "Sep 1, 2017 4:14:21 PM",
    "completed-phase-end": "Sep 1, 2017 4:14:26 PM"
  }
}
```

CodeBuild의 빌드 배지 샘플

AWS CodeBuild 는 이제 프로젝트의 최신 빌드 상태를 표시하는 동적으로 생성된 임베딩 가능한 이미지(배지)를 제공하는 빌드 배지 사용을 지원합니다. 이 이미지는 CodeBuild 프로젝트용으로 생성된 URL을 통해 공개적으로 액세스할 수 있습니다. 이를 통해 누구든지 CodeBuild 프로젝트의 상태를 볼 수 있습니다. 빌드 배지에는 보안 정보가 포함되어 있지 않으므로 인증이 필요하지 않습니다.

주제

- 활성화된 빌드 배지를 사용하여 빌드 프로젝트 생성

- AWS CodeBuild 빌드 배지 액세스
- CodeBuild 빌드 배지 게시
- CodeBuild 배지 상태

활성화된 빌드 배지를 사용하여 빌드 프로젝트 생성

다음 절차 중 하나를 사용하여 빌드 배지가 활성화된 빌드 프로젝트를 생성합니다. AWS CLI 또는를 사용할 수 있습니다 AWS Management Console.

활성화된 빌드 배지를 사용하여 빌드 프로젝트를 생성하려면(AWS CLI)

- 빌드 프로젝트 생성에 대한 자세한 내용은 빌드 프로젝트 생성(AWS CLI) 섹션을 참조하십시오. AWS CodeBuild 프로젝트에 빌드 배지를 포함하려면 *badgeEnabled* 값을 `true`로 지정해야 합니다.

활성화된 빌드 배지를 사용하여 빌드 프로젝트를 생성하려면(콘솔)

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. CodeBuild 정보 페이지가 나타나면 빌드 프로젝트 생성을 선택합니다. 그렇지 않을 경우, 탐색 창에서 빌드를 확장한 후 빌드 프로젝트를 선택하고 빌드 프로젝트 생성을 선택합니다.
3. 프로젝트 이름에 이 빌드 프로젝트의 이름을 입력합니다. 빌드 프로젝트 이름은 각 AWS 계정에서 고유해야 합니다. 또한 선택에 따라 빌드 프로젝트에 대한 설명을 포함하여 다른 사용자가 이 프로젝트의 용도를 이해하도록 도울 수 있습니다.
4. 소스의 소스 공급자에서, 소스 코드 공급자 유형을 선택한 다음, 다음 중 하나를 수행합니다.

> **Note**
>
> CodeBuild는 Amazon S3 소스 공급자에게 빌드 배지를 지원하지 않습니다. 는 아티팩트 전송에 Amazon S3를 AWS CodePipeline 사용하기 때문에 CodePipeline에서 생성된 파이프라인의 일부인 빌드 프로젝트에는 빌드 배지가 지원되지 않습니다.

- CodeCommit을 선택한 후 리포지토리에서 해당 리포지토리의 이름을 선택합니다. 프로젝트의 빌드 상태를 표시하고 삽입 가능하게 하려면 Enable build badge(빌드 배치 활성화)를 선택합니다.

- [GitHub]를 선택했다면, GitHub와 연결(다시 연결)하는 지침을 따르십시오. GitHub 애플리케이션 권한 부여 페이지의 조직 액세스에서 액세스 AWS CodeBuild 하려는 각 리포지토리 옆에 있는 액세스 요청을 선택합니다. Authorize application(애플리케이션 권한 부여)을 선택한 후 AWS CodeBuild 콘솔로 돌아가서 리포지토리에서 소스 코드를 포함하는 리포지토리의 이름을 선택합니다. 프로젝트의 빌드 상태를 표시하고 삽입 가능하게 하려면 Enable build badge(빌드 배치 활성화)를 선택합니다.
- [Bitbucket]을 선택했다면, Bitbucket과 연결(다시 연결)하는 지침을 따르십시오. [Confirm access to your account] 페이지의 [Organization access]에서 [Grant access]를 선택합니다. 액세스 권한 부여를 선택한 후 AWS CodeBuild 콘솔의 리포지토리에서 소스 코드가 포함된 리포지토리의 이름을 선택합니다. 프로젝트의 빌드 상태를 표시하고 삽입 가능하게 하려면 Enable build badge(빌드 배치 활성화)를 선택합니다.

> **Important**
>
> 프로젝트 소스를 업데이트하면 프로젝트 빌드 배지의 정확성에 영향을 미칠 수 있습니다.

5. 환경에서 다음과 같이 합니다.

   [Environment image]에서 다음 중 하나를 수행합니다.

   - 에서 관리하는 Docker 이미지를 사용하려면 관리형 이미지를 AWS CodeBuild선택한 다음 운영 체제, 런타임, 이미지(Image) 및 이미지 버전에서 선택합니다. 사용 가능한 경우 환경 유형에서 항목을 선택합니다.
   - 다른 도커 이미지를 사용하려면 사용자 지정 이미지를 선택합니다. 환경 유형에서 ARM, Linux, Linux GPU 또는 Windows를 선택합니다. Other registry(다른 레지스트리)를 선택한 경우 External registry URL(외부 레지스트리 URL)에 Docker Hub의 도커 이미지 이름 및 태그를 *docker repository*/*docker image name* 형식으로 입력합니다. Amazon ECR을 선택하는 경우 Amazon ECR 리포지토리와 Amazon ECR 이미지를 사용하여 AWS 계정에서 도커 이미지를 선택합니다.
   - 프라이빗 도커 이미지를 사용하려면 사용자 지정 이미지를 선택합니다. 환경 유형에서 ARM, Linux, Linux GPU 또는 Windows를 선택합니다. Image registry(이미지 레지스트리)에서 Other registry(다른 레지스트리)를 선택한 다음 프라이빗 도커 이미지에 대한 보안 인증 정보의 ARN을 입력합니다. 보안 인증은 Secrets Manager에서 생성됩니다. 자세한 내용은 AWS Secrets Manager사용 설명서의 AWS Secrets Manager 이란? 섹션을 참조하세요.

6. 서비스 역할에서 다음 중 하나를 수행합니다.

- CodeBuild 서비스 역할이 없는 경우 새 서비스 역할을 선택합니다. 역할 이름에 새 역할의 이름을 입력합니다.
- CodeBuild 서비스 역할이 있는 경우 기존 서비스 역할을 선택합니다. 역할 ARN에서 서비스 역할을 선택합니다.

> **Note**
>
> 콘솔을 사용하여 빌드 프로젝트를 생성하거나 업데이트하는 경우, 이와 동시에 CodeBuild 서비스 역할을 만들 수 있습니다. 기본적으로 역할은 해당 빌드 프로젝트에서만 작동합니다. 콘솔을 사용하여 이 서비스 역할을 다른 빌드 프로젝트와 연결하는 경우 다른 빌드 프로젝트에서 작동하도록 역할이 업데이트됩니다. 하나의 서비스 역할은 최대 10개의 빌드 프로젝트에서 작동할 수 있습니다.

7. Buildspec에서 다음 중 하나를 수행합니다.

   - buildspec 파일 사용을 선택하여 소스 코드 루트 디렉터리에 있는 buildspec.yml 파일을 사용합니다.
   - 빌드 명령 삽입을 선택하여 콘솔에서 빌드 명령을 삽입합니다.

   자세한 내용은 buildspec 참조 단원을 참조하십시오.

8. 결과물의 유형에서 다음 중 하나를 수행합니다.

   - 빌드 출력 아티팩트를 생성하지 않으려면 No artifacts(아티팩트 없음)를 선택합니다.
   - S3 버킷에 빌드 출력을 저장하려면 Amazon S3를 선택하고 다음 작업을 수행합니다.
     - 빌드 출력 ZIP 파일이나 폴더에 프로젝트 이름을 사용하려는 경우 이름을 비워 둡니다. 그렇지 않으면 이름을 입력합니다. 기본적으로 결과물 이름은 프로젝트의 이름입니다. 다른 이름을 사용하려면 결과물 이름 상자에 해당 이름을 입력합니다. ZIP 파일을 출력하려면 zip 확장명을 포함시킵니다.
     - [Bucket name]에서 출력 버킷의 이름을 선택합니다.
     - 이 절차의 앞부분에서 빌드 명령 삽입을 선택한 경우 출력 파일에 빌드 출력 ZIP 파일 또는 폴더에 넣으려는 빌드의 파일 위치를 입력합니다. 위치가 여러 개인 경우 각 위치를 쉼표로 구분합니다(예: `appspec.yml, target/my-app.jar`). 자세한 내용은 buildspec 구문의 `files` 설명을 참조하십시오.

9. 추가 구성을 확장하고 적절한 옵션을 선택합니다.

10. 빌드 프로젝트 생성을 선택합니다. 검토 페이지에서 빌드 시작을 선택하여 빌드를 실행합니다.

AWS CodeBuild 빌드 배지 액세스

AWS CodeBuild 콘솔 또는를 사용하여 빌드 배지 AWS CLI 에 액세스할 수 있습니다.

- CodeBuild 콘솔에서 빌드 프로젝트 목록에 있는 이름 열에서 빌드 프로젝트에 해당하는 링크를 선택합니다. 빌드 프로젝트: ***project-name*** 페이지의 구성에서 배지 URL 복사를 선택합니다. 자세한 내용은 빌드 프로젝트 세부 정보 보기(콘솔) 단원을 참조하십시오.
- 에서 `batch-get-projects` 명령을 AWS CLI실행합니다. 빌드 배지 URL은 출력의 프로젝트 환경 세부 정보 섹션에 포함됩니다. 자세한 내용은 빌드 프로젝트 세부 정보 보기(AWS CLI) 단원을 참조하십시오.

빌드 배지 요청 URL은 일반적인 기본 분기로 생성되지만, 빌드를 실행하는 데 사용한 소스 리포지토리의 어떤 분기도 지정할 수 있습니다. 예제:

```
https://codebuild.us-east-1.amazon.com/badges?uuid=...&branch=<branch>
```

`branch` 파라미터를 배지 URL의 `tag` 파라미터를 대체하여 소스 리포지토리의 태그를 지정할 수도 있습니다. 예제:

```
https://codebuild.us-east-1.amazon.com/badges?uuid=...&tag=<tag>
```

CodeBuild 빌드 배지 게시

마크다운 이미지의 빌드 배지 URL을 사용하여 마크다운 파일에 최신 빌드의 상태를 표시할 수 있습니다. 이 기능은 소스 리포지토리(예: GitHub 또는 CodeCommit) 의 readme.md 파일에 최신 빌드의 상태를 표시하는 데 유용합니다. 예제:

```
![](<build badge URL>)
```

CodeBuild 배지 상태

CodeBuild 빌드 배지는 다음 상태 중 하나를 가질 수 있습니다.

- PASSING 특정 분기의 최신 빌드가 전달되었습니다.
- FAILING 특정 분기의 최신 빌드가 시간 초과, 실패, 오류 또는 중지되었습니다.

- IN_PROGRESS 특정 분기의 최신 빌드가 진행 중입니다.
- UNKNOWN 프로젝트가 아직 특정 분기 또는 전부에 대한 빌드를 실행하지 않았습니다. 또한 배지 빌드 기능이 비활성화되었을 수 있습니다.

'를 사용한 테스트 보고서 AWS CLI' 샘플

buildspec 파일에 지정한 테스트는 빌드 중에 실행됩니다. 이 샘플은를 사용하여 CodeBuild의 빌드에 테스트를 AWS CLI 통합하는 방법을 보여줍니다. JUnit을 사용하여 단위 테스트를 만들거나, 다른 도구를 사용하여 구성 테스트를 만들 수 있습니다. 그런 다음 테스트 결과를 평가하여 문제를 해결하거나 애플리케이션을 최적화할 수 있습니다.

CodeBuild API 또는 AWS CodeBuild 콘솔을 사용하여 테스트 결과에 액세스할 수 있습니다. 이 샘플에서는 테스트 결과를 S3 버킷으로 내보내도록 보고서를 구성하는 방법을 보여줍니다.

주제

- 테스트 보고서 샘플 실행

테스트 보고서 샘플 실행

다음 단계에 따라 테스트 보고서 샘플을 실행합니다.

주제

- 사전 조건
- 1단계: 보고서 그룹 생성
- 2단계: 보고서 그룹을 사용하여 프로젝트 구성
- 3단계: 보고서 실행 및 결과 보기

사전 조건

- 테스트 케이스를 만듭니다. 이 샘플은 샘플 테스트 보고서에 포함할 테스트 케이스가 있다는 것을 전제로 작성된 것입니다. buildspec 파일에서 테스트 파일의 위치를 지정합니다.

  지원되는 테스트 보고서 파일 형식은 다음과 같습니다.

  - Cucumber JSON(.json)
  - JUnit XML(.xml)

- NUnit XML(.xml)
- NUnit3 XML(.xml)
- TestNG XML(.xml)
- Visual Studio TRX(.trx)
- Visual Studio TRX XML(.xml)

이러한 형식 중 하나로 보고서 파일을 만들 수 있는 테스트 프레임워크로 테스트 케이스를 만듭니다(예: Surefire JUnit plugin, TestNG, Cucumber).

- S3 버킷을 만들고 이름을 기록해 둡니다. 자세한 내용을 알아보려면 Amazon S3 사용 설명서의 S3 버킷을 생성하는 방법을 참조하세요.
- IAM 역할을 생성하고 해당 ARN을 기록해 둡니다. 빌드 프로젝트를 만들 때 ARN이 필요합니다.
- 역할에 다음 권한이 없는 경우 권한을 추가합니다.

```
{
    "Effect": "Allow",
    "Resource": [
        "*"
    ],
    "Action": [
        "codebuild:CreateReportGroup",
        "codebuild:CreateReport",
        "codebuild:UpdateReport",
        "codebuild:BatchPutTestCases"
    ]
}
```

자세한 내용은 테스트 보고 작업에 대한 권한 단원을 참조하십시오.

1단계: 보고서 그룹 생성

1. `CreateReportGroupInput.json`이라는 이름의 파일을 만듭니다.
2. S3 버킷에 테스트 결과를 내보낼 폴더를 만듭니다.
3. 다음을 `CreateReportGroupInput.json`에 복사합니다. *<bucket-name>*는 S3 버킷의 이름을 사용합니다. *<path-to-folder>*은 S3 버킷의 폴더 경로를 입력합니다.

```
{
  "name": "<report-name>",
```

```
  "type": "TEST",
  "exportConfig": {
    "exportConfigType": "S3",
    "s3Destination": {
      "bucket": "<bucket-name>",
      "path": "<path-to-folder>",
      "packaging": "NONE"
    }
  }
}
```

4. CreateReportGroupInput.json을 포함하는 디렉터리에서 다음 명령을 실행합니다.

```
aws codebuild create-report-group --cli-input-json file://
CreateReportGroupInput.json
```

출력은 다음과 같습니다. reportGroup의 ARN을 기록해 둡니다. 이 보고서 그룹을 사용하는 프로젝트를 만들 때 사용합니다.

```
{
  "reportGroup": {
    "arn": "arn:aws:codebuild:us-west-2:123456789012:report-group/<report-name>",
    "name": "<report-name>",
    "type": "TEST",
    "exportConfig": {
      "exportConfigType": "S3",
      "s3Destination": {
        "bucket": "<s3-bucket-name>",
        "path": "<folder-path>",
        "packaging": "NONE",
        "encryptionKey": "arn:aws:kms:us-west-2:123456789012:alias/aws/s3"
      }
    },
    "created": 1570837165.885,
    "lastModified": 1570837165.885
  }
}
```

2단계: 보고서 그룹을 사용하여 프로젝트 구성

보고서를 실행하려면 먼저 보고서 그룹으로 구성된 CodeBuild 빌드 프로젝트를 만듭니다. 보고서 그룹에 지정된 테스트 케이스는 빌드를 실행할 때 실행됩니다.

1. 이름이 `buildspec.yml`인 buildspec 파일을 만듭니다
2. 다음 YAML을 `buildspec.yml` 파일의 템플릿으로 사용합니다. 테스트를 실행하는 명령이 포함되어야 합니다. `reports` 섹션에서는 테스트 케이스 결과가 포함된 파일을 지정합니다. 이러한 파일에는 CodeBuild로 액세스할 수 있는 테스트 결과가 저장됩니다. 작성되고 30일 후에 만료됩니다. 이러한 파일은 S3 버킷으로 내보내는 원시 테스트 케이스 결과 파일과 다릅니다.

```
version: 0.2
    phases:
    install:
        runtime-versions:
            java: openjdk8
    build:
      commands:
        - echo Running tests
        - <enter commands to run your tests>

    reports:
      <report-name-or-arn>: #test file information
      files:
        - '<test-result-files>'
      base-directory: '<optional-base-directory>'
      discard-paths: false #do not remove file paths from test result files
```

> **Note**
>
> 기존 보고서 그룹의 ARN 대신, 생성되지 않은 보고서 그룹의 이름을 지정할 수도 있습니다. ARN 대신 이름을 지정하면 CodeBuild에서 빌드를 실행할 때 보고서 그룹을 만듭니다. 이름에는 프로젝트 이름과 buildspec 파일에 지정한 이름이 `project-name-report-group-name` 형식으로 포함되어 있습니다. 자세한 내용은 테스트 보고서 생성 및 보고서 그룹 이름 지정 섹션을 참조하세요.

3. `project.json`이라는 이름의 파일을 만듭니다. 이 파일에는 create-project 명령에 대한 입력이 들어 있습니다.

4. 다음 JSON을 `project.json`에 복사합니다. `source`는 소스 파일이 들어있는 저장소의 유형과 위치를 입력합니다. `serviceRole`은 사용 중인 역할의 ARN을 지정합니다.

```
{
  "name": "test-report-project",
  "description": "sample-test-report-project",
  "source": {
    "type": "CODECOMMIT|CODEPIPELINE|GITHUB|S3|BITBUCKET|GITHUB_ENTERPRISE|
NO_SOURCE",
    "location": "<your-source-url>"
  },
  "artifacts": {
    "type": "NO_ARTIFACTS"
  },
  "cache": {
    "type": "NO_CACHE"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/standard:5.0",
    "computeType": "small"
  },
  "serviceRole": "arn:aws:iam::<your-aws-account-id>:role/service-role/<your-role-
name>"
}
```

5. `project.json`을 포함하는 디렉터리에서 다음 명령을 실행합니다. 이렇게 하면 이름이 `test-project`인 프로젝트가 생성됩니다.

```
aws codebuild create-project --cli-input-json file://project.json
```

3단계: 보고서 실행 및 결과 보기

이 섹션에서는 이전에 만든 프로젝트의 빌드를 실행합니다. 빌드 프로세스 중에 CodeBuild에서는 테스트 케이스의 결과가 포함된 보고서를 생성합니다. 보고서는 지정한 보고서 그룹에 포함되어 있습니다.

1. 빌드를 시작하려면 다음 명령을 실행합니다. `test-report-project`는 위에서 만든 빌드 프로젝트의 이름입니다. 출력에 나타나는 빌드 ID를 기록해 둡니다.

```
aws codebuild start-build --project-name test-report-project
```

2. 다음 명령을 실행하여 보고서의 ARN을 포함하여 빌드에 대한 정보를 가져옵니다. *<build-id>*은 빌드 ID를 지정합니다. 출력의 `reportArns` 속성에 보고서 ARN을 기록해 둡니다.

```
aws codebuild batch-get-builds --ids <build-id>
```

3. 보고서에 대한 세부 정보를 가져오려면 다음 명령을 실행합니다. *<report-arn>*은 보고서 ARN을 지정합니다.

```
aws codebuild batch-get-reports --report-arns <report-arn>
```

출력은 다음과 같습니다. 이 샘플 출력은 성공했거나, 실패했거나, 건너뛰었거나, 오류가 발생했거나, 알 수 없는 상태를 반환하는 테스트 수를 보여 줍니다.

```
{
  "reports": [
    {
      "status": "FAILED",
      "reportGroupArn": "<report-group-arn>",
      "name": "<report-group-name>",
      "created": 1573324770.154,
      "exportConfig": {
        "exportConfigType": "S3",
        "s3Destination": {
          "bucket": "<amzn-s3-demo-bucket>",
          "path": "<path-to-your-report-results>",
          "packaging": "NONE",
          "encryptionKey": "<encryption-key>"
        }
      },
      "expired": 1575916770.0,
      "truncated": false,
      "executionId": "arn:aws:codebuild:us-west-2:123456789012:build/<name-of-
build-project>:2c254862-ddf6-4831-a53f-6839a73829c1",
      "type": "TEST",
      "arn": "<report-arn>",
      "testSummary": {
        "durationInNanoSeconds": 6657770,
        "total": 11,
        "statusCounts": {
          "FAILED": 3,
          "SKIPPED": 7,
          "ERROR": 0,
```

```
            "SUCCEEDED": 1,
            "UNKNOWN": 0
          }
        }
      }
    ],
    "reportsNotFound": []
}
```

4. 보고서의 테스트 케이스에 대한 정보를 나열하려면 다음 명령을 실행합니다. *<report-arn>*은 보고서의 ARN을 지정합니다. 선택 사항인 `--filter` 매개변수의 경우 하나의 상태 결과(`SUCCEEDED`, `FAILED`, `SKIPPED`, `ERROR` 또는 `UNKNOWN`)를 지정할 수 있습니다.

```
aws codebuild describe-test-cases \
    --report-arn <report-arn> \
    --filter status=SUCCEEDED|FAILED|SKIPPED|ERROR|UNKNOWN
```

출력은 다음과 같습니다.

```
{
  "testCases": [
    {
      "status": "FAILED",
      "name": "Test case 1",
      "expired": 1575916770.0,
      "reportArn": "<report-arn>",
      "prefix": "Cucumber tests for agent",
      "message": "A test message",
      "durationInNanoSeconds": 1540540,
      "testRawDataPath": "<path-to-output-report-files>"
    },
    {
      "status": "SUCCEEDED",
      "name": "Test case 2",
      "expired": 1575916770.0,
      "reportArn": "<report-arn>",
      "prefix": "Cucumber tests for agent",
      "message": "A test message",
      "durationInNanoSeconds": 1540540,
      "testRawDataPath": "<path-to-output-report-files>"
    }
  ]
```

```
}
```

CodeBuild용 Docker 샘플

이 섹션에서는 Docker와 간의 샘플 통합에 대해 설명합니다 AWS CodeBuild.

샘플	설명
CodeBuild용 Docker 사용자 지정 이미지 샘플	이 샘플은 CodeBuild 및 사용자 지정 Docker 빌드 이미지(Docker Hub의 `docker:dind`)를 사용하여 Docker 이미지를 빌드하고 실행합니다.
CodeBuild용 Docker 이미지 빌드 서버 샘플	이 샘플은 Docker 빌드를 관리형 이미지 빌드 서버로 오프로드합니다.
CodeBuild용 Windows Docker 빌드 샘플	이 샘플은 CodeBuild를 사용하여 Windows Docker 이미지를 빌드하고 실행합니다.
CodeBuild용 'Amazon ECR 이미지 리포지토리에 Docker 이미지 게시' 샘플	이 샘플은 빌드 출력으로 도커 이미지를 생산한 다음 도커 이미지를 Amazon Elastic Container Registry(Amazon ECR) 이미지 리포지토리에 푸시합니다.
CodeBuild용 AWS Secrets Manager 샘플이 포함된 프라이빗 레지스트리	이 샘플에서는 CodeBuild 실행 시간 환경으로서 프라이빗 레지스트리에 저장되는 Docker 이미지를 사용하는 방법을 보여줍니다.

CodeBuild용 Docker 사용자 지정 이미지 샘플

다음 샘플은 및 사용자 지정 Docker 빌드 이미지(`docker:dind`Docker Hub의)를 사용하여 Docker 이미지를 빌드 AWS CodeBuild 하고 실행합니다.

대신, Docker 지원을 통해 CodeBuild에서 제공하는 빌드 이미지를 사용하여 도커 이미지를 빌드하는 방법을 알아보려면 'Amazon ECR에 Docker 이미지 게시' 샘플 섹션을 참조하세요.

> **Important**
>
> 이 샘플을 실행하면 AWS 계정에 요금이 부과될 수 있습니다. 여기에는 CodeBuild와 Amazon S3 및 CloudWatch Logs와 관련된 AWS 리소스 AWS KMS및 작업에 대해 발생할 수 있는 요금이 포함됩니다. 자세한 내용은 CodeBuild 요금, Amazon S3 요금, AWS Key Management Service 요금 및 Amazon CloudWatch 요금을 참조하세요.

주제

- 사용자 지정 이미지 샘플의 Docker 실행

사용자 지정 이미지 샘플의 Docker 실행

다음 절차에 따라 사용자 지정 이미지 샘플에서 Docker를 실행합니다. 이 샘플에 대한 자세한 내용은 CodeBuild용 Docker 사용자 지정 이미지 샘플 섹션을 참조하세요.

사용자 지정 이미지 샘플에서 Docker를 실행하려면

1. 이 주제의 디렉터리 구조 및 파일 섹션에 설명된 대로 파일을 생성한 다음 S3 입력 버킷 또는 AWS CodeCommit, GitHub 또는 Bitbucket 리포지토리에 업로드합니다.

   > **Important**
   >
   > *(root directory name)*은 업로드하지 말고, *(root directory name)* 안에 있는 파일만 업로드하십시오.
   > S3 입력 버킷을 사용하고 있는 경우, 파일을 포함하는 ZIP 파일을 생성한 다음, 이를 입력 버킷에 업로드하십시오. *(root directory name)*을 ZIP 파일에 추가하지 말고, *(root directory name)* 안에 있는 파일만 추가하십시오.

2. 빌드 프로젝트를 생성하고, 빌드를 실행하고, 관련 빌드 정보를 확인합니다.

   AWS CLI 를 사용하여 빌드 프로젝트를 생성하는 경우 `create-project` 명령에 대한 JSON 형식의 입력이 이와 비슷할 수 있습니다. (자리 표시자는 사용자의 값으로 바꾸십시오.)

```
{
  "name": "sample-docker-custom-image-project",
  "source": {
    "type": "S3",
```

```
    "location": "codebuild-region-ID-account-ID-input-
bucket/DockerCustomImageSample.zip"
  },
  "artifacts": {
    "type": "NO_ARTIFACTS"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "docker:dind",
    "computeType": "BUILD_GENERAL1_SMALL",
    "privilegedMode": false
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

> **Note**
>
> 기본적으로 비 VPC 빌드에는 Docker 데몬이 활성화됩니다. VPC 빌드에 Docker 컨테이너를 사용하려면 Docker Docs 웹 사이트의 런타임 권한 및 Linux 기능을 참조하고 권한 부여 모드를 활성화합니다. 또한 Windows는 권한 모드를 지원하지 않습니다.

3. 빌드 결과를 확인하려면 빌드 로그에서 `Hello, World!` 문자열에 대해 찾아보십시오. 자세한 내용은 빌드 세부 정보 보기 단원을 참조하십시오.

디렉터리 구조

이 샘플에서는 다음 디렉터리 구조를 가정합니다.

```
(root directory name)
### buildspec.yml
### Dockerfile
```

파일

이 샘플에 사용되는 운영 체제의 기본 이미지는 Ubuntu입니다. 샘플은 이러한 파일을 사용합니다.

`buildspec.yml`(*(root directory name)*에 있음)

```
version: 0.2
```

```
phases:
  pre_build:
    commands:
      - docker build -t helloworld .
  build:
    commands:
      - docker images
      - docker run helloworld echo "Hello, World!"
```

`Dockerfile(`*`root directory name`*`에 있음)`

```
FROM maven:3.3.9-jdk-8

RUN echo "Hello World"
```

CodeBuild용 Docker 이미지 빌드 서버 샘플

다음 샘플은 Docker 빌드를 관리형 이미지 빌드 서버로 오프로드합니다. 이 샘플을 조정하여 CodeBuild 프로젝트 구성에서 전용 관리형 Docker 이미지 빌드 서버를 프로비저닝할 수 있습니다. 프로젝트에 대해 빌드가 적극적으로 실행되는 동안 프로비저닝된 인스턴스가 활성화되고 빌드가 실행되지 않으면 인스턴스가 중지됩니다. 프로비저닝된 인스턴스는 재활용되기 전에 최대 1개월 동안 저장됩니다. 자세한 내용은 CodeBuild Docker 서버 기능을 참조하세요.

> **Important**
>
> 이 샘플을 실행하면 AWS 계정에 요금이 부과될 수 있습니다. 여기에는 CodeBuild와 Amazon S3 및 CloudWatch Logs와 관련된 AWS 리소스 AWS KMS및 작업에 대해 발생할 수 있는 요금이 포함됩니다. 자세한 내용은 CodeBuild 요금, Amazon S3 요금, AWS Key Management Service 요금 및 Amazon CloudWatch 요금을 참조하세요.

주제

- Docker 서버 구성

Docker 서버 구성

다음 절차에 따라 Docker 워크로드를 관리하고 Docker 이미지 계층을 저장하는 CodeBuild 프로젝트의 전용 컴퓨팅 환경을 프로비저닝합니다.

Docker 서버를 구성하려면

1. 이 주제의 디렉터리 구조 및 파일 섹션에 설명된 대로 파일을 생성한 다음 S3 입력 버킷 또는 AWS CodeCommit, GitHub 또는 Bitbucket 리포지토리에 업로드합니다.

   > **Important**
   >
   > *(root directory name)*은 업로드하지 말고, *(root directory name)* 안에 있는 파일만 업로드하십시오.
   > S3 입력 버킷을 사용하고 있는 경우, 파일을 포함하는 ZIP 파일을 생성한 다음, 이를 입력 버킷에 업로드하십시오. *(root directory name)*을 ZIP 파일에 추가하지 말고, *(root directory name)* 안에 있는 파일만 추가하십시오.

2. 빌드 프로젝트를 생성하고, 빌드를 실행하고, 관련 빌드 정보를 확인합니다.

   a. 콘솔의 환경 섹션에서 추가 구성을 선택하고 Docker 서버 구성으로 이동한 다음 이 프로젝트에 대해 Docker 서버 활성화를 선택합니다. 그런 다음 Docker 서버 컴퓨팅 유형을 선택하고 레지스트리 자격 증명을 제공할 수 있습니다.

   b. AWS CLI 를 사용하여 빌드 프로젝트를 생성하는 경우 `create-project` 명령에 대한 JSON 형식의 입력이 이와 비슷할 수 있습니다. (자리 표시자는 사용자의 값으로 바꾸십시오.)

```
{
  "name": "sample-docker-custom-image-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-
bucket/DockerServerSample.zip"
  },
  "artifacts": {
    "type": "NO_ARTIFACTS"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/amazonlinux-x86_64-standard:5.0",
    "computeType": "BUILD_GENERAL1_LARGE",
    "dockerServer": [
         {
            "computeType": "BUILD_GENERAL1_LARGE",
            "securityGroupIds": [ "security-groups-ID" ]
         }
      ]
```

```
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name"
}
```

> **Note**
>
> Docker 서버에 대해 구성된 보안 그룹은 프로젝트에 구성된 VPC의 수신 네트워크 트래픽을 허용해야 합니다. 포트 9876에서 수신을 허용해야 합니다.

3. 빌드 결과를 확인하려면 빌드 로그에서 `Hello, World!` 문자열에 대해 찾아보십시오. 자세한 내용은 빌드 세부 정보 보기 단원을 참조하십시오.

디렉터리 구조

이 샘플에서는 다음 디렉터리 구조를 가정합니다.

```
(root directory name)
### buildspec.yml
### Dockerfile
```

파일

이 샘플에 사용되는 운영 체제의 기본 이미지는 Ubuntu입니다. 샘플은 이러한 파일을 사용합니다.

`buildspec.yml`(*(root directory name)*에 있음)

```
version: 0.2

phases:
  build:
    commands:
      - docker buildx build -t helloworld --load .
      - docker run helloworld echo "Hello, World!"
```

`Dockerfile`(*(root directory name)*에 있음)

```
FROM public.ecr.aws/amazonlinux/amazonlinux:latest

RUN echo "Hello World"
```

CodeBuild용 Windows Docker 빌드 샘플

다음 샘플은 CodeBuild를 사용하여 Windows Docker 이미지를 빌드하고 실행합니다.

주제

- Windows Docker 빌드 샘플 실행

Windows Docker 빌드 샘플 실행

다음 절차에 따라 Windows Docker 빌드를 실행합니다.

Windows Docker 빌드 샘플을 실행하려면

1. 이 주제의 디렉터리 구조 및 파일 섹션에 설명된 대로 파일을 생성한 다음 S3 입력 버킷 또는 AWS CodeCommit, GitHub 또는 Bitbucket 리포지토리에 업로드합니다.

   > **Important**
   >
   > *(root directory name)*은 업로드하지 말고, *(root directory name)* 안에 있는 파일만 업로드하십시오.
   > S3 입력 버킷을 사용하고 있는 경우, 파일을 포함하는 ZIP 파일을 생성한 다음, 이를 입력 버킷에 업로드하십시오. *(root directory name)*을 ZIP 파일에 추가하지 말고, *(root directory name)* 안에 있는 파일만 추가하십시오.

2. `WINDOWS_EC2` 플릿을 생성합니다.

   를 사용하여 AWS CLI 플릿을 생성하는 경우 `create-fleet` 명령에 대한 JSON 형식의 입력은 이와 비슷할 수 있습니다. (자리 표시자는 사용자의 값으로 바꾸십시오.)

   ```
   {
     "name": "fleet-name",
     "baseCapacity": 1,
     "environmentType": "WINDOWS_EC2",
     "computeType": "BUILD_GENERAL1_MEDIUM"
   }
   ```

3. 빌드 프로젝트를 생성하고, 빌드를 실행하고, 관련 빌드 정보를 확인합니다.

   AWS CLI 를 사용하여 빌드 프로젝트를 생성하는 경우 `create-project` 명령에 대한 JSON 형식의 입력이 이와 비슷할 수 있습니다. (자리 표시자는 사용자의 값으로 바꾸십시오.)

```
{
  "name": "project-name",
  "source": {
    "type": "S3",
    "location": "bucket-name/DockerImageSample.zip"
  },
  "artifacts": {
    "type": "NO_ARTIFACTS"
  },
  "environment": {
    "type": "WINDOWS_EC2",
    "image": "Windows",
    "computeType": "BUILD_GENERAL1_MEDIUM",
    "fleet": {
       "fleetArn": "fleet-arn"
    }
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name"
}
```

4. 빌드 결과를 확인하려면 빌드 로그에서 `Hello, World!` 문자열에 대해 찾아보십시오. 자세한 내용은 빌드 세부 정보 보기 단원을 참조하십시오.

디렉터리 구조

이 샘플에서는 다음 디렉터리 구조를 가정합니다.

```
(root directory name)
### buildspec.yml
### Dockerfile
```

파일

이 샘플에 사용되는 운영 체제의 기본 이미지는 `mcr.microsoft.com/windows/servercore:ltsc2022`입니다. 샘플은 이러한 파일을 사용합니다.

buildspec.yml(*(root directory name)*에 있음)

```
version: 0.2
```

```
phases:
  pre_build:
    commands:
      - docker build -t helloworld .
  build:
    commands:
      - docker images
      - docker run helloworld powershell -Command "Write-Host 'Hello World!'"
```

`Dockerfile(`*`root directory name`*`)`에 있음)

```
FROM mcr.microsoft.com/windows/servercore:ltsc2022

RUN powershell -Command "Write-Host 'Hello World'"
```

CodeBuild용 'Amazon ECR 이미지 리포지토리에 Docker 이미지 게시' 샘플

이 샘플은 빌드 출력으로 도커 이미지를 생산한 다음 도커 이미지를 Amazon Elastic Container Registry(Amazon ECR) 이미지 리포지토리에 푸시합니다. 이 샘플을 응용하여 도커 이미지를 Docker Hub에 푸시할 수도 있습니다. 자세한 내용은 'Docker 이미지를 Amazon ECR에 게시' 샘플을 조정하여 Docker Hub로 푸시 단원을 참조하십시오.

사용자 지정 도커 빌드 이미지(도커 허브의 `docker:dind`)를 사용하여 도커 이미지를 빌드하는 방법을 알아보려면 도커 사용자 지정 이미지 샘플 단원을 참조하십시오.

이 샘플은 `golang:1.12`를 참조하여 테스트됩니다.

이 샘플에서는 도커 이미지를 빌드 출력으로 생성하는 새로운 다단계 Docker 빌드 기능을 사용합니다. 그런 다음, 도커 이미지를 Amazon ECR 이미지 리포지토리로 푸시합니다. 다단계 도커 이미지 빌드는 최종 도커 이미지의 크기를 줄이는 데 도움이 됩니다. 자세한 내용은 Use multi-stage builds with Docker를 참조하십시오.

> **Important**
>
> 이 샘플을 실행하면 AWS 계정에 요금이 부과될 수 있습니다. 여기에는 Amazon S3, , AWS KMS CloudWatch Logs AWS CodeBuild 및 Amazon ECR과 관련된 AWS 리소스 및 작업에 대한 및 요금이 포함됩니다. 자세한 내용은 CodeBuild 요금, Amazon S3 요금, AWS Key Management Service 요금, Amazon CloudWatch 요금, Amazon Elastic Container Registry 요금을 참조하세요.

주제

- 'Amazon ECR에 Docker 이미지 게시' 샘플 실행
- 'Docker 이미지를 Amazon ECR에 게시' 샘플을 조정하여 Docker Hub로 푸시

'Amazon ECR에 Docker 이미지 게시' 샘플 실행

다음 절차에 따라 Amazon ECR에 Docker 이미지를 게시하는 샘플을 실행합니다. 이 샘플에 대한 자세한 내용은 CodeBuild용 'Amazon ECR 이미지 리포지토리에 Docker 이미지 게시' 샘플 섹션을 참조하세요.

이 샘플을 실행하려면

1. Amazon ECR에 사용할 이미지 리포지토리가 이미 있으면 3단계로 이동하세요. 그렇지 않으면 AWS 루트 계정 또는 관리자 사용자 대신 사용자를 사용하여 Amazon ECR로 작업하는 경우이 문(*### BEGIN ADDING STATEMENT HERE #### ### END ADDING STATEMENT HERE ###* 사이)을 사용자(또는 사용자가 연결된 IAM 그룹)에 추가합니다. AWS 루트 계정을 사용하는 것은 권장되지 않습니다. 이 문을 사용하면 Docker 이미지를 저장하기 위한 Amazon ECR 리포지토리를 생성할 수 있습니다. 간결하게 나타내고 명령문 추가 위치를 알 수 있도록 줄임표(`...`)가 사용되었습니다. 어떤 명령문도 제거하지 않아야 하며, 이러한 줄임표는 정책에 입력하지 않아야 합니다. 자세한 내용은 사용 설명서의 AWS Management Console을 사용한 인라인 정책 작업을 참조하세요.

   JSON

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

> **Note**
>
> 이 정책을 수정하는 IAM 엔터티에는 정책을 수정하는 IAM의 권한이 있어야 합니다.

2. Amazon ECR에서 이미지 리포지토리를 생성합니다. 빌드 환경을 생성하고 빌드를 실행하는 리전과 동일한 AWS 리전에 리포지토리를 생성해야 합니다. 자세한 내용은 Amazon VPC 사용 설명서의 리포지토리 생성을 참조하세요. 이 리포지토리의 이름은 이 절차의 뒷부분에서 지정하는 리포지토리 이름(`IMAGE_REPO_NAME` 환경 변수에 의해 표시됨)과 일치해야 합니다. Amazon ECR 리포지토리 정책이 CodeBuild 서비스 IAM 역할에 대한 이미지 푸시 액세스를 허용하는지 확인합니다.

3. AWS CodeBuild 서비스 역할에 연결한 정책에이 문(*`### BEGIN ADDING STATEMENT HERE ###`*와 *`### END ADDING STATEMENT HERE ###`* 사이)을 추가합니다. 다음 명령문을 사용하면 CodeBuild가 도커 이미지를 Amazon ECR 리포지토리에 업로드할 수 있습니다. 간결하게 나타내고 명령문 추가 위치를 알 수 있도록 줄임표(`...`)가 사용되었습니다. 어떤 명령문도 제거하지 않아야 하며, 이러한 줄임표는 정책에 입력하지 않아야 합니다.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:CompleteLayerUpload",
                "ecr:GetAuthorizationToken",
                "ecr:InitiateLayerUpload",
                "ecr:PutImage",
                "ecr:UploadLayerPart"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 이 정책을 수정하는 IAM 엔터티에는 정책을 수정하는 IAM의 권한이 있어야 합니다.

4. 이 주제의 디렉터리 구조 및 파일 섹션에 설명된 대로 파일을 생성한 다음 S3 입력 버킷 또는 AWS CodeCommit, GitHub 또는 Bitbucket 리포지토리에 업로드합니다. 자세한 내용은AWS CodePipeline 사용 설명서의 이미지 정의 파일 참조를 참조하세요.

> **Important**
>
> *(root directory name)*은 업로드하지 말고, *(root directory name)* 안에 있는 파일만 업로드하십시오.
> S3 입력 버킷을 사용하고 있는 경우, 파일을 포함하는 ZIP 파일을 생성한 다음, 이를 입력 버킷에 업로드하십시오. *(root directory name)*을 ZIP 파일에 추가하지 말고, *(root directory name)* 안에 있는 파일만 추가하십시오.

5. 빌드 프로젝트를 생성하고, 빌드를 실행하고, 빌드 정보를 확인합니다.

   콘솔을 사용하여 프로젝트를 생성할 경우:

   a. [Operating system]에서 [Ubuntu]를 선택합니다.
   b. 실행 시간에서 표준을 선택합니다.
   c. 이미지에서 aws/codebuild/standard:5.0을 선택합니다.
   d. 다음 환경 변수를 추가합니다.

      - *region-ID* 값이 있는 AWS_DEFAULT_REGION
      - *account-ID* 값이 있는 AWS_ACCOUNT_ID
      - 최신 값이 있는 IMAGE_TAG
      - *Amazon-ECR-repo-name* 값이 있는 IMAGE_REPO_NAME

   AWS CLI 를 사용하여 빌드 프로젝트를 생성하는 경우 `create-project` 명령에 대한 JSON 형식의 입력이 이와 비슷할 수 있습니다. (자리 표시자는 사용자의 값으로 바꾸십시오.)

```
{
  "name": "sample-docker-project",
  "source": {
```

```
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-bucket/DockerSample.zip"
  },
  "artifacts": {
    "type": "NO_ARTIFACTS"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/standard:5.0",
    "computeType": "BUILD_GENERAL1_SMALL",
    "environmentVariables": [
      {
        "name": "AWS_DEFAULT_REGION",
        "value": "region-ID"
      },
      {
        "name": "AWS_ACCOUNT_ID",
        "value": "account-ID"
      },
      {
        "name": "IMAGE_REPO_NAME",
        "value": "Amazon-ECR-repo-name"
      },
      {
        "name": "IMAGE_TAG",
        "value": "latest"
      }
    ],
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

6. CodeBuild가 리포지토리에 도커 이미지를 성공적으로 푸시했는지 확인합니다.

   1. Amazon ECR 콘솔(https://console.aws.amazon.com/ecr/)을 엽니다.
   2. 리포지토리 이름을 선택합니다. Image tag(이미지 태그) 열에 이미지가 있어야 합니다.

디렉터리 구조

이 샘플에서는 다음 디렉터리 구조를 가정합니다.

```
(root directory name)
```

```
### buildspec.yml
### Dockerfile
```

파일

이 샘플은 다음 파일을 사용합니다.

buildspec.yml(*(root directory name)*에 있음)

```
version: 0.2

phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws ecr get-login-password --region $AWS_DEFAULT_REGION | docker login --
username AWS --password-stdin $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t $IMAGE_REPO_NAME:$IMAGE_TAG .
      - docker tag $IMAGE_REPO_NAME:$IMAGE_TAG $AWS_ACCOUNT_ID.dkr.ecr.
$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker image...
      - docker push $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/
$IMAGE_REPO_NAME:$IMAGE_TAG
```

Dockerfile(*(root directory name)*에 있음)

```
FROM golang:1.12-alpine AS build
#Install git
RUN apk add --no-cache git
#Get the hello world package from a GitHub repository
RUN go get github.com/golang/example/hello
WORKDIR /go/src/github.com/golang/example/hello
# Build the project and send the output to /bin/HelloWorld
RUN go build -o /bin/HelloWorld

FROM golang:1.12-alpine
```

```
#Copy the build's output binary from the previous build container
COPY --from=build /bin/HelloWorld /bin/HelloWorld
ENTRYPOINT ["/bin/HelloWorld"]
```

> **Note**
>
> CodeBuild는 사용자 지정 도커 이미지에 대해 `ENTRYPOINT`를 재정의합니다.

'Docker 이미지를 Amazon ECR에 게시' 샘플을 조정하여 Docker Hub로 푸시

Amazon ECR 대신 Docker 이미지가 Docker Hub로 푸시되도록 'Docker 이미지를 Amazon ECR에 게시' 샘플을 조정하려면 샘플의 코드를 편집합니다. 샘플에 대한 자세한 내용은 CodeBuild용 'Amazon ECR 이미지 리포지토리에 Docker 이미지 게시' 샘플 및 'Amazon ECR에 Docker 이미지 게시' 샘플 실행 섹션을 참조하세요.

> **Note**
>
> 17.06 이전의 도커 버전을 사용하는 경우 `--no-include-email` 옵션을 제거합니다.

1. `buildspec.yml` 파일 내의 다음 Amazon ECR 특정 코드 행을 바꿉니다.

```
...
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws ecr get-login-password --region $AWS_DEFAULT_REGION |
 docker login --username AWS --password-stdin $AWS_ACCOUNT_ID.dkr.ecr.
$AWS_DEFAULT_REGION.amazonaws.com
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t $IMAGE_REPO_NAME:$IMAGE_TAG .
      - docker tag $IMAGE_REPO_NAME:$IMAGE_TAG $AWS_ACCOUNT_ID.dkr.ecr.
$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker image...
```

```
      - docker push $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/
$IMAGE_REPO_NAME:$IMAGE_TAG
...
```

다음 도커 허브 관련 코드 행으로 바꿉니다.

```
...
  pre_build:
    commands:
      - echo Logging in to Docker Hub...
      # Type the command to log in to your Docker Hub account here.
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t $IMAGE_REPO_NAME:$IMAGE_TAG .
      - docker tag $IMAGE_REPO_NAME:$IMAGE_TAG $IMAGE_REPO_NAME:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker image...
      - docker push $IMAGE_REPO_NAME:$IMAGE_TAG
...
```

2. 편집된 코드를 S3 입력 버킷 또는 AWS CodeCommit, GitHub 또는 Bitbucket 리포지토리에 업로드합니다.

> **Important**
>
> *(root directory name)*은 업로드하지 말고, *(root directory name)* 안에 있는 파일만 업로드하십시오.
> S3 입력 버킷을 사용하고 있는 경우, 파일을 포함하는 ZIP 파일을 생성한 다음, 이를 입력 버킷에 업로드하십시오. *(root directory name)*을 ZIP 파일에 추가하지 말고, *(root directory name)* 안에 있는 파일만 추가하십시오.

3. `create-project` 명령에 대한 JSON 형식 입력에서 다음 코드 행을 바꿉니다.

```
...
    "environmentVariables": [
      {
        "name": "AWS_DEFAULT_REGION",
```

```
        "value": "region-ID"
      },
      {
        "name": "AWS_ACCOUNT_ID",
        "value": "account-ID"
      },
      {
        "name": "IMAGE_REPO_NAME",
        "value": "Amazon-ECR-repo-name"
      },
      {
        "name": "IMAGE_TAG",
        "value": "latest"
      }
    ]
...
```

다음 코드 행으로 바꿉니다.

```
...
    "environmentVariables": [
      {
        "name": "IMAGE_REPO_NAME",
        "value": "your-Docker-Hub-repo-name"
      },
      {
        "name": "IMAGE_TAG",
        "value": "latest"
      }
    ]
...
```

4. 빌드 환경을 생성하고, 빌드를 실행하고, 관련 빌드 정보를 확인합니다.
5. 가 도커 이미지를 리포지토리에 AWS CodeBuild 성공적으로 푸시했는지 확인합니다. Docker Hub에 로그인하고, 리포지토리로 이동한 다음 [Tags] 탭을 선택합니다. `latest` 태그에 가장 최근의 [Last Updated] 값이 포함되어 있어야 합니다.

CodeBuild용 AWS Secrets Manager 샘플이 포함된 프라이빗 레지스트리

이 샘플은 프라이빗 레지스트리에 AWS CodeBuild 런타임 환경으로 저장된 Docker 이미지를 사용하는 방법을 보여줍니다. 프라이빗 레지스트리의 보안 인증은 AWS Secrets Manager에 저장됩니다. 프라이빗 레지스트리는 모두 CodeBuild와 함께 작동합니다. 이 샘플은 Docker Hub를 사용합니다.

> **Note**
>
> 보안 암호는 작업에 표시되며 파일에 기록될 때 가려지지 않습니다.

주제

- 프라이빗 레지스트리 샘플 요구 사항
- 프라이빗 레지스트리로 CodeBuild 프로젝트 생성
- 자체 호스팅 러너에 대한 프라이빗 레지스트리 자격 증명 구성

프라이빗 레지스트리 샘플 요구 사항

에서 프라이빗 레지스트리를 사용하려면 다음이 있어야 AWS CodeBuild합니다.

- Docker Hub 보안 인증을 저장하는 Secrets Manager 보안 암호. 보안 인증은 프라이빗 리포지토리 액세스에 사용합니다.

  > **Note**
  >
  > 생성한 보안 암호에 대해 요금이 청구됩니다.

- 프라이빗 리포지토리 또는 계정
- Secrets Manager 보안 암호 액세스 권한을 부여하는 CodeBuild 서비스 역할의 IAM 정책

다음 단계를 따라 해당 리소스를 만든 다음, 프라이빗 레지스트리에 저장된 도커 이미지를 사용하여 CodeBuild 빌드 프로젝트를 생성합니다.

프라이빗 레지스트리로 CodeBuild 프로젝트 생성

1. 프리 프라이빗 리포지토리 생성 방법에 대한 자세한 내용은 Docker Hub의 리포지토리를 참조하십시오. 또한 터미널에서 다음 명령을 실행하여 이미지를 가져오고, 이미지의 ID를 확보하고, 새 리포지토리로 푸시할 수 있습니다.

   ```
   docker pull amazonlinux
   docker images amazonlinux --format {{.ID}}
   docker tag image-id your-username/repository-name:tag
   docker login
   docker push your-username/repository-name
   ```

2. AWS Secrets Manager 사용 설명서의 AWS Secrets Manager 보안 암호 생성 단계를 따릅니다.

   a. 3단계의 보안 암호 유형 선택에서 다른 유형의 보안 암호를 선택합니다.

   b. 키/값 페어에서 Docker Hub 사용자 이름에 대한 키-값 페어와 Docker Hub 암호에 대한 키-값 페어를 하나씩 생성합니다.

   c. AWS Secrets Manager 보안 암호 생성의 단계를 계속 따릅니다.

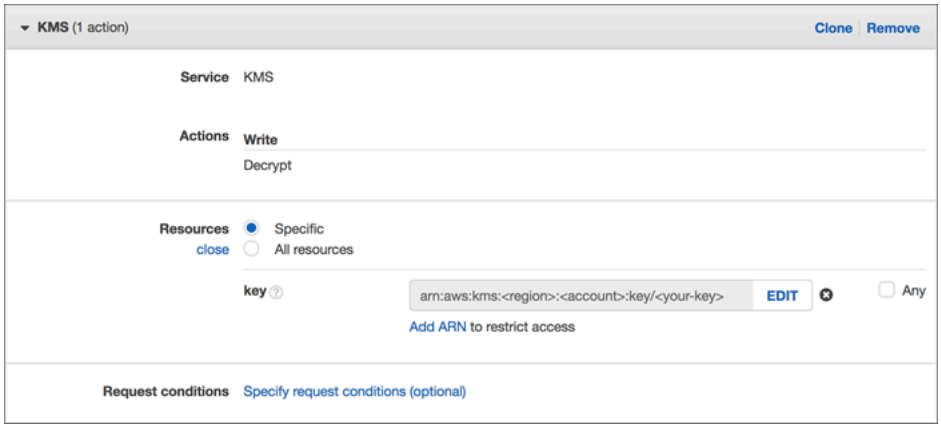
   d. 키는 Docker Hub 보안 인증에 해당하므로 5단계의 자동 교체 구성 페이지에서 이 옵션을 끕니다.

   e. AWS Secrets Manager 보안 암호 생성의 단계에 따라 완료합니다.

   자세한 내용은 AWS Secrets Manager란 무엇입니까?를 참조하세요.

3. 콘솔에서 AWS CodeBuild 프로젝트를 생성하면 CodeBuild가 필요한 권한을 연결합니다. 이외의 AWS KMS 키를 사용하는 경우 서비스 역할에 추가`DefaultEncryptionKey`해야 합니다. 자세한 내용은 IAM 사용 설명서의 역할 수정(콘솔)을 참조하세요.

   Secrets Manager에서 서비스 역할이 작동하려면 최소한 `secretsmanager:GetSecretValue` 권한이 있어야 합니다.



4. 콘솔을 사용하여 환경이 프라이빗 레지스트리에 저장되는 프로젝트를 생성하려면 프로젝트를 생성하면서 다음을 수행합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 단원을 참조하세요.

> **Note**
>
> VPC에 프라이빗 레지스트리가 있는 경우 퍼블릭 인터넷 액세스가 가능해야 합니다. CodeBuild는 VPC의 프라이빗 IP 주소에서 이미지를 끌어올 수 없습니다.

   a. 환경 이미지에서 사용자 지정 이미지를 선택합니다.
   b. 환경 유형에서 Linux 또는 Windows를 선택합니다.
   c. 이미지 레지스트리의 경우 다른 레지스트리를 선택합니다.
   d. 외부 레지스트리 URL에 이미지 위치를 입력하고 레지스트리 보안 인증 - 선택 사항에 Secrets Manager 보안 인증의 ARN 또는 이름을 입력합니다.

> **Note**
>
> 현재 리전에 보안 인증이 없을 경우 ARN을 사용해야 합니다. 다른 리전에 있는 보안 인증의 이름은 사용할 수 없습니다.

자체 호스팅 러너에 대한 프라이빗 레지스트리 자격 증명 구성

다음 지침에 따라 자체 호스팅 러너에 대한 레지스트리 자격 증명을 구성합니다.

> **Note**
>
> 이러한 자격 증명은 이미지가 프라이빗 레지스트리의 자격 증명으로 재정의되는 경우에만 사용됩니다.

AWS Management Console

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 빌드 프로젝트를 생성하거나 기존 프로젝트를 선택합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 및 빌드 프로젝트 설정 변경(콘솔) 섹션을 참조하세요.
3. 환경에서 추가 구성을 선택합니다.
4. 추가 구성에서 레지스트리 자격 증명 - 선택 사항으로에서 보안 암호의 이름 또는 ARN AWS Secrets Manager 을 입력합니다.



AWS CLI

1. 새 프로젝트를 생성하려면 create-project 명령을 실행합니다.

```
aws codebuild create-project \
    --name project-name \
    --source type=source-type,location=source-location \
    --environment "type=environment-type,image=image,computeType=compute-
type,registryCredential={credentialProvider=SECRETS_MANAGER,credential=secret-
name-or-arn},imagePullCredentialsType=CODEBUILD|SERVICE_ROLE" \
    --artifacts type=artifacts-type \
    --service-role arn:aws:iam::account-ID:role/service-role/service-role-name
```

2. 기존 프로젝트를 업데이트하려면 update-project 명령을 실행합니다.

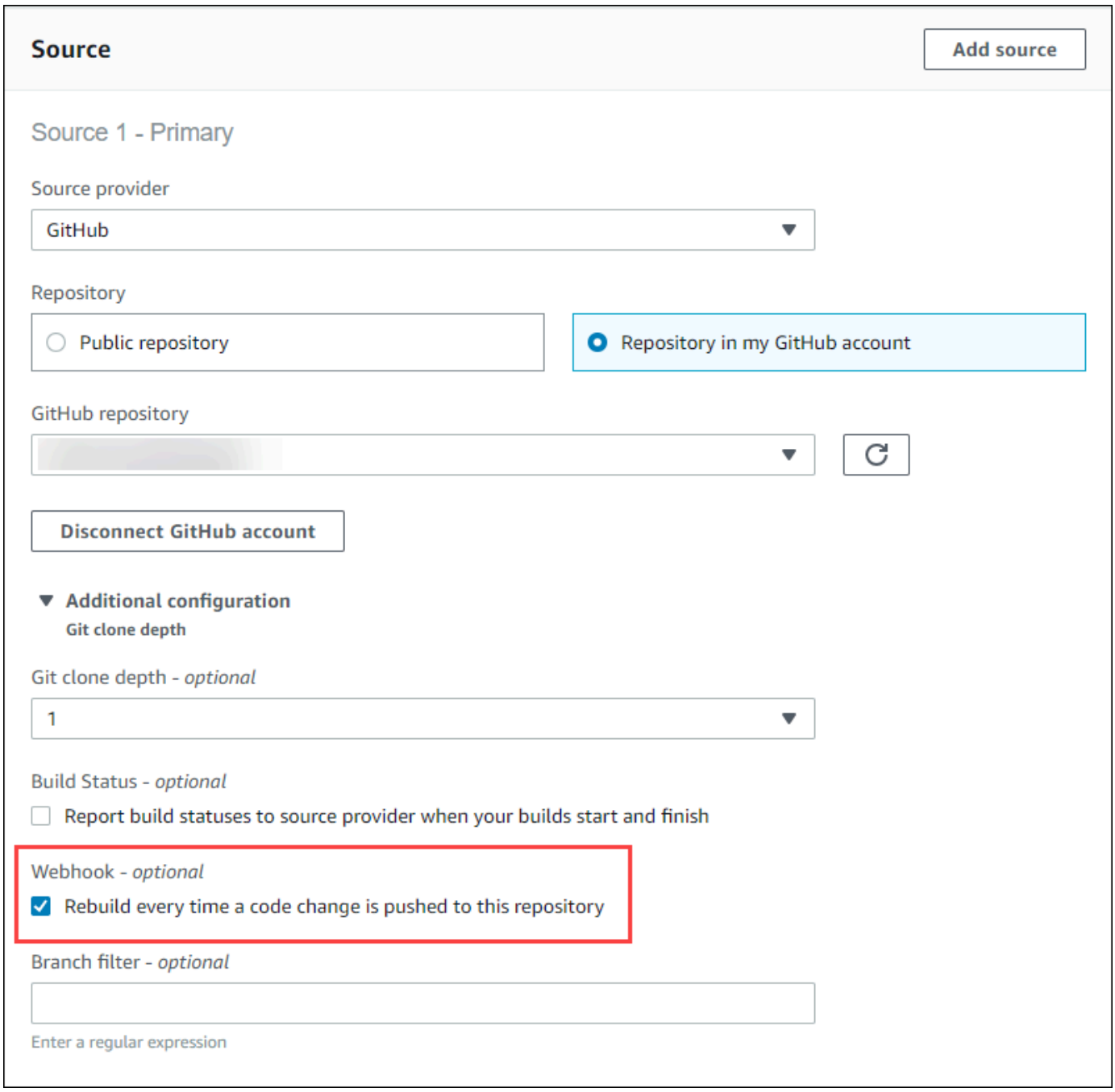
```
aws codebuild update-project \
    --name project-name \
    --environment "type=environment-type,image=image,computeType=compute-
type,registryCredential={credentialProvider=SECRETS_MANAGER,credential=secret-
name-or-arn}"
```

빌드 출력을 S3 버킷에서 호스팅하여 정적 웹사이트 생성

빌드의 아티팩트 암호화를 비활성화할 수 있습니다. 이렇게 하면 웹 사이트를 호스팅하도록 구성된 위치에 아티팩트를 게시할 수 있습니다. (암호화된 아티팩트는 게시할 수 없습니다.) 이 샘플에서는 Webhook를 사용해 빌드를 트리거한 후 웹사이트를 구성하는 S3 버킷에 아티팩트를 게시하는 방법에 대해서 설명합니다.

1. 정적 웹사이트 설정의 지침을 따라 웹사이트처럼 작동하도록 S3 버킷을 구성합니다.
2. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
3. CodeBuild 정보 페이지가 나타나면 빌드 프로젝트 생성을 선택합니다. 그렇지 않을 경우, 탐색 창에서 빌드를 확장한 후 빌드 프로젝트를 선택하고 빌드 프로젝트 생성을 선택합니다.
4. 프로젝트 이름에 이 빌드 프로젝트의 이름을 입력합니다. 빌드 프로젝트 이름은 각 AWS 계정에서 고유해야 합니다. 또한 선택에 따라 빌드 프로젝트에 대한 설명을 포함하여 다른 사용자가 이 프로젝트의 용도를 이해하도록 도울 수 있습니다.
5. 소스의 소스 공급자에서 GitHub를 선택합니다. GitHub와 연결(다시 연결)하는 지침을 따르고 승인을 선택합니다.

   Webhook에서 코드 변경이 이 리포지토리로 푸시될 때마다 다시 빌드를 선택합니다. 이 확인란은 내 GitHub 계정의 리포지토리를 선택한 경우에만 선택할 수 있습니다.



6. 환경에서 다음과 같이 합니다.

   [Environment image]에서 다음 중 하나를 수행합니다.

   - 에서 관리하는 도커 이미지를 사용하려면 관리형 이미지를 AWS CodeBuild선택한 다음 운영 체제, 런타임, 이미지(Image) 및 이미지 버전에서 선택합니다. 사용 가능한 경우 환경 유형에서 항목을 선택합니다.

- 다른 도커 이미지를 사용하려면 사용자 지정 이미지를 선택합니다. 환경 유형에서 ARM, Linux, Linux GPU 또는 Windows를 선택합니다. Other registry(다른 레지스트리)를 선택한 경우 External registry URL(외부 레지스트리 URL)에 Docker Hub의 도커 이미지 이름 및 태그를 *docker repository*/*docker image name* 형식으로 입력합니다. Amazon ECR을 선택하는 경우 Amazon ECR 리포지토리와 Amazon ECR 이미지를 사용하여 AWS 계정에서 도커 이미지를 선택합니다.
- 프라이빗 도커 이미지를 사용하려면 사용자 지정 이미지를 선택합니다. 환경 유형에서 ARM, Linux, Linux GPU 또는 Windows를 선택합니다. Image registry(이미지 레지스트리)에서 Other registry(다른 레지스트리)를 선택한 다음 프라이빗 도커 이미지에 대한 보안 인증 정보의 ARN을 입력합니다. 보안 인증은 Secrets Manager에서 생성됩니다. 자세한 내용은 AWS Secrets Manager사용 설명서의 AWS Secrets Manager 이란? 섹션을 참조하세요.

7. 서비스 역할에서 다음 중 하나를 수행합니다.

   - CodeBuild 서비스 역할이 없는 경우 새 서비스 역할을 선택합니다. 역할 이름에 새 역할의 이름을 입력합니다.
   - CodeBuild 서비스 역할이 있는 경우 기존 서비스 역할을 선택합니다. 역할 ARN에서 서비스 역할을 선택합니다.

   > **Note**
   >
   > 콘솔을 사용하여 빌드 프로젝트를 생성하거나 업데이트하는 경우, 이와 동시에 CodeBuild 서비스 역할을 만들 수 있습니다. 기본적으로 역할은 해당 빌드 프로젝트에서만 작동합니다. 콘솔을 사용하여 이 서비스 역할을 다른 빌드 프로젝트와 연결하는 경우 다른 빌드 프로젝트에서 작동하도록 역할이 업데이트됩니다. 하나의 서비스 역할은 최대 10개의 빌드 프로젝트에서 작동할 수 있습니다.

8. Buildspec에서 다음 중 하나를 수행합니다.

   - buildspec 파일 사용을 선택하여 소스 코드 루트 디렉터리에 있는 buildspec.yml 파일을 사용합니다.
   - 빌드 명령 삽입을 선택하여 콘솔에서 빌드 명령을 삽입합니다.

   자세한 내용은 buildspec 참조 단원을 참조하십시오.

9. 아티팩트의 유형에서 Amazon S3를 선택하여 빌드 출력을 S3 버킷에 저장합니다.
10. 버킷 이름에서 1단계에서 웹사이트처럼 작동하도록 구성한 S3 버킷의 이름을 선택합니다.

11. 환경에서 빌드 명령 삽입을 선택한 경우 출력 파일에 대해 출력 버킷에 넣으려는 빌드의 파일 위치를 입력합니다. 위치가 두 개 이상인 경우 쉼표를 사용하여 각 위치를 구분합니다(예: **`appspec.yml, target/my-app.jar`**). 자세한 내용은 Artifacts reference-key in the buildspec file 단원을 참조하십시오.
12. 아티팩트 암호화 비활성화를 선택합니다.
13. 추가 구성을 확장하고 적절한 옵션을 선택합니다.
14. 빌드 프로젝트 생성을 선택합니다. 빌드 프로젝트 페이지의 빌드 기록에서 빌드 시작을 선택하여 빌드를 실행합니다.
15. (선택 사항) Amazon S3 Developer Guide에 예제: Amazon CloudFront를 이용해 웹 사이트 속도 높이기의 지침을 따르세요.

다중 입력 소스 및 출력 아티팩트 샘플

둘 이상의 입력 소스와 둘 이상의 출력 아티팩트 세트가 있는 AWS CodeBuild 빌드 프로젝트를 생성할 수 있습니다. 이번 샘플은 아래와 같은 빌드 프로젝트를 설정하는 방법에 대한 내용입니다.

- 유형에 따라 여러 가지 소스와 리포지토리를 사용합니다.
- 단일 빌드에서 다수의 S3 버킷에 빌드 아티팩트를 게시합니다.

다음 샘플에서는 빌드 프로젝트를 생성하여 빌드를 실행하는 데 사용합니다. 또한 빌드 프로젝트의 buildspec 파일을 사용해 소스 1개 이상을 포함시키고, 아티팩트 세트 1개 이상을 생성하는 방법에 대해서 설명합니다.

CodeBuild에 대한 여러 소스 입력을 사용하여 여러 출력 아티팩트를 생성하는 파이프라인을 생성하는 방법을 알아보려면 섹션을 참조하세요CodePipeline/CodeBuild와 다중 입력 소스 및 출력 아티팩트 통합 샘플.

주제

- 여러 입력 및 출력으로 빌드 프로젝트 생성
- 소스 없이 빌드 프로젝트 생성

여러 입력 및 출력으로 빌드 프로젝트 생성

다음 절차에 따라 여러 입력 및 출력이 있는 빌드 프로젝트를 생성합니다.

여러 입력 및 출력으로 빌드 프로젝트를 생성하려면

1. 소스를 1개 이상의 S3 버킷, CodeCommit, GitHub, GitHub Enterprise Server 또는 Bitbucket 리포지토리에 업로드합니다.
2. 기본 소스로 사용할 소스를 선택합니다. 기본 소스란 CodeBuild가 buildspec 파일을 찾아서 실행하는 소스를 말합니다.
3. 빌드 프로젝트를 생성합니다. 자세한 내용은 에서 빌드 프로젝트 생성 AWS CodeBuild 단원을 참조하십시오.
4. 빌드 프로젝트를 생성하고, 빌드를 실행하고, 빌드에 대한 정보를 가져옵니다.
5. AWS CLI 를 사용하여 빌드 프로젝트를 생성하는 경우 `create-project` 명령에 대한 JSON 형식의 입력은 다음과 비슷할 수 있습니다.

```
{
  "name": "sample-project",
  "source": {
    "type": "S3",
    "location": "<bucket/sample.zip>"
  },
  "secondarySources": [
    {
      "type": "CODECOMMIT",
      "location": "https://git-codecommit.us-west-2.amazonaws.com/v1/repos/repo",
      "sourceIdentifier": "source1"
    },
    {
      "type": "GITHUB",
      "location": "https://github.com/awslabs/aws-codebuild-jenkins-plugin",
      "sourceIdentifier": "source2"
    }
  ],
  "secondaryArtifacts": [ss
    {
      "type": "S3",
      "location": "<output-bucket>",
      "artifactIdentifier": "artifact1"
    },
    {
      "type": "S3",
      "location": "<other-output-bucket>",
      "artifactIdentifier": "artifact2"
    }
```

```
  ],
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/standard:5.0",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

기본 소스는 source 속성에서 정의됩니다. 그 밖에 다른 소스는 보조 소스라고 불리며, secondarySources에 표시됩니다. 보조 소스는 모두 자체 디렉터리에 설치됩니다. 이 디렉터리는 내장 환경 변수인 CODEBUILD_SRC_DIR_*sourceIdentifer*에 저장됩니다. 자세한 내용은 빌드 환경의 환경 변수 단원을 참조하십시오.

secondaryArtifacts 속성에는 아티팩트 정의 목록이 포함됩니다. 이러한 아티팩트는 secondary-artifacts 블록 내에 중첩되는 buildspec 파일의 artifacts 블록을 사용합니다.

buildspec 파일의 보조 아티팩트는 아티팩트와 동일한 구조를 가지고 있지만 아티팩트 식별자로 구분됩니다.

> **Note**
>
> CodeBuild API에서는 보조 아티팩트의 artifactIdentifier가 CreateProject 및 UpdateProject에서 반드시 필요한 속성입니다. 보조 아티팩트를 참조할 때 사용해야 합니다.

앞서 얘기한 JSON 형식의 입력을 사용하면 프로젝트의 buildspec 파일은 다음과 같은 모습이 될 수 있습니다.

```
version: 0.2

phases:
  install:
    runtime-versions:
      java: openjdk11
  build:
    commands:
      - cd $CODEBUILD_SRC_DIR_source1
```

```
      - touch file1
      - cd $CODEBUILD_SRC_DIR_source2
      - touch file2

artifacts:
  files:
    - '**.*'
  secondary-artifacts:
    artifact1:
      base-directory: $CODEBUILD_SRC_DIR_source1
      files:
        - file1
    artifact2:
      base-directory: $CODEBUILD_SRC_DIR_source2
      files:
        - file2
```

기본 속성의 버전은 API를 사용해 `sourceVersion`의 `StartBuild` 속성에서 재정의할 수 있습니다. 보조 소스 버전을 1개 이상 재정의할 때는 `secondarySourceVersionOverride` 속성을 사용하십시오.

의 `start-build` 명령에 대한 JSON 형식의 입력은 다음과 같을 AWS CLI 수 있습니다.

```
{
   "projectName": "sample-project",
   "secondarySourcesVersionOverride": [
      {
        "sourceIdentifier": "source1",
        "sourceVersion": "codecommit-branch"
      },
      {
        "sourceIdentifier": "source2",
        "sourceVersion": "github-branch"
      },
   ]
}
```

소스 없이 빌드 프로젝트 생성

소스를 구성할 때 **`NO_SOURCE`** 소스 유형을 선택하여 CodeBuild 프로젝트를 구성할 수 있습니다. 소스 유형이 **`NO_SOURCE`**일 경우, 프로젝트에 소스가 없으므로 buildspec 파일을 지정할 수 없습니다. 대신

에 `create-project` CLI 명령에 대한 JSON 형식 입력의 `buildspec` 속성 내 YAML 형식 buildspec 문자열 지정이 필요합니다. 값이 다음과 같을 것입니다.

```
{
  "name": "project-name",
  "source": {
    "type": "NO_SOURCE",
    "buildspec": "version: 0.2\n\nphases:\n  build:\n    commands:\n      - command"
   },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/standard:5.0",
    "computeType": "BUILD_GENERAL1_SMALL",
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

자세한 내용은 빌드 프로젝트 생성(AWS CLI) 단원을 참조하십시오.

CodeBuild용 buildspec 파일 샘플의 런타임 버전

Amazon Linux 2(AL2) 표준 이미지 버전 1.0 이상 또는 Ubuntu 표준 이미지 버전 2.0 이상을 사용하는 경우 buildspec 파일의 `runtime-versions` 섹션에서 하나 이상의 런타임을 지정할 수 있습니다. 다음 샘플은 프로젝트 런타임 변경, 둘 이상의 런타임 지정, 다른 런타임에 종속되는 런타임 지정 방법을 보여줍니다. 지원되는 런타임에 대한 자세한 내용은 CodeBuild가 제공하는 도커 이미지 단원을 참조하십시오.

> **Note**
>
> 빌드 컨테이너에서 도커를 사용할 경우에는 권한이 있는 모드에서 빌드가 실행되어야 합니다. 자세한 내용은 수동으로 AWS CodeBuild 빌드 실행 및 에서 빌드 프로젝트 생성 AWS CodeBuild 섹션을 참조하세요.

주제

- buildspec 파일의 런타임 버전 업데이트
- 런타임 2개 지정

buildspec 파일의 런타임 버전 업데이트

buildspec 파일의 `runtime-versions` 섹션을 업데이트하여 프로젝트에서 사용되는 런타임을 새 버전으로 수정할 수 있습니다. 다음 예제는 Java 버전 8 및 11을 지정하는 방법을 보여 줍니다.

- Java 버전 8을 지정하는 `runtime-versions` 부분:

```
phases:
  install:
    runtime-versions:
      java: corretto8
```

- Java 버전 11을 지정하는 `runtime-versions` 부분:

```
phases:
  install:
    runtime-versions:
      java: corretto11
```

다음 예제는 Ubuntu 표준 이미지 5.0 또는 Amazon Linux 2 표준 이미지 3.0을 사용하여 Python의 다양한 버전을 지정하는 방법을 보여 줍니다.

- Python 버전 3.7을 지정하는 `runtime-versions` 섹션:

```
phases:
  install:
    runtime-versions:
      python: 3.7
```

- Python 버전 3.8을 지정하는 `runtime-versions` 섹션:

```
phases:
  install:
    runtime-versions:
      python: 3.8
```

이 샘플은 Java 버전 8 런타임으로 시작해서 이후 Java 버전 10 런타임으로 업데이트되는 프로젝트를 보여줍니다.

1. Maven을 다운로드하고 설치합니다. 자세한 정보는 Apache Maven 웹 사이트의 Downloading Apache Maven 및 Installing Apache Maven 단원을 참조하십시오.
2. 로컬 컴퓨터나 인스턴스의 빈 디렉터리로 전환한 다음, 아래 Maven 명령을 실행합니다.

```
mvn archetype:generate "-DgroupId=com.mycompany.app" "-DartifactId=ROOT" "-
DarchetypeArtifactId=maven-archetype-webapp" "-DinteractiveMode=false"
```

성공하면 다음 디렉터리 구조 및 파일이 생성됩니다.

```
.
### ROOT
    ### pom.xml
    ### src
        ### main
            ### resources
            ### webapp
                ### WEB-INF
                #   ### web.xml
                ### index.jsp
```

3. 다음 콘텐츠를 가진 buildspec.yml이라는 파일을 생성합니다: 파일을 *(root directory name)*/my-web-app 디렉터리에 저장합니다.

```
version: 0.2

phases:
  install:
    runtime-versions:
      java: corretto8
  build:
    commands:
      - java -version
      - mvn package
artifacts:
  files:
    - '**/*'
  base-directory: 'target/my-web-app'
```

buildspec 파일에서

- `runtime-versions` 부분은 해당 프로젝트에서 Java 런타임 버전 8을 사용하도록 지정합니다.
- `- java -version` 명령은 빌드할 때 프로젝트에서 사용하는 Java 버전을 표시합니다.

파일 구조가 아래와 같이 나타날 것입니다.

```
(root directory name)
### my-web-app
    ### src
    #   ### main
    #   ### resources
    #   ### webapp
    #       ### WEB-INF
    #           ### web.xml
    #               ### index.jsp
    ### buildspec.yml
    ### pom.xml
```

4. my-web-app 디렉터리의 내용을 S3 입력 버킷이나 CodeCommit, GitHub 또는 Bitbucket 리포지토리에 업로드합니다.

> **Important**
>
> *(root directory name)* 또는 *(root directory name)*/my-web-app은 업로드하지 말고, *(root directory name)*/my-web-app 안에 있는 디렉터리 및 파일만 업로드하십시오.
> S3 입력 버킷을 사용하고 있는 경우, 디렉터리 구조 및 파일을 포함하는 ZIP 파일을 생성한 다음, 이를 입력 버킷에 업로드합니다. *(root directory name)* 또는 *(root directory name)*/my-web-app을 ZIP 파일에 추가하지 말고, *(root directory name)*/my-web-app 안에 있는 디렉터리 및 파일만 추가하십시오.

5. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
6. 빌드 프로젝트를 생성합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 및 빌드 실행(콘솔) 섹션을 참조하세요. 다음 설정을 제외하고 모든 설정을 기본값 그대로 둡니다.

   - 환경:
     - 환경 이미지에서 이미지 관리를 선택합니다.
     - 운영 체제에서 Amazon Linux 2를 선택합니다.

- 런타임에서 표준을 선택합니다.
- 이미지의 경우 aws/codebuild/amazonlinux-x86_64-standard:4.0을 선택합니다.

7. 빌드 시작를 선택합니다.
8. Build configuration(빌드 구성)에서 기본값을 적용한 다음 빌드 시작을 선택합니다.
9. 빌드가 완료되면 빌드 로그 탭에서 빌드 출력을 확인합니다. 다음과 유사한 출력 화면이 표시되어야 합니다.

```
[Container] Date Time Phase is DOWNLOAD_SOURCE
[Container] Date Time CODEBUILD_SRC_DIR=/codebuild/output/src460614277/src
[Container] Date Time YAML location is /codebuild/output/src460614277/src/
buildspec.yml
[Container] Date Time Processing environment variables
[Container] Date Time Selecting 'java' runtime version 'corretto8' based on manual
 selections...
[Container] Date Time Running command echo "Installing Java version 8 ..."
Installing Java version 8 ...

[Container] Date Time Running command export JAVA_HOME="$JAVA_8_HOME"

[Container] Date Time Running command export JRE_HOME="$JRE_8_HOME"

[Container] Date Time Running command export JDK_HOME="$JDK_8_HOME"

[Container] Date Time Running command for tool_path in "$JAVA_8_HOME"/bin/*
 "$JRE_8_HOME"/bin/*;
```

10. Java 버전 11의 `runtime-versions` 부분 업데이트:

```
install:
    runtime-versions:
      java: corretto11
```

11. 변경을 저장한 후 빌드를 다시 실행하고 빌드 출력을 확인합니다. Java 설치 버전이 11인지 확인해야 합니다. 다음과 유사한 출력 화면이 표시되어야 합니다.

```
[Container] Date Time Phase is DOWNLOAD_SOURCE
[Container] Date Time CODEBUILD_SRC_DIR=/codebuild/output/src460614277/src
[Container] Date Time YAML location is /codebuild/output/src460614277/src/
buildspec.yml
[Container] Date Time Processing environment variables
```

```
[Container] Date Time Selecting 'java' runtime version 'corretto11' based on manual
 selections...
Installing Java version 11 ...

[Container] Date Time Running command export JAVA_HOME="$JAVA_11_HOME"

[Container] Date Time Running command export JRE_HOME="$JRE_11_HOME"

[Container] Date Time Running command export JDK_HOME="$JDK_11_HOME"

[Container] Date Time Running command for tool_path in "$JAVA_11_HOME"/bin/*
 "$JRE_11_HOME"/bin/*;
```

런타임 2개 지정

동일한 CodeBuild 빌드 프로젝트에서 둘 이상의 런타임을 지정할 수 있습니다. 이 샘플은 두 개의 소스 파일을 사용하는데 하나는 Go 런타임을 사용하고, 다른 하나는 Node.js 런타임을 사용합니다.

1. `my-source`이라는 디렉터리를 생성합니다.
2. `my-source` 디렉터리 안에 이름이 `golang-app`인 디렉터리를 생성합니다.
3. 다음 콘텐츠를 가진 `hello.go`이라는 파일을 생성합니다: 파일을 `golang-app` 디렉터리에 저장합니다.

```
package main
import "fmt"

func main() {
  fmt.Println("hello world from golang")
  fmt.Println("1+1 =", 1+1)
  fmt.Println("7.0/3.0 =", 7.0/3.0)
  fmt.Println(true && false)
  fmt.Println(true || false)
  fmt.Println(!true)
  fmt.Println("good bye from golang")
}
```

4. `my-source` 디렉터리 안에 이름이 `nodejs-app`인 디렉터리를 생성합니다. `golang-app` 디렉터리와 레벨이 같아야 합니다.
5. 다음 콘텐츠를 가진 `index.js`이라는 파일을 생성합니다: 파일을 `nodejs-app` 디렉터리에 저장합니다.

```
console.log("hello world from nodejs");
console.log("1+1 =" + (1+1));
console.log("7.0/3.0 =" + 7.0/3.0);
console.log(true && false);
console.log(true || false);
console.log(!true);
console.log("good bye from nodejs");
```

6. 다음 콘텐츠를 가진 package.json이라는 파일을 생성합니다: 파일을 nodejs-app 디렉터리에 저장합니다.

```
{
  "name": "mycompany-app",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"run some tests here\""
  },
  "author": "",
  "license": "ISC"
}
```

7. 다음 콘텐츠를 가진 buildspec.yml이라는 파일을 생성합니다: my-source 디렉터리에, nodejs-app 및 golang-app 디렉터리와 같은 레벨에 파일을 저장합니다. runtime-versions 섹션은 Node.js 버전 12 및 Go 버전 1.13 런타임을 지정합니다.

```
version: 0.2

phases:
  install:
    runtime-versions:
      golang: 1.13
      nodejs: 12
  build:
    commands:
      - echo Building the Go code...
      - cd $CODEBUILD_SRC_DIR/golang-app
      - go build hello.go
      - echo Building the Node code...
      - cd $CODEBUILD_SRC_DIR/nodejs-app
      - npm run test
```

```
artifacts:
  secondary-artifacts:
    golang_artifacts:
      base-directory: golang-app
      files:
        - hello
    nodejs_artifacts:
      base-directory: nodejs-app
      files:
        - index.js
        - package.json
```

8. 파일 구조가 아래와 같이 나타날 것입니다.

```
my-source
### golang-app
#   ### hello.go
### nodejs.app
#   ### index.js
#   ### package.json
### buildspec.yml
```

9. `my-source` 디렉터리의 내용을 S3 입력 버킷이나 CodeCommit, GitHub 또는 Bitbucket 리포지토리에 업로드합니다.

> **Important**
>
> S3 입력 버킷을 사용하고 있는 경우, 디렉터리 구조 및 파일을 포함하는 ZIP 파일을 생성한 다음, 이를 입력 버킷에 업로드합니다. `my-source`를 ZIP 파일에 추가하지 말고, `my-source`에 있는 디렉터리와 파일만 추가하십시오.

10. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
11. 빌드 프로젝트를 생성합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 및 빌드 실행(콘솔) 섹션을 참조하세요. 다음 설정을 제외하고 모든 설정을 기본값 그대로 둡니다.

    - 환경:
      - 환경 이미지에서 이미지 관리를 선택합니다.
      - 운영 체제에서 Amazon Linux 2를 선택합니다.
      - 런타임에서 표준을 선택합니다.
      - 이미지의 경우 aws/codebuild/amazonlinux-x86_64-standard:4.0을 선택합니다.

12. 빌드 프로젝트 생성을 선택합니다.
13. 빌드 시작를 선택합니다.
14. Build configuration(빌드 구성)에서 기본값을 적용한 다음 빌드 시작을 선택합니다.
15. 빌드가 완료되면 빌드 로그 탭에서 빌드 출력을 확인합니다. 다음과 유사한 출력 화면이 표시되어야 합니다. Go 및 Node.js 런타임의 출력을 보여줍니다. Go 및 Node.js 애플리케이션의 출력도 보여줍니다.

```
[Container] Date Time Processing environment variables
[Container] Date Time Selecting 'golang' runtime version '1.13' based on manual
 selections...
[Container] Date Time Selecting 'nodejs' runtime version '12' based on manual
 selections...
[Container] Date Time Running command echo "Installing Go version 1.13 ..."
Installing Go version 1.13 ...

[Container] Date Time Running command echo "Installing Node.js version 12 ..."
Installing Node.js version 12 ...

[Container] Date Time Running command n $NODE_12_VERSION
   installed : v12.20.1 (with npm 6.14.10)

[Container] Date Time Moving to directory /codebuild/output/src819694850/src
[Container] Date Time Registering with agent
[Container] Date Time Phases found in YAML: 2
[Container] Date Time  INSTALL: 0 commands
[Container] Date Time  BUILD: 1 commands
[Container] Date Time Phase complete: DOWNLOAD_SOURCE State: SUCCEEDED
[Container] Date Time Phase context status code:  Message:
[Container] Date Time Entering phase INSTALL
[Container] Date Time Phase complete: INSTALL State: SUCCEEDED
[Container] Date Time Phase context status code:  Message:
[Container] Date Time Entering phase PRE_BUILD
[Container] Date Time Phase complete: PRE_BUILD State: SUCCEEDED
[Container] Date Time Phase context status code:  Message:
[Container] Date Time Entering phase BUILD
[Container] Date Time Running command echo Building the Go code...
Building the Go code...

[Container] Date Time Running command cd $CODEBUILD_SRC_DIR/golang-app

[Container] Date Time Running command go build hello.go
```

```
[Container] Date Time Running command echo Building the Node code...
Building the Node code...

[Container] Date Time Running command cd $CODEBUILD_SRC_DIR/nodejs-app

[Container] Date Time Running command npm run test

> mycompany-app@1.0.0 test /codebuild/output/src924084119/src/nodejs-app
> echo "run some tests here"

run some tests here
```

를 사용한 소스 버전 샘플 AWS CodeBuild

이 샘플은 커밋 ID 외의 형식(커밋 SHA라고 함)을 사용하여 소스 버전을 지정하는 방법을 입증합니다. 다음 방법으로 소스 버전을 지정할 수 있습니다.

- Amazon S3 소스 공급자의 경우 빌드 입력 ZIP 파일을 나타내는 객체의 버전 ID를 사용합니다.
- CodeCommit의 경우 Bitbucket, GitHub 및 GitHub Enterprise Server는 다음 중 하나를 사용합니다.
  - 풀 요청 참조로서 풀 요청(예: `refs/pull/1/head`).
  - 브랜치 이름으로 브랜치합니다.
  - 커밋 ID.
  - 태그.
  - 참조 및 커밋 ID. 참조는 다음 중 하나일 수 있습니다.
    - 태그(예: `refs/tags/mytagv1.0^{full-commit-SHA}`).
    - 브랜치(예: `refs/heads/mydevbranch^{full-commit-SHA}`).
    - 풀 요청(예: `refs/pull/1/head^{full-commit-SHA}`).
- GitLab 및 GitLab Self Managed의 경우 다음 중 하나를 사용합니다.
  - 브랜치 이름으로 브랜치합니다.
  - 커밋 ID.
  - 태그.

> **Note**
>
> 리포지토리가 GitHub 또는 GitHub Enterprise Server인 경우에만 풀 요청 소스 버전을 지정할 수 있습니다.

참조 및 커밋 ID를 사용하여 버전을 지정하는 경우 빌드의 DOWNLOAD_SOURCE 단계는 버전만을 제공하는 경우보다 더 빠릅니다. 그 이유는 참조 추가 시 CodeBuild가 커밋을 찾기 위해 전체 리포지토리를 다운로드할 필요가 없기 때문입니다.

- 커밋 ID(예: `12345678901234567890123467890123456789`)만을 사용하여 소스 버전을 지정할 수 있습니다. 이 경우 CodeBuild는 버전을 찾기 위해 전체 리포지토리를 다운로드해야 합니다.
- 다음 형식으로 참조 및 커밋 ID를 사용하여 소스 버전을 지정할 수 있습니다. *refs/heads/branchname*^{*full-commit-SHA*}(예: `refs/heads/main^{12345678901234567890123467890123456789}`). 이 경우 CodeBuild는 버전을 찾기 위해 지정한 분기만 다운로드합니다.

> **Note**
>
> 또한 빌드의 DOWNLOAD_SOURCE 단계를 촉진하기 위해 Git clone depth를 낮은 값으로 설정할 수 있습니다. CodeBuild는 더 적은 수의 리포지토리 버전을 다운로드합니다.

주제

- 커밋 ID를 사용하여 GitHub 리포지토리 버전 지정
- 참조 및 커밋 ID를 사용하여 GitHub 리포지토리 버전 지정

커밋 ID를 사용하여 GitHub 리포지토리 버전 지정

커밋 ID(예: `12345678901234567890123467890123456789`)만을 사용하여 소스 버전을 지정할 수 있습니다. 이 경우 CodeBuild는 버전을 찾기 위해 전체 리포지토리를 다운로드해야 합니다.

커밋 ID를 사용하여 GitHub 리포지토리 버전을 지정하려면

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.

2. 빌드 프로젝트를 생성합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 및 빌드 실행(콘솔) 섹션을 참조하세요. 다음 설정을 제외하고 모든 설정을 기본값 그대로 둡니다.

   - 소스에서 다음과 같이 합니다.
     - 소스 공급자에서 GitHub를 선택합니다. GitHub에 연결되어 있지 않은 경우 연결 지침을 따릅니다.
     - 리포지토리에서 퍼블릭 리포지토리를 선택합니다.
     - 리포지토리 URL에 **https://github.com/aws/aws-sdk-ruby.git**을 입력합니다.
   - 환경에서 다음과 같이 합니다.
     - 환경 이미지에서 이미지 관리를 선택합니다.
     - 운영 체제에서 Amazon Linux 2를 선택합니다.
     - 런타임에서 표준을 선택합니다.
     - 이미지의 경우 aws/codebuild/amazonlinux-x86_64-standard:4.0을 선택합니다.
3. 빌드 사양에서 빌드 명령 삽입을 선택한 후 편집기로 전환을 선택합니다.
4. 빌드 명령에서 자리표시자 텍스트를 다음으로 바꿉니다.

```
version: 0.2

phases:
  install:
    runtime-versions:
      ruby: 2.6
  build:
    commands:
       - echo $CODEBUILD_RESOLVED_SOURCE_VERSION
```

   Ubuntu 표준 이미지 2.0 사용 시 `runtime-versions` 섹션이 필요합니다. 여기서 Ruby 버전 2.6 런타임이 지정되지만, 모든 런타임을 사용할 수 있습니다. `echo` 명령은 `CODEBUILD_RESOLVED_SOURCE_VERSION` 환경 변수에 저장된 소스 코드 버전을 표시합니다.
5. Build configuration(빌드 구성)에서 기본값을 적용한 다음 빌드 시작을 선택합니다.
6. 소스 버전에 **046e8b67481d53bdc86c3f6affdd5d1afae6d369**를 입력합니다. 이는 `https://github.com/aws/aws-sdk-ruby.git` 리포지토리의 커밋 SHA입니다.
7. 빌드 시작를 선택합니다.
8. 빌드 완료 시 다음과 같은 모양이어야 합니다.

   - 빌드 로그 탭에서 프로젝트의 어떤 버전이 사용되었습니다. 다음 예를 참고하세요

```
[Container] Date Time Running command echo $CODEBUILD_RESOLVED_SOURCE_VERSION
046e8b67481d53bdc86c3f6affdd5d1afae6d369

[Container] Date Time Phase complete: BUILD State: SUCCEEDED
```

- 환경 변수 탭에서 Resolved source version(해결된 소스 버전)이 빌드 생성에 사용된 커밋 ID와 일치합니다.
- 단계 세부 정보 탭에서 DOWNLOAD_SOURCE 단계의 기간입니다.

참조 및 커밋 ID를 사용하여 GitHub 리포지토리 버전 지정

다음 형식으로 참조 및 커밋 ID를 사용하여 소스 버전을 지정할 수 있습니다. *refs/heads/branchname*^{*full-commit-SHA*}(예: refs/heads/main^{1234567890123456789012346789012345678 9}). 이 경우 CodeBuild는 버전을 찾기 위해 지정한 분기만 다운로드합니다.

참조 및 커밋 ID를 사용하여 GitHub 리포지토리 버전을 지정하려면

1. 커밋 ID를 사용하여 GitHub 리포지토리 버전 지정의 단계를 수행하세요.
2. 왼쪽 탐색 창에서 빌드 프로젝트를 선택한 후 이전에 생성된 프로젝트를 선택합니다.
3. 빌드 시작를 선택합니다.
4. 소스 버전에 **refs/heads/main^{046e8b67481d53bdc86c3f6affdd5d1afae6d369}**를 입력합니다. 이는 *refs/heads/branchname*^{*full-commit-SHA*} 형식의 브랜치 및 커밋 ID와 동일합니다.
5. 빌드 시작를 선택합니다.
6. 빌드 완료 시 다음과 같은 모양이어야 합니다.

   - 빌드 로그 탭에서 프로젝트의 어떤 버전이 사용되었습니다. 다음 예를 참고하세요

   ```
   [Container] Date Time Running command echo $CODEBUILD_RESOLVED_SOURCE_VERSION
   046e8b67481d53bdc86c3f6affdd5d1afae6d369

   [Container] Date Time Phase complete: BUILD State: SUCCEEDED
   ```

   - 환경 변수 탭에서 Resolved source version(해결된 소스 버전)이 빌드 생성에 사용된 커밋 ID와 일치합니다.

- 단계 세부 정보 탭에서 DOWNLOAD_SOURCE 단계의 기간은 커밋 ID만을 사용하여 소스 버전을 지정했을 때의 기간보다 짧아야 합니다.

CodeBuild용 타사 소스 리포지토리 샘플

이 섹션에서는 타사 소스 리포지토리와 CodeBuild 간의 샘플 통합에 대해 설명합니다.

샘플	설명
BitBucket pull 요청 및 웹후크 필터 샘플 – CodeBuild용 'Bitbucket pull 요청 및 웹후크 필터' 샘플 실행 참조	이 샘플에서는 Bitbucket 리포지토리를 사용하여 풀 요청을 생성하는 방법을 보여줍니다. 또한 Bitbucket webhook를 통해 CodeBuild를 트리거하여 프로젝트 빌드를 생성하는 방법을 보여줍니다.
GitHub Enterprise Server 샘플 – CodeBuild용 GitHub Enterprise Server 샘플 실행 참조	이 샘플에서는 GitHub Enterprise Server 리포지토리에 인증서가 설치되어 있을 때 CodeBuild 프로젝트를 설정하는 방법을 보여 줍니다. 또한 코드 변경이 GitHub Enterprise Server 리포지토리로 푸시될 때마다 CodeBuild가 다시 빌드되도록 Webhook를 활성화하는 방법도 보여줍니다.
GitHub pull 요청 및 웹후크 필터 샘플 – CodeBuild용 GitHub pull 요청 및 웹후크 필터 샘플 실행 참조	이 샘플에서는 GitHub Enterprise Server 리포지토리를 사용하여 pull 요청을 생성하는 방법을 보여줍니다. 또한 코드 변경이 GitHub Enterprise Server 리포지토리로 푸시될 때마다 CodeBuild가 다시 빌드되도록 Webhook를 활성화하는 방법도 보여줍니다.

CodeBuild용 'Bitbucket pull 요청 및 웹후크 필터' 샘플 실행

AWS CodeBuild 는 소스 리포지토리가 Bitbucket일 때 웹후크를 지원합니다. 즉, 소스 코드가 Bitbucket 리포지토리에 저장된 CodeBuild 빌드 프로젝트의 경우, Webhook는 코드 변경이 리포지토리에 푸시될 때마다 소스 코드를 다시 빌드하는 데 사용할 수 있습니다. 자세한 내용은 Bitbucket Webhook 이벤트 단원을 참조하십시오.

이 샘플에서는 Bitbucket 리포지토리를 사용하여 풀 요청을 생성하는 방법을 보여줍니다. 또한 Bitbucket webhook를 통해 CodeBuild를 트리거하여 프로젝트 빌드를 생성하는 방법을 보여줍니다.

> **Note**
>
> Webhook를 사용할 때 사용자가 예상치 못한 빌드를 트리거할 수 있습니다. 이 위험을 줄이려면 webhook 사용 모범 사례 섹션을 참조하세요.

주제

- 사전 조건
- 1단계: Bitbucket을 사용하여 빌드 프로젝트 생성 및 웹후크 활성화
- 2단계: Bitbucket 웹후크를 사용하여 빌드 트리거

사전 조건

이 샘플을 실행하려면 AWS CodeBuild 프로젝트를 Bitbucket 계정에 연결해야 합니다.

> **Note**
>
> CodeBuild에서 Bitbucket으로 권한을 업데이트했습니다. 이전에 프로젝트를 Bitbucket에 연결했지만 지금 Bitbucket 연결 오류를 수신한 경우 다시 연결하여 CodeBuild에 webhook 관리 권한을 부여해야 합니다.

1단계: Bitbucket을 사용하여 빌드 프로젝트 생성 및 웹후크 활성화

다음 단계에서는 Bitbucket을 소스 리포지토리로 사용하여 AWS CodeBuild 프로젝트를 생성하고 웹후크를 활성화하는 방법을 설명합니다.

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. CodeBuild 정보 페이지가 나타나면 빌드 프로젝트 생성을 선택합니다. 그렇지 않을 경우, 탐색 창에서 빌드를 확장한 후 빌드 프로젝트를 선택하고 빌드 프로젝트 생성을 선택합니다.
3. 빌드 프로젝트 생성을 선택합니다.
4. 프로젝트 구성에서 다음과 같이 합니다.

프로젝트 이름

이 빌드 프로젝트의 이름을 입력합니다. 빌드 프로젝트 이름은 각 AWS 계정에서 고유해야 합니다. 또한 선택에 따라 빌드 프로젝트에 대한 설명을 포함하여 다른 사용자가 이 프로젝트의 용도를 이해하도록 도울 수 있습니다.

5. 소스에서 다음과 같이 합니다.

소스 공급자

Bitbucket을 선택합니다. Bitbucket과 연결(다시 연결)하는 지침을 따르고 승인을 선택합니다.

리포지토리

내 Bitbucket 계정의 리포지토리를 선택합니다.

이전에 Bitbucket 계정에 연결한 적이 없으면 Bitbucket 사용자 이름과 앱 암호를 입력하고 Bitbucket 보안 인증 저장을 선택합니다. Bitbucket API 토큰의 경우 Atlassian 계정 이메일 주소와 API 토큰을 입력합니다.

Bitbucket 리포지토리

Bitbucket 리포지토리의 URL을 입력합니다.

6. 기본 소스 webhook 이벤트에서 다음을 선택합니다.

> **Note**
>
> 기본 소스 webhook 이벤트 섹션은 이전 단계에서 Bitbucket 계정의 리포지토리를 선택한 경우에만 표시됩니다.

1. 프로젝트를 생성할 때 코드 변경이 이 리포지토리로 푸시될 때마다 다시 빌드를 선택합니다.
2. 이벤트 유형에서 하나 이상의 이벤트를 선택합니다.
3. 이벤트가 빌드를 트리거할 때를 필터링하려면 Start a build under these conditions(다음 조건에서 빌드를 시작)에서 하나 이상의 선택적 필터를 추가합니다.
4. 이벤트가 트리거되지 않을 때를 필터링하려면 Don't start a build under these conditions(다음 조건에서 빌드를 시작하지 않음)에서 하나 이상의 선택적 필터를 추가합니다.
5. 필요한 경우 필터 그룹 추가를 선택하여 다른 필터 그룹을 추가합니다.

Bitbucket webhook 이벤트 유형 및 필터에 대한 자세한 내용은 Bitbucket Webhook 이벤트 섹션을 참조하세요.

7. 환경에서 다음과 같이 합니다.

   환경 이미지

   다음 중 하나를 선택합니다.

   다음에서 관리하는 Docker 이미지를 사용하려면 AWS CodeBuild:

   관리형 이미지를 선택한 후 운영 체제, 런타임, 이미지 및 이미지 버전에서 옵션을 선택합니다. 사용 가능한 경우 환경 유형에서 항목을 선택합니다.

   다른 도커 이미지를 사용하려면:

   사용자 지정 이미지를 선택합니다. 환경 유형에서 ARM, Linux, Linux GPU 또는 Windows를 선택합니다. Other registry(다른 레지스트리)를 선택한 경우 External registry URL(외부 레지스트리 URL)에 Docker Hub의 도커 이미지 이름 및 태그를 *docker repository*/*docker image name* 형식으로 입력합니다. Amazon ECR을 선택하는 경우 Amazon ECR 리포지토리와 Amazon ECR 이미지를 사용하여 AWS 계정에서 도커 이미지를 선택합니다.

   프라이빗 도커 이미지를 사용하려면:

   사용자 지정 이미지를 선택합니다. 환경 유형에서 ARM, Linux, Linux GPU 또는 Windows를 선택합니다. Image registry(이미지 레지스트리)에서 Other registry(다른 레지스트리)를 선택한 다음 프라이빗 도커 이미지에 대한 보안 인증 정보의 ARN을 입력합니다. 보안 인증은 Secrets Manager에서 생성됩니다. 자세한 내용은 란 무엇입니까 AWS Secrets Manager?를 참조하세요. AWS Secrets Manager 사용 설명서의 .

   서비스 역할

   다음 중 하나를 선택합니다.

   - CodeBuild 서비스 역할이 없는 경우 새 서비스 역할을 선택합니다. 역할 이름에 새 역할의 이름을 입력합니다.
   - CodeBuild 서비스 역할이 있는 경우 기존 서비스 역할을 선택합니다. 역할 ARN에서 서비스 역할을 선택합니다.

> **Note**
>
> 콘솔을 사용하여 빌드 프로젝트를 생성하거나 업데이트하는 경우, 이와 동시에 CodeBuild 서비스 역할을 만들 수 있습니다. 기본적으로 역할은 해당 빌드 프로젝트에서만 작동합니다. 콘솔을 사용하여 이 서비스 역할을 다른 빌드 프로젝트와 연결하는 경우 다른 빌드 프로젝트에서 작동하도록 역할이 업데이트됩니다. 하나의 서비스 역할은 최대 10개의 빌드 프로젝트에서 작동할 수 있습니다.

8. Buildspec에서 다음 중 하나를 수행합니다.

   - buildspec 파일 사용을 선택하여 소스 코드 루트 디렉터리에 있는 buildspec.yml 파일을 사용합니다.
   - 빌드 명령 삽입을 선택하여 콘솔에서 빌드 명령을 삽입합니다.

   자세한 내용은 buildspec 참조 단원을 참조하십시오.

9. 결과물에서 다음과 같이 합니다.

   유형

   다음 중 하나를 선택합니다.

   - 빌드 출력 아티팩트를 생성하지 않으려면 No artifacts(아티팩트 없음)를 선택합니다.
   - S3 버킷에 빌드 출력을 저장하려면 Amazon S3를 선택하고 다음 작업을 수행합니다.
     - 빌드 출력 ZIP 파일이나 폴더에 프로젝트 이름을 사용하려는 경우 이름을 비워 둡니다. 그렇지 않으면 이름을 입력합니다. 기본적으로 결과물 이름은 프로젝트의 이름입니다. 다른 이름을 사용하려면 결과물 이름 상자에 해당 이름을 입력합니다. ZIP 파일을 출력하려면 zip 확장명을 포함시킵니다.
     - [Bucket name]에서 출력 버킷의 이름을 선택합니다.
     - 이 절차의 앞부분에서 빌드 명령 삽입을 선택한 경우 출력 파일에 빌드 출력 ZIP 파일 또는 폴더에 넣으려는 빌드의 파일 위치를 입력합니다. 위치가 여러 개인 경우 각 위치를 쉼표로 구분합니다(예: `appspec.yml, target/my-app.jar`). 자세한 내용은 buildspec 구문의 `files` 설명을 참조하십시오.

   추가 구성

   Additional configuration(추가 구성)을 확장하고 옵션을 적절하게 설정합니다.

10. 빌드 프로젝트 생성을 선택합니다. 검토 페이지에서 빌드 시작을 선택하여 빌드를 실행합니다.

2단계: Bitbucket 웹후크를 사용하여 빌드 트리거

Bitbucket 웹후크를 사용하는 프로젝트의 경우 Bitbucket 리포지토리가 소스 코드의 변경을 감지하면 빌드를 AWS CodeBuild 생성합니다.

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 빌드 프로젝트를 선택한 다음 webhook에서 Bitbucket 리포지토리와 연결된 프로젝트를 선택합니다. Bitbucket webhook 프로젝트 생성에 대한 자세한 내용은 the section called “1단계: Bitbucket을 사용하여 빌드 프로젝트 생성 및 웹후크 활성화” 섹션을 참조하세요.
3. 프로젝트의 Bitbucket 리포지토리에서 코드를 변경합니다.
4. Bitbucket 리포지토리에서 풀 요청을 생성합니다. 자세한 내용은 API 요청 생성을 참조하십시오.
5. Bitbucket Webhook 페이지에서 View request(요청 보기)를 선택하여 최신 이벤트 목록을 봅니다.
6. 세부 정보 보기를 선택하여 CodeBuild에서 반환되는 응답에 대한 세부 정보를 확인합니다. 값이 다음과 같을 것입니다.

```
"response":"Webhook received and build started: https://us-
east-1.console.aws.amazon.com/codebuild/home..."
"statusCode":200
```

7. Bitbucket 풀 요청 페이지로 이동하여 빌드 상태를 확인합니다.

CodeBuild용 GitHub Enterprise Server 샘플 실행

AWS CodeBuild 는 GitHub Enterprise Server를 소스 리포지토리로 지원합니다. 이 샘플에서는 GitHub Enterprise Server 리포지토리에 인증서가 설치되어 있을 때 CodeBuild 프로젝트를 설정하는 방법을 보여 줍니다. 또한 코드 변경이 GitHub Enterprise Server 리포지토리로 푸시될 때마다 CodeBuild가 다시 빌드되도록 Webhook를 활성화하는 방법도 보여줍니다.

주제

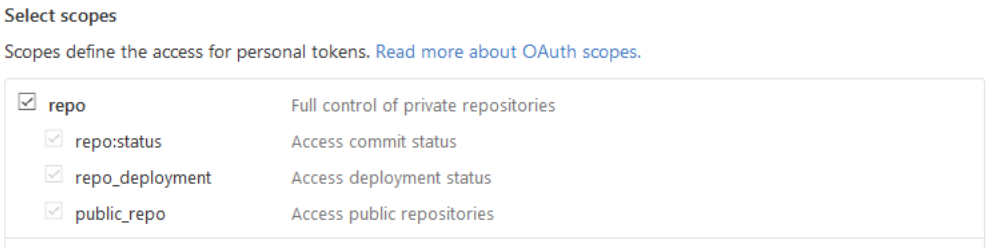
- 사전 조건
- 1단계: GitHub Enterprise Server를 사용하여 빌드 프로젝트 생성 및 웹후크 활성화

사전 조건

1. CodeBuild 프로젝트에 대한 개인 액세스 토큰을 생성합니다. GitHub Enterprise 사용자를 생성하고 이 사용자를 위한 개인 액세스 토큰을 생성하는 것이 좋습니다. 이 토큰을 CodeBuild 프로젝트

를 생성할 때 사용할 수 있도록 클립보드에 복사합니다. 자세한 내용은 GitHub Help 웹 사이트의 Creating a personal access token for the command line을 참조하십시오.

개인 액세스 토큰을 생성할 때 정의에 리포지토리 범위를 포함시킵니다.



2. GitHub Enterprise Server에서 인증서를 다운로드합니다. CodeBuild는 인증서를 사용하여 리포지토리에 신뢰할 수 있는 SSL 연결을 만듭니다.

   Linux/macOS 클라이언트:

   터미널 창에서 다음 명령을 실행합니다.

   ```
   echo -n | openssl s_client -connect HOST:PORTNUMBER \
       | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > /folder/filename.pem
   ```

   명령에서 자리 표시자를 다음 값으로 바꿉니다.

   HOST. GitHub Enterprise Server 리포지토리의 IP 주소입니다.

   PORTNUMBER. 연결에 사용하는 포트 번호입니다(예: 443).

   folder. 인증서를 다운로드한 폴더입니다.

   filename. 인증서 파일의 파일 이름입니다.

   > **Important**
   > 인증서를 .pem 파일로 저장합니다.

   Windows 클라이언트:

   브라우저를 사용하여 GitHub Enterprise Server에서 인증서를 다운로드합니다. 사이트의 인증서 세부 정보를 보려면 자물쇠 아이콘을 선택합니다. 인증서를 내보내는 방법에 대한 자세한 내용은 브라우저 설명서를 참조하십시오.

> **Important**
>
> 인증서를 .pem 파일로 저장합니다.

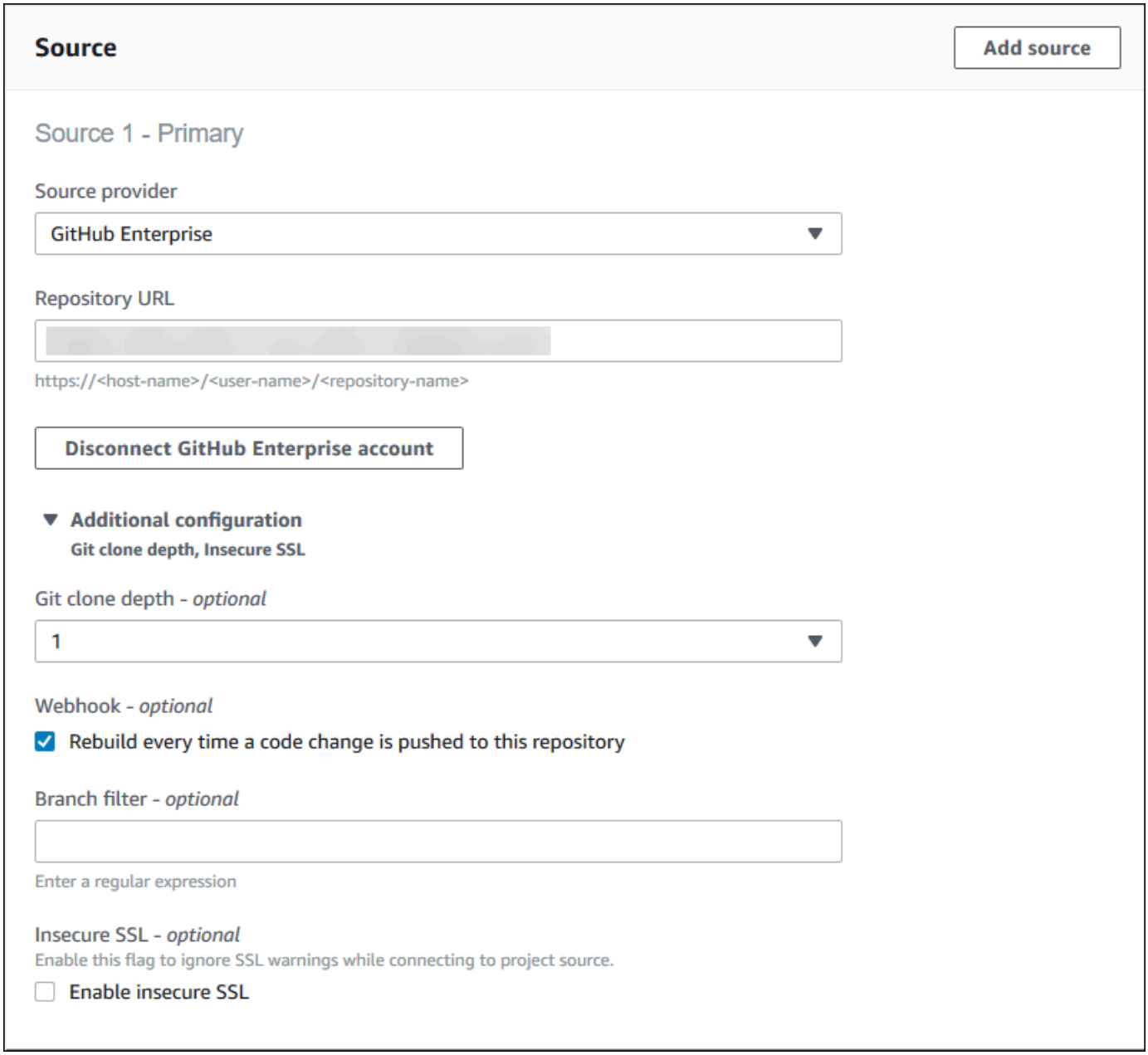
3. S3 버킷으로 인증서 파일을 업로드합니다. S3 버킷을 생성하는 방법에 대한 자세한 내용은 S3 버킷을 생성하려면 어떻게 해야 합니까?를 참조하십시오. S3 버킷으로 객체를 업로드하는 방법에 대한 자세한 내용은 버킷에 파일 및 폴더를 업로드하려면 어떻게 해야 합니까?를 참조하십시오.

> **Note**
>
> 이 버킷은 빌드와 동일한 AWS 리전에 있어야 합니다. 예를 들어, CodeBuild가 미국 동부(오하이오) 리전에서 빌드를 실행하도록 명령을 지정하는 경우, 버킷도 미국 동부(오하이오) 리전에 있어야 합니다.

1단계: GitHub Enterprise Server를 사용하여 빌드 프로젝트 생성 및 웹후크 활성화

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. CodeBuild 정보 페이지가 나타나면 빌드 프로젝트 생성을 선택합니다. 그렇지 않을 경우, 탐색 창에서 빌드를 확장한 후 빌드 프로젝트를 선택하고 빌드 프로젝트 생성을 선택합니다.
3. 프로젝트 이름에 이 빌드 프로젝트의 이름을 입력합니다. 빌드 프로젝트 이름은 각 AWS 계정에서 고유해야 합니다. 또한 선택에 따라 빌드 프로젝트에 대한 설명을 포함하여 다른 사용자가 이 프로젝트의 용도를 이해하도록 도울 수 있습니다.
4. 소스의 소스 공급자에서 GitHub Enterprise Server를 선택합니다.
   - 계정 자격 증명 관리를 선택한 다음 개인 액세스 토큰을 선택합니다. 서비스에서 Secrets Manager(권장)를 선택하고 보안 암호를 구성합니다. 그런 다음 GitHub Enterprise 개인 액세스 토큰에 개인 액세스 토큰을 입력하고 저장을 선택합니다.
   - 리포지토리 URL에 리포지토리 이름을 포함하여 리포지토리에 대한 경로를 입력합니다.
   - 추가 구성을 확장합니다.
   - 코드 변경이 이 리포지토리로 푸시될 때마다 다시 빌드하려면 코드 변경이 이 리포지토리로 푸시될 때마다 다시 빌드를 선택합니다.
   - GitHub Enterprise Server 프로젝트 리포지토리에 연결되어 있는 동안 SSL 경고를 무시하려면 안전하지 않은 SSL 활성화를 선택합니다.

> **Note**
>
> Enable insecure SSL(안전하지 않은 SSL 활성화)는 테스트 용도로만 사용하는 것이 좋습니다. 프로덕션 환경에 사용하면 안 됩니다.



5. 환경에서 다음과 같이 합니다.

   [Environment image]에서 다음 중 하나를 수행합니다.

- 에서 관리하는 도커 이미지를 사용하려면 관리형 이미지를 AWS CodeBuild선택한 다음 운영 체제, 런타임, 이미지(Image) 및 이미지 버전에서 선택합니다. 사용 가능한 경우 환경 유형에서 항목을 선택합니다.
- 다른 도커 이미지를 사용하려면 사용자 지정 이미지를 선택합니다. 환경 유형에서 ARM, Linux, Linux GPU 또는 Windows를 선택합니다. Other registry(다른 레지스트리)를 선택한 경우 External registry URL(외부 레지스트리 URL)에 Docker Hub의 도커 이미지 이름 및 태그를 *docker repository*/*docker image name* 형식으로 입력합니다. Amazon ECR을 선택하는 경우 Amazon ECR 리포지토리와 Amazon ECR 이미지를 사용하여 AWS 계정에서 도커 이미지를 선택합니다.
- 프라이빗 도커 이미지를 사용하려면 사용자 지정 이미지를 선택합니다. 환경 유형에서 ARM, Linux, Linux GPU 또는 Windows를 선택합니다. Image registry(이미지 레지스트리)에서 Other registry(다른 레지스트리)를 선택한 다음 프라이빗 도커 이미지에 대한 보안 인증 정보의 ARN을 입력합니다. 보안 인증은 Secrets Manager에서 생성됩니다. 자세한 내용은 AWS Secrets Manager사용 설명서의 AWS Secrets Manager 이란? 섹션을 참조하세요.

6. 서비스 역할에서 다음 중 하나를 수행합니다.

   - CodeBuild 서비스 역할이 없는 경우 새 서비스 역할을 선택합니다. 역할 이름에 새 역할의 이름을 입력합니다.
   - CodeBuild 서비스 역할이 있는 경우 기존 서비스 역할을 선택합니다. 역할 ARN에서 서비스 역할을 선택합니다.

   > **Note**
   >
   > 콘솔을 사용하여 빌드 프로젝트를 생성하거나 업데이트하는 경우, 이와 동시에 CodeBuild 서비스 역할을 만들 수 있습니다. 기본적으로 역할은 해당 빌드 프로젝트에서만 작동합니다. 콘솔을 사용하여 이 서비스 역할을 다른 빌드 프로젝트와 연결하는 경우 다른 빌드 프로젝트에서 작동하도록 역할이 업데이트됩니다. 하나의 서비스 역할은 최대 10개의 빌드 프로젝트에서 작동할 수 있습니다.

7. 추가 구성을 확장합니다.

   CodeBuild를 사용하여 VPC에서 작업을 수행하려는 경우:

   - VPC에서 CodeBuild가 사용하는 VPC ID를 선택합니다.
   - VPC 서브넷에서 CodeBuild가 사용하는 리소스가 포함된 서브넷을 선택합니다.

- VPC 보안 그룹에서 CodeBuild가 VPC의 리소스에 대한 액세스를 허용하기 위해 사용하는 보안 그룹을 선택합니다.

자세한 내용은 Amazon Virtual Private Cloud AWS CodeBuild 와 함께 사용 단원을 참조하십시오.

8. Buildspec에서 다음 중 하나를 수행합니다.

   - buildspec 파일 사용을 선택하여 소스 코드 루트 디렉터리에 있는 buildspec.yml 파일을 사용합니다.
   - 빌드 명령 삽입을 선택하여 콘솔에서 빌드 명령을 삽입합니다.

   자세한 내용은 buildspec 참조 단원을 참조하십시오.

9. 결과물의 유형에서 다음 중 하나를 수행합니다.

   - 빌드 출력 아티팩트를 생성하지 않으려면 No artifacts(아티팩트 없음)를 선택합니다.
   - S3 버킷에 빌드 출력을 저장하려면 Amazon S3를 선택하고 다음 작업을 수행합니다.
     - 빌드 출력 ZIP 파일이나 폴더에 프로젝트 이름을 사용하려는 경우 이름을 비워 둡니다. 그렇지 않으면 이름을 입력합니다. 기본적으로 결과물 이름은 프로젝트의 이름입니다. 다른 이름을 사용하려면 결과물 이름 상자에 해당 이름을 입력합니다. ZIP 파일을 출력하려면 zip 확장명을 포함시킵니다.
     - [Bucket name]에서 출력 버킷의 이름을 선택합니다.
     - 이 절차의 앞부분에서 빌드 명령 삽입을 선택한 경우 출력 파일에 빌드 출력 ZIP 파일 또는 폴더에 넣으려는 빌드의 파일 위치를 입력합니다. 위치가 여러 개인 경우 각 위치를 쉼표로 구분합니다(예: `appspec.yml, target/my-app.jar`). 자세한 내용은 buildspec 구문의 `files` 설명을 참조하십시오.

10. Cache type(캐시 유형)에서 다음 중 하나를 선택합니다.

    - 캐시를 사용하지 않으려면 [No cache]를 선택합니다.
    - Amazon S3 캐시를 사용하려면 Amazon S3를 선택하고 다음을 수행합니다.
      - 버킷에서 캐시가 저장된 S3 버킷의 이름을 선택합니다.
      - (선택 사항) 캐시 경로 접두사에 Amazon S3 경로 접두사를 입력합니다. Cache path prefix(캐시 경로 접두사) 값은 디렉터리 이름과 비슷합니다. 따라서 캐시를 버킷의 동일한 디렉터리에 저장할 수 있습니다.

> **Important**
>
> 경로 접두사 끝에 후행 슬래시(/)를 추가하지 마십시오.

- 로컬 캐시를 사용하려면 로컬을 선택한 다음 하나 이상의 로컬 캐시 모드를 선택해야 합니다.

> **Note**
>
> Docker 계층 캐시 모드는 Linux에서만 사용할 수 있습니다. 이 모드를 선택할 경우 프로젝트를 권한이 있는 모드에서 실행해야 합니다.

캐시를 사용하면 빌드 환경의 재사용 가능한 특정 부분이 캐시에 저장되고 빌드 전반에서 사용되기 때문에 상당한 빌드 시간을 절약할 수 있습니다. buildspec 파일에 캐시를 지정하는 것에 대한 자세한 정보는 buildspec 구문 단원을 참조하십시오. 캐싱에 대한 자세한 정보는 성능을 개선하기 위한 캐시 빌드을 참조하십시오.

11. 빌드 프로젝트 생성을 선택합니다. 빌드 프로젝트 페이지에서 빌드 시작을 선택합니다.

CodeBuild용 GitHub pull 요청 및 웹후크 필터 샘플 실행

AWS CodeBuild 는 소스 리포지토리가 GitHub인 경우 웹후크를 지원합니다. 즉, 소스 코드가 GitHub 리포지토리에 저장된 CodeBuild 빌드 프로젝트의 경우, Webhook는 코드 변경이 리포지토리에 푸시될 때마다 소스 코드를 다시 빌드하는 데 사용할 수 있습니다. CodeBuild 샘플은 AWS CodeBuild 샘플을 참조하세요.

> **Note**
>
> Webhook를 사용할 때 사용자가 예상치 못한 빌드를 트리거할 수 있습니다. 이 위험을 줄이려면 webhook 사용 모범 사례 섹션을 참조하세요.

주제

- 1단계: GitHub로 빌드 프로젝트 생성 및 웹후크 활성화
- 2단계: 웹후크가 활성화되어 있는지 확인

1단계: GitHub로 빌드 프로젝트 생성 및 웹후크 활성화

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. CodeBuild 정보 페이지가 나타나면 빌드 프로젝트 생성을 선택합니다. 그렇지 않을 경우, 탐색 창에서 빌드를 확장한 후 빌드 프로젝트를 선택하고 빌드 프로젝트 생성을 선택합니다.
3. 빌드 프로젝트 생성을 선택합니다.
4. 프로젝트 구성에서 다음과 같이 합니다.

   프로젝트 이름

   이 빌드 프로젝트의 이름을 입력합니다. 빌드 프로젝트 이름은 각 AWS 계정에서 고유해야 합니다. 또한 선택에 따라 빌드 프로젝트에 대한 설명을 포함하여 다른 사용자가 이 프로젝트의 용도를 이해하도록 도울 수 있습니다.
5. 소스에서 다음과 같이 합니다.

   소스 공급자

   GitHub를 선택하세요. GitHub와 연결(다시 연결)하는 지침을 따르고 승인을 선택합니다.

   리포지토리

   내 GitHub 계정의 리포지토리를 선택합니다.

   GitHub 리포지토리

   GitHub 리포지토리의 URL을 입력합니다.
6. 기본 소스 webhook 이벤트에서 다음을 선택합니다.

   > **Note**
   >
   > 기본 소스 webhook 이벤트 섹션은 이전 단계에서 내 GitHub 계정의 리포지토리를 선택한 경우에만 표시됩니다.

   1. 프로젝트를 생성할 때 코드 변경이 이 리포지토리로 푸시될 때마다 다시 빌드를 선택합니다.
   2. 이벤트 유형에서 하나 이상의 이벤트를 선택합니다.
   3. 이벤트가 빌드를 트리거할 때를 필터링하려면 Start a build under these conditions(다음 조건에서 빌드를 시작)에서 하나 이상의 선택적 필터를 추가합니다.

4. 이벤트가 트리거되지 않을 때를 필터링하려면 Don't start a build under these conditions(다음 조건에서 빌드를 시작하지 않음)에서 하나 이상의 선택적 필터를 추가합니다.
5. 필요한 경우 필터 그룹 추가를 선택하여 다른 필터 그룹을 추가합니다.

GitHub Webhook 이벤트 유형 및 필터에 대한 자세한 내용은 GitHub Webhook 이벤트 섹션을 참조하세요.

7. 환경에서 다음과 같이 합니다.

환경 이미지

다음 중 하나를 선택합니다.

다음에서 관리하는 Docker 이미지를 사용하려면 AWS CodeBuild:

관리형 이미지를 선택한 후 운영 체제, 런타임, 이미지 및 이미지 버전에서 옵션을 선택합니다. 사용 가능한 경우 환경 유형에서 항목을 선택합니다.

다른 도커 이미지를 사용하려면:

사용자 지정 이미지를 선택합니다. 환경 유형에서 ARM, Linux, Linux GPU 또는 Windows를 선택합니다. Other registry(다른 레지스트리)를 선택한 경우 External registry URL(외부 레지스트리 URL)에 Docker Hub의 도커 이미지 이름 및 태그를 *docker repository*/*docker image name* 형식으로 입력합니다. Amazon ECR을 선택하는 경우 Amazon ECR 리포지토리 및 Amazon ECR 이미지를 사용하여 AWS 계정의 Docker 이미지를 선택합니다.

프라이빗 도커 이미지를 사용하려면:

사용자 지정 이미지를 선택합니다. 환경 유형에서 ARM, Linux, Linux GPU 또는 Windows를 선택합니다. Image registry(이미지 레지스트리)에서 Other registry(다른 레지스트리)를 선택한 다음 프라이빗 도커 이미지에 대한 보안 인증 정보의 ARN을 입력합니다. 보안 인증은 Secrets Manager에서 생성됩니다. 자세한 내용은 란 무엇입니까 AWS Secrets Manager?를 참조하세요. AWS Secrets Manager 사용 설명서의 .

서비스 역할

다음 중 하나를 선택합니다.

- CodeBuild 서비스 역할이 없는 경우 새 서비스 역할을 선택합니다. 역할 이름에 새 역할의 이름을 입력합니다.

- CodeBuild 서비스 역할이 있는 경우 기존 서비스 역할을 선택합니다. 역할 ARN에서 서비스 역할을 선택합니다.

> **Note**
>
> 콘솔을 사용하여 빌드 프로젝트를 생성하거나 업데이트하는 경우, 이와 동시에 CodeBuild 서비스 역할을 만들 수 있습니다. 기본적으로 역할은 해당 빌드 프로젝트에서만 작동합니다. 콘솔을 사용하여 이 서비스 역할을 다른 빌드 프로젝트와 연결하는 경우 다른 빌드 프로젝트에서 작동하도록 역할이 업데이트됩니다. 하나의 서비스 역할은 최대 10개의 빌드 프로젝트에서 작동할 수 있습니다.

8. Buildspec에서 다음 중 하나를 수행합니다.

   - buildspec 파일 사용을 선택하여 소스 코드 루트 디렉터리에 있는 buildspec.yml 파일을 사용합니다.
   - 빌드 명령 삽입을 선택하여 콘솔에서 빌드 명령을 삽입합니다.

   자세한 내용은 buildspec 참조 단원을 참조하십시오.

9. 결과물에서 다음과 같이 합니다.

   유형

   다음 중 하나를 선택합니다.

   - 빌드 출력 아티팩트를 생성하지 않으려면 No artifacts(아티팩트 없음)를 선택합니다.
   - S3 버킷에 빌드 출력을 저장하려면 Amazon S3를 선택하고 다음 작업을 수행합니다.
     - 빌드 출력 ZIP 파일이나 폴더에 프로젝트 이름을 사용하려는 경우 이름을 비워 둡니다. 그렇지 않으면 이름을 입력합니다. 기본적으로 결과물 이름은 프로젝트의 이름입니다. 다른 이름을 사용하려면 결과물 이름 상자에 해당 이름을 입력합니다. ZIP 파일을 출력하려면 zip 확장명을 포함시킵니다.
     - [Bucket name]에서 출력 버킷의 이름을 선택합니다.
     - 이 절차의 앞부분에서 빌드 명령 삽입을 선택한 경우 출력 파일에 빌드 출력 ZIP 파일 또는 폴더에 넣으려는 빌드의 파일 위치를 입력합니다. 위치가 여러 개인 경우 각 위치를 쉼표로 구분합니다(예: `appspec.yml, target/my-app.jar`). 자세한 내용은 buildspec 구문의 `files` 설명을 참조하십시오.

추가 구성

Additional configuration(추가 구성)을 확장하고 옵션을 적절하게 설정합니다.

10. 빌드 프로젝트 생성을 선택합니다. 검토 페이지에서 빌드 시작을 선택하여 빌드를 실행합니다.

2단계: 웹후크가 활성화되어 있는지 확인

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 프로젝트 빌드를 선택합니다.
3. 다음 중 하나를 수행하세요.
   - webhook를 검증하려는 빌드 프로젝트의 링크를 선택한 후 빌드 세부 정보를 선택합니다.
   - webhook를 검증하려는 빌드 프로젝트 옆에 있는 라디오 버튼을 선택하고 세부 정보 보기를 선택한 후 빌드 세부 정보를 선택합니다.
4. 기본 소스 webhook 이벤트에서 Webhook URL 링크를 선택합니다.
5. GitHub 리포지토리의 설정 페이지에 있는 웹후크에서 Pull 요청 및 Push가 선택되었는지 확인합니다.
6. GitHub 프로필 설정의 개인 설정, 애플리케이션, 승인된 OAuth 앱에서 선택한 AWS 리전에 액세스할 수 있는 권한이 애플리케이션에 부여되었음을 확인할 수 있습니다.

자습서: 인증서 스토리지에 S3를 사용하여 CodeBuild에서 Fastlane을 사용한 Apple 코드 서명

fastlane은 iOS 및 Android 앱의 베타 배포 및 릴리스를 자동화하는 오픈 소스 자동화 도구로 널리 사용됩니다. 스크린샷 생성, 코드 서명 처리, 애플리케이션 릴리스와 같은 모든 지루한 작업을 처리합니다.

사전 조건

이 자습서를 완료하려면 먼저 다음과 같은 사전 조건을 설정해야 합니다.

- 의 경우 AWS 계정
- Apple 개발자 계정
- 인증서를 저장하기 위한 S3 버킷
- 프로젝트에 설치된 fastlane - fastlane 설치 가이드

1단계: 로컬 시스템에서 S3를 사용하여 Fastlane Match 설정

Fastlane Match는 Fastlane 도구 중 하나이며 로컬 개발 환경과 CodeBuild 모두에서 코드 서명을 위한 원활한 구성을 지원합니다. Fastlane Match는 모든 코드 서명 인증서와 프로비저닝 프로필을 Git 리포지토리/S3 버킷/Google 클라우드 스토리지에 저장하고, 필요한 경우 필요한 인증서와 프로필을 다운로드하여 설치합니다.

이 예제 구성에서는 Amazon S3 버킷을 설정하고 스토리지에 사용합니다.

1. 프로젝트에서 일치 항목 초기화:

```
fastlane match init
```

2. 메시지가 표시되면 스토리지 모드로 S3를 선택합니다.
3. S3를 사용하도록 `Matchfile`을 업데이트합니다.

```
storage_mode("s3")
   s3_bucket("your-s3-bucket-name")
   s3_region("your-aws-region")
   type("appstore") # The default type, can be: appstore, adhoc, enterprise or
 development
```

2단계: Fastfile 설정

다음 경로로 `Fastfile`을 생성하거나 업데이트합니다.

CodeBuild에서는 앱을 빌드하고 서명할 때마다 Fastlane Match를 실행해야 합니다. 이를 수행하는 가장 쉬운 방법은 앱을 빌드하는 경로에 `match` 작업을 추가하는 것입니다.

```
default_platform(:ios)

platform :ios do
  before_all do
    setup_ci
  end

  desc "Build and sign the app"
  lane :build do
    match(type: "appstore", readonly: true)
```

```
    gym(
      scheme: "YourScheme",
      export_method: "app-store"
    )
  end
end
```

> **Note**
>
> 일치 작업이 올바르게 작동하려면 `Fastfile`의 `before_all` 섹션에 `setup_ci`를 추가해야 합니다. 이렇게 하면 적절한 권한이 있는 임시 Fastlane 키체인이 사용됩니다. 이를 사용하지 않으면 빌드 실패 또는 일관되지 않은 결과가 표시될 수 있습니다.

3단계: `fastlane match` 명령을 실행하여 각 인증서 및 프로필 생성

지정된 유형(예: 개발, 앱 스토어, 임시, 엔터프라이즈)에 대한 fastlane match 명령은 원격 스토어에서 사용할 수 없는 경우 인증서와 프로필을 생성합니다. 인증서와 프로필은 fastlane에서 S3에 저장합니다.

```
bundle exec fastlane match appstore
```

명령 실행은 대화식으로 이루어지며 fastlane은 인증서 복호화를 위한 암호 설정을 요청합니다.

4단계: 프로젝트에 대한 애플리케이션 파일 생성

프로젝트에 적합한 애플리케이션 파일을 생성하거나 추가합니다.

1. 프로젝트 빌드 요구 사항에 따라 Gymfile, Appfile, Snapfile, Deliverfile을 생성하거나 추가합니다.
2. 원격 리포지토리에 변경 사항을 커밋합니다.

5단계: Secrets Manager에서 환경 변수 생성

fastlane 세션 쿠키 및 일치하는 암호를 저장하기 위한 보안 암호 2개를 생성합니다. Secrets Manager에서 보안 암호를 생성하는 방법에 대한 자세한 내용은 AWS Secrets Manager 보안 암호 생성을 참조하세요.

1. 다음과 같이 fastlane 세션 쿠키에 액세스합니다.

   a. 보안 암호 키 - FASTLANE_SESSION

   b. 보안 암호 값 - 로컬 시스템에서 다음 명령을 실행하여 생성된 세션 쿠키입니다.

   > **Note**
   >
   > 이 값은 ~/.fastlane/spaceship/my_appleid_username/cookie 로컬 파일에서 인증 후 사용할 수 있습니다.

   ```
   fastlane spaceauth -u <apple account>
   ```

2. Fastlane Match 암호 - Fastlane Match가 S3 버킷에 저장된 인증서와 프로필을 해독하도록 하려면 매치 설정 단계에서 구성한 암호화 암호를 CodeBuild 프로젝트의 환경 변수에 추가해야 합니다.

   a. 보안 암호 키 - MATCH_PASSWORD

   b. 보안 암호 값 - *<### #### ### ##>* 3단계에서 인증서를 생성하는 동안 암호가 설정됩니다.

> **Note**
>
> Secrets Manager에서 위의 보안 암호를 생성할 때는 /CodeBuild/ 접두사를 사용하여 보안 암호 이름을 지정해야 합니다.

6단계: 컴퓨팅 플릿 생성

프로젝트의 컴퓨팅 플릿을 생성합니다.

1. 콘솔에서 CodeBuild로 이동하여 새 컴퓨팅 플릿을 생성합니다.
2. 'macOS'를 운영 체제로 선택하고 적절한 컴퓨팅 유형과 이미지를 선택합니다.

7단계: CodeBuild에서 프로젝트 생성

CodeBuild에서 프로젝트를 생성합니다.

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 빌드 프로젝트를 생성합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 및 빌드 실행(콘솔) 섹션을 참조하세요.
3. 소스 공급자(예: GitHub, CodeCommit)를 설정합니다. 이는 iOS 프로젝트 소스 리포지토리이며 인증서 리포지토리가 아닙니다.
4. 환경에서 다음과 같이 합니다.
   - 예약 용량을 선택합니다.
   - 플릿에서 위에서 생성한 플릿을 선택합니다.
   - CodeBuild가 자동으로 생성할 서비스 역할의 이름을 입력합니다.
   - 아래 환경 변수를 제공합니다.
     - 이름: `MATCH_PASSWORD`, 값: *`<secrets arn>`*, 유형: Secrets Manager(MATCH_PASSWORD의 경우 5단계에서 생성된 보안 암호 ARN)
     - 이름: `FASTLANE_SESSION`, 값: *`<secrets arn>`*, 유형: Secrets Manager(FASTLANE_SESSION의 경우 5단계에서 생성된 보안 암호 ARN)
5. Buildspec에서 다음을 추가합니다.

```
version: 0.2

phases:
  install:
    commands:
      - gem install bundler
      - bundle install
  build:
    commands:
      - echo "Building and signing the app..."
      - bundle exec fastlane build
  post_build:
    commands:
      - echo "Build completed on date"

artifacts:
  files:
    - '*/.ipa'
  name: app-$(date +%Y-%m-%d)
```

8단계: IAM 역할 구성

프로젝트가 생성되면 CodeBuild 프로젝트의 서비스 역할에 인증서가 포함된 S3 버킷에 액세스할 수 있는 권한이 있는지 확인합니다. 다음 정책을 역할에 추가하세요.

JSON

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

9단계: 빌드 실행

빌드를 실행합니다. CodeBuild에서 빌드 상태 및 로그를 검토할 수 있습니다.

작업이 완료되면 작업 로그를 볼 수 있습니다.

문제 해결

- 인증서 가져오기에 문제가 발생하면 S3 액세스를 위해 IAM 권한이 올바르게 설정되었는지 확인합니다.

- 인증서 복호화에 문제가 발생하면 MATCH_PASSWORD 환경 변수에서 올바른 암호를 설정해야 합니다.
- 코드 서명 문제의 경우 Apple Developer 계정에 필요한 인증서 및 프로필이 있고 Xcode 프로젝트의 번들 식별자가 프로비저닝 프로필의 번들 식별자와 일치하는지 확인합니다.

보안 고려 사항

다음은 이 자습서의 보안 고려 사항입니다.

- S3 버킷에 저장 데이터 암호화를 비롯한 적절한 보안 설정이 있는지 확인합니다. 특히 버킷에 퍼블릭 액세스 권한이 없는지 확인하고 액세스 권한이 필요한 CodeBuild 및 시스템으로만 액세스를 제한합니다.
- 를 사용하여 MATCH_PASSWORD 및 FASTLANE_SESSION과 같은 민감한 정보를 AWS Secrets Manager 저장하는 것이 좋습니다.

이 샘플은 인증서 스토리지용 Amazon S3를 통해 CodeBuild에서 Fastlane을 사용하여 iOS 코드 서명을 위한 설정을 제공합니다. 특정 프로젝트 요구 사항 및 CodeBuild 환경에 따라 일부 단계를 조정해야 할 수 있습니다. 이 접근 방식은 AWS 서비스를 활용하여 AWS 에코시스템 내에서 보안 및 통합을 강화합니다.

자습서: 인증서 스토리지에 GitHub를 사용하여 CodeBuild에서 Fastlane을 사용한 Apple 코드 서명

fastlane은 iOS 및 Android 앱의 베타 배포 및 릴리스를 자동화하는 오픈 소스 자동화 도구로 널리 사용됩니다. 스크린샷 생성, 코드 서명 처리, 애플리케이션 릴리스와 같은 모든 지루한 작업을 처리합니다.

이 샘플은 GitHub를 인증서 및 프로비저닝 프로필의 스토리지로 사용하여 Mac 플릿에서 실행되는 CodeBuild 프로젝트에서 Fastlane을 사용하여 Apple 코드 서명을 설정하는 방법을 보여줍니다.

사전 조건

이 자습서를 완료하려면 먼저 다음과 같은 사전 조건을 설정해야 합니다.

- 의 경우 AWS 계정
- Apple 개발자 계정
- 인증서를 저장하기 위한 프라이빗 GitHub 리포지토리

- 프로젝트에 설치된 fastlane - fastlane 설치 가이드

1단계: 로컬 시스템에서 GitHub를 사용하여 Fastlane Match 설정

Fastlane Match는 Fastlane 도구 중 하나이며 로컬 개발 환경과 CodeBuild 모두에서 코드 서명을 위한 원활한 구성을 지원합니다. Fastlane Match는 모든 코드 서명 인증서와 프로비저닝 프로필을 Git 리포지토리/S3 버킷/Google 클라우드 스토리지에 저장하고, 필요한 경우 필요한 인증서와 프로필을 다운로드하여 설치합니다.

이 예제 구성에서는 스토리지용 Git 리포지토리를 설정하고 사용합니다.

1. 프로젝트에서 일치 항목 초기화:

```
fastlane match init
```

2. 메시지가 표시되면 GitHub를 스토리지 모드로 선택합니다.
3. GitHub를 사용하도록 `Matchfile`을 업데이트합니다.

```
git_url("https://github.com/your-username/your-certificate-repo.git")
storage_mode("git")
type("development") # The default type, can be: appstore, adhoc, enterprise or
 development
```

> **Note**
>
> fastlane이 성공적으로 인증하고 복제할 수 있도록 Git 리포지토리에 HTTPS URL을 입력해야 합니다. 그렇지 않으면 일치 항목을 사용하려고 할 때 인증 오류가 표시될 수 있습니다.

2단계: Fastfile 설정

다음 경로로 `Fastfile`을 생성하거나 업데이트합니다.

CodeBuild에서는 앱을 빌드하고 서명할 때마다 Fastlane Match를 실행해야 합니다. 이를 수행하는 가장 쉬운 방법은 앱을 빌드하는 경로에 `match` 작업을 추가하는 것입니다.

```
default_platform(:ios)
```

```
platform :ios do
  before_all do
    setup_ci
  end

  desc "Build and sign the app"
  lane :build do
    match(type: "appstore", readonly: true)
    gym(
      scheme: "YourScheme",
      export_method: "app-store"
    )
  end
end
```

> **Note**
>
> 일치 작업이 올바르게 작동하려면 `Fastfile`의 `before_all` 섹션에 `setup_ci`를 추가해야 합니다. 이렇게 하면 적절한 권한이 있는 임시 Fastlane 키체인이 사용됩니다. 이를 사용하지 않으면 빌드 실패 또는 일관되지 않은 결과가 표시될 수 있습니다.

3단계: **`fastlane match`** 명령을 실행하여 각 인증서 및 프로필 생성

지정된 유형(예: 개발, 앱 스토어, 임시, 엔터프라이즈)에 대한 fastlane match 명령은 원격 스토어에서 사용할 수 없는 경우 인증서와 프로필을 생성합니다. 인증서와 프로필은 fastlane에서 GitHub에 저장합니다.

```
bundle exec fastlane match appstore
```

명령 실행은 대화식으로 이루어지며 fastlane은 인증서 복호화를 위한 암호 설정을 요청합니다.

4단계: 프로젝트에 대한 애플리케이션 파일 생성

프로젝트에 적합한 애플리케이션 파일을 생성하거나 추가합니다.

1. 프로젝트 빌드 요구 사항에 따라 Gymfile, Appfile, Snapfile, Deliverfile을 생성하거나 추가합니다.
2. 원격 리포지토리에 변경 사항을 커밋합니다.

5단계: Secrets Manager에서 환경 변수 생성

fastlane 세션 쿠키 및 일치하는 암호를 저장하기 위한 보안 암호 3개를 생성합니다. Secrets Manager에서 보안 암호를 생성하는 방법에 대한 자세한 내용은 AWS Secrets Manager 보안 암호 생성을 참조하세요.

1. 다음과 같이 fastlane 세션 쿠키에 액세스합니다.

   a. 보안 암호 키 - `FASTLANE_SESSION`

   b. 보안 암호 값 - 로컬 시스템에서 다음 명령을 실행하여 생성된 세션 쿠키입니다.

   > **Note**
   >
   > 이 값은 `~/.fastlane/spaceship/my_appleid_username/cookie` 로컬 파일에서 인증 후 사용할 수 있습니다.

   ```
   fastlane spaceauth -u <Apple_account>
   ```

2. Fastlane Match 암호 - Fastlane Match가 Git 리포지토리에 저장된 인증서와 프로필을 해독하도록 하려면 매치 설정 단계에서 구성한 암호화 암호를 CodeBuild 프로젝트의 환경 변수에 추가해야 합니다.

   a. 보안 암호 키 - `MATCH_PASSWORD`

   b. 보안 암호 값 - `<match passphrase to decrypt certificates>` 3단계에서 인증서를 생성하는 동안 암호가 설정됩니다.

3. Fastlane `MATCH_GIT_BASIC_AUTHORIZATION` - 일치에 대한 기본 권한 부여를 설정합니다.

   a. 보안 암호 키:

   `MATCH_GIT_BASIC_AUTHORIZATION`

   b. 보안 암호 값 - 값은 `username:password` 형식의 사용자 이름 및 개인 액세스 토큰(PAT)의 base64 인코딩 문자열이어야 합니다. 다음 명령을 사용하여 생성할 수 있습니다.

   ```
   echo -n your_github_username:your_personal_access_token | base64
   ```

GitHub 콘솔의 프로필 > 설정 > 개발자 설정 > 개인 액세스 토큰에서 PAT를 생성할 수 있습니다. 자세한 내용은 https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens 가이드를 참조하세요.

> **Note**
>
> Secrets Manager에서 위의 보안 암호를 생성할 때는 `/CodeBuild/` 접두사를 사용하여 보안 암호 이름을 지정해야 합니다.

6단계: 컴퓨팅 플릿 생성

프로젝트의 컴퓨팅 플릿을 생성합니다.

1. 콘솔에서 CodeBuild로 이동하여 새 컴퓨팅 플릿을 생성합니다.
2. `macOS`를 운영 체제로 선택하고 적절한 컴퓨팅 유형과 이미지를 선택합니다.

7단계: CodeBuild에서 프로젝트 생성

CodeBuild에서 프로젝트를 생성합니다.

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 빌드 프로젝트를 생성합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 및 빌드 실행(콘솔) 섹션을 참조하세요.
3. 소스 공급자(예: GitHub, CodeCommit)를 설정합니다. 이는 iOS 프로젝트 소스 리포지토리이며 인증서 리포지토리가 아닙니다.
4. 환경에서 다음과 같이 합니다.
   - 예약 용량을 선택합니다.
   - 플릿에서 위에서 생성한 플릿을 선택합니다.
   - CodeBuild가 자동으로 생성할 서비스 역할의 이름을 입력합니다.
   - 아래 환경 변수를 제공합니다.
     - 이름: `MATCH_PASSWORD`, 값: *`<secrets arn>`*, 유형: Secrets Manager(MATCH_PASSWORD의 경우 5단계에서 생성된 보안 암호 ARN)

- 이름: FASTLANE_SESSION, 값: *<secrets arn>*, 유형: Secrets Manager(FASTLANE_SESSION의 경우 5단계에서 생성된 보안 암호 ARN)
- 이름: MATCH_GIT_BASIC_AUTHORIZATION, 값: *<secrets ARN>*, 유형: Secrets Manager Secrets ARN(MATCH_GIT_BASIC_AUTHORIZATION의 경우 5단계에서 생성됨)

5. Buildspec에서 다음을 추가합니다.

```
version: 0.2

phases:
  install:
    commands:
      - gem install bundler
      - bundle install
  build:
    commands:
      - echo "Building and signing the app..."
      - bundle exec fastlane build
  post_build:
    commands:
      - echo "Build completed on date"

artifacts:
  files:
    - '*/.ipa'
  name: app-$(date +%Y-%m-%d)
```

8단계: 빌드 실행

빌드를 실행합니다. CodeBuild에서 빌드 상태 및 로그를 검토할 수 있습니다.

작업이 완료되면 작업 로그를 볼 수 있습니다.

문제 해결

- GitHub 리포지토리에 액세스하는 데 문제가 발생하면 개인 액세스 토큰과 MATCH_GIT_BASIC_AUTHORIZATION 환경 변수를 다시 확인합니다.
- 인증서 복호화에 문제가 발생하면 MATCH_PASSWORD 환경 변수에서 올바른 암호를 설정해야 합니다.

- 코드 서명 문제의 경우 Apple Developer 계정에 필요한 인증서 및 프로필이 있고 Xcode 프로젝트의 번들 식별자가 프로비저닝 프로필의 번들 식별자와 일치하는지 확인합니다.

보안 고려 사항

다음은 이 자습서의 보안 고려 사항입니다.

- 인증서용 GitHub 리포지토리를 비공개로 유지하고 액세스를 정기적으로 감사합니다.
- 를 사용하여 MATCH_PASSWORD 및 FASTLANE_SESSION과 같은 민감한 정보를 AWS Secrets Manager 저장하는 것이 좋습니다.

이 샘플은 인증서 스토리지용 GitHub를 통해 CodeBuild에서 Fastlane을 사용하여 iOS 코드 서명을 위한 설정을 제공합니다. 특정 프로젝트 요구 사항 및 CodeBuild 환경에 따라 일부 단계를 조정해야 할 수 있습니다. 이 접근 방식은 AWS 서비스를 활용하여 AWS 에코시스템 내에서 보안 및 통합을 강화합니다.

의미 체계 버전 관리를 사용하여 빌드 시 아티팩트 이름 설정

이번 샘플에는 빌드 시 생성되는 아티팩트 이름을 지정하는 방법에 대한 buildspec 파일 예제가 포함되어 있습니다. buildspec 파일에서 지정하는 이름에 Shell 명령과 환경 변수를 포함시켜 고유성을 유지할 수 있습니다. 또한 buildspec 파일에서 지정하는 이름은 프로젝트 생성 시 콘솔에 입력하는 이름을 재정의합니다.

여러 차례 빌드하는 경우 buildspec 파일에서 지정한 아티팩트 이름을 사용하면 출력 아티팩트 파일 이름의 고유성을 유지할 수 있습니다. 예를 들어 빌드 시 날짜와 타임스탬프를 사용해 아티팩트 이름에 삽입할 수 있습니다.

콘솔에서 입력한 아티팩트 이름을 buildspec 파일의 이름으로 재정의하고 싶다면 다음과 같이 실행하십시오.

1. 아티팩트 이름을 buildspec 파일의 이름으로 재정의하도록 빌드 프로젝트를 설정합니다.

   - 콘솔을 사용하여 빌드 프로젝트를 생성하는 경우 의미 체계 버전 관리 사용을 선택합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 단원을 참조하십시오.
   - 를 사용하는 경우 AWS CLI`overrideArtifactName`에 전달된 JSON 형식 파일에서를 true로 설정합니다`create-project`. 자세한 내용은 빌드 프로젝트 생성(AWS CLI) 단원을 참조하십시오.

- AWS CodeBuild API를 사용하는 경우 프로젝트가 생성 또는 업데이트되거나 빌드가 시작될 때 `ProjectArtifacts` 객체에 `overrideArtifactName` 플래그를 설정합니다.

2. buildspec 파일에서 이름을 지정합니다. 아래 buildspec 파일 샘플을 참고하십시오.

아래 Linux 예제는 빌드가 생성된 날짜를 포함시켜 아티팩트 이름을 지정하는 방법을 보여 줍니다.

```
version: 0.2
phases:
  build:
    commands:
      - rspec HelloWorld_spec.rb
artifacts:
  files:
    - '**/*'
  name: myname-$(date +%Y-%m-%d)
```

아래 Linux 예제는 CodeBuild 환경 변수를 사용하여 아티팩트 이름을 지정하는 방법을 보여 줍니다. 자세한 내용은 빌드 환경의 환경 변수 단원을 참조하십시오.

```
version: 0.2
phases:
  build:
    commands:
      - rspec HelloWorld_spec.rb
artifacts:
  files:
    - '**/*'
  name: myname-$AWS_REGION
```

아래 Windows 예제는 빌드가 생성된 날짜와 시간을 포함시켜 아티팩트 이름을 지정하는 방법에 대한 설명입니다.

```
version: 0.2
env:
  variables:
    TEST_ENV_VARIABLE: myArtifactName
phases:
  build:
    commands:
      - cd samples/helloworld
```

```
      - dotnet restore
      - dotnet run
artifacts:
  files:
    - '**/*'
  name: $Env:TEST_ENV_VARIABLE-$(Get-Date -UFormat "%Y%m%d-%H%M%S")
```

아래 Windows 예제는 buildspec 파일에서 선언한 변수와 CodeBuild 환경 변수를 사용하여 아티팩트 이름을 지정하는 방법을 보여 줍니다. 자세한 내용은 빌드 환경의 환경 변수 단원을 참조하십시오.

```
version: 0.2
env:
  variables:
    TEST_ENV_VARIABLE: myArtifactName
phases:
  build:
    commands:
      - cd samples/helloworld
      - dotnet restore
      - dotnet run
artifacts:
  files:
    - '**/*'
  name: $Env:TEST_ENV_VARIABLE-$Env:AWS_REGION
```

자세한 내용은 CodeBuild의 빌드 사양 참조 단원을 참조하십시오.

CodeBuild용 Microsoft Windows 샘플 실행

이러한 샘플은 Microsoft Windows Server 2019, .NET Framework 및 .NET Core SDK를 실행하는 AWS CodeBuild 빌드 환경을 사용하여 F# 및 Visual Basic으로 작성된 코드에서 런타임 파일을 빌드합니다.

> **Important**
>
> 이러한 샘플을 실행하면 AWS 계정에 요금이 부과될 수 있습니다. 여기에는 CodeBuild와 Amazon S3 및 CloudWatch Logs와 관련된 AWS 리소스 AWS KMS및 작업에 대해 발생할 수 있는 요금이 포함됩니다. 자세한 내용은 CodeBuild 요금, Amazon S3 요금, AWS Key Management Service 요금 및 Amazon CloudWatch 요금을 참조하세요.

Windows 샘플 실행

다음 절차에 따라 Windows 샘플을 실행합니다.

Windows 샘플을 실행하려면

1. 이 주제의 디렉터리 구조 및 파일 섹션에 설명된 대로 파일을 생성한 다음, 이를 S3 입력 버킷이나 CodeCommit 또는 GitHub 리포지토리에 업로드합니다.

   > **Important**
   >
   > *(root directory name)*은 업로드하지 말고, *(root directory name)* 안에 있는 파일만 업로드하십시오.
   > S3 입력 버킷을 사용하고 있는 경우, 파일을 포함하는 ZIP 파일을 생성한 다음, 이를 입력 버킷에 업로드하십시오. *(root directory name)*을 ZIP 파일에 추가하지 말고, *(root directory name)* 안에 있는 파일만 추가하십시오.

2. 빌드 프로젝트를 생성합니다. 빌드 프로젝트는 `mcr.microsoft.com/dotnet/framework/sdk:4.8` 이미지를 사용하여 .NET Framework 프로젝트를 빌드해야 합니다.

   AWS CLI 를 사용하여 빌드 프로젝트를 생성하는 경우 `create-project` 명령에 대한 JSON 형식의 입력이 이와 비슷할 수 있습니다. (자리 표시자는 사용자의 값으로 바꾸십시오.)

   ```
   {
   ```

```
  "name": "sample-windows-build-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-bucket/windows-build-input-
artifact.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket",
    "packaging": "ZIP",
    "name": "windows-build-output-artifact.zip"
  },
  "environment": {
    "type": "WINDOWS_SERVER_2019_CONTAINER",
    "image": "mcr.microsoft.com/dotnet/framework/sdk:4.8",
    "computeType": "BUILD_GENERAL1_MEDIUM"
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

3. 빌드를 실행하고 빌드를 수동으로 실행의 단계를 따르세요.
4. 빌드 출력 아티팩트를 얻으려면 S3 출력 버킷에서 *windows-build-output-artifact*.zip 파일을 로컬 컴퓨터나 인스턴스에 다운로드합니다. 콘텐츠를 추출하여 런타임 및 기타 파일로 이동합니다.

   - NET Framework, FSharpHelloWorld.exe를 사용하는 F# 샘플의 런타임 파일은 FSharpHelloWorld\bin\Debug 디렉터리에 있습니다.
   - .NET Framework, VBHelloWorld.exe를 사용하는 Visual Basic 샘플의 런타임 파일은 VBHelloWorld\bin\Debug 디렉터리에 있습니다.

디렉터리 구조

이 샘플은 다음과 같은 디렉터리 구조를 가정합니다.

F# 및 .NET Framework

```
(root directory name)
### buildspec.yml
### FSharpHelloWorld.sln
### FSharpHelloWorld
```

```
    ### App.config
    ### AssemblyInfo.fs
    ### FSharpHelloWorld.fsproj
    ### Program.fs
```

Visual Basic 및 .NET Framework

```
(root directory name)
### buildspec.yml
### VBHelloWorld.sln
### VBHelloWorld
    ### App.config
    ### HelloWorld.vb
    ### VBHelloWorld.vbproj
    ### My Project
        ### Application.Designer.vb
        ### Application.myapp
        ### AssemblyInfo.vb
        ### Resources.Designer.vb
        ### Resources.resx
        ### Settings.Designer.vb
        ### Settings.settings
```

파일

이러한 샘플은 다음 파일을 사용합니다.

F# 및 .NET Framework

`buildspec.yml`(*(root directory name)*에 있음):

```
version: 0.2

env:
  variables:
    SOLUTION: .\FSharpHelloWorld.sln
    PACKAGE_DIRECTORY: .\packages
    DOTNET_FRAMEWORK: 4.8

phases:
  build:
    commands:
```

```
      - '& nuget restore $env:SOLUTION -PackagesDirectory $env:PACKAGE_DIRECTORY'
      - '& msbuild -p:FrameworkPathOverride="C:\Program Files (x86)\Reference
 Assemblies\Microsoft\Framework\.NETFramework\v$env:DOTNET_FRAMEWORK" $env:SOLUTION'
artifacts:
  files:
    - .\FSharpHelloWorld\bin\Debug\*
```

FSharpHelloWorld.sln(*(root directory name)*에 있음):

```
Microsoft Visual Studio Solution File, Format Version 12.00
# Visual Studio 14
VisualStudioVersion = 14.0.25420.1
MinimumVisualStudioVersion = 10.0.40219.1
Project("{F2A71F9B-5D33-465A-A702-920D77279786}") = "FSharpHelloWorld",
 "FSharpHelloWorld\FSharpHelloWorld.fsproj", "{D60939B6-526D-43F4-9A89-577B2980DF62}"
EndProject
Global
  GlobalSection(SolutionConfigurationPlatforms) = preSolution
    Debug|Any CPU = Debug|Any CPU
    Release|Any CPU = Release|Any CPU
  EndGlobalSection
  GlobalSection(ProjectConfigurationPlatforms) = postSolution
    {D60939B6-526D-43F4-9A89-577B2980DF62}.Debug|Any CPU.ActiveCfg = Debug|Any CPU
    {D60939B6-526D-43F4-9A89-577B2980DF62}.Debug|Any CPU.Build.0 = Debug|Any CPU
    {D60939B6-526D-43F4-9A89-577B2980DF62}.Release|Any CPU.ActiveCfg = Release|Any CPU
    {D60939B6-526D-43F4-9A89-577B2980DF62}.Release|Any CPU.Build.0 = Release|Any CPU
  EndGlobalSection
  GlobalSection(SolutionProperties) = preSolution
    HideSolutionNode = FALSE
  EndGlobalSection
EndGlobal
```

App.config(*(root directory name)*\FSharpHelloWorld에 있음):

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.8" />
  </startup>
</configuration>
```

AssemblyInfo.fs(*(root directory name)*\FSharpHelloWorld에 있음):

```
namespace FSharpHelloWorld.AssemblyInfo

open System.Reflection
open System.Runtime.CompilerServices
open System.Runtime.InteropServices

// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[<assembly: AssemblyTitle("FSharpHelloWorld")>]
[<assembly: AssemblyDescription("")>]
[<assembly: AssemblyConfiguration("")>]
[<assembly: AssemblyCompany("")>]
[<assembly: AssemblyProduct("FSharpHelloWorld")>]
[<assembly: AssemblyCopyright("Copyright ©  2017")>]
[<assembly: AssemblyTrademark("")>]
[<assembly: AssemblyCulture("")>]

// Setting ComVisible to false makes the types in this assembly not visible
// to COM components.  If you need to access a type in this assembly from
// COM, set the ComVisible attribute to true on that type.
[<assembly: ComVisible(false)>]

// The following GUID is for the ID of the typelib if this project is exposed to COM
[<assembly: Guid("d60939b6-526d-43f4-9a89-577b2980df62")>]

// Version information for an assembly consists of the following four values:
//
// Major Version
// Minor Version
// Build Number
// Revision
//
// You can specify all the values or you can default the Build and Revision Numbers
// by using the '*' as shown below:
// [<assembly: AssemblyVersion("1.0.*")>]
[<assembly: AssemblyVersion("1.0.0.0")>]
[<assembly: AssemblyFileVersion("1.0.0.0")>]

do
  ()
```

FSharpHelloWorld.fsproj(*(root directory name)*\FSharpHelloWorld에 있음):

```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="14.0" DefaultTargets="Build" xmlns="http://
schemas.microsoft.com/developer/msbuild/2003">
  <Import Project="$(MSBuildExtensionsPath)\
$(MSBuildToolsVersion)\Microsoft.Common.props"
 Condition="Exists('$(MSBuildExtensionsPath)\
$(MSBuildToolsVersion)\Microsoft.Common.props')" />
  <PropertyGroup>
    <Configuration Condition=" '$(Configuration)' == '' ">Debug</Configuration>
    <Platform Condition=" '$(Platform)' == '' ">AnyCPU</Platform>
    <SchemaVersion>2.0</SchemaVersion>
    <ProjectGuid>d60939b6-526d-43f4-9a89-577b2980df62</ProjectGuid>
    <OutputType>Exe</OutputType>
    <RootNamespace>FSharpHelloWorld</RootNamespace>
    <AssemblyName>FSharpHelloWorld</AssemblyName>
    <TargetFrameworkVersion>v4.8</TargetFrameworkVersion>
    <AutoGenerateBindingRedirects>true</AutoGenerateBindingRedirects>
    <TargetFSharpCoreVersion>4.4.0.0</TargetFSharpCoreVersion>
    <Name>FSharpHelloWorld</Name>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
    <DebugSymbols>true</DebugSymbols>
    <DebugType>full</DebugType>
    <Optimize>false</Optimize>
    <Tailcalls>false</Tailcalls>
    <OutputPath>bin\Debug\</OutputPath>
    <DefineConstants>DEBUG;TRACE</DefineConstants>
    <WarningLevel>3</WarningLevel>
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DocumentationFile>bin\Debug\FSharpHelloWorld.XML</DocumentationFile>
    <Prefer32Bit>true</Prefer32Bit>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Release|AnyCPU' ">
    <DebugType>pdbonly</DebugType>
    <Optimize>true</Optimize>
    <Tailcalls>true</Tailcalls>
    <OutputPath>bin\Release\</OutputPath>
    <DefineConstants>TRACE</DefineConstants>
    <WarningLevel>3</WarningLevel>
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DocumentationFile>bin\Release\FSharpHelloWorld.XML</DocumentationFile>
    <Prefer32Bit>true</Prefer32Bit>
  </PropertyGroup>
```

```
  <ItemGroup>
    <Reference Include="mscorlib" />
    <Reference Include="FSharp.Core, Version=$(TargetFSharpCoreVersion),
 Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a">
      <Private>True</Private>
    </Reference>
    <Reference Include="System" />
    <Reference Include="System.Core" />
    <Reference Include="System.Numerics" />
  </ItemGroup>
  <ItemGroup>
    <Compile Include="AssemblyInfo.fs" />
    <Compile Include="Program.fs" />
    <None Include="App.config" />
  </ItemGroup>
  <PropertyGroup>
    <MinimumVisualStudioVersion Condition="'$(MinimumVisualStudioVersion)' == ''">11</
MinimumVisualStudioVersion>
  </PropertyGroup>
  <Choose>
    <When Condition="'$(VisualStudioVersion)' == '11.0'">
      <PropertyGroup Condition="Exists('$(MSBuildExtensionsPath32)\..\Microsoft SDKs\F#
\3.0\Framework\v4.0\Microsoft.FSharp.Targets')">
        <FSharpTargetsPath>$(MSBuildExtensionsPath32)\..\Microsoft SDKs\F#
\3.0\Framework\v4.0\Microsoft.FSharp.Targets</FSharpTargetsPath>
      </PropertyGroup>
    </When>
    <Otherwise>
      <PropertyGroup Condition="Exists('$(MSBuildExtensionsPath32)\Microsoft
\VisualStudio\v$(VisualStudioVersion)\FSharp\Microsoft.FSharp.Targets')">
        <FSharpTargetsPath>$(MSBuildExtensionsPath32)\Microsoft\VisualStudio\v
$(VisualStudioVersion)\FSharp\Microsoft.FSharp.Targets</FSharpTargetsPath>
      </PropertyGroup>
    </Otherwise>
  </Choose>
  <Import Project="$(FSharpTargetsPath)" />
  <!-- To modify your build process, add your task inside one of the targets below and
 uncomment it.
       Other similar extension points exist, see Microsoft.Common.targets.
  <Target Name="BeforeBuild">
  </Target>
  <Target Name="AfterBuild">
  </Target>
  -->
```

```
</Project>
```

Program.fs(*(root directory name)*\FSharpHelloWorld에 있음):

```
// Learn more about F# at http://fsharp.org
// See the 'F# Tutorial' project for more help.

[<EntryPoint>]
let main argv =
  printfn "Hello World"
  0 // return an integer exit code
```

Visual Basic 및 .NET Framework

buildspec.yml(*(root directory name)*에 있음):

```
version: 0.2

env:
  variables:
    SOLUTION: .\VBHelloWorld.sln
    PACKAGE_DIRECTORY: .\packages
    DOTNET_FRAMEWORK: 4.8

phases:
  build:
    commands:
      - '& "C:\ProgramData\chocolatey\bin\NuGet.exe" restore $env:SOLUTION -
PackagesDirectory $env:PACKAGE_DIRECTORY'
      - '& "C:\Program Files (x86)\MSBuild\14.0\Bin\MSBuild.exe" -
p:FrameworkPathOverride="C:\Program Files (x86)\Reference Assemblies\Microsoft
\Framework\.NETFramework\v$env:DOTNET_FRAMEWORK" $env:SOLUTION'
artifacts:
  files:
    - .\VBHelloWorld\bin\Debug\*
```

VBHelloWorld.sln(*(root directory name)*에 있음):

```
Microsoft Visual Studio Solution File, Format Version 12.00
# Visual Studio 14
VisualStudioVersion = 14.0.25420.1
```

```
MinimumVisualStudioVersion = 10.0.40219.1
Project("{F184B08F-C81C-45F6-A57F-5ABD9991F28F}") = "VBHelloWorld", "VBHelloWorld
\VBHelloWorld.vbproj", "{4DCEC446-7156-4FE6-8CCC-219E34DD409D}"
EndProject
Global
  GlobalSection(SolutionConfigurationPlatforms) = preSolution
    Debug|Any CPU = Debug|Any CPU
    Release|Any CPU = Release|Any CPU
  EndGlobalSection
  GlobalSection(ProjectConfigurationPlatforms) = postSolution
    {4DCEC446-7156-4FE6-8CCC-219E34DD409D}.Debug|Any CPU.ActiveCfg = Debug|Any CPU
    {4DCEC446-7156-4FE6-8CCC-219E34DD409D}.Debug|Any CPU.Build.0 = Debug|Any CPU
    {4DCEC446-7156-4FE6-8CCC-219E34DD409D}.Release|Any CPU.ActiveCfg = Release|Any CPU
    {4DCEC446-7156-4FE6-8CCC-219E34DD409D}.Release|Any CPU.Build.0 = Release|Any CPU
  EndGlobalSection
  GlobalSection(SolutionProperties) = preSolution
    HideSolutionNode = FALSE
  EndGlobalSection
EndGlobal
```

App.config(*(root directory name)*\VBHelloWorld에 있음):

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.8" />
  </startup>
</configuration>
```

HelloWorld.vb(*(root directory name)*\VBHelloWorld에 있음):

```
Module HelloWorld

  Sub Main()
    MsgBox("Hello World")
  End Sub

End Module
```

VBHelloWorld.vbproj(*(root directory name)*\VBHelloWorld에 있음):

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<Project ToolsVersion="14.0" DefaultTargets="Build" xmlns="http://
schemas.microsoft.com/developer/msbuild/2003">
  <Import Project="$(MSBuildExtensionsPath)\
$(MSBuildToolsVersion)\Microsoft.Common.props"
 Condition="Exists('$(MSBuildExtensionsPath)\
$(MSBuildToolsVersion)\Microsoft.Common.props')" />
  <PropertyGroup>
    <Configuration Condition=" '$(Configuration)' == '' ">Debug</Configuration>
    <Platform Condition=" '$(Platform)' == '' ">AnyCPU</Platform>
    <ProjectGuid>{4DCEC446-7156-4FE6-8CCC-219E34DD409D}</ProjectGuid>
    <OutputType>Exe</OutputType>
    <StartupObject>VBHelloWorld.HelloWorld</StartupObject>
    <RootNamespace>VBHelloWorld</RootNamespace>
    <AssemblyName>VBHelloWorld</AssemblyName>
    <FileAlignment>512</FileAlignment>
    <MyType>Console</MyType>
    <TargetFrameworkVersion>v4.8</TargetFrameworkVersion>
    <AutoGenerateBindingRedirects>true</AutoGenerateBindingRedirects>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DebugSymbols>true</DebugSymbols>
    <DebugType>full</DebugType>
    <DefineDebug>true</DefineDebug>
    <DefineTrace>true</DefineTrace>
    <OutputPath>bin\Debug\</OutputPath>
    <DocumentationFile>VBHelloWorld.xml</DocumentationFile>
    <NoWarn>42016,41999,42017,42018,42019,42032,42036,42020,42021,42022</NoWarn>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Release|AnyCPU' ">
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DebugType>pdbonly</DebugType>
    <DefineDebug>false</DefineDebug>
    <DefineTrace>true</DefineTrace>
    <Optimize>true</Optimize>
    <OutputPath>bin\Release\</OutputPath>
    <DocumentationFile>VBHelloWorld.xml</DocumentationFile>
    <NoWarn>42016,41999,42017,42018,42019,42032,42036,42020,42021,42022</NoWarn>
  </PropertyGroup>
  <PropertyGroup>
    <OptionExplicit>On</OptionExplicit>
  </PropertyGroup>
  <PropertyGroup>
    <OptionCompare>Binary</OptionCompare>
```

```
  </PropertyGroup>
  <PropertyGroup>
    <OptionStrict>Off</OptionStrict>
  </PropertyGroup>
  <PropertyGroup>
    <OptionInfer>On</OptionInfer>
  </PropertyGroup>
  <ItemGroup>
    <Reference Include="System" />
    <Reference Include="System.Data" />
    <Reference Include="System.Deployment" />
    <Reference Include="System.Xml" />
    <Reference Include="System.Core" />
    <Reference Include="System.Xml.Linq" />
    <Reference Include="System.Data.DataSetExtensions" />
    <Reference Include="System.Net.Http" />
  </ItemGroup>
  <ItemGroup>
    <Import Include="Microsoft.VisualBasic" />
    <Import Include="System" />
    <Import Include="System.Collections" />
    <Import Include="System.Collections.Generic" />
    <Import Include="System.Data" />
    <Import Include="System.Diagnostics" />
    <Import Include="System.Linq" />
    <Import Include="System.Xml.Linq" />
    <Import Include="System.Threading.Tasks" />
  </ItemGroup>
  <ItemGroup>
    <Compile Include="HelloWorld.vb" />
    <Compile Include="My Project\AssemblyInfo.vb" />
    <Compile Include="My Project\Application.Designer.vb">
      <AutoGen>True</AutoGen>
      <DependentUpon>Application.myapp</DependentUpon>
    </Compile>
    <Compile Include="My Project\Resources.Designer.vb">
      <AutoGen>True</AutoGen>
      <DesignTime>True</DesignTime>
      <DependentUpon>Resources.resx</DependentUpon>
    </Compile>
    <Compile Include="My Project\Settings.Designer.vb">
      <AutoGen>True</AutoGen>
      <DependentUpon>Settings.settings</DependentUpon>
      <DesignTimeSharedInput>True</DesignTimeSharedInput>
```

```
    </Compile>
  </ItemGroup>
  <ItemGroup>
    <EmbeddedResource Include="My Project\Resources.resx">
      <Generator>VbMyResourcesResXFileCodeGenerator</Generator>
      <LastGenOutput>Resources.Designer.vb</LastGenOutput>
      <CustomToolNamespace>My.Resources</CustomToolNamespace>
      <SubType>Designer</SubType>
    </EmbeddedResource>
  </ItemGroup>
  <ItemGroup>
    <None Include="My Project\Application.myapp">
      <Generator>MyApplicationCodeGenerator</Generator>
      <LastGenOutput>Application.Designer.vb</LastGenOutput>
    </None>
    <None Include="My Project\Settings.settings">
      <Generator>SettingsSingleFileGenerator</Generator>
      <CustomToolNamespace>My</CustomToolNamespace>
      <LastGenOutput>Settings.Designer.vb</LastGenOutput>
    </None>
    <None Include="App.config" />
  </ItemGroup>
  <Import Project="$(MSBuildToolsPath)\Microsoft.VisualBasic.targets" />
  <!-- To modify your build process, add your task inside one of the targets below and
 uncomment it.
       Other similar extension points exist, see Microsoft.Common.targets.
  <Target Name="BeforeBuild">
  </Target>
  <Target Name="AfterBuild">
  </Target>
  -->
</Project>
```

Application.Designer.vb(*root directory name*\VBHelloWorld\My Project에 있음):

```
'------------------------------------------------------------------------------
' <auto-generated>
'   This code was generated by a tool.
'   Runtime Version:4.0.30319.42000
'
'   Changes to this file may cause incorrect behavior and will be lost if
'   the code is regenerated.
```

```
' </auto-generated>
'------------------------------------------------------------------------------

Option Strict On
Option Explicit On
```

Application.myapp(*(root directory name)*\VBHelloWorld\My Project에 있음):

```
<?xml version="1.0" encoding="utf-8"?>
<MyApplicationData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <MySubMain>false</MySubMain>
  <SingleInstance>false</SingleInstance>
  <ShutdownMode>0</ShutdownMode>
  <EnableVisualStyles>true</EnableVisualStyles>
  <AuthenticationMode>0</AuthenticationMode>
  <ApplicationType>2</ApplicationType>
  <SaveMySettingsOnExit>true</SaveMySettingsOnExit>
</MyApplicationData>
```

AssemblyInfo.vb(*(root directory name)*\VBHelloWorld\My Project에 있음):

```
Imports System
Imports System.Reflection
Imports System.Runtime.InteropServices

' General Information about an assembly is controlled through the following
' set of attributes. Change these attribute values to modify the information
' associated with an assembly.

' Review the values of the assembly attributes

<Assembly: AssemblyTitle("VBHelloWorld")>
<Assembly: AssemblyDescription("")>
<Assembly: AssemblyCompany("")>
<Assembly: AssemblyProduct("VBHelloWorld")>
<Assembly: AssemblyCopyright("Copyright ©  2017")>
<Assembly: AssemblyTrademark("")>

<Assembly: ComVisible(False)>

'The following GUID is for the ID of the typelib if this project is exposed to COM
<Assembly: Guid("137c362b-36ef-4c3e-84ab-f95082487a5a")>
```

```
' Version information for an assembly consists of the following four values:
'
' Major Version
' Minor Version
' Build Number
' Revision
'
' You can specify all the values or you can default the Build and Revision Numbers
' by using the '*' as shown below:
' <Assembly: AssemblyVersion("1.0.*")>

<Assembly: AssemblyVersion("1.0.0.0")>
<Assembly: AssemblyFileVersion("1.0.0.0")>
```

Resources.Designer.vb(*(root directory name)*\VBHelloWorld\My Project에 있음):

```
'------------------------------------------------------------------------------
' <auto-generated>
'   This code was generated by a tool.
'   Runtime Version:4.0.30319.42000
'
'   Changes to this file may cause incorrect behavior and will be lost if
'   the code is regenerated.
' </auto-generated>
'------------------------------------------------------------------------------

Option Strict On
Option Explicit On

Namespace My.Resources

  'This class was auto-generated by the StronglyTypedResourceBuilder
  'class via a tool like ResGen or Visual Studio.
  'To add or remove a member, edit your .ResX file then rerun ResGen
  'with the /str option, or rebuild your VS project.
  '''<summary>
  '''  A strongly-typed resource class, for looking up localized strings, etc.
  '''</summary>

 <Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyTypedRe
 "4.0.0.0"), _
  Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), _
```

```
 Global.System.Runtime.CompilerServices.CompilerGeneratedAttribute(), _
 Global.Microsoft.VisualBasic.HideModuleNameAttribute()> _
 Friend Module Resources

   Private resourceMan As Global.System.Resources.ResourceManager

   Private resourceCulture As Global.System.Globalization.CultureInfo

   '''<summary>
   '''  Returns the cached ResourceManager instance used by this class.
   '''</summary>

<Global.System.ComponentModel.EditorBrowsableAttribute(Global.System.ComponentModel.EditorBrow
_
   Friend ReadOnly Property ResourceManager() As
Global.System.Resources.ResourceManager
     Get
       If Object.ReferenceEquals(resourceMan, Nothing) Then
         Dim temp As Global.System.Resources.ResourceManager = New
Global.System.Resources.ResourceManager("VBHelloWorld.Resources",
GetType(Resources).Assembly)
         resourceMan = temp
       End If
       Return resourceMan
     End Get
   End Property

   '''<summary>
   '''  Overrides the current thread's CurrentUICulture property for all
   '''  resource lookups using this strongly typed resource class.
   '''</summary>

<Global.System.ComponentModel.EditorBrowsableAttribute(Global.System.ComponentModel.EditorBrow
_
   Friend Property Culture() As Global.System.Globalization.CultureInfo
     Get
       Return resourceCulture
     End Get
     Set(ByVal value As Global.System.Globalization.CultureInfo)
       resourceCulture = value
     End Set
   End Property
 End Module
```

```
End Namespace
```

Resources.resx(*(root directory name)*\VBHelloWorld\My Project에 있음):

```
<?xml version="1.0" encoding="utf-8"?>
<root>
  <!--
    Microsoft ResX Schema

    Version 2.0

    The primary goals of this format is to allow a simple XML format
    that is mostly human readable. The generation and parsing of the
    various data types are done through the TypeConverter classes
    associated with the data types.

    Example:

    ... ado.net/XML headers & schema ...
    <resheader name="resmimetype">text/microsoft-resx</resheader>
    <resheader name="version">2.0</resheader>
    <resheader name="reader">System.Resources.ResXResourceReader,
System.Windows.Forms, ...</resheader>
    <resheader name="writer">System.Resources.ResXResourceWriter,
System.Windows.Forms, ...</resheader>
    <data name="Name1"><value>this is my long string</value><comment>this is a
comment</comment></data>
    <data name="Color1" type="System.Drawing.Color, System.Drawing">Blue</data>
    <data name="Bitmap1" mimetype="application/x-microsoft.net.object.binary.base64">
      <value>[base64 mime encoded serialized .NET Framework object]</value>
    </data>
    <data name="Icon1" type="System.Drawing.Icon, System.Drawing"
mimetype="application/x-microsoft.net.object.bytearray.base64">
      <value>[base64 mime encoded string representing a byte array form of the .NET
Framework object]</value>
      <comment>This is a comment</comment>
    </data>

    There are any number of "resheader" rows that contain simple
    name/value pairs.

    Each data row contains a name, and value. The row also contains a
    type or mimetype. Type corresponds to a .NET class that support
```

```
    text/value conversion through the TypeConverter architecture.
    Classes that don't support this are serialized and stored with the
    mimetype set.

    The mimetype is used for serialized objects, and tells the
    ResXResourceReader how to depersist the object. This is currently not
    extensible. For a given mimetype the value must be set accordingly:

    Note - application/x-microsoft.net.object.binary.base64 is the format
    that the ResXResourceWriter will generate, however the reader can
    read any of the formats listed below.

    mimetype: application/x-microsoft.net.object.binary.base64
    value   : The object must be serialized with
            : System.Serialization.Formatters.Binary.BinaryFormatter
            : and then encoded with base64 encoding.

    mimetype: application/x-microsoft.net.object.soap.base64
    value   : The object must be serialized with
            : System.Runtime.Serialization.Formatters.Soap.SoapFormatter
            : and then encoded with base64 encoding.

    mimetype: application/x-microsoft.net.object.bytearray.base64
    value   : The object must be serialized into a byte array
            : using a System.ComponentModel.TypeConverter
            : and then encoded with base64 encoding.
 -->
 <xsd:schema id="root" xmlns="" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
   <xsd:element name="root" msdata:IsDataSet="true">
     <xsd:complexType>
       <xsd:choice maxOccurs="unbounded">
         <xsd:element name="metadata">
           <xsd:complexType>
             <xsd:sequence>
               <xsd:element name="value" type="xsd:string" minOccurs="0" />
             </xsd:sequence>
             <xsd:attribute name="name" type="xsd:string" />
             <xsd:attribute name="type" type="xsd:string" />
             <xsd:attribute name="mimetype" type="xsd:string" />
           </xsd:complexType>
         </xsd:element>
         <xsd:element name="assembly">
           <xsd:complexType>
```

```
              <xsd:attribute name="alias" type="xsd:string" />
              <xsd:attribute name="name" type="xsd:string" />
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="data">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0"
msdata:Ordinal="1" />
                <xsd:element name="comment" type="xsd:string" minOccurs="0"
msdata:Ordinal="2" />
              </xsd:sequence>
              <xsd:attribute name="name" type="xsd:string" msdata:Ordinal="1" />
              <xsd:attribute name="type" type="xsd:string" msdata:Ordinal="3" />
              <xsd:attribute name="mimetype" type="xsd:string" msdata:Ordinal="4" />
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="resheader">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0"
msdata:Ordinal="1" />
              </xsd:sequence>
              <xsd:attribute name="name" type="xsd:string" use="required" />
            </xsd:complexType>
          </xsd:element>
        </xsd:choice>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
  <resheader name="resmimetype">
    <value>text/microsoft-resx</value>
  </resheader>
  <resheader name="version">
    <value>2.0</value>
  </resheader>
  <resheader name="reader">
    <value>System.Resources.ResXResourceReader, System.Windows.Forms, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </resheader>
  <resheader name="writer">
    <value>System.Resources.ResXResourceWriter, System.Windows.Forms, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </resheader>
```

```
</root>
```

Settings.Designer.vb(*(root directory name)*\VBHelloWorld\My Project에 있음):

```
'------------------------------------------------------------------------------
' <auto-generated>
'     This code was generated by a tool.
'     Runtime Version:4.0.30319.42000
'
'     Changes to this file may cause incorrect behavior and will be lost if
'     the code is regenerated.
' </auto-generated>
'------------------------------------------------------------------------------

Option Strict On
Option Explicit On

Namespace My

  <Global.System.Runtime.CompilerServices.CompilerGeneratedAttribute(), _

 Global.System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.VisualStudio.Editors.Settings
 "11.0.0.0"), _

 Global.System.ComponentModel.EditorBrowsableAttribute(Global.System.ComponentModel.EditorBrows
 _
  Partial Friend NotInheritable Class MySettings
    Inherits Global.System.Configuration.ApplicationSettingsBase

    Private Shared defaultInstance As MySettings =
 CType(Global.System.Configuration.ApplicationSettingsBase.Synchronized(New
 MySettings), MySettings)

    #Region "My.Settings Auto-Save Functionality"
      #If _MyType = "WindowsForms" Then
        Private Shared addedHandler As Boolean

        Private Shared addedHandlerLockObject As New Object

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute(),
 Global.System.ComponentModel.EditorBrowsableAttribute(Global.System.ComponentModel.EditorBrows
 _
```

```
        Private Shared Sub AutoSaveSettings(ByVal sender As Global.System.Object, ByVal
 e As Global.System.EventArgs)
          If My.Application.SaveMySettingsOnExit Then
            My.Settings.Save()
          End If
        End Sub
      #End If
    #End Region

    Public Shared ReadOnly Property [Default]() As MySettings
      Get

        #If _MyType = "WindowsForms" Then
          If Not addedHandler Then
            SyncLock addedHandlerLockObject
              If Not addedHandler Then
                AddHandler My.Application.Shutdown, AddressOf AutoSaveSettings
                addedHandler = True
              End If
            End SyncLock
          End If
        #End If
        Return defaultInstance
      End Get
    End Property
  End Class
End Namespace

Namespace My

  <Global.Microsoft.VisualBasic.HideModuleNameAttribute(), _
  Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), _
  Global.System.Runtime.CompilerServices.CompilerGeneratedAttribute()> _
  Friend Module MySettingsProperty

    <Global.System.ComponentModel.Design.HelpKeywordAttribute("My.Settings")> _
    Friend ReadOnly Property Settings() As Global.VBHelloWorld.My.MySettings
      Get
        Return Global.VBHelloWorld.My.MySettings.Default
      End Get
    End Property
  End Module
End Namespace
```

Settings.settings(*(root directory name)*\VBHelloWorld\My Project에 있음):

```
<?xml version='1.0' encoding='utf-8'?>
<SettingsFile xmlns="http://schemas.microsoft.com/VisualStudio/2004/01/settings"
 CurrentProfile="(Default)" UseMySettingsClassName="true">
  <Profiles>
    <Profile Name="(Default)" />
  </Profiles>
  <Settings />
</SettingsFile>
```

에서 빌드 계획 AWS CodeBuild

사용하기 전에 다음 질문에 답 AWS CodeBuild해야 합니다.

1. 소스 코드는 어디에 저장되나요? CodeBuild는 현재 다음 소스 코드 리포지토리 공급자로부터의 빌드를 지원합니다. 소스 코드에는 빌드 사양(buildspec) 파일이 포함되어 있어야 합니다. buildspec은 CodeBuild가 빌드를 실행하는 데 사용하는 YAML 형식의 빌드 명령 및 관련 설정의 모음입니다. 빌드 프로젝트 정의에서 buildspec을 선언할 수 있습니다.

리포지토리 공급자	필수	설명서
CodeCommit	리포지토리 이름. (선택 사항) 소스 코드와 연결된 커밋 ID.	AWS CodeCommit 사용 설명서에서 다음 주제를 참조하십시오. CodeCommit 리포지토리 생성 CodeCommit에서 커밋 생성
Amazon S3	입력 버킷 이름. 소스 코드가 포함된 빌드 입력 ZIP 파일에 해당하는 객체 이름. (선택 사항) 빌드 입력 ZIP 파일에 연결된 버전 ID.	Amazon S3 시작 안내서에서 다음 주제를 참조하세요. 버킷 생성 버킷에 객체 추가
GitHub	리포지토리 이름. (선택 사항) 소스 코드와 연결된 커밋 ID.	GitHub Help 웹 사이트에서 다음 주제를 참조하십시오. 리포지토리 생성

리포지토리 공급자	필수	설명서
Bitbucket	리포지토리 이름. (선택 사항) 소스 코드와 연결된 커밋 ID.	Bitbucket Cloud 설명서 웹 사이트에서 다음 주제를 참조하십시오. 리포지토리 생성

2. 어떤 빌드 명령을 실행해야 하며 어떤 순서로 실행해야 합니까? 기본적으로 CodeBuild는 사용자가 지정한 공급자로부터 빌드 입력을 다운로드하고 사용자가 지정한 버킷에 빌드 출력을 업로드합니다. 빌드 사양을 사용하면 다운로드된 빌드 입력을 원하는 빌드 출력으로 전환하는 방법을 지시할 수 있습니다. 자세한 내용은 buildspec 참조 단원을 참조하십시오.
3. 빌드를 실행하는 데 어떤 런타임 및 도구가 필요합니까? 예를 들어 Java, Ruby, Python 또는 Node.js 중 어떤 용도로 빌드하고 있습니까? 빌드에 Maven이나 Ant 또는 Java, Ruby, Python용 컴파일러가 필요합니까? 빌드에 Git AWS CLI, 또는 기타 도구가 필요합니까?

   CodeBuild는 도커 이미지를 사용하는 빌드 환경에서 빌드를 실행합니다. 이러한 도커 이미지는 CodeBuild가 지원하는 리포지토리 유형에 저장되어 있어야 합니다. 여기에는 CodeBuild 도커 이미지 리포지토리, Docker Hub 및 Amazon Elastic Container Registry(Amazon ECR)가 포함됩니다. CodeBuild 도커 이미지 리포지토리에 대한 자세한 내용은 CodeBuild가 제공하는 도커 이미지 섹션을 참조하세요.
4. CodeBuild에서 자동으로 제공하지 않는 AWS 리소스가 필요합니까? 그렇다면 해당 리소스에는 어떤 보안 정책이 필요합니까? 예를 들어, CodeBuild가 해당 리소스와 함께 작동하는 것을 허용하도록 CodeBuild 서비스 역할을 수정해야 할 수 있습니다.
5. CodeBuild를 사용하여 VPC에서 작업을 수행하려고 하나요? 그렇다면 VPC 구성에 대한 VPC ID, 서브넷 ID 및 보안 그룹 ID가 필요합니다. 자세한 내용은 Amazon Virtual Private Cloud AWS CodeBuild 와 함께 사용 단원을 참조하십시오.

위의 질문에 답을 했다면 빌드를 성공적으로 실행하는 데 필요한 설정 및 리소스가 확인되었을 것입니다. 빌드를 실행하려면 다음을 수행하면 됩니다.

- AWS CodeBuild 콘솔 AWS CLI, 또는 AWS SDKs 사용합니다. 자세한 내용은 빌드를 수동으로 실행 단원을 참조하십시오.

- 에서 파이프라인을 생성하거나 식별한 AWS CodePipeline다음 CodeBuild에 코드를 자동으로 테스트하거나 빌드를 실행하거나 둘 다 실행하도록 지시하는 빌드 또는 테스트 작업을 추가합니다. 자세한 내용은 CodePipeline에서 CodeBuild 사용 단원을 참조하십시오.

CodeBuild의 빌드 사양 참조

이 주제에서는 빌드 사양(buildspec) 파일에 대한 중요한 참조 정보를 제공합니다. buildspec은 CodeBuild가 빌드를 실행하는 데 사용하는 YAML 형식의 빌드 명령 및 관련 설정의 모음입니다. 소스 코드의 일부로 buildspec을 포함하거나, 빌드 프로젝트를 생성할 때 buildspec을 정의할 수 있습니다. 빌드 사양의 작동 방식에 대한 자세한 정보는 CodeBuild 작동 방식 단원을 참조하십시오.

주제

- buildspec 파일 이름 및 스토리지 위치
- buildspec 구문
- buildspec 예제
- buildspec 버전
- 배치 빌드 buildspec 참조

buildspec 파일 이름 및 스토리지 위치

buildspec을 소스 코드의 일부로 포함하는 경우, 기본적으로 buildspec 파일에 `buildspec.yml`이라는 이름을 지정해야 하며 이 파일을 소스 디렉터리의 루트에 배치해야 합니다.

기본 buildspec 파일 이름과 위치를 재정의할 수 있습니다. 예를 들어, 다음을 수행할 수 있습니다.

- 동일한 리포지토리의 다른 빌드에 대해 `buildspec_debug.yml` 및 `buildspec_release.yml`과 같은 다른 buildspec 파일을 사용합니다.
- buildspec 파일을 `config/buildspec.yml`과 같은 소스 디렉터리의 루트가 아닌 다른 위치 또는 S3 버킷에 저장합니다. S3 버킷은 빌드 프로젝트와 동일한 AWS 리전에 있어야 합니다. ARN을 사용하여 buildspec 파일을 지정합니다(예: `arn:aws:s3:::<my-codebuild-sample2>/buildspec.yml`).

buildspec 파일의 이름과 상관없이, 한 빌드 프로젝트에 대해 하나의 buildspec만 지정할 수 있습니다.

기본 buildspec 파일 이름이나 위치 또는 둘 다를 재정의하려면 다음 중 하나를 수행합니다.

- 또는 `update-project` 명령을 실행 AWS CLI `create-project`하여 기본 제공 환경 변수의 `buildspec` 값을 기준으로 대체 buildspec 파일의 경로로 값을 설정합니다`CODEBUILD_SRC_DIR`. SDK에서 `create project` 작업과 동등한 작업을 수행할 수도 있습니다. AWS SDKs 자세한 내용은 빌드 프로젝트 생성 또는 빌드 프로젝트 설정 변경을 참조하세요.

- 명령을 실행 AWS CLI `start-build`하여 기본 제공 환경 변수의 `buildspecOverride` 값을 기준으로 대체 buildspec 파일의 경로로 값을 설정합니다`CODEBUILD_SRC_DIR`. SDK에서 `start build` 작업과 동등한 작업을 수행할 수도 있습니다. AWS SDKs 자세한 내용은 빌드를 수동으로 실행 단원을 참조하십시오.
- AWS CloudFormation 템플릿에서 유형의 리소스`Source`에 있는의 `BuildSpec` 속성을 기본 제공 환경 변수의 값을 기준으로 대체 buildspec 파일의 `AWS::CodeBuild::Project` 경로로 설정합니다`CODEBUILD_SRC_DIR`. 자세한 내용은AWS CloudFormation 사용 설명서의 AWS CodeBuild 프로젝트 소스에서 BuildSpec 속성을 참조하세요.

buildspec 구문

buildspec 파일은 YAML 형식으로 표시해야 합니다.

명령에 YAML에서 지원하지 않는 문자 또는 문자열이 포함된 경우 명령을 따옴표("")로 묶어야 합니다. YAML에서는 콜론(:) 뒤에 공백이 올 수 없으므로 다음 명령을 따옴표로 묶습니다. 명령의 따옴표는 이스케이프(\")됩니다.

```
"export PACKAGE_NAME=$(cat package.json | grep name | head -1 | awk -F: '{ print $2 }'
 | sed 's/[\",]//g')"
```

buildspec 구문은 다음과 같습니다.

```
version: 0.2

run-as: Linux-user-name

env:
  shell: shell-tag
  variables:
    key: "value"
    key: "value"
  parameter-store:
    key: "value"
    key: "value"
  exported-variables:
    - variable
    - variable
  secrets-manager:
    key: secret-id:json-key:version-stage:version-id
  git-credential-helper: no | yes
```

```
proxy:
  upload-artifacts: no | yes
  logs: no | yes

batch:
  fast-fail: false | true
  # build-list:
  # build-matrix:
  # build-graph:
  # build-fanout:

phases:
  install:
    run-as: Linux-user-name
    on-failure: ABORT | CONTINUE | RETRY | RETRY-count | RETRY-regex |
 RETRY-count-regex
    runtime-versions:
      runtime: version
      runtime: version
    commands:
      - command
      - command
    finally:
      - command
      - command

  pre_build:
    run-as: Linux-user-name
    on-failure: ABORT | CONTINUE | RETRY | RETRY-count | RETRY-regex |
 RETRY-count-regex
    commands:
      - command
      - command
    finally:
      - command
      - command

  build:
    run-as: Linux-user-name
    on-failure: ABORT | CONTINUE | RETRY | RETRY-count | RETRY-regex |
 RETRY-count-regex
    commands:
      - command
```

```
      - command
    finally:
      - command
      - command

  post_build:
    run-as: Linux-user-name
    on-failure: ABORT | CONTINUE | RETRY | RETRY-count | RETRY-regex |
 RETRY-count-regex
    commands:
      - command
      - command
    finally:
      - command
      - command

reports:
  report-group-name-or-arn:
    files:
      - location
      - location
    base-directory: location
    discard-paths: no | yes
    file-format: report-format
artifacts:
  files:
    - location
    - location
  name: artifact-name
  discard-paths: no | yes
  base-directory: location
  exclude-paths: excluded paths
  enable-symlinks: no | yes
  s3-prefix: prefix
  secondary-artifacts:
    artifactIdentifier:
      files:
        - location
        - location
      name: secondary-artifact-name
      discard-paths: no | yes
      base-directory: location
    artifactIdentifier:
      files:
```

```
        - location
        - location
      discard-paths: no | yes
      base-directory: location
cache:
  key: key
  fallback-keys:
    - fallback-key
    - fallback-key
  action: restore | save
  paths:
    - path
    - path
```

buildspec에는 다음이 포함됩니다.

버전

필수 매핑. buildspec 버전을 나타냅니다. `0.2`을 사용할 것을 권장합니다.

> **Note**
>
> 버전 0.1도 계속 지원되지만 가능하면 버전 0.2를 사용할 것을 권장합니다. 자세한 내용은 buildspec 버전 단원을 참조하십시오.

run-as

선택적 시퀀스. Linux 사용자만 사용할 수 있습니다. 이 buildspec 파일의 명령을 실행하는 Linux 사용자를 지정합니다. `run-as`는 지정한 사용자에게 읽기 및 실행 권한을 부여합니다. buildspec 파일 처음에 `run-as`를 지정할 경우 모든 명령에 전역적으로 적용됩니다. 모든 buildspec 파일 명령에 대한 사용자를 지정하지 않으려는 경우 `phases` 블록 중 하나에 `run-as`를 사용하여 단계에 명령에 대한 사용자를 지정할 수 있습니다. `run-as`를 지정하지 않으면 모든 명령이 루트 사용자로 실행됩니다.

env

선택적 시퀀스. 하나 이상의 사용자 지정 환경 변수에 대한 정보를 나타냅니다.

> **Note**
>
> 중요한 정보를 보호하기 위해 CodeBuild 로그에 다음 항목이 숨겨져 있습니다.

- AWS 액세스 키 IDs. 자세한 내용은AWS Identity and Access Management 사용 설명서에서 IAM 사용자의 액세스 키 관리를 참조하세요.
- 파라미터 스토어를 사용하여 지정된 문자열입니다. 자세한 내용은 Amazon EC2 Systems Manager 사용 설명서의 Systems Manager Parameter Store 및 Systems Manager Parameter Store 콘솔 연습을 참조하세요.
- 를 사용하여 지정된 문자열입니다 AWS Secrets Manager. 자세한 내용은 키 관리 단원을 참조하십시오.

> **Note**
>
> 마스킹은 Amazon EC2 Systems Manager 파라미터 스토어 또는에 저장된 정확한 값과 일치합니다 AWS Secrets Manager. 로그에 표시되기 전에 빌드 명령이 보안 암호를 변환하는 경우 결과는 마스킹되지 않은 다른 문자열입니다. 예를 들어 Base64로 인코딩하거나, 되돌리거나, 사례를 변경하거나, 다른 텍스트와 연결하면 빌드 로그에 표시될 수 있는 다른 값이 생성됩니다. 값을 마스킹된 상태로 유지하려면 빌드에서 사용하는 형식으로 저장합니다. 예를 들어, 빌드에 Base64-encoded 값이 필요한 경우 Base64-encoded 문자열을 AWS Secrets Manager 또는 Amazon EC2 Systems Manager Parameter Store에 직접 저장합니다. 이렇게 하면 로깅된 값이 저장된 값과 일치합니다.

env/shell

선택적 시퀀스. Linux 또는 Windows 운영 체제에서 지원되는 쉘을 지정합니다.

Linux 운영 체제의 경우 지원되는 쉘 태그는 다음과 같습니다.

- `bash`
- `/bin/sh`

Windows 운영 체제의 경우 지원되는 쉘 태그는 다음과 같습니다.

- `powershell.exe`
- `cmd.exe`

env/variables

env가 지정된 경우 및 사용자 지정 환경 변수를 일반 텍스트로 정의하려고 할 때 필요합니다. *key*/*value* 스칼라 매핑을 포함하며, 각 매핑은 일반 텍스트의 단일 사용자 지정 환경 변수를 나타냅니다. *key*는 사용자 지정 환경 변수의 이름이고, *value*는 이 변수의 값입니다.

> **Important**
>
> 중요한 값을 환경 변수에 저장하지 마세요. 환경 변수는 CodeBuild 콘솔 및 AWS CLI와 같은 도구를 사용하여 일반 텍스트로 표시할 수 있습니다. 이 단원의 뒷부분에서 설명하는 것처럼 중요한 값의 경우 `parameter-store` 또는 `secrets-manager` 매핑을 사용하는 것이 좋습니다.
>
> 사용자가 설정한 환경 변수는 기존 환경 변수를 대체합니다. 예를 들어 도커 이미지에 값이 `my_value`인 `MY_VAR`이라는 환경 변수가 이미 포함되어 있는데, 사용자가 `MY_VAR` 환경 변수의 값을 `other_value`로 설정하면, `my_value`가 `other_value`로 바뀝니다. 마찬가지로, 도커 이미지에 값이 `/usr/local/sbin:/usr/local/bin`인 `PATH`라는 환경 변수가 이미 포함되어 있는데, 사용자가 `PATH` 환경 변수의 값을 `$PATH:/usr/share/ant/bin`으로 설정하면, `/usr/local/sbin:/usr/local/bin`이 `$PATH:/usr/share/ant/bin` 리터럴 값으로 바뀝니다.
>
> `CODEBUILD_`로 시작하는 이름으로 환경 변수를 설정하지 마십시오. 이 접두사는 내부 전용으로 예약되어 있습니다.
>
> 여러 위치에서 동일한 이름의 환경 변수가 정의되는 경우, 다음과 같이 값이 결정됩니다.
>
> - 시작 빌드 작업 호출의 값이 가장 높은 우선 순위를 갖습니다. 빌드를 생성할 때 환경 변수를 추가 또는 재정의할 수 있습니다. 자세한 내용은 수동으로 AWS CodeBuild 빌드 실행 단원을 참조하십시오.
> - 빌드 프로젝트 정의의 값이 다음 우선 순위를 갖습니다. 프로젝트를 생성 또는 편집할 때 프로젝트 수준에서 환경 변수를 추가할 수 있습니다. 자세한 내용은 에서 빌드 프로젝트 생성 AWS CodeBuild 및 에서 빌드 프로젝트 설정 변경 AWS CodeBuild 섹션을 참조하세요.
> - buildspec 선언의 값이 가장 낮은 우선 순위를 갖습니다.

env/parameter-store

env가 지정되었으며 Amazon EC2 Systems Manager Parameter Store에 저장된 사용자 지정 환경 변수를 검색하려고 할 때 필요합니다. *key*/*value* 스칼라 매핑을 포함하며, 각 매핑은 Amazon EC2 Systems Manager Parameter Store에 저장된 하나의 사용자 지정 환경 변수를 나타냅니

다. *key*는 이 사용자 지정 환경 변수를 참조하기 위해 나중에 빌드 명령에서 사용할 이름이고, *value*는 Amazon EC2 Systems Manager Parameter Store에 저장되는 사용자 지정 환경 변수의 이름입니다. 중요한 값을 저장하려면 Amazon EC2 Systems Manager 사용 설명서의 Systems Manager Parameter Store 및 안내: 문자열 파라미터 생성 및 테스트(콘솔)를 참조하세요.

> **Important**
>
> CodeBuild가 Amazon EC2 Systems Manager Parameter Store에 저장된 사용자 지정 환경 변수를 검색하도록 허용하려면 `ssm:GetParameters` 작업을 CodeBuild 서비스 역할에 추가해야 합니다. 자세한 내용은 CodeBuild가 다른 AWS 서비스와 상호 작용하도록 허용 단원을 참조하십시오.
>
> Amazon EC2 Systems Manager Parameter Store에서 검색하는 환경 변수는 기존 환경 변수를 대체합니다. 예를 들어 도커 이미지에 값이 `MY_VAR`인 `my_value`이라는 환경 변수가 이미 포함되어 있는데, 사용자가 `MY_VAR` 환경 변수의 값을 `other_value`로 검색하면, `my_value`가 `other_value`로 바뀝니다. 마찬가지로, 도커 이미지에 값이 `PATH`인 `/usr/local/sbin:/usr/local/bin`라는 환경 변수가 이미 포함되어 있는데, 사용자가 `PATH` 환경 변수의 값을 `$PATH:/usr/share/ant/bin`으로 검색하면, `/usr/local/sbin:/usr/local/bin`이 `$PATH:/usr/share/ant/bin` 리터럴 값으로 바뀝니다.
>
> `CODEBUILD_`로 시작하는 이름으로 환경 변수를 저장하지 마십시오. 이 접두사는 내부 전용으로 예약되어 있습니다.
>
> 여러 위치에서 동일한 이름의 환경 변수가 정의되는 경우, 다음과 같이 값이 결정됩니다.
>
> - 시작 빌드 작업 호출의 값이 가장 높은 우선 순위를 갖습니다. 빌드를 생성할 때 환경 변수를 추가 또는 재정의할 수 있습니다. 자세한 내용은 수동으로 AWS CodeBuild 빌드 실행 단원을 참조하십시오.
> - 빌드 프로젝트 정의의 값이 다음 우선 순위를 갖습니다. 프로젝트를 생성 또는 편집할 때 프로젝트 수준에서 환경 변수를 추가할 수 있습니다. 자세한 내용은 에서 빌드 프로젝트 생성 AWS CodeBuild 및 에서 빌드 프로젝트 설정 변경 AWS CodeBuild 섹션을 참조하세요.
> - buildspec 선언의 값이 가장 낮은 우선 순위를 갖습니다.

env/secrets-manager

에 저장된 사용자 지정 환경 변수를 검색하려는 경우 필요합니다 AWS Secrets Manager. 다음 패턴을 사용하여 Secrets Manager `reference-key`를 지정합니다.

<key>: *<secret-id>*:*<json-key>*:*<version-stage>*:*<version-id>*

<key>

(필수) 로컬 환경 변수 이름입니다. 이 이름을 사용하면 빌드 중에 변수에 액세스할 수 있습니다.

<secret-id>

(필수) 보안 암호에 대한 고유 식별자 역할을 하는 이름 또는 Amazon 리소스 이름(ARN)입니다. AWS 계정에 있는 암호에 액세스하려면 암호 이름을 지정합니다. 다른 AWS 계정의 보안 암호에 액세스하려면 보안 암호 ARN을 지정합니다.

<json-key>

(선택 사항) 해당 값을 검색하려는 Secrets Manager 키-값 페어의 키 이름을 지정합니다. `json-key`를 지정하지 않는 경우 CodeBuild에서 전체 보안 암호 텍스트를 검색합니다.

<version-stage>

(선택 사항) 버전에 연결된 레이블을 스테이징하여 검색하려는 보안 암호 버전을 지정합니다. 스테이징 레이블은 교체 프로세스 도중 다른 버전을 추적하는 데 사용됩니다. `version-stage`를 사용하는 경우 `version-id`를 지정하지 마십시오. 버전 스테이지 또는 버전 ID를 지정하지 않으면 기본값은 `AWSCURRENT`의 버전 스테이지 값으로 버전을 검색하는 것입니다.

<version-id>

(선택 사항) 사용하려는 암호의 버전에 대한 고유 식별자를 지정합니다. `version-id`을 지정할 경우 `version-stage`을 지정하지 마십시오. 버전 스테이지 또는 버전 ID를 지정하지 않으면 기본값은 `AWSCURRENT`의 버전 스테이지 값으로 버전을 검색하는 것입니다.

다음 예제에서 `TestSecret`은 Secrets Manager에 저장되는 키-값 페어의 이름입니다. `TestSecret`의 키는 `MY_SECRET_VAR`입니다. 빌드 중에 `LOCAL_SECRET_VAR` 이름을 사용하여 변수에 액세스합니다.

```
env:
  secrets-manager:
    LOCAL_SECRET_VAR: "TestSecret:MY_SECRET_VAR"
```

자세한 내용은AWS Secrets Manager 사용 설명서에서 AWS Secrets Manager란 무엇입니까? 단원을 참조하세요.

env/exported-variables

선택적 매핑. 내보낼 환경 변수를 나열하는 데 사용됩니다. `exported-variables`에서 별도의 행에 내보낼 각 변수의 이름을 지정합니다. 내보낼 변수는 빌드 중에 컨테이너에서 사용할 수 있어야 합니다. 내보내는 변수는 환경 변수일 수 있습니다.

내보낸 환경 변수는와 함께 현재 빌드 단계에서 파이프라인의 후속 단계로 환경 변수를 AWS CodePipeline 내보내는 데 사용됩니다. 자세한 내용을 알아보려면AWS CodePipeline 사용 설명서의 변수 작업을 참조하세요.

빌드하는 동안 변수의 값은 `install` 단계부터 사용할 수 있습니다. 이 값은 `install` 단계 시작과 `post_build` 단계 끝 사이에서 업데이트할 수 있습니다. `post_build` 단계가 끝나면 내보낸 변수의 값을 변경할 수 없습니다.

> **Note**
>
> 다음은 내보낼 수 없습니다.
>
> - 빌드 프로젝트에서 지정된 Amazon EC2 Systems Manager Parameter Store 보안 암호
> - 빌드 프로젝트에서 지정된 Secrets Manager 보안 암호
> - `AWS_`로 시작하는 환경 변수

env/git-credential-helper

선택적 매핑. CodeBuild가 Git 보안 인증 도우미를 사용하여 Git 보안 인증을 제공하는지를 나타내는 데 사용됩니다. Git 보안 인증 도우미가 사용되는 경우 `yes`입니다. 그렇지 않은 경우 `no`이거나 값이 지정되지 않은 것입니다. 자세한 내용은 Git 웹사이트의 gitcredentials를 참조하십시오.

> **Note**
>
> 퍼블릭 Git 리포지토리에 대해 webhook에서 트리거한 빌드의 경우에는 `git-credential-helper`가 지원되지 않습니다.

proxy

선택적 시퀀스. 명시적 프록시 서버에서 빌드를 실행할 경우 설정을 나타내는 데 사용됩니다. 자세한 내용은 명시적 프록시 서버에서 CodeBuild 실행 단원을 참조하십시오.

proxy/upload-artifacts

선택적 매핑. 명시적 프록시 서버에서 빌드가 아티팩트를 업로드하도록 하려면 `yes`로 설정합니다. 기본값은 `no`입니다.

proxy/logs

선택적 매핑. 명시적 프록시 서버에서 빌드에 대해 yes로 설정하여 CloudWatch 로그를 생성합니다. 기본값은 no입니다.

단계

필수 시퀀스. 빌드의 각 단계 동안 CodeBuild가 실행하는 명령을 나타냅니다.

> **Note**
>
> buildspec 버전 0.1에서 CodeBuild는 빌드 환경에 있는 기본 쉘의 개별 인스턴스에서 각 명령을 실행합니다. 즉, 각 명령이 다른 모든 명령과 독립적으로 실행됩니다. 따라서 기본적으로 이전 명령의 상태에 따라 실행되는 단일 명령을 실행할 수 없습니다(예: 디렉터리 변경 또는 환경 변수 설정). 이 제한 사항을 해결하려면 이 문제를 해결하는 버전 0.2를 사용하는 것이 좋습니다. buildspec 버전 0.1을 사용해야 하는 경우 빌드 환경의 쉘 및 명령 단원의 접근 방식을 따르는 것이 좋습니다.

phases/*/run-as

선택적 시퀀스. 해당 명령을 실행하는 Linux 사용자를 지정하려면 빌드 단계에 사용합니다. buildspec 파일의 상단에 모든 명령에 대해 전역적으로 `run-as`도 지정할 경우 단계 수준 사용자가 우선 적용됩니다. 예를 들어 전역적으로 User-1을 `run-as` 지정하고 `install` 단계의 경우 `run-as` 문만 User-2를 지정하는 경우, buildspec 파일의 모든 명령은 User-2로 실행되는 단계의 명령을 제외하고 User-1로 실행됩니다. `install` User-2

phases/*/on-failure

선택적 시퀀스. 단계 중에 오류가 발생할 경우 수행할 작업을 지정합니다. 다음 값 중 하나일 수 있습니다:

- `ABORT` - 빌드를 중단합니다.
- `CONTINUE` - 다음 단계를 계속 진행합니다.
- `RETRY` - 정규식 `.*`과 일치하는 오류 메시지와 함께 빌드를 최대 3회 재시도합니다.
- `RETRY-`*`count`* - 정규식 `.*`과 일치하는 오류 메시지와 함께 *#*에 표시된 대로 지정된 횟수만큼 빌드를 재시도합니다. *#*는 0에서 100 사이여야 합니다. 예를 들어, 유효한 값에는 `RETRY-4` 및 `RETRY-8`이 포함됩니다.

- RETRY-*regex* - 빌드를 최대 3회 재시도하고 #### 사용하여 지정된 오류 메시지와 일치하는 정규식을 포함합니다. 예를 들어, 유효한 값에는 `Retry-.*Error: Unable to connect to database.*` 및 `RETRY-invalid+`이 포함됩니다.
- RETRY-*count*-*regex* - #에 표시된 대로 지정된 횟수만큼 빌드를 재시도합니다. #는 0에서 100 사이여야 합니다. ###을 사용하여 오류 메시지와 일치하는 정규식을 포함할 수도 있습니다. 예를 들어, 유효한 값에는 `Retry-3-.*connection timed out.*` 및 `RETRY-8-invalid+`이 포함됩니다.

이 속성을 지정하지 않으면 오류 프로세스는 빌드 단계 진행에 표시된 전환 단계를 따릅니다.

> **Important**
>
> Lambda 컴퓨팅 또는 예약 용량을 사용할 때는 `on-failure` 속성이 지원되지 않습니다. 이 속성은 CodeBuild에서 제공하는 EC2 컴퓨팅 이미지에서만 작동합니다.

phases/*/finally

선택적 블록. `finally` 블록에 지정된 명령은 `commands` 블록의 명령 후에 실행됩니다. `finally` 블록의 명령은 `commands` 블록의 명령이 실패할 경우에도 실행됩니다. 예를 들어 명령 3개가 포함된 `commands` 블록에서 첫 번째 명령이 실패할 경우, CodeBuild는 나머지 두 명령을 건너뛰고 `finally` 블록의 명령을 실행합니다. `commands` 및 `finally` 블록의 모든 명령이 성공적으로 실행되면 단계가 성공합니다. 어느 단계의 명령이 하나라도 실패하면 그 단계는 실패합니다.

허용되는 빌드 단계 이름은 다음과 같습니다.

phases/install

선택적 시퀀스. 설치 중에 CodeBuild가 실행하는 명령을 나타냅니다(있는 경우). 빌드 환경에서 패키지를 설치하는 경우에만 `install` 단계를 사용하는 것이 좋습니다. 예를 들어, Mocha나 RSpec 같은 코드 테스팅 프레임워크를 설치하기 위해 이 단계를 사용할 수 있습니다.

phases/install/runtime-versions

선택적 시퀀스. 런타임 버전은 Ubuntu 표준 이미지 5.0 이상 및 Amazon Linux 2 표준 이미지 4.0 이상에서 지원됩니다. 지정된 경우 적어도 하나의 실행 시간이 이 섹션에 포함되어야 합니다. 특정 버전을 사용하여 런타임을 지정하거나, 주 버전 다음에 `.x`를 사용하여 CodeBuild가 주 버전을 해당 최신 부 버전과 함께 사용하도록 지정하거나, 최신 주 버전 및 부 버전을 사용하도록 `latest`합니다(예: `ruby: 3.2`, `nodejs: 18.x` 또는 `java: latest`). 숫자나 환경 변수

를 사용하여 실행 시간을 지정할 수 있습니다. 예를 들어 Amazon Linux 2 표준 이미지 4.0을 사용하는 경우 다음은 Java 버전 17, Python 버전 3의 최신 부 버전, Ruby 환경 변수에 포함된 버전이 설치되도록 지정합니다. 자세한 내용은 CodeBuild가 제공하는 도커 이미지 단원을 참조하십시오.

```
phases:
  install:
    runtime-versions:
      java: corretto8
      python: 3.x
      ruby: "$MY_RUBY_VAR"
```

빌드 사양 파일의 `runtime-versions` 섹션에서 하나 이상의 런타임을 지정할 수 있습니다. 런타임이 다른 런타임에 종속되는 경우 빌드 사양 파일에서 종속 런타임을 지정할 수도 있습니다. buildspec 파일에 런타임을 지정하지 않은 경우 CodeBuild에서는 사용하는 이미지에서 사용할 수 있는 기본 런타임을 선택합니다. 하나 이상의 런타임을 지정하는 경우 CodeBuild에서는 해당 런타임만 사용합니다. 종속 런타임을 지정하지 않은 경우 CodeBuild에서 종속 런타임을 선택하려고 시도합니다.

> **Note**
>
> 런타임 버전을 지정하지 않으면 CodeBuild는 기본 버전을 사용합니다. 이전 기본 버전이 수명 종료(EOL)에 도달하면 기본 버전이 변경될 수 있습니다. 빌드 환경이 예기치 않게 변경되지 않도록 buildspec 파일에 런타임 버전을 지정하는 것이 좋습니다.

지정된 두 실행 시간이 충돌하면 빌드가 실패합니다. 예를 들어 `android: 29`와 `java: openjdk11`이 충돌하므로 둘 다 지정하면 빌드가 실패합니다.

사용 가능한 런타임에 대한 자세한 내용은 사용 가능한 런타임 섹션을 참조하세요.

> **Note**
>
> `runtime-versions` 섹션을 지정하고 Ubuntu 표준 이미지 2.0 이상 또는 Amazon Linux 2(AL2) 표준 이미지 1.0 이상 외의 이미지를 사용하는 경우, 빌드에서 "`Skipping install of runtimes. Runtime version selection is not supported by this build image`"라는 경고가 발생합니다.

phases/install/commands

선택적 시퀀스. 스칼라 시퀀스를 포함합니다. 여기서 각 스칼라는 CodeBuild가 설치 중에 실행하는 단일 명령을 나타냅니다. 빌드 단계마다, CodeBuild는 처음부터 끝까지 한 번에 하나씩 나열된 순서로 각 명령을 실행합니다.

phases/pre_build

선택적 시퀀스. 빌드 전에 CodeBuild가 실행하는 명령을 나타냅니다(있는 경우). 예를 들어, Amazon ECR에 로그인하기 위해 이 단계를 사용할 수 있습니다. 또는 npm 종속성을 설치할 수도 있습니다.

phases/pre_build/commands

`pre_build`를 지정한 경우 필수 시퀀스입니다. 스칼라 시퀀스를 포함합니다. 여기서 각 스칼라는 CodeBuild가 빌드 전에 실행하는 단일 명령을 나타냅니다. 빌드 단계마다, CodeBuild는 처음부터 끝까지 한 번에 하나씩 나열된 순서로 각 명령을 실행합니다.

phases/build

선택적 시퀀스. 빌드 중에 CodeBuild가 실행하는 명령을 나타냅니다(있는 경우). 예를 들어, Mocha, RSpec 또는 sbt를 실행하기 위해 이 단계를 사용할 수 있습니다.

phases/build/commands

`build`를 지정한 경우 필수입니다. 스칼라 시퀀스를 포함합니다. 여기서 각 스칼라는 CodeBuild가 빌드 중에 실행하는 단일 명령을 나타냅니다. 빌드 단계마다, CodeBuild는 처음부터 끝까지 한 번에 하나씩 나열된 순서로 각 명령을 실행합니다.

phases/post_build

선택적 시퀀스. 빌드 후에 CodeBuild가 실행하는 명령을 나타냅니다(있는 경우). 예를 들어, Maven을 사용하여 빌드 아티팩트를 JAR 또는 WAR 파일에 패키지할 수 있으며, Amazon ECR에 도커 이미지를 푸시할 수도 있습니다. 그런 다음, Amazon SNS를 통해 빌드 알림을 전송할 수도 있습니다.

phases/post_build/commands

`post_build`를 지정한 경우 필수입니다. 스칼라 시퀀스를 포함합니다. 여기서 각 스칼라는 CodeBuild가 빌드 후에 실행하는 단일 명령을 나타냅니다. 빌드 단계마다, CodeBuild는 처음부터 끝까지 한 번에 하나씩 나열된 순서로 각 명령을 실행합니다.

보고서

report-group-name-or-arn

선택적 시퀀스. 보고서를 전송할 보고서 그룹을 지정합니다. 프로젝트에는 최대 5개의 보고서 그룹이 포함될 수 있습니다. 기존 보고서 그룹의 ARN 또는 새 보고서 그룹의 이름을 지정합니다. 이름을 지정하는 경우 CodeBuild는 프로젝트 이름과 `<project-name>-<report-group-name>` 형식으로 지정한 이름을 사용하여 보고서 그룹을 생성합니다. 보고서 그룹 이름은 `$REPORT_GROUP_NAME`과 같은 빌드 사양의 환경 변수를 사용하여 설정할 수도 있습니다. 자세한 내용은 보고서 그룹 이름 지정 단원을 참조하십시오.

reports/<report-group>/files

필수 시퀀스. 보고서에 의해 생성된 테스트 결과의 원시 데이터가 포함된 위치를 나타냅니다. 스칼라 시퀀스를 포함하며, 각 스칼라는 CodeBuild가 원래 빌드 위치와 관련하여 테스트 파일을 찾을 수 있는 별도의 위치 또는 설정된 경우 `base-directory`를 나타냅니다. 위치에는 다음이 포함될 수 있습니다.

- 단일 파일(예: `my-test-report-file.json`).
- 하위 디렉터리의 단일 파일입니다(예: *`my-subdirectory`*`/my-test-report-file.json` 또는 *`my-parent-subdirectory`*`/`*`my-subdirectory`*`/my-test-report-file.json`).
- `'**/*'`는 모든 파일을 재귀적으로 나타냅니다.
- *`my-subdirectory`*`/*`는 *`my-subdirectory`*라는 하위 디렉터리에 있는 모든 파일을 나타냅니다.
- *`my-subdirectory`*`/**/*`는 *`my-subdirectory`*라는 하위 디렉터리에서 시작하는 모든 파일을 재귀적으로 나타냅니다.

reports/<report-group>/file-format

선택적 매핑. 보고서 파일 형식을 나타냅니다. 지정하지 않으면 `JUNITXML`가 사용됩니다. 이 값은 대소문자를 구분하지 않습니다. 가능한 값은 다음과 같습니다.

테스트 보고서

`CUCUMBERJSON`

Cucumber JSON

`JUNITXML`

JUnit XML

`NUNITXML`

NUnit XML

`NUNIT3XML`

NUnit 3 XML

`TESTNGXML`

TestNG XML

`VISUALSTUDIOTRX`

Visual Studio TRX

코드 범위 보고서

`CLOVERXML`

Clover XML

`COBERTURAXML`

Cobertura XML

`JACOCOXML`

JacoCo XML

`SIMPLECOV`

SimpleCov JSON

> **Note**
>
> CodeBuild는 simplecov-json이 아닌 simplecov에서 생성된 JSON 코드 범위 보고서를 수락합니다.

보고서/<report-group>/기본 디렉터리

선택적 매핑. 원시 테스트 파일을 찾을 위치를 결정할 때 CodeBuild가 사용하는 원래 빌드 위치를 기준으로 하나 이상의 최상위 디렉터리를 나타냅니다.

reports/<report-group>/discard-paths

선택 사항. 보고서 파일 디렉터리가 출력에서 평면화되는지 여부를 지정합니다. 이 값이 지정되지 않거나 no를 포함하는 경우 보고서 파일은 디렉터리 구조가 손상되지 않은 상태로 출력됩니다. yes가 포함된 경우 모든 테스트 파일이 동일한 출력 디렉터리에 배치됩니다. 예를 들어, 테스트 결과에 대한 경로가 com/myapp/mytests/TestResult.xml인 경우 yes를 지정하면 이 파일이 /TestResult.xml에 배치됩니다.

artifacts

선택적 시퀀스. CodeBuild가 빌드 출력을 찾을 수 있는 위치 및 CodeBuild가 S3 출력 버킷에 업로드하기 위해 빌드 출력을 준비하는 방식에 대한 정보를 나타냅니다. 도커 이미지를 빌드하여 Amazon ECR에 푸시하는 경우 또는 소스 코드에 단위 테스트만 실행하고 빌드는 하지 않는 경우 등에는 이 시퀀스가 필요하지 않습니다.

> **Note**
>
> Amazon S3 메타데이터에는 Amazon S3에 아티팩트를 게시하는 CodeBuild 빌드의 buildArn이 포함된 x-amz-meta-codebuild-buildarn라는 CodeBuild 헤더가 있습니다. 알림에 대한 소스 추적을 허용하고 아티팩트가 생성된 빌드를 참조할 수 있도록 하기 위해 buildArn이 추가되었습니다.

artifacts/files

필수 시퀀스. 빌드 환경의 빌드 출력 결과물을 포함하는 위치를 나타냅니다. 스칼라 시퀀스를 포함하며, 각 스칼라는 CodeBuild가 빌드 출력 결과물을 찾을 수 있는 개별 위치를 원래 빌드 위치 또는 기본 디렉터리(설정한 경우)를 기준으로 나타냅니다. 위치에는 다음이 포함될 수 있습니다.

- 단일 파일(예: my-file.jar).
- 하위 디렉터리의 단일 파일입니다(예: *my-subdirectory*/my-file.jar 또는 *my-parent-subdirectory*/*my-subdirectory*/my-file.jar).
- '**/*'는 모든 파일을 재귀적으로 나타냅니다.
- *my-subdirectory*/*는 *my-subdirectory*라는 하위 디렉터리에 있는 모든 파일을 나타냅니다.
- *my-subdirectory*/**/*는 *my-subdirectory*라는 하위 디렉터리에서 시작하는 모든 파일을 재귀적으로 나타냅니다.

빌드 출력 아티팩트 위치를 지정하면 CodeBuild가 빌드 환경의 원래 빌드 위치를 찾을 수 있습니다. 빌드 아티팩트 출력 위치 앞에 원래 빌드 위치 경로를 추가하거나 `./` 같은 것을 지정하지 않아도 됩니다. 이 위치에 대한 경로를 알고 싶으면 빌드 중에 `echo $CODEBUILD_SRC_DIR`과 같은 명령을 실행하면 됩니다. 각 빌드 환경의 위치는 약간씩 다를 수 있습니다.

artifacts/name

선택적 이름. 빌드 아티팩트의 이름을 지정합니다. 이 이름은 다음 중 하나에 해당될 때 사용됩니다.

- CodeBuild API를 사용하여 빌드를 생성하면 프로젝트가 업데이트되거나 생성되거나 빌드가 시작될 때 `ProjectArtifacts` 객체에 `overrideArtifactName` 플래그가 설정됩니다.
- CodeBuild 콘솔을 사용하여 빌드를 생성하면 buildspec 파일에 이름이 지정되고, 프로젝트를 만들거나 업데이트할 때 의미 체계 버전 관리 활성화를 선택합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 단원을 참조하십시오.

빌드할 때 계산되는 buildspec 파일에서 이름을 지정할 수 있습니다. buildspec 파일에 지정된 이름은 Shell 명령 언어를 사용합니다. 예를 들어 아티팩트 이름이 항상 고유하도록 날짜와 시간을 아티팩트 이름에 추가할 수 있습니다. 고유한 결과물 이름을 사용하면 결과물을 덮어쓰지 않을 수 있습니다. 자세한 정보는 Shell 명령 언어를 참조하십시오.

- 다음은 아티팩트가 생성된 날짜가 추가된 아티팩트 이름의 예입니다

```
version: 0.2
phases:
  build:
    commands:
      - rspec HelloWorld_spec.rb
artifacts:
  files:
    - '**/*'
  name: myname-$(date +%Y-%m-%d)
```

- 다음은 CodeBuild 환경 변수를 사용하는 아티팩트 이름의 예입니다. 자세한 내용은 빌드 환경의 환경 변수 단원을 참조하십시오.

```
version: 0.2
phases:
  build:
    commands:
      - rspec HelloWorld_spec.rb
artifacts:
  files:
    - '**/*'
```

```
  name: myname-$AWS_REGION
```

- 다음은 CodeBuild 환경 변수를 사용하여 아티팩트 생성 날짜가 추가된 아티팩트 이름의 예입니다.

```
version: 0.2
phases:
  build:
    commands:
      - rspec HelloWorld_spec.rb
artifacts:
  files:
    - '**/*'
  name: $AWS_REGION-$(date +%Y-%m-%d)
```

이름에 경로 정보를 추가하여 이름의 경로를 기준으로 명명된 아티팩트가 디렉터리에 배치되도록 할 수 있습니다. 이 예제에서는 빌드 아티팩트가 출력의 builds/<build number>/my-artifacts 아래에 배치됩니다.

```
version: 0.2
phases:
  build:
    commands:
      - rspec HelloWorld_spec.rb
artifacts:
  files:
    - '**/*'
  name: builds/$CODEBUILD_BUILD_NUMBER/my-artifacts
```

artifacts/discard-paths

선택 사항. 빌드 아티팩트 디렉터리가 출력에서 평면화되는지 여부를 지정합니다. 이 값이 지정되지 않거나 no를 포함하는 경우 빌드 아티팩트는 디렉터리 구조가 손상되지 않은 상태로 출력됩니다. yes가 포함된 경우 모든 빌드 아티팩트가 동일한 출력 디렉터리에 배치됩니다. 예를 들어, 빌드 출력 아티팩트의 파일 경로가 com/mycompany/app/HelloWorld.java인 경우 yes를 지정하면 이 파일이 /HelloWorld.java에 배치됩니다.

artifacts/base-directory

선택적 매핑. CodeBuild가 빌드 출력 아티팩트에 포함할 파일 및 하위 디렉터리를 결정하는 데 사용하는 원래 빌드 위치에 상대적인 하나 이상의 최상위 디렉터리를 나타냅니다. 유효한 값으로는 다음이 포함됩니다.

- 단일 최상위 디렉터리입니다(예: `my-directory`).
- `'my-directory*'`는 이름이 `my-directory`로 시작하는 모든 최상위 디렉터리를 나타냅니다.

빌드 출력 결과물에는 이 최상위 디렉터리가 포함되지 않으며, 파일 및 하위 디렉터리만 포함됩니다.

포함할 파일 및 하위 디렉터리를 보다 더 제한하려면 `files` 및 `discard-paths`를 사용하면 됩니다. 예를 들어 다음 디렉터리구조에서

```
.
### my-build-1
#   ### my-file-1.txt
### my-build-2
    ### my-file-2.txt
    ### my-subdirectory
        ### my-file-3.txt
```

다음 `artifacts` 시퀀스에 대해:

```
artifacts:
  files:
    - '*/my-file-3.txt'
  base-directory: my-build-2
```

다음 하위 디렉터리 및 파일이 빌드 출력 결과물에 포함됩니다.

```
.
### my-subdirectory
    ### my-file-3.txt
```

다음 `artifacts` 시퀀스 동안

```
artifacts:
  files:
    - '**/*'
  base-directory: 'my-build*'
  discard-paths: yes
```

다음 파일이 빌드 출력 결과물에 포함됩니다.

```
.
### my-file-1.txt
### my-file-2.txt
### my-file-3.txt
```

artifacts/exclude-paths

선택적 매핑. CodeBuild가 빌드 아티팩트에서 제외할 base-directory에 상대적인 하나 이상의 경로를 나타냅니다. 별표(*) 문자는 폴더의 경계를 넘지 않고 0개 이상의 이름 구성 요소 문자와 해당합니다. 이중 별표(**)는 모든 디렉터리에서 이름 구성 요소의 0개 이상 문자와 일치합니다.

exclude-paths의 예는 다음과 같습니다.

- 모든 디렉터리에서 파일을 제외하려면: "**/*file-name*/**/*"
- 모든 dot 폴더를 제외하려면: "**/.*/**/*"
- 모든 dot 파일을 제외하려면: "**/.*"

artifacts/enable-symlinks

선택 사항. 출력 유형이 ZIP인 경우 내부 심볼 링크를 ZIP 파일에 보존할지 여부를 지정합니다. 이 파일에 yes가 포함된 경우 소스의 모든 내부 심볼 링크가 아티팩트 ZIP 파일에 보존됩니다.

artifacts/s3-prefix

선택 사항. 아티팩트가 Amazon S3 버킷으로 출력되고 네임스페이스 유형이 BUILD_ID일 때 사용되는 접두사를 지정합니다. 사용될 경우 버킷의 출력 경로는 <s3-prefix>/<build-id>/<name>.zip입니다.

artifacts/secondary-artifacts

선택적 시퀀스. 1개 이상의 아티팩트 정의를 아티팩트 식별자와 아티팩트 정의를 연결하는 매핑으로 나타냅니다. 이 블록의 각 아티팩트 식별자는 프로젝트의 secondaryArtifacts 속성에 정의된 아티팩트와 일치해야 합니다. 각 정의는 위의 artifacts 블록과 동일한 구문을 갖습니다.

> **Note**
>
> 2차 아티팩트만 정의된 경우에도 artifacts/files 시퀀스는 항상 필요합니다.

예를 들어 프로젝트가 다음과 같은 구조라고 가정할 경우,

```
{
```

```
  "name": "sample-project",
  "secondaryArtifacts": [
    {
      "type": "S3",
      "location": "<output-bucket1>",
      "artifactIdentifier": "artifact1",
      "name": "secondary-artifact-name-1"
    },
    {
      "type": "S3",
      "location": "<output-bucket2>",
      "artifactIdentifier": "artifact2",
      "name": "secondary-artifact-name-2"
    }
  ]
}
```

buildpec 파일은 다음과 비슷합니다.

```
version: 0.2

phases:
build:
  commands:
    - echo Building...
artifacts:
  files:
    - '**/*'
  secondary-artifacts:
    artifact1:
      files:
        - directory/file1
      name: secondary-artifact-name-1
    artifact2:
      files:
        - directory/file2
      name: secondary-artifact-name-2
```

cache

선택적 시퀀스. CodeBuild가 S3 캐시 버킷에 캐시를 업로드하기 위해 준비할 수 있는 파일에 대한 정보를 나타냅니다. 프로젝트의 캐시 유형이 `No Cache`인 경우 이 시퀀스는 필수가 아닙니다.

캐시/키

선택적 시퀀스. 캐시를 검색하거나 복원할 때 사용되는 기본 키를 나타냅니다. CodeBuild는 기본 키와 정확히 일치합니다.

다음은 키의 예입니다.

```
key: npm-key-$(codebuild-hash-files package-lock.json) }
```

캐시/폴백 키

선택적 시퀀스. 프라이머리 키를 사용하여 캐시를 찾을 수 없을 때 순차적으로 사용되는 대체 키 목록을 나타냅니다. 최대 5개의 폴백 키가 지원되며 각 폴백 키는 접두사 검색을 사용하여 일치됩니다. 키가 제공되지 않으면 이 시퀀스는 무시됩니다.

다음은 폴백 키의 예입니다.

```
fallback-keys:
    - npm-key-$(codebuild-hash-files package-lock.json) }
    - npm-key-
    - npm-
```

캐시/작업

선택적 시퀀스. 캐시에서 수행할 작업을 지정합니다. 유효한 값으로는 다음이 포함됩니다.

- `restore`는 업데이트를 저장하지 않고 캐시만 복원합니다.
- `save`는 이전 버전을 복원하지 않고 캐시만 저장합니다.

값이 제공되지 않으면 CodeBuild는 기본적으로 복원과 저장을 모두 수행합니다.

cache/paths

필수 시퀀스. 캐시의 위치를 나타냅니다. 스칼라 시퀀스를 포함하며, 각 스칼라는 CodeBuild가 빌드 출력 결과물을 찾을 수 있는 개별 위치를 원래 빌드 위치 또는 기본 디렉터리(설정한 경우)를 기준으로 나타냅니다. 위치에는 다음이 포함될 수 있습니다.

- 단일 파일(예: `my-file.jar`).
- 하위 디렉터리의 단일 파일입니다(예: *`my-subdirectory`*`/my-file.jar` 또는 *`my-parent-subdirectory`*`/`*`my-subdirectory`*`/my-file.jar`).
- `'**/*'`는 모든 파일을 재귀적으로 나타냅니다.

- *my-subdirectory*/*는 *my-subdirectory*라는 하위 디렉터리에 있는 모든 파일을 나타냅니다.
- *my-subdirectory*/**/*는 *my-subdirectory*라는 하위 디렉터리에서 시작하는 모든 파일을 재귀적으로 나타냅니다.

> **Important**
>
> buildspec 선언은 올바른 YAML이어야 하므로 buildspec 선언의 공백 설정이 중요합니다. buildspec 선언의 공백 수가 잘못되면 빌드가 즉시 실패할 수 있습니다. YAML 유효성 검사기를 사용하여 buildspec 선언이 올바른 YAML인지 여부를 테스트할 수 있습니다.
> 빌드 프로젝트를 생성 AWS CLI하거나 업데이트할 때 또는 AWS SDKs를 사용하여 buildspec을 선언하는 경우 buildspec은 필수 공백 및 줄 바꿈 이스케이프 문자와 함께 YAML 형식으로 표현된 단일 문자열이어야 합니다. 다음 섹션에 예가 나와 있습니다.
> buildspec.yml 파일 대신 CodeBuild 또는 AWS CodePipeline 콘솔을 사용하는 경우 `build` 단계에 대해서만 명령을 삽입할 수 있습니다. 앞에 나온 구문을 사용하는 대신, 빌드 단계 중에 실행하려는 모든 명령을 하나의 행에 나열합니다. 명령이 여러 개인 경우 각 명령을 `&&`로 구분합니다(예: `mvn test && mvn package`).
> buildspec.yml 파일 대신 CodeBuild 또는 CodePipeline 콘솔을 사용하여 빌드 환경의 빌드 출력 아티팩트 위치를 지정할 수 있습니다. 앞에 나온 구문을 사용하는 대신, 모든 위치를 하나의 행에 나열합니다. 위치가 여러 개인 경우 각 위치를 쉼표로 구분합니다(예: `buildspec.yml, target/my-app.jar`).

buildspec 예제

다음은 buildspec.yml 파일의 예입니다.

```
version: 0.2

env:
  variables:
    JAVA_HOME: "/usr/lib/jvm/java-8-openjdk-amd64"
  parameter-store:
    LOGIN_PASSWORD: /CodeBuild/dockerLoginPassword

phases:
  install:
    commands:
```

```
      - echo Entered the install phase...
      - apt-get update -y
      - apt-get install -y maven
    finally:
      - echo This always runs even if the update or install command fails
  pre_build:
    commands:
      - echo Entered the pre_build phase...
      - docker login -u User -p $LOGIN_PASSWORD
    finally:
      - echo This always runs even if the login command fails
  build:
    commands:
      - echo Entered the build phase...
      - echo Build started on `date`
      - mvn install
    finally:
      - echo This always runs even if the install command fails
  post_build:
    commands:
      - echo Entered the post_build phase...
      - echo Build completed on `date`

reports:
  arn:aws:codebuild:your-region:your-aws-account-id:report-group/report-group-name-1:
    files:
      - "**/*"
    base-directory: 'target/tests/reports'
    discard-paths: no
  reportGroupCucumberJson:
    files:
      - 'cucumber/target/cucumber-tests.xml'
    discard-paths: yes
    file-format: CUCUMBERJSON # default is JUNITXML
artifacts:
  files:
    - target/messageUtil-1.0.jar
  discard-paths: yes
  secondary-artifacts:
    artifact1:
      files:
        - target/artifact-1.0.jar
      discard-paths: yes
    artifact2:
```

```
      files:
        - target/artifact-2.0.jar
      discard-paths: yes
cache:
  paths:
    - '/root/.m2/**/*'
```

다음은 AWS CLI또는 AWS SDKs.

```
"version: 0.2\n\nenv:\n  variables:\n    JAVA_HOME: \"/usr/lib/jvm/java-8-openjdk-
amd64\\"\n  parameter-store:\n    LOGIN_PASSWORD: /CodeBuild/dockerLoginPassword\n
 phases:\n\n  install:\n    commands:\n      - echo Entered the install phase...\n
 - apt-get update -y\n      - apt-get install -y maven\n    finally:\n      - echo This
 always runs even if the update or install command fails \n  pre_build:\n    commands:
\n      - echo Entered the pre_build phase...\n      - docker login -u User -p
 $LOGIN_PASSWORD\n    finally:\n      - echo This always runs even if the login command
 fails \n  build:\n    commands:\n      - echo Entered the build phase...\n      - echo
 Build started on `date`\n      - mvn install\n    finally:\n      - echo This always
 runs even if the install command fails\n  post_build:\n    commands:\n      - echo
 Entered the post_build phase...\n      - echo Build completed on `date`\n\n reports:
\n  reportGroupJunitXml:\n    files:\n      - \"**/*\"\n    base-directory: 'target/
tests/reports'\n    discard-paths: false\n  reportGroupCucumberJson:\n    files:\n
 - 'cucumber/target/cucumber-tests.xml'\n    file-format: CUCUMBERJSON\n\nartifacts:\n
 files:\n    - target/messageUtil-1.0.jar\n  discard-paths: yes\n  secondary-artifacts:
\n    artifact1:\n      files:\n       - target/messageUtil-1.0.jar\n      discard-
paths: yes\n    artifact2:\n      files:\n       - target/messageUtil-1.0.jar\n
 discard-paths: yes\n cache:\n  paths:\n    - '/root/.m2/**/*'"
```

다음은 CodeBuild 또는 CodePipeline 콘솔과 함께 사용하기 위한 build 단계의 명령 예입니다.

```
echo Build started on `date` && mvn install
```

아래 예에서

- JAVA_HOME의 키 및 /usr/lib/jvm/java-8-openjdk-amd64의 값이 있는 일반 텍스트의 사용자 지정 환경 변수가 설정됩니다.
- Amazon EC2 Systems Manager Parameter Store에 저장된 dockerLoginPassword라는 사용자 지정 환경 변수는 나중에 LOGIN_PASSWORD 키를 사용하여 빌드 명령에서 참조됩니다.
- 이러한 빌드 단계 이름은 변경할 수 없습니다. 이 예에서 실행될 명령은 apt-get update -y 및 apt-get install -y maven(Apache Maven을 설치하는 데 사용), mvn install(소스 코드를

빌드 출력 아티팩트로 컴파일, 테스트 및 패키징하고 빌드 출력 아티팩트를 내부 리포지토리에 설치하는 데 사용), `docker login`(Amazon EC2 Systems Manager Parameter Store에 설정한 사용자 지정 환경 변수 `dockerLoginPassword`의 값에 해당하는 암호로 Docker에 로그인하는 데 사용) 및 여러 `echo` 명령입니다. `echo` 명령은 CodeBuild가 명령을 실행하는 방식 및 명령 실행 순서를 보여주기 위해 포함된 것입니다.

- `files`는 빌드 출력 위치에 업로드할 파일을 나타냅니다. 이 예에서 CodeBuild는 단일 파일 `messageUtil-1.0.jar`을 업로드합니다. `messageUtil-1.0.jar` 파일은 빌드 환경의 `target`이라는 상대적 디렉터리에서 찾을 수 있습니다. `discard-paths: yes`가 지정되어 있으므로, `messageUtil-1.0.jar`가 바로 업로드됩니다(중간의 `target` 디렉터리를 거치지 않음). 파일 이름 `messageUtil-1.0.jar` 및 상대적 디렉터리 이름 `target`은 Apache Maven이 이 예제에서만 빌드 출력 결과물을 생성 및 저장하는 방식에 따라 달라집니다. 사용자 자체 시나리오에서는 이러한 파일 이름과 디렉터리가 다릅니다.
- `reports`는 빌드 중에 보고서를 생성하는 두 개의 보고서 그룹을 나타냅니다.
  - `arn:aws:codebuild:your-region:your-aws-account-id:report-group/report-group-name-1`는 보고서 그룹의 ARN을 지정합니다. 테스트 프레임워크에 의해 생성된 테스트 결과는 `target/tests/reports` 디렉터리에 있습니다. 파일 형식은 `JunitXml`이고 테스트 결과가 포함된 파일에서 경로가 제거되지 않습니다.
  - `reportGroupCucumberJson`는 새 보고서 그룹을 지정합니다. 프로젝트 이름이 `my-project`인 경우 빌드가 실행될 때 이름이 `my-project-reportGroupCucumberJson`인 보고서 그룹이 생성됩니다. 테스트 프레임워크에 의해 생성된 테스트 결과가 `cucumber/target/cucumber-tests.xml`에 있습니다. 테스트 파일 형식은 `CucumberJson`이고 테스트 결과가 포함된 파일에서 경로가 제거됩니다.

buildspec 버전

다음 표에는 buildspec 버전과 버전 간의 변경 사항이 나열되어 있습니다.

버전	변경 사항
0.2	• `environment_variables` 의 이름이 `env`로 다시 지정되었습니다. • `plaintext` 의 이름이 `variables` 로 다시 지정되었습니다. • `artifacts` 의 `type` 속성이 사용 중단되었습니다.

버전	변경 사항
	• 버전 0.1에서는 빌드 환경의 기본 셸에 있는 별도의 인스턴스에서 각 빌드 명령을 AWS CodeBuild 실행합니다. 버전 0.2에서 CodeBuild는 빌드 환경에 있는 기본 셸의 동일한 인스턴스에서 모든 빌드 명령을 실행합니다.
0.1	이는 빌드 사양 형식의 초기 정의입니다.

배치 빌드 buildspec 참조

이 주제에는 배치 빌드 속성에 대한 buildspec 참조가 포함되어 있습니다.

일괄

선택적 매핑. 프로젝트에 대한 배치 빌드 설정입니다.

배치/빠른 실패

선택 사항. 하나 이상의 빌드 태스크가 실패할 경우 배치 빌드의 동작을 지정합니다.

`false`

기본값입니다. 실행 중인 모든 빌드가 완료됩니다.

`true`

빌드 태스크 중 하나가 실패하면 실행 중인 모든 빌드가 중지됩니다.

기본적으로 모든 배치 빌드 태스크는 buildspec 파일에 지정된 env 및 phases와 같은 빌드 설정으로 실행됩니다. batch/*<batch-type>*/buildspec 파라미터에 다른 env 값이나 다른 buildspec 파일을 지정하여 기본 빌드 설정을 재정의할 수 있습니다.

batch 속성의 내용은 지정된 배치 빌드 유형에 따라 달라집니다. 가능한 배치 빌드 유형은 다음과 같습니다.

- batch/build-graph
- batch/build-list

- batch/build-matrix
- batch/build-fanout

batch/build-graph

빌드 그래프를 정의합니다. 빌드 그래프는 일괄 처리의 다른 태스크에 종속되는 일련의 태스크를 정의합니다. 자세한 내용은 빌드 그래프 단원을 참조하십시오.

이 요소에는 빌드 태스크의 배열이 포함되어 있습니다. 각 빌드 태스크는 다음 속성을 포함합니다.

identifier

필수 사항입니다. 태스크의 식별자입니다.

buildspec

선택 사항. 태스크에 사용할 buildspec 파일의 경로 및 파일 이름입니다. 이 파라미터를 지정하지 않으면 현재 buildspec인 파일이 사용됩니다.

debug-session

선택 사항. 이 배치 빌드에 세션 디버깅을 활성화할지를 여부를 나타내는 부울 값입니다. 세션 디버깅에 대한 자세한 내용은 Session Manager를 사용하여 빌드 디버그 섹션을 참조하세요.

`false`

세션 디버깅이 비활성화되었습니다.

`true`

세션 디버깅이 활성화되었습니다.

depend-on

선택 사항. 이 태스크가 의존하는 태스크 식별자의 배열입니다. 이 태스크는 이러한 태스크가 완료될 때까지 실행되지 않습니다.

env

선택 사항. 태스크에 대한 빌드 환경 재정의입니다. 여기에는 다음 속성이 포함됩니다.

compute-type

태스크에 사용할 컴퓨팅 유형의 식별자입니다. 가능한 값은 the section called “빌드 환경 컴퓨팅 모드 및 유형”의 computeType을 참조하세요.

플릿

작업에 사용할 플릿의 식별자입니다. 자세한 정보는 the section called “예약 용량 플릿에서 빌드 실행”을 참조하세요.

image

태스크에 사용할 이미지의 식별자입니다. 가능한 값은 the section called “CodeBuild가 제공하는 도커 이미지”의 이미지 식별자를 참조하세요.

privileged-mode

Docker 컨테이너 내부에서 Docker 데몬(daemon)을 실행할지 여부를 나타내는 부울 값입니다. 빌드 프로젝트가 도커 이미지를 빌드하는 데 사용되는 경우에만 `true`로 설정합니다. 그렇지 않으면 도커 데몬과 상호 작용을 시도하는 빌드가 실패합니다. 기본 설정은 `false`입니다.

type

태스크에 사용할 환경 유형의 식별자입니다. 가능한 값은 the section called “빌드 환경 컴퓨팅 모드 및 유형”에서 환경 유형을 참조하세요.

variables

빌드 환경에 표시될 환경 변수입니다. 자세한 내용은 env/variables 섹션을 참조하세요.

> **Note**
> 단일 빌드의 동일한 식별자에는 컴퓨팅 유형과 플릿을 제공할 수 없습니다.

ignore-failure

선택 사항. 이 빌드 태스크의 실패를 무시할 수 있는지 여부를 나타내는 부울 값입니다.

`false`

기본값입니다. 이 빌드 태스크가 실패하면 배치 빌드가 실패합니다.

`true`

이 빌드 태스크가 실패하더라도 배치 빌드는 여전히 성공할 수 있습니다.

다음은 빌드 그래프 buildSpec 항목의 예제입니다.

```
batch:
  fast-fail: false
  build-graph:
    - identifier: build1
      env:
        variables:
          BUILD_ID: build1
      ignore-failure: false
    - identifier: build2
      buildspec: build2.yml
      env:
        variables:
          BUILD_ID: build2
      depend-on:
        - build1
    - identifier: build3
      env:
        variables:
          BUILD_ID: build3
      depend-on:
        - build2
    - identifier: build4
      env:
        compute-type: ARM_LAMBDA_1GB
    - identifier: build5
      env:
        fleet: fleet_name
```

batch/build-list

빌드 목록을 정의합니다. 빌드 목록은 병렬로 실행되는 여러 태스크를 정의하는 데 사용됩니다. 자세한 내용은 빌드 목록 단원을 참조하십시오.

이 요소에는 빌드 태스크의 배열이 포함되어 있습니다. 각 빌드 태스크는 다음 속성을 포함합니다.

identifier

필수 사항입니다. 태스크의 식별자입니다.

buildspec

선택 사항. 태스크에 사용할 buildspec 파일의 경로 및 파일 이름입니다. 이 파라미터를 지정하지 않으면 현재 buildspec인 파일이 사용됩니다.

debug-session

선택 사항. 이 배치 빌드에 세션 디버깅을 활성화할지를 여부를 나타내는 부울 값입니다. 세션 디버깅에 대한 자세한 내용은 Session Manager를 사용하여 빌드 디버그 섹션을 참조하세요.

`false`

세션 디버깅이 비활성화되었습니다.

`true`

세션 디버깅이 활성화되었습니다.

env

선택 사항. 태스크에 대한 빌드 환경 재정의입니다. 여기에는 다음 속성이 포함됩니다.

compute-type

태스크에 사용할 컴퓨팅 유형의 식별자입니다. 가능한 값은 the section called “빌드 환경 컴퓨팅 모드 및 유형”의 computeType을 참조하세요.

플릿

작업에 사용할 플릿의 식별자입니다. 자세한 정보는 the section called “예약 용량 플릿에서 빌드 실행”을 참조하세요.

image

태스크에 사용할 이미지의 식별자입니다. 가능한 값은 the section called “CodeBuild가 제공하는 도커 이미지”의 이미지 식별자를 참조하세요.

privileged-mode

Docker 컨테이너 내부에서 Docker 데몬(daemon)을 실행할지 여부를 나타내는 부울 값입니다. 빌드 프로젝트가 도커 이미지를 빌드하는 데 사용되는 경우에만 `true`로 설정합니다. 그렇지 않으면 도커 데몬과 상호 작용을 시도하는 빌드가 실패합니다. 기본 설정은 `false`입니다.

type

태스크에 사용할 환경 유형의 식별자입니다. 가능한 값은 the section called “빌드 환경 컴퓨팅 모드 및 유형”에서 환경 유형을 참조하세요.

variables

빌드 환경에 표시될 환경 변수입니다. 자세한 내용은 env/variables 섹션을 참조하세요.

> **Note**
>
> 단일 빌드의 동일한 식별자에는 컴퓨팅 유형과 플릿을 제공할 수 없습니다.

ignore-failure

선택 사항. 이 빌드 태스크의 실패를 무시할 수 있는지 여부를 나타내는 부울 값입니다.

`false`

기본값입니다. 이 빌드 태스크가 실패하면 배치 빌드가 실패합니다.

`true`

이 빌드 태스크가 실패하더라도 배치 빌드는 여전히 성공할 수 있습니다.

다음은 빌드 목록 buildSpec 항목의 예제입니다.

```
batch:
  fast-fail: false
  build-list:
    - identifier: build1
      env:
        variables:
          BUILD_ID: build1
      ignore-failure: false
    - identifier: build2
      buildspec: build2.yml
      env:
        variables:
          BUILD_ID: build2
      ignore-failure: true
    - identifier: build3
      env:
        compute-type: ARM_LAMBDA_1GB
    - identifier: build4
      env:
        fleet: fleet_name
    - identifier: build5
      env:
        compute-type: GENERAL_LINUX_XLAGRE
```

batch/build-matrix

빌드 매트릭스를 정의합니다. 빌드 매트릭스는 병렬로 실행되는 다양한 구성의 태스크를 정의합니다. CodeBuild는 가능한 각 구성 조합에 대해 별도의 빌드를 생성합니다. 자세한 내용은 빌드 매트릭스 단원을 참조하십시오.

static

정적 속성은 모든 빌드 태스크에 적용됩니다.

ignore-failure

선택 사항. 이 빌드 태스크의 실패를 무시할 수 있는지 여부를 나타내는 부울 값입니다.

`false`

기본값입니다. 이 빌드 태스크가 실패하면 배치 빌드가 실패합니다.

`true`

이 빌드 태스크가 실패하더라도 배치 빌드는 여전히 성공할 수 있습니다.

env

선택 사항. 모든 태스크에 대한 빌드 환경 재정의입니다.

privileged-mode

Docker 컨테이너 내부에서 Docker 데몬(daemon)을 실행할지 여부를 나타내는 부울 값입니다. 빌드 프로젝트가 도커 이미지를 빌드하는 데 사용되는 경우에만 `true`로 설정합니다. 그렇지 않으면 도커 데몬과 상호 작용을 시도하는 빌드가 실패합니다. 기본 설정은 `false`입니다.

type

태스크에 사용할 환경 유형의 식별자입니다. 가능한 값은 the section called “빌드 환경 컴퓨팅 모드 및 유형”에서 환경 유형을 참조하세요.

dynamic

동적 속성은 빌드 매트릭스를 정의합니다.

buildspec

선택 사항. 이러한 태스크에 사용할 buildspec 파일의 경로와 파일 이름을 포함하는 배열입니다. 이 파라미터를 지정하지 않으면 현재 buildspec인 파일이 사용됩니다.

env

선택 사항. 이러한 태스크에 대해 빌드 환경이 재정의됩니다.

compute-type

이러한 태스크에 사용할 컴퓨팅 유형의 식별자가 포함된 배열입니다. 가능한 값은 the section called “빌드 환경 컴퓨팅 모드 및 유형”의 computeType을 참조하세요.

image

이러한 태스크에 사용할 이미지의 식별자가 포함된 배열입니다. 가능한 값은 the section called “CodeBuild가 제공하는 도커 이미지”의 이미지 식별자를 참조하세요.

variables

이러한 태스크에 대한 빌드 환경에 표시될 환경 변수를 포함하는 배열입니다. 자세한 정보는 env/variables을 참조하세요.

다음은 빌드 매트릭스 buildSpec 항목의 예제입니다.

```
batch:
  build-matrix:
    static:
      ignore-failure: false
    dynamic:
      buildspec:
        - matrix1.yml
        - matrix2.yml
      env:
        variables:
          MY_VAR:
            - VALUE1
            - VALUE2
            - VALUE3
```

자세한 내용은 빌드 매트릭스 단원을 참조하십시오.

batch/build-fanout

빌드 팬아웃을 정의합니다. 빌드 팬아웃은 병렬로 실행되는 여러 빌드로 분할되는 작업을 정의하는 데 사용됩니다. 자세한 내용은 배치 빌드에서 병렬 테스트 실행 단원을 참조하십시오.

이 요소에는 여러 빌드로 분할할 수 있는 빌드 작업이 포함되어 있습니다. build-fanout 섹션은 다음 속성을 포함하고 있습니다.

병렬 처리

필수 사항입니다. 병렬로 테스트를 실행할 빌드 수입니다.

ignore-failure

선택 사항. 팬아웃 빌드 작업에서 실패를 무시할 수 있는지를 나타내는 부울 값입니다. 이 값인 ignore-failure는 모든 팬아웃 빌드에 적용됩니다.

false

기본값입니다. 팬아웃 빌드 작업이 실패하면 배치 빌드가 실패합니다.

true

팬아웃 빌드 작업이 실패하더라도 배치 빌드는 여전히 성공할 수 있습니다.

다음은 빌드 팬아웃 buildspec 항목의 예제입니다.

```
version: 0.2

batch:
   fast-fail: false
   build-fanout:
     parallelism: 5
     ignore-failure: false

phases:
  install:
    commands:
      - npm install
   build:
    commands:
      - mkdir -p test-results
      - cd test-results
      - |
        codebuild-tests-run \
         --test-command 'npx jest --runInBand --coverage' \
         --files-search "codebuild-glob-search '**/test/**/*.test.js'" \
         --sharding-strategy 'equal-distribution'
```

자세한 내용은 빌드 팬아웃 및 codebuild-tests-run CLI 명령 사용 섹션을 참조하세요.

에 대한 빌드 환경 참조 AWS CodeBuild

를 호출 AWS CodeBuild 하여 빌드를 실행할 때 빌드 환경에 대한 정보를 제공해야 합니다. 빌드 환경은 CodeBuild가 빌드를 실행하는 데 사용하는 운영 체제, 프로그래밍 언어 런타임 및 도구의 조합을 나타냅니다. 빌드 환경의 작동 방식에 대한 자세한 내용은 CodeBuild 작동 방식 섹션을 참조하세요.

빌드 환경에는 도커 이미지가 들어 있습니다. 자세한 내용은 Docker Docs 웹 사이트에서 Docker Glossary를 참조하십시오.

빌드 환경에 대한 정보를 CodeBuild에 제공할 때는 지원되는 리포지토리 유형의 도커 이미지 식별자를 지정해야 합니다. 여기에는 CodeBuild Docker 이미지 리포지토리, Docker Hub에서 공개적으로 사용 가능한 이미지, AWS 계정에 액세스 권한이 있는 Amazon Elastic Container Registry(Amazon ECR) 리포지토리가 포함됩니다.

- CodeBuild 도커 이미지 리포지토리에 저장된 도커 이미지는 해당 서비스에 사용하도록 최적화되어 있으므로 이를 사용하는 것이 좋습니다. 자세한 내용은 CodeBuild가 제공하는 도커 이미지 단원을 참조하십시오.
- Docker Hub에 저장되어 있는 공개적으로 사용 가능한 도커 이미지 식별자를 가져오려면 Docker Docs 웹 사이트의 리포지토리 검색을 참조하십시오.
- AWS 계정의 Amazon ECR 리포지토리에 저장되어 있는 도커 이미지로 작업하는 방법을 알아보려면 Amazon ECR 샘플 섹션을 참조하세요.

도커 이미지 식별자 외에도 빌드 환경에서 사용할 컴퓨팅 리소스 세트를 지정할 수 있습니다. 자세한 내용은 빌드 환경 컴퓨팅 모드 및 유형 단원을 참조하십시오.

주제

- CodeBuild가 제공하는 도커 이미지
- 빌드 환경 컴퓨팅 모드 및 유형
- 빌드 환경의 쉘 및 명령
- 빌드 환경의 환경 변수
- 빌드 환경의 배경 작업

CodeBuild가 제공하는 도커 이미지

지원되는 이미지는 CodeBuild에서 사용할 수 있는 이미지의 최신 주 버전이며 부 및 패치 버전 업데이트로 업데이트됩니다. CodeBuild는 지원되는 이미지를 머신의 Amazon Machine Images(AMI)에 캐싱하여 빌드의 프로비저닝 기간을 최적화합니다. 캐싱의 이점을 활용하고 빌드의 프로비저닝 기간을 최소화하려면 `aws/codebuild/amazonlinux-x86_64-standard:4.0-1.0.0`과 같은 보다 세분화된 버전 대신, CodeBuild 콘솔의 이미지 버전 섹션에서 이 런타임 버전에 대해 항상 최신 이미지 사용을 선택합니다.

주제

- 현재 Docker 이미지 목록 가져오기
- EC2 컴퓨팅 이미지
- Lambda 컴퓨팅 이미지
- 더 이상 사용되지 않는 CodeBuild 이미지
- 사용 가능한 런타임
- 실행 시간 버전

현재 Docker 이미지 목록 가져오기

CodeBuild는 도커 이미지 목록을 자주 업데이트하여 최신 이미지를 추가하고 이전 이미지는 더 이상 사용하지 않습니다. 최신 목록을 가져오려면 다음 중 하나를 수행합니다.

- CodeBuild 콘솔에 있는 빌드 프로젝트 생성 마법사 또는 빌드 프로젝트 편집 페이지의 환경 이미지에서 관리형 이미지를 선택합니다. 운영 체제, 런타임 및 런타임 버전 드롭다운 목록에서 선택합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 또는 빌드 프로젝트 설정 변경(콘솔) 섹션을 참조하세요.
- 의 경우 `list-curated-environment-images` 명령을 AWS CLI실행합니다.

```
aws codebuild list-curated-environment-images
```

- AWS SDKs의 경우 대상 프로그래밍 언어에 대한 `ListCuratedEnvironmentImages` 작업을 호출합니다. 자세한 내용은 AWS SDKs 및 도구 참조 섹션을 참조하세요.

EC2 컴퓨팅 이미지

AWS CodeBuild 는 CodeBuild에서 EC2 컴퓨팅에 사용할 수 있는 다음 Docker 이미지를 지원합니다.

> **Note**
>
> Windows Server Core 2019 플랫폼의 기본 이미지는 다음 리전에서만 사용할 수 있습니다.
>
> - 미국 동부(버지니아 북부)
> - 미국 동부(오하이오)
> - 미국 서부(오리건)
> - 유럽(아일랜드)

플랫폼	이미지 식별자	정의
Amazon Linux 2	`aws/codebuild/amazonlinux-x86_64-standard:4.0`	al/standard/4.0
Amazon Linux 2023	`aws/codebuild/amazonlinux-x86_64-standard:5.0`	al/standard/5.0
Amazon Linux 2023	`aws/codebuild/amazonlinux-x86_64-standard:6.0`	al/standard/6.0
Amazon Linux 2	`aws/codebuild/amazonlinux-x86_64-standard:corretto8`	al/standard/corretto8
Amazon Linux 2	`aws/codebuild/amazonlinux-x86_64-standard:corretto11`	al/standard/corretto11
Amazon Linux 2	`aws/codebuild/amazonlinux-aarch64-standard:2.0`	al/aarch64/standard/2.0

플랫폼	이미지 식별자	정의
Amazon Linux 2023	`aws/codebuild/amazonlinux-aarch64-standard:3.0`	al/aarch64/standard/3.0
Ubuntu 20.04	`aws/codebuild/standard:5.0`	ubuntu/standard/5.0
Ubuntu 22.04	`aws/codebuild/standard:6.0`	ubuntu/standard/6.0
Ubuntu 22.04	`aws/codebuild/standard:7.0`	ubuntu/standard/7.0
Ubuntu 24.04	`aws/codebuild/standard:8.0`	ubuntu/standard/8.0
Windows Server Core 2019	`aws/codebuild/windows-base:2019-1.0`	해당 사항 없음
Windows Server Core 2019	`aws/codebuild/windows-base:2019-2.0`	해당 사항 없음
Windows Server Core 2019	`aws/codebuild/windows-base:2019-3.0`	해당 사항 없음
Windows Server Core 2022	`aws/codebuild/windows-base:2022-1.0`	해당 사항 없음

Note

2024년 11월 22일에 Linux 기반 표준 런타임 이미지의 별칭이 `amazonlinux2`에서 `amazonlinux`로 업데이트되었습니다. 이전 별칭이 여전히 유효하므로 수동 업데이트가 필요하지 않습니다.

Lambda 컴퓨팅 이미지

AWS CodeBuild 는 CodeBuild에서 AWS Lambda 컴퓨팅에 사용할 수 있는 다음 Docker 이미지를 지원합니다.

> **Important**
>
> Lambda 컴퓨팅 또는 예약 용량을 사용할 때는 `on-failure` 속성이 지원되지 않습니다. 이 속성은 CodeBuild에서 제공하는 EC2 컴퓨팅 이미지에서만 작동합니다.

`aarch64` 아키텍처

플랫폼	이미지 식별자	정의
Amazon Linux 2	`aws/codebuild/amazonlinux-aarch64-lambda-standard:dotnet6`	al-lambda/aarch64/dotnet6
Amazon Linux 2023	`aws/codebuild/amazonlinux-aarch64-lambda-standard:dotnet8`	al-lambda/aarch64/dotnet8
Amazon Linux 2023	`aws/codebuild/amazonlinux-aarch64-lambda-standard:dotnet10`	al-lambda/aarch64/dotnet10
Amazon Linux 2	`aws/codebuild/amazonlinux-aarch64-lambda-standard:go1.21`	al-lambda/aarch64/go1.21
Amazon Linux 2023	`aws/codebuild/amazonlinux-aarch64-lambda-standard:go1.24`	al-lambda/aarch64/go1.24

플랫폼	이미지 식별자	정의
Amazon Linux 2023	`aws/codebuild/amazonlinux-aarch64-lambda-standard:go1.25`	al-lambda/aarch64/go1.25
Amazon Linux 2	`aws/codebuild/amazonlinux-aarch64-lambda-standard:corretto11`	al-lambda/aarch64/corretto11
Amazon Linux 2	`aws/codebuild/amazonlinux-aarch64-lambda-standard:corretto17`	al-lambda/aarch64/corretto17
Amazon Linux 2023	`aws/codebuild/amazonlinux-aarch64-lambda-standard:corretto21`	al-lambda/aarch64/corretto21
Amazon Linux 2023	`aws/codebuild/amazonlinux-aarch64-lambda-standard:corretto25`	al-lambda/aarch64/corretto25
Amazon Linux 2	`aws/codebuild/amazonlinux-aarch64-lambda-standard:nodejs18`	al-lambda/aarch64/nodejs18
Amazon Linux 2023	`aws/codebuild/amazonlinux-aarch64-lambda-standard:nodejs20`	al-lambda/aarch64/nodejs20

플랫폼	이미지 식별자	정의
Amazon Linux 2023	`aws/codebuild/amazonlinux-aarch64-lambda-standard:nodejs22`	al-lambda/aarch64/nodejs22
Amazon Linux 2023	`aws/codebuild/amazonlinux-aarch64-lambda-standard:nodejs24`	al-lambda/aarch64/nodejs24
Amazon Linux 2	`aws/codebuild/amazonlinux-aarch64-lambda-standard:python3.11`	al-lambda/aarch64/python3.11
Amazon Linux 2023	`aws/codebuild/amazonlinux-aarch64-lambda-standard:python3.12`	al-lambda/aarch64/python3.12
Amazon Linux 2023	`aws/codebuild/amazonlinux-aarch64-lambda-standard:python3.13`	al-lambda/aarch64/python3.13
Amazon Linux 2023	`aws/codebuild/amazonlinux-aarch64-lambda-standard:python3.14`	al-lambda/aarch64/python3.14
Amazon Linux 2	`aws/codebuild/amazonlinux-aarch64-lambda-standard:ruby3.2`	al-lambda/aarch64/ruby3.2

플랫폼	이미지 식별자	정의
Amazon Linux 2023	`aws/codebuild/amazonlinux-aarch64-lambda-standard:ruby3.4`	al-lambda/aarch64/ruby3.4

x86_64 아키텍처

플랫폼	이미지 식별자	정의
Amazon Linux 2	`aws/codebuild/amazonlinux-x86_64-lambda-standard:dotnet6`	al-lambda/x86_64/dotnet6
Amazon Linux 2023	`aws/codebuild/amazonlinux-x86_64-lambda-standard:dotnet8`	al-lambda/x86_64/dotnet8
Amazon Linux 2023	`aws/codebuild/amazonlinux-x86_64-lambda-standard:dotnet10`	al-lambda/x86_64/dotnet10
Amazon Linux 2	`aws/codebuild/amazonlinux-x86_64-lambda-standard:go1.21`	al-lambda/x86_64/go1.21
Amazon Linux 2023	`aws/codebuild/amazonlinux-x86_64-lambda-standard:go1.24`	al-lambda/x86_64/go1.24
Amazon Linux 2023	`aws/codebuild/amazonlinux-x86_64-lambda-standard:go1.25`	al-lambda/x86_64/go1.25

플랫폼	이미지 식별자	정의
Amazon Linux 2	`aws/codebuild/amazonlinux-x86_64-lambda-standard:corretto11`	al-lambda/x86_64/corretto11
Amazon Linux 2	`aws/codebuild/amazonlinux-x86_64-lambda-standard:corretto17`	al-lambda/x86_64/corretto17
Amazon Linux 2023	`aws/codebuild/amazonlinux-x86_64-lambda-standard:corretto21`	al-lambda/x86_64/corretto21
Amazon Linux 2023	`aws/codebuild/amazonlinux-x86_64-lambda-standard:corretto25`	al-lambda/x86_64/corretto25
Amazon Linux 2	`aws/codebuild/amazonlinux-x86_64-lambda-standard:nodejs18`	al-lambda/x86_64/nodejs18
Amazon Linux 2023	`aws/codebuild/amazonlinux-x86_64-lambda-standard:nodejs20`	al-lambda/x86_64/nodejs20
Amazon Linux 2023	`aws/codebuild/amazonlinux-x86_64-lambda-standard:nodejs22`	al-lambda/x86_64/nodejs22

플랫폼	이미지 식별자	정의
Amazon Linux 2023	`aws/codebuild/amazonlinux-x86_64-lambda-standard:nodejs24`	al-lambda/x86_64/nodejs24
Amazon Linux 2	`aws/codebuild/amazonlinux-x86_64-lambda-standard:python3.11`	al-lambda/x86_64/python3.11
Amazon Linux 2023	`aws/codebuild/amazonlinux-x86_64-lambda-standard:python3.12`	al-lambda/x86_64/python3.12
Amazon Linux 2023	`aws/codebuild/amazonlinux-x86_64-lambda-standard:python3.13`	al-lambda/x86_64/python3.13
Amazon Linux 2023	`aws/codebuild/amazonlinux-x86_64-lambda-standard:python3.14`	al-lambda/x86_64/python3.14
Amazon Linux 2	`aws/codebuild/amazonlinux-x86_64-lambda-standard:ruby3.2`	al-lambda/x86_64/ruby3.2
Amazon Linux 2023	`aws/codebuild/amazonlinux-x86_64-lambda-standard:ruby3.4`	al-lambda/x86_64/ruby3.4

더 이상 사용되지 않는 CodeBuild 이미지

더 이상 사용되지 않는 이미지는 CodeBuild에서 더 이상 캐시하거나 업데이트하지 않는 이미지입니다. 더 이상 사용되지 않는 이미지는 더 이상 부 버전 업데이트나 패치 버전 업데이트를 받지 않으며, 더 이상 업데이트되지 않으므로 이미지를 사용하는 것이 안전하지 않을 수 있습니다. CodeBuild 프로젝트가 이전 이미지 버전을 사용하도록 구성된 경우 프로비저닝 프로세스는 이 도커 이미지를 다운로드하고 이를 사용하여 컨테이너화된 런타임 환경을 생성하여 프로비저닝 기간과 전체 빌드 기간을 늘릴 수 있습니다.

CodeBuild는 다음 도커 이미지를 더 이상 사용하지 않습니다. 이러한 이미지는 계속 사용할 수 있지만 빌드 호스트에 캐시되지 않으므로 프로비저닝 시간이 길어집니다.

플랫폼	이미지 식별자	정의	사용 중단 날짜
Amazon Linux 2	`aws/codebuild/amazonlinux2-x86_64-standard:3.0`	al2/standard/3.0	2023년 5월 9일
Ubuntu 18.04	`aws/codebuild/standard:4.0`	ubuntu/standard/4.0	2023년 3월 31일
Amazon Linux 2	`aws/codebuild/amazonlinux2-aarch64-standard:1.0`	al2/aarch64/standard/1.0	2023년 3월 31일
Ubuntu 18.04	`aws/codebuild/standard:3.0`	ubuntu/standard/3.0	2022년 6월 30일
Amazon Linux 2	`aws/codebuild/amazonlinux2-x86_64-standard:2.0`	al2/standard/2.0	2022년 6월 30일

주제

- 사용 가능한 런타임

- 실행 시간 버전

사용 가능한 런타임

빌드 사양 파일의 `runtime-versions` 섹션에서 하나 이상의 런타임을 지정할 수 있습니다. 런타임이 다른 런타임에 종속되는 경우 빌드 사양 파일에서 종속 런타임을 지정할 수도 있습니다. buildspec 파일에 런타임을 지정하지 않은 경우 CodeBuild에서는 사용하는 이미지에서 사용할 수 있는 기본 런타임을 선택합니다. 하나 이상의 런타임을 지정하는 경우 CodeBuild에서는 해당 런타임만 사용합니다. 종속 런타임을 지정하지 않은 경우 CodeBuild에서 종속 런타임을 선택하려고 시도합니다.

> **Note**
>
> 런타임 버전이 지정되지 않은 경우 CodeBuild는 기본 버전을 사용합니다. 이전 기본 버전이 수명 종료(EOL)에 도달하면 기본 버전이 변경될 수 있습니다. 빌드 환경이 예기치 않게 변경되지 않도록 buildspec 파일에 런타임 버전을 지정하는 것이 좋습니다.

자세한 내용은 Specify runtime versions in the buildspec file 섹션을 참조하세요.

주제

- Linux 이미지 런타임
- macOS 이미지 런타임
- Windows 이미지 런타임

Linux 이미지 런타임

다음 표에는 사용 가능한 런타임과 이를 지원하는 표준 Linux 이미지가 나와 있습니다.

Ubuntu 및 Amazon Linux 플랫폼 런타임

실행 시간 이름	버전	이미지
dotnet	6.0	Amazon Linux 2 x86_64 Lambda 표준:dotnet6 Amazon Linux 2 AArch64 Lambda 표준:dotnet6

실행 시간 이름	버전	이미지
		Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 AArch64 표준:3.0 Ubuntu 22.04 표준:7.0
	8.0	Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 x86_64 표준:6.0 Amazon Linux 2023 AArch64 표준:3.0 Amazon Linux 2023 AArch64 표준:4.0 Ubuntu 22.04 표준:7.0 Ubuntu 24.04 표준:8.0
	10.0	Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 x86_64 표준:6.0 Amazon Linux 2023 AArch64 표준:3.0 Amazon Linux 2023 AArch64 표준:4.0 Ubuntu 22.04 표준:7.0 Ubuntu 24.04 표준:8.0

실행 시간 이름	버전	이미지
golang	1.20	Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 AArch64 표준:3.0 Ubuntu 22.04 표준:7.0
	1.21	Amazon Linux 2 x86_64 Lambda 표준:go1.21 Amazon Linux 2 AArch64 Lambda 표준:go1.21 Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 AArch64 표준:3.0 Ubuntu 22.04 표준:7.0
	1.22	Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 AArch64 표준:3.0 Ubuntu 22.04 표준:7.0
	1.23	Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 AArch64 표준:3.0 Ubuntu 22.04 표준:7.0

실행 시간 이름	버전	이미지
	1.24	Amazon Linux 2023 x86_64 Lambda 표준:go1.24 Amazon Linux 2023 AArch64 Lambda 표준:go1.24 Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 x86_64 표준:6.0 Amazon Linux 2023 AArch64 표준:3.0 Amazon Linux 2023 AArch64 표준:4.0 Ubuntu 22.04 표준:7.0 Ubuntu 24.04 표준:8.0
	1.25	Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 x86_64 표준:6.0 Amazon Linux 2023 AArch64 표준:3.0 Amazon Linux 2023 AArch64 표준:4.0 Ubuntu 22.04 표준:7.0 Ubuntu 24.04 표준:8.0

실행 시간 이름	버전	이미지
	1.26	Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 x86_64 표준:6.0 Amazon Linux 2023 AArch64 표준:3.0 Amazon Linux 2023 AArch64 표준:4.0 Ubuntu 22.04 표준:7.0 Ubuntu 24.04 표준:8.0
java	corretto8	Amazon Linux 2 x86_64 표준:corretto8 Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 x86_64 표준:6.0 Amazon Linux 2023 AArch64 표준:3.0 Amazon Linux 2023 AArch64 표준:4.0 Ubuntu 22.04 표준:7.0 Ubuntu 24.04 표준:8.0

실행 시간 이름	버전	이미지
	corretto11	Amazon Linux 2 x86_64 표준:corretto11 Amazon Linux 2 x86_64 Lambda 표준:corretto11 Amazon Linux 2 AArch64 Lambda 표준:corretto11 Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 x86_64 표준:6.0 Amazon Linux 2023 AArch64 표준:3.0 Amazon Linux 2023 AArch64 표준:4.0 Ubuntu 22.04 표준:7.0 Ubuntu 24.04 표준:8.0

실행 시간 이름	버전	이미지
	corretto17	Amazon Linux 2 x86_64 Lambda 표준:corretto17 Amazon Linux 2 AArch64 Lambda 표준:corretto17 Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 x86_64 표준:6.0 Amazon Linux 2023 AArch64 표준:3.0 Amazon Linux 2023 AArch64 표준:4.0 Ubuntu 22.04 표준:7.0 Ubuntu 24.04 표준:8.0

실행 시간 이름	버전	이미지
	corretto21	Amazon Linux 2 x86_64 Lambda 표준:corretto21 Amazon Linux 2 AArch64 Lambda 표준:corretto21 Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 x86_64 표준:6.0 Amazon Linux 2023 AArch64 표준:3.0 Amazon Linux 2023 AArch64 표준:4.0 Ubuntu 22.04 표준:7.0 Ubuntu 24.04 표준:8.0
	corretto25	Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 x86_64 표준:6.0 Amazon Linux 2023 AArch64 표준:3.0 Amazon Linux 2023 AArch64 표준:4.0 Ubuntu 22.04 표준:7.0 Ubuntu 24.04 표준:8.0

실행 시간 이름	버전	이미지
nodejs	18	Amazon Linux 2 x86_64 Lambda 표준:nodejs18 Amazon Linux 2 AArch64 Lambda 표준:nodejs18 Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 AArch64 표준:3.0 Ubuntu 22.04 표준:7.0
	20	Amazon Linux 2 x86_64 Lambda 표준:nodejs20 Amazon Linux 2 AArch64 Lambda 표준:nodejs20 Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 AArch64 표준:3.0 Ubuntu 22.04 표준:7.0

실행 시간 이름	버전	이미지
	22	Amazon Linux 2023 x86_64 Lambda 표준:nodejs22 Amazon Linux 2023 AArch64 Lambda 표준:nodejs22 Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 x86_64 표준:6.0 Amazon Linux 2023 AArch64 표준:3.0 Amazon Linux 2023 AArch64 표준:4.0 Ubuntu 22.04 표준:7.0 Ubuntu 24.04 표준:8.0
	24	Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 x86_64 표준:6.0 Amazon Linux 2023 AArch64 표준:3.0 Amazon Linux 2023 AArch64 표준:4.0 Ubuntu 22.04 표준:7.0 Ubuntu 24.04 표준:8.0

실행 시간 이름	버전	이미지
	26	Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 x86_64 표준:6.0 Ubuntu 22.04 표준:7.0 Ubuntu 24.04 표준:8.0
php	8.1	Amazon Linux 2023 AArch64 표준:3.0
	8.2	Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 x86_64 표준:6.0 Amazon Linux 2023 AArch64 표준:3.0 Amazon Linux 2023 AArch64 표준:4.0 Ubuntu 22.04 표준:7.0 Ubuntu 24.04 표준:8.0

실행 시간 이름	버전	이미지
	8.3	Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 x86_64 표준:6.0 Amazon Linux 2023 AArch64 표준:3.0 Amazon Linux 2023 AArch64 표준:4.0 Ubuntu 22.04 표준:7.0 Ubuntu 24.04 표준:8.0
	8.4	Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 x86_64 표준:6.0 Amazon Linux 2023 AArch64 표준:3.0 Amazon Linux 2023 AArch64 표준:4.0 Ubuntu 22.04 표준:7.0 Ubuntu 24.04 표준:8.0

실행 시간 이름	버전	이미지
	8.5	Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 x86_64 표준:6.0 Amazon Linux 2023 AArch64 표준:3.0 Amazon Linux 2023 AArch64 표준:4.0 Ubuntu 22.04 표준:7.0 Ubuntu 24.04 표준:8.0
python	3.9	Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 AArch64 표준:3.0 Ubuntu 22.04 표준:7.0
	3.10	Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 x86_64 표준:6.0 Amazon Linux 2023 AArch64 표준:3.0 Amazon Linux 2023 AArch64 표준:4.0 Ubuntu 22.04 표준:7.0 Ubuntu 24.04 표준:8.0

실행 시간 이름	버전	이미지
	3.11	Amazon Linux 2 x86_64 Lambda 표준:python3.11 Amazon Linux 2 AArch64 Lambda 표준:python3.11 Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 x86_64 표준:6.0 Amazon Linux 2023 AArch64 표준:3.0 Amazon Linux 2023 AArch64 표준:4.0 Ubuntu 22.04 표준:7.0 Ubuntu 24.04 표준:8.0

실행 시간 이름	버전	이미지
	3.12	Amazon Linux 2 x86_64 Lambda 표준:python3.12 Amazon Linux 2 AArch64 Lambda 표준:python3.12 Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 x86_64 표준:6.0 Amazon Linux 2023 AArch64 표준:3.0 Amazon Linux 2023 AArch64 표준:4.0 Ubuntu 22.04 표준:7.0 Ubuntu 24.04 표준:8.0

실행 시간 이름	버전	이미지
	3.13	Amazon Linux 2023 x86_64 Lambda 표준:python3.13 Amazon Linux 2023 AArch64 Lambda 표준:python3.13 Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 x86_64 표준:6.0 Amazon Linux 2023 AArch64 표준:3.0 Amazon Linux 2023 AArch64 표준:4.0 Ubuntu 22.04 표준:7.0 Ubuntu 24.04 표준:8.0
	3.14	Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 x86_64 표준:6.0 Amazon Linux 2023 AArch64 표준:3.0 Amazon Linux 2023 AArch64 표준:4.0 Ubuntu 22.04 표준:7.0 Ubuntu 24.04 표준:8.0

실행 시간 이름	버전	이미지
ruby	3.1	Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 AArch64 표준:3.0 Ubuntu 22.04 표준:7.0
	3.2	Amazon Linux 2 x86_64 Lambda 표준:ruby3.2 Amazon Linux 2 AArch64 Lambda 표준:ruby3.2 Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 AArch64 표준:3.0 Ubuntu 22.04 표준:7.0
	3.3	Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 x86_64 표준:6.0 Amazon Linux 2023 AArch64 표준:3.0 Amazon Linux 2023 AArch64 표준:4.0 Ubuntu 22.04 표준:7.0 Ubuntu 24.04 표준:8.0

실행 시간 이름	버전	이미지
	3.4	Amazon Linux 2023 x86_64 Lambda 표준:ruby3.4 Amazon Linux 2023 AArch64 Lambda 표준:ruby3.4 Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 x86_64 표준:6.0 Amazon Linux 2023 AArch64 표준:3.0 Amazon Linux 2023 AArch64 표준:4.0 Ubuntu 22.04 표준:7.0 Ubuntu 24.04 표준:8.0
	4.0	Amazon Linux 2023 x86_64 표준:5.0 Amazon Linux 2023 x86_64 표준:6.0 Amazon Linux 2023 AArch64 표준:3.0 Amazon Linux 2023 AArch64 표준:4.0 Ubuntu 22.04 표준:7.0 Ubuntu 24.04 표준:8.0

실행 시간 이름	버전	이미지
rust	1.94	Amazon Linux 2023 AArch64 표준:3.0 Amazon Linux 2023 AArch64 표준:4.0
	1.97	Amazon Linux 2023 x86_64 표준:6.0 Ubuntu 24.04 표준:8.0

macOS 이미지 런타임

> **Important**
>
> Mac 빌드용 CodeBuild 큐레이션 이미지에는 macOS 및 Xcode가 사전 설치되어 있습니다. Xcode 소프트웨어를 사용하면 Xcode 및 Apple SDK 계약을 인정하고 이해하며 이에 동의하는 것입니다. 계약 약관에 동의하지 않는 경우 Xcode 소프트웨어를 사용하지 마십시오. 대신 자체 Amazon Machine Images(AMI)를 제공합니다. 자세한 내용은 예약 용량 macOS 플릿을 구성하려면 어떻게 해야 합니까? 섹션을 참조하세요.

다음 표에는 macOS에서 지원하는 사용 가능한 런타임이 포함되어 있습니다.

macOS 플랫폼 런타임

실행 시간 이름	버전	이미지	추가 참고 사항
bash	3.2.57	macos-arm-base:15 macos-arm-base:26	
clang	17.0.0	macos-arm-base:15 macos-arm-base:26	
dotnet sdk	8.0.416	macos-arm-base:15	

실행 시간 이름	버전	이미지	추가 참고 사항
	8.0.417	macos-arm-base:26	
	10.0.101	macos-arm-base:15	
	10.0.102	macos-arm-base:26	
gcc	11.5.0	macos-arm-base:15 macos-arm-base:26	`gcc-11` 별칭을 사용하여 사용 가능
	12.4.0	macos-arm-base:15 macos-arm-base:26	`gcc-12` 별칭을 사용하여 사용 가능
	13.4.0	macos-arm-base:15 macos-arm-base:26	`gcc-13` 별칭을 사용하여 사용 가능
	14.3.0	macos-arm-base:15 macos-arm-base:26	`gcc-14` 별칭을 사용하여 사용 가능
gnu	11.5.0	macos-arm-base:15 macos-arm-base:26	`gfortran-11` 별칭을 사용하여 사용 가능
	12.4.0	macos-arm-base:15 macos-arm-base:26	`gfortran-12` 별칭을 사용하여 사용 가능
	13.4.0	macos-arm-base:15 macos-arm-base:26	`gfortran-13` 별칭을 사용하여 사용 가능
	14.3.0	macos-arm-base:15 macos-arm-base:26	`gfortran-14` 별칭을 사용하여 사용 가능

실행 시간 이름	버전	이미지	추가 참고 사항
golang	1.24.11	macos-arm-base:15 macos-arm-base:26	
	1.25.4	macos-arm-base:15 macos-arm-base:26	
java	Corretto8	macos-arm-base:15 macos-arm-base:26	
	corretto11	macos-arm-base:15 macos-arm-base:26	
	Corretto17	macos-arm-base:15 macos-arm-base:26	
	corretto21	macos-arm-base:15 macos-arm-base:26	
	Corretto25	macos-arm-base:15 macos-arm-base:26	
kotlin	2.2.21	macos-arm-base:15	
	2.3.0	macos-arm-base:26	
mono	6.14.1	macos-arm-base:15 macos-arm-base:26	
nodejs	20.19.6	macos-arm-base:15 macos-arm-base:26	

실행 시간 이름	버전	이미지	추가 참고 사항
	22.21.1	macos-arm-base:15 macos-arm-base:26	
	24.11.1	macos-arm-base:15 macos-arm-base:26	
perl	5.34.1	macos-arm-base:15 macos-arm-base:26	
php	8.2.29	macos-arm-base:15	
	8.2.30	macos-arm-base:26	
	8.3.28	macos-arm-base:15	
	8.3.30	macos-arm-base:26	
	8.4.15	macos-arm-base:15	
	8.4.17	macos-arm-base:26	
	8.5.0	macos-arm-base:15	
	8.5.2	macos-arm-base:26	
python	3.10.19	macos-arm-base:15 macos-arm-base:26	
	3.11.14	macos-arm-base:15 macos-arm-base:26	
	3.12.12	macos-arm-base:15 macos-arm-base:26	

실행 시간 이름	버전	이미지	추가 참고 사항
	3.13.10	macos-arm-base:15 macos-arm-base:26	
	3.14.1	macos-arm-base:15 macos-arm-base:26	
ruby	3.2.9	macos-arm-base:15 macos-arm-base:26	
	3.3.10	macos-arm-base:15 macos-arm-base:26	
	3.4.7	macos-arm-base:15 macos-arm-base:26	
rust	1.91.1	macos-arm-base:15	
	1.92.0	macos-arm-base:26	
swift	6.2.1	macos-arm-base:15	
	6.2.3	macos-arm-base:26	
Xcode	26.1.1	macos-arm-base:15	
	26.2	macos-arm-base:26	

Windows 이미지 런타임

Windows Server Core 2019의 기본 이미지에는 다음 실행 시간이 포함됩니다.

Windows 플랫폼 런타임

실행 시간 이름	Windows Server Core 2019 표준:1.0 버전	Windows Server Core 2019 표준:2.0 버전	Windows Server Core 2019 표준:3.0 버전
dotnet	3.1 5.0	3.1 6.0 7.0	8.0
dotnet sdk	3.1 5.0	3.1 6.0 7.0	8.0
golang	1.14	1.18	1.21 1.22 1.23
gradle	6.7	7.6	8.12
java	corretto11	corretto11 corretto17	Corretto8 corretto11 corretto17 corretto21
maven	3.6	3.8	3.9
nodejs	14.15	16.19	20.18 22.13
php	7.4	8.1	8.3 8.4

실행 시간 이름	Windows Server Core 2019 표준:1.0 버전	Windows Server Core 2019 표준:2.0 버전	Windows Server Core 2019 표준:3.0 버전
powershell	7.1	7.2	7.4
python	3.8	3.10	3.10 3.11 3.12 3.13
ruby	2.7	3.1	3.2 3.3 3.4

실행 시간 버전

buildspec 파일의 `runtime-versions` 섹션에서 런타임을 지정할 때 특정 버전, 특정 메이저 버전 및 최신 마이너 버전 또는 최신 버전을 지정할 수 있습니다. 다음 표에는 사용 가능한 런타임과 이를 지정하는 방법이 나와 있습니다. 모든 이미지에서 모든 런타임 버전을 사용할 수 있는 것은 아닙니다. 사용자 지정 이미지에는 런타임 버전 선택도 지원되지 않습니다. 자세한 내용은 사용 가능한 런타임 섹션을 참조하세요. 사전 설치된 런타임 버전 대신 사용자 지정 런타임 버전을 설치하고 사용하려면 사용자 지정 런타임 버전 섹션을 참조하세요.

Ubuntu 및 Amazon Linux 2 플랫폼 런타임 버전

실행 시간 이름	버전	특정 버전	특정 메이저 버전 및 최신 마이너 버전	최신 버전
dotnet	6.0	`dotnet: 6.0`	`dotnet: 6.x`	`dotnet: latest`
	8.0	`dotnet: 8.0`	`dotnet: 8.x`	
	10.0	`dotnet: 10.0`	`dotnet: 10.x`	

실행 시간 이름	버전	특정 버전	특정 메이저 버전 및 최신 마이너 버전	최신 버전
golang	1.20	golang: 1.20	golang: 1.x	golang: latest
	1.21	golang: 1.21		
	1.22	golang: 1.22		
	1.23	golang: 1.23		
	1.24	golang: 1.24		
	1.25	golang: 1.25		
	1.26	golang: 1.26		
java	corretto8	java: corretto	java: corretto .x	java: latest
	corretto11	java: corretto 1	java: corretto 1.x	
	corretto17	java: corretto 7	java: corretto 7.x	
	corretto21	java: corretto 1	java: corretto 1.x	
	corretto25	java: corretto 5	java: corretto 5.x	
nodejs	18	nodejs: 18	nodejs: 18.x	nodejs: latest
	20	nodejs: 20	nodejs: 20.x	
	22	nodejs: 22	nodejs: 22.x	
	24	nodejs: 24	nodejs: 24.x	

실행 시간 이름	버전	특정 버전	특정 메이저 버전 및 최신 마이너 버전	최신 버전
php	8.1	`php: 8.1`	`php: 8.x`	`php: latest`
	8.2	`php: 8.2`		
	8.3	`php: 8.3`		
	8.4	`php: 8.4`		
	8.5	`php: 8.5`		
python	3.9	`python: 3.9`	`python: 3.x`	`python: latest`
	3.10	`python: 3.10`		
	3.11	`python: 3.11`		
	3.12	`python: 3.12`		
	3.13	`python: 3.13`		
	3.14	`python: 3.14`		
ruby	3.1	`ruby: 3.1`	`ruby: 3.x`	`ruby: latest`
	3.2	`ruby: 3.2`		
	3.3	`ruby: 3.3`		
	3.4	`ruby: 3.4`		
	4.0	`ruby: 4.0`	`ruby: 4.x`	
rust	1.94	`rust: 1.94`	`rust: 1.x`	`rust: latest`

빌드 사양을 사용하여 `install` 빌드 단계에서 다른 구성 요소(예: AWS CLI Apache Maven, Apache Ant, Mocha, RSpec 등)를 설치할 수 있습니다. 자세한 내용은 buildspec 예제 섹션을 참조하세요.

사용자 지정 런타임 버전

CodeBuild 관리형 이미지에서 사전 설치된 런타임 버전을 사용하는 대신 원하는 사용자 지정 버전을 설치하고 사용할 수 있습니다. 다음 표에는 사용 가능한 사용자 지정 런타임과 이를 지정하는 방법이 나와 있습니다.

> **Note**
>
> 사용자 지정 런타임 버전 선택은 Ubuntu 및 Amazon Linux 이미지에서만 지원됩니다.

사용자 지정 런타임 버전

실행 시간 이름	구문	예제
dotnet	`<major>.<minor>.<patch>`	`5.0.408`
golang	`<major>.<minor>` `<major>.<minor>.<patch>`	`1.19` `1.19.1`
java	`corretto<major>`	`corretto15`

nodejs	`<major>`
	`<major>.<minor>`
	`<major>.<minor>.<patch>`
		`14`
		`14.21`
		`14.21.3`

php	`<major>.<minor>.<patch>`	`8.0.30`

python	`<major>`
	`<major>.<minor>`
	`<major>.<minor>.<patch>`
		`3`
		`3.7`
		`3.7.16`

ruby	`<major>.<minor>.<patch>`	`3.0.6`

사용자 지정 런타임 buildspec 예제

다음은 사용자 지정 런타임 버전을 지정하는 buildspec의 예입니다.

```
version: 0.2
phases:
  install:
    runtime-versions:
      java: corretto15
      php: 8.0.30
      ruby: 3.0.6
      golang: 1.19
      python: 3.7
      nodejs: 14
      dotnet: 5.0.408
```

빌드 환경 컴퓨팅 모드 및 유형

CodeBuild에서 CodeBuild가 빌드를 실행하는 데 사용하는 컴퓨팅 및 런타임 환경 이미지를 지정할 수 있습니다. 컴퓨팅은 CodeBuild에서 관리하고 유지하는 컴퓨팅 엔진(CPU, 메모리, 운영 체제)을 말합니다. 런타임 환경 이미지는 선택한 컴퓨팅 플랫폼에서 실행되는 컨테이너 이미지이며, 빌드에 필요할 수 있는 추가 도구(예: AWS CLI)가 포함되어 있습니다.

주제

- 컴퓨팅 정보
- 예약 용량 환경 유형 정보
- 온디맨드 환경 유형 정보

컴퓨팅 정보

CodeBuild는 EC2 및 AWS Lambda 컴퓨팅 모드를 제공합니다. EC2는 빌드 AWS Lambda 중에 최적화된 유연성을 제공하고 최적화된 시작 속도를 제공합니다.는 시작 지연 시간이 짧아 더 빠른 빌드를 AWS Lambda 지원합니다. AWS Lambda 또한는 자동으로 확장되므로 빌드가 대기열에서 실행되기를 기다리지 않습니다. 자세한 내용은 AWS Lambda 컴퓨팅 기반 빌드 실행 단원을 참조하십시오.

EC2 컴퓨팅 모드를 사용하면 온디맨드 또는 예약 용량 플릿으로 빌드를 실행할 수 있습니다. 온디맨드 플릿의 경우 `BUILD_GENERAL1_SMALL` 또는와 같이 사전 정의된 컴퓨팅 유형을 선택할 수 있습니다`BUILD_GENERAL1_LARGE`. 자세한 내용은 온디맨드 환경 유형 정보 단원을 참조하십시오. 예약 용

량 플릿의 경우 vCPU, 메모리 및 디스크 공간을 포함한 컴퓨팅 구성을 선택할 수 있습니다. 구성을 지정한 후 CodeBuild는 요구 사항에 맞는 지원되는 컴퓨팅 유형을 선택합니다. 자세한 내용은 예약 용량 환경 유형 정보 단원을 참조하십시오.

예약 용량 환경 유형 정보

AWS CodeBuild 는 예약 용량 플릿에 Linux x86, Arm, GPU, Windows 및 macOS 환경 유형을 제공합니다. 다음 표에는 리전별로 정렬된 사용 가능한 시스템 유형, 메모리, vCPU 및 디스크 공간이 나와 있습니다.

US East (N. Virginia)

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
ARM	2	4GiB	64GB	GENERAL	reserved.arm.2cpu.4gib
ARM	4	8GiB	128GB	GENERAL	reserved.arm.4cpu.8gib
ARM	8	16GiB	128GB	GENERAL	reserved.arm.8cpu.16gib
ARM	16	32GiB	256GB	GENERAL	reserved.arm.16cpu.32gib
ARM	32	64GiB	256GB	GENERAL	reserved.arm.32cpu.64gib
ARM	48	96GiB	512GB	GENERAL	reserved.arm.48cpu.96gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
ARM	64	128GiB	824GB	GENERAL	reserved.arm.64cpu.128gib
ARM EC2	2	4GiB	64GB	GENERAL	reserved.arm.2cpu.4gib
ARM EC2	4	8GiB	128GB	GENERAL	reserved.arm.4cpu.8gib
ARM EC2	8	16GiB	128GB	GENERAL	reserved.arm.8cpu.16gib
Linux	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux	16	32GiB	256GB	GENERAL	reserved.x86-64.16cpu.32gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Linux	36	72GiB	256GB	GENERAL	reserved.x86-64.36cpu.72gib
Linux	48	96GiB	512GB	GENERAL	reserved.x86-64.48cpu.96gib
Linux	72	144GiB	824GB	GENERAL	reserved.x86-64.72cpu.144gib
Linux	96	192GiB	824GB	GENERAL	reserved.x86-64.96cpu.192gib
Linux	48	96GiB	824 GB(SSD)	NVME	reserved.x86-64.48cpu.96gib.nvme
Linux	72	144GiB	824 GB(SSD)	NVME	reserved.x86-64.72cpu.144gib.nvme
Linux EC2	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Linux EC2	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux EC2	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux GPU	4	16GiB	235GB(SSD)	NVME	reserved.gpu.4cpu.16gib.nvme
Linux GPU	8	32GiB	435GB(SSD)	NVME	reserved.gpu.8cpu.32gib.nvme
Linux GPU	16	64GiB	585GB(SSD)	NVME	reserved.gpu.16cpu.64gib.nvme
Linux GPU	32	128GiB	885GB(SSD)	NVME	reserved.gpu.32cpu.128gib.nvme
Linux GPU	48	192GiB	3785GB(SSD)	NVME	reserved.gpu.48cpu.192gib.nvme

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Linux GPU	64	256GiB	1885GB(SSD)	NVME	reserved.gpu.64cpu.256gib.nvme
Linux GPU	96	384 GiB	3785GB(SSD)	NVME	reserved.gpu.96cpu.384gib.nvme
macOS	8	24GiB	128GB	GENERAL	reserved.arm.m2.8cpu.24gib
macOS	12	32GiB	256GB	GENERAL	reserved.arm.m2.12cpu.32gib
Windows	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib
Windows	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Windows	16	32GiB	256GB	GENERAL	reserved.x86-64.16cpu.32gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Windows	32	64GiB	256GB	GENERAL	reserved.x86-64.32cpu.64gib
Windows	36	72GiB	256GB	GENERAL	reserved.x86-64.36cpu.72gib
Windows	48	96GiB	512GB	GENERAL	reserved.x86-64.48cpu.96gib
Windows	64	128GiB	512GB	GENERAL	reserved.x86-64.64cpu.128gib
Windows	72	144GiB	824GB	GENERAL	reserved.x86-64.72cpu.144gib
Windows	96	192GiB	824GB	GENERAL	reserved.x86-64.96cpu.192gib
Windows EC2	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Windows EC2	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib

요금 식별자에 대한 자세한 내용은 https://aws.amazon.com/codebuild/pricing/을 참조하세요.

US East (Ohio)

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
ARM	2	4GiB	64GB	GENERAL	reserved.arm.2cpu.4gib
ARM	4	8GiB	128GB	GENERAL	reserved.arm.4cpu.8gib
ARM	8	16GiB	128GB	GENERAL	reserved.arm.8cpu.16gib
ARM	16	32GiB	256GB	GENERAL	reserved.arm.16cpu.32gib
ARM	32	64GiB	256GB	GENERAL	reserved.arm.32cpu.64gib
ARM	48	96GiB	512GB	GENERAL	reserved.arm.48cpu.96gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
ARM	64	128GiB	824GB	GENERAL	reserved.arm.64cpu.128gib
ARM EC2	2	4GiB	64GB	GENERAL	reserved.arm.2cpu.4gib
ARM EC2	4	8GiB	128GB	GENERAL	reserved.arm.4cpu.8gib
ARM EC2	8	16GiB	128GB	GENERAL	reserved.arm.8cpu.16gib
Linux	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux	16	32GiB	256GB	GENERAL	reserved.x86-64.16cpu.32gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Linux	36	72GiB	256GB	GENERAL	reserved.x86-64.36cpu.72gib
Linux	48	96GiB	512GB	GENERAL	reserved.x86-64.48cpu.96gib
Linux	96	192GiB	824GB	GENERAL	reserved.x86-64.96cpu.192gib
Linux	48	96GiB	824 GB(SSD)	NVME	reserved.x86-64.48cpu.96gib.nvme
Linux	72	144GiB	824 GB(SSD)	NVME	reserved.x86-64.72cpu.144gib.nvme
Linux EC2	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux EC2	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Linux EC2	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux GPU	4	16GiB	235GB(SSD)	NVME	reserved.gpu.4cpu.16gib.nvme
Linux GPU	8	32GiB	435GB(SSD)	NVME	reserved.gpu.8cpu.32gib.nvme
Linux GPU	16	64GiB	585GB(SSD)	NVME	reserved.gpu.16cpu.64gib.nvme
Linux GPU	32	128GiB	885GB(SSD)	NVME	reserved.gpu.32cpu.128gib.nvme
Linux GPU	48	192GiB	3785GB(SSD)	NVME	reserved.gpu.48cpu.192gib.nvme
macOS	8	24GiB	128GB	GENERAL	reserved.arm.m2.8cpu.24gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
macOS	12	32GiB	256GB	GENERAL	reserved.arm.m2.12cpu.32gib
Windows	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib
Windows	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Windows	16	32GiB	256GB	GENERAL	reserved.x86-64.16cpu.32gib
Windows	32	64GiB	256GB	GENERAL	reserved.x86-64.32cpu.64gib
Windows	36	72GiB	256GB	GENERAL	reserved.x86-64.36cpu.72gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Windows	48	96GiB	512GB	GENERAL	reserved.x86-64.48cpu.96gib
Windows	64	128GiB	512GB	GENERAL	reserved.x86-64.64cpu.128gib
Windows	96	192GiB	824GB	GENERAL	reserved.x86-64.96cpu.192gib
Windows EC2	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows EC2	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib

요금 식별자에 대한 자세한 내용은 https://aws.amazon.com/codebuild/pricing/을 참조하세요.

US West (N. California)

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
ARM	2	4GiB	64GB	GENERAL	reserved.arm.2cpu.4gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
ARM	4	8GiB	128GB	GENERAL	reserved.arm.4cpu.8gib
ARM	8	16GiB	128GB	GENERAL	reserved.arm.8cpu.16gib
ARM	16	32GiB	256GB	GENERAL	reserved.arm.16cpu.32gib
ARM	32	64GiB	256GB	GENERAL	reserved.arm.32cpu.64gib
ARM	48	96GiB	824GB	GENERAL	reserved.arm.48cpu.96gib
ARM	64	128GiB	824GB	GENERAL	reserved.arm.64cpu.128gib
ARM EC2	2	4GiB	64GB	GENERAL	reserved.arm.2cpu.4gib
ARM EC2	4	8GiB	128GB	GENERAL	reserved.arm.4cpu.8gib
ARM EC2	8	16GiB	128GB	GENERAL	reserved.arm.8cpu.16gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Linux	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux	16	32GiB	256GB	GENERAL	reserved.x86-64.16cpu.32gib
Linux	32	64GiB	256GB	GENERAL	reserved.x86-64.32cpu.64gib
Linux	36	72GiB	256GB	GENERAL	reserved.x86-64.36cpu.72gib
Linux	48	96GiB	512GB	GENERAL	reserved.x86-64.48cpu.96gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Linux	64	128GiB	512GB	GENERAL	reserved.x86-64.64cpu.128gib
Linux	72	144GiB	824GB	GENERAL	reserved.x86-64.72cpu.144gib
Linux	96	192GiB	824GB	GENERAL	reserved.x86-64.96cpu.192gib
Linux	72	144GiB	824 GB(SSD)	NVME	reserved.x86-64.72cpu.144gib.nvme
Linux EC2	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux EC2	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux EC2	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Windows	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Windows	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Windows	16	32GiB	256GB	GENERAL	reserved.x86-64.16cpu.32gib
Windows	32	64GiB	256GB	GENERAL	reserved.x86-64.32cpu.64gib
Windows	36	72GiB	256GB	GENERAL	reserved.x86-64.36cpu.72gib
Windows	48	96GiB	512GB	GENERAL	reserved.x86-64.48cpu.96gib
Windows	64	128GiB	512GB	GENERAL	reserved.x86-64.64cpu.128gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Windows	72	144GiB	824GB	GENERAL	reserved.x86-64.72cpu.144gib
Windows	96	192GiB	824GB	GENERAL	reserved.x86-64.96cpu.192gib
Windows EC2	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows EC2	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib

요금 식별자에 대한 자세한 내용은 https://aws.amazon.com/codebuild/pricing/을 참조하세요.

US West (Oregon)

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
ARM	2	4GiB	64GB	GENERAL	reserved.arm.2cpu.4gib
ARM	4	8GiB	128GB	GENERAL	reserved.arm.4cpu.8gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
ARM	8	16GiB	128GB	GENERAL	reserved.arm.8cpu.16gib
ARM	16	32GiB	256GB	GENERAL	reserved.arm.16cpu.32gib
ARM	32	64GiB	256GB	GENERAL	reserved.arm.32cpu.64gib
ARM	48	96GiB	512GB	GENERAL	reserved.arm.48cpu.96gib
ARM	64	128GiB	824GB	GENERAL	reserved.arm.64cpu.128gib
ARM EC2	2	4GiB	64GB	GENERAL	reserved.arm.2cpu.4gib
ARM EC2	4	8GiB	128GB	GENERAL	reserved.arm.4cpu.8gib
ARM EC2	8	16GiB	128GB	GENERAL	reserved.arm.8cpu.16gib
Linux	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Linux	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux	16	32GiB	256GB	GENERAL	reserved.x86-64.16cpu.32gib
Linux	36	72GiB	256GB	GENERAL	reserved.x86-64.36cpu.72gib
Linux	48	96GiB	512GB	GENERAL	reserved.x86-64.48cpu.96gib
Linux	72	144GiB	824GB	GENERAL	reserved.x86-64.72cpu.144gib
Linux	96	192GiB	824GB	GENERAL	reserved.x86-64.96cpu.192gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Linux	48	96GiB	824 GB(SSD)	NVME	reserved.x86-64.48cpu.96gib.nvme
Linux	72	144GiB	824 GB(SSD)	NVME	reserved.x86-64.72cpu.144gib.nvme
Linux EC2	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux EC2	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux EC2	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux GPU	4	16GiB	235GB(SSD)	NVME	reserved.gpu.4cpu.16gib.nvme
Linux GPU	8	32GiB	435GB(SSD)	NVME	reserved.gpu.8cpu.32gib.nvme

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Linux GPU	16	64GiB	585GB(SSD)	NVME	reserved.gpu.16cpu.64gib.nvme
Linux GPU	32	128GiB	885GB(SSD)	NVME	reserved.gpu.32cpu.128gib.nvme
Linux GPU	48	192GiB	3785GB(SSD)	NVME	reserved.gpu.48cpu.192gib.nvme
Linux GPU	64	256GiB	1885GB(SSD)	NVME	reserved.gpu.64cpu.256gib.nvme
macOS	8	24GiB	128GB	GENERAL	reserved.arm.m2.8cpu.24gib
macOS	12	32GiB	256GB	GENERAL	reserved.arm.m2.12cpu.32gib
Windows	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Windows	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Windows	16	32GiB	256GB	GENERAL	reserved.x86-64.16cpu.32gib
Windows	32	64GiB	256GB	GENERAL	reserved.x86-64.32cpu.64gib
Windows	36	72GiB	256GB	GENERAL	reserved.x86-64.36cpu.72gib
Windows	48	96GiB	512GB	GENERAL	reserved.x86-64.48cpu.96gib
Windows	64	128GiB	512GB	GENERAL	reserved.x86-64.64cpu.128gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Windows	72	144GiB	824GB	GENERAL	reserved.x86-64.72cpu.144gib
Windows	96	192GiB	824GB	GENERAL	reserved.x86-64.96cpu.192gib
Windows EC2	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows EC2	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib

요금 식별자에 대한 자세한 내용은 https://aws.amazon.com/codebuild/pricing/을 참조하세요.

Asia Pacific (Tokyo)

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
ARM	2	4GiB	64GB	GENERAL	reserved.arm.2cpu.4gib
ARM	4	8GiB	128GB	GENERAL	reserved.arm.4cpu.8gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
ARM	8	16GiB	128GB	GENERAL	reserved.arm.8cpu.16gib
ARM	16	32GiB	256GB	GENERAL	reserved.arm.16cpu.32gib
ARM	32	64GiB	256GB	GENERAL	reserved.arm.32cpu.64gib
ARM	48	96GiB	512GB	GENERAL	reserved.arm.48cpu.96gib
ARM	64	128GiB	824GB	GENERAL	reserved.arm.64cpu.128gib
ARM EC2	2	4GiB	64GB	GENERAL	reserved.arm.2cpu.4gib
ARM EC2	4	8GiB	128GB	GENERAL	reserved.arm.4cpu.8gib
ARM EC2	8	16GiB	128GB	GENERAL	reserved.arm.8cpu.16gib
Linux	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Linux	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux	16	32GiB	256GB	GENERAL	reserved.x86-64.16cpu.32gib
Linux	36	72GiB	256GB	GENERAL	reserved.x86-64.36cpu.72gib
Linux	48	96GiB	512GB	GENERAL	reserved.x86-64.48cpu.96gib
Linux	72	144GiB	824GB	GENERAL	reserved.x86-64.72cpu.144gib
Linux	96	192GiB	824GB	GENERAL	reserved.x86-64.96cpu.192gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Linux	48	96GiB	824 GB(SSD)	NVME	reserved.x86-64.48cpu.96gib.nvme
Linux	72	144GiB	824 GB(SSD)	NVME	reserved.x86-64.72cpu.144gib.nvme
Linux EC2	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux EC2	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux EC2	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux GPU	4	16GiB	235GB(SSD)	NVME	reserved.gpu.4cpu.16gib.nvme
Linux GPU	8	32GiB	435GB(SSD)	NVME	reserved.gpu.8cpu.32gib.nvme

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Linux GPU	48	192GiB	3785GB(SSD)	NVME	reserved.gpu.48cpu.192gib.nvme
Windows	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib
Windows	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Windows	16	32GiB	256GB	GENERAL	reserved.x86-64.16cpu.32gib
Windows	32	64GiB	256GB	GENERAL	reserved.x86-64.32cpu.64gib
Windows	36	72GiB	256GB	GENERAL	reserved.x86-64.36cpu.72gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Windows	48	96GiB	512GB	GENERAL	reserved.x86-64.48cpu.96gib
Windows	64	128GiB	512GB	GENERAL	reserved.x86-64.64cpu.128gib
Windows	72	144GiB	824GB	GENERAL	reserved.x86-64.72cpu.144gib
Windows	96	192GiB	824GB	GENERAL	reserved.x86-64.96cpu.192gib
Windows EC2	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows EC2	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib

요금 식별자에 대한 자세한 내용은 https://aws.amazon.com/codebuild/pricing/을 참조하세요.

Asia Pacific (Malaysia)

환경 유형	vCPU	Memory	시스템 유형	컴퓨팅 인스턴스 유형
ARM	2	4GiB	GENERAL	reserved:arm.g1.small
ARM	4	8GiB	GENERAL	reserved:arm.g1.medium
ARM	8	16GiB	GENERAL	reserved:arm.g1.large
ARM	32	64GiB	GENERAL	reserved:arm.g1.xlarge
ARM	48	96GiB	GENERAL	reserved:arm.g1.2xlarge
Linux	2	4GiB	GENERAL	reserved:x86-64.g1.small
Linux	4	8GiB	GENERAL	reserved:x86-64.g1.medium
Linux	8	16GiB	GENERAL	reserved:x86-64.g1.large

요금 식별자에 대한 자세한 내용은 https://aws.amazon.com/codebuild/pricing/을 참조하세요.

Asia Pacific (New Zealand)

환경 유형	vCPU	Memory	시스템 유형	컴퓨팅 인스턴스 유형
ARM	2	4GiB	GENERAL	reserved:arm.g1.small
ARM	4	8GiB	GENERAL	reserved:arm.g1.medium
ARM	8	16GiB	GENERAL	reserved:arm.g1.large
ARM	32	64GiB	GENERAL	reserved:arm.g1.xlarge
ARM	48	96GiB	GENERAL	reserved:arm.g1.2xlarge
Linux	2	4GiB	GENERAL	reserved:x86-64.g1.small
Linux	4	8GiB	GENERAL	reserved:x86-64.g1.medium
Linux	8	16GiB	GENERAL	reserved:x86-64.g1.large

요금 식별자에 대한 자세한 내용은 https://aws.amazon.com/codebuild/pricing/을 참조하세요.

Asia Pacific (Thailand)

환경 유형	vCPU	Memory	시스템 유형	컴퓨팅 인스턴스 유형
ARM	2	4GiB	GENERAL	reserved:arm.g1.small
ARM	4	8GiB	GENERAL	reserved:arm.g1.medium
ARM	8	16GiB	GENERAL	reserved:arm.g1.large
ARM	32	64GiB	GENERAL	reserved:arm.g1.xlarge
ARM	48	96GiB	GENERAL	reserved:arm.g1.2xlarge
Linux	2	4GiB	GENERAL	reserved:x86-64.g1.small
Linux	4	8GiB	GENERAL	reserved:x86-64.g1.medium
Linux	8	16GiB	GENERAL	reserved:x86-64.g1.large

요금 식별자에 대한 자세한 내용은 https://aws.amazon.com/codebuild/pricing/을 참조하세요.

Asia Pacific (Mumbai)

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
ARM	2	4GiB	64GB	GENERAL	reserved.arm.2cpu.4gib
ARM	4	8GiB	128GB	GENERAL	reserved.arm.4cpu.8gib
ARM	8	16GiB	128GB	GENERAL	reserved.arm.8cpu.16gib
ARM	16	32GiB	256GB	GENERAL	reserved.arm.16cpu.32gib
ARM	32	64GiB	256GB	GENERAL	reserved.arm.32cpu.64gib
ARM	48	96GiB	512GB	GENERAL	reserved.arm.48cpu.96gib
ARM	64	128GiB	824GB	GENERAL	reserved.arm.64cpu.128gib
ARM EC2	2	4GiB	64GB	GENERAL	reserved.arm.2cpu.4gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
ARM EC2	4	8GiB	128GB	GENERAL	reserved.arm.4cpu.8gib
ARM EC2	8	16GiB	128GB	GENERAL	reserved.arm.8cpu.16gib
Linux	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux	16	32GiB	256GB	GENERAL	reserved.x86-64.16cpu.32gib
Linux	36	72GiB	256GB	GENERAL	reserved.x86-64.36cpu.72gib
Linux	48	96GiB	512GB	GENERAL	reserved.x86-64.48cpu.96gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Linux	72	144GiB	824GB	GENERAL	reserved.x86-64.72cpu.144gib
Linux	96	192GiB	824GB	GENERAL	reserved.x86-64.96cpu.192gib
Linux	72	144GiB	824 GB(SSD)	NVME	reserved.x86-64.72cpu.144gib.nvme
Linux EC2	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux EC2	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux EC2	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux GPU	4	16GiB	235GB(SSD)	NVME	reserved.gpu.4cpu.16gib.nvme

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Linux GPU	8	32GiB	435GB(SSD)	NVME	reserved.gpu.8cpu.32gib.nvme
Linux GPU	16	64GiB	585GB(SSD)	NVME	reserved.gpu.16cpu.64gib.nvme
Windows	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib
Windows	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Windows	16	32GiB	256GB	GENERAL	reserved.x86-64.16cpu.32gib
Windows	32	64GiB	256GB	GENERAL	reserved.x86-64.32cpu.64gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Windows	36	72GiB	256GB	GENERAL	reserved.x86-64.36cpu.72gib
Windows	48	96GiB	512GB	GENERAL	reserved.x86-64.48cpu.96gib
Windows	64	128GiB	512GB	GENERAL	reserved.x86-64.64cpu.128gib
Windows	72	144GiB	824GB	GENERAL	reserved.x86-64.72cpu.144gib
Windows	96	192GiB	824GB	GENERAL	reserved.x86-64.96cpu.192gib
Windows EC2	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows EC2	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib

요금 식별자에 대한 자세한 내용은 https://aws.amazon.com/codebuild/pricing/을 참조하세요.

Asia Pacific (Singapore)

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
ARM	2	4GiB	64GB	GENERAL	reserved.arm.2cpu.4gib
ARM	4	8GiB	128GB	GENERAL	reserved.arm.4cpu.8gib
ARM	8	16GiB	128GB	GENERAL	reserved.arm.8cpu.16gib
ARM	16	32GiB	256GB	GENERAL	reserved.arm.16cpu.32gib
ARM	32	64GiB	256GB	GENERAL	reserved.arm.32cpu.64gib
ARM	48	96GiB	512GB	GENERAL	reserved.arm.48cpu.96gib
ARM	64	128GiB	824GB	GENERAL	reserved.arm.64cpu.128gib
ARM EC2	2	4GiB	64GB	GENERAL	reserved.arm.2cpu.4gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
ARM EC2	4	8GiB	128GB	GENERAL	reserved.arm.4cpu.8gib
ARM EC2	8	16GiB	128GB	GENERAL	reserved.arm.8cpu.16gib
Linux	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux	16	32GiB	256GB	GENERAL	reserved.x86-64.16cpu.32gib
Linux	36	72GiB	256GB	GENERAL	reserved.x86-64.36cpu.72gib
Linux	48	96GiB	512GB	GENERAL	reserved.x86-64.48cpu.96gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Linux	72	144GiB	824GB	GENERAL	reserved.x86-64.72cpu.144gib
Linux	96	192GiB	824GB	GENERAL	reserved.x86-64.96cpu.192gib
Linux	48	96GiB	824 GB(SSD)	NVME	reserved.x86-64.48cpu.96gib.nvme
Linux	72	144GiB	824 GB(SSD)	NVME	reserved.x86-64.72cpu.144gib.nvme
Linux EC2	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux EC2	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux EC2	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Windows	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Windows	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Windows	16	32GiB	256GB	GENERAL	reserved.x86-64.16cpu.32gib
Windows	32	64GiB	256GB	GENERAL	reserved.x86-64.32cpu.64gib
Windows	36	72GiB	256GB	GENERAL	reserved.x86-64.36cpu.72gib
Windows	48	96GiB	512GB	GENERAL	reserved.x86-64.48cpu.96gib
Windows	64	128GiB	512GB	GENERAL	reserved.x86-64.64cpu.128gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Windows	72	144GiB	824GB	GENERAL	reserved.x86-64.72cpu.144gib
Windows	96	192GiB	824GB	GENERAL	reserved.x86-64.96cpu.192gib
Windows EC2	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows EC2	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib

요금 식별자에 대한 자세한 내용은 https://aws.amazon.com/codebuild/pricing/을 참조하세요.

Asia Pacific (Sydney)

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
ARM	2	4GiB	64GB	GENERAL	reserved.arm.2cpu.4gib
ARM	4	8GiB	128GB	GENERAL	reserved.arm.4cpu.8gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
ARM	8	16GiB	128GB	GENERAL	reserved.arm.8cpu.16gib
ARM	16	32GiB	256GB	GENERAL	reserved.arm.16cpu.32gib
ARM	32	64GiB	256GB	GENERAL	reserved.arm.32cpu.64gib
ARM	48	96GiB	512GB	GENERAL	reserved.arm.48cpu.96gib
ARM	64	128GiB	824GB	GENERAL	reserved.arm.64cpu.128gib
ARM EC2	2	4GiB	64GB	GENERAL	reserved.arm.2cpu.4gib
ARM EC2	4	8GiB	128GB	GENERAL	reserved.arm.4cpu.8gib
ARM EC2	8	16GiB	128GB	GENERAL	reserved.arm.8cpu.16gib
Linux	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Linux	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux	16	32GiB	256GB	GENERAL	reserved.x86-64.16cpu.32gib
Linux	36	72GiB	256GB	GENERAL	reserved.x86-64.36cpu.72gib
Linux	48	96GiB	512GB	GENERAL	reserved.x86-64.48cpu.96gib
Linux	72	144GiB	824GB	GENERAL	reserved.x86-64.72cpu.144gib
Linux	96	192GiB	824GB	GENERAL	reserved.x86-64.96cpu.192gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Linux	48	96GiB	824 GB(SSD)	NVME	reserved.x86-64.48cpu.96gib.nvme
Linux	72	144GiB	824 GB(SSD)	NVME	reserved.x86-64.72cpu.144gib.nvme
Linux EC2	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux EC2	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux EC2	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux GPU	4	16GiB	235GB(SSD)	NVME	reserved.gpu.4cpu.16gib.nvme
Linux GPU	8	32GiB	435GB(SSD)	NVME	reserved.gpu.8cpu.32gib.nvme

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Linux GPU	16	64GiB	585GB(SSD)	NVME	reserved.gpu.16cpu.64gib.nvme
Linux GPU	48	192GiB	3785GB(SSD)	NVME	reserved.gpu.48cpu.192gib.nvme
macOS	8	24GiB	128GB	GENERAL	reserved.arm.m2.8cpu.24gib
macOS	12	32GiB	256GB	GENERAL	reserved.arm.m2.12cpu.32gib
Windows	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib
Windows	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Windows	16	32GiB	256GB	GENERAL	reserved.x86-64.16cpu.32gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Windows	32	64GiB	256GB	GENERAL	reserved.x86-64.32cpu.64gib
Windows	36	72GiB	256GB	GENERAL	reserved.x86-64.36cpu.72gib
Windows	48	96GiB	512GB	GENERAL	reserved.x86-64.48cpu.96gib
Windows	64	128GiB	512GB	GENERAL	reserved.x86-64.64cpu.128gib
Windows	72	144GiB	824GB	GENERAL	reserved.x86-64.72cpu.144gib
Windows	96	192GiB	824GB	GENERAL	reserved.x86-64.96cpu.192gib
Windows EC2	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Windows EC2	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib

요금 식별자에 대한 자세한 내용은 https://aws.amazon.com/codebuild/pricing/을 참조하세요.

Europe (Frankfurt)

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
ARM	2	4GiB	64GB	GENERAL	reserved.arm.2cpu.4gib
ARM	4	8GiB	128GB	GENERAL	reserved.arm.4cpu.8gib
ARM	8	16GiB	128GB	GENERAL	reserved.arm.8cpu.16gib
ARM	16	32GiB	256GB	GENERAL	reserved.arm.16cpu.32gib
ARM	32	64GiB	256GB	GENERAL	reserved.arm.32cpu.64gib
ARM	48	96GiB	512GB	GENERAL	reserved.arm.48cpu.96gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
ARM	64	128GiB	824GB	GENERAL	reserved.arm.64cpu.128gib
ARM EC2	2	4GiB	64GB	GENERAL	reserved.arm.2cpu.4gib
ARM EC2	4	8GiB	128GB	GENERAL	reserved.arm.4cpu.8gib
ARM EC2	8	16GiB	128GB	GENERAL	reserved.arm.8cpu.16gib
Linux	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux	16	32GiB	256GB	GENERAL	reserved.x86-64.16cpu.32gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Linux	36	72GiB	256GB	GENERAL	reserved.x86-64.36cpu.72gib
Linux	48	96GiB	512GB	GENERAL	reserved.x86-64.48cpu.96gib
Linux	72	144GiB	824GB	GENERAL	reserved.x86-64.72cpu.144gib
Linux	96	192GiB	824GB	GENERAL	reserved.x86-64.96cpu.192gib
Linux	48	96GiB	824 GB(SSD)	NVME	reserved.x86-64.48cpu.96gib.nvme
Linux	72	144GiB	824 GB(SSD)	NVME	reserved.x86-64.72cpu.144gib.nvme
Linux EC2	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Linux EC2	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux EC2	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux GPU	4	16GiB	235GB(SSD)	NVME	reserved.gpu.4cpu.16gib.nvme
Linux GPU	8	32GiB	435GB(SSD)	NVME	reserved.gpu.8cpu.32gib.nvme
Linux GPU	16	64GiB	585GB(SSD)	NVME	reserved.gpu.16cpu.64gib.nvme
Linux GPU	32	128GiB	885GB(SSD)	NVME	reserved.gpu.32cpu.128gib.nvme
Linux GPU	48	192GiB	3785GB(SSD)	NVME	reserved.gpu.48cpu.192gib.nvme

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
macOS	8	24GiB	128GB	GENERAL	reserved.arm.m2.8cpu.24gib
Windows	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib
Windows	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Windows	16	32GiB	256GB	GENERAL	reserved.x86-64.16cpu.32gib
Windows	32	64GiB	256GB	GENERAL	reserved.x86-64.32cpu.64gib
Windows	36	72GiB	256GB	GENERAL	reserved.x86-64.36cpu.72gib
Windows	48	96GiB	512GB	GENERAL	reserved.x86-64.48cpu.96gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Windows	64	128GiB	512GB	GENERAL	reserved.x86-64.64cpu.128gib
Windows	72	144GiB	824GB	GENERAL	reserved.x86-64.72cpu.144gib
Windows	96	192GiB	824GB	GENERAL	reserved.x86-64.96cpu.192gib
Windows EC2	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows EC2	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib

요금 식별자에 대한 자세한 내용은 https://aws.amazon.com/codebuild/pricing/을 참조하세요.

Europe (Ireland)

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
ARM	2	4GiB	64GB	GENERAL	reserved.arm.2cpu.4gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
ARM	4	8GiB	128GB	GENERAL	reserved.arm.4cpu.8gib
ARM	8	16GiB	128GB	GENERAL	reserved.arm.8cpu.16gib
ARM	16	32GiB	256GB	GENERAL	reserved.arm.16cpu.32gib
ARM	32	64GiB	256GB	GENERAL	reserved.arm.32cpu.64gib
ARM	48	96GiB	512GB	GENERAL	reserved.arm.48cpu.96gib
ARM	64	128GiB	824GB	GENERAL	reserved.arm.64cpu.128gib
ARM EC2	2	4GiB	64GB	GENERAL	reserved.arm.2cpu.4gib
ARM EC2	4	8GiB	128GB	GENERAL	reserved.arm.4cpu.8gib
ARM EC2	8	16GiB	128GB	GENERAL	reserved.arm.8cpu.16gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Linux	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux	16	32GiB	256GB	GENERAL	reserved.x86-64.16cpu.32gib
Linux	36	72GiB	256GB	GENERAL	reserved.x86-64.36cpu.72gib
Linux	48	96GiB	512GB	GENERAL	reserved.x86-64.48cpu.96gib
Linux	72	144GiB	824GB	GENERAL	reserved.x86-64.72cpu.144gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Linux	96	192GiB	824GB	GENERAL	reserved.x86-64.96cpu.192gib
Linux	48	96GiB	824 GB(SSD)	NVME	reserved.x86-64.48cpu.96gib.nvme
Linux	72	144GiB	824 GB(SSD)	NVME	reserved.x86-64.72cpu.144gib.nvme
Linux EC2	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux EC2	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux EC2	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux GPU	4	16GiB	235GB(SSD)	NVME	reserved.gpu.4cpu.16gib.nvme

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Linux GPU	8	32GiB	435GB(SSD)	NVME	reserved.gpu.8cpu.32gib.nvme
Linux GPU	16	64GiB	585GB(SSD)	NVME	reserved.gpu.16cpu.64gib.nvme
Linux GPU	32	128GiB	885GB(SSD)	NVME	reserved.gpu.32cpu.128gib.nvme
Linux GPU	48	192GiB	3785GB(SSD)	NVME	reserved.gpu.48cpu.192gib.nvme
Windows	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib
Windows	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Windows	16	32GiB	256GB	GENERAL	reserved.x86-64.16cpu.32gib
Windows	32	64GiB	256GB	GENERAL	reserved.x86-64.32cpu.64gib
Windows	36	72GiB	256GB	GENERAL	reserved.x86-64.36cpu.72gib
Windows	48	96GiB	512GB	GENERAL	reserved.x86-64.48cpu.96gib
Windows	64	128GiB	512GB	GENERAL	reserved.x86-64.64cpu.128gib
Windows	72	144GiB	824GB	GENERAL	reserved.x86-64.72cpu.144gib
Windows	96	192GiB	824GB	GENERAL	reserved.x86-64.96cpu.192gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Windows EC2	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows EC2	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib

요금 식별자에 대한 자세한 내용은 https://aws.amazon.com/codebuild/pricing/을 참조하세요.

South America (São Paulo)

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
ARM	2	4GiB	64GB	GENERAL	reserved.arm.2cpu.4gib
ARM	4	8GiB	128GB	GENERAL	reserved.arm.4cpu.8gib
ARM	8	16GiB	128GB	GENERAL	reserved.arm.8cpu.16gib
ARM	16	32GiB	256GB	GENERAL	reserved.arm.16cpu.32gib
ARM	32	64GiB	256GB	GENERAL	reserved.arm.32cpu.64gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
ARM	48	96GiB	512GB	GENERAL	reserved.arm.48cpu.96gib
ARM EC2	2	4GiB	64GB	GENERAL	reserved.arm.2cpu.4gib
ARM EC2	4	8GiB	128GB	GENERAL	reserved.arm.4cpu.8gib
ARM EC2	8	16GiB	128GB	GENERAL	reserved.arm.8cpu.16gib
Linux	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib
Linux	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Linux	16	32GiB	256GB	GENERAL	reserved.x86-64.16cpu.32gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Linux	36	72GiB	256GB	GENERAL	reserved.x86-64.36cpu.72gib
Linux	48	96GiB	512GB	GENERAL	reserved.x86-64.48cpu.96gib
Linux	72	144GiB	824GB	GENERAL	reserved.x86-64.72cpu.144gib
Linux	96	192GiB	824GB	GENERAL	reserved.x86-64.96cpu.192gib
Linux	48	96GiB	824 GB(SSD)	NVME	reserved.x86-64.48cpu.96gib.nvme
Linux	72	144GiB	824 GB(SSD)	NVME	reserved.x86-64.72cpu.144gib.nvme
Linux EC2	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Linux EC2	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Linux EC2	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Windows	2	4GiB	64GB	GENERAL	reserved.x86-64.2cpu.4gib
Windows	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib
Windows	16	32GiB	256GB	GENERAL	reserved.x86-64.16cpu.32gib
Windows	32	64GiB	256GB	GENERAL	reserved.x86-64.32cpu.64gib
Windows	36	72GiB	256GB	GENERAL	reserved.x86-64.36cpu.72gib

환경 유형	vCPU	Memory	디스크 공간	시스템 유형	컴퓨팅 인스턴스 유형
Windows	48	96GiB	512GB	GENERAL	reserved.x86-64.48cpu.96gib
Windows	64	128GiB	512GB	GENERAL	reserved.x86-64.64cpu.128gib
Windows	72	144GiB	824GB	GENERAL	reserved.x86-64.72cpu.144gib
Windows	96	192GiB	824GB	GENERAL	reserved.x86-64.96cpu.192gib
Windows EC2	4	8GiB	128GB	GENERAL	reserved.x86-64.4cpu.8gib
Windows EC2	8	16GiB	128GB	GENERAL	reserved.x86-64.8cpu.16gib

요금 식별자에 대한 자세한 내용은 https://aws.amazon.com/codebuild/pricing/을 참조하세요.

컴퓨팅 유형을 선택하려면:

- CodeBuild 콘솔의 컴퓨팅 플릿 구성 페이지에서 vCPU, 메모리 및 디스크의 옵션 중 하나를 선택합니다. 자세한 내용은 예약 용량 플릿 생성 단원을 참조하십시오.

- 의 경우 `create-fleet` 또는 `update-fleet` 명령을 AWS CLI실행하여의 값을 `computeType`로 지정합니다`ATTRIBUTE_BASED_COMPUTE`. 자세한 내용은 create-fleet 또는 update-fleet을 참조하세요.
- AWS SDKs의 경우 대상 프로그래밍 언어에 해당하는 `CreateFleet` 또는 `UpdateFleet` 작업을 호출하여의 값을 `computeType`로 지정합니다`ATTRIBUTE_BASED_COMPUTE`. 자세한 내용은 AWS SDKs 및 도구 참조 단원을 참조하십시오.

> **Note**
>
> AWS CLI 및 AWS SDKs의 경우와 같은 `computeType` 입력을 사용하여 대신 `BUILD_GENERAL1_SMALL`컴퓨팅 유형을 선택할 수 있습니다`ATTRIBUTE_BASED_COMPUTE`. 자세한 내용은 온디맨드 환경 유형 정보 단원을 참조하십시오.

지원되는 인스턴스 패밀리

AWS CodeBuild 는 예약된 용량 플릿에 대해 다음 인스턴스를 지원합니다.

- 범용: M5 | M5a | M5ad | M5d | M5dn | M5n | M5zn | M6a | M6g | M6gd | M6i | M6id | M6idn | M6in | M7a | M7g | M7gd | M7i | M7i-flex | M8g | T3 | T3a | T4g
- 컴퓨팅 최적화: C5 | C5a | C5ad | C5d | C5n | C6a | C6g | C6gd | C6gn | C6i | C6id | C6in | C7a | C7g | C7gd | C7gn | C7i | C7i-flex | C8g
- 메모리 최적화: R5 | R5a | R5ad | R5b | R5d | R5dn | R5n | R6a | R6g | R6gd | R6i | R6idn | R6in | R6id | R7a | R7g | R7gd | R7i | R7iz | R8g | U-3tb1 | U-6tb1 | U-9tb1 | U-12tb1 | U-18tb1 | U-24tb1 | U7i-6tb | U7i-8tb | U7i-12tb | U7in-16tb | U7in-24tb | U7in-32tb | X1 | X1e | X2gd | X2idn | X2iedn | X2iezn | X8g | z1d
- 스토리지 최적화: D3 | D3en | I3 | I3en | I4g | I4i | I7ie | I8g | Im4gn | Is4gen
- 가속 컴퓨팅: DL1 | DL2q | F1 | F2 | G4ad | G4dn | G5 | G5g | G6 | G6e | Gr6 | Inf1 | Inf2 | P3 | P3dn | P4d | P5 | P5e | P5en | Trn1 | Trn1n | Trn2 | VT1
- 고성능 컴퓨팅: Hpc6a | Hpc6id | Hpc7a | Hpc7g
- 이전 세대: A1

특정 인스턴스 유형으로 예약 용량 플릿을 생성하려면 다음 단계를 따르세요.

- CodeBuild 콘솔의 컴퓨팅 플릿 구성 페이지에서 용량 구성 섹션으로 이동합니다. 컴퓨팅 선택 모드에서 수동 입력을 선택하고 컴퓨팅 인스턴스 유형의 드롭다운 메뉴에서 인스턴스 유형 중 하나를 선택합니다. 자세한 내용은 예약 용량 플릿 생성 단원을 참조하십시오.
- 의 경우 `create-fleet` 또는 `update-fleet` 명령을 AWS CLI실행하여의 값을 `computeType`로 지정`CUSTOM_INSTANCE_TYPE`하고의 값을 지정된 인스턴스 유형`ComputeConfigurationinstanceType`으로 지정합니다. 자세한 내용은 create-fleet 또는 update-fleet을 참조하세요.
- AWS SDKs의 경우 대상 프로그래밍 언어에 해당하는 `CreateFleet` 또는 `UpdateFleet` 작업을 호출하여의 값을 `computeType`로 지정`CUSTOM_INSTANCE_TYPE`하고 ComputeConfiguration을 지정된 인스턴스 유형`instanceType`으로 지정합니다. 자세한 내용은 AWS SDKs 및 도구 참조 단원을 참조하십시오.

온디맨드 환경 유형 정보

AWS CodeBuild 는 EC2 컴퓨팅 모드에 사용할 수 있는 다음과 같은 메모리, vCPUs 및 디스크 공간이 있는 빌드 환경을 제공합니다.

컴퓨팅 유형	환경 computeType 값	환경 유형 값	Memory	vCPU	디스크 공간
ARM Small [1]	`BUILD_GENERAL1_SMALL`	`ARM_CONTAINER` `ARM_EC2`	4GiB	2	64GB
ARM Medium [1]	`BUILD_GENERAL1_MEDIUM`	`ARM_CONTAINER` `ARM_EC2`	8GiB	4	128GB
ARM Large [1]	`BUILD_GENERAL1_LARGE`	`ARM_CONTAINER` `ARM_EC2`	16GiB	8	128GB

컴퓨팅 유형	환경 computeType 값	환경 유형 값	Memory	vCPU	디스크 공간
ARM XLarge [1]	BUILD_GENERAL1_XLARGE	ARM_CONTAINER	64GiB	32	256GB
ARM 2XLarge [1]	BUILD_GENERAL1_2XLARGE	ARM_CONTAINER	96GiB	48	824GB
Linux Small [1]	BUILD_GENERAL1_SMALL	LINUX_CONTAINER LINUX_EC2	4GiB	2	64GB
Linux Medium [1]	BUILD_GENERAL1_MEDIUM	LINUX_CONTAINER LINUX_EC2	8GiB	4	128GB
Linux Large [1]	BUILD_GENERAL1_LARGE	LINUX_CONTAINER LINUX_EC2	16GiB	8	128GB
Linux XLarge [1]	BUILD_GENERAL1_XLARGE	LINUX_CONTAINER	72GiB	36	256GB
Linux 2XLarge	BUILD_GENERAL1_2XLARGE	LINUX_CONTAINER	144GiB	72	824 GB(SSD)
Linux GPU Sma	BUILD_GENERAL1_SMALL	LINUX_GPU_CONTAINER	16GiB	4	235GB(SSD)

컴퓨팅 유형	환경 computeType 값	환경 유형 값	Memory	vCPU	디스크 공간
Linux GPU Larg	BUILD_GENERAL1_LARGE	LINUX_GPU_CONTAINER	255GiB	32	50GB
Windows Mediu [1]	BUILD_GENERAL1_MEDIUM	WINDOWS_SERVER_2019_CONTAINER WINDOWS_SERVER_2022_CONTAINER WINDOWS_EC2	8GiB	4	128GB
Windows Large [1]	BUILD_GENERAL1_LARGE	WINDOWS_SERVER_2019_CONTAINER WINDOWS_SERVER_2022_CONTAINER WINDOWS_EC2	16GiB	8	128GB
Windows XLarg [1]	BUILD_GENERAL1_XLARGE	WINDOWS_SERVER_2022_CONTAINER	72GiB	36	256GB

컴퓨팅 유형	환경 computeType 값	환경 유형 값	Memory	vCPU	디스크 공간
Windows 2XLar [1]	BUILD_GENERAL1_2XLARGE	WINDOWS_SERVER_2022_CONTAINER	144GiB	72	824GB

[1] 이 이미지 유형의 최신 버전이 캐시됩니다. 보다 구체적인 버전을 지정하면 CodeBuild는 캐시된 버전 대신 해당 버전을 프로비저닝합니다. 이로 인해 빌드 시간이 길어질 수 있습니다. 예를 들어 캐싱을 사용하려면 aws/codebuild/amazonlinux-x86_64-standard:5.0-1.0.0과 같이 보다 세분화된 버전 대신 aws/codebuild/amazonlinux-x86_64-standard:5.0을 지정합니다.

AWS CodeBuild 는 AWS Lambda 컴퓨팅 모드에 사용할 수 있는 다음과 같은 메모리 및 디스크 공간을 빌드 환경에 제공합니다.

컴퓨팅 유형	환경 computeType 값	환경 유형 값	Memory	디스크 공간
ARM Lambda 1GB	BUILD_LAMBDA_1GB	ARM_LAMBDA_CONTAINER	1GiB	10GB
ARM Lambda 2GB	BUILD_LAMBDA_2GB	ARM_LAMBDA_CONTAINER	2GiB	10GB
ARM Lambda 4GB	BUILD_LAMBDA_4GB	ARM_LAMBDA_CONTAINER	4GiB	10GB
ARM Lambda 8GB	BUILD_LAMBDA_8GB	ARM_LAMBDA_CONTAINER	8GiB	10GB

컴퓨팅 유형	환경 computeType 값	환경 유형 값	Memory	디스크 공간
ARM Lambda 10GI	BUILD_LAMBDA_10GB	ARM_LAMBDA_CONTAINER	10GiB	10GB
Linux Lambda 1GB	BUILD_LAMBDA_1GB	LINUX_LAMBDA_CONTAINER	1GiB	10GB
Linux Lambda 2GB	BUILD_LAMBDA_2GB	LINUX_LAMBDA_CONTAINER	2GiB	10GB
Linux Lambda 4GB	BUILD_LAMBDA_4GB	LINUX_LAMBDA_CONTAINER	4GiB	10GB
Linux Lambda 8GB	BUILD_LAMBDA_8GB	LINUX_LAMBDA_CONTAINER	8GiB	10GB
Linux Lambda 10G	BUILD_LAMBDA_10GB	LINUX_LAMBDA_CONTAINER	10GiB	10GB

다른 환경 유형을 사용할 때는 캐시된 이미지를 사용하여 빌드 시간을 줄이는 것이 좋습니다.

각 빌드 환경에 대해 나열된 디스크 공간은 CODEBUILD_SRC_DIR 환경 변수로 지정된 디렉터리에서만 사용할 수 있습니다.

컴퓨팅 유형을 선택하려면:

- CodeBuild 콘솔에 있는 빌드 프로젝트 생성 마법사 또는 빌드 프로젝트 편집 페이지의 환경에서 추가 구성을 확장한 후 컴퓨팅 유형 옵션 중 하나를 선택합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 또는 빌드 프로젝트 설정 변경(콘솔)을 참조하세요.

- 의 경우 `environment` 객체 `computeType` 값을 지정하여 `create-project` 또는 `update-project` 명령을 AWS CLI실행합니다. 자세한 내용은 빌드 프로젝트 생성(AWS CLI) 또는 빌드 프로젝트 설정 변경(AWS CLI)을 참조하세요.
- AWS SDKs의 경우 대상 프로그래밍 언어에 해당하는 `CreateProject` 또는 `UpdateProject` 작업을 호출하여 `environment` 객체의 `computeType` 값에 해당하는 값을 지정합니다. 자세한 내용은 AWS SDKs 및 도구 참조 단원을 참조하십시오.

일부 환경 및 컴퓨팅 유형에는 다음과 같은 리전 가용성 제한이 있습니다.

- 컴퓨팅 유형 Linux GPU Small(`LINUX_GPU_CONTAINER`)은 다음 리전에서만 사용할 수 있습니다.
  - 미국 동부(버지니아 북부)
  - 미국 서부(오레곤)
  - 아시아 태평양(도쿄)
  - 캐나다(중부)
  - 유럽(프랑크푸르트)
  - 유럽(아일랜드)
  - 유럽(런던)
- 컴퓨팅 유형 Linux GPU Large(`LINUX_GPU_CONTAINER`)는 다음 리전에서만 사용할 수 있습니다.
  - 미국 동부(오하이오)
  - 미국 동부(버지니아 북부)
  - 미국 서부(오레곤)
  - 아시아 태평양(서울)
  - 아시아 태평양(시드니)
  - 아시아 태평양(도쿄)
  - 캐나다(중부)
  - 중국(베이징)
  - 중국(닝샤)
  - 유럽(프랑크푸르트)
  - 유럽(아일랜드)
  - 유럽(런던)
- 컴퓨팅 유형 `BUILD_GENERAL1_2XLARGE`는 다음 리전에서만 사용할 수 있습니다.
  - 미국 동부(오하이오)

- 미국 동부(버지니아 북부)
- 미국 서부(캘리포니아 북부)
- 미국 서부(오리건)
- 아시아 태평양(하이데라바드)
- 아시아 태평양(홍콩)
- 아시아 태평양(자카르타)
- 아시아 태평양(멜버른)
- 아시아 태평양(뭄바이)
- 아시아 태평양(서울)
- 아시아 태평양(싱가포르)
- 아시아 태평양(시드니)
- 아시아 태평양(도쿄)
- 캐나다(중부)
- 중국(베이징)
- 중국(닝샤)
- 유럽(프랑크푸르트)
- 유럽(아일랜드)
- 유럽(런던)
- 유럽(파리)
- 유럽(스페인)
- 유럽(스톡홀름)
- 유럽(취리히)
- 이스라엘(텔아비브)
- Middle East (Bahrain)
- 중동(UAE)
- 남아메리카(상파울루)

- 환경 유형 `ARM_CONTAINER`는 다음 리전에서만 사용할 수 있습니다.
  - 미국 동부(오하이오)
  - 미국 동부(버지니아 북부)
  - 미국 서부(캘리포니아 북부)

- 미국 서부(오리건)
- 아시아 태평양(홍콩)
- 아시아 태평양(자카르타)
- 아시아 태평양(하이데라바드)
- 아시아 태평양(말레이시아)
- 아시아 태평양(뭄바이)
- 아시아 태평양(뉴질랜드)
- 아시아 태평양(오사카)
- 아시아 태평양(서울)
- 아시아 태평양(싱가포르)
- 아시아 태평양(시드니)
- 아시아 태평양(태국)
- 아시아 태평양(도쿄)
- 캐나다(중부)
- 중국(베이징)
- 중국(닝샤)
- 유럽(프랑크푸르트)
- 유럽(아일랜드)
- 유럽(런던)
- 유럽(밀라노)
- 유럽(파리)
- 유럽(스페인)
- 유럽(스톡홀름)
- 이스라엘(텔아비브)
- Middle East (Bahrain)
- 중동(UAE)
- 남아메리카(상파울루)

- 환경 유형 `WINDOWS_SERVER_2022_CONTAINER`는 다음 리전에서만 사용할 수 있습니다.
  - 미국 동부(오하이오)
  - 미국 동부(버지니아 북부)

- 미국 서부(오레곤)
- 아시아 태평양(시드니)
- 아시아 태평양(도쿄)
- 유럽(프랑크푸르트)
- 유럽(아일랜드)
- 남아메리카(상파울루)

- 환경 유형 LINUX_EC2(BUILD_GENERAL1_SMALL, BUILD_GENERAL1_MEDIUM, BUILD_GENERAL1_LARGE)는 다음 리전에서만 사용할 수 있습니다.
  - 미국 동부(오하이오)
  - 미국 동부(버지니아 북부)
  - 미국 서부(캘리포니아 북부)
  - 미국 서부(오리건)
  - 아프리카(케이프타운)
  - 아시아 태평양(홍콩)
  - 아시아 태평양(자카르타)
  - 아시아 태평양(멜버른)
  - 아시아 태평양(하이데라바드)
  - 아시아 태평양(말레이시아)
  - 아시아 태평양(뭄바이)
  - 아시아 태평양(뉴질랜드)
  - 아시아 태평양(오사카)
  - 아시아 태평양(서울)
  - 아시아 태평양(싱가포르)
  - 아시아 태평양(시드니)
  - 아시아 태평양(태국)
  - 아시아 태평양(도쿄)
  - 캐나다(중부)
  - 중국(베이징)
  - 중국(닝샤)
  - 유럽(프랑크푸르트)

- 유럽(아일랜드)
- 유럽(런던)
- 유럽(밀라노)
- 유럽(파리)
- 유럽(스페인)
- 유럽(스톡홀름)
- 유럽(취리히)
- 이스라엘(텔아비브)
- Middle East (Bahrain)
- 중동(UAE)
- 남아메리카(상파울루)
- AWS GovCloud (US-West)
- AWS GovCloud (US-East)

- 환경 유형 `ARM_EC2`(`BUILD_GENERAL1_SMALL`, `BUILD_GENERAL1_MEDIUM`, `BUILD_GENERAL1_LARGE`)는 다음 리전에서만 사용할 수 있습니다.
  - 미국 동부(오하이오)
  - 미국 동부(버지니아 북부)
  - 미국 서부(캘리포니아 북부)
  - 미국 서부(오리건)
  - 아시아 태평양(홍콩)
  - 아시아 태평양(자카르타)
  - 아시아 태평양(하이데라바드)
  - 아시아 태평양(말레이시아)
  - 아시아 태평양(뭄바이)
  - 아시아 태평양(뉴질랜드)
  - 아시아 태평양(오사카)
  - 아시아 태평양(서울)
  - 아시아 태평양(태국)
  - 아시아 태평양(싱가포르)
  - 아시아 태평양(시드니)

- 아시아 태평양(도쿄)
- 캐나다(중부)
- 중국(베이징)
- 중국(닝샤)
- 유럽(프랑크푸르트)
- 유럽(아일랜드)
- 유럽(런던)
- 유럽(밀라노)
- 유럽(파리)
- 유럽(스페인)
- 유럽(스톡홀름)
- 유럽(취리히)
- 이스라엘(텔아비브)
- Middle East (Bahrain)
- 남아메리카(상파울루)
- AWS GovCloud (US-West)
- AWS GovCloud (US-East)

- 환경 유형 `WINDOWS_EC2`(`BUILD_GENERAL1_MEDIUM`, `BUILD_GENERAL1_LARGE`)는 다음 리전에서만 사용할 수 있습니다.
  - 미국 동부(오하이오)
  - 미국 동부(버지니아 북부)
  - 미국 서부(오레곤)
  - 아시아 태평양(시드니)
  - 아시아 태평양(도쿄)
  - 유럽(프랑크푸르트)
  - 유럽(아일랜드)
  - 남아메리카(상파울루)
- 컴퓨팅 모드 AWS Lambda (`ARM_LAMBDA_CONTAINER` 및 `LINUX_LAMBDA_CONTAINER`)는 다음 리전에서만 사용할 수 있습니다.
  - 미국 동부(버지니아 북부)
  - 미국 동부(오하이오)

- 미국 서부(캘리포니아 북부)
- 미국 서부(오리건)
- 아시아 태평양(뭄바이)
- 아시아 태평양(싱가포르)
- 아시아 태평양(시드니)
- 아시아 태평양(도쿄)
- 유럽(프랑크푸르트)
- 유럽(아일랜드)
- 남아메리카(상파울루)

- 컴퓨팅 모드 `MAC_ARM`은 다음 리전에서만 사용할 수 있습니다.
  - 미국 동부(버지니아 북부)
  - 미국 동부(오하이오)
  - 미국 서부(오리건)
  - 아시아 태평양(시드니)
  - 유럽(프랑크푸르트)

컴퓨팅 유형 `BUILD_GENERAL1_2XLARGE`은 압축되지 않은 최대 100GB의 도커 이미지가 지원됩니다.

> **Note**
>
> 사용자 지정 빌드 환경 이미지의 경우 CodeBuild는 컴퓨팅 유형에 관계없이 Linux 및 Windows에서 최대 50GB의 도커 이미지를 지원합니다. 빌드 이미지의 크기를 확인하려면 Docker를 사용하여 `docker images` *`REPOSITORY`*`:`*`TAG`* 명령을 실행합니다.

Amazon EFS를 사용하여 빌드 컨테이너의 더 많은 공간에 액세스할 수 있습니다. 자세한 내용은 용 Amazon Elastic File System 샘플 AWS CodeBuild 단원을 참조하십시오. 빌드 중 컨테이너 디스크 공간을 조정하려면 권한을 가진 모드에서 빌드를 실행해야 합니다.

> **Note**
>
> 기본적으로 비 VPC 빌드에는 Docker 데몬이 활성화됩니다. VPC 빌드에 Docker 컨테이너를 사용하려면 Docker Docs 웹 사이트의 런타임 권한 및 Linux 기능을 참조하고 권한 부여 모드를 활성화합니다. 또한 Windows는 권한 모드를 지원하지 않습니다.

빌드 환경의 쉘 및 명령

빌드 수명 주기 동안 빌드 환경에서 AWS CodeBuild 실행할에 대한 명령 세트를 제공합니다(예: 빌드 종속성 설치, 소스 코드 테스트 및 컴파일). 이러한 명령을 지정하는 데에는 다음과 같은 몇 가지 방법이 있습니다.

- 빌드 사양 파일을 만들어 소스 코드에 포함합니다. 이 파일에서 빌드 수명 주기의 각 단계에서 실행할 명령을 지정합니다. 자세한 내용은 CodeBuild의 빌드 사양 참조 단원을 참조하십시오.
- CodeBuild 콘솔을 사용하여 빌드 프로젝트를 생성합니다. 빌드 명령 삽입에서 빌드 명령에 `build` 단계에서 실행하려는 명령을 입력합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 단원을 참조하십시오.
- CodeBuild 콘솔을 사용하여 빌드 프로젝트 설정을 변경합니다. 빌드 명령 삽입에서 빌드 명령에 `build` 단계에서 실행하려는 명령을 입력합니다. 자세한 내용은 빌드 프로젝트 설정 변경(콘솔) 단원을 참조하십시오.
- AWS CLI 또는 AWS SDKs를 사용하여 빌드 프로젝트를 생성하거나 빌드 프로젝트의 설정을 변경합니다. 명령을 사용하여 빌드 사양 파일이 들어 있는 소스 코드를 참조하거나, 빌드 사양 파일에 해당하는 파일의 내용이 들어 있는 단일 문자열을 지정합니다. 자세한 내용은 빌드 프로젝트 생성 또는 빌드 프로젝트 설정 변경을 참조하세요.
- AWS CLI 또는 AWS SDKs 사용하여 buildspec 파일 또는 동등한 buildspec 파일의 내용이 포함된 단일 문자열을 지정하여 빌드를 시작합니다. 자세한 정보는 빌드를 수동으로 실행의 `buildspecOverride` 값에 대한 설명을 참조하십시오.

Shell 명령 언어(sh) 명령을 지정할 수 있습니다. 빌드 사양 버전 0.1에서 CodeBuild는 빌드 환경에 있는 서로 다른 인스턴스에서 각 Shell 명령을 실행합니다. 즉, 각 명령이 다른 모든 명령과 독립적으로 실행됩니다. 따라서 기본적으로 이전 명령의 상태에 따라 실행되는 단일 명령을 실행할 수 없습니다(예: 디렉터리 변경 또는 환경 변수 설정). 이 제한 사항을 해결하려면 이 문제를 해결하는 버전 0.2를 사용하는 것이 좋습니다. 버전 0.1을 사용해야 하는 경우 다음 접근 방식을 따르는 것이 좋습니다.

- 기본 쉘의 단일 인스턴스에서 실행하려는 명령이 들어 있는 쉘 스크립트를 소스 코드에 포함합니다. 예를 들어, `my-script.sh`와 같은 명령이 들어 있는 `cd MyDir; mkdir -p mySubDir; cd mySubDir; pwd;`라는 파일을 소스 코드에 포함할 수 있습니다. 그런 다음 빌드 사양 파일에서 `./my-script.sh` 명령을 지정합니다.
- buildspec 파일 또는 해당 `build` 단계의 빌드 명령 설정에서만 기본 쉘의 단일 인스턴스에서 실행하려는 모든 명령이 포함된 단일 명령을 입력합니다(예: `cd MyDir && mkdir -p mySubDir && cd mySubDir && pwd`).

CodeBuild에서 오류가 발생하는 경우 기본 쉘의 자체 인스턴스에서 단일 명령을 실행하는 것보다 오류를 해결하기가 더 어려울 수 있습니다.

Windows Server Core 이미지에서 실행되는 명령은 Powershell 쉘을 사용합니다.

빌드 환경의 환경 변수

AWS CodeBuild 는 빌드 명령에 사용할 수 있는 여러 환경 변수를 제공합니다.

AWS_DEFAULT_REGION

빌드가 실행 중인 AWS 리전(예: `us-east-1`). 이 환경 변수는 AWS CLI에 의해 주로 사용됩니다.

AWS_REGION

빌드가 실행 중인 AWS 리전(예: `us-east-1`). 이 환경 변수는 주로 AWS SDKs에서 사용됩니다.

코드빌드_배치_빌드_식별자

배치 빌드의 빌드 식별자입니다. 이는 배치 buildspec에 지정되어 있습니다. 자세한 내용은 the section called “배치 buildspec 참조” 단원을 참조하십시오.

CODEBUILD_BUILD_ARN

빌드의 Amazon 리소스 이름(ARN)입니다(예: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:build/codebuild-demo-project:b1e6661e-e4f2-4156-9ab9-82a19EXAMPLE`).

CODEBUILD_BUILD_ID

빌드의 CodeBuild ID입니다(예: `codebuild-demo-project:b1e6661e-e4f2-4156-9ab9-82a19EXAMPLE`).

CODEBUILD_BUILD_IMAGE

CodeBuild 빌드 이미지 식별자입니다(예: `aws/codebuild/standard:2.0`).

CODEBUILD_BUILD_NUMBER

프로젝트의 현재 빌드 번호입니다.

CODEBUILD_BUILD_SUCCEEDING

현재 빌드가 성공적으로 진행되는지 여부입니다. 빌드가 실패하는 경우 `0`으로 설정하고, 성공하는 경우 `1`로 설정합니다.

CODEBUILD_INITIATOR

빌드를 시작한 엔터티입니다. CodePipeline이 빌드를 시작한 경우 파이프라인의 이름입니다(예: `codepipeline/my-demo-pipeline`). 사용자가 빌드를 시작한 경우 사용자의 이름입니다(예: `MyUserName`). CodeBuild용 Jenkins 플러그인이 빌드를 시작했으면 문자열 `CodeBuild-Jenkins-Plugin`입니다.

CODEBUILD_KMS_KEY_ID

CodeBuild가 빌드 출력 아티팩트를 암호화하는 데 사용하는 AWS KMS 키의 식별자입니다(예: `arn:aws:kms:`*`region-ID`*`:`*`account-ID`*`:key/`*`key-ID`* 또는 `alias/`*`key-alias`*).

CODEBUILD_PROJECT_ARN

프로젝트의 Amazon 리소스 이름(ARN)입니다(예: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:project/`*`project-name`*).

CODEBUILD_PUBLIC_BUILD_URL

퍼블릭 빌드 웹사이트에 있는 이 빌드의 빌드 결과 URL입니다. 이 변수는 빌드 프로젝트에 퍼블릭 빌드가 활성화된 경우에만 설정됩니다. 자세한 내용은 단원을 참조하십시오퍼블릭 빌드 프로젝트 URL 가져오기.

CODEBUILD_RESOLVED_SOURCE_VERSION

빌드 소스 코드의 버전 식별자입니다. 내용은 다음 소스 코드 리포지토리에 따라 달라집니다.

CodeCommit, GitHub, GitHub Enterprise Server 및 Bitbucket

이 변수에는 커밋 ID가 포함됩니다.

CodePipeline

이 변수에는 CodePipeline에서 제공하는 소스 수정 버전이 포함되어 있습니다.

CodePipeline이 소스 수정 사항을 확인할 수 없는 경우(예: 소스가 버전 관리가 활성화되지 않은 Amazon S3 버킷인 경우), 이 환경 변수는 설정되지 않습니다.

Amazon S3

이 변수는 설정되지 않습니다.

해당하는 경우 CODEBUILD_RESOLVED_SOURCE_VERSION 변수는 DOWNLOAD_SOURCE 단계 이후에만 사용할 수 있습니다.

CODEBUILD_SOURCE_REPO_URL

입력 아티팩트 또는 소스 코드 리포지토리에 대한 URL입니다. Amazon S3의 경우 s3:// 뒤에 버킷 이름과 입력 아티팩트에 대한 경로가 옵니다. CodeCommit과 GitHub의 경우 리포지토리의 복제 URL입니다. CodePipeline에서 빌드를 시작한 경우 이 환경 변수는 비어 있을 수 있습니다.

보조 소스의 경우 보조 소스 리포지토리 URL의 환경 변수는 CODEBUILD_SOURCE_REPO_URL_*<sourceIdentifier>*입니다. 여기서 *<sourceIdentifier>*는 사용자가 생성한 소스 식별자입니다.

CODEBUILD_SOURCE_VERSION

값의 형식은 소스 코드 리포지토리에 따라 다릅니다.

- Amazon S3의 경우 입력 아티팩트에 연결된 버전 ID입니다.
- CodeCommit의 경우, 커밋 ID 또는 빌드할 소스 코드 버전과 연관된 분기 이름입니다.
- GitHub, GitHub Enterprise Server, Bitbucket의 경우, 커밋 ID, 분기 이름 또는 빌드할 소스 코드 버전과 연관된 태그 이름입니다.

> **Note**
>
> Webhook pull 요청 이벤트에서 트리거하는 GitHub 또는 GitHub Enterprise Server 빌드의 경우 pr/*pull-request-number*입니다.

보조 소스의 경우 보조 소스 버전의 환경 변수는 CODEBUILD_SOURCE_VERSION_*<sourceIdentifier>*입니다. 여기서 *<sourceIdentifier>*는 사용자가 생성한 소스 식별자입니다. 자세한 내용은 다중 입력 소스 및 출력 아티팩트 샘플 단원을 참조하십시오.

CODEBUILD_SRC_DIR

CodeBuild가 빌드에 사용하는 디렉터리 경로입니다(예: /tmp/src123456789/src).

보조 소스를 사용하는 경우 보조 소스 디렉터리 경로의 환경 변수는 CODEBUILD_SRC_DIR_*<sourceIdentifier>*입니다. 여기서 *<sourceIdentifier>*는 사용

자가 생성한 소스 식별자입니다. 자세한 내용은 다중 입력 소스 및 출력 아티팩트 샘플 단원을 참조하십시오.

CODEBUILD_START_TIME

밀리초 단위의 Unix 타임스탬프로 지정된 빌드의 시작 시간입니다.

CODEBUILD_WEBHOOK_ACTOR_ACCOUNT_ID

Webhook 이벤트를 트리거한 사용자의 계정 ID입니다.

CODEBUILD_WEBHOOK_BASE_REF

현재 빌드를 트리거하는 Webhook 이벤트의 기본 참조 이름입니다. pull 요청의 경우 이를 브랜치 참조라고 합니다.

CODEBUILD_WEBHOOK_EVENT

현재 빌드를 트리거하는 Webhook 이벤트입니다.

CODEBUILD_WEBHOOK_MERGE_COMMIT

빌드에 사용된 병합 커밋의 식별자입니다. 이 변수는 Bitbucket 풀 요청이 스쿼시 전략과 병합되고 pull 요청 분기가 닫힐 때 설정됩니다. 이 경우 원래의 풀 요청 커밋은 더 이상 존재하지 않으므로 이 환경 변수에는 스쿼시된 병합 커밋의 식별자가 포함됩니다.

CODEBUILD_WEBHOOK_PREV_COMMIT

현재 빌드를 트리거하는 webhook 푸시 이벤트 전 최신 커밋의 ID입니다.

CODEBUILD_WEBHOOK_HEAD_REF

현재 빌드를 트리거하는 Webhook 이벤트의 헤드 참조 이름입니다. 브랜치 참조 또는 태그 참조일 수 있습니다.

CODEBUILD_WEBHOOK_TRIGGER

빌드를 트리거하는 Webhook 이벤트를 표시합니다. 이 변수는 Webhook가 트리거하는 빌드에만 사용할 수 있습니다. 이 값은 GitHub, GitHub Enterprise Server 또는 Bitbucket이 CodeBuild로 전송하는 페이로드에서 구문 분석됩니다. 값의 형식은 빌드를 트리거한 이벤트 유형에 따라 다릅니다.

- pull 요청이 트리거한 빌드의 경우 `pr/`*`pull-request-number`*입니다.
- 새 브랜치를 생성하거나 브랜치로 커밋을 푸시하여 트리거된 빌드의 경우 `branch/`*`branch-name`*입니다.
- 리포지토리로 태그를 푸시하여 트리거된 빌드의 경우 `tag/`*`tag-name`*입니다.

HOME

이 환경 변수는 항상 `/root`로 설정되어 있습니다.

AWS CodeBuild 는 자체 호스팅 러너 빌드에 대한 환경 변수 집합도 지원합니다. CodeBuild 자체 호스팅 실행기에 대한 자세한 내용은 자습서: CodeBuild 호스팅 GitHub Action 실행기 구성 섹션을 참조하세요.

CODEBUILD_RUNNER_OWNER

자체 호스팅 실행기 빌드를 트리거하는 리포지토리의 소유자입니다.

CODEBUILD_RUNNER_REPO

자체 호스팅 실행기 빌드를 트리거하는 리포지토리의 이름입니다.

CODEBUILD_RUNNER_REPO_DOMAIN

자체 호스팅 실행기 빌드를 트리거하는 리포지토리의 도메인입니다. 지정된 GitHub Enterprise 빌드만 해당됩니다.

CODEBUILD_WEBHOOK_LABEL

빌드 재정의를 구성하는 데 사용되는 레이블과 빌드 중에 자체 호스팅된 실행기입니다.

CODEBUILD_WEBHOOK_RUN_ID

빌드와 연결된 워크플로의 실행 ID입니다.

CODEBUILD_WEBHOOK_JOB_ID

빌드와 연결된 작업의 작업 ID입니다.

CODEBUILD_WEBHOOK_WORKFLOW_NAME

웹후크 요청 페이로드에 빌드가 있는 경우 빌드와 연결된 워크플로의 이름입니다.

CODEBUILD_RUNNER_WITH_BUILDSPEC

자체 호스팅 실행기 요청 레이블에 buildspec 재정의가 구성된 경우 이 재정의는 `true`로 설정됩니다.

자체 환경 변수를 사용하여 빌드 환경을 제공할 수도 있습니다. 자세한 내용은 다음 항목을 참조하세요.

- CodePipeline에서 CodeBuild 사용

- 빌드 프로젝트 생성
- 빌드 프로젝트 설정 변경
- 빌드를 수동으로 실행
- buildspec 참조

빌드 환경에서 사용 가능한 모든 환경 변수를 나열하려면 빌드 중에 `printenv` (Linux 기반 빌드 환경의 경우) 또는 `"Get-ChildItem Env:"` (Windows 기반 빌드 환경의 경우) 명령을 실행하면 됩니다. 앞에서 나열한 환경 변수를 제외하고, `CODEBUILD_`로 시작하는 환경 변수는 CodeBuild에서 내부적으로 사용합니다. 이러한 환경 변수는 빌드 명령에서 사용하면 안 됩니다.

> **Important**
>
> 환경 변수를 사용하여 민감한 값, 특히 AWS 액세스 키 IDs를 저장하지 않는 것이 좋습니다. 환경 변수는 CodeBuild 콘솔 및 AWS CLI와 같은 도구를 사용하여 일반 텍스트로 표시할 수 있습니다.
>
> 중요한 값을 Amazon EC2 Systems Manager Parameter Store에 저장한 후에 buildspec에서 검색하는 것이 좋습니다. 중요한 값을 저장하려면 Amazon EC2 Systems Manager 사용 설명서의 Systems Manager Parameter Store 및 안내: 문자열 파라미터 생성 및 테스트(콘솔)를 참조하세요. 검색하려면 buildspec 구문의 `parameter-store` 매핑을 참조하십시오.

CODEBUILD_BUILD_URL

이 빌드에 대한 빌드 결과의 URL입니다.

빌드 환경의 배경 작업

빌드 환경에서 배경 작업을 실행할 수 있습니다. 이렇게 하려면 buildspec에서 빌드 프로세스가 쉘을 종료하더라도 nohup 명령을 사용하여 명령을 배경 작업으로서 실행합니다. disown 명령을 사용하여 실행 중인 배경 작업을 강제로 중단합니다.

예:

- 배경 프로세스를 시작하고 나중에 완료될 때까지 기다립니다.

```
|
nohup sleep 30 & echo $! > pidfile
```

```
…
wait $(cat pidfile)
```

- 배경 프로세스를 시작하고 완료될 때까지 기다리지 않습니다.

```
|
nohup sleep 30 & disown $!
```

- 배경 프로세스를 시작하고 나중에 종료합니다.

```
|
nohup sleep 30 & echo $! > pidfile
…
kill $(cat pidfile)
```

빌드 프로젝트

이 빌드 프로젝트에는 소스 코드를 가져올 위치, 사용할 빌드 환경, 실행할 빌드 명령 및 빌드 출력을 저장할 위치를 비롯하여 빌드 실행 방법에 대한 정보가 포함되어 있습니다.

빌드 프로젝트 작업을 수행할 때 이러한 태스크를 수행할 수 있습니다.

주제

- 에서 빌드 프로젝트 생성 AWS CodeBuild
- 알림 규칙 생성
- 에서 빌드 프로젝트 설정 변경 AWS CodeBuild
- CodeBuild의 다중 액세스 토큰
- 에서 빌드 프로젝트 삭제 AWS CodeBuild
- 퍼블릭 빌드 프로젝트 URL 가져오기
- 빌드 프로젝트 공유
- 빌드 프로젝트 태그 지정
- 에서 실행기 사용 AWS CodeBuild
- 에서 웹후크 사용 AWS CodeBuild
- 에서 빌드 프로젝트의 세부 정보 보기 AWS CodeBuild
- 에서 빌드 프로젝트 이름 보기 AWS CodeBuild

에서 빌드 프로젝트 생성 AWS CodeBuild

AWS CodeBuild 콘솔 AWS CLI, 또는 AWS SDKs 사용하여 빌드 프로젝트를 생성할 수 있습니다.

주제

- 사전 조건
- 빌드 프로젝트 만들기(콘솔)
- 빌드 프로젝트 생성(AWS CLI)
- 빌드 프로젝트(AWS SDKs) 생성
- 빌드 프로젝트 생성(CloudFormation)

사전 조건

빌드 프로젝트를 생성하기 전에 빌드 계획의 질문에 답변하세요.

빌드 프로젝트 만들기(콘솔)

https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.

CodeBuild 정보 페이지가 나타나면 빌드 프로젝트 생성을 선택합니다. 그렇지 않을 경우, 탐색 창에서 빌드를 확장한 후 빌드 프로젝트를 선택하고 빌드 프로젝트 생성을 선택합니다.

빌드 프로젝트 생성을 선택합니다.

다음 섹션을 채웁니다. 완료되면 페이지 하단에서 빌드 프로젝트 생성을 선택합니다.

섹션:

- 프로젝트 구성
- 소스
- 환경
- Buildspec
- 배치 구성
- 아티팩트
- 로그

프로젝트 구성

프로젝트 이름

이 빌드 프로젝트의 이름을 입력합니다. 빌드 프로젝트 이름은 각 AWS 계정에서 고유해야 합니다.

설명

선택적으로 빌드 프로젝트에 대한 설명을 입력하여 다른 사용자가 이 프로젝트의 용도를 이해하도록 도울 수 있습니다.

빌드 배지

(선택 사항) 프로젝트의 빌드 상태를 표시하고 삽입 가능하게 하려면 빌드 배지 활성화를 선택합니다. 자세한 내용은 빌드 배지 샘플 섹션을 참조하세요.

> **Note**
>
> 소스 공급자가 Amazon S3인 경우 빌드 배지가 적용되지 않습니다.

동시 빌드 제한 활성화

(선택 사항) 이 프로젝트의 동시 빌드 수를 제한하려면 다음 단계를 수행합니다.

1. 이 프로젝트에서 시작할 수 있는 동시 빌드 수 제한을 선택합니다.
2. 동시 빌드 제한에 이 프로젝트에 허용되는 최대 동시 빌드 수를 입력합니다. 이 제한은 계정에 설정된 동시 빌드 제한보다 클 수 없습니다. 계정 제한보다 큰 숫자를 입력하려고 하면 오류 메시지가 표시됩니다.

현재 빌드 수가 이 한도 이하인 경우에만 새 빌드가 시작됩니다. 현재 빌드 수가 이 한도에 도달하면 새 빌드가 제한되고 실행되지 않습니다.

추가 정보

(선택 사항) 태그에 지원 AWS 서비스에서 사용할 태그의 이름과 값을 입력합니다. [Add row]를 사용하여 태그를 추가합니다. 최대 50개의 태그를 추가할 수 있습니다.

소스

소스 공급자

소스 코드 공급자 유형을 선택합니다. 다음 목록을 사용하여 소스 공급자에 알맞은 유형을 선택합니다.

> **Note**
>
> CodeBuild에서는 Bitbucket Server를 지원하지 않습니다.

Amazon S3

버킷

소스 코드가 포함된 입력 버킷의 이름을 선택합니다.

S3 객체 키 또는 S3 폴더

소스 코드가 포함된 ZIP 파일의 이름 또는 폴더 경로를 입력합니다. S3 버킷의 모든 항목을 다운로드하려면 슬래시(/)를 입력합니다.

소스 버전

입력 파일의 빌드를 나타내는 객체의 버전 ID를 입력합니다. 자세한 내용은 단원을 참조하십시오를 사용한 소스 버전 샘플 AWS CodeBuild.

CodeCommit

리포지토리

사용할 리포지토리를 선택합니다.

참조 유형

분기, Git 태그 또는 커밋 ID를 선택하여 소스 코드 버전을 지정합니다. 자세한 내용은 를 사용한 소스 버전 샘플 AWS CodeBuild 섹션을 참조하세요.

> **Note**
>
> `811dd1ba1aba14473856cee38308caed7190c0d` 또는 `5392f7`과 같이 커밋 ID처럼 보이지 않는 Git 분기 이름을 선택하는 것이 좋습니다. 이렇게 하면 실제 커밋과 Git 체크아웃이 충돌하는 것을 방지할 수 있습니다.

Git clone 깊이

이력이 지정된 커밋 수로 잘린 부분 복제를 생성하려면 선택합니다. 전체 복제가 필요할 경우 전체를 선택합니다.

Git 하위 모듈

리포지토리에 Git 하위 모듈을 포함하려면 Use Git submodules(Git 하위 모듈 사용)를 선택합니다.

Bitbucket

자격 증명

기본 소스 자격 증명 또는 사용자 지정 소스 자격 증명을 선택하고 지침에 따라 기본 소스 자격 증명을 관리하거나 소스 자격 증명을 사용자 지정합니다.

연결 유형

AWS CodeConnections, OAuth, 앱 암호 또는 개인 액세스 토큰을 선택하여 CodeBuild에 연결합니다.

Connection

Bitbucket 연결 또는 Secrets Manager 암호를 선택하여 지정된 연결 유형을 통해 연결합니다.

리포지토리

내 Bitbucket 계정의 리포지토리 또는 퍼블릭 리포지토리를 선택하고 리포지토리 URL을 입력합니다.

소스 버전

분기, 커밋 ID, 태그 또는 참조와 커밋 ID를 입력합니다. 자세한 내용은 를 사용한 소스 버전 샘플 AWS CodeBuild 섹션을 참조하세요.

> **Note**
>
> `811dd1ba1aba14473856cee38308caed7190c0d` 또는 `5392f7`과 같이 커밋 ID처럼 보이지 않는 Git 분기 이름을 선택하는 것이 좋습니다. 이렇게 하면 실제 커밋과 Git 체크아웃이 충돌하는 것을 방지할 수 있습니다.

Git clone 깊이

Git clone 깊이를 선택하면 이력이 지정된 커밋 수로 잘린 부분 복제가 생성됩니다. 전체 복제가 필요할 경우 전체를 선택합니다.

Git 하위 모듈

리포지토리에 Git 하위 모듈을 포함하려면 Use Git submodules(Git 하위 모듈 사용)를 선택합니다.

빌드 상태

빌드 시작 및 완료 상태가 소스 공급자에게 보고되도록 하려면 빌드가 시작되고 완료될 때 소스 공급자에게 빌드 상태 보고를 선택합니다.

소스 공급자에게 빌드 상태를 보고하려면 소스 공급자와 연결된 사용자에게 리포지토리에 대한 쓰기 권한이 있어야 합니다. 사용자에게 쓰기 권한이 없는 경우 빌드 상태를 업데이트할 수 없습니다. 자세한 내용은 소스 공급자 액세스 섹션을 참조하세요.

상태 컨텍스트의 경우 Bitbucket 커밋 상태의 `name` 파라미터에 사용할 값을 입력합니다. 자세한 내용은 Bitbucket API 설명서의 빌드를 참조하세요.

대상 URL에 Bitbucket 커밋 상태의 `url` 파라미터에 사용할 값을 입력합니다. 자세한 내용은 Bitbucket API 설명서의 빌드를 참조하세요.

Webhook에 의해 트리거된 빌드의 상태는 항상 소스 공급자에게 보고됩니다. 콘솔 또는 API 직접 호출에서 시작된 빌드의 상태를 소스 공급자에게 보고하려면 이 설정을 선택해야 합니다.

프로젝트 빌드가 webhook에 의해 트리거되는 경우 이 설정의 변경 사항을 적용하려면 새 커밋을 리포지토리에 푸시해야 합니다.

코드 변경 사항이 이 리포지토리에 푸시될 때마다 CodeBuild가 소스 코드를 빌드하도록 하려면 기본 소스 webhook 이벤트에서 코드 변경이 이 리포지토리에 푸시될 때마다 다시 빌드를 선택합니다. webhook 및 필터 그룹에 대한 자세한 내용은 Bitbucket Webhook 이벤트 섹션을 참조하세요.

GitHub

자격 증명

기본 소스 자격 증명 또는 사용자 지정 소스 자격 증명을 선택하고 지침에 따라 기본 소스 자격 증명을 관리하거나 소스 자격 증명을 사용자 지정합니다.

연결 유형

GitHub 앱, OAuth 또는 개인 액세스 토큰을 선택하여 CodeBuild에 연결합니다.

Connection

GitHub 연결 또는 Secrets Manager 보안 암호를 선택하여 지정된 연결 유형을 통해 연결합니다.

리포지토리

내 GitHub 계정의 리포지토리, 퍼블릭 리포지토리 또는 GitHub 범위 웹후크를 선택하고 리포지토리 URL을 입력합니다.

소스 버전

분기, 커밋 ID, 태그 또는 참조와 커밋 ID를 입력합니다. 자세한 내용은 를 사용한 소스 버전 샘플 AWS CodeBuild 섹션을 참조하세요.

> **Note**
>
> `811dd1ba1aba14473856cee38308caed7190c0d` 또는 `5392f7`과 같이 커밋 ID처럼 보이지 않는 Git 분기 이름을 선택하는 것이 좋습니다. 이렇게 하면 실제 커밋과 Git 체크아웃이 충돌하는 것을 방지할 수 있습니다.

Git clone 깊이

Git clone 깊이를 선택하면 이력이 지정된 커밋 수로 잘린 부분 복제가 생성됩니다. 전체 복제가 필요할 경우 전체를 선택합니다.

Git 하위 모듈

리포지토리에 Git 하위 모듈을 포함하려면 Use Git submodules(Git 하위 모듈 사용)를 선택합니다.

빌드 상태

빌드 시작 및 완료 상태가 소스 공급자에게 보고되도록 하려면 빌드가 시작되고 완료될 때 소스 공급자에게 빌드 상태 보고를 선택합니다.

소스 공급자에게 빌드 상태를 보고하려면 소스 공급자와 연결된 사용자에게 리포지토리에 대한 쓰기 권한이 있어야 합니다. 사용자에게 쓰기 권한이 없는 경우 빌드 상태를 업데이트할 수 없습니다. 자세한 내용은 소스 공급자 액세스 섹션을 참조하세요.

상태 컨텍스트의 경우 GitHub 커밋 상태의 `context` 파라미터에 사용할 값을 입력합니다. 자세한 내용은 GitHub 개발자 안내서의 커밋 상태 생성을 참조하세요.

대상 URL에 GitHub 커밋 상태의 `target_url` 파라미터에 사용할 값을 입력합니다. 자세한 내용은 GitHub 개발자 안내서의 커밋 상태 생성을 참조하세요.

Webhook에 의해 트리거된 빌드의 상태는 항상 소스 공급자에게 보고됩니다. 콘솔 또는 API 직접 호출에서 시작된 빌드의 상태를 소스 공급자에게 보고하려면 이 설정을 선택해야 합니다.

프로젝트 빌드가 webhook에 의해 트리거되는 경우 이 설정의 변경 사항을 적용하려면 새 커밋을 리포지토리에 푸시해야 합니다.

코드 변경 사항이 이 리포지토리에 푸시될 때마다 CodeBuild가 소스 코드를 빌드하도록 하려면 기본 소스 webhook 이벤트에서 코드 변경이 이 리포지토리에 푸시될 때마다 다시 빌드를 선택합니다. webhook 및 필터 그룹에 대한 자세한 내용은 GitHub Webhook 이벤트 섹션을 참조하세요.

GitHub Enterprise Server

자격 증명

기본 소스 자격 증명 또는 사용자 지정 소스 자격 증명을 선택하고 지침에 따라 기본 소스 자격 증명을 관리하거나 소스 자격 증명을 사용자 지정합니다.

연결 유형

AWS CodeConnections CodeBuild에 연결할 또는 개인 액세스 토큰을 선택합니다.

Connection

GitHub Enterprise 연결 또는 Secrets Manager 보안 암호를 선택하여 지정된 연결 유형을 통해 연결합니다.

리포지토리

내 GitHub Enterprise 계정의 리포지토리 또는 GitHub Enterprise 범위 웹후크를 선택하고 리포지토리 URL을 입력합니다.

소스 버전

풀 요청, 분기, 커밋 ID, 태그 또는 참조와 커밋 ID를 입력합니다. 자세한 내용은 를 사용한 소스 버전 샘플 AWS CodeBuild 섹션을 참조하세요.

> **Note**
>
> `811dd1ba1aba14473856cee38308caed7190c0d` 또는 `5392f7`과 같이 커밋 ID처럼 보이지 않는 Git 분기 이름을 선택하는 것이 좋습니다. 이렇게 하면 실제 커밋과 Git 체크아웃이 충돌하는 것을 방지할 수 있습니다.

Git clone 깊이

Git clone 깊이를 선택하면 이력이 지정된 커밋 수로 잘린 부분 복제가 생성됩니다. 전체 복제가 필요할 경우 전체를 선택합니다.

Git 하위 모듈

리포지토리에 Git 하위 모듈을 포함하려면 Use Git submodules(Git 하위 모듈 사용)를 선택합니다.

빌드 상태

빌드 시작 및 완료 상태가 소스 공급자에게 보고되도록 하려면 빌드가 시작되고 완료될 때 소스 공급자에게 빌드 상태 보고를 선택합니다.

소스 공급자에게 빌드 상태를 보고하려면 소스 공급자와 연결된 사용자에게 리포지토리에 대한 쓰기 권한이 있어야 합니다. 사용자에게 쓰기 권한이 없는 경우 빌드 상태를 업데이트할 수 없습니다. 자세한 내용은 소스 공급자 액세스 섹션을 참조하세요.

상태 컨텍스트의 경우 GitHub 커밋 상태의 `context` 파라미터에 사용할 값을 입력합니다. 자세한 내용은 GitHub 개발자 안내서의 커밋 상태 생성을 참조하세요.

대상 URL에 GitHub 커밋 상태의 `target_url` 파라미터에 사용할 값을 입력합니다. 자세한 내용은 GitHub 개발자 안내서의 커밋 상태 생성을 참조하세요.

Webhook에 의해 트리거된 빌드의 상태는 항상 소스 공급자에게 보고됩니다. 콘솔 또는 API 직접 호출에서 시작된 빌드의 상태를 소스 공급자에게 보고하려면 이 설정을 선택해야 합니다.

프로젝트 빌드가 webhook에 의해 트리거되는 경우 이 설정의 변경 사항을 적용하려면 새 커밋을 리포지토리에 푸시해야 합니다.

비보안 SSL

GitHub Enterprise 프로젝트 리포지토리에 연결되어 있는 동안 SSL을 무시하려면 Enable insecure SSL(안전하지 않은 SSL 활성화)을 선택합니다.

코드 변경 사항이 이 리포지토리에 푸시될 때마다 CodeBuild가 소스 코드를 빌드하도록 하려면 기본 소스 webhook 이벤트에서 코드 변경이 이 리포지토리에 푸시될 때마다 다시 빌드를 선택합니다. webhook 및 필터 그룹에 대한 자세한 내용은 GitHub Webhook 이벤트 섹션을 참조하세요.

GitLab

자격 증명

기본 소스 자격 증명 또는 사용자 지정 소스 자격 증명을 선택하고 지침에 따라 기본 소스 자격 증명을 관리하거나 소스 자격 증명을 사용자 지정합니다.

연결 유형

AWS CodeConnections는 GitLab을 CodeBuild에 연결하는 데 사용됩니다.

Connection

연결할 GitLab 연결을 선택합니다 AWS CodeConnections.

리포지토리

사용할 리포지토리를 선택합니다.

소스 버전

pull 요청, 분기, 커밋 ID, 태그 또는 참조와 커밋 ID를 입력합니다. 자세한 내용은 를 사용한 소스 버전 샘플 AWS CodeBuild 섹션을 참조하세요.

> **Note**
>
> `811dd1ba1aba14473856cee38308caed7190c0d` 또는 `5392f7`과 같이 커밋 ID처럼 보이지 않는 Git 분기 이름을 선택하는 것이 좋습니다. 이렇게 하면 실제 커밋과 Git 체크아웃이 충돌하는 것을 방지할 수 있습니다.

Git clone 깊이

Git clone 깊이를 선택하면 이력이 지정된 커밋 수로 잘린 부분 복제가 생성됩니다. 전체 복제가 필요할 경우 전체를 선택합니다.

빌드 상태

빌드 시작 및 완료 상태가 소스 공급자에게 보고되도록 하려면 빌드가 시작되고 완료될 때 소스 공급자에게 빌드 상태 보고를 선택합니다.

소스 공급자에게 빌드 상태를 보고하려면 소스 공급자와 연결된 사용자에게 리포지토리에 대한 쓰기 권한이 있어야 합니다. 사용자에게 쓰기 권한이 없는 경우 빌드 상태를 업데이트할 수 없습니다. 자세한 내용은 소스 공급자 액세스 섹션을 참조하세요.

GitLab Self Managed

자격 증명

기본 소스 자격 증명 또는 사용자 지정 소스 자격 증명을 선택하고 지침에 따라 기본 소스 자격 증명을 관리하거나 소스 자격 증명을 사용자 지정합니다.

연결 유형

AWS CodeConnections는 GitLab Self Managed를 CodeBuild에 연결하는 데 사용됩니다.

Connection

연결할 GitLab 자체 관리형 연결을 선택합니다 AWS CodeConnections.

리포지토리

사용할 리포지토리를 선택합니다.

소스 버전

pull 요청, 분기, 커밋 ID, 태그 또는 참조와 커밋 ID를 입력합니다. 자세한 내용은 를 사용한 소스 버전 샘플 AWS CodeBuild 섹션을 참조하세요.

> **Note**
>
> 811dd1ba1aba14473856cee38308caed7190c0d 또는 5392f7과 같이 커밋 ID처럼 보이지 않는 Git 분기 이름을 선택하는 것이 좋습니다. 이렇게 하면 실제 커밋과 Git 체크아웃이 충돌하는 것을 방지할 수 있습니다.

Git clone 깊이

Git clone 깊이를 선택하면 이력이 지정된 커밋 수로 잘린 부분 복제가 생성됩니다. 전체 복제가 필요할 경우 전체를 선택합니다.

빌드 상태

빌드 시작 및 완료 상태가 소스 공급자에게 보고되도록 하려면 빌드가 시작되고 완료될 때 소스 공급자에게 빌드 상태 보고를 선택합니다.

소스 공급자에게 빌드 상태를 보고하려면 소스 공급자와 연결된 사용자에게 리포지토리에 대한 쓰기 권한이 있어야 합니다. 사용자에게 쓰기 권한이 없는 경우 빌드 상태를 업데이트할 수 없습니다. 자세한 내용은 소스 공급자 액세스 섹션을 참조하세요.

환경

프로비저닝 모델

다음 중 하나를 수행하세요.

- 에서 관리하는 온디맨드 플릿을 사용하려면 온디맨드를 AWS CodeBuild선택합니다. CodeBuild는 온디맨드 플릿을 통해 빌드에 맞는 컴퓨팅 기능을 제공합니다. 빌드가 완료되면 머신이 파괴됩니다. 온디맨드 플릿은 완전 관리형이며 수요 급증을 처리하기 위한 자동 조정 기능을 포함합니다.
- 에서 관리하는 예약 용량 플릿을 사용하려면 예약 용량을 AWS CodeBuild선택한 다음 플릿 이름을 선택합니다. 예약 용량 플릿을 사용하면 빌드 환경을 위한 전용 인스턴스 세트를 구성할 수 있습니다. 이러한 시스템은 유휴 상태로 유지되어 빌드 또는 테스트를 즉시 처리하고 빌드 기간을 단축할 수 있습니다. 예약 용량 플릿을 사용하면 머신이 항상 실행되며 프로비저닝되는 한 비용이 계속 발생합니다.

자세한 내용은 예약 용량 플릿에서 빌드 실행 섹션을 참조하세요.

환경 이미지

다음 중 하나를 수행하세요.

- 에서 관리하는 도커 이미지를 사용하려면 관리형 이미지를 AWS CodeBuild선택한 다음 운영 체제, 런타임, 이미지(Image) 및 이미지 버전에서 선택합니다. 사용 가능한 경우 환경 유형에서 항목을 선택합니다.
- 다른 도커 이미지를 사용하려면 사용자 지정 이미지를 선택합니다. 환경 유형에서 ARM, Linux, Linux GPU 또는 Windows를 선택합니다. Other registry(다른 레지스트리)를 선택한 경우 External registry URL(외부 레지스트리 URL)에 Docker Hub의 도커 이미지 이름 및 태그를 *docker repository*/*docker image name* 형식으로 입력합니다. Amazon ECR을 선택하는 경우 Amazon ECR 리포지토리와 Amazon ECR 이미지를 사용하여 AWS 계정에서 도커 이미지를 선택합니다.
- 프라이빗 도커 이미지를 사용하려면 사용자 지정 이미지를 선택합니다. 환경 유형에서 ARM, Linux, Linux GPU 또는 Windows를 선택합니다. Image registry(이미지 레지스트리)에서 Other registry(다른 레지스트리)를 선택한 다음 프라이빗 도커 이미지에 대한 보안 인증 정보의 ARN을 입력합니다. 보안 인증은 Secrets Manager에서 생성됩니다. 자세한 내용은 AWS Secrets Manager사용 설명서의 AWS Secrets Manager 이란? 섹션을 참조하세요.

> **Note**
>
> CodeBuild는 사용자 지정 도커 이미지에 대해 `ENTRYPOINT`를 재정의합니다.

컴퓨팅

다음 중 하나를 수행하세요.

- EC2 컴퓨팅을 사용하려면 EC2를 선택합니다. EC2 컴퓨팅은 작업 실행 중에 최적화된 유연성을 제공합니다.
- Lambda 컴퓨팅을 사용하려면 Lambda를 선택합니다. Lambda 컴퓨팅은 빌드에 최적화된 시작 속도를 제공합니다. Lambda는 시작 지연 시간이 짧아 더 빠른 빌드를 지원합니다. Lambda도 자동으로 확장되므로 빌드가 대기열에서 실행되기를 기다리지 않습니다. 자세한 내용은 AWS Lambda 컴퓨팅 기반 빌드 실행 섹션을 참조하세요.

서비스 역할

다음 중 하나를 수행하세요.

- CodeBuild 서비스 역할이 없는 경우 새 서비스 역할을 선택합니다. 역할 이름에 새 역할의 이름을 입력합니다.
- CodeBuild 서비스 역할이 있는 경우 기존 서비스 역할을 선택합니다. 역할 ARN에서 서비스 역할을 선택합니다.

> **Note**
>
> 콘솔을 사용하여 빌드 프로젝트를 생성하는 경우, 이와 동시에 CodeBuild 서비스 역할을 생성할 수 있습니다. 기본적으로 역할은 해당 빌드 프로젝트에서만 작동합니다. 콘솔을 사용하여 이 서비스 역할을 다른 빌드 프로젝트와 연결하는 경우 다른 빌드 프로젝트에서 작동하도록 역할이 업데이트됩니다. 하나의 서비스 역할은 최대 10개의 빌드 프로젝트에서 작동할 수 있습니다.

추가 구성

자동 재시도 제한

빌드 실패 후 추가 자동 재시도 횟수를 지정합니다. 예를 들어 자동 재시도 제한이 2로 설정된 경우 CodeBuild는 `RetryBuild` API를 호출하여 빌드를 추가로 최대 2회 자동 재시도합니다.

제한 시간

빌드가 완료되지 않는 경우 CodeBuild가 해당 빌드를 중지할 제한 시간으로 5분에서 36시간 사이의 값을 지정합니다. [hours] 및 [minutes]가 비어 있는 경우 기본값인 60분이 사용됩니다.

권한 있음

(선택 사항) 이 빌드 프로젝트를 사용하여 Docker 이미지를 빌드하려는 경우에만 Docker 이미지를 빌드하거나 빌드에서 승격된 권한을 얻으려는 경우 이 플래그 활성화를 선택합니다. 그렇

지 않으면 Docker 데몬과 상호 작용을 시도하는 모든 연결된 빌드가 실패합니다. 또한 빌드가 상호 작용할 수 있도록 Docker 데몬을 시작해야 합니다. 이렇게 하는 한 가지 방법은 다음 빌드 명령을 실행하여 buildspec의 `install` 단계에서 Docker 데몬을 초기화하는 것입니다. 선택한 빌드 환경 이미지가 Docker 지원을 통해 CodeBuild에서 제공되지 않는 경우 이 명령을 실행하지 마세요.

> **Note**
>
> 기본적으로 비 VPC 빌드에는 Docker 데몬이 활성화됩니다. VPC 빌드에 Docker 컨테이너를 사용하려면 Docker Docs 웹 사이트의 런타임 권한 및 Linux 기능을 참조하고 권한 부여 모드를 활성화합니다. 또한 Windows는 권한 모드를 지원하지 않습니다.

```
- nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --
host=tcp://127.0.0.1:2375 --storage-driver=overlay2 &
- timeout 15 sh -c "until docker info; do echo .; sleep 1; done"
```

VPC

CodeBuild를 사용하여 VPC에서 작업을 수행하려는 경우:

- VPC에서 CodeBuild가 사용하는 VPC ID를 선택합니다.
- VPC 서브넷에서 CodeBuild가 사용하는 리소스가 포함된 서브넷을 선택합니다.
- VPC 보안 그룹에서 CodeBuild가 VPC의 리소스에 대한 액세스를 허용하기 위해 사용하는 보안 그룹을 선택합니다.

자세한 내용은 Amazon Virtual Private Cloud AWS CodeBuild 와 함께 사용 섹션을 참조하세요.

컴퓨팅

사용 가능한 옵션 중 하나를 선택합니다.

레지스트리 자격 증명

프로젝트가 비공개 레지스트리 이미지로 구성된 경우 레지스트리 자격 증명을 지정합니다.

> **Note**
>
> 이 자격 증명은 이미지가 프라이빗 레지스트리의 이미지로 재정의된 경우에만 사용됩니다.

환경 변수

사용할 빌드의 각 환경 변수에 대해 이름 및 값을 입력하고 유형을 선택합니다.

> **Note**
>
> CodeBuild는 AWS 리전의 환경 변수를 자동으로 설정합니다. buildspec.yml에 추가하지 않은 경우 다음 환경 변수를 설정해야 합니다.
>
> - AWS_ACCOUNT_ID
> - IMAGE_REPO_NAME
> - IMAGE_TAG

콘솔과 AWS CLI 사용자는 환경 변수를 볼 수 있습니다. 환경 변수의 가시성에 대한 문제가 없다면 [Name] 및 [Value] 필드를 설정한 다음 [Type]을 [Plaintext]로 설정합니다.

액세스 키 ID, AWS 보안 AWS 액세스 키 또는 암호와 같은 민감한 값을 가진 환경 변수를 Amazon EC2 Systems Manager 파라미터 스토어 또는에 파라미터로 저장하는 것이 좋습니다 AWS Secrets Manager.

Amazon EC2 Systems Manager Parameter Store를 사용하는 경우 유형에서 파라미터를 선택합니다. 이름에 CodeBuild가 참조할 식별자를 입력합니다. 값에 Amazon EC2 Systems Manager Parameter Store에 저장되는 파라미터의 이름을 입력합니다. 예를 들어 `/CodeBuild/dockerLoginPassword`라는 이름의 파라미터를 사용하여 유형에서 파라미터를 선택합니다. 이름에 `LOGIN_PASSWORD`를 입력합니다. 값에 `/CodeBuild/dockerLoginPassword`를 입력합니다.

> **Important**
>
> Amazon EC2 Systems Manager Parameter Store를 사용하는 경우 `/CodeBuild/`로 시작하는 파라미터 이름(예: `/CodeBuild/dockerLoginPassword`)으로 파라미터를 저장하는 것이 좋습니다. CodeBuild 콘솔을 사용하여 Amazon EC2 Systems Manager에서 파라미터를 생성할 수 있습니다. 파라미터 생성을 선택하고 대화 상자에 표시되는 지시에 따릅니다. (그 대화 상자에서 KMS 키에 대해 계정에 있는 AWS KMS 키의 ARN을 지정할 수 있습니다. Amazon EC2 Systems Manager는이 키를 사용하여 저장 중에 파라미터의 값을 암호화하고 검색 중에 복호화합니다.) CodeBuild 콘솔을 사용하여 파라미터를 생성하는 경우 콘솔은 이름이 `/CodeBuild/`인 파라미터가 저장되면 시작합

니다. 자세한 내용은 Amazon EC2 Systems Manager 사용 설명서의 Systems Manager Parameter Store 및 Systems Manager Parameter Store 콘솔 연습을 참조하세요.
빌드 프로젝트가 Amazon EC2 Systems Manager Parameter Store에 저장된 파라미터를 참조하는 경우 해당 빌드 프로젝트의 서비스 역할은 `ssm:GetParameters` 작업을 허용해야 합니다. 앞에서 새 서비스 역할을 선택한 경우에는 CodeBuild는 빌드 프로젝트의 기본 서비스 역할에 이 작업을 포함합니다. Existing service role(기존 서비스 역할)을 선택한 경우에는 이 작업을 서비스 역할에 별도로 포함해야 합니다.
빌드 프로젝트가 `/CodeBuild/`로 시작되지 않는 파라미터 이름으로 Amazon EC2 Systems Manager Parameter Store에 저장된 파라미터를 참조하는 경우 새 서비스 역할을 선택하면 `/CodeBuild/`로 시작하지 않는 파라미터 이름에 액세스할 수 있도록 해당 서비스 역할을 업데이트해야 합니다. 이는 서비스 역할이 `/CodeBuild/`로 시작하는 파라미터 이름에만 액세스할 수 있기 때문입니다.
새 서비스 역할을 선택하면 Amazon EC2 Systems Manager Parameter Store에 있는 `/CodeBuild/` 네임스페이스의 모든 파라미터를 해독할 권한이 서비스 역할에 포함됩니다.
사용자가 설정한 환경 변수는 기존 환경 변수를 대체합니다. 예를 들어 도커 이미지에 값이 `my_value`인 `MY_VAR`이라는 환경 변수가 이미 포함되어 있는데, 사용자가 `MY_VAR` 환경 변수의 값을 `other_value`로 설정하면, `my_value`가 `other_value`로 바뀝니다. 마찬가지로, 도커 이미지에 값이 `/usr/local/sbin:/usr/local/bin`인 `PATH`라는 환경 변수가 이미 포함되어 있는데, 사용자가 `PATH` 환경 변수의 값을 `$PATH:/usr/share/ant/bin`으로 설정하면, `/usr/local/sbin:/usr/local/bin`이 `$PATH:/usr/share/ant/bin` 리터럴 값으로 바뀝니다.
`CODEBUILD_`로 시작하는 이름으로 환경 변수를 설정하지 마십시오. 이 접두사는 내부 전용으로 예약되어 있습니다.
여러 위치에서 동일한 이름의 환경 변수가 정의되는 경우, 다음과 같이 값이 결정됩니다.

- 시작 빌드 작업 호출의 값이 가장 높은 우선 순위를 갖습니다.
- 빌드 프로젝트 정의의 값이 다음 우선 순위를 갖습니다.
- buildspec 선언의 값이 가장 낮은 우선 순위를 갖습니다.

Secrets Manager를 사용하는 경우 유형으로 Secrets Manager로 선택합니다. 이름에 CodeBuild가 참조할 식별자를 입력합니다. 값에 *`secret-id`*`:`*`json-key`*`:`*`version-stage`*`:`*`version-id`* 패턴을 사용하여 `reference-key`를 입력합니다. 자세한 내용은 Secrets Manager reference-key in the buildspec file 섹션을 참조하세요.

> **Important**
>
> Secrets Manager를 사용하는 경우 이름이 `/CodeBuild/`로 시작하는 암호를 저장하는 것이 좋습니다(예: `/CodeBuild/dockerLoginPassword`). 자세한 내용은 AWS Secrets Manager사용 설명서의 AWS Secrets Manager 이란? 섹션을 참조하세요.
> 빌드 프로젝트가 Secrets Manager에 저장된 암호를 참조하는 경우 해당 빌드 프로젝트의 서비스 역할은 `secretsmanager:GetSecretValue` 작업을 허용해야 합니다. 앞에서 새 서비스 역할을 선택한 경우에는 CodeBuild는 빌드 프로젝트의 기본 서비스 역할에 이 작업을 포함합니다. Existing service role(기존 서비스 역할)을 선택한 경우에는 이 작업을 서비스 역할에 별도로 포함해야 합니다.
> 빌드 프로젝트가 `/CodeBuild/`로 시작되지 않는 보안 암호 이름으로 Secrets Manager에 저장된 암호를 참조하는 경우 새 서비스 역할을 선택하면 `/CodeBuild/`로 시작하지 않는 보안 암호 이름에 액세스할 수 있도록 해당 서비스 역할을 업데이트해야 합니다. 이는 서비스 역할이 `/CodeBuild/`로 시작하는 암호 이름에만 액세스할 수 있기 때문입니다.
> 새 서비스 역할을 선택하면 에 있는 `/CodeBuild/` 네임스페이스의 모든 암호를 해독할 권한이 서비스 역할에 포함됩니다.

Buildspec

빌드 사양

다음 중 하나를 수행하세요.

- 소스 코드에 buildspec 파일이 있는 경우 Use a buildspec file(빌드 사양 파일 사용) 을 선택합니다. 기본적으로 CodeBuild는 소스 코드 루트 디렉터리에서 `buildspec.yml`이라는 파일을 찾습니다. buildspec 파일이 다른 이름이나 위치를 사용하는 경우 Buildspec 이름에 소스 루트의 경로를 입력합니다(예: `buildspec-two.yml` 또는 `configuration/buildspec.yml`). buildspec 파일이 S3 버킷에 있는 경우 해당 파일은 빌드 프로젝트와 동일한 AWS 리전에 있어야 합니다. ARN을 사용하여 buildspec 파일을 지정합니다(예: `arn:aws:s3:::`*`<my-codebuild-sample2>`*`/buildspec.yml`).
- 소스 코드에 buildspec 파일이 포함되어 있지 않거나, 소스 코드의 루트 디렉터리에 있는 `buildspec.yml` 파일의 `build` 단계에 지정된 것과 다른 빌드 명령 세트를 실행하려는 경우 빌드 명령 삽입을 선택합니다. 빌드 명령의 `build` 단계에서 실행하려는 명령을 입력합니다. 명령이 여러 개인 경우 각 명령을 `&&`로 구분합니다(예: `mvn test && mvn package`). 다른 구문에서 명령을 실행하려는 경우 또는 `build` 단계에 대해 특히 긴 명령 목록이 있는 경우에는 소스 코

드 루트 디렉터리에 `buildspec.yml` 파일을 추가하고, 이 파일에 명령을 추가한 다음, 소스 코드 루트 디렉터리에서 buildspec.yml 사용을 선택합니다.

자세한 내용은 buildspec 참조 섹션을 참조하세요.

배치 구성

빌드 그룹을 단일 작업으로 실행할 수 있습니다. 자세한 내용은 배치로 빌드 실행 섹션을 참조하세요.

배치 구성 정의

이 프로젝트에서 배치 빌드를 허용하려면 선택합니다.

배치 서비스 역할

배치 빌드에 대한 서비스 역할을 제공합니다.

다음 중 하나를 선택합니다.

- 배치 서비스 역할이 없는 경우 새 서비스 역할을 선택합니다. 서비스 역할에 새 역할의 이름을 입력합니다.
- 배치 서비스 역할이 있는 경우 기존 서비스 역할을 선택합니다. 서비스 역할에서 서비스 역할을 선택합니다.

배치 빌드는 배치 구성에 새로운 보안 역할을 도입합니다. CodeBuild가 사용자 대신 `StartBuild`, `StopBuild` 및 `RetryBuild` 작업을 호출하여 배치의 일부로 빌드를 실행할 수 있어야 하므로 이 새 역할이 필요합니다. 고객은 다음과 같은 두 가지 이유로 빌드에 사용하는 것과 동일한 역할이 아닌 새 역할을 사용해야 합니다.

- 빌드 역할 `StartBuild`, `StopBuild` 및 `RetryBuild` 권한을 부여하면 단일 빌드에서 buildspec을 통해 더 많은 빌드를 시작할 수 있습니다.
- CodeBuild 배치 빌드는 배치의 빌드에 사용할 수 있는 빌드 및 컴퓨팅 유형의 수를 제한하는 제한을 제공합니다. 빌드 역할에 이러한 권한이 있는 경우 빌드 자체가 이러한 제한을 우회할 수 있습니다.

배치에 허용되는 컴퓨팅 유형

배치에 허용되는 컴퓨팅 유형을 선택합니다. 해당하는 항목을 모두 선택합니다.

배치에 허용되는 플릿

배치에 허용되는 플릿을 선택합니다. 해당하는 항목을 모두 선택합니다.

배치에 허용되는 최대 빌드 수

배치에 허용되는 최대 빌드 수를 입력합니다. 이 제한을 초과하는 배치는 실패합니다.

배치 제한 시간

배치 빌드가 완료되는 최대 시간을 입력합니다.

아티팩트 결합

배치의 모든 아티팩트를 단일 위치로 결합을 선택하면 배치의 모든 아티팩트가 단일 위치로 결합됩니다.

배치 보고서 모드

배치 빌드에 대해 원하는 빌드 상태 보고서 모드를 선택합니다.

> **Note**
>
> 이 필드는 프로젝트 소스가 Bitbucket, GitHub 또는 GitHub Enterprise이고 소스에서 빌드 시작 및 종료 시 소스 공급자에게 빌드 상태 보고를 선택한 경우에만 사용할 수 있습니다.

빌드 집계

배치의 모든 빌드 상태를 단일 상태 보고서로 통합하려면 선택합니다.

개별 빌드

배치에 있는 모든 빌드의 빌드 상태를 별도로 보고하려면 선택합니다.

아티팩트

유형

다음 중 하나를 수행하세요.

- 빌드 출력 결과물을 생성하지 않으려면 [No artifacts]를 선택합니다. 빌드 테스트만 실행하고 있는 경우 또는 Amazon ECR 리포지토리에 도커 이미지를 푸시하려는 경우에 이 작업을 원할 수 있습니다.
- S3 버킷에 빌드 출력을 저장하려면 Amazon S3를 선택하고 다음 작업을 수행합니다.
  - 빌드 출력 ZIP 파일이나 폴더에 프로젝트 이름을 사용하려는 경우 이름을 비워 둡니다. 그렇지 않으면 이름을 입력합니다. (ZIP 파일을 출력하고 ZIP 파일에 파일 확장명을 넣으려는 경우, ZIP 파일 이름 뒤에 이를 포함하십시오.)

- buildspec 파일에 지정된 이름으로 콘솔에서 지정한 이름을 재정의하려는 경우 의미 체계 버전 관리 사용을 선택합니다. buildspec 파일의 이름은 빌드 시 계산되며 Shell 명령 언어를 사용합니다. 예를 들어 아티팩트 이름이 항상 고유하도록 날짜와 시간을 아티팩트 이름에 추가할 수 있습니다. 고유한 결과물 이름을 사용하면 결과물을 덮어쓰지 않을 수 있습니다. 자세한 내용은 buildspec 구문 섹션을 참조하세요.
- [Bucket name]에서 출력 버킷의 이름을 선택합니다.
- 이 절차의 앞부분에서 빌드 명령 삽입을 선택한 경우 출력 파일에 빌드 출력 ZIP 파일 또는 폴더에 넣으려는 빌드의 파일 위치를 입력합니다. 위치가 여러 개인 경우 각 위치를 쉼표로 구분합니다(예: `appspec.yml, target/my-app.jar`). 자세한 내용은 buildspec 구문의 `files` 설명을 참조하십시오.
- 빌드 아티팩트를 암호화하지 않으려면 Remove artifacts encryption(결과물 암호화 제거)을 선택합니다.

각각 원하는 보조 아티팩트 세트마다 다음과 같이 실행합니다.

1. Atrifact identifier(아티팩트 식별자)에서 128자 미만으로 영숫자와 밑줄만 포함된 값을 입력합니다.
2. Add artifact(아티팩트 추가)를 선택합니다.
3. 이전 단계에 따라 보조 결과물을 구성합니다.
4. Save artifact(아티팩트 저장)를 선택합니다.

추가 구성

암호화 키

(선택 사항) 다음 중 하나를 수행하십시오.

- 계정에서 Amazon S3에 대한 AWS 관리형 키 를 사용하여 빌드 출력 아티팩트를 암호화하려면 암호화 키를 비워 둡니다. 기본값입니다.
- 고객 관리형 키를 사용하여 빌드 출력 아티팩트를 암호화하려면 암호화 키에 KMS 키의 ARN을 입력합니다. `arn:aws:kms:`*`region-ID`*`:`*`account-ID`*`:key/`*`key-ID`* 형식을 사용합니다.

캐시 유형

Cache type(캐시 유형)에서 다음 중 하나를 선택합니다.

- 캐시를 사용하지 않으려면 [No cache]를 선택합니다.
- Amazon S3 캐시를 사용하려면 Amazon S3를 선택하고 다음을 수행합니다.

- 버킷에서 캐시가 저장된 S3 버킷의 이름을 선택합니다.
- (선택 사항) 캐시 경로 접두사에 Amazon S3 경로 접두사를 입력합니다. Cache path prefix(캐시 경로 접두사) 값은 디렉터리 이름과 비슷합니다. 따라서 캐시를 버킷의 동일한 디렉터리에 저장할 수 있습니다.

> **Important**
>
> 경로 접두사 끝에 후행 슬래시(/)를 추가하지 마십시오.

- 로컬 캐시를 사용하려면 로컬을 선택한 다음 하나 이상의 로컬 캐시 모드를 선택해야 합니다.

> **Note**
>
> Docker 계층 캐시 모드는 Linux에서만 사용할 수 있습니다. 이 모드를 선택할 경우 프로젝트를 권한이 있는 모드에서 실행해야 합니다.

캐시를 사용하면 빌드 환경의 재사용 가능한 특정 부분이 캐시에 저장되고 빌드 전반에서 사용되기 때문에 상당한 빌드 시간을 절약할 수 있습니다. buildspec 파일에 캐시를 지정하는 것에 대한 자세한 정보는 buildspec 구문 단원을 참조하십시오. 캐싱에 대한 자세한 정보는 성능을 개선하기 위한 캐시 빌드을 참조하십시오.

로그

생성하려는 로그를 선택합니다. Amazon CloudWatch Logs 또는 Amazon S3 로그 중 하나 또는 둘 다를 생성할 수 있습니다.

CloudWatch

Amazon CloudWatch Logs 로그를 원할 경우:

CloudWatch 로그

CloudWatch 로그를 선택합니다.

그룹 이름

Amazon CloudWatch Logs 로그 그룹의 이름을 입력합니다.

스트림 이름

Amazon CloudWatch Logs 로그 스트림 이름을 입력합니다.

S3

Amazon S3 로그를 원할 경우:

S3 로그

S3 로그를 선택합니다.

버킷

로그에 대한 S3 버킷 이름을 선택합니다.

경로 접두사

로그의 접두사를 입력합니다.

S3 로그 암호화 비활성화

S3 로그를 암호화하지 않으려면 선택합니다.

빌드 프로젝트 생성(AWS CLI)

CodeBuild AWS CLI 에서를 사용하는 방법에 대한 자세한 내용은 섹션을 참조하세요명령줄 참조.

를 사용하여 CodeBuild 빌드 프로젝트를 생성하려면 JSON 형식의 프로젝트 구조를 AWS CLI생성하고 구조를 입력한 다음 `create-project` 명령을 호출하여 프로젝트를 생성합니다.

JSON 파일 생성

`--generate-cli-skeleton` 옵션을 사용하여 `create-project` 명령으로 스켈레톤 JSON 파일을 생성합니다.

```
aws codebuild create-project --generate-cli-skeleton > <json-file>
```

이렇게 하면 *<json-file>*로 지정된 경로와 파일 이름을 가진 JSON 파일이 생성됩니다.

JSON 파일 채우기

JSON 데이터를 다음과 같이 수정하고 결과를 저장합니다.

```
{
  "name": "<project-name>",
  "description": "<description>",
  "source": {
```

```
    "type": "CODECOMMIT" | "CODEPIPELINE" | "GITHUB" | "GITHUB_ENTERPRISE" | "GITLAB" |
"GITLAB_SELF_MANAGED" | "BITBUCKET" | "S3" | "NO_SOURCE",
    "location": "<source-location>",
    "gitCloneDepth": "<git-clone-depth>",
    "buildspec": "<buildspec>",
    "InsecureSsl": "<insecure-ssl>",
    "reportBuildStatus": "<report-build-status>",
    "buildStatusConfig": {
      "context": "<context>",
      "targetUrl": "<target-url>"
    },
    "gitSubmodulesConfig": {
      "fetchSubmodules": "<fetch-submodules>"
    },
    "auth": {
      "type": "<auth-type>",
      "resource": "<auth-resource>"
    },
    "sourceIdentifier": "<source-identifier>"
  },
  "secondarySources": [
    {
        "type": "CODECOMMIT" | "CODEPIPELINE" | "GITHUB" | "GITHUB_ENTERPRISE" |
"GITLAB" | "GITLAB_SELF_MANAGED" | "BITBUCKET" | "S3" | "NO_SOURCE",
        "location": "<source-location>",
        "gitCloneDepth": "<git-clone-depth>",
        "buildspec": "<buildspec>",
        "InsecureSsl": "<insecure-ssl>",
        "reportBuildStatus": "<report-build-status>",
        "auth": {
          "type": "<auth-type>",
          "resource": "<auth-resource>"
        },
        "sourceIdentifier": "<source-identifier>"
    }
  ],
  "secondarySourceVersions": [
    {
      "sourceIdentifier": "<secondary-source-identifier>",
      "sourceVersion": "<secondary-source-version>"
    }
  ],
  "sourceVersion": "<source-version>",
  "artifacts": {
```

```
    "type": "CODEPIPELINE" | "S3" | "NO_ARTIFACTS",
    "location": "<artifacts-location>",
    "path": "<artifacts-path>",
    "namespaceType": "<artifacts-namespacetype>",
    "name": "<artifacts-name>",
    "overrideArtifactName": "<override-artifact-name>",
    "packaging": "<artifacts-packaging>"
  },
  "secondaryArtifacts": [
    {
      "type": "CODEPIPELINE" | "S3" | "NO_ARTIFACTS",
      "location": "<secondary-artifact-location>",
      "path": "<secondary-artifact-path>",
      "namespaceType": "<secondary-artifact-namespaceType>",
      "name": "<secondary-artifact-name>",
      "packaging": "<secondary-artifact-packaging>",
      "artifactIdentifier": "<secondary-artifact-identifier>"
    }
  ],
  "cache": {
    "type": "<cache-type>",
    "location": "<cache-location>",
    "mode": [
      "<cache-mode>"
    ]
  },
  "environment": {
    "type": "LINUX_CONTAINER" | "LINUX_GPU_CONTAINER" | "ARM_CONTAINER" |
"WINDOWS_SERVER_2019_CONTAINER" | "WINDOWS_SERVER_2022_CONTAINER",
    "image": "<image>",
    "computeType": "BUILD_GENERAL1_SMALL" | "BUILD_GENERAL1_MEDIUM" |
"BUILD_GENERAL1_LARGE" | "BUILD_GENERAL1_2XLARGE",
    "certificate": "<certificate>",
    "environmentVariables": [
      {
        "name": "<environmentVariable-name>",
        "value": "<environmentVariable-value>",
        "type": "<environmentVariable-type>"
      }
    ],
    "registryCredential": [
      {
        "credential": "<credential-arn-or-name>",
        "credentialProvider": "<credential-provider>"
```

```
      }
    ],
    "imagePullCredentialsType": "CODEBUILD" | "SERVICE_ROLE",
    "privilegedMode": "<privileged-mode>"
  },
  "serviceRole": "<service-role>",
  "autoRetryLimit": <auto-retry-limit>,
  "timeoutInMinutes": <timeout>,
  "queuedTimeoutInMinutes": <queued-timeout>,
  "encryptionKey": "<encryption-key>",
  "tags": [
    {
      "key": "<tag-key>",
      "value": "<tag-value>"
    }
  ],
  "vpcConfig": {
    "securityGroupIds": [
        "<security-group-id>"
    ],
    "subnets": [
        "<subnet-id>"
    ],
    "vpcId": "<vpc-id>"
  },
  "badgeEnabled": "<badge-enabled>",
  "logsConfig": {
    "cloudWatchLogs": {
      "status": "<cloudwatch-logs-status>",
      "groupName": "<group-name>",
      "streamName": "<stream-name>"
    },
    "s3Logs": {
      "status": "<s3-logs-status>",
      "location": "<s3-logs-location>",
      "encryptionDisabled": "<s3-logs-encryption-disabled>"
    }
  },
  "fileSystemLocations": [
    {
      "type": "EFS",
      "location": "<EFS-DNS-name-1>:/<directory-path>",
      "mountPoint": "<mount-point>",
      "identifier": "<efs-identifier>",
```

```
      "mountOptions": "<efs-mount-options>"
    }
  ],
  "buildBatchConfig": {
    "serviceRole": "<batch-service-role>",
    "combineArtifacts": <combine-artifacts>,
    "restrictions": {
      "maximumBuildsAllowed": <max-builds>,
      "computeTypesAllowed": [
        "<compute-type>"
      ],
      "fleetsAllowed": [
        "<fleet-name>"
      ]
    },
    "timeoutInMins": <batch-timeout>,
    "batchReportMode": "REPORT_AGGREGATED_BATCH" | "REPORT_INDIVIDUAL_BUILDS"
  },
  "concurrentBuildLimit": <concurrent-build-limit>
}
```

다음을 바꿉니다.

이름

필수 사항입니다. 이 빌드 프로젝트의 이름입니다. 이 이름은 AWS 계정의 모든 빌드 프로젝트에서 고유해야 합니다.

description

선택 사항. 이 빌드 프로젝트의 설명입니다.

source

필수 사항입니다. 이 빌드 프로젝트의 소스 코드 설정에 대한 정보가 들어 있는 ProjectSource 객체입니다. `source` 객체를 추가한 후에는 를 사용해 소스를 최대 12개까지 더 추가할 수 있습니다. 이러한 설정에는 다음이 포함됩니다.

source/type

필수 사항입니다. 빌드할 소스 코드가 포함된 리포지토리의 유형입니다. 유효한 값으로는 다음이 포함됩니다.

- CODECOMMIT
- CODEPIPELINE
- GITHUB
- GITHUB_ENTERPRISE
- GITLAB
- GITLAB_SELF_MANAGED
- BITBUCKET
- S3
- NO_SOURCE

NO_SOURCE를 사용할 경우 프로젝트에 소스가 없으므로 buildspec은 파일이 될 수 없습니다. 대신에 buildspec 속성을 사용하여 buildspec에 대해 YAML 형식이 지정된 문자열을 지정해야 합니다. 자세한 내용은 소스 없이 빌드 프로젝트 생성 섹션을 참조하세요.

source/location

*<source-type>*을 CODEPIPELINE으로 설정하지 않은 경우 필수입니다. 지정한 리포지토리 유형의 소스 코드 위치입니다.

- CodeCommit의 경우 HTTPS가 소스 코드 및 buildspec 파일을 포함하는 리포지토리에 URL을 복제합니다(예: https://git-codecommit.*<region-id>*.amazonaws.com/v1/repos/*<repo-name>*).
- Amazon S3의 경우 빌드 입력 버킷 이름 다음에 소스 코드 및 buildspec이 포함된 ZIP 파일의 경로와 이름이 옵니다. 예제:
  - 입력 버킷의 루트에 있는 ZIP 파일의 경우: *<bucket-name>*/*<object-name>*.zip
  - 입력 버킷의 하위 폴더에 있는 ZIP 파일의 경우: *<bucket-name>*/*<subfoler-path>*/*<object-name>*.zip
- GitHub의 경우 HTTPS가 소스 코드 및 buildspec 파일을 포함하는 리포지토리에 URL을 복제합니다. URL에 github.com이 포함되어야 합니다. AWS 계정을 GitHub 계정에 연결해야 합니다. 이렇게 하려면 CodeBuild 콘솔을 사용하여 빌드 프로젝트를 생성합니다.
  - [Authorize application]을 선택합니다. (GitHub 계정에 연결했으면 빌드 프로젝트 생성을 완료하지 않아도 됩니다. CodeBuild 콘솔을 닫아도 됩니다.)
- GitHub Enterprise Server의 경우 HTTP 또는 HTTPS가 소스 코드 및 buildspec 파일을 포함하는 리포지토리에 URL을 복제합니다. 또한 AWS 계정을 GitHub Enterprise Server 계정에 연결해야 합니다. 이렇게 하려면 CodeBuild 콘솔을 사용하여 빌드 프로젝트를 생성합니다.

1. GitHub Enterprise Server에서 개인용 액세스 토큰을 생성합니다.
2. CodeBuild 프로젝트를 생성할 때 사용할 수 있도록 이 토큰을 클립보드에 복사합니다. 자세한 내용은 GitHub Help 웹 사이트의 Creating a personal access token for the command line을 참조하십시오.
3. 콘솔을 사용하여 CodeBuild 프로젝트를 생성하는 경우, 소스의 소스 공급자에서 GitHub Enterprise를 선택합니다.
4. [Personal Access Token]에서 클리보드에 복사한 토큰을 붙여 넣습니다. 토큰 저장을 선택합니다. 이제 CodeBuild 계정이 GitHub Enterprise Server 계정에 연결되었습니다.

- GitLab 및 GitLab self-managed의 경우 HTTPS가 소스 코드 및 buildspec 파일을 포함하는 리포지토리에 URL을 복제합니다. GitLab을 사용하는 경우 URL에 gitlab.com이 포함되어야 합니다. GitLab self-managed를 사용하는 경우 URL에 gitlab.com을 포함할 필요가 없습니다. AWS 계정을 GitLab 또는 GitLab 자체 관리형 계정에 연결해야 합니다. 이렇게 하려면 CodeBuild 콘솔을 사용하여 빌드 프로젝트를 생성합니다.
  - 개발자 도구 탐색 창에서 설정, 연결을 선택한 다음, 연결 생성을 선택합니다. 이 페이지에서 GitLab 또는 GitLab self-managed 연결을 생성한 다음 GitLab에 연결을 선택합니다.
- Bitbucket의 경우 HTTPS가 소스 코드 및 buildspec 파일을 포함하는 리포지토리에 URL을 복제합니다. URL에 bitbucket.org가 포함되어야 합니다. 또한 AWS 계정을 Bitbucket 계정에 연결해야 합니다. 이렇게 하려면 CodeBuild 콘솔을 사용하여 빌드 프로젝트를 생성합니다.
  1. 콘솔을 사용하여 Bitbucket에 연결(또는 재연결)하면 Bitbucket [Confirm access to your account] 페이지에서 [Grant access]를 선택합니다. (Bitbucket 계정에 연결했으면 빌드 프로젝트 생성을 완료하지 않아도 됩니다. CodeBuild 콘솔을 닫아도 됩니다.)
- 의 경우에 대한 `location` 값을 지정하지 AWS CodePipeline마십시오`source`. CodePipeline에서 파이프라인을 생성하면 파이프라인의 소스 단계에서 소스 코드 위치를 지정하게 되므로 이 값이 무시됩니다.

source/gitCloneDepth

선택 사항. 다운로드할 이력의 수준입니다. 최소값은 0입니다. 이 값이 0이거나, 25를 초과하거나, 지정되지 않은 경우 각 빌드 프로젝트에서 전체 이력이 다운로드됩니다. 소스 유형이 Amazon S3일 경우 이 값이 지원되지 않습니다.

source/buildspec

선택 사항. 사용할 빌드 사양 정의 또는 파일입니다. 이 값을 제공하지 않거나 빈 문자열로 설정하는 경우 소스 코드에 루트 디렉터리의 `buildspec.yml` 파일이 포함되어 있어야 합니다. 이 값이 설정된 경우 인라인 buildspec 정의, 기본 소스의 루트 디렉터리에 상대적인 대체 buildspec 파일의 경로 또는 S3 버킷의 경로가 될 수 있습니다. 버킷은 빌드 프로젝트와 동일한 AWS 리전에 있어야

합니다. ARN을 사용하여 buildspec 파일을 지정합니다(예: `arn:aws:s3:::<my-codebuild-sample2>/buildspec.yml`). 자세한 내용은 buildspec 파일 이름 및 스토리지 위치 섹션을 참조하세요.

source/auth

CodeBuild가 빌드할 소스 코드에 액세스하기 위한 권한 부여 설정에 대한 정보를 포함합니다.

source/auth/type

필수 사항입니다. 사용할 권한 부여 유형입니다. 유효값은 다음과 같습니다.

- `OAUTH`
- `CODECONNECTIONS`
- `SECRETS_MANAGER`

source/auth/resource

선택 사항. 지정된 권한 부여 유형에 적용되는 리소스 값입니다. Secrets Manager ARN 또는 AWS CodeConnections ARN일 수 있습니다.

source/reportBuildStatus

빌드의 시작 및 완료 상태를 소스 공급자에게 보낼지 여부를 지정합니다. GitHub, GitHub Enterprise Server 또는 Bitbucket이 아닌 소스 공급자로 설정하는 경우, `invalidInputException`이 발생합니다.

소스 공급자에게 빌드 상태를 보고하려면 소스 공급자와 연결된 사용자에게 리포지토리에 대한 쓰기 권한이 있어야 합니다. 사용자에게 쓰기 권한이 없는 경우 빌드 상태를 업데이트할 수 없습니다. 자세한 내용은 소스 공급자 액세스 섹션을 참조하세요.

source/buildStatusConfig

CodeBuild 빌드 프로젝트가 빌드 상태를 소스 공급자에게 보고하는 방법을 정의하는 정보를 포함합니다. 이 옵션은 소스 유형이 `GITHUB`, `GITHUB_ENTERPRISE` 또는 `BITBUCKET`인 경우에만 사용됩니다.

source/buildStatusConfig/context

Bitbucket 소스의 경우 이 파라미터는 Bitbucket 커밋 상태의 `name` 파라미터에 사용됩니다. GitHub 소스의 경우 이 파라미터는 GitHub 커밋 상태의 `context` 파라미터에 사용됩니다.

예를 들어 CodeBuild 환경 변수를 사용하여 `context`에 빌드 번호와 webhook 트리거를 포함할 수 있습니다.

```
AWS CodeBuild sample-project Build #$CODEBUILD_BUILD_NUMBER -
 $CODEBUILD_WEBHOOK_TRIGGER
```

그 결과 webhook 풀 요청 이벤트에 의해 트리거된 빌드 #24에 대해 다음과 같은 컨텍스트가 나타납니다.

```
AWS CodeBuild sample-project Build #24 - pr/8
```

source/buildStatusConfig/targetUrl

Bitbucket 소스의 경우 이 파라미터는 Bitbucket 커밋 상태의 `url` 파라미터에 사용됩니다. GitHub 소스의 경우 이 파라미터는 GitHub 커밋 상태의 `target_url` 파라미터에 사용됩니다.

예를 들어 `targetUrl`를 `https://aws.amazon.com/codebuild/`*`<path to build>`*로 설정하면 커밋 상태가 이 URL에 연결됩니다.

CodeBuild 환경 변수를 `targetUrl`에 포함하여 URL에 추가 정보를 추가할 수도 있습니다. 예를 들어 URL에 빌드 리전을 영역을 추가하려면 다음과 같이 `targetUrl`을 설정합니다.

```
"targetUrl": "https://aws.amazon.com/codebuild/<path to build>?region=
$AWS_REGION"
```

빌드 리전이 `us-east-2`이면 다음과 같이 확장됩니다.

```
https://aws.amazon.com/codebuild/<path to build>?region=us-east-2
```

source/gitSubmodulesConfig

선택 사항. Git 하위 모듈 구성에 대한 정보입니다. CodeCommit, GitHub, GitHub Enterprise Server 및 Bitbucket에서만 사용됩니다.

source/gitSubmodulesConfig/fetchSubmodules

리포지토리에 Git 하위 모듈을 포함하려면 `fetchSubmodules`를 `true`로 설정합니다. 포함된 Git 하위 모듈은 HTTPS로 구성해야 합니다.

source/InsecureSsl

선택 사항. GitHub Enterprise Server에서만 사용됩니다. GitHub Enterprise Server 프로젝트 리포지토리에 연결되어 있는 동안 TLS 경고를 무시하려면 이 값을 `true`로 설정합니다. 기본값은

`false`입니다. `InsecureSsl`은 테스트 용도로만 사용해야 합니다. 프로덕션 환경에 사용하면 안 됩니다.

source/sourceIdentifier

프로젝트 소스의 사용자 정의 식별자입니다. 기본 소스의 경우 선택 사항입니다. 보조 소스에 필요합니다.

secondarySources

선택 사항. 빌드 프로젝트의 보조 소스에 대한 정보가 들어 있는 ProjectSource 객체의 배열입니다. 보조 소스는 12개까지 추가할 수 있습니다. `secondarySources` 객체는 객체에서 사용하는 것과 동일한 속성을 사용합니다. 보조 소스 객체에는 `sourceIdentifier`가 필요합니다.

secondarySourceVersions

선택 사항. ProjectSourceVersion 객체의 배열입니다. 빌드 레벨에서 `secondarySourceVersions`이 지정되어 있으면 그 버전이 이 버전보다 우선합니다.

sourceVersion

선택 사항. 이 프로젝트에 대해 빌드할 빌드 입력의 버전입니다. 지정하지 않으면 최신 버전이 사용됩니다. 지정하면 다음 중 하나여야 합니다.

- CodeCommit의 경우 사용할 커밋 ID, 분기 또는 Git 태그입니다.
- GitHub의 경우, 빌드하려는 소스 코드의 버전에 해당하는 커밋 ID, 풀 요청 ID, 분기 이름 또는 태그 이름입니다. 풀 요청 ID가 지정된 경우 `pr/pull-request-ID` 형식을 사용해야 합니다(예: `pr/25`). 분기 이름이 지정되어 있으면 분기의 HEAD 커밋 ID가 사용됩니다. 지정되지 않은 경우 기본 분기의 HEAD 커밋 ID가 사용됩니다.
- GitLab의 경우 커밋 ID, pull 요청 ID, 분기 이름, 태그 이름 또는 참조와 커밋 ID를 입력합니다. 자세한 내용은 를 사용한 소스 버전 샘플 AWS CodeBuild 섹션을 참조하세요.
- Bitbucket의 경우, 빌드하려는 소스 코드의 버전에 해당하는 커밋 ID, 분기 이름 또는 태그 이름입니다. 분기 이름이 지정되어 있으면 분기의 HEAD 커밋 ID가 사용됩니다. 지정되지 않은 경우 기본 분기의 HEAD 커밋 ID가 사용됩니다.
- Amazon S3의 경우 사용할 빌드 입력 ZIP 파일을 나타내는 객체의 버전 ID입니다.

빌드 수준에서 `sourceVersion`이 지정되어 있으면 (프로젝트 수준에서) 그 버전이 이 `sourceVersion`보다 우선합니다. 자세한 내용은 를 사용한 소스 버전 샘플 AWS CodeBuild 섹션을 참조하세요.

artifacts

필수 사항입니다. 이 빌드 프로젝트의 출력 아티팩트 설정에 대한 정보가 들어 있는 ProjectArtifacts 객체입니다. `artifacts` 객체를 추가한 후에는 를 사용해 아티팩트를 최대 12개까지 더 추가할 수 있습니다. 이러한 설정에는 다음이 포함됩니다.

artifacts/type

필수 사항입니다. 빌드 출력 결과물의 유형입니다. 유효값은 다음과 같습니다.

- `CODEPIPELINE`
- `NO_ARTIFACTS`
- `S3`

artifacts/location

`S3` 아티팩트 유형에만 사용됩니다. 다른 아티팩트 유형에는 사용되지 않습니다.

사전 요구 사항에서 생성했거나 식별한 출력 버킷의 이름입니다.

artifacts/path

`S3` 아티팩트 유형에만 사용됩니다. 다른 아티팩트 유형에는 사용되지 않습니다.

ZIP 파일 또는 폴더를 배치할 출력 버킷의 경로입니다. `path`의 값을 지정하지 않으면 CodeBuild가 `namespaceType`(지정된 경우) 및 `name`을 사용하여 빌드 출력 ZIP 파일이나 폴더의 경로 및 이름을 결정합니다. 예를 들어 `path`에 대해 `MyPath`, `name`에 대해 `MyArtifact.zip`을 를 지정하면 경로와 이름은 `MyPath/MyArtifact.zip`이 됩니다.

artifacts/namespaceType

`S3` 아티팩트 유형에만 사용됩니다. 다른 아티팩트 유형에는 사용되지 않습니다.

빌드 출력 ZIP 파일 또는 폴더의 네임스페이스입니다. 유효한 값에는 `BUILD_ID` 및 `NONE`(이)가 있습니다. 빌드 출력 ZIP 파일 또는 폴더의 경로에 빌드 ID를 삽입하려면 `BUILD_ID`를 사용하고 그렇지 않은 경우 `NONE`을 사용합니다. `namespaceType`의 값을 지정하지 않으면 CodeBuild가 `path`(지정된 경우) 및 `name`을 사용하여 빌드 출력 ZIP 파일이나 폴더의 경로 및 이름을 결정합니다. 예를 들어 `path`에 대해 `MyPath`, `namespaceType`에 대해 `BUILD_ID`, `name`에 대해 `MyArtifact.zip`을 지정하면 경로와 이름은 `MyPath/`*`build-ID`*`/MyArtifact.zip`이 됩니다.

artifacts/name

`S3` 아티팩트 유형에만 사용됩니다. 다른 아티팩트 유형에는 사용되지 않습니다.

`location` 안에 있는 빌드 출력 ZIP 파일 또는 폴더의 경로 및 이름입니다. 예를 들어 `path`에 대해 `MyPath`, `name`에 대해 `MyArtifact.zip`을 를 지정하면 경로와 이름은 `MyPath/MyArtifact.zip`이 됩니다.

artifacts/overrideArtifactName

S3 아티팩트 유형에만 사용됩니다. 다른 아티팩트 유형에는 사용되지 않습니다.

선택 사항. `true`로 설정할 경우 buildspec 파일의 `artifacts` 블록에서 지정한 이름이 `name`을 재정의합니다. 자세한 내용은 CodeBuild의 빌드 사양 참조 섹션을 참조하세요.

artifacts/packaging

S3 아티팩트 유형에만 사용됩니다. 다른 아티팩트 유형에는 사용되지 않습니다.

선택 사항. 아티팩트를 패키징하는 방법을 지정합니다. 허용되는 값:

NONE

빌드 아티팩트가 포함된 폴더를 생성합니다. 이것이 기본값입니다.

ZIP

빌드 아티팩트가 포함된 ZIP 파일을 생성합니다.

secondaryArtifacts

선택 사항. 빌드 프로젝트의 보조 아티팩트 설정에 대한 정보가 들어 있는 ProjectArtifacts 객체의 배열입니다. 보조 아티팩트는 최대 12개까지 추가할 수 있습니다. `secondaryArtifacts`는 객체에서 사용하는 것과 동일한 수의 설정을 사용합니다.

cache

필수 사항입니다. 이 빌드 프로젝트의 캐시 설정에 대한 정보가 들어 있는 ProjectCache 객체입니다. 자세한 내용은 캐시 빌드 섹션을 참조하세요.

환경

필수 사항입니다. 이 프로젝트의 빌드 환경 설정에 대한 정보가 들어 있는 ProjectEnvironment 객체입니다. 이러한 설정은 다음과 같습니다.

environment/type

필수 사항입니다. 빌드 환경의 유형입니다. 자세한 내용은 CodeBuild API 참조의 유형을 참조하세요.

environment/image

필수 사항입니다. 이 빌드 환경에서 사용하는 도커 이미지 식별자입니다. 일반적으로 이 식별자는 *image-name*:*tag*로 표현됩니다. 예를 들어 CodeBuild가 도커 이미지를 관리하는 데 사용하는 Docker 리포지토리에서 이 값은 aws/codebuild/standard:5.0이 될 수 있습니다. Docker Hub에서는 maven:3.3.9-jdk-8입니다. Amazon ECR에서는 *account-id*.dkr.ecr.*region-id*.amazonaws.com/*your-Amazon-ECR-repo-name*:*tag*입니다. 자세한 내용은 CodeBuild가 제공하는 도커 이미지 섹션을 참조하세요.

environment/computeType

필수 사항입니다. 이 빌드 환경에서 사용하는 컴퓨팅 리소스를 지정합니다. 자세한 내용은 CodeBuild API 참조의 computeType을 참조하세요.

environment/certificate

선택 사항. PEM 인코딩된 인증서를 포함하는 Amazon S3 버킷, 경로 접두사 및 객체 키의 ARN입니다. 객체 키는 단지 .pem 파일일 수도 있고 PEM 인코딩된 인증서를 포함하는 .zip 파일일 수도 있습니다. 예를 들어 Amazon S3 버킷 이름이 *<my-bucket>*이고 경로 접두사는 *<cert>*이고 객체 키 이름이 *<certificate.pem>*인 경우 certificate에 허용되는 형식은 *<my-bucket/cert/certificate.pem>* 또는 arn:aws:s3:::*<my-bucket/cert/certificate.pem>*입니다.

environment/environmentVariables

선택 사항. 이 빌드 환경에 지정하려는 환경 변수를 포함하는 EnvironmentVariable 객체의 배열입니다. 각 환경 변수는 name, value, type의 name, value, type을 포함하는 객체로 표현됩니다.

콘솔과 AWS CLI 사용자는 모든 환경 변수를 볼 수 있습니다. 환경 변수의 가시성에 대한 문제가 없으면 name 및 value를 설정하고 type을 PLAINTEXT로 설정합니다.

액세스 키 ID, AWS 보안 AWS 액세스 키 또는 암호와 같은 민감한 값을 가진 환경 변수를 Amazon EC2 Systems Manager 파라미터 스토어 또는에 파라미터로 저장하는 것이 좋습니다 AWS Secrets Manager. name의 저장된 파라미터의 경우 CodeBuild가 참조할 식별자를 설정합니다.

Amazon EC2 Systems Manager Parameter Store를 사용하는 경우, value에 대해 파라미터 이름을 Parameter Store에 저장된 것으로 설정합니다. type를 PARAMETER_STORE으로 설정합니다. 이름이 /CodeBuild/dockerLoginPassword인 파라미터를 예제로 사용하여 name을 LOGIN_PASSWORD로 설정합니다. value을 /CodeBuild/dockerLoginPassword으로 설정합니다. type를 PARAMETER_STORE로 설정합니다.

Important

Amazon EC2 Systems Manager Parameter Store를 사용하는 경우 `/CodeBuild/`로 시작하는 파라미터 이름(예: `/CodeBuild/dockerLoginPassword`)으로 파라미터를 저장하는 것이 좋습니다. CodeBuild 콘솔을 사용하여 Amazon EC2 Systems Manager에서 파라미터를 생성할 수 있습니다. 파라미터 생성을 선택하고 대화 상자에 표시되는 지시에 따릅니다. (그 대화 상자에서 KMS 키에 대해 계정에 있는 AWS KMS 키의 ARN을 지정할 수 있습니다. Amazon EC2 Systems Manager는이 키를 사용하여 저장 중에 파라미터의 값을 암호화하고 검색 중에 복호화합니다.) CodeBuild 콘솔을 사용하여 파라미터를 생성하는 경우 콘솔은 이름이 `/CodeBuild/`인 파라미터가 저장되면 시작합니다. 자세한 내용은 Amazon EC2 Systems Manager 사용 설명서의 Systems Manager Parameter Store 및 Systems Manager Parameter Store 콘솔 연습을 참조하세요.

빌드 프로젝트가 Amazon EC2 Systems Manager Parameter Store에 저장된 파라미터를 참조하는 경우 해당 빌드 프로젝트의 서비스 역할은 `ssm:GetParameters` 작업을 허용해야 합니다. 앞에서 새 서비스 역할을 선택한 경우에는 CodeBuild는 빌드 프로젝트의 기본 서비스 역할에 이 작업을 포함합니다. Existing service role(기존 서비스 역할)을 선택한 경우에는 이 작업을 서비스 역할에 별도로 포함해야 합니다.

빌드 프로젝트가 `/CodeBuild/`로 시작되지 않는 파라미터 이름으로 Amazon EC2 Systems Manager Parameter Store에 저장된 파라미터를 참조하는 경우 새 서비스 역할을 선택하면 `/CodeBuild/`로 시작하지 않는 파라미터 이름에 액세스할 수 있도록 해당 서비스 역할을 업데이트해야 합니다. 이는 서비스 역할이 `/CodeBuild/`로 시작하는 파라미터 이름에만 액세스할 수 있기 때문입니다.

새 서비스 역할을 선택하면 Amazon EC2 Systems Manager Parameter Store에 있는 `/CodeBuild/` 네임스페이스의 모든 파라미터를 해독할 권한이 서비스 역할에 포함됩니다.

사용자가 설정한 환경 변수는 기존 환경 변수를 대체합니다. 예를 들어 도커 이미지에 값이 `my_value`인 `MY_VAR`이라는 환경 변수가 이미 포함되어 있는데, 사용자가 `MY_VAR` 환경 변수의 값을 `other_value`로 설정하면, `my_value`가 `other_value`로 바뀝니다. 마찬가지로, 도커 이미지에 값이 `/usr/local/sbin:/usr/local/bin`인 `PATH`라는 환경 변수가 이미 포함되어 있는데, 사용자가 `PATH` 환경 변수의 값을 `$PATH:/usr/share/ant/bin`으로 설정하면, `/usr/local/sbin:/usr/local/bin`이 `$PATH:/usr/share/ant/bin` 리터럴 값으로 바뀝니다.

`CODEBUILD_`로 시작하는 이름으로 환경 변수를 설정하지 마십시오. 이 접두사는 내부 전용으로 예약되어 있습니다.

여러 위치에서 동일한 이름의 환경 변수가 정의되는 경우, 다음과 같이 값이 결정됩니다.

- 시작 빌드 작업 호출의 값이 가장 높은 우선 순위를 갖습니다.
- 빌드 프로젝트 정의의 값이 다음 우선 순위를 갖습니다.

- buildspec 선언의 값이 가장 낮은 우선 순위를 갖습니다.

Secrets Manager를 `value` 사용하는 경우 파라미터 이름을 Secrets Manager에 저장된 항목으로 설정합니다. `type`를 `SECRETS_MANAGER`으로 설정합니다. 이름이 `/CodeBuild/dockerLoginPassword`인 보안 암호를 예제로 사용하여 `name`을 `LOGIN_PASSWORD`로 설정합니다. `value`을 `/CodeBuild/dockerLoginPassword`으로 설정합니다. `type`를 `SECRETS_MANAGER`로 설정합니다.

> **Important**
>
> Secrets Manager를 사용하는 경우 이름이 `/CodeBuild/`로 시작하는 암호를 저장하는 것이 좋습니다(예: `/CodeBuild/dockerLoginPassword`). 자세한 내용은 AWS Secrets Manager사용 설명서의 AWS Secrets Manager 이란? 섹션을 참조하세요.
> 빌드 프로젝트가 Secrets Manager에 저장된 암호를 참조하는 경우 해당 빌드 프로젝트의 서비스 역할은 `secretsmanager:GetSecretValue` 작업을 허용해야 합니다. 앞에서 새 서비스 역할을 선택한 경우에는 CodeBuild는 빌드 프로젝트의 기본 서비스 역할에 이 작업을 포함합니다. Existing service role(기존 서비스 역할)을 선택한 경우에는 이 작업을 서비스 역할에 별도로 포함해야 합니다.
> 빌드 프로젝트가 `/CodeBuild/`로 시작되지 않는 보안 암호 이름으로 Secrets Manager에 저장된 암호를 참조하는 경우 새 서비스 역할을 선택하면 `/CodeBuild/`로 시작하지 않는 보안 암호 이름에 액세스할 수 있도록 해당 서비스 역할을 업데이트해야 합니다. 이는 서비스 역할이 `/CodeBuild/`로 시작하는 암호 이름에만 액세스할 수 있기 때문입니다.
> 새 서비스 역할을 선택하면 에 있는 `/CodeBuild/` 네임스페이스의 모든 암호를 해독할 권한이 서비스 역할에 포함됩니다.

environment/registryCredential

선택 사항. 프라이빗 Docker 레지스트리에 대한 액세스를 제공하는 보안 인증을 지정하는 RegistryCredential 객체입니다.

environment/registryCredential/credential

AWS Managed Services를 사용하여 생성한 보안 인증의 ARN 또는 이름을 지정합니다. 현재 리전에 있는 자격 증명의 이름만 사용할 수 있습니다.

environment/registryCredential/credentialProvider

유일한 유효 값은 `SECRETS_MANAGER`입니다.

이를 설정할 경우 다음과 같이 해야 합니다.

- `imagePullCredentials`를 `SERVICE_ROLE`로 설정해야 합니다.
- 이 이미지는 큐레이트된 이미지 또는 Amazon ECR 이미지일 수 없습니다.

environment/imagePullCredentialsType

선택 사항. CodeBuild가 빌드에 이미지를 가져올 때 사용하는 보안 인증 유형입니다. 두 가지 값을 사용할 수 있습니다.

CODEBUILD

`CODEBUILD`는 CodeBuild가 자체 보안 인증을 사용하도록 지정합니다. CodeBuild 서비스 보안 주체를 신뢰하려면 Amazon ECR 리포지토리 정책을 편집해야 합니다.

SERVICE_ROLE

CodeBuild가 해당 빌드 프로젝트의 서비스 역할을 사용하도록 지정합니다.

교차 계정 또는 프라이빗 레지스트리 이미지를 사용할 경우 `SERVICE_ROLE` 자격 증명을 사용해야 합니다. CodeBuild 큐레이트 이미지를 사용할 경우 `CODEBUILD` 보안 인증을 사용해야 합니다.

environment/privilegedMode

이 빌드 프로젝트를 사용하여 도커 이미지를 빌드하려는 경우에만 `true`로 설정합니다. 그렇지 않으면 Docker 데몬과 상호 작용을 시도하는 모든 연결된 빌드가 실패합니다. 또한 빌드가 상호 작용할 수 있도록 Docker 데몬을 시작해야 합니다. 이를 수행하는 한 가지 방법은 다음 빌드 명령을 실행하여 buildspec 파일의 `install` 단계에서 Docker 데몬을 초기화하는 것입니다. 지정한 빌드 환경 이미지가 Docker 지원을 통해 CodeBuild에서 제공하지 않은 경우 이 명령을 실행하지 마세요.

> **Note**
>
> 기본적으로 비 VPC 빌드에는 Docker 데몬이 활성화됩니다. VPC 빌드에 Docker 컨테이너를 사용하려면 Docker Docs 웹 사이트의 런타임 권한 및 Linux 기능을 참조하고 권한 부여 모드를 활성화합니다. 또한 Windows는 권한 모드를 지원하지 않습니다.

```
- nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --
host=tcp://127.0.0.1:2375 --storage-driver=overlay2 &
- timeout 15 sh -c "until docker info; do echo .; sleep 1; done"
```

serviceRole

필수 사항입니다. CodeBuild가 사용자 대신 서비스와 상호 작용하기 위해 사용하는 서비스 역할의 ARN입니다(예: `arn:aws:iam::account-id:role/role-name`).

autoRetryLimit

선택 사항. 빌드 실패 후 추가 자동 재시도 횟수입니다. 예를 들어 자동 재시도 제한이 2로 설정된 경우 CodeBuild는 `RetryBuild` API를 호출하여 빌드를 추가로 최대 2회 자동 재시도합니다.

timeoutInMinutes

선택 사항. 빌드가 완료되지 않는 경우 CodeBuild가 해당 빌드를 중지하는 시간(5분에서 2160분(36시간) 사이)입니다. 지정하지 않을 경우 기본값인 60을 사용합니다. 시간 제한으로 인해 CodeBuild가 빌드를 중지했는지 및 중지한 시간을 확인하려면 `batch-get-builds` 명령을 실행합니다. 빌드가 중지되었는지 확인하려면 출력에서 `FAILED`의 `buildStatus` 값을 살펴봅니다. 빌드가 시간 초과된 시간을 확인하려면 출력에서 `TIMED_OUT`의 `phaseStatus`와 연결된 `endTime` 값을 살펴봅니다.

queuedTimeoutInMinutes

선택 사항. 5~480(8시간) 사이의 분 수로, CodeBuild는 아직 대기열에 있는 경우 빌드를 중지합니다. 지정하지 않을 경우 기본값인 60을 사용합니다.

encryptionKey

선택 사항. CodeBuild가 빌드 출력을 암호화하는 데 AWS KMS key 사용하는의 별칭 또는 ARN입니다. 별칭을 지정하는 경우 `arn:aws:kms:region-ID:account-ID:key/key-ID` 형식을 사용하고, 별칭이 있는 경우 `alias/key-alias` 형식을 사용합니다. 지정하지 않으면 Amazon S3용 AWS관리형 KMS 키가 사용됩니다.

tags

선택 사항. 이 빌드 프로젝트와 연결할 태그를 제공하는 Tag 객체 배열입니다. 최대 50개의 태그를 지정할 수 있습니다. 이러한 태그는 CodeBuild 빌드 프로젝트 태그를 지원하는 모든 AWS 서비스에서 사용할 수 있습니다. 각 태그는 `key`와 `value`가 있는 객체로 표현됩니다.

vpcConfig

선택 사항. 프로젝트의 VPC 구성에 대한 정보가 포함된 VpcConfig 객체입니다. 자세한 내용은 Amazon Virtual Private Cloud AWS CodeBuild 와 함께 사용 섹션을 참조하세요.

이러한 속성은 다음과 같습니다.

vpcId

필수 사항입니다. CodeBuild가 사용하는 VPC ID입니다. 리전의 모든 VPC ID 목록을 가져오려면 다음 명령을 실행합니다.

```
aws ec2 describe-vpcs --region <region-ID>
```

subnets

필수 사항입니다. CodeBuild에서 사용되는 리소스가 포함된 서브넷 ID의 배열입니다. 이 ID를 얻으려면 다음 명령을 실행합니다.

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=<vpc-id>" --region <region-ID>
```

securityGroupIds

필수 사항입니다. CodeBuild가 VPC의 리소스에 대한 액세스를 허용하기 위해 사용하는 보안 그룹 ID 배열입니다. 이 ID를 얻으려면 다음 명령을 실행합니다.

```
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=<vpc-id>" --<region-ID>
```

badgeEnabled

선택 사항. CodeBuild 프로젝트에 빌드 배지를 포함할지 여부를 지정합니다. 빌드 배지를 활성화하려면 `true`로 설정하고, 그렇지 않으면 `false`로 설정합니다. 자세한 내용은 CodeBuild의 빌드 배지 샘플 섹션을 참조하세요.

logsConfig

이 빌드의 로그가 있는 위치에 대한 정보가 들어 있는 LogsConfig 객체입니다.

logsConfig/cloudWatchLogs

로그를 CloudWatch Logs로 푸시하는 것과 관련된 정보가 들어 있는 CloudWatchLogsConfig 객체입니다.

logsConfig/s3Logs

로그를 Amazon S3로 푸시하는 방법에 대한 정보가 들어 있는 S3LogsConfig 객체입니다.

fileSystemLocations

선택 사항. Amazon EFS 구성에 대한 정보가 포함된 ProjectFileSystemsLocation 객체의 배열입니다.

buildBatchConfig

선택 사항. `buildBatchConfig` 객체는 프로젝트에 대한 배치 빌드 구성 정보를 포함하는 ProjectBuildBatchConfig 구조입니다.

buildBatchConfig/serviceRole

배치 빌드 프로젝트에 대한 서비스 역할 ARN입니다.

buildBatchConfig/combineArtifacts

배치 빌드의 빌드 아티팩트를 단일 아티팩트 위치로 결합할지 여부를 지정하는 부울 값입니다.

buildBatchConfig/restrictions/maximumBuildsAllowed

허용되는 최대 빌드 수입니다.

buildBatchConfig/restrictions/computeTypesAllowed

배치 빌드에 허용되는 컴퓨팅 유형을 지정하는 문자열 배열입니다. 이러한 값은 빌드 환경 컴퓨팅 유형를 참조하세요.

buildBatchConfig/restrictions/fleetsAllowed

배치 빌드에 허용되는 플릿을 지정하는 문자열 배열입니다. 자세한 내용은 예약 용량 플릿에서 빌드 실행을 참조하세요.

buildBatchConfig/timeoutInMinutes

배치 빌드를 완료해야 하는 최대 시간(분)입니다.

buildBatchConfig/batchReportMode

배치 빌드를 위해 빌드 상태 보고서를 소스 제공자에게 보내는 방법을 지정합니다. 유효한 값으로는 다음이 포함됩니다.

`REPORT_AGGREGATED_BATCH`

(기본값) 모든 빌드 상태를 단일 상태 보고서로 집계합니다.

`REPORT_INDIVIDUAL_BUILDS`

각 개별 빌드에 대해 별도의 상태 보고서를 보냅니다.

concurrentBuildLimit

이 프로젝트에 허용된 최대 동시 실행 빌드 수입니다.

현재 빌드 수가 이 한도 이하인 경우에만 새 빌드가 시작됩니다. 현재 빌드 수가 이 한도에 도달하면 새 빌드가 제한되고 실행되지 않습니다.

프로젝트 생성

프로젝트를 생성하려면 JSON 파일을 전달하여 **create-project** 명령을 다시 실행합니다.

```
aws codebuild create-project --cli-input-json file://<json-file>
```

성공하면 Project 객체의 JSON 표현이 콘솔 출력에 나타납니다. 이 데이터의 예는 CreateProject 응답 구문을 참조하세요.

빌드 프로젝트 이름을 제외한 모든 빌드 프로젝트 설정은 나중에 변경할 수 있습니다. 자세한 내용은 빌드 프로젝트 설정 변경(AWS CLI) 섹션을 참조하세요.

빌드 실행을 시작하려면 빌드 실행(AWS CLI) 단원을 참조하십시오.

소스 코드가 GitHub 리포지토리에 저장되고 코드 변경이 리포지토리로 푸시될 때마다 CodeBuild가 자동으로 소스 코드를 다시 빌드하게 하려면 빌드 실행 자동 시작(AWS CLI) 섹션을 참조하세요.

빌드 프로젝트(AWS SDKs) 생성

를 SDK와 AWS CodeBuild 함께 사용하는 방법에 대한 자세한 내용은 섹션을 참조하세요AWS SDKs 및 도구 참조. AWS SDKs

빌드 프로젝트 생성(CloudFormation)

AWS CodeBuild 와 함께를 사용하는 방법에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 CodeBuild용 CloudFormation 템플릿을 CloudFormation참조하세요.

알림 규칙 생성

알림 규칙을 사용하여 빌드 성공 및 실패와 같은 중요한 변경 사항이 발생할 경우 사용자에게 알릴 수 있습니다. 알림 규칙은 알림을 보내는 데 사용되는 이벤트와 Amazon SNS 주제를 모두 지정합니다. 자세한 내용은 알림이란 무엇입니까?를 참조하세요.

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

알림 규칙을 생성하려면 (콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codebuild/ CodeBuild 콘솔을 엽니다.
2. 빌드, 빌드 프로젝트를 차례로 선택한 다음 알림을 추가할 빌드 프로젝트를 선택합니다.
3. 빌드 프로젝트 페이지에서 알림을 선택하고 Create notification rule(알림 규칙 생성)을 선택합니다. 빌드 프로젝트의 설정 페이지로 이동하여 Create notification rule(알림 규칙 생성)를 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 CodeBuild 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 빌드 프로젝트의 알림 규칙 이벤트를 참조하십시오.
7. 대상에서 다음 중 하나를 수행합니다.

   - 알림과 함께 사용할 리소스를 이미 구성한 경우 대상 유형 선택에서 채팅 애플리케이션의 Amazon Q Developer(Slack) 또는 SNS 주제를 선택합니다. 대상 선택에서 클라이언트의 이름(채팅 애플리케이션에서 Amazon Q Developer로 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 규칙을 만드는 중에는 채팅 애플리케이션 클라이언트에 Amazon Q Developer를 만들 수 없습니다. 채팅 애플리케이션에서 Amazon Q Developer(Slack)를 선택하면 채팅 애플리케이션에서 Amazon Q Developer의 클라이언트를 구성하라는 버튼이 표시됩니다. 이 옵션을 선택하면 채팅 애플리케이션 콘솔에 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 채팅룸에 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 codestarnotifications 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 AWS 계정의 *MyBuildProject*라는 빌드 프로젝트에 **MyNotificationRule** 대한 라는 알림 규칙을 보여줍니다. 빌드가 성공적일 때 *codestar-notifications-MyNotificationTopic*이라는 Amazon SNS 주제에 전체 세부 정보 유형과 함께 알림이 전송됩니다.

```
{
    "Name": "MyNotificationRule",
    "EventTypeIds": [
        "codebuild-project-build-state-succeeded"
    ],
    "Resource": "arn:aws:codebuild:us-east-2:123456789012:MyBuildProject",
    "Targets": [
        {
            "TargetType": "SNS",
```

```
            "TargetAddress": "arn:aws:sns:us-east-2:123456789012:codestar-
notifications-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 codestarnotifications 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"
}
```

에서 빌드 프로젝트 설정 변경 AWS CodeBuild

AWS CodeBuild 콘솔 AWS CLI, 또는 AWS SDKs를 사용하여 빌드 프로젝트의 설정을 변경할 수 있습니다.

빌드 프로젝트에 테스트 보고를 추가하는 경우 IAM 역할에 테스트 보고서 권한에서 설명하는 사용 권한이 있는지 확인합니다.

주제

- 빌드 프로젝트 설정 변경(콘솔)
- 빌드 프로젝트 설정 변경(AWS CLI)
- 빌드 프로젝트의 설정 변경(AWS SDKs)

빌드 프로젝트 설정 변경(콘솔)

빌드 프로젝트의 설정을 변경하려면 다음 절차에 수행하세요.

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 프로젝트 빌드를 선택합니다.
3. 다음 중 하나를 수행하세요.
   - 변경하려는 빌드 프로젝트의 링크를 선택한 다음 Edit project(프로젝트 편집)를 선택합니다.
   - 변경하려는 빌드 프로젝트 옆에 있는 라디오 버튼을 선택하고 세부 정보 보기를 선택한 후 빌드 세부 정보를 선택합니다.

다음 섹션을 수정할 수 있습니다.

Sections

- 프로젝트 구성
- 소스
- 환경
- Buildspec
- 배치 구성
- 아티팩트
- 로그

프로젝트 구성

프로젝트 구성 섹션에서 편집을 선택합니다. 변경이 완료되면 구성 업데이트를 선택하여 새 구성을 저장합니다.

다음 속성을 수정할 수 있습니다.

설명

선택적으로 빌드 프로젝트에 대한 설명을 입력하여 다른 사용자가 이 프로젝트의 용도를 이해하도록 도울 수 있습니다.

빌드 배지

프로젝트의 빌드 상태를 표시하고 삽입 가능하게 하려면 Enable build badge(빌드 배치 활성화)를 선택합니다. 자세한 내용은 빌드 배지 샘플 섹션을 참조하세요.

> **Note**
>
> 소스 공급자가 Amazon S3인 경우 빌드 배지가 적용되지 않습니다.

동시 빌드 제한 활성화

이 프로젝트의 동시 빌드 수를 제한하려면 다음 단계를 수행합니다.

1. 이 프로젝트에서 시작할 수 있는 동시 빌드 수 제한을 선택합니다.
2. 동시 빌드 제한에 이 프로젝트에 허용되는 최대 동시 빌드 수를 입력합니다. 이 제한은 계정에 설정된 동시 빌드 제한보다 클 수 없습니다. 계정 제한보다 큰 숫자를 입력하려고 하면 오류 메시지가 표시됩니다.

현재 빌드 수가 이 한도 이하인 경우에만 새 빌드가 시작됩니다. 현재 빌드 수가 이 한도에 도달하면 새 빌드가 제한되고 실행되지 않습니다.

퍼블릭 빌드 액세스 활성화

AWS 계정에 액세스할 수 없는 사용자를 포함하여 프로젝트의 빌드 결과를 퍼블릭이 사용할 수 있도록 하려면 퍼블릭 빌드 액세스 활성화를 선택하고 빌드 결과를 퍼블릭으로 설정할지 확인합니다. 퍼블릭 빌드 프로젝트에는 다음과 같은 속성이 사용됩니다.

퍼블릭 빌드 서비스 역할

CodeBuild에서 새 서비스 역할을 생성하도록 하려면 새 서비스 역할을 선택하고, 기존 서비스 역할을 사용하려면 기존 서비스 역할을 선택합니다.

퍼블릭 빌드 서비스 역할을 사용하면 CodeBuild가 프로젝트 빌드에 대한 CloudWatch Logs를 읽고 Amazon S3 아티팩트를 다운로드할 수 있습니다. 이 역할은 프로젝트의 빌드 로그와 아티팩트를 퍼블릭으로 지정하는 데 필요합니다.

서비스 역할

새 서비스 역할 또는 기존 서비스 역할의 이름을 입력합니다.

프로젝트의 빌드 결과를 프라이빗으로 설정하려면 퍼블릭 빌드 액세스 활성화를 선택 취소합니다.

자세한 내용은 퍼블릭 빌드 프로젝트 URL 가져오기 섹션을 참조하세요.

> **Warning**
>
> 프로젝트의 빌드 결과를 퍼블릭으로 유지할 때는 다음에 유의해야 합니다.

- 프로젝트가 프라이빗일 때 실행된 빌드를 포함하여 프로젝트의 모든 빌드 결과, 로그, 아티팩트는 누구나 이용할 수 있습니다.
- 모든 빌드 로그와 아티팩트는 누구나 이용할 수 있습니다. 환경 변수, 소스 코드 및 기타 중요한 정보가 빌드 로그와 아티팩트에 출력되었을 수 있습니다. 빌드 로그에 출력되는 정보에 주의해야 합니다. 몇 가지 모범 사례는 다음과 같습니다.
  - 환경 변수에 민감한 값, 특히 AWS 액세스 키 IDs 및 보안 액세스 키를 저장하지 마십시오. Amazon EC2 Systems Manager 파라미터 스토어 또는를 사용하여 민감한 값을 저장하는 AWS Secrets Manager 것이 좋습니다.
  - webhook를 최대한 안전하게 보호하려면 webhook 사용 모범 사례에 따라 빌드를 트리거할 수 있는 엔터티를 제한하고, buildspec을 프로젝트 자체에 저장하지 마세요.
- 악의적인 사용자는 퍼블릭 빌드를 사용하여 악성 아티팩트를 배포할 수 있습니다. 프로젝트 관리자는 모든 풀 요청을 검토하여 풀 요청이 합법적인 변경인지 확인하는 것이 좋습니다. 또한 체크섬으로 모든 아티팩트의 유효성을 검사하여 올바른 아티팩트가 다운로드되고 있는지 확인하는 것이 좋습니다.

추가 정보

태그에 지원 AWS 서비스에서 사용할 태그의 이름과 값을 입력합니다. [Add row]를 사용하여 태그를 추가합니다. 최대 50개의 태그를 추가할 수 있습니다.

소스

소스 섹션에서 편집을 선택합니다. 변경이 완료되면 구성 업데이트를 선택하여 새 구성을 저장합니다.

다음 속성을 수정할 수 있습니다.

소스 공급자

소스 코드 공급자 유형을 선택합니다. 다음 목록을 사용하여 소스 공급자에 알맞은 유형을 선택합니다.

> **Note**
>
> CodeBuild에서는 Bitbucket Server를 지원하지 않습니다.

Amazon S3

버킷

소스 코드가 포함된 입력 버킷의 이름을 선택합니다.

S3 객체 키 또는 S3 폴더

소스 코드가 포함된 ZIP 파일의 이름 또는 폴더 경로를 입력합니다. S3 버킷의 모든 항목을 다운로드하려면 슬래시(/)를 입력합니다.

소스 버전

입력 파일의 빌드를 나타내는 객체의 버전 ID를 입력합니다. 자세한 내용은 단원을 참조하십시오를 사용한 소스 버전 샘플 AWS CodeBuild.

CodeCommit

리포지토리

사용할 리포지토리를 선택합니다.

참조 유형

분기, Git 태그 또는 커밋 ID를 선택하여 소스 코드 버전을 지정합니다. 자세한 내용은 를 사용한 소스 버전 샘플 AWS CodeBuild 섹션을 참조하세요.

> **Note**
>
> `811dd1ba1aba14473856cee38308caed7190c0d` 또는 `5392f7`과 같이 커밋 ID처럼 보이지 않는 Git 분기 이름을 선택하는 것이 좋습니다. 이렇게 하면 실제 커밋과 Git 체크아웃이 충돌하는 것을 방지할 수 있습니다.

Git clone 깊이

이력이 지정된 커밋 수로 잘린 부분 복제를 생성하려면 선택합니다. 전체 복제가 필요할 경우 전체를 선택합니다.

Git 하위 모듈

리포지토리에 Git 하위 모듈을 포함하려면 Use Git submodules(Git 하위 모듈 사용)를 선택합니다.

Bitbucket

자격 증명

기본 소스 자격 증명 또는 사용자 지정 소스 자격 증명을 선택하고 지침에 따라 기본 소스 자격 증명을 관리하거나 소스 자격 증명을 사용자 지정합니다.

연결 유형

AWS CodeConnections, OAuth, 앱 암호 또는 개인 액세스 토큰을 선택하여 CodeBuild에 연결합니다.

Connection

Bitbucket 연결 또는 Secrets Manager 암호를 선택하여 지정된 연결 유형을 통해 연결합니다.

리포지토리

내 Bitbucket 계정의 리포지토리 또는 퍼블릭 리포지토리를 선택하고 리포지토리 URL을 입력합니다.

소스 버전

분기, 커밋 ID, 태그 또는 참조와 커밋 ID를 입력합니다. 자세한 내용은 를 사용한 소스 버전 샘플 AWS CodeBuild 섹션을 참조하세요.

> **Note**
>
> `811dd1ba1aba14473856cee38308caed7190c0d` 또는 `5392f7`과 같이 커밋 ID처럼 보이지 않는 Git 분기 이름을 선택하는 것이 좋습니다. 이렇게 하면 실제 커밋과 Git 체크아웃이 충돌하는 것을 방지할 수 있습니다.

Git clone 깊이

Git clone 깊이를 선택하면 이력이 지정된 커밋 수로 잘린 부분 복제가 생성됩니다. 전체 복제가 필요할 경우 전체를 선택합니다.

Git 하위 모듈

리포지토리에 Git 하위 모듈을 포함하려면 Use Git submodules(Git 하위 모듈 사용)를 선택합니다.

빌드 상태

빌드 시작 및 완료 상태가 소스 공급자에게 보고되도록 하려면 빌드가 시작되고 완료될 때 소스 공급자에게 빌드 상태 보고를 선택합니다.

소스 공급자에게 빌드 상태를 보고하려면 소스 공급자와 연결된 사용자에게 리포지토리에 대한 쓰기 권한이 있어야 합니다. 사용자에게 쓰기 권한이 없는 경우 빌드 상태를 업데이트할 수 없습니다. 자세한 내용은 소스 공급자 액세스 섹션을 참조하세요.

상태 컨텍스트의 경우 Bitbucket 커밋 상태의 `name` 파라미터에 사용할 값을 입력합니다. 자세한 내용은 Bitbucket API 설명서의 빌드를 참조하세요.

대상 URL에 Bitbucket 커밋 상태의 `url` 파라미터에 사용할 값을 입력합니다. 자세한 내용은 Bitbucket API 설명서의 빌드를 참조하세요.

Webhook에 의해 트리거된 빌드의 상태는 항상 소스 공급자에게 보고됩니다. 콘솔 또는 API 직접 호출에서 시작된 빌드의 상태를 소스 공급자에게 보고하려면 이 설정을 선택해야 합니다.

프로젝트 빌드가 webhook에 의해 트리거되는 경우 이 설정의 변경 사항을 적용하려면 새 커밋을 리포지토리에 푸시해야 합니다.

코드 변경 사항이 이 리포지토리에 푸시될 때마다 CodeBuild가 소스 코드를 빌드하도록 하려면 기본 소스 webhook 이벤트에서 코드 변경이 이 리포지토리에 푸시될 때마다 다시 빌드를 선택합니다. webhook 및 필터 그룹에 대한 자세한 내용은 Bitbucket Webhook 이벤트 섹션을 참조하세요.

GitHub

자격 증명

기본 소스 자격 증명 또는 사용자 지정 소스 자격 증명을 선택하고 지침에 따라 기본 소스 자격 증명을 관리하거나 소스 자격 증명을 사용자 지정합니다.

연결 유형

GitHub 앱, OAuth 또는 개인 액세스 토큰을 선택하여 CodeBuild에 연결합니다.

Connection

GitHub 연결 또는 Secrets Manager 보안 암호를 선택하여 지정된 연결 유형을 통해 연결합니다.

리포지토리

내 GitHub 계정의 리포지토리, 퍼블릭 리포지토리 또는 GitHub 범위 웹후크를 선택하고 리포지토리 URL을 입력합니다.

소스 버전

분기, 커밋 ID, 태그 또는 참조와 커밋 ID를 입력합니다. 자세한 내용은 를 사용한 소스 버전 샘플 AWS CodeBuild 섹션을 참조하세요.

> **Note**
>
> `811dd1ba1aba14473856cee38308caed7190c0d` 또는 `5392f7`과 같이 커밋 ID처럼 보이지 않는 Git 분기 이름을 선택하는 것이 좋습니다. 이렇게 하면 실제 커밋과 Git 체크아웃이 충돌하는 것을 방지할 수 있습니다.

Git clone 깊이

Git clone 깊이를 선택하면 이력이 지정된 커밋 수로 잘린 부분 복제가 생성됩니다. 전체 복제가 필요할 경우 전체를 선택합니다.

Git 하위 모듈

리포지토리에 Git 하위 모듈을 포함하려면 Use Git submodules(Git 하위 모듈 사용)를 선택합니다.

빌드 상태

빌드 시작 및 완료 상태가 소스 공급자에게 보고되도록 하려면 빌드가 시작되고 완료될 때 소스 공급자에게 빌드 상태 보고를 선택합니다.

소스 공급자에게 빌드 상태를 보고하려면 소스 공급자와 연결된 사용자에게 리포지토리에 대한 쓰기 권한이 있어야 합니다. 사용자에게 쓰기 권한이 없는 경우 빌드 상태를 업데이트할 수 없습니다. 자세한 내용은 소스 공급자 액세스 섹션을 참조하세요.

상태 컨텍스트의 경우 GitHub 커밋 상태의 `context` 파라미터에 사용할 값을 입력합니다. 자세한 내용은 GitHub 개발자 안내서의 커밋 상태 생성을 참조하세요.

대상 URL에 GitHub 커밋 상태의 `target_url` 파라미터에 사용할 값을 입력합니다. 자세한 내용은 GitHub 개발자 안내서의 커밋 상태 생성을 참조하세요.

Webhook에 의해 트리거된 빌드의 상태는 항상 소스 공급자에게 보고됩니다. 콘솔 또는 API 직접 호출에서 시작된 빌드의 상태를 소스 공급자에게 보고하려면 이 설정을 선택해야 합니다.

프로젝트 빌드가 webhook에 의해 트리거되는 경우 이 설정의 변경 사항을 적용하려면 새 커밋을 리포지토리에 푸시해야 합니다.

코드 변경 사항이 이 리포지토리에 푸시될 때마다 CodeBuild가 소스 코드를 빌드하도록 하려면 기본 소스 webhook 이벤트에서 코드 변경이 이 리포지토리에 푸시될 때마다 다시 빌드를 선택합니다. webhook 및 필터 그룹에 대한 자세한 내용은 GitHub Webhook 이벤트 섹션을 참조하세요.

GitHub Enterprise Server

자격 증명

기본 소스 자격 증명 또는 사용자 지정 소스 자격 증명을 선택하고 지침에 따라 기본 소스 자격 증명을 관리하거나 소스 자격 증명을 사용자 지정합니다.

연결 유형

AWS CodeConnections CodeBuild에 연결할 또는 개인 액세스 토큰을 선택합니다.

Connection

GitHub Enterprise 연결 또는 Secrets Manager 보안 암호를 선택하여 지정된 연결 유형을 통해 연결합니다.

리포지토리

내 GitHub Enterprise 계정의 리포지토리 또는 GitHub Enterprise 범위 웹후크를 선택하고 리포지토리 URL을 입력합니다.

소스 버전

풀 요청, 분기, 커밋 ID, 태그 또는 참조와 커밋 ID를 입력합니다. 자세한 내용은 를 사용한 소스 버전 샘플 AWS CodeBuild 섹션을 참조하세요.

> **Note**
>
> `811dd1ba1aba14473856cee38308caed7190c0d` 또는 `5392f7`과 같이 커밋 ID처럼 보이지 않는 Git 분기 이름을 선택하는 것이 좋습니다. 이렇게 하면 실제 커밋과 Git 체크아웃이 충돌하는 것을 방지할 수 있습니다.

Git clone 깊이

Git clone 깊이를 선택하면 이력이 지정된 커밋 수로 잘린 부분 복제가 생성됩니다. 전체 복제가 필요할 경우 전체를 선택합니다.

Git 하위 모듈

리포지토리에 Git 하위 모듈을 포함하려면 Use Git submodules(Git 하위 모듈 사용)를 선택합니다.

빌드 상태

빌드 시작 및 완료 상태가 소스 공급자에게 보고되도록 하려면 빌드가 시작되고 완료될 때 소스 공급자에게 빌드 상태 보고를 선택합니다.

소스 공급자에게 빌드 상태를 보고하려면 소스 공급자와 연결된 사용자에게 리포지토리에 대한 쓰기 권한이 있어야 합니다. 사용자에게 쓰기 권한이 없는 경우 빌드 상태를 업데이트할 수 없습니다. 자세한 내용은 소스 공급자 액세스 섹션을 참조하세요.

상태 컨텍스트의 경우 GitHub 커밋 상태의 `context` 파라미터에 사용할 값을 입력합니다. 자세한 내용은 GitHub 개발자 안내서의 커밋 상태 생성을 참조하세요.

대상 URL에 GitHub 커밋 상태의 `target_url` 파라미터에 사용할 값을 입력합니다. 자세한 내용은 GitHub 개발자 안내서의 커밋 상태 생성을 참조하세요.

Webhook에 의해 트리거된 빌드의 상태는 항상 소스 공급자에게 보고됩니다. 콘솔 또는 API 직접 호출에서 시작된 빌드의 상태를 소스 공급자에게 보고하려면 이 설정을 선택해야 합니다.

프로젝트 빌드가 webhook에 의해 트리거되는 경우 이 설정의 변경 사항을 적용하려면 새 커밋을 리포지토리에 푸시해야 합니다.

비보안 SSL

GitHub Enterprise 프로젝트 리포지토리에 연결되어 있는 동안 SSL을 무시하려면 Enable insecure SSL(안전하지 않은 SSL 활성화)을 선택합니다.

코드 변경 사항이 이 리포지토리에 푸시될 때마다 CodeBuild가 소스 코드를 빌드하도록 하려면 기본 소스 webhook 이벤트에서 코드 변경이 이 리포지토리에 푸시될 때마다 다시 빌드를 선택합니다. webhook 및 필터 그룹에 대한 자세한 내용은 GitHub Webhook 이벤트 섹션을 참조하세요.

GitLab

자격 증명

기본 소스 자격 증명 또는 사용자 지정 소스 자격 증명을 선택하고 지침에 따라 기본 소스 자격 증명을 관리하거나 소스 자격 증명을 사용자 지정합니다.

연결 유형

AWS CodeConnections는 GitLab을 CodeBuild에 연결하는 데 사용됩니다.

Connection

연결할 GitLab 연결을 선택합니다 AWS CodeConnections.

리포지토리

사용할 리포지토리를 선택합니다.

소스 버전

pull 요청, 분기, 커밋 ID, 태그 또는 참조와 커밋 ID를 입력합니다. 자세한 내용은 를 사용한 소스 버전 샘플 AWS CodeBuild 섹션을 참조하세요.

> **Note**
>
> `811dd1ba1aba14473856cee38308caed7190c0d` 또는 `5392f7`과 같이 커밋 ID처럼 보이지 않는 Git 분기 이름을 선택하는 것이 좋습니다. 이렇게 하면 실제 커밋과 Git 체크아웃이 충돌하는 것을 방지할 수 있습니다.

Git clone 깊이

Git clone 깊이를 선택하면 이력이 지정된 커밋 수로 잘린 부분 복제가 생성됩니다. 전체 복제가 필요할 경우 전체를 선택합니다.

빌드 상태

빌드 시작 및 완료 상태가 소스 공급자에게 보고되도록 하려면 빌드가 시작되고 완료될 때 소스 공급자에게 빌드 상태 보고를 선택합니다.

소스 공급자에게 빌드 상태를 보고하려면 소스 공급자와 연결된 사용자에게 리포지토리에 대한 쓰기 권한이 있어야 합니다. 사용자에게 쓰기 권한이 없는 경우 빌드 상태를 업데이트할 수 없습니다. 자세한 내용은 소스 공급자 액세스 섹션을 참조하세요.

GitLab Self Managed

자격 증명

기본 소스 자격 증명 또는 사용자 지정 소스 자격 증명을 선택하고 지침에 따라 기본 소스 자격 증명을 관리하거나 소스 자격 증명을 사용자 지정합니다.

연결 유형

AWS CodeConnections는 GitLab 자체 관리형을 CodeBuild에 연결하는 데 사용됩니다.

Connection

연결할 GitLab 자체 관리형 연결을 선택합니다 AWS CodeConnections.

리포지토리

사용할 리포지토리를 선택합니다.

소스 버전

pull 요청, 분기, 커밋 ID, 태그 또는 참조와 커밋 ID를 입력합니다. 자세한 내용은 를 사용한 소스 버전 샘플 AWS CodeBuild 섹션을 참조하세요.

> **Note**
>
> 811dd1ba1aba14473856cee38308caed7190c0d 또는 5392f7과 같이 커밋 ID처럼 보이지 않는 Git 분기 이름을 선택하는 것이 좋습니다. 이렇게 하면 실제 커밋과 Git 체크아웃이 충돌하는 것을 방지할 수 있습니다.

Git clone 깊이

Git clone 깊이를 선택하면 이력이 지정된 커밋 수로 잘린 부분 복제가 생성됩니다. 전체 복제가 필요할 경우 전체를 선택합니다.

빌드 상태

빌드 시작 및 완료 상태가 소스 공급자에게 보고되도록 하려면 빌드가 시작되고 완료될 때 소스 공급자에게 빌드 상태 보고를 선택합니다.

소스 공급자에게 빌드 상태를 보고하려면 소스 공급자와 연결된 사용자에게 리포지토리에 대한 쓰기 권한이 있어야 합니다. 사용자에게 쓰기 권한이 없는 경우 빌드 상태를 업데이트할 수 없습니다. 자세한 내용은 소스 공급자 액세스 섹션을 참조하세요.

환경

환경 섹션에서 편집을 선택합니다. 변경이 완료되면 구성 업데이트를 선택하여 새 구성을 저장합니다.

다음 속성을 수정할 수 있습니다.

프로비저닝 모델

프로비저닝 모델을 변경하려면 프로비저닝 모델 변경을 선택하고 다음 중 하나를 수행합니다.

- 에서 관리하는 온디맨드 플릿을 사용하려면 온디맨드를 AWS CodeBuild선택합니다. CodeBuild는 온디맨드 플릿을 통해 빌드에 맞는 컴퓨팅 기능을 제공합니다. 빌드가 완료되면 머신이 파괴됩니다. 온디맨드 플릿은 완전 관리형이며 수요 급증을 처리하기 위한 자동 조정 기능을 포함합니다.
- 에서 관리하는 예약 용량 플릿을 사용하려면 예약 용량을 AWS CodeBuild선택한 다음 플릿 이름을 선택합니다. 예약 용량 플릿을 사용하면 빌드 환경을 위한 전용 인스턴스 세트를 구성할 수 있습니다. 이러한 시스템은 유휴 상태로 유지되어 빌드 또는 테스트를 즉시 처리하고 빌드 기간을 단축할 수 있습니다. 예약 용량 플릿을 사용하면 머신이 항상 실행되며 프로비저닝되는 한 비용이 계속 발생합니다.

자세한 내용은 예약 용량 플릿에서 빌드 실행 섹션을 참조하세요.

환경 이미지

빌드 이미지를 변경하려면 이미지 재정의를 선택하고 다음 중 하나를 수행합니다.

- 에서 관리하는 Docker 이미지를 사용하려면 관리형 이미지를 AWS CodeBuild선택한 다음 운영 체제, 런타임, 이미지(Image) 및 이미지 버전에서 선택합니다. 사용 가능한 경우 환경 유형에서 항목을 선택합니다.
- 다른 도커 이미지를 사용하려면 사용자 지정 이미지를 선택합니다. 환경 유형에서 ARM, Linux, Linux GPU 또는 Windows를 선택합니다. Other registry(다른 레지스트리)를 선택한 경우 External registry URL(외부 레지스트리 URL)에 Docker Hub의 도커 이미지 이름 및 태그를 *docker repository*/*docker image name* 형식으로 입력합니다. Amazon ECR을 선택하는 경우 Amazon ECR 리포지토리와 Amazon ECR 이미지를 사용하여 AWS 계정에서 도커 이미지를 선택합니다.
- 프라이빗 도커 이미지를 사용하려면 사용자 지정 이미지를 선택합니다. 환경 유형에서 ARM, Linux, Linux GPU 또는 Windows를 선택합니다. Image registry(이미지 레지스트리)에서 Other registry(다른 레지스트리)를 선택한 다음 프라이빗 도커 이미지에 대한 보안 인증 정보의 ARN을 입력합니다. 보안 인증은 Secrets Manager에서 생성됩니다. 자세한 내용은 AWS Secrets Manager사용 설명서의 AWS Secrets Manager 이란? 섹션을 참조하세요.

> **Note**
>
> CodeBuild는 사용자 지정 도커 이미지에 대해 `ENTRYPOINT`를 재정의합니다.

서비스 역할

다음 중 하나를 수행하세요.

- CodeBuild 서비스 역할이 없는 경우 새 서비스 역할을 선택합니다. 역할 이름에 새 역할의 이름을 입력합니다.
- CodeBuild 서비스 역할이 있는 경우 기존 서비스 역할을 선택합니다. 역할 ARN에서 서비스 역할을 선택합니다.

> **Note**
>
> 콘솔을 사용하여 빌드 프로젝트를 생성하는 경우, 이와 동시에 CodeBuild 서비스 역할을 생성할 수 있습니다. 기본적으로 역할은 해당 빌드 프로젝트에서만 작동합니다. 콘솔을 사용하여 이 서비스 역할을 다른 빌드 프로젝트와 연결하는 경우 다른 빌드 프로젝트에서 작동하도록 역할이 업데이트됩니다. 하나의 서비스 역할은 최대 10개의 빌드 프로젝트에서 작동할 수 있습니다.

추가 구성

제한 시간

빌드가 완료되지 않는 경우 CodeBuild가 해당 빌드를 중지할 제한 시간으로 5분에서 36시간 사이의 값을 지정합니다. [hours] 및 [minutes]가 비어 있는 경우 기본값인 60분이 사용됩니다.

권한 있음

이 빌드 프로젝트를 사용하여 Docker 이미지를 빌드하려는 경우에만 Docker 이미지를 빌드하거나 빌드에서 승격된 권한을 얻으려는 경우 이 플래그 활성화를 선택합니다. 그렇지 않으면 Docker 데몬과 상호 작용을 시도하는 모든 연결된 빌드가 실패합니다. 또한 빌드가 상호 작용할 수 있도록 Docker 데몬을 시작해야 합니다. 이렇게 하는 한 가지 방법은 다음 빌드 명령을 실행하여 buildspec의 `install` 단계에서 Docker 데몬을 초기화하는 것입니다. 선택한 빌드 환경 이미지가 Docker 지원을 통해 CodeBuild에서 제공되지 않는 경우 이 명령을 실행하지 마세요.

> **Note**
>
> 기본적으로 비 VPC 빌드에는 Docker 데몬이 활성화됩니다. VPC 빌드에 Docker 컨테이너를 사용하려면 Docker Docs 웹 사이트의 런타임 권한 및 Linux 기능을 참조하고 권한 부여 모드를 활성화합니다. 또한 Windows는 권한 모드를 지원하지 않습니다.

```
- nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --
host=tcp://127.0.0.1:2375 --storage-driver=overlay2 &
- timeout 15 sh -c "until docker info; do echo .; sleep 1; done"
```

VPC

CodeBuild를 사용하여 VPC에서 작업을 수행하려는 경우:

- VPC에서 CodeBuild가 사용하는 VPC ID를 선택합니다.
- VPC 서브넷에서 CodeBuild가 사용하는 리소스가 포함된 서브넷을 선택합니다.
- VPC 보안 그룹에서 CodeBuild가 VPC의 리소스에 대한 액세스를 허용하기 위해 사용하는 보안 그룹을 선택합니다.

자세한 내용은 Amazon Virtual Private Cloud AWS CodeBuild 와 함께 사용 섹션을 참조하세요.

컴퓨팅

사용 가능한 옵션 중 하나를 선택합니다.

레지스트리 자격 증명

프로젝트가 비공개 레지스트리 이미지로 구성된 경우 레지스트리 자격 증명을 지정합니다.

> **Note**
>
> 이 자격 증명은 이미지가 프라이빗 레지스트리의 이미지로 재정의된 경우에만 사용됩니다.

환경 변수

사용할 빌드의 각 환경 변수에 대해 이름 및 값을 입력하고 유형을 선택합니다.

> **Note**
>
> CodeBuild는 AWS 리전의 환경 변수를 자동으로 설정합니다. buildspec.yml에 추가하지 않은 경우 다음 환경 변수를 설정해야 합니다.
>
> - AWS_ACCOUNT_ID
> - IMAGE_REPO_NAME
> - IMAGE_TAG

콘솔과 AWS CLI 사용자는 환경 변수를 볼 수 있습니다. 환경 변수의 가시성에 대한 문제가 없다면 [Name] 및 [Value] 필드를 설정한 다음 [Type]을 [Plaintext]로 설정합니다.

액세스 키 ID, AWS 보안 AWS 액세스 키 또는 암호와 같은 민감한 값을 가진 환경 변수를 Amazon EC2 Systems Manager 파라미터 스토어 또는에 파라미터로 저장하는 것이 좋습니다 AWS Secrets Manager.

Amazon EC2 Systems Manager Parameter Store를 사용하는 경우 유형에서 파라미터를 선택합니다. 이름에 CodeBuild가 참조할 식별자를 입력합니다. 값에 Amazon EC2 Systems Manager Parameter Store에 저장되는 파라미터의 이름을 입력합니다. 예를 들어 `/CodeBuild/dockerLoginPassword`라는 이름의 파라미터를 사용하여 유형에서 파라미터를 선택합니다. 이름에 `LOGIN_PASSWORD`를 입력합니다. 값에 `/CodeBuild/dockerLoginPassword`를 입력합니다.

> **Important**
>
> Amazon EC2 Systems Manager Parameter Store를 사용하는 경우 `/CodeBuild/`로 시작하는 파라미터 이름(예: `/CodeBuild/dockerLoginPassword`)으로 파라미터를 저장하는 것이 좋습니다. CodeBuild 콘솔을 사용하여 Amazon EC2 Systems Manager에서 파라미터를 생성할 수 있습니다. 파라미터 생성을 선택하고 대화 상자에 표시되는 지시에 따릅니다. (그 대화 상자에서 KMS 키에 대해 계정에 있는 AWS KMS 키의 ARN을 지정할 수 있습니다. Amazon EC2 Systems Manager는이 키를 사용하여 저장 중에 파라미터의 값을 암호화하고 검색 중에 복호화합니다.) CodeBuild 콘솔을 사용하여 파라미터를 생성하는 경우 콘솔은 이름이 `/CodeBuild/`인 파라미터가 저장되면 시작합니다. 자세한 내용은 Amazon EC2 Systems Manager 사용 설명서의 Systems Manager Parameter Store 및 Systems Manager Parameter Store 콘솔 연습을 참조하세요.
> 빌드 프로젝트가 Amazon EC2 Systems Manager Parameter Store에 저장된 파라미터를 참조하는 경우 해당 빌드 프로젝트의 서비스 역할은 `ssm:GetParameters` 작업을 허용해야 합니다. 앞에서 새 서비스 역할을 선택한 경우에는 CodeBuild는 빌드 프로젝트의 기본 서비스 역할에 이 작업을 포함합니다. Existing service role(기존 서비스 역할)을 선택한 경우에는 이 작업을 서비스 역할에 별도로 포함해야 합니다.
> 빌드 프로젝트가 `/CodeBuild/`로 시작되지 않는 파라미터 이름으로 Amazon EC2 Systems Manager Parameter Store에 저장된 파라미터를 참조하는 경우 새 서비스 역할을 선택하면 `/CodeBuild/`로 시작하지 않는 파라미터 이름에 액세스할 수 있도록 해당 서비스 역할을 업데이트해야 합니다. 이는 서비스 역할이 `/CodeBuild/`로 시작하는 파라미터 이름에만 액세스할 수 있기 때문입니다.

새 서비스 역할을 선택하면 Amazon EC2 Systems Manager Parameter Store에 있는 /CodeBuild/ 네임스페이스의 모든 파라미터를 해독할 권한이 서비스 역할에 포함됩니다.
사용자가 설정한 환경 변수는 기존 환경 변수를 대체합니다. 예를 들어 도커 이미지에 값이 my_value인 MY_VAR이라는 환경 변수가 이미 포함되어 있는데, 사용자가 MY_VAR 환경 변수의 값을 other_value로 설정하면, my_value가 other_value로 바뀝니다. 마찬가지로, 도커 이미지에 값이 /usr/local/sbin:/usr/local/bin인 PATH라는 환경 변수가 이미 포함되어 있는데, 사용자가 PATH 환경 변수의 값을 $PATH:/usr/share/ant/bin으로 설정하면, /usr/local/sbin:/usr/local/bin이 $PATH:/usr/share/ant/bin 리터럴 값으로 바뀝니다.
CODEBUILD_로 시작하는 이름으로 환경 변수를 설정하지 마십시오. 이 접두사는 내부 전용으로 예약되어 있습니다.
여러 위치에서 동일한 이름의 환경 변수가 정의되는 경우, 다음과 같이 값이 결정됩니다.

- 시작 빌드 작업 호출의 값이 가장 높은 우선 순위를 갖습니다.
- 빌드 프로젝트 정의의 값이 다음 우선 순위를 갖습니다.
- buildspec 선언의 값이 가장 낮은 우선 순위를 갖습니다.

Secrets Manager를 사용하는 경우 유형으로 Secrets Manager로 선택합니다. 이름에 CodeBuild가 참조할 식별자를 입력합니다. 값에 *secret-id*:*json-key*:*version-stage*:*version-id* 패턴을 사용하여 reference-key를 입력합니다. 자세한 내용은 Secrets Manager reference-key in the buildspec file 섹션을 참조하세요.

Important

Secrets Manager를 사용하는 경우 이름이 /CodeBuild/로 시작하는 암호를 저장하는 것이 좋습니다(예: /CodeBuild/dockerLoginPassword). 자세한 내용은 AWS Secrets Manager사용 설명서의 AWS Secrets Manager 이란? 섹션을 참조하세요.
빌드 프로젝트가 Secrets Manager에 저장된 암호를 참조하는 경우 해당 빌드 프로젝트의 서비스 역할은 secretsmanager:GetSecretValue 작업을 허용해야 합니다. 앞에서 새 서비스 역할을 선택한 경우에는 CodeBuild는 빌드 프로젝트의 기본 서비스 역할에 이 작업을 포함합니다. Existing service role(기존 서비스 역할)을 선택한 경우에는 이 작업을 서비스 역할에 별도로 포함해야 합니다.
빌드 프로젝트가 /CodeBuild/로 시작되지 않는 보안 암호 이름으로 Secrets Manager에 저장된 암호를 참조하는 경우 새 서비스 역할을 선택하면 /CodeBuild/로 시작하지

않는 보안 암호 이름에 액세스할 수 있도록 해당 서비스 역할을 업데이트해야 합니다. 이는 서비스 역할이 /CodeBuild/로 시작하는 암호 이름에만 액세스할 수 있기 때문입니다.
새 서비스 역할을 선택하면 에 있는 /CodeBuild/ 네임스페이스의 모든 암호를 해독할 권한이 서비스 역할에 포함됩니다.

Buildspec

Buildspec 섹션에서 편집을 선택합니다. 변경이 완료되면 구성 업데이트를 선택하여 새 구성을 저장합니다.

다음 속성을 수정할 수 있습니다.

빌드 사양

다음 중 하나를 수행하세요.

- 소스 코드에 buildspec 파일이 있는 경우 Use a buildspec file(빌드 사양 파일 사용) 을 선택합니다. 기본적으로 CodeBuild는 소스 코드 루트 디렉터리에서 `buildspec.yml`이라는 파일을 찾습니다. buildspec 파일이 다른 이름이나 위치를 사용하는 경우 Buildspec 이름에 소스 루트의 경로를 입력합니다(예: `buildspec-two.yml` 또는 `configuration/buildspec.yml`). buildspec 파일이 S3 버킷에 있는 경우 해당 파일은 빌드 프로젝트와 동일한 AWS 리전에 있어야 합니다. ARN을 사용하여 buildspec 파일을 지정합니다(예: `arn:aws:s3:::<my-codebuild-sample2>/buildspec.yml`).
- 소스 코드에 buildspec 파일이 포함되어 있지 않거나, 소스 코드의 루트 디렉터리에 있는 `buildspec.yml` 파일의 `build` 단계에 지정된 것과 다른 빌드 명령 세트를 실행하려는 경우 빌드 명령 삽입을 선택합니다. 빌드 명령의 `build` 단계에서 실행하려는 명령을 입력합니다. 명령이 여러 개인 경우 각 명령을 &&로 구분합니다(예: `mvn test && mvn package`). 다른 구문에서 명령을 실행하려는 경우 또는 `build` 단계에 대해 특히 긴 명령 목록이 있는 경우에는 소스 코드 루트 디렉터리에 `buildspec.yml` 파일을 추가하고, 이 파일에 명령을 추가한 다음, 소스 코드 루트 디렉터리에서 buildspec.yml 사용을 선택합니다.

자세한 내용은 buildspec 참조 섹션을 참조하세요.

배치 구성

배치 구성 섹션에서 편집을 선택합니다. 변경이 완료되면 구성 업데이트를 선택하여 새 구성을 저장합니다. 자세한 내용은 배치로 빌드 실행 섹션을 참조하세요.

다음 속성을 수정할 수 있습니다.

배치 서비스 역할

배치 빌드에 대한 서비스 역할을 제공합니다.

다음 중 하나를 선택합니다.

- 배치 서비스 역할이 없는 경우 새 서비스 역할을 선택합니다. 서비스 역할에 새 역할의 이름을 입력합니다.
- 배치 서비스 역할이 있는 경우 기존 서비스 역할을 선택합니다. 서비스 역할에서 서비스 역할을 선택합니다.

배치 빌드는 배치 구성에 새로운 보안 역할을 도입합니다. CodeBuild가 사용자 대신 `StartBuild`, `StopBuild` 및 `RetryBuild` 작업을 호출하여 배치의 일부로 빌드를 실행할 수 있어야 하므로 이 새 역할이 필요합니다. 고객은 다음과 같은 두 가지 이유로 빌드에 사용하는 것과 동일한 역할이 아닌 새 역할을 사용해야 합니다.

- 빌드 역할 `StartBuild`, `StopBuild` 및 `RetryBuild` 권한을 부여하면 단일 빌드에서 buildspec을 통해 더 많은 빌드를 시작할 수 있습니다.
- CodeBuild 배치 빌드는 배치의 빌드에 사용할 수 있는 빌드 및 컴퓨팅 유형의 수를 제한하는 제한을 제공합니다. 빌드 역할에 이러한 권한이 있는 경우 빌드 자체가 이러한 제한을 우회할 수 있습니다.

배치에 허용되는 컴퓨팅 유형

배치에 허용되는 컴퓨팅 유형을 선택합니다. 해당하는 항목을 모두 선택합니다.

배치에 허용되는 플릿

배치에 허용되는 플릿을 선택합니다. 해당하는 항목을 모두 선택합니다.

배치에 허용되는 최대 빌드 수

배치에 허용되는 최대 빌드 수를 입력합니다. 이 제한을 초과하는 배치는 실패합니다.

배치 제한 시간

배치 빌드가 완료되는 최대 시간을 입력합니다.

아티팩트 결합

배치의 모든 아티팩트를 단일 위치로 결합을 선택하면 배치의 모든 아티팩트가 단일 위치로 결합됩니다.

배치 보고서 모드

배치 빌드에 대해 원하는 빌드 상태 보고서 모드를 선택합니다.

> **Note**
>
> 이 필드는 프로젝트 소스가 Bitbucket, GitHub 또는 GitHub Enterprise이고 소스에서 빌드 시작 및 종료 시 소스 공급자에게 빌드 상태 보고를 선택한 경우에만 사용할 수 있습니다.

빌드 집계

배치의 모든 빌드 상태를 단일 상태 보고서로 통합하려면 선택합니다.

개별 빌드

배치에 있는 모든 빌드의 빌드 상태를 별도로 보고하려면 선택합니다.

아티팩트

아티팩트 섹션에서 편집을 선택합니다. 변경이 완료되면 구성 업데이트를 선택하여 새 구성을 저장합니다.

다음 속성을 수정할 수 있습니다.

유형

다음 중 하나를 수행하세요.

- 빌드 출력 결과물을 생성하지 않으려면 [No artifacts]를 선택합니다. 빌드 테스트만 실행하고 있는 경우 또는 Amazon ECR 리포지토리에 도커 이미지를 푸시하려는 경우에 이 작업을 원할 수 있습니다.
- S3 버킷에 빌드 출력을 저장하려면 Amazon S3를 선택하고 다음 작업을 수행합니다.
  - 빌드 출력 ZIP 파일이나 폴더에 프로젝트 이름을 사용하려는 경우 이름을 비워 둡니다. 그렇지 않으면 이름을 입력합니다. (ZIP 파일을 출력하고 ZIP 파일에 파일 확장명을 넣으려는 경우, ZIP 파일 이름 뒤에 이를 포함하십시오.)
  - buildspec 파일에 지정된 이름으로 콘솔에서 지정한 이름을 재정의하려는 경우 의미 체계 버전 관리 사용을 선택합니다. buildspec 파일의 이름은 빌드 시 계산되며 Shell 명령 언어를 사용합니다. 예를 들어 아티팩트 이름이 항상 고유하도록 날짜와 시간을 아티팩트 이름에 추가할 수 있습니다. 고유한 결과물 이름을 사용하면 결과물을 덮어쓰지 않을 수 있습니다. 자세한 내용은 buildspec 구문 섹션을 참조하세요.

- [Bucket name]에서 출력 버킷의 이름을 선택합니다.
- 이 절차의 앞부분에서 빌드 명령 삽입을 선택한 경우 출력 파일에 빌드 출력 ZIP 파일 또는 폴더에 넣으려는 빌드의 파일 위치를 입력합니다. 위치가 여러 개인 경우 각 위치를 쉼표로 구분합니다(예: `appspec.yml, target/my-app.jar`). 자세한 내용은 buildspec 구문의 `files` 설명을 참조하십시오.
- 빌드 아티팩트를 암호화하지 않으려면 Remove artifacts encryption(결과물 암호화 제거)을 선택합니다.

각각 원하는 보조 아티팩트 세트마다 다음과 같이 실행합니다.

1. Atrifact identifier(아티팩트 식별자)에서 128자 미만으로 영숫자와 밑줄만 포함된 값을 입력합니다.
2. Add artifact(아티팩트 추가)를 선택합니다.
3. 이전 단계에 따라 보조 결과물을 구성합니다.
4. Save artifact(아티팩트 저장)를 선택합니다.

추가 구성

암호화 키

다음 중 하나를 수행하세요.

- 계정의 AWS 관리형 키 Amazon S3를 사용하여 빌드 출력 아티팩트를 암호화하려면 암호화 키를 비워 둡니다. 기본값입니다.
- 고객 관리형 키를 사용하여 빌드 출력 아티팩트를 암호화하려면 암호화 키에 고객 관리형 키의 ARN을 입력합니다. `arn:aws:kms:`*`region-ID`*`:`*`account-ID`*`:key/`*`key-ID`* 형식을 사용합니다.

캐시 유형

Cache type(캐시 유형)에서 다음 중 하나를 선택합니다.

- 캐시를 사용하지 않으려면 [No cache]를 선택합니다.
- Amazon S3 캐시를 사용하려면 Amazon S3를 선택하고 다음을 수행합니다.
  - 버킷에서 캐시가 저장된 S3 버킷의 이름을 선택합니다.
  - (선택 사항) 캐시 경로 접두사에 Amazon S3 경로 접두사를 입력합니다. Cache path prefix(캐시 경로 접두사) 값은 디렉터리 이름과 비슷합니다. 따라서 캐시를 버킷의 동일한 디렉터리에 저장할 수 있습니다.

> **Important**
>
> 경로 접두사 끝에 후행 슬래시(/)를 추가하지 마십시오.

- 로컬 캐시를 사용하려면 로컬을 선택한 다음 하나 이상의 로컬 캐시 모드를 선택해야 합니다.

> **Note**
>
> Docker 계층 캐시 모드는 Linux에서만 사용할 수 있습니다. 이 모드를 선택할 경우 프로젝트를 권한이 있는 모드에서 실행해야 합니다.

캐시를 사용하면 빌드 환경의 재사용 가능한 특정 부분이 캐시에 저장되고 빌드 전반에서 사용되기 때문에 상당한 빌드 시간을 절약할 수 있습니다. buildspec 파일에 캐시를 지정하는 것에 대한 자세한 정보는 buildspec 구문 단원을 참조하십시오. 캐싱에 대한 자세한 정보는 성능을 개선하기 위한 캐시 빌드을 참조하십시오.

로그

로그 섹션에서 편집을 선택합니다. 변경이 완료되면 구성 업데이트를 선택하여 새 구성을 저장합니다.

다음 속성을 수정할 수 있습니다.

생성하려는 로그를 선택합니다. Amazon CloudWatch Logs 또는 Amazon S3 로그 중 하나 또는 둘 다를 생성할 수 있습니다.

CloudWatch

Amazon CloudWatch Logs 로그를 원할 경우:

CloudWatch 로그

CloudWatch 로그를 선택합니다.

그룹 이름

Amazon CloudWatch Logs 로그 그룹의 이름을 입력합니다.

스트림 이름

Amazon CloudWatch Logs 로그 스트림 이름을 입력합니다.

S3

Amazon S3 로그를 원할 경우:

S3 로그

S3 로그를 선택합니다.

버킷

로그에 대한 S3 버킷 이름을 선택합니다.

경로 접두사

로그의 접두사를 입력합니다.

S3 로그 암호화 비활성화

S3 로그를 암호화하지 않으려면 선택합니다.

빌드 프로젝트 설정 변경(AWS CLI)

와 AWS CLI 함께를 사용하는 방법에 대한 자세한 내용은 단원을 AWS CodeBuild참조하십시오명령줄 참조.

를 사용하여 CodeBuild 프로젝트를 업데이트하려면 업데이트된 속성으로 JSON 파일을 AWS CLI생성하고 해당 파일을 `update-project` 명령에 전달합니다. 업데이트 파일에 포함되지 않은 속성은 변경되지 않습니다.

업데이트 JSON 파일에는 `name` 속성과 수정된 속성만 필요합니다. `name` 속성은 수정할 프로젝트를 식별합니다. 수정된 구조의 경우 해당 구조의 필수 파라미터도 포함되어야 합니다. 예를 들어, 프로젝트에 대한 환경을 수정하려면 `environment/type` 및 `environment/computeType` 속성이 필요합니다. 다음은 환경 이미지를 업데이트하는 예입니다.

```
{
  "name": "<project-name>",
  "environment": {
    "type": "LINUX_CONTAINER",
    "computeType": "BUILD_GENERAL1_SMALL",
    "image": "aws/codebuild/amazonlinux-x86_64-standard:4.0"
  }
}
```

프로젝트의 현재 속성 값을 가져와야 하는 경우 batch-get-projects 명령을 사용하여 수정하려는 프로젝트의 현재 속성을 가져오고 결과를 파일에 기록합니다.

```
aws codebuild batch-get-projects --names "<project-name>" > project-info.json
```

project-info.json 파일에는 프로젝트 배열이 포함되어 있으므로 프로젝트를 업데이트하는 데 직접 사용할 수는 없습니다. 하지만 *project-info.json* 파일에서 수정하려는 속성을 복사한 다음, 수정하려는 속성의 기준으로 업데이트 파일에 붙여넣을 수 있습니다. 자세한 내용은 빌드 프로젝트 세부 정보 보기(AWS CLI) 섹션을 참조하세요.

빌드 프로젝트 생성(AWS CLI)에 설명된 대로 업데이트 JSON 파일을 수정하고 결과를 저장합니다. 업데이트 JSON 파일의 수정이 끝나면 update-project 명령을 실행하여 업데이트 JSON 파일을 전달합니다.

```
aws codebuild update-project --cli-input-json file://<update-project-file>
```

성공하면 업데이트된 프로젝트 JSON이 출력에 나타납니다. 필수 파라미터가 누락된 경우 누락된 파라미터를 식별하는 오류 메시지가 출력에 표시됩니다. 예를 들어, environment/type 파라미터가 누락된 경우 표시되는 오류 메시지는 다음과 같습니다.

```
aws codebuild update-project --cli-input-json file://update-project.json

Parameter validation failed:
Missing required parameter in environment: "type"
```

빌드 프로젝트의 설정 변경(AWS SDKs)

를 SDK와 AWS CodeBuild 함께 사용하는 방법에 대한 자세한 내용은 섹션을 참조하세요AWS SDKs 및 도구 참조. AWS SDKs

CodeBuild의 다중 액세스 토큰

CodeBuild는 AWS Secrets Manager 또는 AWS CodeConnections 연결을 통해 보안 암호에서 타사 공급자에게 액세스 토큰 소싱을 지원합니다. 보안 암호 또는 연결을 GitHub, GitHub Enterprise 또는 Bitbucket과 같은 지정된 타사 공급자와의 상호 작용을 위한 기본 자격 증명으로 설정할 수 있습니다.

소스 자격 증명을 세 가지 다른 수준에서 설정할 수 있습니다.

1. 모든 프로젝트의 계정 수준 자격 증명: 이는 AWS 계정의 모든 프로젝트에 대한 기본 자격 증명입니다. 프로젝트 또는 소스 수준 자격 증명이 지정되지 않은 경우 프로젝트에 사용됩니다.
2. 특정 리포지토리의 소스 수준 자격 증명: Secrets Manager 보안 암호 또는 AWS CodeConnections 연결이 프로젝트 소스에 정의된 경우입니다. 이러한 자격 증명은 지정된 소스 리포지토리의 작업에만 사용됩니다. 이렇게 하면 동일한 프로젝트에서 서로 다른 권한 범위를 가진 다중 액세스 토큰을 설정할 수 있으며 기본 계정 수준 자격 증명을 사용하지 않을 수 있습니다.
3. 프로젝트 수준 대체 자격 증명: `NO_SOURCE`를 기본 소스 유형으로 사용하여 프로젝트 수준 대체 자격 증명을 설정하고 보안 암호 또는 연결을 정의할 수 있습니다. 프로젝트에 여러 소스가 있지만 동일한 자격 증명을 사용하려는 경우 또는 프로젝트에 기본 계정 수준 자격 증명을 사용하지 않으려는 경우이 옵션을 사용합니다.

주제

- 1단계: Secrets Manager 보안 암호 또는 AWS CodeConnections 연결 생성
- 2단계: Secrets Manager 보안 암호에 대한 IAM 역할 액세스 권한을 CodeBuild 프로젝트에 부여
- 3단계: Secrets Manager 또는 AWS CodeConnections 토큰 구성
- 추가 설정 옵션

1단계: Secrets Manager 보안 암호 또는 AWS CodeConnections 연결 생성

다음 지침에 따라 Secrets Manager 보안 암호 또는 AWS CodeConnections 연결을 생성합니다.

- Secrets Manager 보안 암호에 토큰 생성 및 저장.
- GitHub에 대한 연결 생성
- GitHub Enterprise Server에 대한 연결 생성
- Bitbucket에 대한 연결 생성

2단계: Secrets Manager 보안 암호에 대한 IAM 역할 액세스 권한을 CodeBuild 프로젝트에 부여

> **Note**
>
> 계속하기 전에 Secrets Manager 또는에서 생성된 토큰에 액세스할 수 있어야 합니다 AWS CodeConnections.

CodeBuild 프로젝트 IAM 역할에 Secrets Manager 또는에 대한 액세스 권한을 부여하려면 다음 IAM 정책을 추가 AWS CodeConnections해야 합니다.

CodeBuild 프로젝트에 IAM 역할 액세스 권한을 부여하려면

1. CodeBuild 프로젝트의 CodeBuild가 다른 AWS 서비스와 상호 작용하도록 허용에 대한 지침에 따라 CodeBuild 프로젝트의 IAM 역할을 생성합니다.
2. 다음 중 하나를 수행하세요.
   - CodeBuild 프로젝트 역할에 보안 암호에 대한 액세스 권한을 부여하는 다음 IAM 정책을 추가합니다.

     JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": [
                "arn:aws:iam::*:role/Service*"
            ]
        }
    ]
}
```

     (선택 사항) AWS KMS 고객 관리형 키를 사용하여 Secrets Manager 보안 암호를 암호화하는 경우 다음 정책 설명을 추가하여 액세스 권한을 부여할 수 있습니다.

     JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:iam::*:role/Service*",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:SecretARN":
 "arn:aws:iam::*:role/Service*"
                }
            }
        }
    ]
}
```

- CodeBuild 프로젝트 역할에 연결에 대한 액세스 권한을 부여하는 다음 IAM 정책을 추가합니다.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codeconnections:GetConnectionToken",
                "codeconnections:GetConnection"
            ],
            "Resource": [
                "arn:aws:iam::*:role/Service*"
            ]
        }
    ]
}
```

3단계: Secrets Manager 또는 AWS CodeConnections 토큰 구성

Secrets Manager 또는 AWS CodeConnections 토큰을 사용하여 소스 자격 증명을 세 가지 수준으로 설정할 수 있습니다.

Secrets Manager 또는 AWS CodeConnections 토큰을 계정 수준 자격 증명으로 구성

Secrets Manager 보안 암호 또는 AWS CodeConnections 연결을 계정 수준 자격 증명으로 구성하고 프로젝트에서 사용할 수 있습니다.

AWS Management Console

에서 연결을 계정 수준 자격 증명으로 구성하려면 AWS Management Console

1. 소스 공급자에서 Bitbucket, GitHub 또는 GitHub Enterprise를 선택합니다.
2. 자격 증명에서 다음 중 하나를 수행합니다.
   - 기본 소스 자격 증명을 선택하여 계정의 기본 소스 자격 증명을 사용하여 모든 프로젝트에 적용합니다.
     a. 소스 공급자에 연결되지 않은 경우 기본 소스 자격 증명 관리를 선택합니다.
     b. 자격 증명 유형에서 자격 증명 유형을 선택합니다.
     c. 를 선택한 경우 기존 연결을 사용하거나 새 연결을 생성하도록 AWS CodeConnections선택합니다.

        다른 자격 증명 유형을 선택한 경우 서비스에서 토큰을 저장하는 데 사용할 서비스를 선택하고 다음을 수행합니다.

        - Secrets Manager를 사용하도록 선택한 경우 기존 보안 암호 연결을 사용하거나 새 보안 암호를 생성하고 저장을 선택할 수 있습니다. 새 암호를 생성하는 방법에 대한 자세한 정보는 Secrets Manager 보안 암호에 토큰 생성 및 저장 섹션을 참조하세요.
        - CodeBuild 사용하도록 선택한 경우 토큰 또는 사용자 이름과 앱 암호를 입력하고 저장을 선택합니다. Bitbucket API 토큰의 경우 Atlassian 계정 이메일 주소와 API 토큰을 입력합니다.
   - 사용자 지정 소스 자격 증명을 선택하여 사용자 지정 소스 자격 증명을 사용하여 계정의 기본 설정을 재정의합니다.
     a. 자격 증명 유형에서 자격 증명 유형을 선택합니다.
     b. 연결에서 기존 연결을 선택하거나 연결을 새로 생성합니다.

AWS CLI

에서 연결을 계정 수준 자격 증명으로 구성하려면 AWS CLI

- 터미널(Linux, macOS, Unix) 또는 명령 프롬프트(Windows)를 엽니다. AWS CLI 를 사용하여 import-source-credentials 명령을 실행합니다.

  다음 명령을 사용하여 Secrets Manager 보안 암호를 구성합니다.

  ```
  aws codebuild import-source-credentials \
      --token "<secret-arn>" \
      --server-type <source-provider> \
      --auth-type SECRETS_MANAGER \
      --region <aws-region>
  ```

  다음 명령을 사용하여 AWS CodeConnections 연결을 구성합니다.

  ```
  aws codebuild import-source-credentials \
      --token "<connection-arn>" \
      --server-type <source-provider> \
      --auth-type CODECONNECTIONS \
      --region <aws-region>
  ```

  이 명령을 사용하면 토큰을 계정 수준 기본 소스 자격 증명으로 가져올 수 있습니다. ImportSourceCredentials API를 사용하여 자격 증명을 가져올 때 CodeBuild는 프로젝트에 더 구체적인 자격 증명 세트가 구성되지 않은 한 웹후크, 빌드 상태 보고 및 git 복제 작업과 같은 소스 공급자와의 모든 상호 작용에 토큰을 사용합니다.

이제 빌드 프로젝트에서 토큰을 사용하고 실행할 수 있습니다. 자세한 내용은 에서 빌드 프로젝트 생성 AWS CodeBuild 및 수동으로 AWS CodeBuild 빌드 실행 섹션을 참조하세요.

여러 토큰을 소스 수준 자격 증명으로 구성

Secrets Manager 보안 암호 또는 AWS CodeConnections 연결을 소스 수준 자격 증명으로 사용하려면 CodeBuild 프로젝트의 토큰을 직접 참조하고 빌드를 시작합니다.

AWS Management Console

에서 여러 토큰을 소스 수준 자격 증명으로 구성하려면 AWS Management Console

1. 소스 공급자에서 GitHub를 선택합니다.
2. 자격 증명에서 다음 중 하나를 수행합니다.
   - 기본 소스 자격 증명을 선택하여 계정의 기본 소스 자격 증명을 사용하여 모든 프로젝트에 적용합니다.
     a. GitHub에 연결되지 않은 경우 기본 소스 자격 증명 관리를 선택합니다.
     b. 자격 증명 유형에서 GitHub 앱을 선택합니다.
     c. 연결에서 기존 연결을 선택하거나 연결을 새로 생성합니다.
   - 사용자 지정 소스 자격 증명을 선택하여 사용자 지정 소스 자격 증명을 사용하여 계정의 기본 설정을 재정의합니다.
     a. 자격 증명 유형에서 GitHub 앱을 선택합니다.
     b. 연결에서 기존 연결을 선택하거나 연결을 새로 생성합니다.
3. 소스 추가를 선택하고 소스 공급자 및 자격 증명을 선택하는 프로세스를 반복합니다.

AWS CLI

에서 여러 토큰을 소스 수준 자격 증명으로 구성하려면 AWS CLI

- 터미널(Linux, macOS, Unix) 또는 명령 프롬프트(Windows)를 엽니다. AWS CLI 를 사용하여 create-project 명령을 실행합니다.

  다음 명령을 사용합니다.

```
aws codebuild create-project --region <aws-region> \
    --name <project-name> \
    --artifacts type=NO_ARTIFACTS \
    --environment "type=LINUX_CONTAINER,
                   computeType=BUILD_GENERAL1_SMALL,
                   image=aws/codebuild/amazonlinux-x86_64-standard:5.0" \
    --service-role <service-role-name> \
    --source "type=GITHUB,
              location=<github-repository-1>,
              auth={type=SECRETS_MANAGER,resource=<secret-or-connection-arn-1>}"
 \
```

```
    --secondary-sources "type=GITHUB,
              location=<github-repository-2>,
              auth={type=SECRETS_MANAGER,resource=<secret-or-connection-arn-2>},
              sourceIdentifier=secondary"

aws codebuild start-build --region <aws-region> --project-name <project-name>
```

프로젝트 수준 소스 자격 증명 대체 설정

프로젝트 수준 소스 자격 증명 대체를 설정하려면 프로젝트의 기본 소스에 NO_SOURCE를 사용하고 토큰을 참조합니다.

```
aws codebuild create-project \
    --name <project-name> \
    --service-role <service-role-name> \
    --artifacts type=NO_ARTIFACTS \
    --environment "type=LINUX_CONTAINER,
                   computeType=BUILD_GENERAL1_SMALL,
                   image=aws/codebuild/amazonlinux-x86_64-standard:5.0" \
    --service-role <service-role-name> \
    --source "type=NO_SOURCE,
              auth={type=SECRETS_MANAGER,resource=<secret-or-connection-arn>},
              buildspec=<buildspec>"
    --secondary-sources "type=GITHUB,
                         location=<github-repository>,
                         sourceIdentifier=secondary"

aws codebuild start-build --region <aws-region> --project-name <project_name>
```

NO_SOURCE를 사용하는 경우 일반적으로 외부 소스를 사용하여 buildspec을 가져오도록 직접 구성되지 않았으므로 소스 모델 내에 buildspec이 제공됩니다. 일반적으로 NO_SOURCE 소스는 buildspec 내에서 모든 관련 리포지토리 복제를 처리합니다. 이러한 작업에 대해 구성된 자격 증명을 사용할 수 있도록 하려면 buildspec에서 `git-credential-helper` 옵션을 활성화할 수 있습니다.

```
env:
  git-credential-helper: yes
```

빌드 중에 CodeBuild는 구성된 토큰에서 `AuthServer` 필드를 읽고 해당 특정 타사 소스 공급자에 대한 모든 git 요청에 토큰 자격 증명을 사용합니다.

추가 설정 옵션

CloudFormation 템플릿을 사용하여 Secrets Manager 계정 수준 자격 증명을 구성할 수 있습니다. 다음 CloudFormation 템플릿을 사용하여 계정 수준 자격 증명을 설정할 수 있습니다.

```
Parameters:
  GitHubToken:
    Type: String
    NoEcho: true
    Default: placeholder
Resources:
  CodeBuildAuthTokenSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: CodeBuild auth token
      Name: codebuild-auth-token
      SecretString:
        !Join
          - ''
          - - '{"ServerType":"GITHUB","AuthType":"PERSONAL_ACCESS_TOKEN","Token":"'
            - !Ref GitHubToken
            - '"}'
      Tags:
        - Key: codebuild:source:provider
          Value: github
        - Key: codebuild:source:type
          Value: personal_access_token
  CodeBuildSecretsManagerAccountCredential:
    Type: AWS::CodeBuild::SourceCredential
    Properties:
      ServerType: GITHUB
      AuthType: SECRETS_MANAGER
      Token: !Ref CodeBuildAuthTokenSecret
```

> **Note**
>
> 동일한 스택에서 프로젝트를 생성하는 경우 DependsOn CloudFormation 속성을 사용하여 프로젝트 전에 `AccountCredential`이 생성되도록 합니다.

CloudFormation 템플릿을 사용하여 Secrets Manager 다중 소스 수준 자격 증명을 구성할 수도 있습니다. 다음 CloudFormation 템플릿을 사용하여 여러 토큰을 사용하여 여러 소스를 가져올 수 있습니다.

```
Parameters:
  GitHubTokenOne:
    Type: String
    NoEcho: true
    Default: placeholder
  GitHubTokenTwo:
    Type: String
    NoEcho: true
    Default: placeholder

Resources:
  CodeBuildSecretsManagerProject:
    Type: AWS::CodeBuild::Project
    Properties:
      Name: codebuild-multitoken-example
      ServiceRole: <service-role>
      Environment:
        Type: LINUX_CONTAINER
        ComputeType: BUILD_GENERAL1_SMALL
        Image: aws/codebuild/amazonlinux-x86_64-standard:5.0
      Source:
        Type: GITHUB
        Location: <github-repository-one>
        Auth:
          Type: SECRETS_MANAGER
          Resource: !Ref CodeBuildAuthTokenSecretOne
      SecondarySources:
        - Type: GITHUB
          Location: <github-repository-two>
          Auth:
            Type: SECRETS_MANAGER
            Resource: !Ref CodeBuildAuthTokenSecretTwo
          SourceIdentifier: secondary
      Artifacts:
        Type: NO_ARTIFACTS
      LogsConfig:
        CloudWatchLogs:
          Status: ENABLED
  CodeBuildProjectIAMRoleSecretAccess:
    Type: AWS::IAM::RolePolicy
```

```
    Properties:
      RoleName: <role-name>
      PolicyName: CodeBuildProjectIAMRoleSecretAccessPolicy
      PolicyDocument:
        Version: '2012-10-17       '
        Statement:
          - Effect: Allow
            Action:
              - secretsmanager:GetSecretValue
            Resource:
              - !Ref CodeBuildAuthTokenSecretOne
              - !Ref CodeBuildAuthTokenSecretTwo
  CodeBuildAuthTokenSecretOne:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: CodeBuild auth token one
      Name: codebuild-auth-token-one
      SecretString:
        !Join
          - ''
          - - '{"ServerType":"GITHUB","AuthType":"PERSONAL_ACCESS_TOKEN","Token":"'
            - !Ref GitHubTokenOne
            - '"}'
      Tags:
        - Key: codebuild:source:provider
          Value: github
        - Key: codebuild:source:type
          Value: personal_access_token
  CodeBuildAuthTokenSecretTwo:
    Type: AWS::SecretsManager::Secret
    Properties:
      Description: CodeBuild auth token two
      Name: codebuild-auth-token-two
      SecretString:
        !Join
          - ''
          - - '{"ServerType":"GITHUB","AuthType":"PERSONAL_ACCESS_TOKEN","Token":"'
            - !Ref GitHubTokenTwo
            - '"}'
      Tags:
        - Key: codebuild:source:provider
          Value: github
        - Key: codebuild:source:type
```

```
          Value: personal_access_token
```

에서 빌드 프로젝트 삭제 AWS CodeBuild

CodeBuild 콘솔 AWS CLI또는 AWS SDKs 사용하여 CodeBuild에서 빌드 프로젝트를 삭제할 수 있습니다. 프로젝트를 삭제하면 해당 빌드가 삭제되지 않습니다.

> **Warning**
>
> 빌드 및 리소스 정책이 있는 프로젝트는 삭제할 수 없습니다. 리소스 정책 및 빌드가 있는 프로젝트를 삭제하려면 먼저 리소스 정책을 제거하고 빌드를 삭제합니다.

주제

- 빌드 프로젝트 삭제(콘솔)
- 빌드 프로젝트 삭제(AWS CLI)
- 빌드 프로젝트(AWS SDKs) 삭제

빌드 프로젝트 삭제(콘솔)

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 프로젝트 빌드를 선택합니다.
3. 다음 중 하나를 수행하세요.
   - 삭제하려는 빌드 프로젝트 옆에 있는 라디오 버튼을 선택한 후 삭제를 선택합니다.
   - 삭제하려는 빌드 프로젝트의 링크를 선택한 다음, [Delete]를 선택합니다.

> **Note**
>
> 기본적으로 가장 최근에 실행한 10개의 빌드 프로젝트가 표시됩니다. 더 많은 빌드 프로젝트를 보려면 Projects per page(페이지당 프로젝트)에서 다른 값을 선택하거나 뒤로 및 앞으로 화살표를 선택하여 프로젝트를 봅니다.

빌드 프로젝트 삭제(AWS CLI)

1. `delete-project` 명령을 실행합니다.

   ```
   aws codebuild delete-project --name name
   ```

   다음과 같이 자리 표시자를 바꿉니다.

   - *name*: 필수 문자열입니다. 삭제할 빌드 프로젝트의 이름입니다. 사용 가능한 빌드 프로젝트 목록을 보려면 `list-projects` 명령을 실행합니다. 자세한 내용은 빌드 프로젝트 이름 목록 보기(AWS CLI) 단원을 참조하십시오.
2. 이 명령이 제대로 실행되면 출력에 데이터나 오류가 표시되지 않습니다.

와 AWS CLI 함께를 사용하는 방법에 대한 자세한 내용은 단원을 AWS CodeBuild참조하십시오명령줄 참조.

빌드 프로젝트(AWS SDKs) 삭제

를 SDK와 AWS CodeBuild 함께 사용하는 방법에 대한 자세한 내용은 섹션을 참조하세요AWS SDKs 및 도구 참조. AWS SDKs

퍼블릭 빌드 프로젝트 URL 가져오기

AWS CodeBuild 를 사용하면 빌드 프로젝트의 빌드 결과, 로그 및 아티팩트를 일반 대중이 사용할 수 있도록 할 수 있습니다. 이렇게 하면 소스 리포지토리의 기여자가 AWS 계정에 액세스할 필요 없이 빌드의 결과를 보고 아티팩트를 다운로드할 수 있습니다.

프로젝트의 빌드를 공개하면 프로젝트가 프라이빗일 때 실행된 빌드를 포함하여 프로젝트의 모든 빌드 결과, 로그, 아티팩트는 누구나 사용할 수 있습니다. 마찬가지로 퍼블릭 빌드 프로젝트를 프라이빗으로 설정하면 해당 프로젝트의 빌드 결과를 더 이상 공개할 수 없습니다.

프로젝트 빌드 결과의 퍼블릭 가시성을 변경하는 방법에 대한 자세한 내용은 퍼블릭 빌드 액세스 활성화 섹션을 참조하세요.

CodeBuild는 프로젝트에 고유한 퍼블릭 빌드의 URL을 제공합니다.

빌드 프로젝트의 퍼블릭 URL을 가져오려면 다음 절차를 수행하세요.

퍼블릭 빌드 프로젝트의 URL을 가져오려면

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 프로젝트 빌드를 선택합니다.
3. 퍼블릭 URL을 가져오려는 빌드 프로젝트의 링크를 선택합니다.
4. 퍼블릭 URL은 구성 섹션의 퍼블릭 프로젝트 URL 필드에 표시됩니다. 링크를 선택하여 URL을 열거나 복사 버튼을 사용하여 URL을 복사할 수 있습니다.

Warning

프로젝트의 빌드 결과를 공개할 때는 다음 사항을 염두에 두어야 합니다.

- 프로젝트가 프라이빗일 때 실행된 빌드를 포함하여 프로젝트의 모든 빌드 결과, 로그, 아티팩트는 누구나 이용할 수 있습니다.
- 모든 빌드 로그와 아티팩트는 누구나 이용할 수 있습니다. 환경 변수, 소스 코드 및 기타 중요한 정보가 빌드 로그와 아티팩트에 출력되었을 수 있습니다. 빌드 로그에 출력되는 정보에 주의해야 합니다. 몇 가지 모범 사례는 다음과 같습니다.
  - 환경 변수에 민감한 값, 특히 AWS 액세스 키 IDs 및 보안 액세스 키를 저장하지 마십시오. Amazon EC2 Systems Manager 파라미터 스토어 또는를 사용하여 민감한 값을 저장하는 AWS Secrets Manager 것이 좋습니다.
  - webhook를 최대한 안전하게 보호하려면 webhook 사용 모범 사례에 따라 빌드를 트리거할 수 있는 엔터티를 제한하고, buildspec을 프로젝트 자체에 저장하지 마세요.
- 악의적인 사용자는 퍼블릭 빌드를 사용하여 악성 아티팩트를 배포할 수 있습니다. 프로젝트 관리자는 모든 풀 요청을 검토하여 풀 요청이 합법적인 변경인지 확인하는 것이 좋습니다. 또한 체크섬으로 모든 아티팩트의 유효성을 검사하여 올바른 아티팩트가 다운로드되고 있는지 확인하는 것이 좋습니다.

빌드 프로젝트 공유

프로젝트 공유를 통해 프로젝트 소유자는 AWS CodeBuild 프로젝트를 다른 AWS 계정 또는 사용자와 공유할 수 있습니다. 이 모델에서는 프로젝트를 소유한 계정(소유자)이 다른 계정(소비자)과 프로젝트를 공유합니다. 소비자는 프로젝트를 편집하거나 실행할 수 없습니다.

주제

- 프로젝트 공유
- 관련 서비스
- 사용자와 공유된 CodeBuild 프로젝트 액세스
- 공유 프로젝트의 공유 해제
- 공유 프로젝트 식별
- 공유 프로젝트 권한

프로젝트 공유

소비자는 AWS CLI 및 AWS CodeBuild 콘솔을 모두 사용하여 공유한 프로젝트와 빌드를 볼 수 있습니다. 소비자는 프로젝트를 편집하거나 실행할 수 없습니다.

기존 리소스 공유에 프로젝트를 추가하거나 AWS RAM 콘솔에서 프로젝트를 만들 수 있습니다.

> **Note**
>
> 리소스 공유에 추가된 빌드가 있는 프로젝트는 삭제할 수 없습니다.

프로젝트를 조직 단위 또는 전체 조직과 공유하려면 AWS Organizations와의 공유를 활성화해야 합니다. 자세한 내용은 AWS RAM 사용 설명서에서 AWS Organizations를 사용하여 공유 사용을 참조하세요.

AWS CodeBuild 콘솔, AWS RAM 콘솔 또는를 사용하여 소유한 프로젝트를 AWS CLI 공유할 수 있습니다.

프로젝트 공유를 위한 전제 조건

프로젝트 공유를 시작하기 전에 AWS 계정이 프로젝트를 소유하고 있는지 확인합니다. 사용자와 공유된 프로젝트를 공유할 수 없습니다.

소유한 프로젝트를 공유하려면(CodeBuild 콘솔)

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 프로젝트 빌드를 선택합니다.

> **Note**
>
> 기본적으로 가장 최근의 빌드 프로젝트 10개만 표시됩니다. 더 많은 빌드 프로젝트를 보려면 기어 아이콘을 선택하고 페이지당 프로젝트 수에서 다른 값을 선택하거나 뒤로 및 앞으로 화살표를 사용합니다.

3. 공유할 프로젝트를 선택한 다음 공유를 선택합니다. 자세한 내용은AWS RAM 사용 설명서의 리소스 공유 생성을 참조하세요.

소유한 프로젝트를 공유하려면(AWS RAM 콘솔)

AWS RAM 사용 설명서에서 리소스 공유 생성을 참조하세요.

소유한 프로젝트를 공유하려면(AWS RAM 명령)

create-resource-share 명령을 사용합니다.

소유한 프로젝트를 공유하려면(CodeBuild 명령)

put-resource-policy 명령을 사용합니다.

1. 이름이 `policy.json`인 파일을 만들고 다음으로 복사합니다.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement":[{
    "Effect":"Allow",
    "Action":[
      "codebuild:BatchGetProjects",
      "codebuild:BatchGetBuilds",
      "codebuild:ListBuildsForProject"],
    "Resource":"arn:aws:iam::*:role/Service*"
  }]
}
```

2. 프로젝트 ARN 및 식별자로 `policy.json`을 업데이트하여 공유합니다. 다음 예제에서는 123456789012로 식별되는 AWS 계정의 루트 사용자에게 읽기 전용 액세스 권한을 부여합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[{
    "Effect":"Allow",
    "Principal":{
      "AWS": [
        "123456789012"
      ]
    },
    "Action":[
      "codebuild:BatchGetProjects",
      "codebuild:BatchGetBuilds",
      "codebuild:ListBuildsForProject"],
    "Resource":"arn:aws:codebuild:us-west-2:123456789012:project/my-project"
  }]
}
```

3. put-resource-policy 명령을 실행합니다.

```
aws codebuild put-resource-policy --resource-arn <project-arn> --policy file://
policy.json
```

4. AWS RAM 리소스 공유 ARN을 가져옵니다.

```
aws ram list-resources --resource-owner SELF --resource-arns <project-arn>
```

이렇게 하면 다음과 비슷한 응답이 반환됩니다.

```
{
  "resources": [
    {
      "arn": "<project-arn>",
      "type": "<type>",
      "resourceShareArn": "<resource-share-arn>",
      "creationTime": "<creation-time>",
      "lastUpdatedTime": "<last-update-time>"
    }
  ]
```

```
}
```

응답에서 다음 단계에서 사용할 *<resource-share-arn>* 값을 복사합니다.

5. AWS RAM promote-resource-share-created-from-policy 명령을 실행합니다.

```
aws ram promote-resource-share-created-from-policy --resource-share-arn <resource-share-arn>
```

관련 서비스

프로젝트 공유는 모든 AWS 계정 또는를 통해 AWS 리소스를 공유할 수 있는 서비스AWS RAM인 AWS Resource Access Manager ()와 통합됩니다 AWS Organizations. AWS RAM에서는 리소스와 리소스를 공유할 소비자를 지정하는 리소스 공유를 생성하여 리소스를 공유합니다. 소비자는 개별 AWS 계정,의 조직 단위 AWS Organizations또는의 전체 조직일 수 있습니다 AWS Organizations.

자세한 내용은 AWS RAM 사용 설명서를 참조하십시오.

사용자와 공유된 CodeBuild 프로젝트 액세스

공유 프로젝트에 액세스하려면 소비자의 IAM 역할에 `BatchGetProjects` 권한이 필요합니다. 다음 정책을 해당 IAM 역할에 연결할 수 있습니다.

```
{
    "Effect": "Allow",
    "Resource": [
        "*"
    ],
    "Action": [
        "codebuild:BatchGetProjects"
    ]
}
```

자세한 내용은 에 자격 증명 기반 정책 사용 AWS CodeBuild 단원을 참조하십시오.

공유 프로젝트의 공유 해제

빌드를 포함하여 공유가 해제된 프로젝트는 소유자만 액세스할 수 있습니다. 프로젝트를 공유 해제하면 이전에 공유한 AWS 계정 또는 사용자가 프로젝트 또는 해당 빌드에 액세스할 수 없습니다.

소유하고 있는 공유 프로젝트의 공유를 해제하려면 리소스 공유에서 제거해야 합니다. AWS CodeBuild 콘솔, AWS RAM 콘솔 또는를 사용하여이 작업을 수행할 수 AWS CLI 있습니다.

소유한 공유 프로젝트를 공유 해제하려면(AWS RAM 콘솔)

AWS RAM 사용 설명서에서 리소스 공유 업데이트를 참조하세요.

소유한 공유 프로젝트의 공유를 해제하려면(AWS CLI)

disassociate-resource-share 명령을 사용합니다.

소유한 프로젝트를 공유 해제하려면(CodeBuild 명령)

delete-resource-policy 명령을 실행하고 공유를 해제할 프로젝트의 ARN을 지정합니다.

```
aws codebuild delete-resource-policy --resource-arn project-arn
```

공유 프로젝트 식별

소유자와 소비자는 AWS CLI 를 사용하여 공유 프로젝트를 식별할 수 있습니다.

AWS 계정 또는 사용자와 공유된 프로젝트를 식별하려면(AWS CLI)

list-shared-projects 명령을 사용하여 공유된 프로젝트를 반환합니다.

공유 프로젝트 권한

소유자에 대한 권한

프로젝트 소유자는 프로젝트를 편집하고 빌드를 실행하는 데 사용할 수 있습니다.

소비자에 대한 권한

프로젝트 소비자는 프로젝트와 해당 빌드를 볼 수 있지만 프로젝트를 편집하거나 빌드를 실행하는 데 사용할 수는 없습니다.

빌드 프로젝트 태그 지정

태그는 사용자 또는가 AWS 리소스에 AWS 할당하는 사용자 지정 속성 레이블입니다. 각 AWS 태그에는 두 부분이 있습니다.

- 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. 프로젝트에서 포함할 수 있는 태그 수와 태그 키 및 값 제한에 대한 자세한 내용은 Tags 단원을 참조하십시오..

태그는 AWS 리소스를 식별하고 구성하는 데 도움이 됩니다. 많은 AWS 서비스가 태그 지정을 지원하므로 다른 서비스의 리소스에 동일한 태그를 할당하여 리소스가 관련이 있음을 나타낼 수 있습니다. 예를 들어 S3 버킷에 할당한 것과 동일한 태그를 CodeBuild 프로젝트에 할당할 수 있습니다. 태그 사용에 대한 자세한 내용은 태그 지정 모범 사례를 참조하세요.

CodeBuild에서 기본 리소스는 프로젝트와 보고서 그룹입니다. CodeBuild 콘솔, AWS CLI, CodeBuild APIs 또는 AWS SDKs 사용하여 프로젝트의 태그를 추가, 관리 및 제거할 수 있습니다. 태그로 프로젝트를 식별, 구성 및 추적하는 것 외에도 IAM 정책의 태그를 사용하여 프로젝트를 보고, 상호 작용할 수 있는 사용자를 제어할 수 있습니다. 태그 기반 액세스 정책의 예는 태그를 사용하여 AWS CodeBuild 리소스에 대한 액세스 제어 단원을 참조하세요.

> **Important**
>
> 예약 용량 기능을 사용할 때 소스 파일, Docker 계층, 빌드스펙에 지정된 캐시된 디렉터리를 포함하여 플릿 인스턴스에 캐시된 데이터를 동일한 계정 내의 다른 프로젝트에서 액세스할 수 있습니다. 이는 설계에 의한 것이며 동일한 계정 내의 프로젝트가 플릿 인스턴스를 공유할 수 있도록 허용합니다.

주제

- 프로젝트에 태그 추가
- 프로젝트의 태그 보기
- 프로젝트의 태그 편집
- 프로젝트에서 태그 제거

프로젝트에 태그 추가

프로젝트에 태그를 추가하면 AWS 리소스를 식별 및 구성하고 리소스에 대한 액세스를 관리하는 데 도움이 될 수 있습니다. 먼저 프로젝트에 하나 이상의 태그(키-값 페어)를 추가합니다. 프로젝트에 태그 수에 대한 제한이 있음을 알아두십시오. 키 및 값 필드에서 사용할 수 있는 문자에 대한 제한이 있습니다. 자세한 내용은 Tags 단원을 참조하십시오. 태그가 생성된 후 해당 태그를 기준으로 프로젝트에 대

한 액세스를 관리하는 IAM 정책을 생성할 수 있습니다. CodeBuild 콘솔 또는를 사용하여 프로젝트에 태그를 AWS CLI 추가할 수 있습니다.

> **Important**
>
> 예약 용량 기능을 사용할 때 소스 파일, Docker 계층, 빌드스펙에 지정된 캐시된 디렉터리를 포함하여 플릿 인스턴스에 캐시된 데이터를 동일한 계정 내의 다른 프로젝트에서 액세스할 수 있습니다. 이는 설계에 의한 것이며 동일한 계정 내의 프로젝트가 플릿 인스턴스를 공유할 수 있도록 허용합니다.

프로젝트를 생성할 때 프로젝트에 태그를 추가하는 방법에 대한 자세한 내용은 프로젝트에 태그 추가(콘솔) 단원을 참조하십시오.

> **Important**
>
> 프로젝트에 태그를 추가하기 전에 프로젝트와 같은 리소스에 대한 액세스를 제어하는 태그를 사용할 수도 있는 모든 IAM 정책을 검토하세요. 태그 기반 액세스 정책의 예는 태그를 사용하여 AWS CodeBuild 리소스에 대한 액세스 제어 단원을 참조하세요.

주제

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

프로젝트에 태그 추가(콘솔)

CodeBuild 콘솔을 사용하여 CodeBuild 프로젝트에 하나 이상의 태그를 추가할 수 있습니다.

1. https://console.aws.amazon.com/codebuild/에서 CodeBuild 콘솔을 엽니다.
2. 빌드 프로젝트에서 태그를 추가할 프로젝트의 이름을 선택합니다.
3. 탐색 창에서 설정을 선택합니다. 빌드 프로젝트 태그를 선택합니다.
4. 프로젝트에 태그가 추가되지 않은 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.

프로젝트에 태그 추가(AWS CLI)

프로젝트를 생성할 때 프로젝트에 태그를 추가하려면 빌드 프로젝트 생성(AWS CLI) 단원을 참조하십시오. `create-project.json`에서 태그를 추가합니다.

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

성공한 경우 이 명령은 아무 것도 반환하지 않습니다.

프로젝트의 태그 보기

태그는 AWS 리소스를 식별 및 구성하고 리소스에 대한 액세스를 관리하는 데 도움이 될 수 있습니다. 태그 사용에 대한 자세한 내용은 태그 지정 모범 사례 백서를 참조하십시오. 태그 기반 액세스 정책의 예는 태그를 사용하여 AWS CodeBuild 리소스에 대한 액세스 제어 단원을 참조하세요.

프로젝트의 태그 보기(콘솔)

CodeBuild 콘솔을 사용하여 CodeBuild 프로젝트와 연결된 태그를 볼 수 있습니다.

1. https://console.aws.amazon.com/codebuild/에서 CodeBuild 콘솔을 엽니다.
2. 빌드 프로젝트에서 태그를 보려는 프로젝트의 이름을 선택합니다.
3. 탐색 창에서 설정을 선택합니다. 빌드 프로젝트 태그를 선택합니다.

프로젝트의 태그 보기(AWS CLI)

빌드 프로젝트의 태그를 보려면 다음 명령을 실행합니다. `--names` 파라미터에 대해 프로젝트 이름을 사용합니다.

```
aws codebuild batch-get-projects --names your-project-name
```

성공하면, 이 명령은 다음과 같은 내용을 포함하는 빌드 프로젝트에 대한 JSON 형식의 정보를 반환합니다.

```
{
    "tags": {
        "Status": "Secret",
```

```
        "Team": "JanesProject"
    }
}
```

프로젝트에 태그가 없는 경우에는 tags 섹션이 비어 있습니다.

```
"tags": []
```

프로젝트의 태그 편집

프로젝트와 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 내용은 Tags 단원을 참조하십시오.

> **Important**
>
> 프로젝트의 태그를 편집하면 해당 프로젝트에 대한 액세스에 영향을 줄 수 있습니다. 프로젝트에 대한 태그의 이름(키) 또는 값을 편집하기 전에 프로젝트와 같은 리소스에 대한 액세스를 제어하는 태그의 키 또는 값을 사용할 수도 있는 모든 IAM 정책을 검토하세요. 태그 기반 액세스 정책의 예는 태그를 사용하여 AWS CodeBuild 리소스에 대한 액세스 제어 단원을 참조하세요.

프로젝트의 태그 편집(콘솔)

CodeBuild 콘솔을 사용하여 CodeBuild 프로젝트와 연결된 태그를 편집할 수 있습니다.

1. https://console.aws.amazon.com/codebuild/에서 CodeBuild 콘솔을 엽니다.
2. 빌드 프로젝트에서 태그를 편집할 프로젝트의 이름을 선택합니다.
3. 탐색 창에서 설정을 선택합니다. 빌드 프로젝트 태그를 선택합니다.
4. 편집을 선택합니다.
5. 다음 중 하나를 수행하세요.
   - 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.

프로젝트의 태그 편집(AWS CLI)

빌드 프로젝트에서 태그를 추가, 변경 또는 삭제하려면 빌드 프로젝트 설정 변경(AWS CLI) 단원을 참조하십시오. 프로젝트를 업데이트하는 데 사용하는 JSON 형식 데이터의 `tags` 섹션을 업데이트합니다.

프로젝트에서 태그 제거

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

> **Important**
>
> 프로젝트의 태그를 제거하면 해당 프로젝트에 대한 액세스에 영향을 줄 수 있습니다. 프로젝트에서 태그를 제거하기 전에 프로젝트와 같은 리소스에 대한 액세스를 제어하는 태그의 키 또는 값을 사용할 수도 있는 모든 IAM 정책을 검토하세요. 태그 기반 액세스 정책의 예는 태그를 사용하여 AWS CodeBuild 리소스에 대한 액세스 제어 단원을 참조하세요.

프로젝트에서 태그 제거(콘솔)

CodeBuild 콘솔을 사용하면 태그와 CodeBuild 프로젝트 간의 연결을 제거할 수 있습니다.

1. https://console.aws.amazon.com/codebuild/에서 CodeBuild 콘솔을 엽니다.
2. 빌드 프로젝트에서 태그를 제거할 프로젝트의 이름을 선택합니다.
3. 탐색 창에서 설정을 선택합니다. 빌드 프로젝트 태그를 선택합니다.
4. 편집을 선택합니다.
5. 제거할 태그를 찾은 다음 태그 제거를 선택합니다.
6. 태그 제거를 마쳤으면 제출을 선택합니다.

프로젝트에서 태그 제거(AWS CLI)

빌드 프로젝트에서 하나 이상의 태그를 삭제하려면 빌드 프로젝트 설정 변경(AWS CLI) 단원을 참조하십시오. 삭제하려는 태그가 포함되지 않은 업데이트된 태그 목록으로 JSON 형식 데이터의 `tags` 섹션을 업데이트합니다. 모든 태그를 삭제하려면 `tags` 섹션을 다음과 같이 업데이트하십시오.

```
"tags: []"
```

> **Note**
>
> CodeBuild 빌드 프로젝트를 삭제하면 삭제된 빌드 프로젝트에서 모든 태그 연결이 제거됩니다. 프로젝트를 삭제하기 전에 태그를 제거할 필요가 없습니다.

에서 실행기 사용 AWS CodeBuild

AWS CodeBuild 는 GitHub Actions 실행기, 자체 관리형 GitLab 실행기 및 Buildkite 실행기와의 통합을 지원합니다.

주제

- 의 자체 호스팅 GitHub Actions 실행기 AWS CodeBuild
- 의 자체 관리형 GitLab 실행기 AWS CodeBuild
- 의 자체 관리형 Buildkite 실행기 AWS CodeBuild

의 자체 호스팅 GitHub Actions 실행기 AWS CodeBuild

GitHub Action 워크플로 작업을 처리하기 위해 CodeBuild 컨테이너에서 자체 호스팅 GitHub Action 실행기를 설정하도록 프로젝트를 구성할 수 있습니다. 이는 CodeBuild 프로젝트를 사용하여 웹후크를 설정하고 GitHub 작업 워크플로 YAML을 업데이트하여 CodeBuild 시스템에서 호스팅되는 자체 호스팅 실행기를 사용하여 수행할 수 있습니다.

GitHub Action 작업을 실행하도록 CodeBuild 프로젝트를 구성하는 상위 단계는 다음과 같습니다.

1. 아직 생성하지 않은 경우 개인 액세스 토큰을 생성하거나 OAuth 앱에 연결하여 프로젝트를 GitHub에 연결합니다.
2. CodeBuild 콘솔로 이동하여 웹후크를 사용하여 CodeBuild 프로젝트를 생성하고 웹후크 필터를 설정합니다.
3. GitHub에서 GitHub Action 워크플로 YAML을 업데이트하여 빌드 환경을 구성합니다.

자세한 절차는 자습서: CodeBuild 호스팅 GitHub Action 실행기 구성 섹션을 참조하세요.

이 기능을 사용하면 GitHub Actions 워크플로 작업을와 네이티브 통합하여 IAM AWS AWS Secrets Manager , 통합 AWS CloudTrail및 Amazon VPC와 같은 기능을 통해 보안 및 편의를 제공할 수 있습니다. ARM 기반 인스턴스를 포함한 최신 인스턴스 유형에 액세스할 수 있습니다.

주제

- CodeBuild 호스팅 GitHub Action 실행기 정보
- 자습서: CodeBuild 호스팅 GitHub Action 실행기 구성
- 웹후크 문제 해결
- CodeBuild 호스팅 GitHub Action 실행기에서 지원되는 레이블 재정의
- CodeBuild 호스팅 GitHub Action 실행기에서 지원되는 이미지 계산

CodeBuild 호스팅 GitHub Action 실행기 정보

다음은 CodeBuild 호스팅 GitHub Action 실행기에 대한 몇 가지 일반적인 질문입니다.

레이블에 이미지 및 인스턴스 재정의를 언제 포함해야 합니까?

각 GitHub Action 워크플로 작업에 대해 서로 다른 빌드 환경을 지정하기 위해 레이블에 이미지 및 인스턴스 재정의를 포함할 수 있습니다. 이는 여러 CodeBuild 프로젝트 또는 웹후크를 생성할 필요 없이 수행할 수 있습니다. 예를 들어 워크플로 작업에 매트릭스를 사용해야 하는 경우 유용합니다.

```
name: Hello World
on: [push]
jobs:
  Hello-World-Job:
    runs-on:
      - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
        image:${{ matrix.os }}
        instance-size:${{ matrix.size }}
    strategy:
      matrix:
        include:
          - os: arm-3.0
            size: small
          - os: linux-5.0
            size: large
    steps:
      - run: echo "Hello World!"
```

> **Note**
>
> `runs-on`에 GitHub Action 컨텍스트가 포함된 레이블이 여러 개 있는 경우 따옴표가 필요할 수 있습니다.

이 기능에 CloudFormation 를 사용할 수 있습니까?

예, 프로젝트 웹후크에서 GitHub Actions 워크플로 작업 이벤트 필터를 지정하는 필터 그룹을 CloudFormation 템플릿에 포함할 수 있습니다.

```
Triggers:
  Webhook: true
  FilterGroups:
    - - Type: EVENT
        Pattern: WORKFLOW_JOB_QUEUED
```

자세한 내용은 GitHub Webhook 이벤트 필터링(CloudFormation) 섹션을 참조하세요.

CloudFormation 템플릿에서 프로젝트 자격 증명을 설정하는 데 도움이 필요한 경우 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS::CodeBuild::SourceCredential을 참조하세요.

이 기능을 사용할 때 암호를 마스킹하려면 어떻게 해야 하나요?

기본적으로 로그에 인쇄된 보안 암호는 마스킹되지 않습니다. 보안 암호를 마스킹하려면 다음 구문을 사용할 수 있습니다. `::add-mask::`*`value`*. 다음은 YAML에서 이 구문을 사용하는 방법의 예입니다.

```
name: Secret Job
on: [push]
jobs:
  Secret-Job:
    runs-on: codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
    env:
      SECRET_NAME: "secret-name"
    steps:
      - run: echo "::add-mask::$SECRET_NAME"
```

자세한 내용은 GitHub의 로그에서 값 마스킹을 참조하세요.

단일 프로젝트 내의 여러 리포지토리에서 GitHub Action 웹후크 이벤트를 수신할 수 있나요?

CodeBuild는 지정된 조직 또는 엔터프라이즈에서 이벤트를 수신하는 조직 및 글로벌 수준 웹후크를 지원합니다. 자세한 내용은 GitHub 글로벌 및 조직 웹후크 섹션을 참조하세요.

CodeBuild 호스팅 GitHub Action 실행기 사용을 지원하는 리전은 무엇입니까?

CodeBuild 호스팅 GitHub Action 실행기는 모든 CodeBuild 리전에서 지원됩니다. CodeBuild를 사용할 수 AWS 리전 있는 위치에 대한 자세한 내용은 AWS 리전별 서비스를 참조하세요.

CodeBuild 호스팅 GitHub Action 실행기 사용을 지원하는 플랫폼은 무엇입니까?

CodeBuild 호스팅 GitHub Action 실행기는 Amazon EC2와 AWS Lambda 컴퓨팅 모두에서 지원됩니다. Amazon Linux 2, Amazon Linux 2023, Ubuntu 및 Windows Server Core 2019 플랫폼을 사용할 수 있습니다. 자세한 내용은 EC2 컴퓨팅 이미지 및 Lambda 컴퓨팅 이미지 섹션을 참조하세요.

자습서: CodeBuild 호스팅 GitHub Action 실행기 구성

이 자습서에서는 GitHub Action 작업을 실행하도록 CodeBuild 프로젝트를 구성하는 방법을 보여줍니다. GitHub Action과 CodeBuild를 함께 사용하는 방법에 대한 자세한 내용은 자습서: CodeBuild 호스팅 GitHub Action 실행기 구성 섹션을 참조하세요.

이 자습서를 완료하려면 먼저 다음을 수행해야 합니다.

- 개인 액세스 토큰, Secrets Manager 보안 암호, OAuth 앱 또는 GitHub 앱으로 연결합니다. OAuth 앱에 연결하려면 CodeBuild 콘솔을 사용하여 연결해야 합니다. 개인 액세스 토큰을 생성하려는 경우 CodeBuild 콘솔을 사용하거나 ImportSourceCredentials API를 사용할 수 있습니다. 자세한 지침은 CodeBuild의 GitHub 및 GitHub Enterprise Server 액세스 섹션을 참조하세요.
- CodeBuild를 GitHub 계정에 연결합니다. 이렇게 하려면 다음 중 하나를 수행할 수 있습니다.
  - 콘솔에서 GitHub를 소스 공급자로 추가할 수 있습니다. 개인 액세스 토큰, Secrets Manager 보안 암호, OAuth 앱 또는 GitHub 앱으로 연결할 수 있습니다. 지침은 CodeBuild의 GitHub 및 GitHub Enterprise Server 액세스 섹션을 참조하세요.
  - ImportSourceCredentials API를 통해 GitHub 자격 증명을 가져올 수 있습니다. 이는 개인 액세스 토큰으로만 수행할 수 있습니다. OAuth 앱을 사용하여 연결하는 경우 콘솔을 사용하여 대신 연결해야 합니다. 지침은 액세스 토큰을 사용하여 GitHub에 연결(CLI) 섹션을 참조하세요.

> **Note**
>
> 이는 계정에 대해 GitHub에 연결하지 않은 경우에만 수행하면 됩니다.

1단계: 웹후크를 사용하여 CodeBuild 프로젝트 생성

이 단계에서는 웹후크를 사용하여 CodeBuild 프로젝트를 생성하고 GitHub 콘솔에서 검토합니다. 소스 공급자로 GitHub Enterprise를 선택할 수도 있습니다. GitHub Enterprise 내에서 웹후크를 생성하는 방법에 대한 자세한 내용은 GitHub 수동 웹후크 섹션을 참조하세요.

웹후크를 사용하여 CodeBuild 프로젝트를 생성하려면

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 빌드 프로젝트를 생성합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 및 빌드 실행(콘솔) 섹션을 참조하세요.
3. 프로젝트 유형에서 Runner 프로젝트를 선택합니다.

   러너에서:

   a. Runner 공급자에서 GitHub를 선택합니다.

   b. 러너 위치에서 리포지토리를 선택합니다.

   c. 리포지토리 아래의 리포지토리 URL에서 https://github.com/user-name/repository-name 선택합니다.

   > **Note**
   >
   > 기본적으로 프로젝트는 단일 리포지토리에 대한 `WORKFLOW_JOB_QUEUED` 이벤트만 수신합니다. 조직 또는 엔터프라이즈 내의 모든 리포지토리에 대한 이벤트를 수신하려면 GitHub 글로벌 및 조직 웹후크 섹션을 참조하세요.

4. • 환경에서 다음과 같이 합니다.
   - 지원되는 환경 이미지와 컴퓨팅을 선택합니다. GitHub Action 워크플로 YAML의 레이블을 사용하여 이미지 및 인스턴스 설정을 재정의할 수 있는 옵션이 있습니다. 자세한 내용은 2단계: GitHub Action 워크플로 YAML 업데이트 섹션을 참조하세요.
   - Buildspec에서 다음과 같이 합니다.
     - `buildspec-override:true`가 레이블로 추가되지 않는 한 buildspec은 무시됩니다. 대신 CodeBuild는 자체 호스팅 러너를 설정하는 명령을 사용하도록 이를 재정의합니다.
5. 기본값으로 계속 진행한 다음 빌드 프로젝트 생성을 선택합니다.
6. `https://github.com/user-name/repository-name/settings/hooks`에서 GitHub 콘솔을 열어 웹후크가 생성되었고 워크플로 작업 이벤트를 전달할 수 있는지 확인합니다.

2단계: GitHub Action 워크플로 YAML 업데이트

이 단계에서는 GitHub에서 GitHub Action 워크플로 YAML 파일을 업데이트하여 빌드 환경을 구성하고 CodeBuild에서 GitHub Action 자체 호스팅 실행기를 사용합니다. 자세한 내용은 자체 호스팅 실행

기에서 레이블 사용 및 CodeBuild 호스팅 GitHub Action 실행기에서 지원되는 레이블 재정의 섹션을 참조하세요.

GitHub Action 워크플로 YAML 업데이트

GitHub Action 워크플로 YAML의 `GitHub` 설정으로 이동하여 `runs-on`을 업데이트하여 빌드 환경을 구성합니다. 이렇게 하려면 다음 중 하나를 수행할 수 있습니다.

- 프로젝트 이름과 실행 ID를 지정할 수 있습니다. 이 경우 빌드는 컴퓨팅, 이미지, 이미지 버전 및 인스턴스 크기에 대한 기존 프로젝트 구성을 사용합니다. 프로젝트 이름은 GitHub Action 작업의 AWS관련 설정을 특정 CodeBuild 프로젝트에 연결하는 데 필요합니다. YAML에 프로젝트 이름을 포함하면 CodeBuild가 올바른 프로젝트 설정으로 작업을 호출할 수 있습니다. CodeBuild는 실행 ID를 제공하여 빌드를 특정 워크플로 실행에 매핑하고 워크플로 실행이 취소되면 빌드를 중지합니다. 자세한 내용은 github 컨텍스트를 참조하세요.

  ```
  runs-on: codebuild-<project-name>-${{ github.run_id }}-${{ github.run_attempt }}
  ```

  > **Note**
  >
  > *<project-name>*이 이전 단계에서 생성한 프로젝트의 이름과 일치하는지 확인합니다. 일치하지 않으면 CodeBuild는 웹후크를 처리하지 않고 GitHub Action 워크플로가 중단될 수 있습니다.

  다음은 GitHub Action 워크플로 YAML의 예입니다.

  ```
  name: Hello World
  on: [push]
  jobs:
    Hello-World-Job:
      runs-on:
        - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
      steps:
        - run: echo "Hello World!"
  ```

- 레이블에서 이미지 및 컴퓨팅 유형을 재정의할 수도 있습니다. 큐레이션된 이미지 목록은 섹션을 참조CodeBuild 호스팅 GitHub Action 실행기에서 지원되는 이미지 계산하세요. 사용자 지정 이미지 사용은 섹션을 참조하세요CodeBuild 호스팅 GitHub Action 실행기에서 지원되는 레이블 재정의. 레이

블의 컴퓨팅 유형 및 이미지가 프로젝트의 환경 설정을 재정의합니다. CodeBuild EC2 또는 Lambda 컴퓨팅 빌드의 환경 설정을 재정의하려면 다음 구문을 사용합니다.

```
runs-on:
  - codebuild-<project-name>-${{ github.run_id }}-${{ github.run_attempt }}
    image:<environment-type>-<image-identifier>
    instance-size:<instance-size>
```

다음은 GitHub Action 워크플로 YAML의 예입니다.

```
name: Hello World
on: [push]
jobs:
  Hello-World-Job:
    runs-on:
      - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
        image:arm-3.0
        instance-size:small
    steps:
      - run: echo "Hello World!"
```

- 레이블에서 빌드에 사용되는 플릿을 재정의할 수 있습니다. 이렇게 하면 지정된 플릿을 사용하도록 프로젝트에 구성된 플릿 설정이 재정의됩니다. 자세한 내용은 예약 용량 플릿에서 빌드 실행 섹션을 참조하세요. Amazon EC2 컴퓨팅 빌드의 플릿 설정을 재정의하려면 다음 구문을 사용합니다.

```
runs-on:
  - codebuild-<project-name>-${{ github.run_id }}-${{ github.run_attempt }}
    fleet:<fleet-name>
```

빌드에 사용되는 플릿과 이미지를 모두 재정의하려면 다음 구문을 사용합니다.

```
runs-on:
  - codebuild-<project-name>-${{ github.run_id }}-${{ github.run_attempt }}
    fleet:<fleet-name>
    image:<environment-type>-<image-identifier>
```

다음은 GitHub Action 워크플로 YAML의 예입니다.

```
name: Hello World
on: [push]
```

```
jobs:
  Hello-World-Job:
    runs-on:
      - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
        fleet:myFleet
        image:arm-3.0
    steps:
      - run: echo "Hello World!"
```

- 사용자 지정 이미지에서 GitHub Action 작업을 실행하려면 CodeBuild 프로젝트에서 사용자 지정 이미지를 구성하고 이미지 재정의 레이블을 제공하지 않아도 됩니다. CodeBuild는 이미지 재정의 레이블이 제공되지 않은 경우 프로젝트에 구성된 이미지를 사용합니다.
- 선택적으로 CodeBuild가 지원하는 레이블 이외의 레이블을 제공할 수 있습니다. 이러한 레이블은 빌드의 속성을 재정의할 목적으로 무시되지만 웹후크 요청에 실패하지는 않습니다. 예를 들어 레이블로 `testLabel`을 추가해도 레이블은 빌드 실행을 방해하지 않습니다.

> **Note**
>
> CodeBuild 환경에서 GitHub 호스팅 실행기가 제공하는 종속 항목을 사용할 수 없는 경우 워크플로 실행에서 GitHub Action을 사용하여 종속 항목을 설치할 수 있습니다. 예를 들어 setup-python 작업을 사용하여 빌드 환경에 Python을 설치할 수 있습니다.

INSTALL, PRE_BUILD 및 POST_BUILD 단계에서 buildspec 명령 실행

기본적으로 CodeBuild는 자체 호스팅 GitHub Action 빌드를 실행할 때 모든 buildspec 명령을 무시합니다. 빌드 중에 buildspec 명령을 실행하려면 `buildspec-override:true`를 레이블에 접미사로 추가할 수 있습니다.

```
runs-on:
  - codebuild-<project-name>-${{ github.run_id }}-${{ github.run_attempt }}
    buildspec-override:true
```

이 명령을 사용하면 CodeBuild는 컨테이너의 기본 소스 폴더에 `actions-runner`라는 폴더를 생성합니다. `BUILD` 단계 중에 GitHub Action 실행기가 시작되면 실행기가 `actions-runner` 디렉터리에서 실행됩니다.

자체 호스팅 GitHub Action 빌드에서 buildspec 재정의를 사용하는 경우 몇 가지 제한이 있습니다.

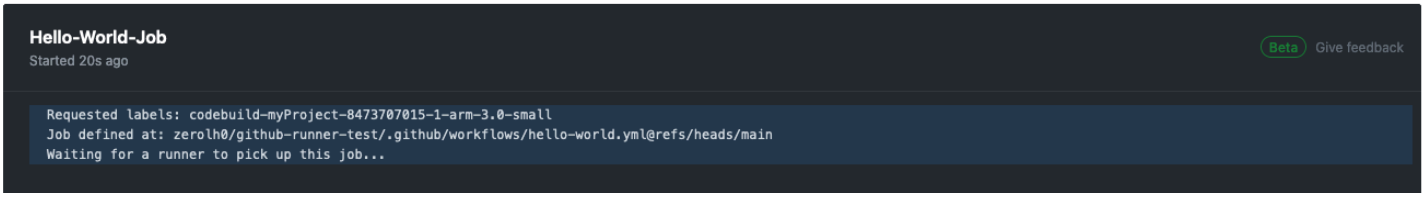
- CodeBuild는 BUILD 단계에서 자체 호스팅된 실행기가 실행되므로 BUILD 단계 중에 buildspec 명령을 실행하지 않습니다.
- CodeBuild는 DOWNLOAD_SOURCE 단계 중에 기본 또는 보조 소스를 다운로드하지 않습니다. buildspec 파일이 구성된 경우 해당 파일만 프로젝트의 기본 소스에서 다운로드됩니다.
- PRE_BUILD 또는 INSTALL 단계에서 빌드 명령이 실패하면 CodeBuild는 자체 호스팅된 실행기를 시작하지 않으므로 GitHub Actions 워크플로 작업을 수동으로 취소해야 합니다.
- CodeBuild는 만료 시간이 1시간인 DOWNLOAD_SOURCE 단계 중에 실행기 토큰을 가져옵니다. PRE_BUILD 또는 INSTALL 단계가 1시간을 초과하면 GitHub 자체 호스팅 실행기가 시작되기 전에 실행기 토큰이 만료될 수 있습니다.

3단계: 결과 검토

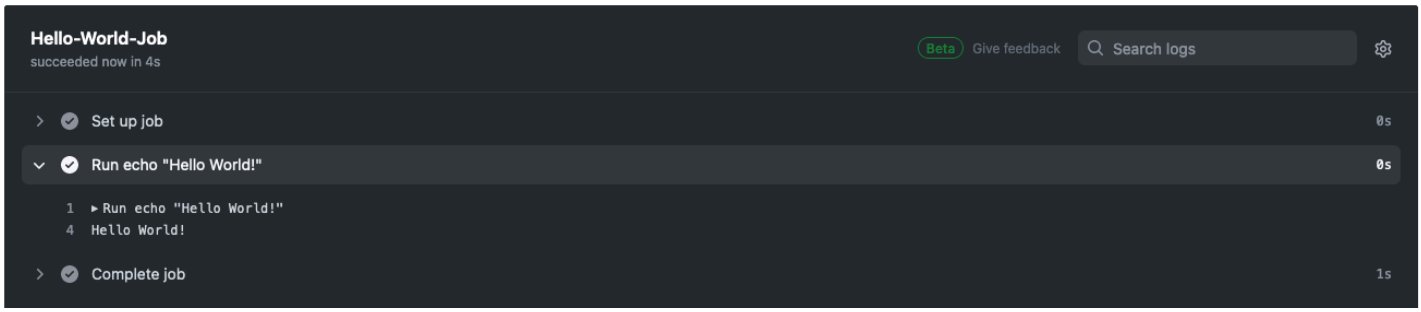
GitHub Action 워크플로 실행이 발생할 때마다 CodeBuild는 웹후크를 통해 워크플로 작업 이벤트를 수신합니다. 워크플로의 각 작업에 대해 CodeBuild는 임시 GitHub Action 실행기를 실행하기 위한 빌드를 시작합니다. 실행기는 단일 워크플로 작업을 실행할 책임이 있습니다. 작업이 완료되면 실행기와 관련 빌드 프로세스가 즉시 종료됩니다.

워크플로 작업 로그를 보려면 GitHub에서 리포지토리로 이동하여 작업을 선택하고 원하는 워크플로를 선택한 다음 로그를 검토하려는 특정 작업을 선택합니다.

CodeBuild에서 자체 호스팅된 실행기가 작업을 픽업할 때까지 기다리는 동안 로그에서 요청된 레이블을 검토할 수 있습니다.



작업이 완료되면 작업 로그를 볼 수 있습니다.



GitHub Actions 실행기 구성 옵션

프로젝트 구성에서 다음 환경 변수를 지정하여 자체 호스팅 러너의 설정 구성을 수정할 수 있습니다.

`CODEBUILD_CONFIG_GITHUB_ACTIONS_ORG_REGISTRATION_NAME`

CodeBuild는 자체 호스팅 러너를이 환경 변수의 값으로 지정된 조직 이름에 등록합니다. 조직 수준에서 러너를 등록하는 방법과 필요한 권한에 대한 자세한 내용은 조직의 just-in-time 러너에 대한 구성 생성을 참조하세요.

`CODEBUILD_CONFIG_GITHUB_ACTIONS_ENTERPRISE_REGISTRATION_NAME`

CodeBuild는 자체 호스팅 러너를이 환경 변수의 값으로 지정된 엔터프라이즈 이름에 등록합니다. 엔터프라이즈 수준에서 러너를 등록하는 방법과 필요한 권한에 대한 자세한 내용은 엔터프라이즈 just-in-time 러너에 대한 구성 생성을 참조하세요.

> **Note**
>
> 엔터프라이즈 러너는 기본적으로 조직 리포지토리에서 사용할 수 없습니다. 자체 호스팅 러너가 워크플로 작업을 픽업하려면 러너 그룹 액세스 설정을 구성해야 할 수 있습니다. 자세한 내용은 엔터프라이즈 러너를 리포지토리에 사용할 수 있도록 만들기를 참조하세요.

`CODEBUILD_CONFIG_GITHUB_ACTIONS_RUNNER_GROUP_ID`

CodeBuild는이 환경 변수의 값으로 저장된 정수 실행기 그룹 ID에 자체 호스팅 실행기를 등록합니다. 기본적으로이 값은 1입니다. 자체 호스팅 러너 그룹에 대한 자세한 내용은 그룹을 사용하여 자체 호스팅 러너에 대한 액세스 관리를 참조하세요.

`CODEBUILD_CONFIG_GITHUB_ACTIONS_ORG_REGISTRATION_NAME`

GitHub Actions 워크플로 YAML 파일을 사용하여 조직 수준 실행기 등록을 구성하려면 다음 구문을 사용할 수 있습니다.

```
name: Hello World
on: [push]
jobs:
  Hello-World-Job:
    runs-on:
      - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
        organization-registration-name:myOrganization
    steps:
```

```
      - run: echo "Hello World!"
```

CODEBUILD_CONFIG_GITHUB_ACTIONS_ENTERPRISE_REGISTRATION_NAME

GitHub Actions 워크플로 YAML 파일을 사용하여 엔터프라이즈 수준 실행기 등록을 구성하려면 다음 구문을 사용할 수 있습니다.

```
name: Hello World
on: [push]
jobs:
  Hello-World-Job:
    runs-on:
      - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
        enterprise-registration-name:myEnterprise
    steps:
      - run: echo "Hello World!"
```

CODEBUILD_CONFIG_GITHUB_ACTIONS_RUNNER_GROUP_ID

GitHub Actions 워크플로 YAML 파일을 사용하여 특정 러너 그룹 ID에 러너 등록을 구성하려면 다음 구문을 사용할 수 있습니다.

```
name: Hello World
on: [push]
jobs:
  Hello-World-Job:
    runs-on:
      - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
        registration-group-id:3
    steps:
      - run: echo "Hello World!"
```

GitHub Action 웹후크 이벤트 필터링(CloudFormation)

템플릿의 CloudFormation 다음 YAML 형식 부분은 빌드가 true로 평가될 때 빌드를 트리거하는 필터 그룹을 생성합니다. 다음 필터 그룹은 정규식 `\[CI-CodeBuild\]`와 일치하는 워크플로 이름을 가진 GitHub Action 워크플로 작업 요청을 지정합니다.

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
```

```
    Name: MyProject
    ServiceRole: service-role
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      Type: LINUX_CONTAINER
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:5.0
    Source:
      Type: GITHUB
      Location: CODEBUILD_DEFAULT_WEBHOOK_SOURCE_LOCATION
    Triggers:
      Webhook: true
      ScopeConfiguration:
        Name: organization-name
        Scope: GITHUB_ORGANIZATION
      FilterGroups:
        - - Type: EVENT
            Pattern: WORKFLOW_JOB_QUEUED
          - Type: WORKFLOW_NAME
            Pattern: \[CI-CodeBuild\]
```

GitHub Action 웹후크 이벤트 필터링(AWS CDK)

다음 AWS CDK 템플릿은 빌드가 true로 평가될 때 빌드를 트리거하는 필터 그룹을 생성합니다. 다음 필터 그룹은 GitHub Action 워크플로 작업 요청을 지정합니다.

```
import { aws_codebuild as codebuild } from 'aws-cdk-lib';
import {EventAction, FilterGroup} from "aws-cdk-lib/aws-codebuild";

const source = codebuild.Source.gitHub({
      owner: 'owner',
      repo: 'repo',
      webhook: true,
      webhookFilters: [FilterGroup.inEventOf(EventAction.WORKFLOW_JOB_QUEUED)],
    })
```

GitHub Action 웹후크 이벤트 필터링(Terraform)

다음 Terraform 템플릿은 빌드가 true로 평가될 때 빌드를 트리거하는 필터 그룹을 생성합니다. 다음 필터 그룹은 GitHub Action 워크플로 작업 요청을 지정합니다.

```
resource "aws_codebuild_webhook" "example" {
```

```
  project_name = aws_codebuild_project.example.name
  build_type   = "BUILD"
  filter_group {
    filter {
      type    = "EVENT"
      pattern = "WORKFLOW_JOB_QUEUED"
    }
  }
}
```

GitHub Action 웹후크 이벤트 필터링(AWS CLI)

다음 AWS CLI 명령은 빌드가 true로 평가될 때 빌드를 트리거하는 GitHub Actions 워크플로 작업 요청 필터 그룹을 사용하여 자체 호스팅 GitHub Actions 실행기 프로젝트를 생성합니다. GitHub

```
aws codebuild create-project \
--name <project name> \
--source "{\"type\":\"GITHUB\",\"location\":\"<repository location>\",\"buildspec\":
\"\"}" \
--artifacts {"\"type\":\"NO_ARTIFACTS\""} \
--environment "{\"type\": \"LINUX_CONTAINER\",\"image\": \"aws/codebuild/amazonlinux-
x86_64-standard:5.0\",\"computeType\": \"BUILD_GENERAL1_MEDIUM\"}" \
--service-role "<service role ARN>"
```

```
aws codebuild create-webhook \
--project-name <project name> \
--filter-groups "[[{\"type\":\"EVENT\",\"pattern\":\"WORKFLOW_JOB_QUEUED\"}]]"
```

웹후크 문제 해결

문제: 자습서: CodeBuild 호스팅 GitHub Action 실행기 구성에서 설정한 웹후크가 작동하지 않거나 GitHub에서 워크플로 작업이 중단되고 있습니다.

가능한 원인:

- Webhook 워크플로 작업 이벤트가 빌드를 트리거하지 못할 수 있습니다. 응답 로그를 검토하여 응답 또는 오류 메시지를 확인합니다.
- 레이블 구성으로 인해 작업이 잘못된 러너 에이전트에 할당되고 있습니다. 이 문제는 단일 워크플로 실행 내의 작업 중 하나에 다른 작업보다 레이블이 적은 경우에 발생할 수 있습니다. 예를 들어 동일한 워크플로 실행에 다음 레이블이 있는 작업이 두 개 있는 경우:

- 작업 1: `codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}`
- 작업 2: `codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}`, `instance-size:medium`

자체 호스팅된 GitHub Actions 작업을 라우팅할 때 GitHub는 모든 작업의 지정된 레이블이 있는 러너로 작업을 라우팅합니다. 이 동작은 작업 1 또는 작업 2에 대해 생성된 러너가 작업 1을 선택할 수 있지만 추가 레이블이 있으므로 작업 2에 대해 생성된 러너만 작업 2를 선택할 수 있음을 의미합니다. 작업 2용으로 생성된 실행기가 작업 1을 선택하면 작업 1 실행기에 레이블이 없으므로 작업 2가 중단됩니다. `instance-size:medium`

권장 솔루션:

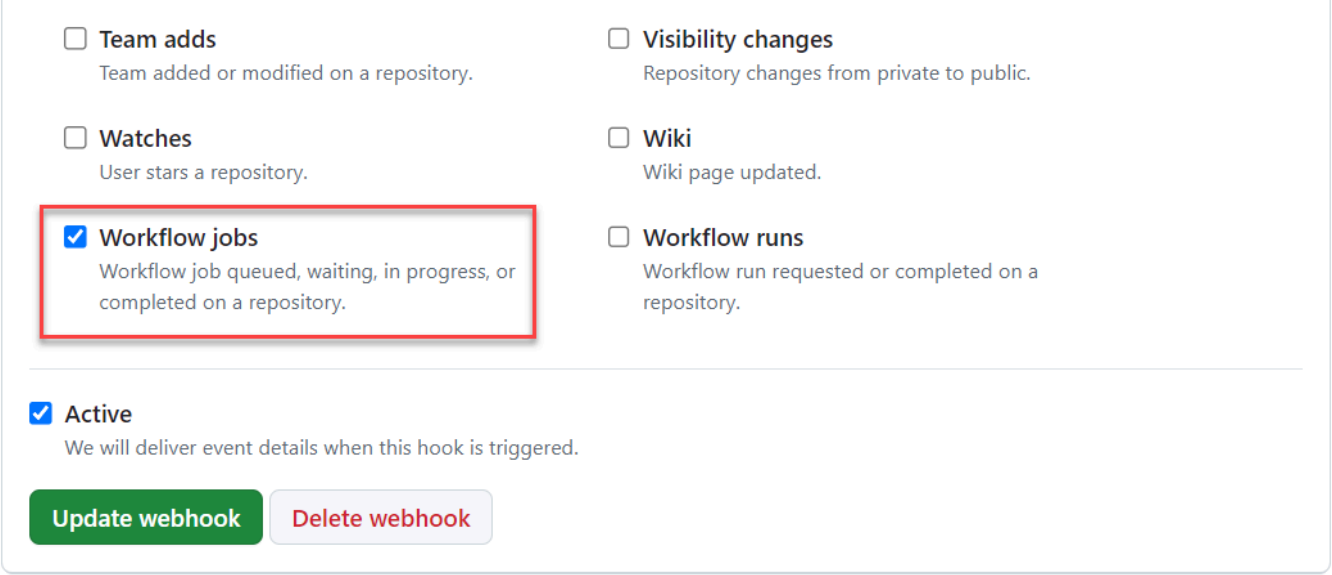
동일한 워크플로 실행 내에서 여러 작업을 생성할 때 각 작업에 대해 동일한 수의 레이블 재정의를 사용하거나 `job1` 또는와 같은 사용자 지정 레이블을 각 작업에 할당합니다`job2`.

각 작업에 고유한 사용자 지정 레이블을 할당하는 것이 가장 신뢰할 수 있는 옵션입니다. 따라서 작업의 레이블 세트가 다른 작업의 하위 집합이 되지 않습니다. 모든 작업에 다른 작업이 없는 레이블이 있는 경우 GitHub는 각 작업에서 사용하는 다른 재정의 수에 관계없이 각 작업을 해당 작업에 대해 생성된 실행기와만 일치시킵니다. 다음 워크플로에서 각 작업에는 고유한 레이블(`job1` 및 `job2`)이 있습니다. 작업 2에는 추가 `instance-size:medium` 재정의가 있지만 작업 1은 더 이상 작업 2에 대해 생성된 러너와 일치시킬 수 없으므로 두 작업 모두 중단되지 않습니다.

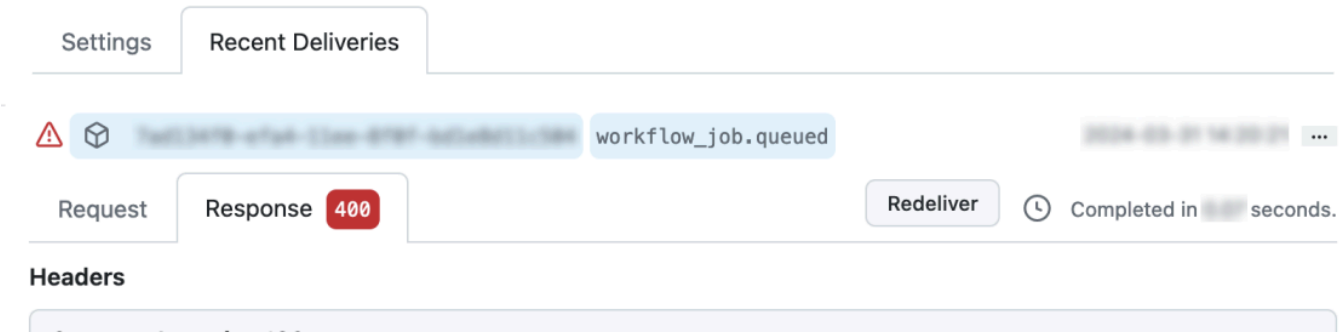
```
name: Hello World
on: [push]
jobs:
  job1:
    runs-on:
      - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
      - job1
    steps:
      - run: echo "Hello from Job 1!"
  job2:
    runs-on:
      - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
      - instance-size:medium
      - job2
    steps:
      - run: echo "Hello from Job 2!"
```

오류가 지속되면 다음 지침에 따라 문제를 디버깅합니다.

1. `https://github.com/user-name/repository-name/settings/hooks`에서 GitHub 콘솔을 열어 리포지토리의 웹후크 설정을 확인합니다. 이 페이지에는 리포지토리를 위해 생성된 웹후크가 표시됩니다.
2. 편집을 선택하고 워크플로 작업 이벤트를 전달하도록 웹후크가 활성화되어 있는지 확인합니다.



3. 최근 전송 탭으로 이동하여 해당 `workflow_job.queued` 이벤트를 찾아 이벤트를 확장합니다.
4. 페이로드의 레이블 필드를 검토하고 예상대로 작동하는지 확인합니다.
5. 마지막으로 응답 탭을 검토합니다. 응답 탭에는 CodeBuild에서 반환된 응답 또는 오류 메시지가 포함되어 있기 때문입니다.



6. 또는 GitHub의 API를 사용하여 웹후크 실패를 디버깅할 수 있습니다. 리포지토리 웹후크 API의 목록 전송을 사용하여 웹후크의 최근 전송을 볼 수 있습니다.

```
gh api \
  -H "Accept: application/vnd.github+json" \
  -H "X-GitHub-Api-Version: 2022-11-28" \
  /repos/owner/repo/hooks/hook-id/deliveries
```

디버깅하고 전송 ID를 기록하려는 웹후크 전송을 찾은 후 리포지토리 웹후크에 대한 전송 가져오기 API를 사용할 수 있습니다. 웹후크의 전송 페이로드에 대한 CodeBuild의 응답은 `response` 섹션에서 확인할 수 있습니다.

```
gh api \
  -H "Accept: application/vnd.github+json" \
  -H "X-GitHub-Api-Version: 2022-11-28" \
  /repos/owner/repo/hooks/hook-id/deliveries/delivery-id
```

문제: 배포 보호 규칙이 활성화된 GitHub 작업은 배포가 승인되기 전에 CodeBuild 내에 빌드됩니다.

가능한 원인: CodeBuild는 배포가 승인되었는지 확인하기 위해 GitHub Actions 작업과 연결된 배포 및 환경을 가져옵니다. CodeBuild가 배포 또는 환경을 가져오지 못하면 CodeBuild 빌드가 조기에 트리거될 수 있습니다.

권장 솔루션: CodeBuild 프로젝트와 연결된 자격 증명에 GitHub 내의 배포 및 작업에 대한 읽기 권한이 있는지 확인합니다.

CodeBuild 호스팅 GitHub Action 실행기에서 지원되는 레이블 재정의

GitHub Action 워크플로 YAML에서 자체 호스팅 실행기 빌드를 수정하는 다양한 레이블 재정의를 제공할 수 있습니다. CodeBuild에서 인식하지 못하는 빌드는 무시되지만 웹후크 요청에 실패하지는 않습니다. 다음 워크플로 YAML은 이미지, 인스턴스 크기, 플릿 및 buildspec에 대한 `job1` 재정의를 사용하고 이미지 재정의만 `job2` 사용하는 두 가지 작업을 정의합니다. GitHub가 각 작업을 생성된 러너로 라우팅하도록 각 작업에 고유한 레이블(`job1` 및 `job2`)을 지정합니다.

```
name: Hello World
on: [push]
jobs:
  job1:
    runs-on:
      - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
      - image:arm-3.0
      - instance-size:small
      - fleet:myFleet
      - buildspec-override:true
      - job1
    steps:
      - run: echo "Hello from Job 1!"
  job2:
```

```
    runs-on:
      - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
      - image:linux-5.0
      - job2
    steps:
      - run: echo "Hello from Job 2!"
```

매트릭스 전략을 사용하여 여러 구성에서 단일 작업을 실행할 수도 있습니다. 단일 매트릭스 작업의 확장에는 항상 동일한 수의 레이블이 있으므로 러너 일치 문제는 해당 레이블에 영향을 주지 않습니다. 그러나 워크플로 실행에 둘 이상의 매트릭스 작업이 포함된 경우 GitHub가 다른 매트릭스 작업에 대해 생성된 러너와 작업을 일치시키지 않도록 각 매트릭스 작업에 고유한 사용자 지정 레이블(예: `matrix-job-1` 및 `matrix-job-2`)을 지정합니다.

```
name: Hello World
on: [push]
jobs:
  Hello-World-Job:
    runs-on:
      - codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
      - image:${{ matrix.os }}
      - instance-size:${{ matrix.size }}
      - fleet:myFleet
      - buildspec-override:true
      - matrix-job-1
    strategy:
      matrix:
        include:
          - os: arm-3.0
            size: small
          - os: linux-5.0
            size: large
    steps:
      - run: echo "Hello World!"
```

> **Note**
>
> 워크플로 작업이 GitHub에서 중단되는 경우 웹후크 문제 해결 및 사용자 지정 레이블을 사용하여 작업 라우팅을 참조하세요.

`codebuild-<project-name>-${{github.run_id}}-${{github.run_attempt}}` (필수)

- 예시: `codebuild-fake-project-${{ github.run_id }}-${{ github.run_attempt }}`
- 모든 GitHub Action 워크플로 YAML에 필요합니다. *<project name>*은 자체 호스팅 실행기 웹후크가 구성된 프로젝트의 이름과 같아야 합니다.

`image:`*<environment-type>*`-`*<image-identifier>*

- 예시: `image:arm-3.0`
- 큐레이션된 이미지로 자체 호스팅 러너 빌드를 시작할 때 사용되는 이미지 및 환경 유형을 재정의합니다. 지원되는 값에 대한 자세한 내용은 CodeBuild 호스팅 GitHub Action 실행기에서 지원되는 이미지 계산 섹션을 참조하세요.
  - 사용자 지정 이미지와 함께 사용되는 이미지 및 환경 유형을 재정의하려면 `image:custom-`*<environment-type>*`-`*<custom-image-identifier>*
  - 예시: `image:custom-arm-public.ecr.aws/codebuild/amazonlinux-aarch64-standard:3.0`

> **Note**
>
> 사용자 지정 이미지가 프라이빗 레지스트리에 있는 경우 섹션을 참조하세요자체 호스팅 러너에 대한 프라이빗 레지스트리 자격 증명 구성.

`instance-size:`*<instance-size>*

- 예시: `instance-size:medium`
- 자체 호스팅 실행기 빌드를 시작할 때 사용되는 인스턴스 유형을 재정의합니다. 지원되는 값에 대한 자세한 내용은 CodeBuild 호스팅 GitHub Action 실행기에서 지원되는 이미지 계산 섹션을 참조하세요.

`fleet:`*<fleet-name>*

- 예시: `fleet:myFleet`
- 지정된 플릿을 사용하도록 프로젝트에 구성된 플릿 설정을 재정의합니다. 자세한 내용은 예약 용량 플릿에서 빌드 실행 섹션을 참조하세요.

`buildspec-override:`*<boolean>*

- 예시: `buildspec-override:true`
- `true`로 설정된 경우 빌드가 `INSTALL`, `PRE_BUILD` 및 `POST_BUILD` 단계에서 buildspec 명령을 실행하도록 허용합니다.

`<unique-label>` (워크플로 실행에 여러 작업이 있는 경우 권장)

- 예시: `job1`
- 정의한 사용자 지정 레이블로, 동일한 워크플로 실행에서 다른 작업이 사용하지 않습니다. 각 작업에 고유한 레이블을 할당하면 GitHub는 각 작업을 해당 작업에 대해 생성된 러너로만 라우팅합니다. 그렇지 않으면 동일한 워크플로 실행의 작업에서 레이블 재정의 수가 다르면 레이블이 더 적은 작업을 더 많은 레이블이 있는 작업에 대해 생성된 러너가 선택할 수 있으므로 러너 없이 두 번째 작업을 남길 수 있습니다. 자세한 내용은 웹후크 문제 해결 섹션을 참조하세요.

단일 레이블 재정의

CodeBuild를 사용하면 다음을 사용하여 단일 레이블에 여러 재정의를 제공할 수 있습니다.

> **Note**
>
> 모든 재정의는 단일 레이블로 결합되므로 각 작업의 실행기는 정확히 하나의 레이블로 등록됩니다. 즉,에 설명된 다중 작업 실행기 일치 문제는 각 작업에 고유한 레이블이 없더라도이 형식으로 발생할 웹후크 문제 해결 수 없습니다. 이 동작을 선호하고 작업에 지원되는 재정의만 필요한 경우 단일 레이블 형식을 사용합니다. 지원되지 않는 재정의가 필요한 경우 사용하지 마세요. 예를 들어 사용자 지정 이미지(프라이빗 레지스트리의 이미지 포함)는 `image:custom-<environment-type>-<custom-image-identifier>` 레이블 형식을 사용해야만 지정할 수 있습니다. 이러한 재정의의 경우이 주제의 앞부분에서 설명한 별도의 레이블 형식을 사용합니다.

- Amazon EC2/Lambda 컴퓨팅 빌드의 환경 설정을 재정의하려면 다음 구문을 사용합니다.

```
runs-on: codebuild-<project-name>-${{ github.run_id }}-
${{ github.run_attempt }}-<environment-type>-<image-identifier>-<instance-size>
```

- Amazon EC2 컴퓨팅 빌드에 대한 플릿 설정을 재정의하려면 다음 구문을 사용합니다.

```
runs-on: codebuild-<project-name>-${{ github.run_id }}-${{ github.run_attempt }}-
fleet-<fleet-name>
```

- 빌드에 사용되는 플릿과 이미지를 모두 재정의하려면 다음 구문을 사용합니다.

```
runs-on: codebuild-<project-name>-${{ github.run_id }}-
${{ github.run_attempt }}-image-<image-version>-fleet-<fleet-name>
```

- 빌드 중에 buildspec 명령을 실행하려면 `-with-buildspec`를 레이블에 접미사로 추가할 수 있습니다.

```
runs-on: codebuild-<project-name>-${{ github.run_id }}-
${{ github.run_attempt }}-<image>-<image-version>-<instance-size>-with-buildspec
```

- 선택적으로 이미지를 재정의하지 않고 인스턴스 크기 재정의를 제공할 수 있습니다. Amazon EC2 빌드의 경우 환경 유형과 이미지 식별자를 모두 제외할 수 있습니다. Lambda 빌드의 경우 이미지 식별자를 제외할 수 있습니다.

CodeBuild 호스팅 GitHub Action 실행기에서 지원되는 이미지 계산

자습서: CodeBuild 호스팅 GitHub Action 실행기 구성에서 구성한 레이블에서 처음 세 열의 값을 사용하여 Amazon EC2 환경 설정을 재정의할 수 있습니다. CodeBuild는 다음과 같은 Amazon EC2 컴퓨팅 이미지를 제공합니다. 해당 내용은 다음을 참조하세요.

환경 유형	이미지 식별자	인스턴스 크기	플랫폼	해결된 이미지	정의

`linux`	`4.0`	`small`
		`medium`
		`large`
		`xlarge`
			Amazon Linux 2	`aws/codebuild/amazonlinux-x86_64-standard:4.0`	al/standard/4.0

`linux`	`5.0`	`2xlarge` `gpu_small`	Amazon Linux 2023	`aws/codebuild/amazonlinux-x86_64-sta`	al/standard/5.0

환경 유형	이미지 식별자	인스턴스 크기	플랫폼	해결된 이미지	정의
		gpu_large		ndard:5.0	
linux-ec2	latest	small medium large	Amazon Linux 2023	aws/codebuild/ami/amazonlinux-x86_64-base:latest	없음
arm	2.0	small medium large xlarge 2xlarge	Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-standard:2.0	al/aarch64/standard/2.0
arm	3.0		Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-standard:3.0	al/aarch64/standard/3.0
arm-ec2	latest	small medium large	Amazon Linux 2023	aws/codebuild/ami/amazonlinux-arm-base:latest	없음

환경 유형	이미지 식별자	인스턴스 크기	플랫폼	해결된 이미지	정의
ubuntu	5.0	small medium large xlarge 2xlarge gpu_small gpu_large	Ubuntu 20.04	aws/codebuild/standard:5.0	ubuntu/standard/5.0
ubuntu	6.0		Ubuntu 22.04	aws/codebuild/standard:6.0	ubuntu/standard/6.0
ubuntu	7.0		Ubuntu 22.04	aws/codebuild/standard:7.0	ubuntu/standard/7.0
windows	1.0	medium large	Windows Server Core 2019	aws/codebuild/windows-base:2019-1.0	해당 사항 없음
			Windows Server Core 2022	aws/codebuild/windows-base:2022-1.0	해당 사항 없음
windows	2.0		Windows Server Core 2019	aws/codebuild/windows-base:2019-2.0	해당 사항 없음
windows	3.0		Windows Server Core 2019	aws/codebuild/windows-base:2019-3.0	해당 사항 없음

환경 유형	이미지 식별자	인스턴스 크기	플랫폼	해결된 이미지	정의
windows-ec2	2022	medium large	Windows Server Core 2022	aws/codebuild/ami/windows-base:2022	없음

또한 다음 값을 사용하여 Lambda 환경 설정을 재정의할 수 있습니다. CodeBuild Lambda 컴퓨팅에 대한 자세한 내용은 AWS Lambda 컴퓨팅 기반 빌드 실행 섹션을 참조하세요. CodeBuild는 다음 Lambda 컴퓨팅 이미지를 지원합니다.

환경 유형	이미지 식별자	인스턴스 크기			
linux-lambda arm-lambda	dotnet6 go1.21 corretto11 corretto17 corretto21 nodejs18 nodejs20 python3.11 python3.12	1GB 2GB 4GB 8GB 10GB			

환경 유형	이미지 식별자	인스턴스 크기			
	`ruby3.2`				

자세한 내용은 빌드 환경 컴퓨팅 모드 및 유형 및 CodeBuild가 제공하는 도커 이미지 섹션을 참조하세요.

의 자체 관리형 GitLab 실행기 AWS CodeBuild

GitLab은 CI/CD 파이프라인에서 GitLab 작업을 실행하는 두 가지 실행 모드를 제공합니다. 한 가지 모드는 GitLab에서 관리하고 GitLab과 완전히 통합된 GitLab 호스팅 실행기입니다. 다른 모드는 자체 관리형 실행기로, GitLab CI/CD 파이프라인에서 작업을 실행할 수 있도록 사용자 지정 환경을 가져올 수 있습니다.

GitLab CI/CD 파이프라인 작업을 실행하도록 CodeBuild 프로젝트를 구성하는 상위 단계는 다음과 같습니다.

1. 아직 생성하지 않은 경우 개인 OAuth 앱에 연결하여 프로젝트를 GitLab에 연결합니다.
2. CodeBuild 콘솔로 이동하여 웹후크를 사용하여 CodeBuild 프로젝트를 생성하고 웹후크 필터를 설정합니다.
3. GitLab에서 GitLab CI/CD 파이프라인 YAML을 업데이트하여 빌드 환경을 구성합니다.

자세한 절차는 자습서: CodeBuild 호스팅 GitLab 실행기 구성 섹션을 참조하세요.

이 기능을 사용하면 GitLab CI/CD 파이프라인 작업이 AWS와 네이티브 통합될 수 있으며, IAM, AWS CloudTrail및 Amazon VPC와 같은 기능을 통해 보안 및 편의성을 제공합니다. ARM 기반 인스턴스를 포함하여 최신 인스턴스 유형에 액세스할 수 있습니다.

주제

- CodeBuild 호스팅 GitLab 실행기 정보
- 자습서: CodeBuild 호스팅 GitLab 실행기 구성
- CodeBuild 호스팅 GitLab 실행기에서 지원되는 레이블 재정의
- CodeBuild 호스팅 GitLab 실행기로 지원되는 이미지 계산

CodeBuild 호스팅 GitLab 실행기 정보

다음은 CodeBuild 호스팅 GitLab 실행기에 대한 몇 가지 일반적인 질문입니다.

CodeBuild 호스팅 GitLab 실행기에 지원되는 소스 유형은 무엇입니까?

CodeBuild 호스팅 GitLab 실행기는 `GITLAB` 및 `GITLAB_SELF_MANAGED` 소스 유형에 대해 지원됩니다.

레이블에 이미지 및 인스턴스 재정의를 언제 포함해야 합니까?

각 GitLab CI/CD 파이프라인 작업에 대해 서로 다른 빌드 환경을 지정하기 위해 레이블에 이미지 및 인스턴스 재정의를 포함할 수 있습니다. 이는 여러 CodeBuild 프로젝트 또는 웹후크를 생성할 필요 없이 수행할 수 있습니다.

이 기능에 CloudFormation 를 사용할 수 있습니까?

예, 프로젝트 웹후크에서 GitLab 워크플로 작업 이벤트 필터를 지정하는 필터 그룹을 CloudFormation 템플릿에 포함할 수 있습니다.

```
Triggers:
  Webhook: true
  FilterGroups:
    - - Type: EVENT
        Pattern: WORKFLOW_JOB_QUEUED
```

자세한 내용은 GitLab 웹후크 이벤트 필터링(CloudFormation) 섹션을 참조하세요.

CloudFormation 템플릿에서 프로젝트 자격 증명을 설정하는 데 도움이 필요한 경우 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS::CodeBuild::SourceCredential을 참조하세요.

이 기능을 사용할 때 암호를 마스킹하려면 어떻게 해야 하나요?

기본적으로 로그에 인쇄된 보안 암호는 마스킹되지 않습니다. 보안 암호를 마스킹하려면 CI/CD 환경 변수 설정을 업데이트하면 됩니다.

GitLab에서 보안 암호를 마스킹하려면

1. GitLab 설정에서 CI/CD를 선택합니다.
2. 변수에서 마스킹하려는 보안 암호에 대해 편집을 선택합니다.

3. 가시성에서 변수 마스킹을 선택한 다음 변수 업데이트를 선택하여 변경 사항을 저장합니다.

단일 그룹 내의 여러 프로젝트에서 GitLab 웹후크 이벤트를 받을 수 있나요?

CodeBuild는 지정된 GitLab 그룹에서 이벤트를 수신하는 그룹 웹후크를 지원합니다. 자세한 내용은 GitLab 그룹 웹후크 섹션을 참조하세요.

자체 관리형 실행기의 도커 실행기에서 작업을 실행할 수 있나요? 예를 들어 특정 이미지에서 파이프라인 작업을 실행하여 별도의 격리된 컨테이너에서 동일한 빌드 환경을 유지하고 싶습니다.

사용자 지정 이미지로 프로젝트를 생성하거나 `.gitlab-ci.yml` 파일의 이미지를 재정의하여 특정 이미지로 CodeBuild에서 GitLab 자체 관리형 실행기를 실행할 수 있습니다.

CodeBuild의 자체 관리형 실행기는 어떤 실행기로 실행되나요?

CodeBuild의 자체 관리형 실행기는 쉘 실행기로 실행되며, 여기서 빌드는 도커 컨테이너 내에서 실행 중인 GitLab 실행기와 함께 로컬로 실행됩니다.

자체 관리형 실행기와 함께 buildspec 명령을 제공할 수 있나요?

예, 자체 관리형 실행기와 함께 buildspec 명령을 추가할 수 있습니다. GitLab 리포지토리에 buildspec.yml 파일을 제공하고 작업의 태그 섹션에서 `buildspec-override:true` 태그를 사용할 수 있습니다. 자세한 내용은 buildspec 파일 이름 및 스토리지 위치 섹션을 참조하세요.

CodeBuild 호스팅 GitLab 실행기 사용을 지원하는 리전은 무엇입니까?

CodeBuild 호스팅 GitLab 실행기는 모든 CodeBuild 리전에서 지원됩니다. CodeBuild를 사용할 수 AWS 리전 있는 위치에 대한 자세한 내용은 AWS 리전별 서비스를 참조하세요.

CodeBuild 호스팅 GitLab 실행기 사용을 지원하는 플랫폼은 무엇입니까?

CodeBuild 호스팅 GitLab 실행기는 Amazon EC2와 AWS Lambda 컴퓨팅 모두에서 지원됩니다. Amazon Linux 2, Amazon Linux 2023, Ubuntu 및 Windows Server Core 2019 플랫폼을 사용할 수 있습니다. 자세한 내용은 EC2 컴퓨팅 이미지 및 Lambda 컴퓨팅 이미지 섹션을 참조하세요.

자습서: CodeBuild 호스팅 GitLab 실행기 구성

이 자습서에서는 GitLab CI/CD 파이프라인 작업을 실행하도록 CodeBuild 프로젝트를 구성하는 방법을 보여줍니다. GitLab 또는 GitLab Self Managed with CodeBuild 사용에 대한 자세한 내용은 의 자체 관리형 GitLab 실행기 AWS CodeBuild 섹션을 참조하세요.

이 자습서를 완료하려면 먼저 다음을 수행해야 합니다.

- 를 사용하여 OAuth 앱에 연결합니다 AWS CodeConnections. OAuth 앱에 연결할 때는 CodeBuild 콘솔을 사용하여 연결해야 합니다. 자세한 지침은 CodeBuild의 GitLab 액세스 섹션을 참조하세요.
- CodeBuild를 GitLab 계정에 연결합니다. 콘솔에서 GitLab을 소스 공급자로 추가하여 연결할 수 있습니다. 지침은 CodeBuild의 GitLab 액세스 섹션을 참조하세요.

> **Note**
>
> 이는 계정에 대해 GitLab에 연결하지 않은 경우에만 수행하면 됩니다.
> 이 기능을 사용하면 CodeBuild에 GitLab OAuth 앱`manage_runner`의 `create_runner` 및 와 같은 추가 권한이 필요합니다. 특정 GitLab 계정에 AWS CodeConnections 대한가 있는 경우 권한 업데이트를 자동으로 요청하지 않습니다. 이렇게 하려면 AWS CodeConnections 콘솔로 이동하여 동일한 GitLab 계정에 대한 더미 연결을 생성하여 재인증을 트리거하여 추가 권한을 얻을 수 있습니다. 이 단계를 완료한 후 모든 기존 연결에서 러너 기능을 사용할 수 있습니다. 완료되면 더미 연결을 삭제할 수 있습니다.

1단계: 웹후크를 사용하여 CodeBuild 프로젝트 생성

이 단계에서는 웹후크를 사용하여 CodeBuild 프로젝트를 생성하고 GitLab 콘솔에서 검토합니다.

웹후크를 사용하여 CodeBuild 프로젝트를 생성하려면

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 빌드 프로젝트를 생성합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 및 빌드 실행(콘솔) 섹션을 참조하세요.

   프로젝트 유형에서 Runner 프로젝트를 선택합니다.

   - 러너에서:
     - Runner 공급자에서 GitLab을 선택합니다.
     - 자격 증명에서 다음 중 하나를 선택합니다.
       - 기본 소스 자격 증명을 선택합니다. 기본 연결은 모든 프로젝트에서 기본 GitLab 연결을 적용합니다.
       - 사용자 지정 소스 자격 증명을 선택합니다. 사용자 지정 연결은 계정의 기본 설정을 재정의하는 사용자 지정 GitLab 연결을 적용합니다.

> **Note**
>
> 공급자에 대한 연결을 아직 생성하지 않은 경우 새 GitLab 연결을 생성해야 합니다. 지침은 GitLab에 CodeBuild 연결 섹션을 참조하세요.

  - 러너 위치에서 리포지토리를 선택합니다.
  - 리포지토리 이름에서 네임스페이스와 함께 프로젝트 경로를 지정하여 GitLab의 프로젝트 이름을 선택합니다.
- 환경에서 다음과 같이 합니다.
  - 지원되는 환경 이미지와 컴퓨팅을 선택합니다. GitLab CI/CD 파이프라인 YAML의 레이블을 사용하여 이미지 및 인스턴스 설정을 재정의할 수 있습니다. 자세한 내용은 2단계: 리포지토리에 .gitlab-ci.yml 파일 생성 섹션을 참조하세요.
- Buildspec에서 다음과 같이 합니다.
  - `buildspec-override:true`가 레이블로 추가되지 않는 한 buildspec은 무시됩니다. 대신 CodeBuild는 자체 관리형 실행기를 설정하는 명령을 사용하도록 이를 재정의합니다.
-

3. 기본값으로 계속 진행한 다음 빌드 프로젝트 생성을 선택합니다.
4. `https://gitlab.com/`*`user-name`*`/`*`repository-name`*`/-/hooks`에서 GitLab 콘솔을 열어 웹후크가 생성되었고 워크플로 작업 이벤트를 전달할 수 있는지 확인합니다.

2단계: 리포지토리에 .gitlab-ci.yml 파일 생성

이 단계에서는 GitLab에서 `.gitlab-ci.yml` 파일을 생성하여 빌드 환경을 구성하고 CodeBuild에서 GitLab 자체 관리형 실행기를 사용합니다. 자세한 내용은 자체 관리형 실행기 사용을 참조하세요.

GitLab CI/CD 파이프라인 YAML 업데이트

`https://gitlab.com/`*`user-name`*`/`*`project-name`*`/-/tree/`*`branch-name`* 섹션으로 이동하여 리포지토리에서 `.gitlab-ci.yml` 파일을 생성합니다. 다음 중 하나를 수행하여 빌드 환경을 구성할 수 있습니다.

- CodeBuild 프로젝트 이름을 지정할 수 있습니다. 이 경우 빌드는 컴퓨팅, 이미지, 이미지 버전 및 인스턴스 크기에 대한 기존 프로젝트 구성을 사용합니다. 프로젝트 이름은 GitLab 작업의 AWS관련 설정을 특정 CodeBuild 프로젝트에 연결하는 데 필요합니다. YAML에 프로젝트 이름을 포함하면 CodeBuild가 올바른 프로젝트 설정으로 작업을 호출할 수 있습니다.

```
tags:
    - codebuild-<codebuild-project-name>-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
```

$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME는 빌드를 특정 파이프라인 작업 실행에 매핑하고 파이프라인 실행이 취소될 때 빌드를 중지하는 데 필요합니다.

> **Note**
>
> *<project-name>*이 CodeBuild에서 생성한 프로젝트의 이름과 일치하는지 확인합니다. 일치하지 않으면 CodeBuild는 웹후크를 처리하지 않고 GitLab CI/CD 파이프라인이 중단될 수 있습니다.

다음은 GitLab CI/CD 파이프라인 YAML의 예입니다.

```
workflow:
  name: HelloWorld
stages:          # List of stages for jobs, and their order of execution
  - build

build-job:       # This job runs in the build stage, which runs first.
  stage: build
  script:
    - echo "Hello World!"
  tags:
    - codebuild-myProject-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
```

- 태그에서 이미지 및 컴퓨팅 유형을 재정의할 수도 있습니다. 큐레이션된 이미지 목록은 섹션을 참조CodeBuild 호스팅 GitLab 실행기로 지원되는 이미지 계산하세요. 사용자 지정 이미지 사용은 섹션을 참조하세요CodeBuild 호스팅 GitLab 실행기에서 지원되는 레이블 재정의. 태그의 컴퓨팅 유형과 이미지가 프로젝트의 환경 설정을 재정의합니다. Amazon EC2 컴퓨팅 빌드의 환경 설정을 재정의하려면 다음 구문을 사용합니다.

```
tags:
    - codebuild-<codebuild-project-name>-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
    - image:<environment-type>-<image-identifier>
    - instance-size:<instance-size>
```

다음은 GitLab CI/CD 파이프라인 YAML의 예입니다.

```
stages:
  - build

build-job:
  stage: build
  script:
    - echo "Hello World!"
  tags:
    - codebuild-myProject-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
    - image:arm-3.0
    - instance-size:small
```

- 태그의 빌드에 사용되는 플릿을 재정의할 수 있습니다. 이렇게 하면 지정된 플릿을 사용하도록 프로젝트에 구성된 플릿 설정이 재정의됩니다. 자세한 내용은 예약 용량 플릿에서 빌드 실행 섹션을 참조하세요. Amazon EC2 컴퓨팅 빌드의 플릿 설정을 재정의하려면 다음 구문을 사용합니다.

  ```
  tags:
      - codebuild-<codebuild-project-name>-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
      - fleet:<fleet-name>
  ```

  빌드에 사용되는 플릿과 이미지를 모두 재정의하려면 다음 구문을 사용합니다.

  ```
  tags:
      - codebuild-<codebuild-project-name>-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
      - fleet:<fleet-name>
      - image:<environment-type>-<image-identifier>
  ```

  다음은 GitLab CI/CD 파이프라인 YAML의 예입니다.

  ```
  stages:
    - build

  build-job:
    stage: build
    script:
      - echo "Hello World!"
    tags:
      - codebuild-myProject-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
      - fleet:myFleet
      - image:arm-3.0
  ```

- 사용자 지정 이미지에서 GitLab CI/CD 파이프라인 작업을 실행하려면 CodeBuild 프로젝트에서 사용자 지정 이미지를 구성하고 이미지 재정의 레이블을 제공하지 않아도 됩니다. CodeBuild는 이미지 재정의 레이블이 제공되지 않은 경우 프로젝트에 구성된 이미지를 사용합니다.

`.gitlab-ci.yml`에 변경 사항을 커밋하면 GitLab 파이프라인이 트리거되고 `build-job`이 CodeBuild에서 빌드를 시작하는 웹후크 알림을 보냅니다.

INSTALL, PRE_BUILD 및 POST_BUILD 단계에서 buildspec 명령 실행

기본적으로 CodeBuild는 자체 관리형 GitLab 빌드를 실행할 때 빌드 사양 명령을 무시합니다. 빌드 중에 buildspec 명령을 실행하려면 `buildspec-override:true`를 접미사로 `tags`에 추가할 수 있습니다.

```
tags:
    - codebuild-<codebuild-project-name>-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
    - buildspec-override:true
```

이 명령을 사용하면 CodeBuild는 컨테이너의 기본 소스 폴더에 `gitlab-runner`라는 폴더를 생성합니다. `BUILD` 단계 중에 GitLab 실행기가 시작되면 실행기가 `gitlab-runner` 디렉터리에서 실행됩니다.

자체 관리형 GitLab 빌드에서 buildspec 재정의를 사용할 때 몇 가지 제한 사항이 있습니다.

- CodeBuild는 `BUILD` 단계에서 자체 관리형 실행기가 실행되므로 `BUILD` 단계 중에 buildspec 명령을 실행하지 않습니다.
- CodeBuild는 `DOWNLOAD_SOURCE` 단계 중에 기본 또는 보조 소스를 다운로드하지 않습니다. buildspec 파일이 구성된 경우 해당 파일만 프로젝트의 기본 소스에서 다운로드됩니다.
- `PRE_BUILD` 또는 `INSTALL` 단계에서 빌드 명령이 실패하면 CodeBuild는 자체 관리형 실행기를 시작하지 않으므로 GitLab CI/CD 파이프라인 작업을 수동으로 취소해야 합니다.
- CodeBuild는 만료 시간이 1시간인 `DOWNLOAD_SOURCE` 단계 중에 실행기 토큰을 가져옵니다. `PRE_BUILD` 또는 `INSTALL` 단계가 1시간을 초과하면 GitLab 자체 관리형 실행기가 시작되기 전에 실행기 토큰이 만료될 수 있습니다.

3단계: 결과 검토

GitLab CI/CD 파이프라인 실행이 발생할 때마다 CodeBuild는 웹후크를 통해 CI/CD 파이프라인 작업 이벤트를 수신합니다. CI/CD 파이프라인의 각 작업에 대해 CodeBuild는 임시 GitLab 실행기를 실행하

기 위한 빌드를 시작합니다. 실행기는 단일 CI/CD 파이프라인 작업을 실행할 책임이 있습니다. 작업이 완료되면 실행기와 관련 빌드 프로세스가 즉시 종료됩니다.

CI/CD 파이프라인 작업 로그를 보려면 GitLab의 리포지토리로 이동하여 빌드, 작업을 선택한 다음 로그를 검토하려는 특정 작업을 선택합니다.

CodeBuild에서 자체 관리형 실행기가 작업을 픽업할 때까지 기다리는 동안 로그에서 요청된 레이블을 검토할 수 있습니다.

GitLab 웹후크 이벤트 필터링(CloudFormation)

템플릿의 CloudFormation 다음 YAML 형식 부분은 빌드가 true로 평가될 때 빌드를 트리거하는 필터 그룹을 생성합니다. 다음 필터 그룹은 정규식 `\[CI-CodeBuild\]`와 일치하는 CI/CD 파이프라인 이름이 있는 GitLab CI/CD 파이프라인 작업 요청을 지정합니다.

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Name: MyProject
    ServiceRole: service-role
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      Type: LINUX_CONTAINER
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:5.0
    Source:
      Type: GITLAB
      Location: CODEBUILD_DEFAULT_WEBHOOK_SOURCE_LOCATION
    Triggers:
      Webhook: true
      ScopeConfiguration:
        Name: group-name
        Scope: GITLAB_GROUP
      FilterGroups:
        - - Type: EVENT
            Pattern: WORKFLOW_JOB_QUEUED
          - Type: WORKFLOW_NAME
            Pattern: \[CI-CodeBuild\]
```

CodeBuild 호스팅 GitLab 실행기에서 지원되는 레이블 재정의

GitLab CI/CD 파이프라인 YAML에서 자체 관리형 실행기 빌드를 수정하는 다양한 레이블 재정의를 제공할 수 있습니다. CodeBuild에서 인식하지 못하는 빌드는 무시되지만 웹후크 요청에 실패하지는 않습니다. 예를 들어, 다음 YAML에는 이미지, 인스턴스 크기, 플릿 및 빌드 사양에 대한 재정의가 포함됩니다.

```
workflow:
  name: HelloWorld
stages:
  - build

build-job:
  stage: build
  script:
    - echo "Hello World!"
  tags:
    - codebuild-myProject-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME
    - image:arm-3.0
    - instance-size:small
    - fleet:myFleet
    - buildspec-override:true
```

`codebuild-`*`<project-name>`*`-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME` (필수)

- 예시: `codebuild-myProject-$CI_PROJECT_ID-$CI_PIPELINE_IID-$CI_JOB_NAME`
- 모든 GitLab CI/CD 파이프라인 YAMLs에 필요합니다. *`<project name>`*은 자체 관리형 실행기 웹후크가 구성된 프로젝트의 이름과 같아야 합니다.

`image:`*`<environment-type>`*`-`*`<image-identifier>`*

- 예시: `image:arm-3.0`
- 자체 관리형 실행기 빌드를 시작할 때 사용되는 이미지 및 환경 유형을 재정의합니다. 지원되는 값에 대한 자세한 내용은 CodeBuild 호스팅 GitLab 실행기로 지원되는 이미지 계산 섹션을 참조하세요.
  - 사용자 지정 이미지와 함께 사용되는 이미지 및 환경 유형을 재정의하려면 `image:custom-`*`<environment-type>`*`-`*`<custom-image-identifier>`*
  - 예시: `image:custom-arm-public.ecr.aws/codebuild/amazonlinux-aarch64-standard:3.0`

> **Note**
>
> 사용자 지정 이미지가 프라이빗 레지스트리에 있는 경우 섹션을 참조하세요자체 호스팅 러너에 대한 프라이빗 레지스트리 자격 증명 구성.

`instance-size:<instance-size>`

- 예시: `instance-size:small`
- 자체 관리형 실행기 빌드를 시작할 때 사용되는 인스턴스 유형을 재정의합니다. 지원되는 값에 대한 자세한 내용은 CodeBuild 호스팅 GitLab 실행기로 지원되는 이미지 계산 섹션을 참조하세요.

`fleet:<fleet-name>`

- 예시: `fleet:myFleet`
- 지정된 플릿을 사용하도록 프로젝트에 구성된 플릿 설정을 재정의합니다. 자세한 내용은 예약 용량 플릿에서 빌드 실행 섹션을 참조하세요.

`buildspec-override:<boolean>`

- 예시: `buildspec-override:true`
- `true`로 설정된 경우 빌드가 `INSTALL`, `PRE_BUILD` 및 `POST_BUILD` 단계에서 buildspec 명령을 실행하도록 허용합니다.

CodeBuild 호스팅 GitLab 실행기로 지원되는 이미지 계산

자습서: CodeBuild 호스팅 GitLab 실행기 구성에서 구성한 레이블에서 처음 세 열의 값을 사용하여 Amazon EC2 환경 설정을 재정의할 수 있습니다. CodeBuild는 다음과 같은 Amazon EC2 컴퓨팅 이미지를 제공합니다. 해당 내용은 다음을 참조하세요.

환경 유형	이미지 식별자	인스턴스 크기	플랫폼	이미지	정의

`linux`	`4.0`	`small`
		
		`medium`
			Amazon Linux 2	`aws/codebuild/amaz`	al/standard/4.0

환경 유형	이미지 식별자	인스턴스 크기	플랫폼	이미지	정의
		large xlarge 2xlarge		onlinux-x86_64-standard:4.0	
linux	5.0	gpu_small gpu_large	Amazon Linux 2023	aws/codebuild/amazonlinux-x86_64-standard:5.0	al/standard/5.0
linux-ec2	latest	small medium large	Amazon Linux 2023	aws/codebuild/ami/amazonlinux-x86_64-base:latest	없음
arm	2.0	small medium large xlarge	Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-standard:2.0	al/aarch64/standard/2.0
arm	3.0	2xlarge	Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-standard:3.0	al/aarch64/standard/3.0

환경 유형	이미지 식별자	인스턴스 크기	플랫폼	이미지	정의
arm-ec2	latest	small medium large	Amazon Linux 2023	aws/codebuild/ami/amazonlinux-arm-base:latest	없음
ubuntu	5.0	small medium large xlarge 2xlarge gpu_small gpu_large	Ubuntu 20.04	aws/codebuild/standard:5.0	ubuntu/standard/5.0
ubuntu	6.0		Ubuntu 22.04	aws/codebuild/standard:6.0	ubuntu/standard/6.0
ubuntu	7.0		Ubuntu 22.04	aws/codebuild/standard:7.0	ubuntu/standard/7.0
windows	1.0	medium large	Windows Server Core 2019	aws/codebuild/windows-base:2019-1.0	해당 사항 없음
			Windows Server Core 2022	aws/codebuild/windows-base:2022-1.0	해당 사항 없음
windows	2.0		Windows Server Core 2019	aws/codebuild/windows-base:2019-2.0	해당 사항 없음

환경 유형	이미지 식별자	인스턴스 크기	플랫폼	이미지	정의
windows	3.0		Windows Server Core 2019	aws/codebuild/windows-base:2019-3.0	해당 사항 없음
windows-ec2	2022	medium large	Windows Server Core 2022	aws/codebuild/ami/windows-base:2022	없음

또한 다음 값을 사용하여 Lambda 환경 설정을 재정의할 수 있습니다. CodeBuild Lambda 컴퓨팅에 대한 자세한 내용은 AWS Lambda 컴퓨팅 기반 빌드 실행 섹션을 참조하세요. CodeBuild는 다음 Lambda 컴퓨팅 이미지를 지원합니다.

환경 유형	실행 시간 버전	인스턴스 크기			
linux-lambda	dotnet6	1GB			
arm-lambda	go1.21	2GB			
	corretto11	4GB			
		8GB			
	corretto17	10GB			
	corretto21				
	nodejs18				
	nodejs20				

환경 유형	실행 시간 버전	인스턴스 크기			

	`python3.11`
	
	`python3.12`
	
	`ruby3.2`				

자세한 내용은 빌드 환경 컴퓨팅 모드 및 유형 및 CodeBuild가 제공하는 도커 이미지 섹션을 참조하세요.

의 자체 관리형 Buildkite 실행기 AWS CodeBuild

CodeBuild 컨테이너에서 자체 호스팅된 Buildkite 러너를 설정하여 Buildkite 작업을 처리하도록 프로젝트를 구성할 수 있습니다. 이는 CodeBuild 프로젝트를 사용하여 웹후크를 설정하고 CodeBuild 시스템에서 호스팅되는 자체 호스팅 러너를 사용하도록 Buildkite 파이프라인 YAML 단계를 업데이트하여 수행할 수 있습니다.

Buildkite 작업을 실행하도록 CodeBuild 프로젝트를 구성하는 상위 단계는 다음과 같습니다.

- CodeBuild 콘솔로 이동하여 Buildkite 실행기 프로젝트 실행기 유형 구성으로 CodeBuild 프로젝트를 생성합니다.
- Buildkite 조직에 `job.scheduled` 웹후크를 추가합니다.
- Buildkite에서 Buildkite 파이프라인 YAML 단계를 업데이트하여 빌드 환경을 구성합니다.

자세한 절차는 자습서: CodeBuild 호스팅 Buildkite 실행기 구성 섹션을 참조하세요. 이 기능을 사용하면 Buildkite 작업이 네이티브 통합을 통해 IAM AWS, AWS Secrets Manager AWS CloudTrail및 Amazon VPC와 같은 기능을 통해 보안 및 편의를 제공할 수 있습니다. ARM 기반 인스턴스를 포함하여 최신 인스턴스 유형에 액세스할 수 있습니다.

CodeBuild 호스팅 Buildkite 실행기 정보

다음은 CodeBuild 호스팅 Buildkite 실행기에 대한 몇 가지 일반적인 질문입니다.

레이블에 이미지 및 인스턴스 재정의를 언제 포함해야 합니까?

레이블에 이미지 및 인스턴스 재정의를 포함하여 각 Buildkite 작업에 대해 서로 다른 빌드 환경을 지정할 수 있습니다. 이는 여러 CodeBuild 프로젝트 또는 웹후크를 생성할 필요 없이 수행할 수 있습니다. 예를 들어, 이는 Buildkite 작업에 매트릭스를 사용해야 할 때 유용합니다.

```
agents:
  queue: "myQueue"
steps:
  - command: "echo \"Hello World\""
    agents:
      project: "codebuild-myProject"
      image: "{{matrix.os}}"
      instance-size: "{{matrix.size}}"
    matrix:
      setup:
        os:
          - "arm-3.0"
          - "al2-5.0"
        size:
          - "small"
          - "large"
```

CodeBuild가 Buildkite 내에서 웹후크를 자동으로 생성할 수 있나요?

현재 Buildkite에서는 콘솔을 사용하여 모든 웹후크를 수동으로 생성해야 합니다. 의 자습서에 따라 Buildkite 콘솔에서 수동으로 Buildkite 웹후크를 자습서: CodeBuild 호스팅 Buildkite 실행기 구성 생성할 수 있습니다.

CloudFormation 를 사용하여 Buildkite 웹후크를 생성할 수 있습니까?

CloudFormation Buildkite는 콘솔을 사용하여 수동으로 웹후크를 생성해야 하므로는 현재 Buildkite 러너 웹후크에 대해 지원되지 않습니다.

CodeBuild 호스팅 Buildkite 실행기 사용을 지원하는 리전은 어디입니까?

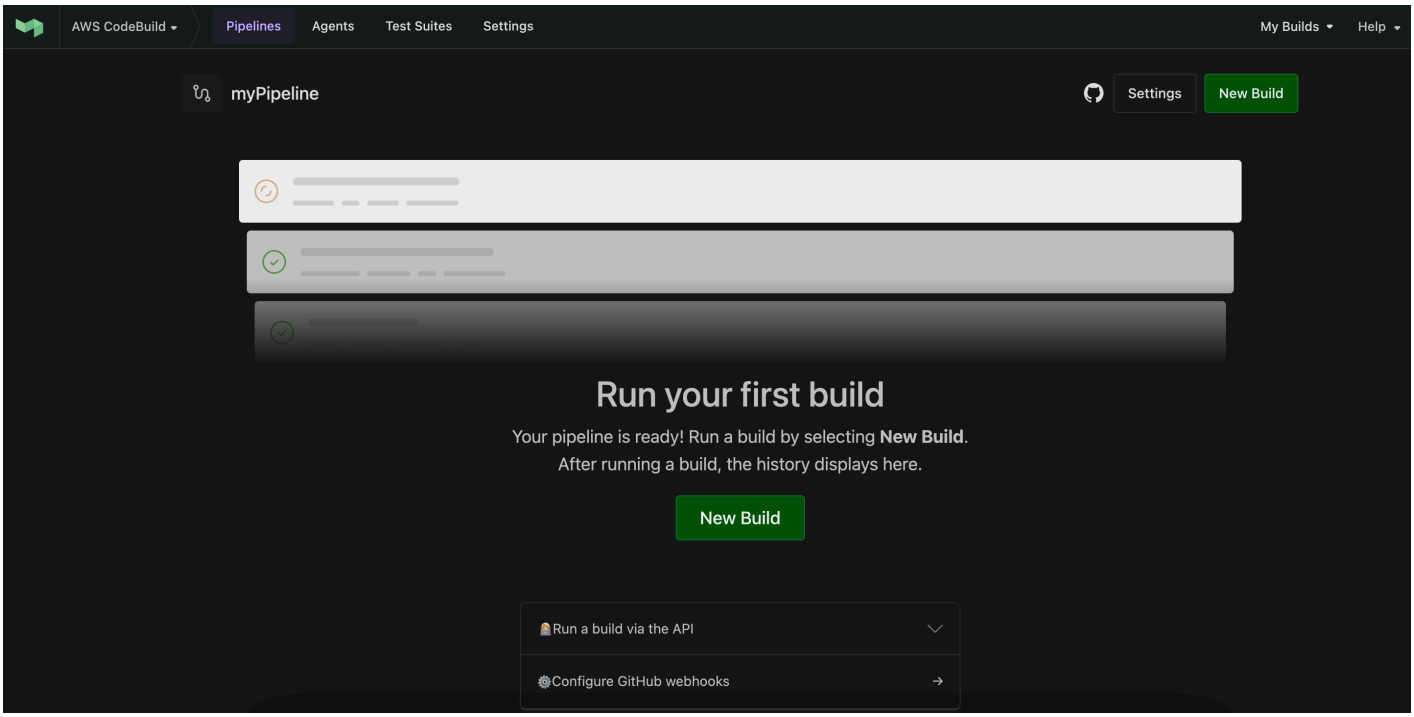
CodeBuild 호스팅 Buildkite 실행기는 모든 CodeBuild 리전에서 지원됩니다. CodeBuild를 사용할 수 있는 AWS 리전에 대한 자세한 내용은 AWS 리전별 서비스를 참조하세요.

자습서: CodeBuild 호스팅 Buildkite 실행기 구성

이 자습서에서는 Buildkite 작업을 실행하도록 CodeBuild 프로젝트를 구성하는 방법을 보여줍니다. CodeBuild에서 Buildkite를 사용하는 방법에 대한 자세한 내용은 섹션을 참조하세요의 자체 관리형 Buildkite 실행기 AWS CodeBuild.

이 자습서를 완료하려면 먼저 다음을 수행해야 합니다.

- Buildkite 조직에 액세스할 수 있습니다. Buildkite 계정 및 조직 설정에 대한 자세한 내용은이 시작하기 자습서를 참조하세요.
- 자체 호스팅 러너를 사용하도록 구성된 Buildkite 파이프라인, 클러스터 및 대기열을 생성합니다. 이러한 리소스 설정에 대한 자세한 내용은 Buildkite 파이프라인 설정 자습서를 참조하세요.

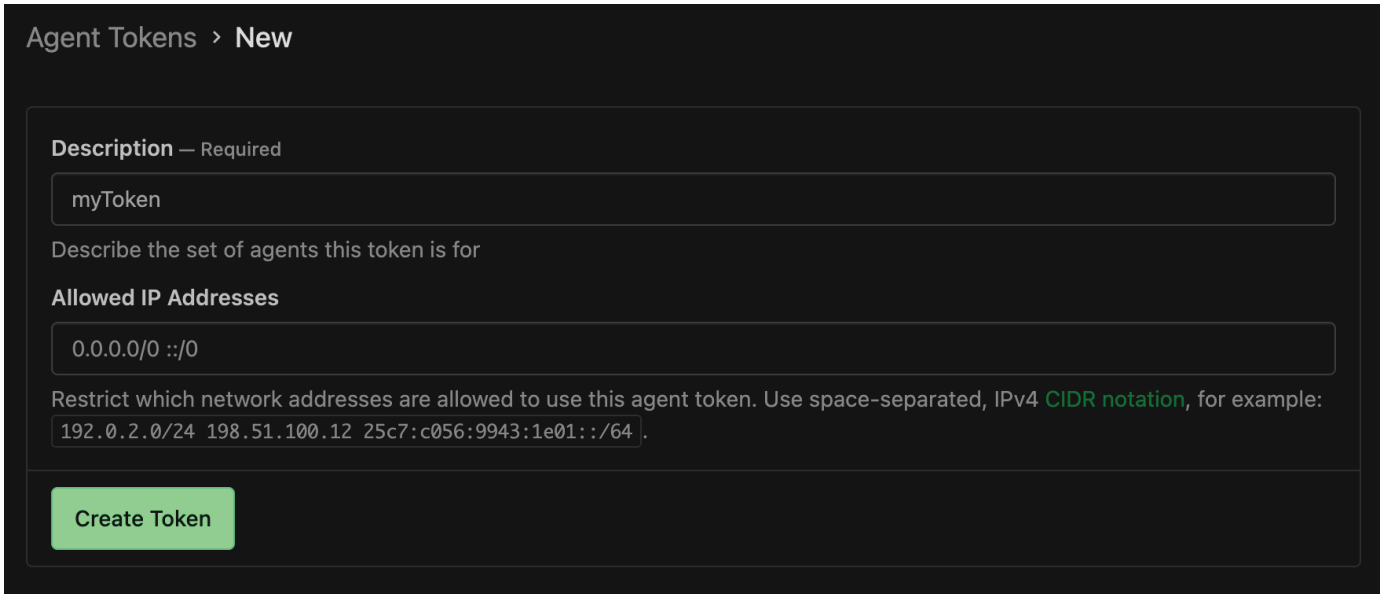


1단계: Buildkite 에이전트 토큰 생성

이 단계에서는 CodeBuild 자체 호스팅 러너를 인증하는 데 사용할 에이전트 토큰을 Buildkite 내에서 생성합니다. 이 리소스에 대한 자세한 내용은 Buildkite 에이전트 토큰을 참조하세요.

Buildkite 에이전트 토큰을 생성하려면

1. Buildkite 클러스터에서 에이전트 토큰을 선택한 다음 새 토큰을 선택합니다.
2. 토큰에 설명을 추가하고 토큰 생성을 클릭합니다.
3. 에이전트 토큰 값은 나중에 CodeBuild 프로젝트 설정 중에 사용되므로 저장합니다.



2단계: Webhook를 사용하여 CodeBuild 프로젝트 생성

웹후크를 사용하여 CodeBuild 프로젝트를 생성하려면

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 자체 호스팅 빌드 프로젝트를 생성합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 및 빌드 실행(콘솔) 섹션을 참조하세요.
   - 프로젝트 구성에서 Runner 프로젝트를 선택합니다. 러너에서:
     - 러너 공급자에서 Buildkite를 선택합니다.
     - Buildkite 에이전트 토큰에서 보안 암호 생성 페이지를 사용하여 새 에이전트 토큰 생성을 선택합니다. 위에서 생성한 Buildkite 에이전트 토큰 AWS Secrets Manager 과 동일한 보안 암호 값을 사용하여에서 새 보안 암호를 생성하라는 메시지가 표시됩니다.
     - (선택 사항) 작업에 CodeBuild 관리형 자격 증명을 사용하려면 Buildkite 소스 자격 증명 옵션에서 작업의 소스 리포지토리 공급자를 선택하고 자격 증명이 계정에 대해 구성되어 있는지 확인합니다. 또한, Buildkite 파이프라인이 HTTPS를 사용한 체크아웃을 사용하는지 확인합니다.

> **Note**
>
> Buildkite는 작업의 소스를 가져오려면 빌드 환경 내에 소스 자격 증명이 필요합니다. 사용 가능한 소스 자격 증명 옵션은 섹션을 참조프라이빗 리포지토리에 Buildkite 인증하세요.

- (선택 사항) 환경에서:
  - 지원되는 환경 이미지와 컴퓨팅을 선택합니다.

    Buildkite YAML 단계에서 레이블을 사용하여 이미지 및 인스턴스 설정을 재정의할 수 있습니다. 자세한 내용은 4단계: Buildkite 파이프라인 단계 업데이트 섹션을 참조하세요.
- (선택 사항) Buildspec에서:
  - 가 레이블로 추가되지 않는 한 buildspec`buildspec-override: "true"`은 기본적으로 무시됩니다. 대신 CodeBuild는 자체 호스팅 러너를 설정하는 명령을 사용하도록 재정의합니다.

> **Note**
>
> CodeBuild는 Buildkite 자체 호스팅 러너 빌드에 대한 buildspec 파일을 지원하지 않습니다. 인라인 buildspecs의 경우 CodeBuild 관리형 소스 자격 증명을 구성한 경우 buildspec에서 git-credential-helper를 활성화해야 합니다.

3. 기본값으로 계속 진행한 다음 빌드 프로젝트 생성을 선택합니다.
4. 웹후크 생성 팝업에서 페이로드 URL 및 보안 암호 값을 저장합니다. 팝업의 지침에 따라 새 Buildkite 조직 웹후크를 생성하거나 다음 섹션으로 계속 진행합니다.

3단계: Buildkite 내에서 CodeBuild 웹후크 생성

이 단계에서는 CodeBuild 웹후크의 페이로드 URL 및 보안 암호 값을 사용하여 Buildkite 내에 새 웹후크를 생성합니다. 이 웹후크는 유효한 Buildkite 작업이 시작될 때 CodeBuild 내에서 빌드를 트리거하는 데 사용됩니다.

Buildkite에서 새 웹후크를 생성하려면

1. Buildkite 조직의 설정 페이지를 엽니다.
2. 통합에서 알림 서비스를 선택합니다.

3. Webhook 상자 옆에 있는 추가를 선택합니다. Webhook 알림 추가 페이지에서 다음 구성을 사용합니다.

   a. Webhook URL에서 저장된 페이로드 URL 값을 추가합니다.

   b. 토큰에서 토큰을 X-Buildkite-Token으로 전송이 선택되어 있는지 확인합니다. 토큰 필드에 웹후크 보안 암호 값을 추가합니다.

   c. 에서 토큰을 X-Buildkite-Token으로 전송이 선택되어 있는지 확인합니다. 토큰 필드에 웹후크 보안 암호 값을 추가합니다.

   d. 이벤트에서 `job.scheduled` 웹후크 이벤트를 선택합니다.

   e. (선택 사항) 파이프라인에서 선택적으로 특정 파이프라인에 대한 빌드만 트리거하도록 선택할 수 있습니다.

4. Webhook 알림 추가를 선택합니다.

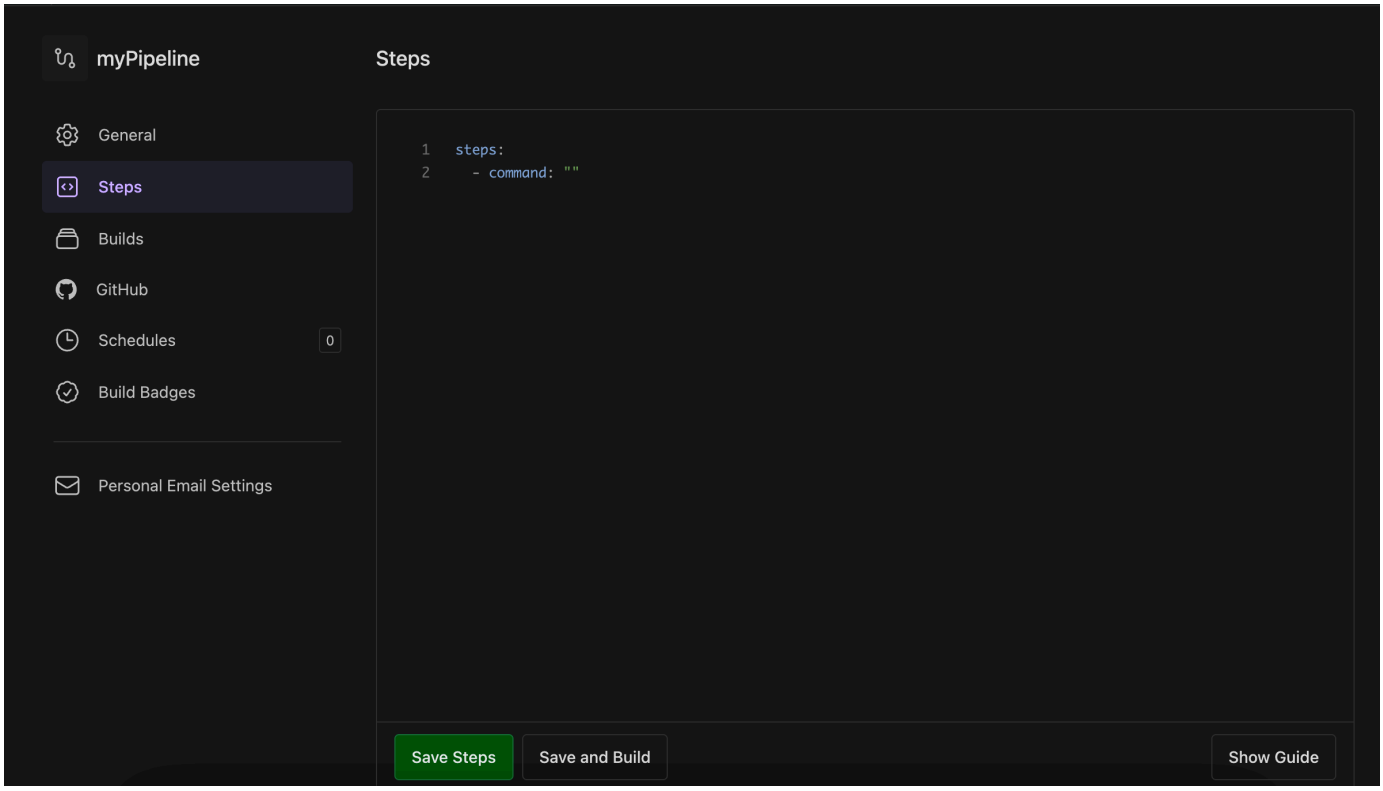
4단계: Buildkite 파이프라인 단계 업데이트

이 단계에서는 Buildkite 파이프라인의 단계를 업데이트하여 필요한 레이블과 선택적 재정의를 추가합니다. 지원되는 레이블 재정의의 전체 목록은 섹션을 참조하세요CodeBuild 호스팅 Buildkite 실행기에서 지원되는 레이블 재정의.

파이프라인 단계 업데이트

1. Buildkite 파이프라인을 선택하고 설정을 선택한 다음 단계를 선택하여 Buildkite 파이프라인 단계 페이지를 엽니다.

   아직 선택하지 않았다면 YAML 단계로 변환을 선택합니다.



2. 최소한 CodeBuild 파이프라인의 이름을 참조하는 Buildkite 에이전트 태그를 지정해야 합니다. 프로젝트 이름은 Buildkite 작업의 AWS관련 설정을 특정 CodeBuild 프로젝트에 연결하는 데 필요합니다. YAML에 프로젝트 이름을 포함하면 CodeBuild가 올바른 프로젝트 설정으로 작업을 호출할 수 있습니다.

```
agents:
  project: "codebuild-<project name>"
```

다음은 프로젝트 레이블 태그만 있는 Buildkite 파이프라인 단계의 예입니다.

```
agents:
  project: "codebuild-myProject"
steps:
  - command: "echo \"Hello World\""
```

레이블에서 이미지 및 컴퓨팅 유형을 재정의할 수도 있습니다. 사용 가능한 이미지 목록은 CodeBuild 호스팅 Buildkite 실행기에서 지원되는 컴퓨팅 이미지 섹션을 참조하세요. 레이블의 컴퓨팅 유형 및 이미지가 프로젝트의 환경 설정을 재정의합니다. CodeBuild EC2 또는 Lambda 컴퓨팅 빌드의 환경 설정을 재정의하려면 다음 구문을 사용합니다.

```
agents:
  project: "codebuild-<project name>"
  image: "<environment-type>-<image-identifier>"
  instance-size: "<instance-size>"
```

다음은 이미지 및 인스턴스 크기 재정의가 있는 Buildkite 파이프라인 단계의 예입니다.

```
agents:
  project: "codebuild-myProject"
  image: "arm-3.0"
  instance-size: "small"
steps:
  - command: "echo \"Hello World\""
```

레이블에서 빌드에 사용되는 플릿을 재정의할 수 있습니다. 이렇게 하면 지정된 플릿을 사용하도록 프로젝트에 구성된 플릿 설정이 재정의됩니다. 자세한 내용은 예약된 용량 플릿에서 빌드 실행을 참조하세요.

Amazon EC2 컴퓨팅 빌드의 플릿 설정을 재정의하려면 다음 구문을 사용합니다.

```
agents:
  project: "codebuild-<project name>"
  fleet: "<fleet-name>"
```

빌드에 사용되는 플릿과 이미지를 모두 재정의하려면 다음 구문을 사용합니다.

```
agents:
  project: "codebuild-<project name>"
  fleet: "<fleet-name>"
  image: "<environment-type>-<image-identifier>"
```

다음은 플릿 및 이미지 재정의가 있는 Buildkite 파이프라인 단계의 예입니다.

```
agents:
  project: "codebuild-myProject"
  fleet: "myFleet"
  image: "arm-3.0"
steps:
  - command: "echo \"Hello World\""
```

3. 자체 호스팅 Buildkite 실행기 빌드 중에 인라인 buildspec 명령을 실행하도록 선택할 수 있습니다(INSTALL, PRE_BUILD 및 POST_BUILD 단계에 대해 buildspec 명령 실행자세한 내용은 참조). Buildkite 자체 호스팅 러너 빌드 중에 CodeBuild 빌드가 buildspec 명령을 실행하도록 지정하려면 다음 구문을 사용합니다.

```
agents:
  project: "codebuild-<project name>"
  buildspec-override: "true"
```

다음은 buildspec 재정의가 있는 Buildkite 파이프라인의 예입니다.

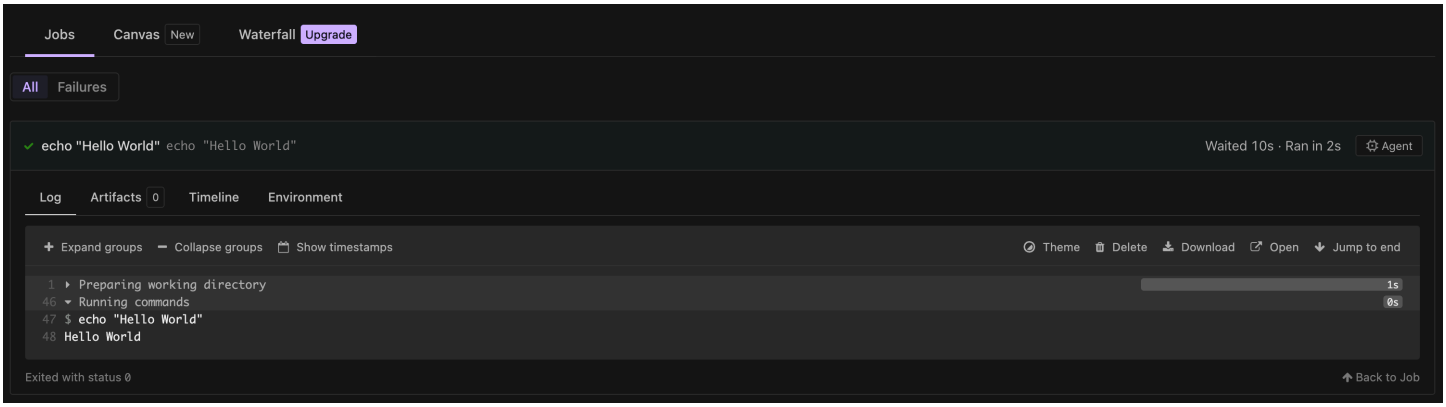
```
agents:
  project: "codebuild-myProject"
  buildspec-override: "true"
steps:
  - command: "echo \"Hello World\""
```

4. 선택적으로 CodeBuild가 지원하는 레이블 이외의 레이블을 제공할 수 있습니다. 이러한 레이블은 빌드의 속성을 재정의할 목적으로 무시되지만 웹후크 요청에 실패하지는 않습니다. 예를 들어 레이블로 `myLabel: "testLabel"`을 추가해도 레이블은 빌드 실행을 방해하지 않습니다.

5단계: 결과 검토

파이프라인에서 Buildkite 작업이 시작될 때마다 CodeBuild는 Buildkite `job.scheduled` 웹후크를 통해 웹후크 이벤트를 수신합니다. Buildkite 빌드의 각 작업에 대해 CodeBuild는 임시 Buildkite 실행기를 실행하기 위한 빌드를 시작합니다. 실행기는 단일 Buildkite 작업을 실행할 책임이 있습니다. 작업이 완료되면 실행기와 관련 빌드 프로세스가 즉시 종료됩니다.

워크플로 작업 로그를 보려면 Buildkite 파이프라인으로 이동하여 최신 빌드를 선택합니다(새 빌드를 선택하여 새 빌드를 트리거할 수 있음). 각 작업에 연결된 CodeBuild 빌드가 시작되고 작업을 픽업하면 Buildkite 콘솔 내에 해당 작업에 대한 로그가 표시되어야 합니다.



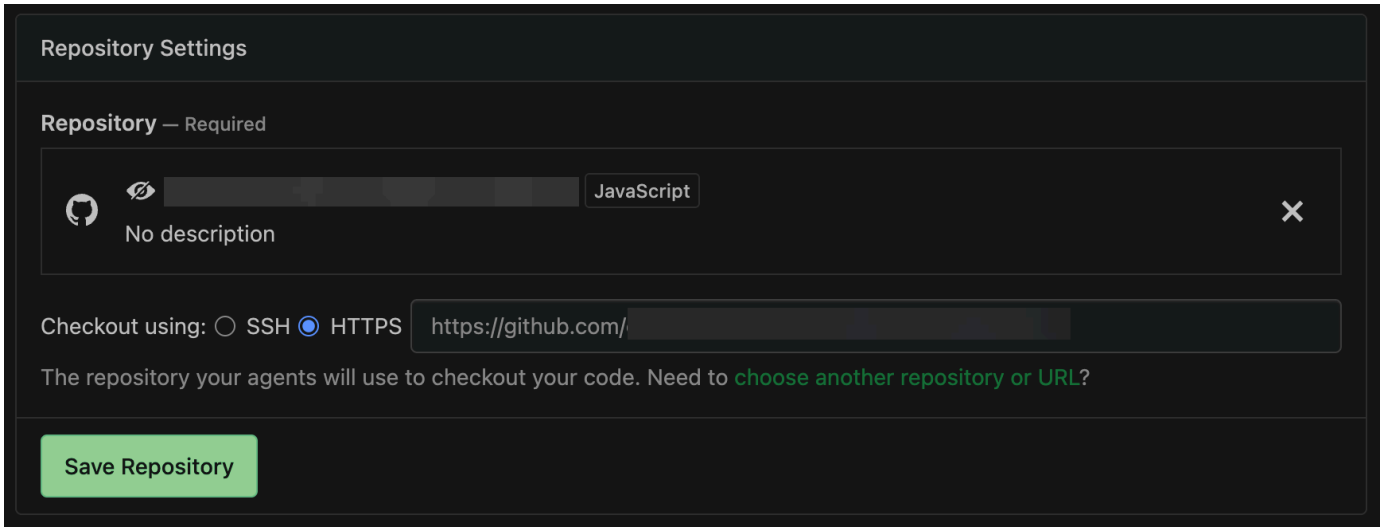
프라이빗 리포지토리에 Buildkite 인증

Buildkite 파이프라인 내에 프라이빗 리포지토리가 구성된 경우 Buildkite는 프라이빗 리포지토리에서 가져오기 위해 자체 호스팅된 러너에 자격 증명을 벤딩하지 않으므로 빌드 환경 내에서 리포지토리를 가져오려면 추가 권한이 필요합니다. Buildkite 자체 호스팅 러너 에이전트를 외부 프라이빗 소스 리포지토리에 인증하려면 다음 옵션 중 하나를 사용할 수 있습니다.

CodeBuild로 인증하려면

CodeBuild는 지원되는 소스 유형에 대한 관리형 자격 증명 처리를 제공합니다. CodeBuild 소스 자격 증명을 사용하여 작업의 소스 리포지토리를 가져오려면 다음 단계를 사용할 수 있습니다.

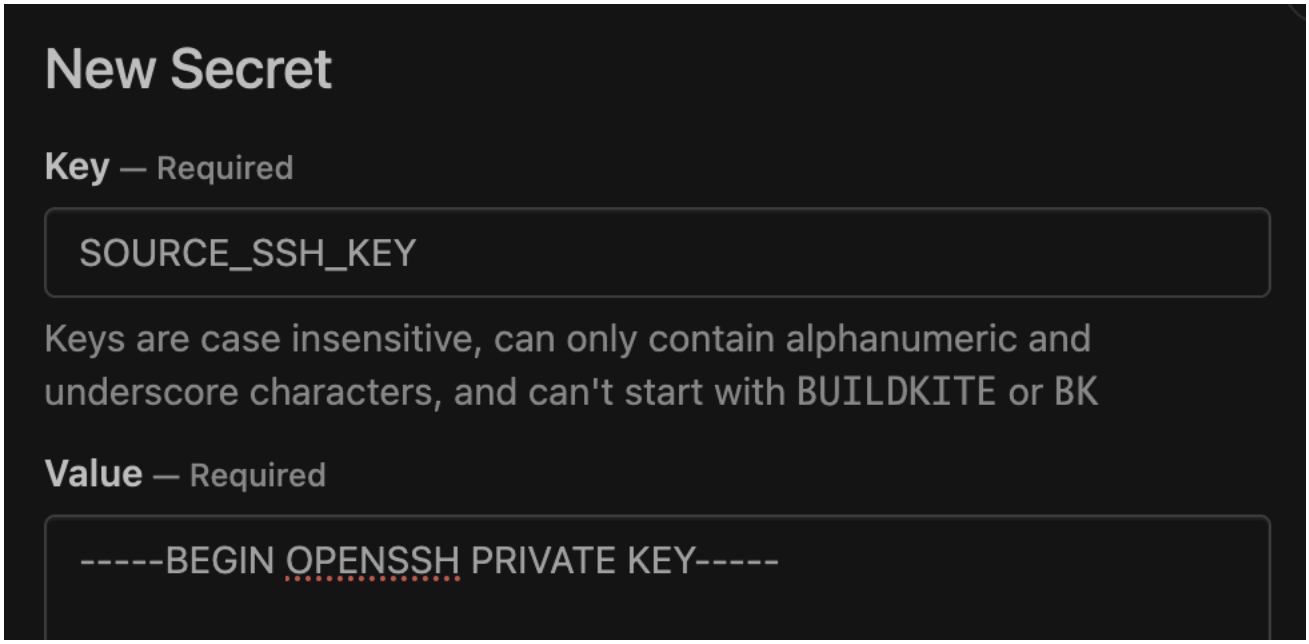
1. CodeBuild 콘솔에서의 단계를 사용하여 프로젝트 편집으로 이동하거나 새 CodeBuild 프로젝트를 생성합니다2단계: Webhook를 사용하여 CodeBuild 프로젝트 생성.
2. Buildkite 소스 자격 증명 옵션에서 작업의 소스 리포지토리 공급자를 선택합니다.
    1. 계정 수준 CodeBuild 자격 증명을 사용하려면 자격 증명이 올바르게 구성되었는지 확인합니다. 또한 프로젝트에 인라인 buildspec이 구성된 경우 git-credential-helper가 활성화되어 있는지 확인합니다.
    2. 프로젝트 수준 CodeBuild 자격 증명을 사용하려면 이 프로젝트에 대해서만 자격 증명 재정의 사용을 선택하고 프로젝트에 대한 자격 증명을 설정합니다.
3. Buildkite 파이프라인 설정에서 리포지토리 설정으로 이동합니다. HTTPS를 사용하여 소스 리포지토리 체크아웃 설정을 체크아웃으로 설정



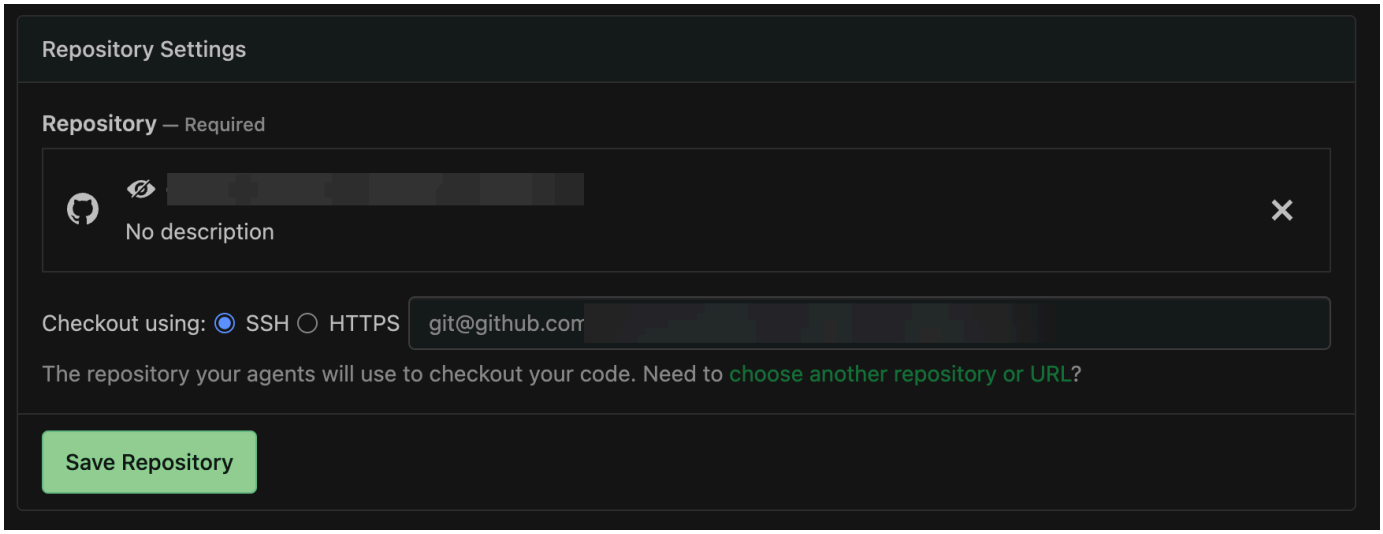
Buildkite 보안 암호로 인증하려면

Buildkite는 ssh 키를 사용하여 외부 소스 리포지토리에 자체 호스팅된 러너를 인증하는 데 사용할 수 있는 ssh-checkout 플러그인을 유지합니다. 키 값은 Buildkite 보안 암호로 저장되며 프라이빗 리포지토리를 가져오려고 할 때 Buildkite 자체 호스팅 러너 에이전트가 자동으로 가져옵니다. Buildkite 파이프라인에 대한 ssh-checkout 플러그인을 구성하려면 다음 단계를 사용할 수 있습니다.

1. 이메일 주소를 사용하여 프라이빗 및 퍼블릭 ssh 키를 생성합니다. 예: `ssh-keygen -t rsa -b 4096 -C "myEmail@address.com"`
2. 프라이빗 소스 리포지토리에 퍼블릭 키를 추가합니다. 예를 들어 이 가이드에 따라 GitHub 계정에 키를 추가할 수 있습니다.
3. Buildkite 클러스터에 새 SSH 키 보안 암호를 추가합니다. Buildkite 클러스터 내에서 보안 암호 → 새 보안 암호를 선택합니다. 키 필드에 보안 암호의 이름을 추가하고 값 필드에 프라이빗 SSH 키를 추가합니다.



4. Buildkite 파이프라인 내에서 리포지토리 설정으로 이동하여 SSH를 사용하도록 체크아웃을 설정합니다.



5. `git-ssh-checkout` 플러그인을 사용하도록 파이프라인 YAML 단계를 업데이트합니다. 예를 들어 다음 파이프라인 YAML 파일은 위의 Buildkite 보안 키와 함께 체크아웃 작업을 사용합니다.

```
agents:
  project: "codebuild-myProject"
steps:
  - command: "npm run build"
    plugins:
      - git-ssh-checkout#v0.4.1:
          ssh-secret-key-name: 'SOURCE_SSH_KEY'
```

6. CodeBuild 내에서 Buildkite 자체 호스팅 러너 작업을 실행할 때 이제 Buildkite는 프라이빗 리포지토리를 가져올 때 구성된 보안 암호 값을 자동으로 사용합니다.

러너 구성 옵션

프로젝트 구성에서 다음 환경 변수를 지정하여 자체 호스팅 러너의 설정 구성을 수정할 수 있습니다.

- `CODEBUILD_CONFIG_BUILDKITE_AGENT_TOKEN`: CodeBuild는 Buildkite 자체 호스팅 러너 에이전트를 등록하기 위해에서 AWS Secrets Manager 이 환경 변수의 값으로 구성된 보안 암호 값을 가져옵니다. 이 환경 변수는 유형이어야 하며 `SECRETS_MANAGER`값은 Secrets Manager의 보안 암호 이름이어야 합니다. 모든 Buildkite 실행기 프로젝트에는 Buildkite 에이전트 토큰 환경 변수가 필요합니다.
- `CODEBUILD_CONFIG_BUILDKITE_CREDENTIAL_DISABLE`: 기본적으로 CodeBuild는 계정 또는 프로젝트 수준 소스 자격 증명을 빌드 환경에 로드합니다. Buildkite 에이전트가 이러한 자격 증명을 사용하여 작업의 소스 리포지토리를 가져오기 때문입니다. 이 동작을 비활성화하려면 값이 로 설정된 상태에서이 환경 변수를 프로젝트에 추가하면 소스 자격 증명`true`이 빌드 환경에 로드되지 않습니다.

INSTALL, PRE_BUILD 및 POST_BUILD 단계에 대해 buildspec 명령 실행

기본적으로 CodeBuild는 자체 호스팅 Buildkite 러너 빌드를 실행할 때 모든 buildspec 명령을 무시합니다. 빌드 중에 buildspec 명령을 실행하려면

```
buildspec-override: "true"
```

는 레이블에 접미사로 추가할 수 있습니다.

```
agents:
  project: "codebuild-<project name>"
  buildspec-override: "true"
```

이 명령을 사용하면 CodeBuild는 컨테이너의 기본 소스 폴더에 `buildkite-runner`라는 폴더를 생성합니다. `BUILD` 단계 중에 Buildkite 실행기가 시작되면 실행기가 `buildkite-runner` 디렉터리에서 실행됩니다.

자체 호스팅 Buildkite 빌드에서 buildspec 재정의를 사용할 때 몇 가지 제한 사항이 있습니다.

- Buildkite 에이전트는 작업의 소스 리포지토리를 가져오려면 빌드 환경 내에 소스 자격 증명이 있어야 합니다. 인증에 CodeBuild 소스 자격 증명을 사용하는 경우 buildspec`git-credential-helper`에서를 활성화해야 합니다. 예를 들어 다음 buildspec을 사용하여 Buildkite 빌드에 `git-credential-helper` 대해를 활성화할 수 있습니다.

```
version: 0.2
env:
  git-credential-helper: yes
phases:
  pre_build:
    commands:
       - echo "Hello World"
```

- CodeBuild는 `BUILD` 단계에서 자체 호스팅된 실행기가 실행되므로 `BUILD` 단계 중에 buildspec 명령을 실행하지 않습니다.
- CodeBuild는 Buildkite 러너 빌드에 대한 buildspec 파일을 지원하지 않습니다. Buildkite 자체 호스팅 러너에는 인라인 buildspecs만 지원됩니다.
- `PRE_BUILD` 또는 `INSTALL` 단계에서 빌드 명령이 실패하면 CodeBuild는 자체 호스팅된 실행기를 시작하지 않으므로 Buildkite 작업을 수동으로 취소해야 합니다.

프로그래밍 방식으로 Buildkite 실행기 설정

프로그래밍 방식으로 Buildkite 러너 프로젝트를 구성하려면 다음 리소스를 구성해야 합니다.

프로그래밍 방식으로 Buildkite 실행기를 생성하려면

1. Buildkite 에이전트 토큰을 생성하고 토큰을 내에 일반 텍스트로 저장합니다 AWS Secrets Manager.
2. 원하는 구성으로 CodeBuild 프로젝트를 설정합니다. 다음과 같은 추가 속성을 구성해야 합니다.

   1. 이름이 `CODEBUILD_CONFIG_BUILDKITE_AGENT_TOKEN`이고, 유형이 이고`SECRETS_MANAGER`, 값이 Buildkite 클러스터와 연결된 Buildkite 에이전트 토큰과 동일한 환경 값입니다.
   2. 소스 유형이과 같음 `NO_SOURCE`
   3. 프로젝트 서비스 역할의 1단계에서 생성된 보안 암호에 액세스할 수 있는 권한

   예를 들어 다음 명령을 사용하여 CLI를 통해 유효한 Buildkite 러너 프로젝트를 생성할 수 있습니다.

```
aws codebuild create-project \
--name buildkite-runner-project \
--source "{\"type\": \"NO_SOURCE\",\"buildspec\":\"\"}" \
--environment "{\"image\":\"aws/codebuild/amazonlinux-x86_64-standard:5.0\",
\"type\":\"LINUX_CONTAINER\",\"computeType\":\"BUILD_GENERAL1_MEDIUM\",
\"environmentVariables\":[{\"name\":\"CODEBUILD_CONFIG_BUILDKITE_AGENT_TOKEN\",
\"type\":\"SECRETS_MANAGER\",\"value\":\"<buildkite-secret-name>\"}]}" \
--artifacts "{\"type\": \"NO_ARTIFACTS\"}" \
--service-role <service-role>
```

3. 2단계에서 생성한 프로젝트에 Buildkite 실행기 웹후크를 생성합니다. 웹후크를 생성할 때 다음 구성 옵션을 사용해야 합니다.

   1. build-type은와 같아야 합니다. `RUNNER_BUILDKITE_BUILD`
   2. 유형`EVENT`과 패턴이 같은 필터 `WORKFLOW_JOB_QUEUED`

   예를 들어 다음 명령을 사용하여 CLI를 통해 유효한 Buildkite 러너 웹후크를 생성할 수 있습니다.

   ```
   aws codebuild create-webhook \
   --project-name buildkite-runner-project \
   --filter-groups "[[{\"type\":\"EVENT\",\"pattern\":\"WORKFLOW_JOB_QUEUED\"}]]" \
   --build-type RUNNER_BUILDKITE_BUILD
   ```

4. `create-webhook` 호출에서 반환한 페이로드 URL 및 보안 암호 값을 저장하고 자격 증명을 사용하여 Buildkite 콘솔 내에서 웹후크를 생성합니다. 이 리소스를 설정하는 방법에 자습서: CodeBuild 호스팅 Buildkite 실행기 구성 대한 가이드는의 Buildkite 내에서 3단계: CodeBuild 웹후크 생성을 참조할 수 있습니다.

실패한 빌드 또는 중단 작업에 대한 웹후크 문제 해결

문제:

에서 설정한 웹후크자습서: CodeBuild 호스팅 Buildkite 실행기 구성가 작동하지 않거나 워크플로 작업이 Buildkite에서 중단되고 있습니다.

가능한 원인:

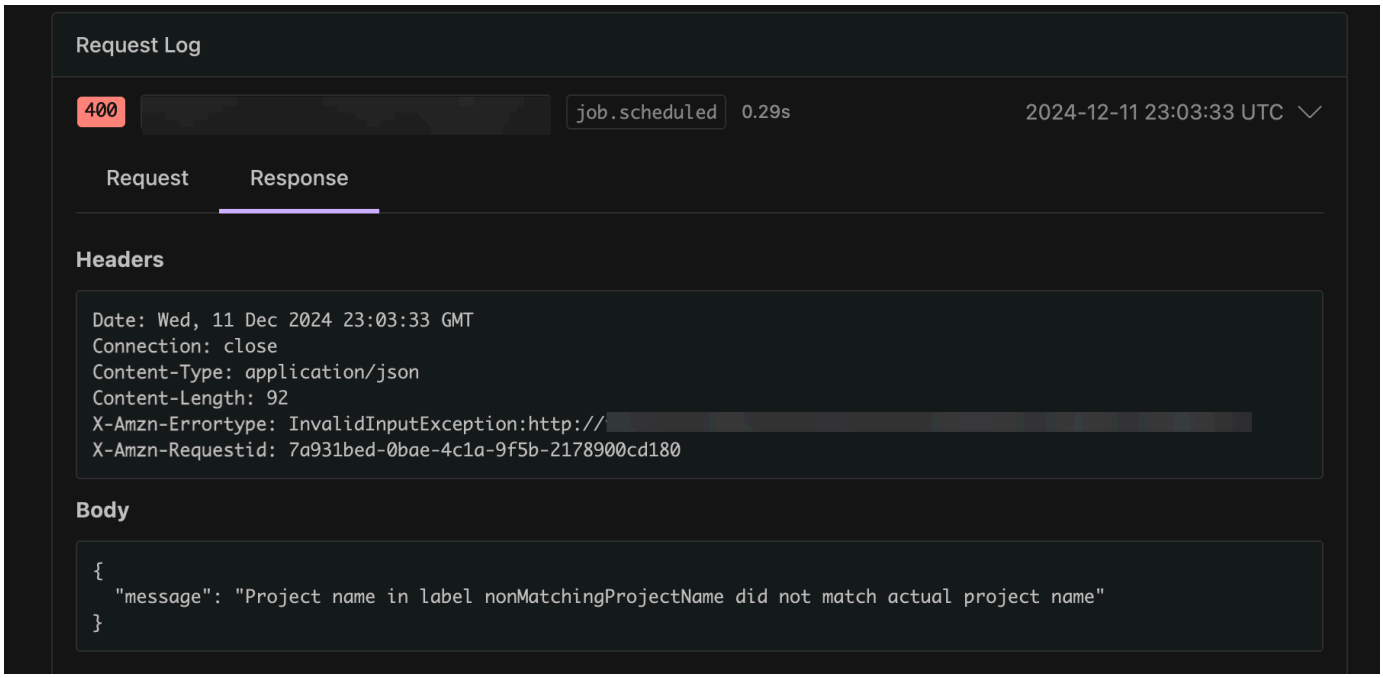
- webhook job.scheduled 이벤트가 빌드를 트리거하지 못할 수 있습니다. 응답 로그를 검토하여 응답 또는 오류 메시지를 확인합니다.

- 작업을 처리하기 위해 Buildkite 자체 호스팅 러너 에이전트를 시작하기 전에 CodeBuild 빌드가 실패합니다.

권장 솔루션:

실패한 Buildkite 웹후크 이벤트를 디버깅하려면:

1. Buildkite 조직 설정에서 알림 서비스로 이동하여 CodeBuild 웹후크를 선택한 다음 요청 로그를 찾습니다.
2. 멈춘 Buildkite 작업과 연결된 `job.scheduled` 웹후크 이벤트를 찾습니다. Webhook 페이로드 내의 작업 ID 필드를 사용하여 Webhook 이벤트를 Buildkite 작업과 연결할 수 있습니다.
3. 응답 탭을 선택하고 응답 본문을 확인합니다. 응답 상태 코드가 `200` 이고 응답 본문에 예상치 못한 메시지가 포함되어 있지 않은지 확인합니다.



Webhook 권한 문제 해결

문제:

권한 문제로 인해 Buildkite 작업이 작업의 소스 리포지토리를 체크아웃하지 못합니다.

가능한 원인:

- CodeBuild에는 작업의 소스 리포지토리를 체크아웃할 수 있는 충분한 권한이 없습니다.

- 파이프라인의 리포지토리 설정은 CodeBuild 관리형 자격 증명에 대한 SSH를 사용하여 체크아웃하도록 설정됩니다.

권장 솔루션:

- CodeBuild에 작업의 소스 리포지토리를 체크아웃하도록 구성된 충분한 권한이 있는지 확인합니다. 또한 CodeBuild 프로젝트의 서비스 역할에 구성된 소스 권한 옵션에 액세스할 수 있는 충분한 권한이 있는지 확인합니다.
- CodeBuild 관리형 소스 리포지토리 자격 증명을 사용하는 경우 Buildkite 파이프라인이 HTTPS를 사용한 체크아웃을 사용하도록 구성되어 있는지 확인합니다.

CodeBuild 호스팅 Buildkite 실행기에서 지원되는 레이블 재정의

Buildkite 파이프라인 단계에서 에이전트 태그 레이블을 지정하면 자체 호스팅된 러너 빌드를 수정하는 다양한 레이블 재정의를 제공할 수 있습니다. CodeBuild에서 인식하지 못하는 빌드는 무시되지만 웹후크 요청에 실패하지는 않습니다. 예를 들어 다음 워크플로 YAML에는 이미지, 인스턴스 크기, 플릿 및 buildspec에 대한 재정의를 포함합니다.

```
agents:
  queue: "myQueue"
steps:
  - command: "echo \"Hello World\""
    agents:
      project: "codebuild-myProject"
      image: "{{matrix.os}}"
      instance-size: "{{matrix.size}}"
      buildspec-override: "true"
    matrix:
      setup:
        os:
          - "arm-3.0"
          - "al2-5.0"
        size:
          - "small"
          - "large"
```

`project:codebuild-<project-name>` (필수)

- 예시: `project: "codebuild-myProject"`

- 모든 Buildkite 파이프라인 단계 구성에 필요합니다. *<project name>*은 자체 호스팅 러너 웹후크가 구성된 프로젝트의 이름과 같아야 합니다.

`queue: "<queue-name>"`

- 예시: `queue: "<queue-name>"`
- Buildkite 작업을 특정 대기열로 라우팅하는 데 사용됩니다. 자세한 내용은 Buildkite 에이전트 대기열 태그를 참조하세요.

`image: "<environment-type>-<image-identifier>"`

- 예시: `image: "arm-3.0"`
- 큐레이션된 이미지로 자체 호스팅 러너 빌드를 시작할 때 사용되는 이미지 및 환경 유형을 재정의합니다. 지원되는 값에 대한 자세한 내용은 CodeBuild 호스팅 Buildkite 실행기에서 지원되는 컴퓨팅 이미지 섹션을 참조하세요.
  1. 사용자 지정 이미지와 함께 사용되는 이미지 및 환경 유형을 재정의하려면를 사용합니다. `image: "custom-<environment-type>-<custom-image-identifier>"`
  2. 예제:

```
image:
      "custom-arm-public.ecr.aws/codebuild/amazonlinux-aarch64-standard:3.0"
```

> **Note**
>
> 사용자 지정 이미지가 프라이빗 레지스트리에 있는 경우 CodeBuild 프로젝트에서 적절한 레지스트리 자격 증명을 구성해야 합니다.

`instance-size: "<instance-size>"`

- 예시: `instance-size: "medium"`
- 자체 호스팅 실행기 빌드를 시작할 때 사용되는 인스턴스 유형을 재정의합니다. 지원되는 값에 대한 자세한 내용은 CodeBuild 호스팅 Buildkite 실행기에서 지원되는 컴퓨팅 이미지 섹션을 참조하세요.

`fleet: "<fleet-name>"`

- 예시: `fleet: "myFleet"`
- 지정된 플릿을 사용하도록 프로젝트에 구성된 플릿 설정을 재정의합니다. 자세한 내용은 예약된 용량 플릿에서 빌드 실행을 참조하세요.

```
buildspec-override: "<boolean>"
```

- 예시: `buildspec-override: "true"`
- `true`로 설정된 경우 빌드가 `INSTALL`, `PRE_BUILD` 및 `POST_BUILD` 단계에서 buildspec 명령을 실행하도록 허용합니다.

CodeBuild 호스팅 Buildkite 실행기에서 지원되는 컴퓨팅 이미지

의 자체 관리형 Buildkite 실행기 AWS CodeBuild에서 구성한 레이블에서 처음 세 열의 값을 사용하여 Amazon EC2 환경 설정을 재정의할 수 있습니다. CodeBuild는 다음과 같은 Amazon EC2 컴퓨팅 이미지를 제공합니다. 해당 내용은 다음을 참조하세요.

환경 유형	이미지 식별자	인스턴스 크기	플랫폼	해결된 이미지	정의

`linux`	`4.0`	`small`
		`medium`
		`large`
		`xlarge`
			Amazon Linux 2	`aws/codebuild/amazonlinux-x86_64-standard:4.0`	al/standard/4.0

`linux`	`5.0`	`2xlarge`
		`gpu_small`
		`gpu_large`
			Amazon Linux 2023	`aws/codebuild/amazonlinux-x86_64-standard:5.0`	al/standard/5.0

`linux-ec2`	`latest`	`small` `medium`	Amazon Linux 2023	`aws/codebuild/ami/amazonlinux-x86_64`	없음

환경 유형	이미지 식별자	인스턴스 크기	플랫폼	해결된 이미지	정의
		large		-base:latest	
arm	2.0	small medium large xlarge	Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-standard:2.0	al/aarch64/standard/2.0
arm	3.0	2xlarge	Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-standard:3.0	al/aarch64/standard/3.0
arm-ec2	latest	small medium large	Amazon Linux 2023	aws/codebuild/ami/amazonlinux-arm-base:latest	없음

환경 유형	이미지 식별자	인스턴스 크기	플랫폼	해결된 이미지	정의
ubuntu	5.0	small medium large xlarge 2xlarge gpu_small gpu_large	Ubuntu 20.04	aws/codebuild/standard:5.0	ubuntu/standard/5.0
ubuntu	6.0		Ubuntu 22.04	aws/codebuild/standard:6.0	ubuntu/standard/6.0
ubuntu	7.0		Ubuntu 22.04	aws/codebuild/standard:7.0	ubuntu/standard/7.0
windows	1.0	medium large	Windows Server Core 2019	aws/codebuild/windows-base:2019-1.0	해당 사항 없음
			Windows Server Core 2022	aws/codebuild/windows-base:2022-1.0	해당 사항 없음
windows	2.0		Windows Server Core 2019	aws/codebuild/windows-base:2019-2.0	해당 사항 없음
windows	3.0		Windows Server Core 2019	aws/codebuild/windows-base:2019-3.0	해당 사항 없음

환경 유형	이미지 식별자	인스턴스 크기	플랫폼	해결된 이미지	정의
windows-ec2	2022	medium large	Windows Server Core 2022	aws/codebuild/ami/windows-base:2022	없음

또한 다음 값을 사용하여 Lambda 환경 설정을 재정의할 수 있습니다. CodeBuild Lambda 컴퓨팅에 대한 자세한 내용은 AWS Lambda 컴퓨팅 기반 빌드 실행 섹션을 참조하세요. CodeBuild는 다음 Lambda 컴퓨팅 이미지를 지원합니다.

환경 유형	이미지 식별자	인스턴스 크기			
linux-lambda arm-lambda	dotnet6 go1.21 corretto11 corretto17 corretto21 nodejs18 nodejs20 python3.11 python3.12	1GB 2GB 4GB 8GB 10GB			

환경 유형	이미지 식별자	인스턴스 크기			
	`ruby3.2`				

자세한 내용은 빌드 환경 컴퓨팅 모드 및 유형 및 CodeBuild가 제공하는 도커 이미지 섹션을 참조하세요.

에서 웹후크 사용 AWS CodeBuild

AWS CodeBuild 는 GitHub, GitHub Enterprise Server, GitLab, GitLab 자체 관리형 및 Bitbucket과의 웹후크 통합을 지원합니다.

주제

- 에서 웹후크를 사용하는 모범 사례 AWS CodeBuild
- Bitbucket Webhook 이벤트
- GitHub 글로벌 및 조직 웹후크
- GitHub 수동 웹후크
- GitHub Webhook 이벤트
- GitLab 그룹 웹후크
- GitLab 수동 웹후크
- GitLab 웹후크 이벤트
- Buildkite 수동 웹후크
- 풀 요청 설명 승인

에서 웹후크를 사용하는 모범 사례 AWS CodeBuild

퍼블릭 리포지토리를 사용하여 웹후크를 설정하는 프로젝트의 경우 다음 옵션을 사용하는 것이 좋습니다.

`ACTOR_ACCOUNT_ID` 필터 설정

> 프로젝트의 웹후크 필터 그룹에 `ACTOR_ACCOUNT_ID` 필터를 추가하여 빌드를 트리거할 수 있는 사용자를 지정합니다. CodeBuild로 전달되는 모든 webhook 이벤트는 행위자의 식별자를 지정

하는 발신자 정보와 함께 제공됩니다. CodeBuild는 필터에 제공된 정규 표현식 패턴을 기준으로 webhook를 필터링합니다. 이 필터를 사용하여 빌드를 트리거할 수 있는 특정 사용자를 지정할 수 있습니다. 자세한 내용은 GitHub Webhook 이벤트 및 Bitbucket Webhook 이벤트 섹션을 참조하세요.

`FILE_PATH` 필터 설정

프로젝트의 웹후크 필터 그룹에 `FILE_PATH` 필터를 추가하여 변경 시 빌드를 트리거할 수 있는 파일을 포함하거나 제외합니다. 예를 들어 `excludeMatchedPattern` 속성과 함께 `^buildspec.yml$`과 같은 정규 표현식 패턴을 사용하여 `buildspec.yml` 파일 변경에 대한 빌드 요청을 거부할 수 있습니다. 자세한 내용은 GitHub Webhook 이벤트 및 Bitbucket Webhook 이벤트 섹션을 참조하세요.

빌드 IAM 역할의 권한 범위 좁히기

webhook로 트리거되는 빌드는 프로젝트에 지정된 IAM 서비스 역할을 사용합니다. 서비스 역할의 권한을 빌드를 실행하는 데 필요한 최소 권한 세트로 설정하는 것이 좋습니다. 예를 들어 테스트 및 배포 시나리오에서는 테스트용 프로젝트 하나와 배포용 프로젝트 하나를 생성합니다. 테스트 프로젝트는 리포지토리의 webhook 빌드를 수락하지만 리소스에 대한 쓰기 권한은 제공하지 않습니다. 배포 프로젝트는 리소스에 대한 쓰기 권한을 제공하고 webhook 필터는 신뢰할 수 있는 사용자만 빌드를 트리거할 수 있도록 구성됩니다.

인라인 또는 Amazon S3 저장 buildspec 사용

프로젝트 내에서 buildspec 인라인을 정의하거나 Amazon S3 버킷에 buildspec 파일을 저장하는 경우, buildspec 파일은 프로젝트 소유자만 볼 수 있습니다. 이렇게 하면 풀 요청이 buildspec 파일의 코드를 변경하여 원치 않는 빌드를 트리거하는 것을 방지할 수 있습니다. 자세한 내용은 CodeBuild API 참조의 ProjectSource.buildspec을 참조하세요.

Bitbucket Webhook 이벤트

Webhook 필터 그룹을 사용하여 어느 Bitbucket Webhook 이벤트가 빌드를 트리거할지 지정할 수 있습니다. 예를 들어 특정 분기가 변경된 경우에만 빌드가 트리거되도록 지정할 수 있습니다.

하나 이상의 웹후크 필터 그룹을 생성하여 어느 웹후크 이벤트가 빌드를 트리거할지 지정할 수 있습니다. 필터 그룹이 true로 평가(그룹 내 모든 필터가 true로 평가)되면 빌드가 트리거됩니다. 필터 그룹을 생성할 때 다음을 지정합니다.

이벤트

Bitbucket의 경우 다음 이벤트 중 하나 이상을 선택할 수 있습니다.

- PUSH
- PULL_REQUEST_CREATED
- PULL_REQUEST_UPDATED
- PULL_REQUEST_MERGED
- PULL_REQUEST_CLOSED

webhook의 이벤트 유형은 X-Event-Key 필드의 헤더에 있습니다. 다음 표에서는 X-Event-Key 헤더 값이 이벤트 유형에 매핑되는 방법을 보여 줍니다.

> **Note**
>
> PULL_REQUEST_MERGED 이벤트 유형을 사용하는 webhook 필터 그룹을 생성할 경우 Bitbucket webhook 설정의 merged 이벤트를 활성화해야 합니다. PULL_REQUEST_CLOSED 이벤트 유형을 사용하는 웹후크 필터 그룹을 생성할 경우 Bitbucket 웹후크 설정에서 declined 이벤트를 활성화해야 합니다.

X-Event-Key 헤더 값	이벤트 유형
repo:push	PUSH
pullrequest:created	PULL_REQUEST_CREATED
pullrequest:updated	PULL_REQUEST_UPDATED
pullrequest:fulfilled	PULL_REQUEST_MERGED
pullrequest:rejected	PULL_REQUEST_CLOSED

PULL_REQUEST_MERGED의 경우 풀 요청이 스쿼시 전략에 병합되고 풀 요청 분기가 닫히면 원래의 풀 요청 커밋은 더 이상 존재하지 않게 됩니다. 이 경우 CODEBUILD_WEBHOOK_MERGE_COMMIT 환경 변수에는 스쿼시된 병합 커밋의 식별자가 포함됩니다.

하나 이상의 선택적 필터

정규식을 사용하여 필터를 지정합니다. 이벤트가 빌드를 트리거하려면 연결된 그룹 내의 필터가 모두 true로 평가되어야 합니다.

CodeBuild는 미리 보기, 뒤로 보기, 역참조 또는 원자 그룹을 지원하지 않는 RE2 정규식 구문을 사용하여 필터 패턴을 평가합니다. 자세한 내용은 GitHub 웹 사이트의 RE2 구문 페이지를 참조하세요.

`ACTOR_ACCOUNT_ID`(콘솔의 `ACTOR_ID`)

Bitbucket 계정 ID가 정규식 패턴과 일치하면 Webhook 이벤트가 빌드를 트리거합니다. Webhook 필터 페이로드에 있는 `actor` 객체의 `account_id` 속성에 이 값이 표시됩니다.

`HEAD_REF`

헤드 참조가 정규식 패턴(예: `refs/heads/branch-name` 및 `refs/tags/tag-name`)과 일치하면 webhook 이벤트가 빌드를 트리거합니다. `HEAD_REF` 필터가 브랜치나 태그의 Git 참조 이름을 평가합니다. Webhook 페이로드의 `push` 객체에 있는 `new` 객체의 `name` 필드에 브랜치 또는 태그 이름이 표시됩니다. pull 요청 이벤트의 경우 webhook 페이로드의 `source` 객체에 있는 `branch` 객체의 `name` 필드에 브랜치 이름이 표시됩니다.

`BASE_REF`

베이스 참조가 정규식 패턴과 일치하면 webhook 이벤트가 빌드를 트리거합니다. `BASE_REF` 필터는 풀 요청 이벤트에서만 작동합니다(예: `refs/heads/branch-name`). `BASE_REF` 필터가 브랜치의 Git 참조 이름을 평가합니다. Webhook 페이로드의 `destination` 객체에 있는 `branch` 객체의 `name` 필드에 브랜치 이름이 표시됩니다.

`FILE_PATH`

변경된 파일의 경로가 정규식 패턴과 일치하면 webhook가 빌드를 트리거합니다.

> **Note**
>
> `FILE_PATH` 필터는 푸시 또는 풀 요청에서 처음 3,000개의 변경된 파일에 대해 평가됩니다. 필터가 의도한 대로 적용되도록 하려면 변경된 파일 수를이 한도 내에서 유지합니다. 풀 요청 이벤트의 경우 풀 요청 빌드 정책을 사용하여 빌드를 시작할 수 있는 풀 요청을 제어하는 것이 좋습니다.

`COMMIT_MESSAGE`

헤드 커밋 메시지가 정규식 패턴과 일치하면 webhook가 빌드를 트리거합니다.

`WORKFLOW_NAME`

워크플로 이름이 정규식 패턴과 일치하면 웹후크가 빌드를 트리거합니다.

> **Note**
>
> Bitbucket 리포지토리의 webhook 설정에서 webhook 페이로드를 찾을 수 있습니다.

주제

- Bitbucket Webhook 이벤트 필터링(콘솔)
- Bitbucket Webhook 이벤트 필터링(SDK)
- Bitbucket Webhook 이벤트 필터링(CloudFormation)

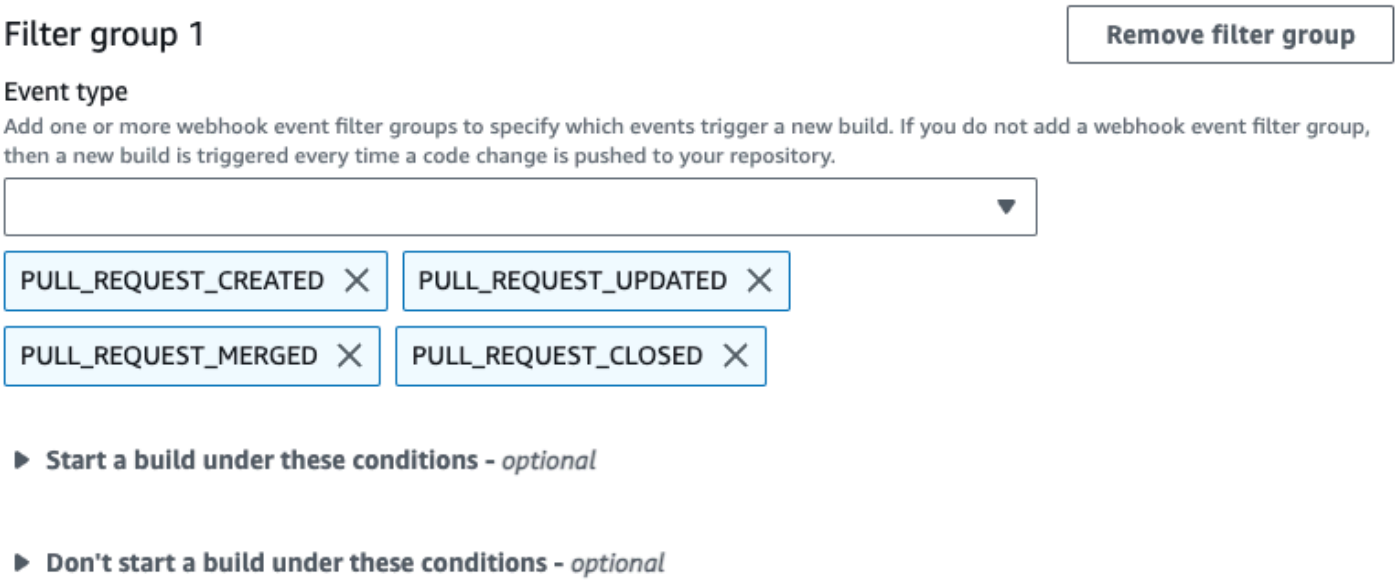
Bitbucket Webhook 이벤트 필터링(콘솔)

AWS Management Console 를 사용하여 웹후크 이벤트를 필터링하려면:

1. 프로젝트를 생성할 때 코드 변경이 이 리포지토리로 푸시될 때마다 다시 빌드를 선택합니다.
2. 이벤트 유형에서 하나 이상의 이벤트를 선택합니다.
3. 이벤트가 빌드를 트리거할 때를 필터링하려면 Start a build under these conditions(다음 조건에서 빌드를 시작)에서 하나 이상의 선택적 필터를 추가합니다.
4. 이벤트가 트리거되지 않을 때를 필터링하려면 Don't start a build under these conditions(다음 조건에서 빌드를 시작하지 않음)에서 하나 이상의 선택적 필터를 추가합니다.
5. Add filter group(필터 그룹 추가)를 선택하여 다른 필터 그룹을 추가합니다.

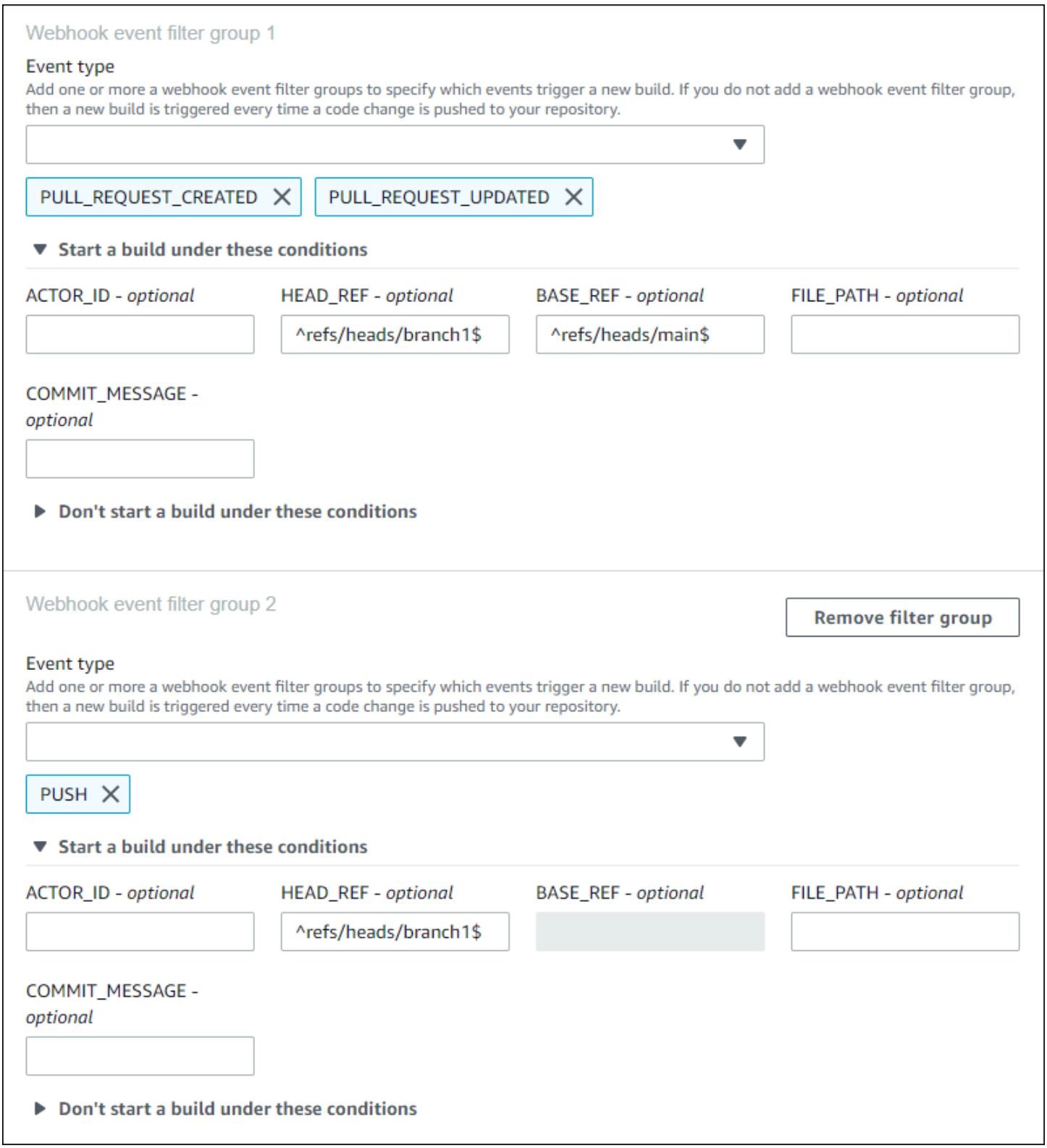
자세한 내용은 빌드 프로젝트 만들기(콘솔) 및 AWS CodeBuild API 참조의 WebHookFilter를 참조하세요.

이 예제에서는 Webhook 필터 그룹이 pull 요청에 대해서만 빌드를 트리거합니다.

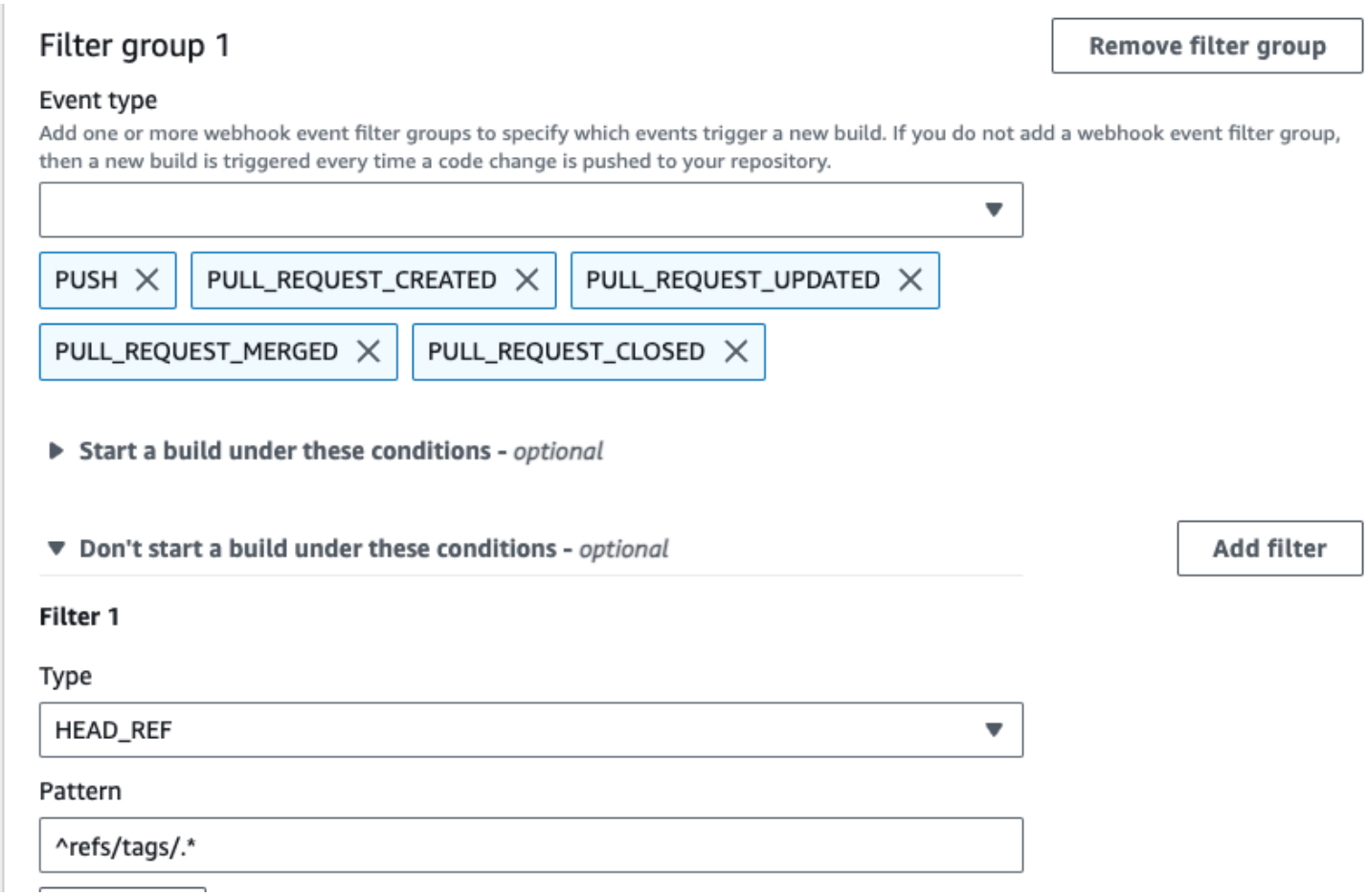


두 필터 그룹을 사용하는 예제에서는 하나 또는 두 필터 그룹이 true로 평가되면 빌드가 트리거됩니다.

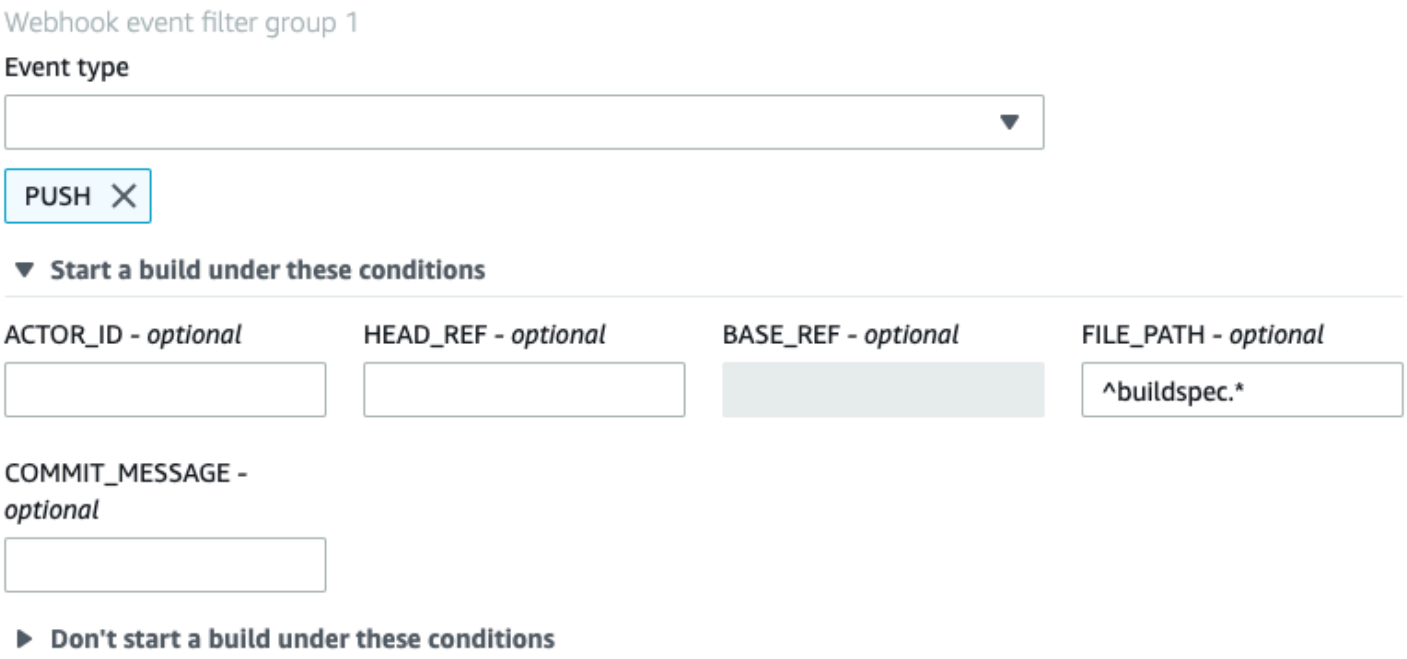
- 첫 번째 필터 그룹은 정규식 `^refs/heads/main$`와 일치하는 Git 참조 이름과 `^refs/heads/branch1!`와 일치하는 헤드 참조를 갖는 브랜치에서 생성 또는 업데이트된 pull 요청을 지정합니다.
- 두 번째 필터 그룹은 정규식 `^refs/heads/branch1$`와 일치하는 Git 참조 이름을 갖는 브랜치에서 push 요청을 지정합니다.



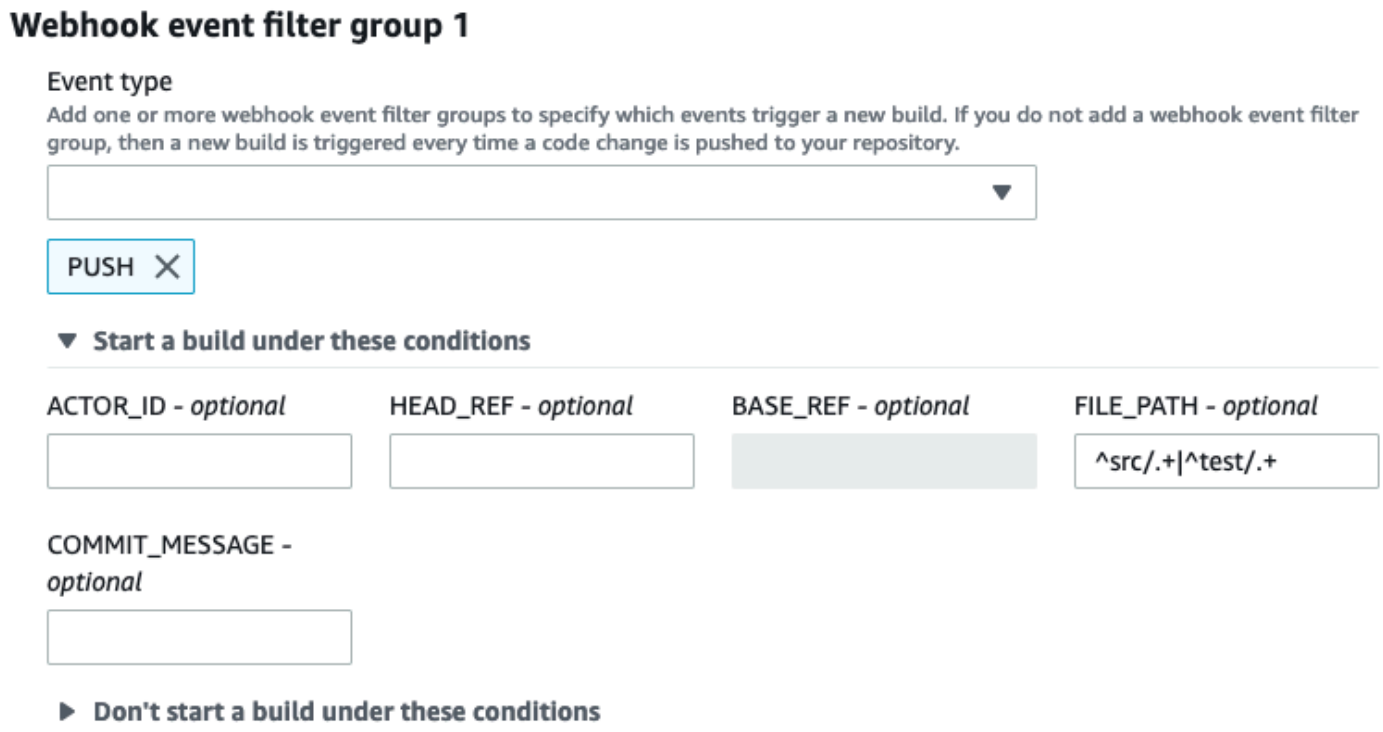
이 예제에서는 Webhook 필터 그룹이 태그 이벤트를 제외한 모든 요청에 대해 빌드를 트리거합니다.



이 예제에서는 Webhook 필터 그룹이 정규식 `^buildspec.*`와 일치하는 이름을 갖는 파일이 변경될 때만 빌드를 트리거합니다.

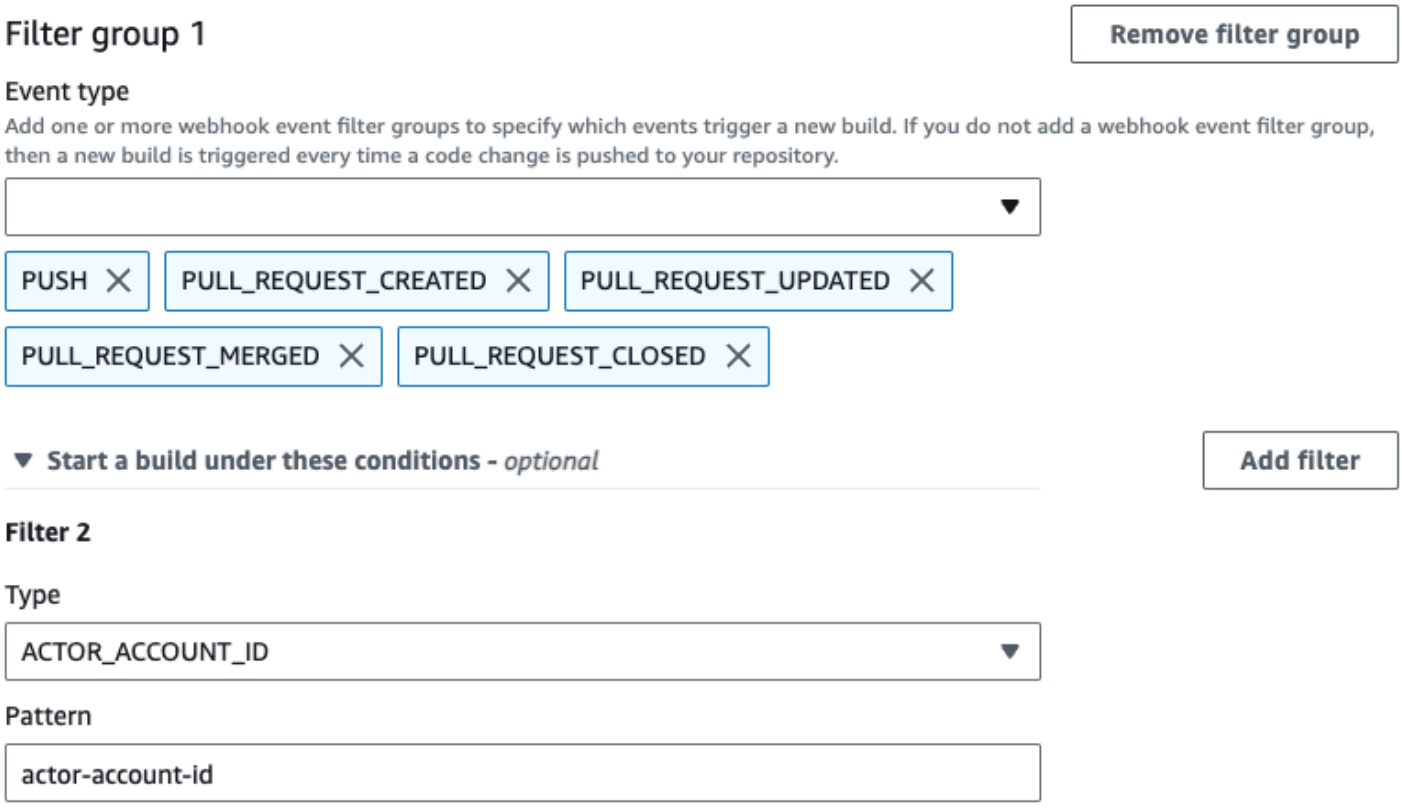


이 예제에서 Webhook 필터 그룹은 파일이 `src` 또는 `test` 폴더에서 변경된 경우에만 빌드를 트리거합니다.

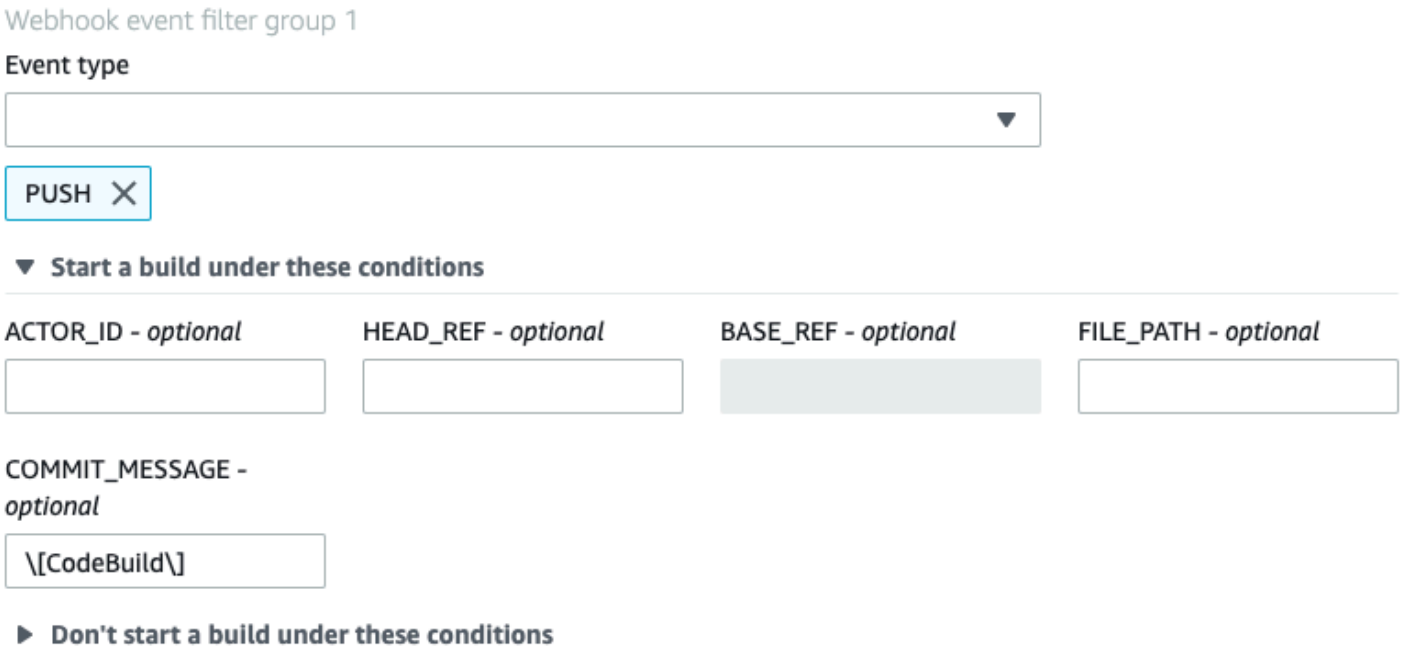


이 예제에서는 Webhook 필터 그룹이 계정 ID가 정규식 `actor-account-id`와 일치하지 않는 Bitbucket 사용자가 변경을 수행한 경우에만 빌드를 트리거합니다.

> **Note**
>
> Bitbucket 계정 ID를 확인하는 자세한 방법은 https://api.bitbucket.org/2.0/users/*user-name*을 참조하십시오. 여기서 *user-name*은 사용자의 Bitbucket 사용자 이름입니다.



이 예제에서 webhook 필터 그룹은 헤드 커밋 메시지가 정규식 `\[CodeBuild\]`와 일치할 때 푸시 이벤트에 대한 빌드를 트리거합니다.



Bitbucket Webhook 이벤트 필터링(SDK)

AWS CodeBuild SDK를 사용하여 웹후크 이벤트를 필터링하려면 `CreateWebhook` 또는 `UpdateWebhook` API 메서드의 요청 구문에서 `filterGroups` 필드를 사용합니다. 자세한 내용은 CodeBuild API 참조의 WebHookFilter를 참조하세요.

pull 요청에 대해서만 빌드를 트리거하는 Webhook 필터를 생성하려면 요청 구문에 다음을 삽입합니다.

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED, PULL_REQUEST_MERGED,
 PULL_REQUEST_CLOSED"
    }
  ]
]
```

지정된 브랜치에 대해서만 빌드를 트리거하는 Webhook 필터를 생성하려면 `pattern` 파라미터를 사용하여 브랜치 이름을 필터링하는 정규식을 지정합니다. 두 필터 그룹을 사용하는 예제에서는 하나 또는 두 필터 그룹이 true로 평가되면 빌드가 트리거됩니다.

- 첫 번째 필터 그룹은 정규식 ^refs/heads/main$와 일치하는 Git 참조 이름과 ^refs/heads/myBranch$와 일치하는 헤드 참조를 갖는 브랜치에서 생성 또는 업데이트된 pull 요청을 지정합니다.
- 두 번째 필터 그룹은 정규식 ^refs/heads/myBranch$와 일치하는 Git 참조 이름을 갖는 브랜치에서 push 요청을 지정합니다.

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED, PULL_REQUEST_CLOSED"
    },
    {
      "type": "HEAD_REF",
      "pattern": "^refs/heads/myBranch$"
    },
    {
      "type": "BASE_REF",
      "pattern": "^refs/heads/main$"
    }
  ],
  [
    {
      "type": "EVENT",
      "pattern": "PUSH"
    },
    {
      "type": "HEAD_REF",
      "pattern": "^refs/heads/myBranch$"
    }
  ]
]
```

excludeMatchedPattern 파라미터를 사용하여 빌드를 트리거하지 않을 이벤트를 지정할 수 있습니다. 이 예제에서는 태그 이벤트를 제외한 모든 요청에 대해 빌드가 트리거됩니다.

```
"filterGroups": [
  [
    {
      "type": "EVENT",
```

```
      "pattern": "PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
 PULL_REQUEST_MERGED, PULL_REQUEST_CLOSED"
    },
    {
      "type": "HEAD_REF",
      "pattern": "^refs/tags/.*",
      "excludeMatchedPattern": true
    }
  ]
]
```

계정 ID가 `actor-account-id`인 Bitbucket 사용자가 변경을 수행한 경우에만 빌드를 트리거하는 필터를 생성할 수 있습니다.

> **Note**
>
> Bitbucket 계정 ID를 확인하는 자세한 방법은 https://api.bitbucket.org/2.0/users/*user-name*을 참조하십시오. 여기서 *user-name*은 사용자의 Bitbucket 사용자 이름입니다.

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
 PULL_REQUEST_MERGED, PULL_REQUEST_CLOSED"
    },
    {
      "type": "ACTOR_ACCOUNT_ID",
      "pattern": "actor-account-id"
    }
  ]
]
```

`pattern` 인수의 정규식과 일치하는 이름을 갖는 파일이 변경되는 경우에만 빌드를 트리거하는 필터를 생성할 수 있습니다. 이 예제에서는 필터 그룹이 정규식 `^buildspec.*`와 일치하는 이름을 갖는 파일이 변경될 때만 빌드가 트리거되도록 지정합니다.

```
"filterGroups": [
  [
    {
```

```
      "type": "EVENT",
      "pattern": "PUSH"
    },
    {
      "type": "FILE_PATH",
      "pattern": "^buildspec.*"
    }
  ]
]
```

이 예제에서 필터 그룹은 파일이 src 또는 test 폴더에서 변경될 때만 빌드가 트리거되도록 지정합니다.

```
"filterGroups": [
    [
        {
            "type": "EVENT",
            "pattern": "PUSH"
        },
        {
            "type": "FILE_PATH",
            "pattern": "^src/.+|^test/.+"
        }
    ]
]
```

헤드 커밋 메시지가 패턴 인수의 정규식과 일치할 때만 빌드를 트리거하는 필터를 생성할 수 있습니다. 이 예제에서 필터 그룹은 푸시 이벤트의 헤드 커밋 메시지가 정규식 \[CodeBuild\]와 일치할 때만 빌드가 트리거되도록 지정합니다.

```
  "filterGroups": [
    [
      {
        "type": "EVENT",
        "pattern": "PUSH"
      },
      {
        "type": "COMMIT_MESSAGE",
        "pattern": "\[CodeBuild\]"
      }
    ]
  ]
```

Bitbucket Webhook 이벤트 필터링(CloudFormation)

CloudFormation 템플릿을 사용하여 웹후크 이벤트를 필터링하려면 AWS CodeBuild 프로젝트의 `FilterGroups` 속성을 사용합니다. 다음 CloudFormation 템플릿의 YAML 형식 부분은 두 개의 필터 그룹을 생성합니다. 이들은 하나 또는 둘 모두 true로 평가되면 빌드를 트리거합니다.

- 첫 번째 필터 그룹은 계정 ID가 `12345`와 일치하지 않는 Bitbucket 사용자가 정규식 `^refs/heads/main$`와 일치하는 Git 참조 이름을 갖는 브랜치에서 생성 또는 업데이트한 pull 요청을 지정합니다.
- 두 번째 필터 그룹은 정규식 `^refs/heads/.*`와 일치하는 Git 참조 이름을 갖는 브랜치에서 생성되는 push 요청을 지정합니다.
- 세 번째 필터 그룹은 정규식 `\[CodeBuild\]`와 일치하는 헤드 커밋 메시지에서 push 요청을 지정합니다.

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Name: MyProject
    ServiceRole: service-role
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      Type: LINUX_CONTAINER
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:5.0
    Source:
      Type: BITBUCKET
      Location: source-location
    Triggers:
      Webhook: true
      FilterGroups:
        - - Type: EVENT
            Pattern: PULL_REQUEST_CREATED,PULL_REQUEST_UPDATED
          - Type: BASE_REF
            Pattern: ^refs/heads/main$
            ExcludeMatchedPattern: false
          - Type: ACTOR_ACCOUNT_ID
            Pattern: 12345
            ExcludeMatchedPattern: true
        - - Type: EVENT
            Pattern: PUSH
          - Type: HEAD_REF
```

```
          Pattern: ^refs/heads/.*
        - Type: FILE_PATH
          Pattern: READ_ME
          ExcludeMatchedPattern: true
      - - Type: EVENT
          Pattern: PUSH
        - Type: COMMIT_MESSAGE
          Pattern: \[CodeBuild\]
        - Type: FILE_PATH
          Pattern: ^src/.+|^test/.+
```

GitHub 글로벌 및 조직 웹후크

CodeBuild GitHub 글로벌 또는 조직 웹후크를 사용하여 GitHub 조직 또는 엔터프라이즈 내의 모든 리포지토리에서 웹후크 이벤트를 기반으로 빌드를 시작할 수 있습니다. 글로벌 및 조직 웹후크는 기존 GitHub 웹후크 이벤트 유형에서 작동하며 CodeBuild 웹후크를 생성할 때 범위 구성을 추가하여 구성할 수 있습니다. 글로벌 및 조직 웹후크를 사용하여 CodeBuild 내에서 자체 호스팅 GitHub Action 실행기를 설정하여 단일 프로젝트 내의 여러 리포지토리에서 `WORKFLOW_JOB_QUEUED` 이벤트를 수신할 수도 있습니다.

주제

- 글로벌 또는 조직 GitHub 웹후크 설정
- GitHub 글로벌 또는 조직 웹후크 이벤트 필터링(콘솔)
- GitHub 조직 웹후크 이벤트 필터링(CloudFormation)

글로벌 또는 조직 GitHub 웹후크 설정

글로벌 또는 조직 GitHub 웹후크를 설정하는 상위 단계는 다음과 같습니다. 글로벌 및 조직 GitHub 웹후크에 대한 자세한 내용은 GitHub 글로벌 및 조직 웹후크 섹션을 참조하세요.

1. 프로젝트의 소스 위치를 `CODEBUILD_DEFAULT_WEBHOOK_SOURCE_LOCATION`으로 설정합니다.
2. 웹후크의 범위 구성에서 범위가 조직인지 글로벌 웹후크인지에 따라 `GITHUB_ORGANIZATION` 또는 `GITHUB_GLOBAL` 중 하나로 설정합니다. 자세한 내용은 웹후크 유형을 참조하세요.
3. 웹후크의 범위 구성의 일부로 이름을 지정합니다. 조직 웹후크의 경우 조직 이름이고 글로벌 웹후크의 경우 엔터프라이즈 이름입니다.

> **Note**
>
> 프로젝트의 소스 유형이 GITHUB_ENTERPRISE인 경우 웹후크 범위 구성의 일부로 도메인을 지정해야 합니다.

4. (선택 사항) 조직 또는 엔터프라이즈 내의 특정 리포지토리에 대해서만 웹후크 이벤트를 수신하려면 웹후크를 생성할 때 REPOSITORY_NAME을 필터로 지정할 수 있습니다.
5. 조직 웹후크를 생성하는 경우 CodeBuild에 GitHub 내에서 조직 수준 웹후크를 생성할 수 있는 권한이 있는지 확인합니다. 조직 웹후크 권한을 사용하여 GitHub 개인 액세스 토큰을 생성하거나 CodeBuild OAuth를 사용할 수 있습니다. 자세한 내용은 GitHub 및 GitHub Enterprise Server 액세스 토큰 단원을 참조하십시오.

   조직 웹후크는 기존 GitHub 웹후크 이벤트 유형에서 작동합니다.
6. 글로벌 웹후크를 생성하는 경우 웹후크를 수동으로 생성해야 합니다. GitHub 내에서 웹후크를 수동으로 생성하는 방법에 대한 자세한 내용은 GitHub 수동 웹후크 섹션을 참조하세요.

   글로벌 웹후크는 WORKFLOW_JOB_QUEUED 이벤트 유형만 지원합니다. 자세한 내용은 자습서: CodeBuild 호스팅 GitHub Action 실행기 구성 단원을 참조하십시오.

GitHub 글로벌 또는 조직 웹후크 이벤트 필터링(콘솔)

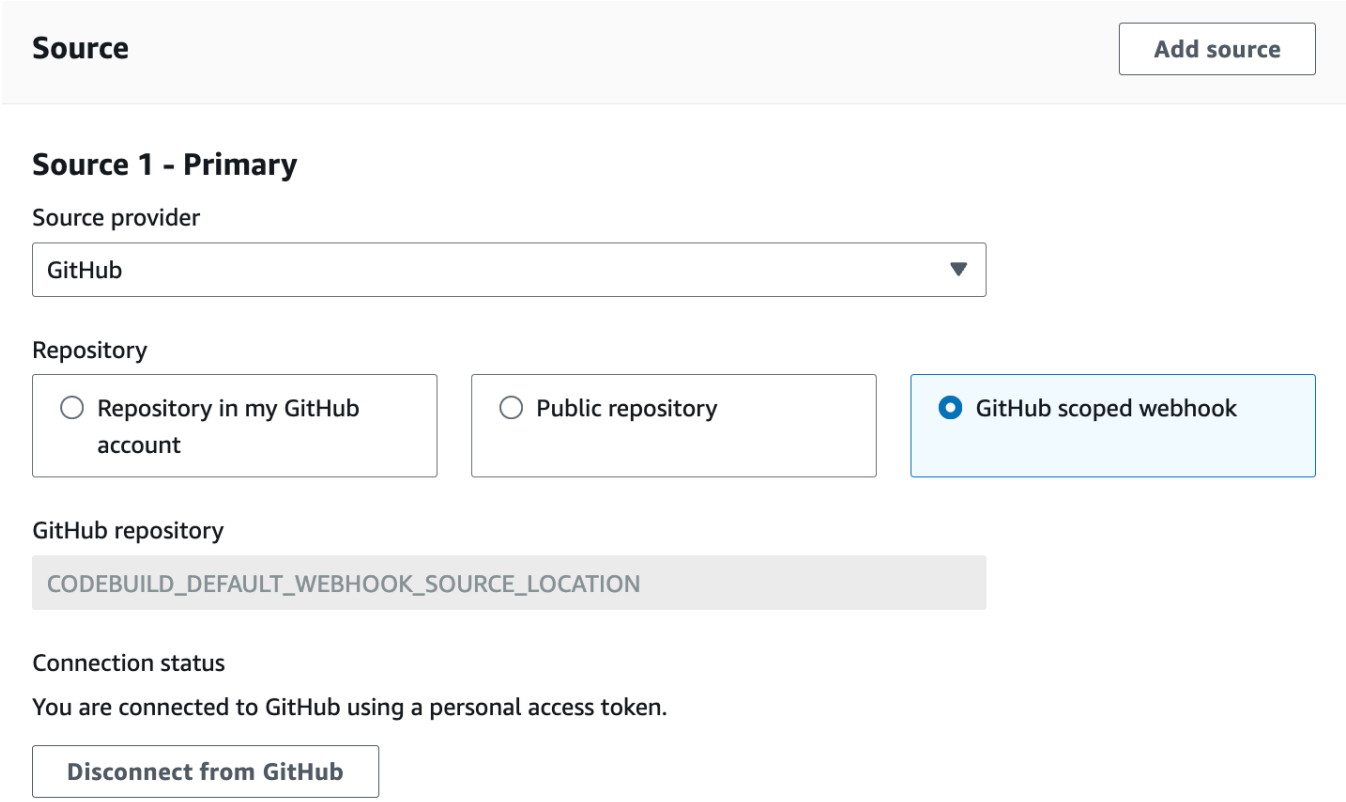
콘솔을 통해 GitHub 프로젝트를 생성할 때 다음 옵션을 선택하여 프로젝트 내에 GitHub 글로벌 또는 조직 웹후크를 생성합니다. 글로벌 및 조직 GitHub 웹후크에 대한 자세한 내용은 GitHub 글로벌 및 조직 웹후크 섹션을 참조하세요.

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 빌드 프로젝트를 생성합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 및 빌드 실행(콘솔) 섹션을 참조하세요.

   - 소스에서 다음과 같이 합니다.
     - 소스 공급자에서 GitHub 또는 GitHub Enterprise를 선택합니다.
     - 리포지토리에서 GitHub 범위 웹후크를 선택합니다.

       GitHub 리포지토리는 자동으로 CODEBUILD_DEFAULT_WEBHOOK_SOURCE_LOCATION으로 설정되며, 이는 글로벌 및 조직 웹후크에 필요한 소스 위치입니다.

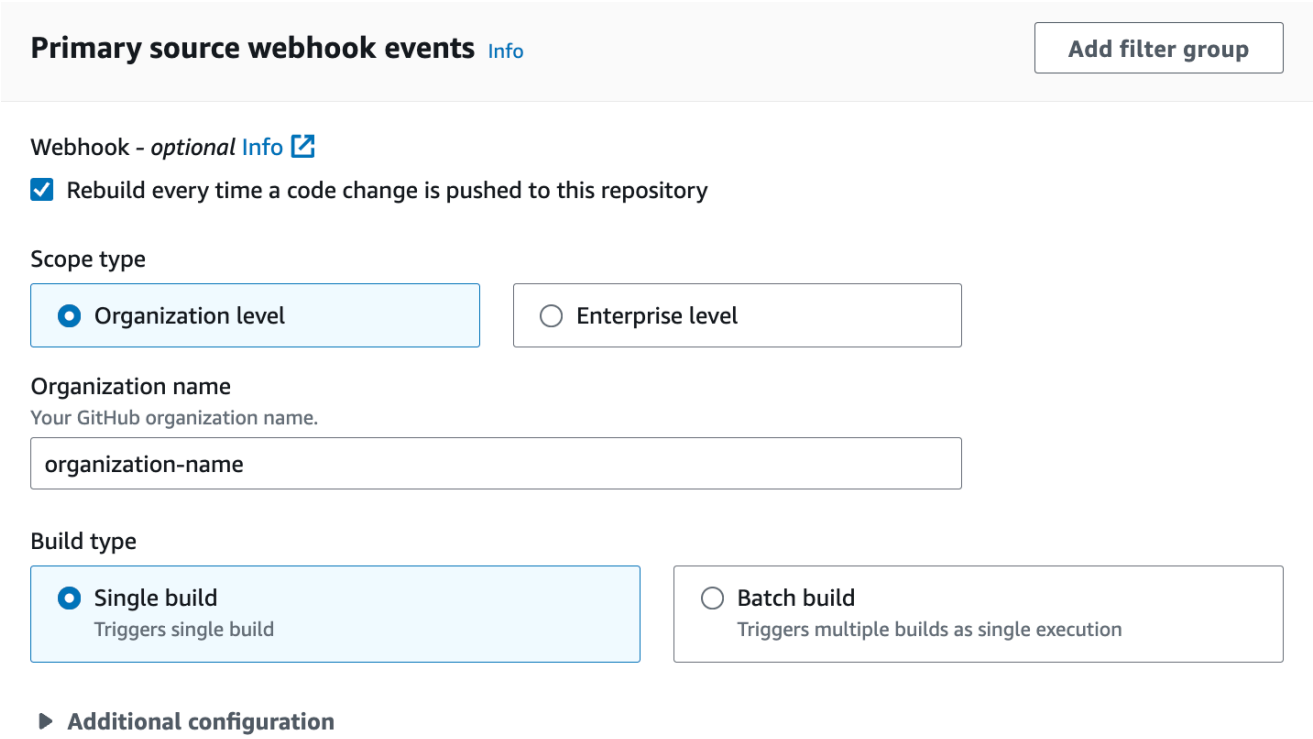
> **Note**
>
> 조직 웹후크를 사용하는 경우 CodeBuild에 GitHub 내에서 조직 수준 웹후크를 생성할 수 있는 권한이 있는지 확인합니다. 기존 OAuth 연결을 사용하는 경우 CodeBuild에 이 권한을 부여하려면 연결을 다시 생성해야 할 수 있습니다. 또는 CodeBuild 수동 웹후크 기능을 사용하여 수동으로 웹후크를 생성할 수 있습니다. 기존 GitHub OAuth 토큰이 있고 조직 권한을 추가하려는 경우 OAuth 토큰의 권한을 취소하고 CodeBuild 콘솔을 통해 토큰을 다시 연결할 수 있습니다.



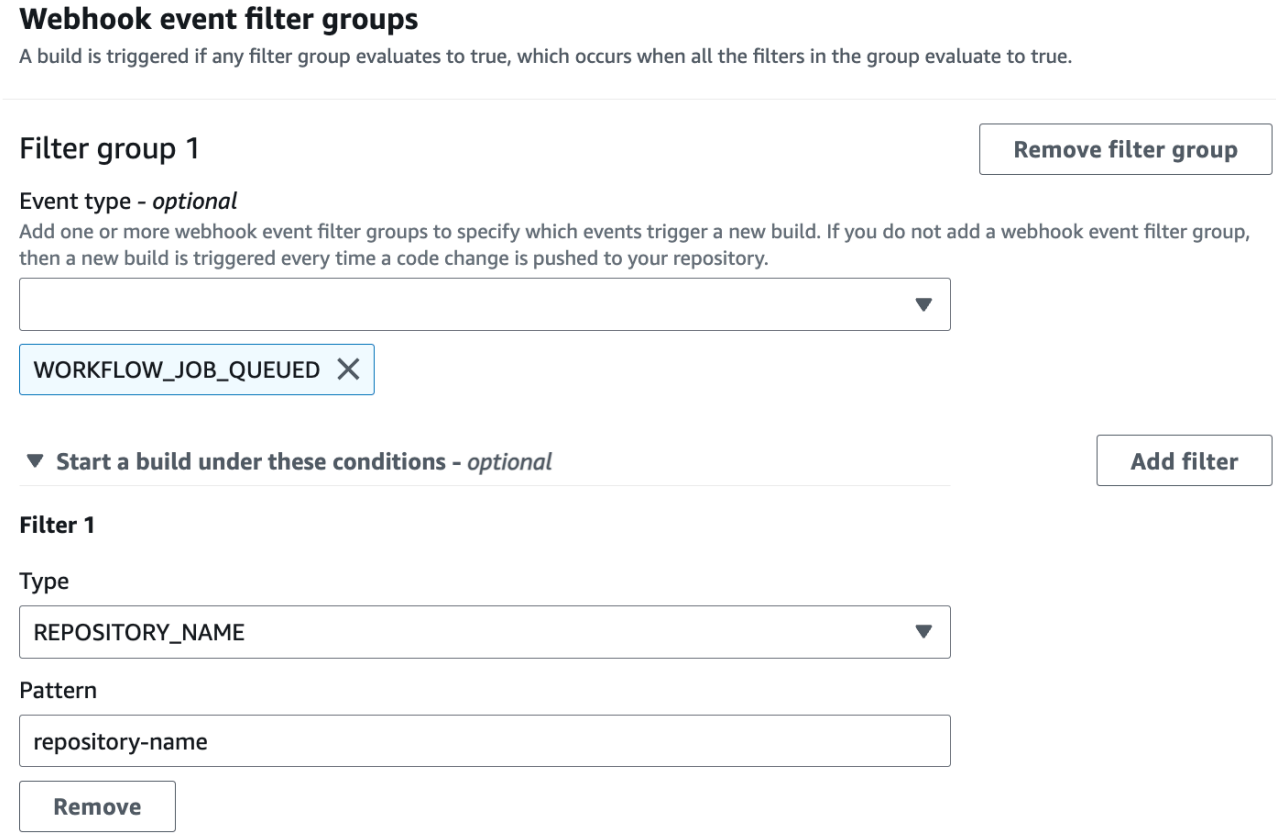
- 기본 소스 웹후크 이벤트에서:
  - 범위 유형 에서 조직 웹후크를 생성하는 경우 조직 수준을 선택하고 글로벌 웹후크를 생성하는 경우 엔터프라이즈 수준을 선택합니다.
  - 웹후크가 글로벌 웹후크인지 아니면 조직 웹후크인지에 따라 이름에 엔터프라이즈 또는 조직 이름을 입력합니다.

    프로젝트의 소스 유형이 `GITHUB_ENTERPRISE`인 경우 웹후크 조직 구성의 일부로 도메인을 지정해야 합니다. 예를 들어 조직의 URL이 **`https://domain.com/orgs/org-name`**인 경우 도메인은 **`https://domain.com`**입니다.

> **Note**
>
> 웹후크가 생성된 후에는 이 이름을 변경할 수 없습니다. 이름을 변경하려면 웹후크를 삭제하고 다시 생성할 수 있습니다. 웹후크를 완전히 제거하려면 프로젝트 소스 위치를 GitHub 리포지토리로 업데이트할 수도 있습니다.



- (선택 사항) 웹후크 이벤트 필터 그룹에서 새 빌드를 트리거할 이벤트를 지정할 수 있습니다. 특정 리포지토리의 웹후크 이벤트에서만 빌드를 트리거하는 필터로 `REPOSITORY_NAME`을 지정할 수도 있습니다.



이벤트 유형을 `WORKFLOW_JOB_QUEUED`로 설정하여 자체 호스팅 GitHub Action 실행기를 설정할 수도 있습니다. 자세한 내용은 [자습서: CodeBuild 호스팅 GitHub Action 실행기 구성] 단원을 참조하십시오.

3. 기본값으로 계속 진행한 다음 빌드 프로젝트 생성을 선택합니다.

GitHub 조직 웹후크 이벤트 필터링(CloudFormation)

CloudFormation 템플릿을 사용하여 조직 웹후크 이벤트를 필터링하려면 프로젝트의 `ScopeConfiguration` 속성을 사용합니다 AWS CodeBuild . 글로벌 및 조직 GitHub 웹후크에 대한 자세한 내용은 [GitHub 글로벌 및 조직 웹후크] 섹션을 참조하세요.

> **Note**
>
> 글로벌 웹후크 및 GitHub Enterprise 웹후크는에서 지원되지 않습니다 CloudFormation.

CloudFormation 템플릿의 다음 YAML 형식 부분은 4개의 필터 그룹을 생성합니다. 이들은 하나 또는 모두 true로 평가되면 빌드를 트리거합니다.

- 첫 번째 필터 그룹은 계정 ID가 12345와 일치하지 않는 GitHub 사용자가 정규식 ^refs/heads/main$와 일치하는 Git 참조 이름을 갖는 브랜치에서 생성 또는 업데이트한 pull 요청을 지정합니다.
- 두 번째 필터 그룹은 정규식 ^refs/heads/.*와 일치하는 Git 참조 이름을 갖는 브랜치에서 정규식 READ_ME와 일치하는 이름을 갖는 파일에 생성되는 push 요청을 지정합니다.
- 세 번째 필터 그룹은 정규식 \[CodeBuild\]와 일치하는 헤드 커밋 메시지에서 push 요청을 지정합니다.
- 네 번째 필터 그룹은 정규식 \[CI-CodeBuild\]와 일치하는 워크플로 이름을 가진 GitHub Action 워크플로 작업 요청을 지정합니다.

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Name: MyProject
    ServiceRole: service-role
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      Type: LINUX_CONTAINER
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:5.0
    Source:
      Type: GITHUB
      Location: source-location
    Triggers:
      Webhook: true
      ScopeConfiguration:
        Name: organization-name
        Scope: GITHUB_ORGANIZATION
      FilterGroups:
        - - Type: EVENT
            Pattern: PULL_REQUEST_CREATED,PULL_REQUEST_UPDATED
          - Type: BASE_REF
            Pattern: ^refs/heads/main$
            ExcludeMatchedPattern: false
          - Type: ACTOR_ACCOUNT_ID
            Pattern: 12345
            ExcludeMatchedPattern: true
        - - Type: EVENT
            Pattern: PUSH
          - Type: HEAD_REF
```

```
          Pattern: ^refs/heads/.*
        - Type: FILE_PATH
          Pattern: READ_ME
          ExcludeMatchedPattern: true
      - - Type: EVENT
          Pattern: PUSH
        - Type: COMMIT_MESSAGE
          Pattern: \[CodeBuild\]
        - Type: FILE_PATH
          Pattern: ^src/.+|^test/.+
      - - Type: EVENT
          Pattern: WORKFLOW_JOB_QUEUED
        - Type: WORKFLOW_NAME
          Pattern: \[CI-CodeBuild\]
```

GitHub 수동 웹후크

CodeBuild가 GitHub 내에서 웹후크를 자동으로 생성하려고 시도하지 않도록 수동 GitHub 웹후크를 구성할 수 있습니다. CodeBuild는 호출의 일부로 페이로드 URL을 반환하여 웹후크를 생성합니다.이 URL은 GitHub 내에서 웹후크를 수동으로 생성하는 데 사용할 수 있습니다. CodeBuild가 GitHub 계정에서 웹후크를 생성할 수 있는 허용 목록에 없는 경우에도 빌드 프로젝트의 웹후크를 수동으로 생성할 수 있습니다.

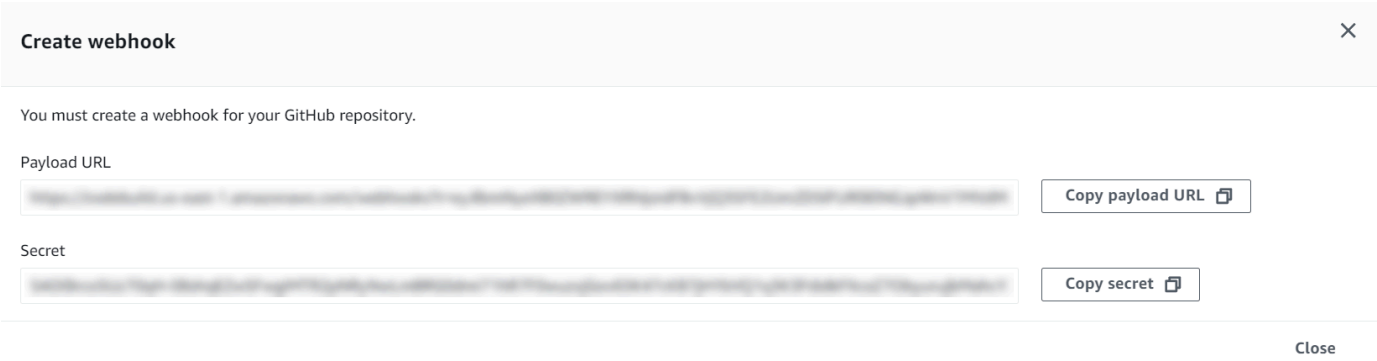
다음 절차에 따라 GitHub 수동 웹후크를 생성합니다.

GitHub 수동 웹후크를 생성하려면

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 빌드 프로젝트를 생성합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 및 빌드 실행(콘솔) 섹션을 참조하세요.
   - 소스에서 다음과 같이 합니다.
     - 소스 공급자에서 GitHub를 선택합니다.
     - 리포지토리에 대해 내 GitHub 계정의 리포지토리를 선택합니다.
     - 리포지토리 URL에 **https://github.com/*user-name*/*repository-name***을 입력합니다.
   - 기본 소스 웹후크 이벤트에서:
     - 웹후크 - 선택 사항에서 코드 변경이 이 리포지토리로 푸시될 때마다 다시 빌드를 선택합니다.

- 추가 구성 및 수동 생성 - 선택 사항을 선택하고 GitHub 콘솔에서 이 리포지토리에 대한 웹후크 수동 생성을 선택합니다.

3. 기본값으로 계속 진행한 다음 빌드 프로젝트 생성을 선택합니다. 나중에 사용할 페이로드 URL 및 보안 암호 값을 기록해 둡니다.



4. https://github.com/*user-name*/*repository-name*/settings/hooks에서 GitHub 콘솔을 열고 웹후크 추가를 선택합니다.

   - 페이로드 URL에 앞서 기록한 페이로드 URL 값을 입력합니다.
   - 콘텐츠 유형에 대해 application/json을 선택합니다.
   - 보안 암호에 앞서 기록한 보안 암호 값을 입력합니다.
   - CodeBuild로 웹후크 페이로드를 전송할 개별 이벤트를 구성합니다. 이 웹후크를 트리거할 이벤트는 무엇입니까?에서 개별 이벤트 선택을 선택한 다음, 푸시, Pull 요청 및 릴리스 이벤트 중에서 선택합니다. WORKFLOW_JOB_QUEUED 이벤트에 대한 빌드를 시작하려면 워크플로 작업을 선택합니다. GitHub Action 실행기에 대한 자세한 내용은 자습서: CodeBuild 호스팅 GitHub Action 실행기 구성 섹션을 참조하세요. CodeBuild에서 지원하는 이벤트 유형에 대한 자세한 내용은 GitHub Webhook 이벤트 섹션을 참조하세요.

5. 웹후크 추가를 선택합니다.

GitHub Webhook 이벤트

Webhook 필터 그룹을 사용하여 어느 GitHub Webhook 이벤트가 빌드를 트리거할지 지정할 수 있습니다. 예를 들어 특정 분기가 변경된 경우에만 빌드가 트리거되도록 지정할 수 있습니다.

하나 이상의 웹후크 필터 그룹을 생성하여 어느 웹후크 이벤트가 빌드를 트리거할지 지정할 수 있습니다. 필터 그룹이 true로 평가(그룹 내 모든 필터가 true로 평가)되면 빌드가 트리거됩니다. 필터 그룹을 생성할 때 다음을 지정합니다.

이벤트

GitHub의 경우 PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED, PULL_REQUEST_REOPENED, PULL_REQUEST_MERGED, PULL_REQUEST_CLOSED, RELEASED, PRERELEASED 및 WORKFLOW_JOB_QUEUED 이벤트 가운데 하나 이상을 선택할 수 있습니다. webhook 이벤트 유형은 webhook 페이로드의 X-GitHub-Event 헤더에 있습니다. X-GitHub-Event 헤더에서 pull_request 또는 push를 볼 수 있습니다. 풀 요청 이벤트의 경우 유형은 webhook 이벤트 페이로드의 action 필드에 있습니다. 다음 표에서는 X-GitHub-Event 헤더 값과 webhook 풀 요청 페이로드 action 필드 값이 사용 가능한 이벤트 유형에 매핑되는 방법을 보여줍니다.

X-GitHub-Event 헤더 값	Webhook 이벤트 페이로드 **action** 값	이벤트 유형
pull_request	opened	PULL_REQUEST_CREATED
pull_request	reopened	PULL_REQUEST_REOPENED
pull_request	synchronize	PULL_REQUEST_UPDATED
pull_request	closed 및 merged 필드는 true임	PULL_REQUEST_MERGED
pull_request	closed 및 merged 필드는 false임	PULL_REQUEST_CLOSED
push	해당 사항 없음	PUSH
release	릴리스	RELEASED
release	사전 릴리스	PRERELEASED
workflow_job	queued	WORKFLOW_JOB_QUEUED

> **Note**
>
> GitHub 및 GitHub Enterprise Server에서만 PULL_REQUEST_REOPENED 이벤트 유형을 사용할 수 있습니다. GitHub에서만 RELEASED 및 PRERELEASED 이벤트 유형을 사용할 수 있

습니다. WORKFLOW_JOB_QUEUED에 대한 자세한 내용은 자습서: CodeBuild 호스팅 GitHub Action 실행기 구성 섹션을 참조하세요.

하나 이상의 선택적 필터

정규식을 사용하여 필터를 지정합니다. 이벤트가 빌드를 트리거하려면 연결된 그룹 내의 필터가 모두 true로 평가되어야 합니다.

CodeBuild는 미리 보기, 뒤로 보기, 역참조 또는 원자 그룹을 지원하지 않는 RE2 정규식 구문을 사용하여 필터 패턴을 평가합니다. 자세한 내용은 GitHub 웹 사이트의 RE2 구문 페이지를 참조하세요.

ACTOR_ACCOUNT_ID(콘솔의 ACTOR_ID)

GitHub 또는 GitHub Enterprise Server 계정 ID가 정규식 패턴과 일치하면 Webhook 이벤트가 빌드를 트리거합니다. webhook 페이로드에 있는 sender 객체의 id 속성에서 이 값을 찾을 수 있습니다.

HEAD_REF

헤드 참조가 정규식 패턴(예: refs/heads/branch-name 또는 refs/tags/tag-name)과 일치하면 webhook 이벤트가 빌드를 트리거합니다. 푸시 이벤트의 경우 webhook 페이로드의 ref 속성에서 참조 이름을 찾을 수 있습니다. 풀 요청 이벤트의 경우 webhook 페이로드에 있는 head 객체의 ref 속성에서 브랜치 이름을 찾을 수 있습니다.

BASE_REF

기본 참조가 정규식 패턴(예: refs/heads/branch-name)과 일치하면 webhook 이벤트가 빌드를 트리거합니다. 풀 요청 이벤트에서만 BASE_REF 필터를 사용할 수 있습니다. webhook 페이로드에 있는 base 객체의 ref 속성에서 브랜치 이름을 찾을 수 있습니다.

FILE_PATH

변경된 파일의 경로가 정규식 패턴과 일치하면 webhook가 빌드를 트리거합니다. FILE_PATH 필터는 GitHub push 및 pull 요청 이벤트와 GitHub Enterprise Server push 이벤트에서 사용할 수 있습니다. GitHub Enterprise Server pull 요청 이벤트에서는 사용할 수 없습니다.

> **Note**
>
> FILE_PATH 필터는 푸시 또는 풀 요청에서 처음 100개의 변경된 파일에 대해 평가됩니다. 필터가 의도한 대로 적용되도록 하려면 변경된 파일 수를이 한도 내에서 유지합니

다. 풀 요청 이벤트의 경우 풀 요청 빌드 정책을 사용하여 빌드를 시작할 수 있는 풀 요청을 제어하는 것이 좋습니다.

`COMMIT_MESSAGE`

헤드 커밋 메시지가 정규식 패턴과 일치하면 webhook가 빌드를 트리거합니다. `COMMIT_MESSAGE` 필터는 GitHub push 및 pull 요청 이벤트와 GitHub Enterprise Server push 이벤트에서 사용할 수 있습니다. GitHub Enterprise Server pull 요청 이벤트에서는 사용할 수 없습니다.

`TAG_NAME`

릴리스의 태그 이름이 정규식 패턴과 일치하면 웹후크가 빌드를 트리거합니다. `TAG_NAME` 필터는 GitHub 릴리스 및 사전 릴리스된 요청 이벤트와 함께 사용할 수 있습니다.

`RELEASE_NAME`

릴리스 이름이 정규식 패턴과 일치하면 웹후크가 빌드를 트리거합니다. `RELEASE_NAME` 필터는 GitHub 릴리스 및 사전 릴리스된 요청 이벤트와 함께 사용할 수 있습니다.

`REPOSITORY_NAME`

리포지토리 이름이 정규식 패턴과 일치하면 웹후크가 빌드를 트리거합니다. `REPOSITORY_NAME` 필터는 GitHub 글로벌 또는 조직 웹후크에서만 사용할 수 있습니다.

`ORGANIZATION_NAME`

조직 이름이 정규식 패턴과 일치하면 웹후크가 빌드를 트리거합니다. `ORGANIZATION_NAME` 필터는 GitHub 글로벌 웹후크에서만 사용할 수 있습니다.

`WORKFLOW_NAME`

워크플로 이름이 정규식 패턴과 일치하면 웹후크가 빌드를 트리거합니다. `WORKFLOW_NAME` 필터는 GitHub Action 워크플로 작업 대기열 요청 이벤트와 함께 사용할 수 있습니다.

> **Note**
>
> GitHub 리포지토리의 webhook 설정에서 webhook 페이로드를 찾을 수 있습니다.

주제

- GitHub Webhook 이벤트 필터링(콘솔)
- GitHub Webhook 이벤트 필터링(SDK)
- GitHub Webhook 이벤트 필터링(CloudFormation)

GitHub Webhook 이벤트 필터링(콘솔)

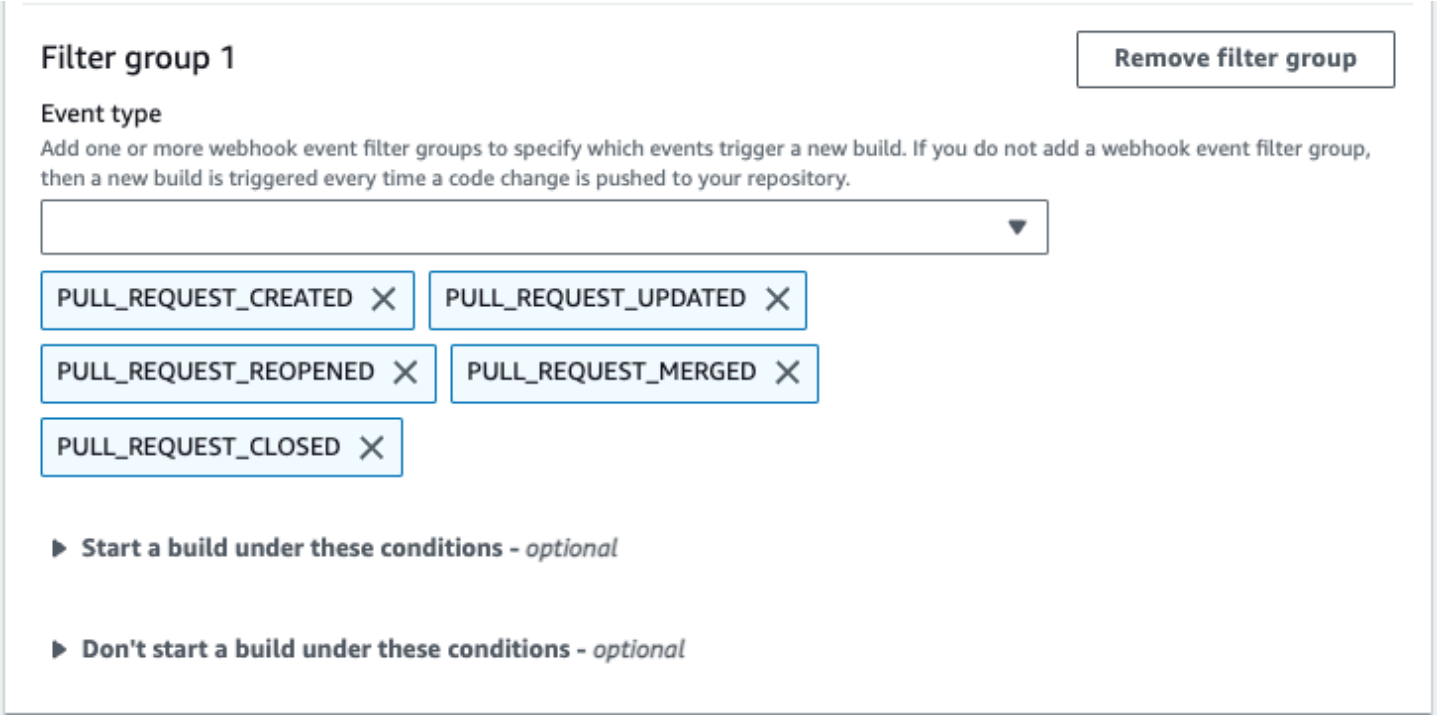
다음 지침에 따라 AWS Management Console을 사용하여 GitHub 웹후크 이벤트를 필터링합니다. GitHub 웹후크 이벤트 유형에 대한 자세한 내용은 GitHub Webhook 이벤트 섹션을 참조하세요.

기본 소스 webhook 이벤트에서 다음을 선택합니다. 이 섹션은 소스 리포지토리로 GitHub 계정의 리포지토리를 선택한 경우에만 사용할 수 있습니다.

1. 프로젝트를 생성할 때 코드 변경이 이 리포지토리로 푸시될 때마다 다시 빌드를 선택합니다.
2. 이벤트 유형에서 하나 이상의 이벤트를 선택합니다.
3. 이벤트가 빌드를 트리거할 때를 필터링하려면 Start a build under these conditions(다음 조건에서 빌드를 시작)에서 하나 이상의 선택적 필터를 추가합니다.
4. 이벤트가 트리거되지 않을 때를 필터링하려면 Don't start a build under these conditions(다음 조건에서 빌드를 시작하지 않음)에서 하나 이상의 선택적 필터를 추가합니다.
5. 필요한 경우 필터 그룹 추가를 선택하여 다른 필터 그룹을 추가합니다.

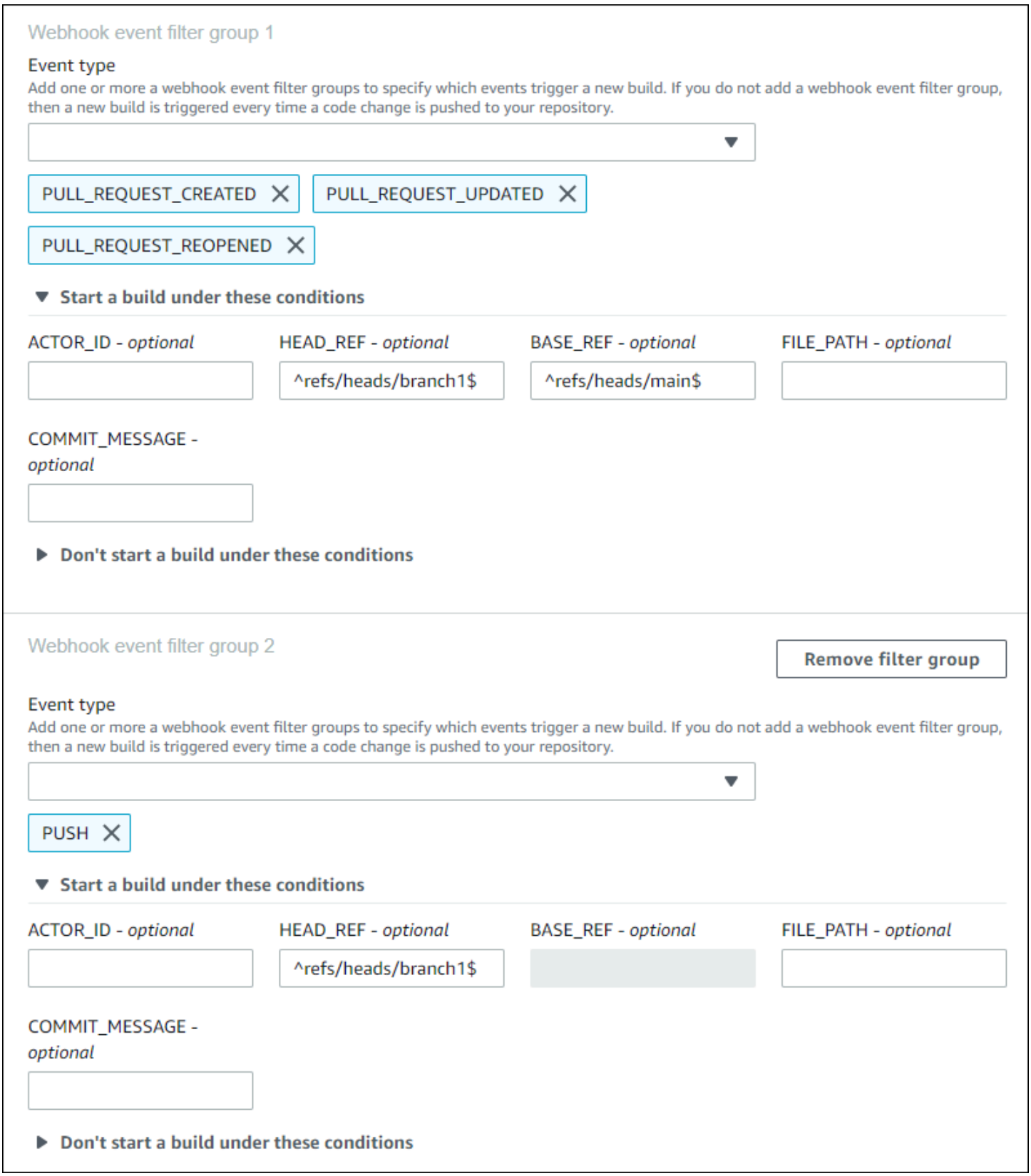
자세한 내용은 빌드 프로젝트 만들기(콘솔) 및 AWS CodeBuild API 참조의 WebHookFilter를 참조하세요.

이 예제에서는 Webhook 필터 그룹이 pull 요청에 대해서만 빌드를 트리거합니다.

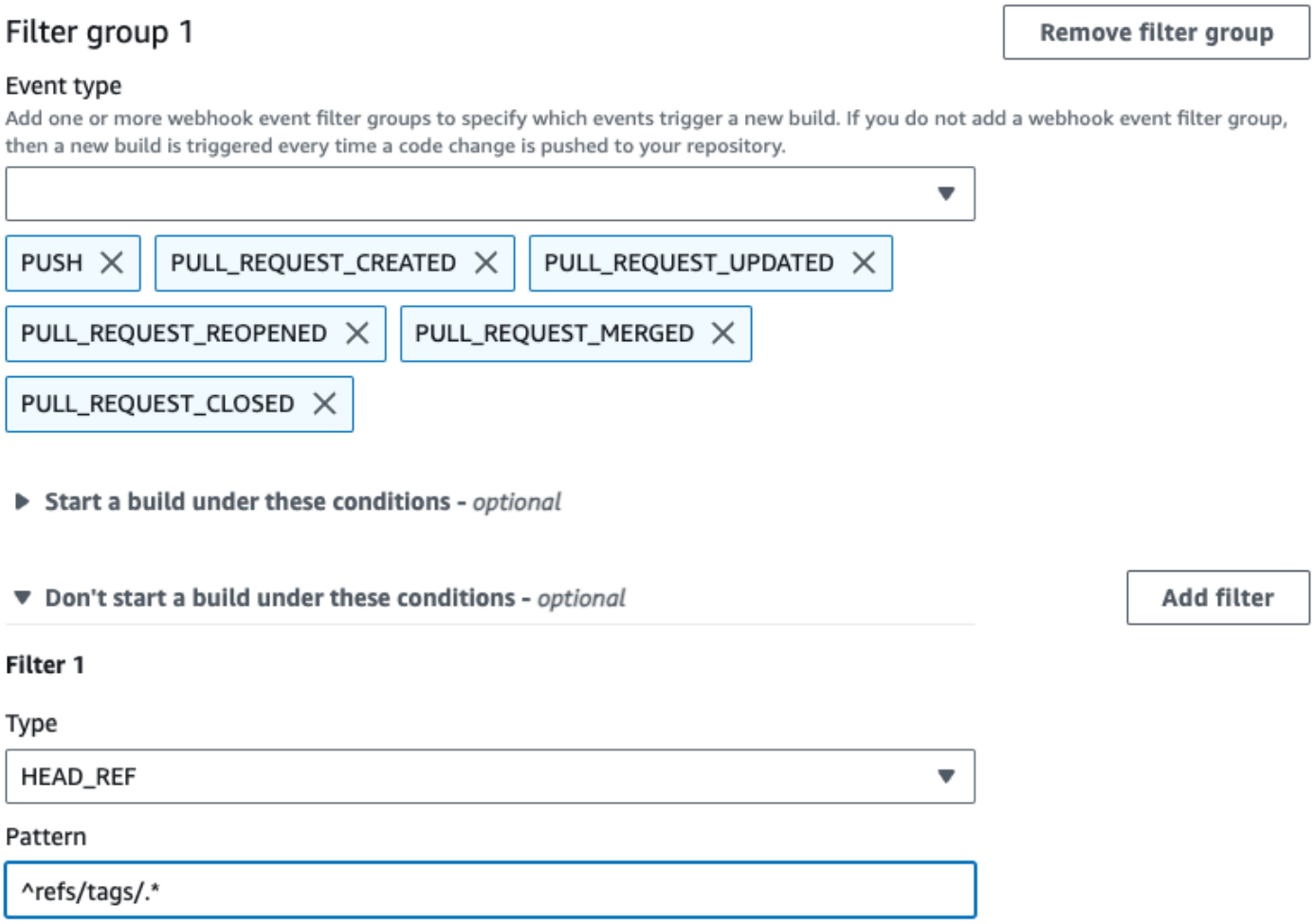


두 Webhook 필터 그룹을 사용하는 예제에서는 하나 또는 두 필터 그룹이 true로 평가되면 빌드가 트리거됩니다.

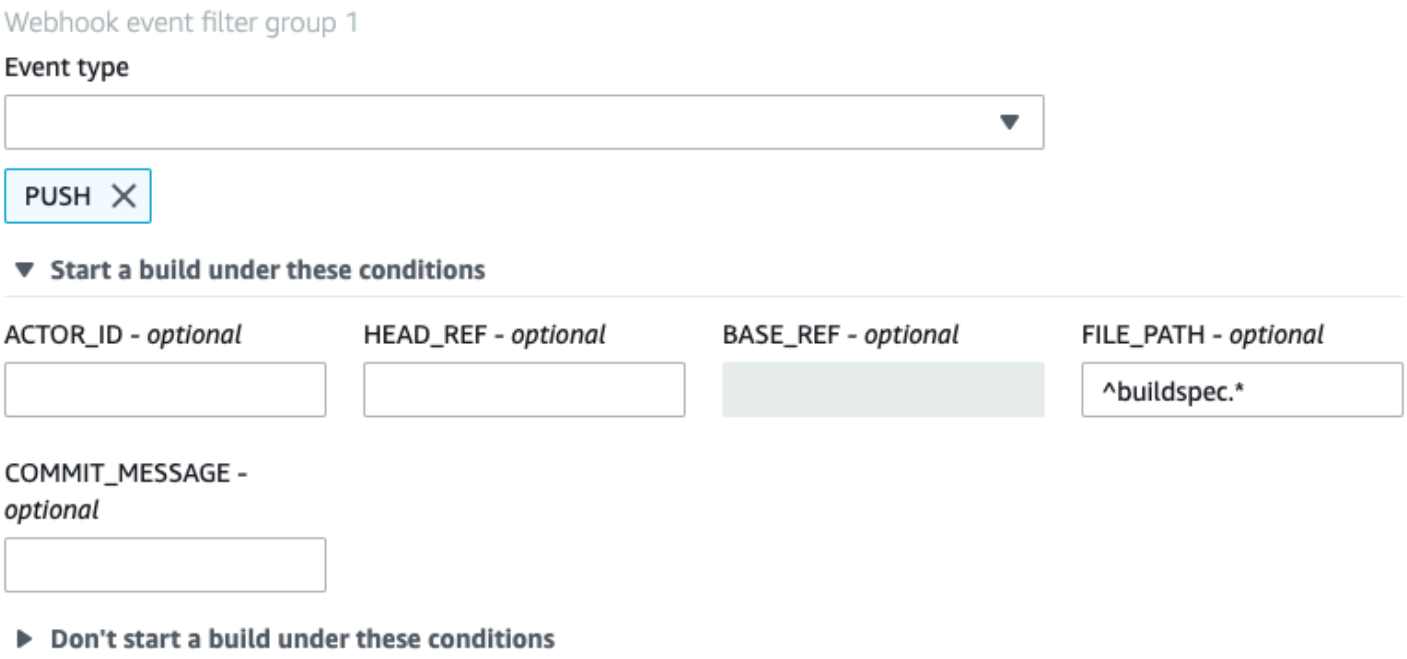
- 첫 번째 필터 그룹은 정규식 `^refs/heads/main$`와 일치하는 Git 참조 이름과 `^refs/heads/branch1$`와 일치하는 헤드 참조를 갖는 브랜치에서 생성되거나 업데이트되거나 다시 열린 pull 요청을 지정합니다.
- 두 번째 필터 그룹은 정규식 `^refs/heads/branch1$`와 일치하는 Git 참조 이름을 갖는 브랜치에서 push 요청을 지정합니다.



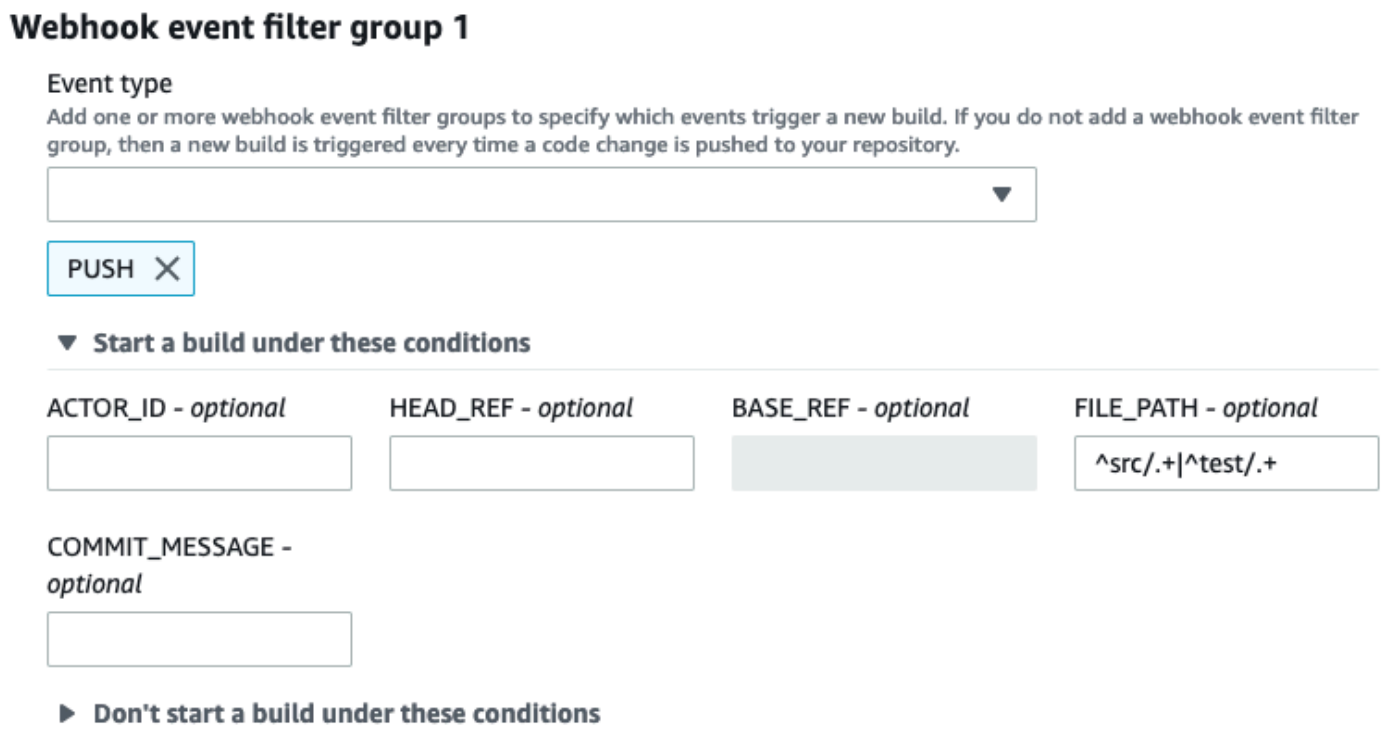
이 예제에서는 Webhook 필터 그룹이 태그 이벤트를 제외한 모든 요청에 대해 빌드를 트리거합니다.



이 예제에서는 Webhook 필터 그룹이 정규식 `^buildspec.*`와 일치하는 이름을 갖는 파일이 변경될 때만 빌드를 트리거합니다.

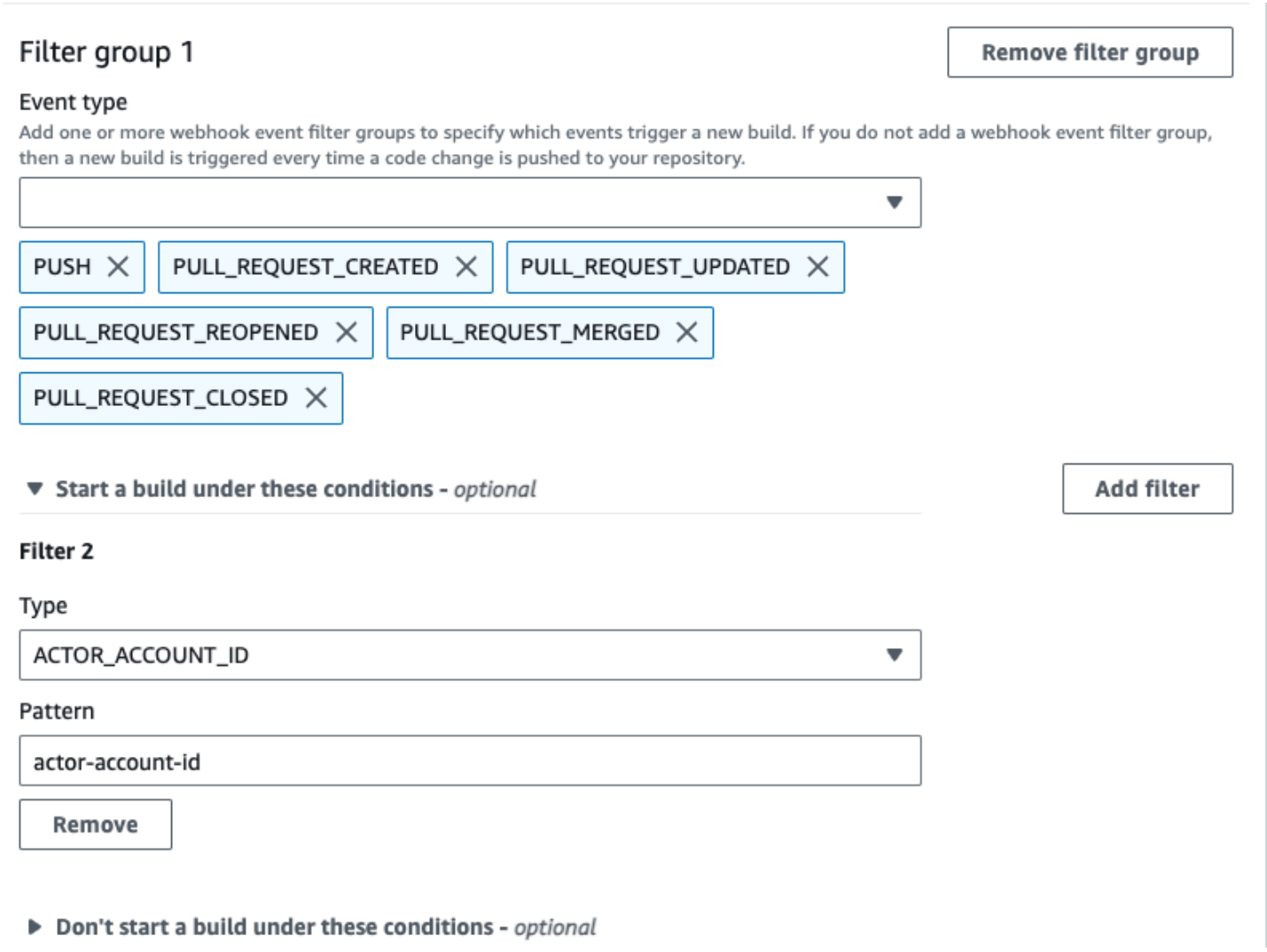


이 예제에서 Webhook 필터 그룹은 파일이 `src` 또는 `test` 폴더에서 변경된 경우에만 빌드를 트리거합니다.

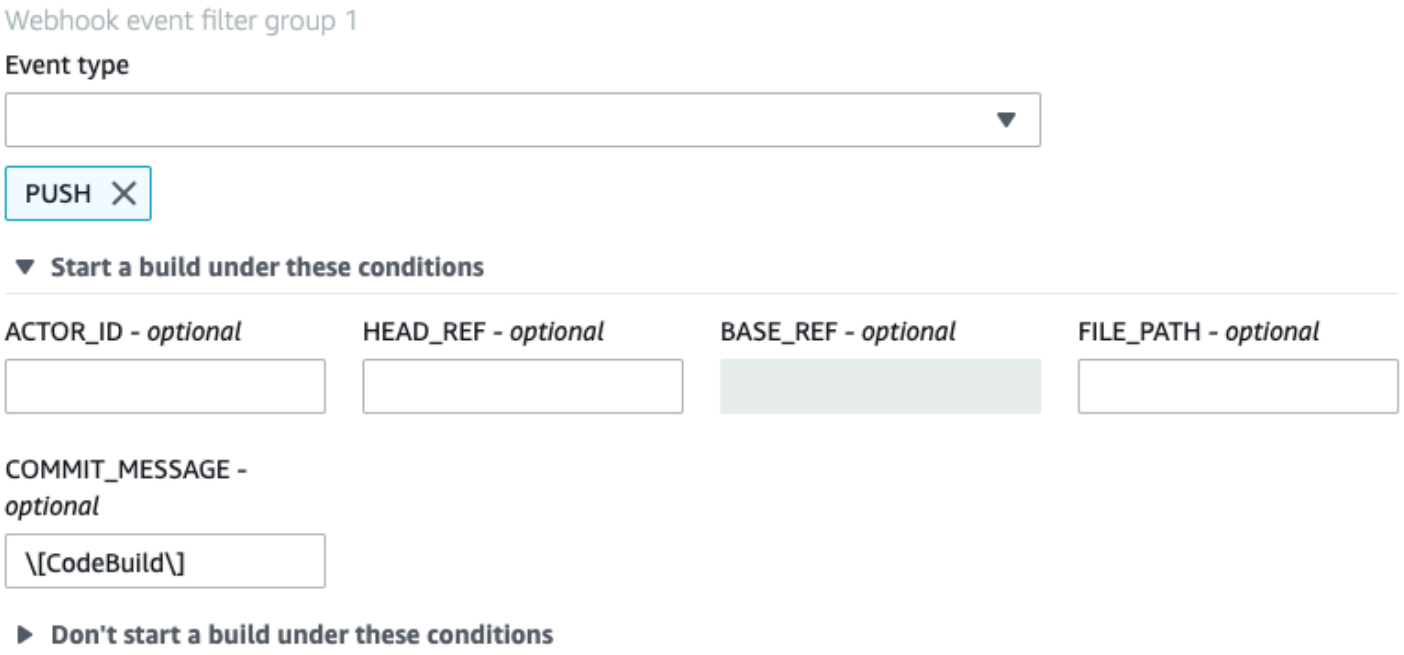


이 예제에서는 Webhook 필터 그룹이 계정 ID가 정규식 `actor-account-id`와 일치하는 지정된 GitHub 또는 GitHub Enterprise Server 사용자가 변경을 수행한 경우에만 빌드를 트리거합니다.

> **Note**
>
> GitHub 계정 ID를 확인하는 자세한 방법은 https://api.github.com/users/*user-name*을 참조하십시오. 여기서 *user-name*은 사용자의 GitHub 사용자 이름입니다.

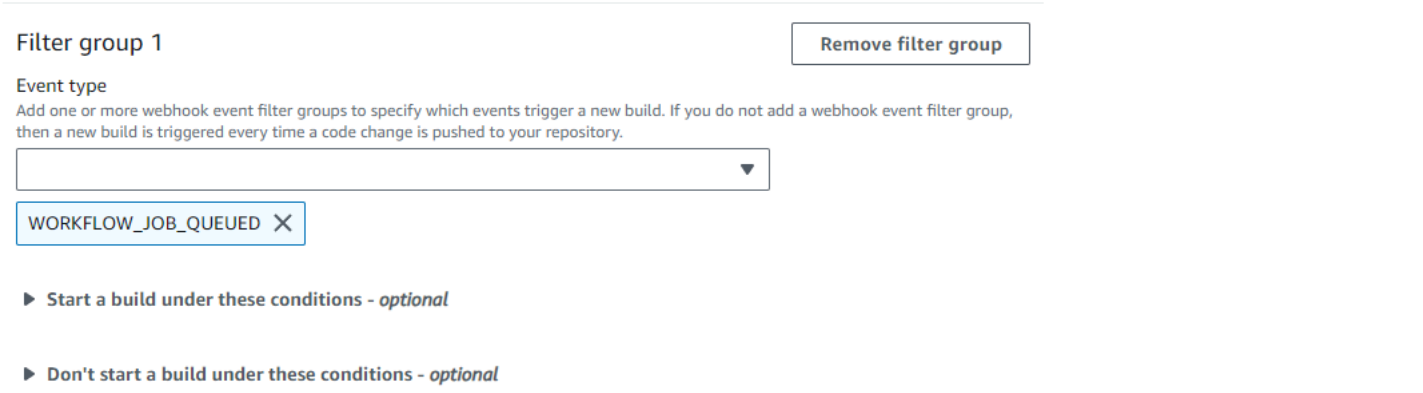


이 예제에서 webhook 필터 그룹은 헤드 커밋 메시지가 정규식 `\[CodeBuild\]`와 일치할 때 푸시 이벤트에 대한 빌드를 트리거합니다.

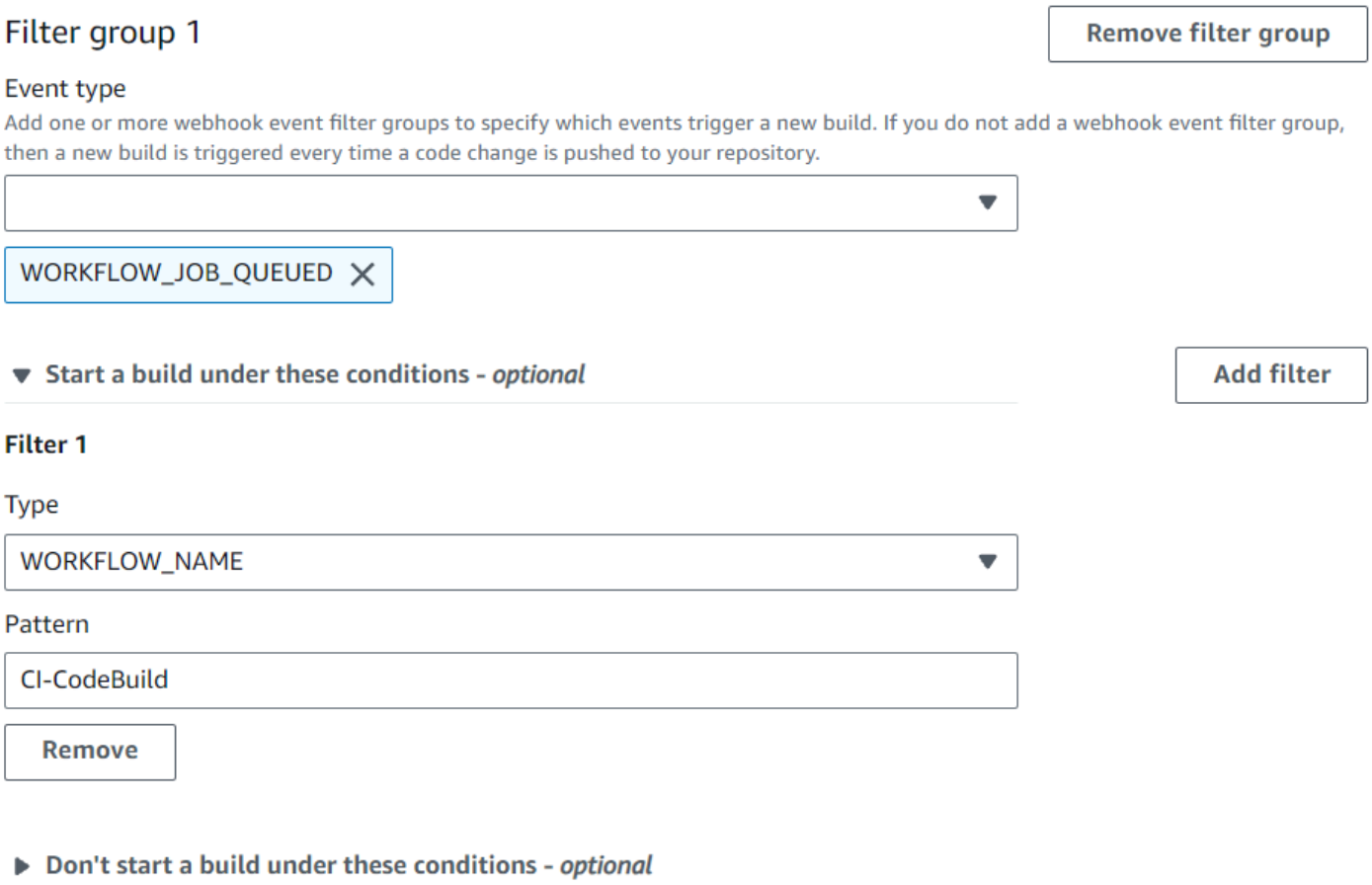


이 예제에서는 웹후크 필터 그룹이 GitHub Action 워크플로 작업 이벤트에 대한 빌드만 트리거합니다.

> **Note**
>
> CodeBuild는 웹후크에 WORKFLOW_JOB_QUEUED 이벤트 필터가 포함된 필터 그룹이 있는 경우에만 GitHub Action 워크플로 작업을 처리합니다.



이 예제에서 웹후크 필터 그룹은 정규식 `CI-CodeBuild`와 일치하는 워크플로 이름에 대한 빌드를 트리거합니다.



GitHub Webhook 이벤트 필터링(SDK)

AWS CodeBuild SDK를 사용하여 웹후크 이벤트를 필터링하려면 `CreateWebhook` 또는 `UpdateWebhook` API 메서드의 요청 구문에서 `filterGroups` 필드를 사용합니다. 자세한 내용은 CodeBuild API 참조의 WebHookFilter를 참조하세요.

GitHub 웹후크 이벤트 유형에 대한 자세한 내용은 GitHub Webhook 이벤트 섹션을 참조하세요.

pull 요청에 대해서만 빌드를 트리거하는 Webhook 필터를 생성하려면 요청 구문에 다음을 삽입합니다.

```
"filterGroups": [
   [
        {
            "type": "EVENT",
            "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
 PULL_REQUEST_REOPENED, PULL_REQUEST_MERGED, PULL_REQUEST_CLOSED"
        }
```

```
    ]
]
```

지정된 브랜치에 대해서만 빌드를 트리거하는 Webhook 필터를 생성하려면 pattern 파라미터를 사용하여 브랜치 이름을 필터링하는 정규식을 지정합니다. 두 필터 그룹을 사용하는 예제에서는 하나 또는 두 필터 그룹이 true로 평가되면 빌드가 트리거됩니다.

- 첫 번째 필터 그룹은 정규식 ^refs/heads/main$와 일치하는 Git 참조 이름과 ^refs/heads/myBranch$와 일치하는 헤드 참조를 갖는 브랜치에서 생성되거나 업데이트되거나 다시 열린 pull 요청을 지정합니다.
- 두 번째 필터 그룹은 정규식 ^refs/heads/myBranch$와 일치하는 Git 참조 이름을 갖는 브랜치에서 push 요청을 지정합니다.

```
"filterGroups": [
    [
        {
            "type": "EVENT",
            "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
 PULL_REQUEST_REOPENED"
        },
        {
            "type": "HEAD_REF",
            "pattern": "^refs/heads/myBranch$"
        },
        {
            "type": "BASE_REF",
            "pattern": "^refs/heads/main$"
        }
    ],
    [
        {
            "type": "EVENT",
            "pattern": "PUSH"
        },
        {
            "type": "HEAD_REF",
            "pattern": "^refs/heads/myBranch$"
        }
    ]
]
```

excludeMatchedPattern 파라미터를 사용하여 빌드를 트리거하지 않을 이벤트를 지정할 수 있습니다. 예를 들어 이 예제에서는 태그 이벤트를 제외한 모든 요청에 대해 빌드가 트리거됩니다.

```
"filterGroups": [
    [
        {
            "type": "EVENT",
            "pattern": "PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
 PULL_REQUEST_REOPENED, PULL_REQUEST_MERGED, PULL_REQUEST_CLOSED"
        },
        {
            "type": "HEAD_REF",
            "pattern": "^refs/tags/.*",
            "excludeMatchedPattern": true
        }
    ]
]
```

pattern 인수의 정규식과 일치하는 이름을 갖는 파일이 변경되는 경우에만 빌드를 트리거하는 필터를 생성할 수 있습니다. 이 예제에서는 필터 그룹이 정규식 ^buildspec.*와 일치하는 이름을 갖는 파일이 변경될 때만 빌드가 트리거되도록 지정합니다.

```
"filterGroups": [
    [
        {
            "type": "EVENT",
            "pattern": "PUSH"
        },
        {
            "type": "FILE_PATH",
            "pattern": "^buildspec.*"
        }
    ]
]
```

이 예제에서 필터 그룹은 파일이 src 또는 test 폴더에서 변경될 때만 빌드가 트리거되도록 지정합니다.

```
"filterGroups": [
    [
        {
```

```
            "type": "EVENT",
            "pattern": "PUSH"
        },
        {
            "type": "FILE_PATH",
            "pattern": "^src/.+|^test/.+"
        }
    ]
]
```

계정 ID가 `actor-account-id`인 지정된 GitHub 또는 GitHub Enterprise Server 사용자가 변경을 수행한 경우에만 빌드를 트리거하는 필터를 생성할 수 있습니다.

> **Note**
>
> GitHub 계정 ID를 확인하는 자세한 방법은 https://api.github.com/users/*user-name*을 참조하십시오. 여기서 *user-name*은 사용자의 GitHub 사용자 이름입니다.

```
"filterGroups": [
    [
        {
            "type": "EVENT",
            "pattern": "PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
 PULL_REQUEST_REOPENED, PULL_REQUEST_MERGED, PULL_REQUEST_CLOSED"
        },
        {
            "type": "ACTOR_ACCOUNT_ID",
            "pattern": "actor-account-id"
        }
    ]
]
```

헤드 커밋 메시지가 패턴 인수의 정규식과 일치할 때만 빌드를 트리거하는 필터를 생성할 수 있습니다. 이 예제에서 필터 그룹은 푸시 이벤트의 헤드 커밋 메시지가 정규식 `\[CodeBuild\]`와 일치할 때만 빌드가 트리거되도록 지정합니다.

```
"filterGroups": [
    [
        {
            "type": "EVENT",
```

```
            "pattern": "PUSH"
        },
        {
            "type": "COMMIT_MESSAGE",
            "pattern": "\[CodeBuild\]"
        }
    ]
]
```

GitHub Action 워크플로에 대한 빌드를 트리거하는 웹후크 필터를 생성하려면 요청 구문에 다음을 삽입합니다.

```
"filterGroups": [
   [
        {
            "type": "EVENT",
            "pattern": "WORKFLOW_JOB_QUEUED"
        }
    ]
]
```

GitHub Webhook 이벤트 필터링(CloudFormation)

CloudFormation 템플릿을 사용하여 웹후크 이벤트를 필터링하려면 AWS CodeBuild 프로젝트의 `FilterGroups` 속성을 사용합니다.

GitHub 웹후크 이벤트 유형에 대한 자세한 내용은 GitHub Webhook 이벤트 섹션을 참조하세요.

다음 CloudFormation 템플릿의 YAML 형식 부분은 두 개의 필터 그룹을 생성합니다. 이들은 하나 또는 둘 모두 true로 평가되면 빌드를 트리거합니다.

- 첫 번째 필터 그룹은 계정 ID가 `12345`와 일치하지 않는 GitHub 사용자가 정규식 `^refs/heads/main$`와 일치하는 Git 참조 이름을 갖는 브랜치에서 생성 또는 업데이트한 pull 요청을 지정합니다.
- 두 번째 필터 그룹은 정규식 `^refs/heads/.*`와 일치하는 Git 참조 이름을 갖는 브랜치에서 정규식 `READ_ME`와 일치하는 이름을 갖는 파일에 생성되는 push 요청을 지정합니다.
- 세 번째 필터 그룹은 정규식 `\[CodeBuild\]`와 일치하는 헤드 커밋 메시지에서 push 요청을 지정합니다.
- 네 번째 필터 그룹은 정규식 `\[CI-CodeBuild\]`와 일치하는 워크플로 이름을 가진 GitHub Action 워크플로 작업 요청을 지정합니다.

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Name: MyProject
    ServiceRole: service-role
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      Type: LINUX_CONTAINER
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:5.0
    Source:
      Type: GITHUB
      Location: source-location
    Triggers:
      Webhook: true
      FilterGroups:
        - - Type: EVENT
            Pattern: PULL_REQUEST_CREATED,PULL_REQUEST_UPDATED
          - Type: BASE_REF
            Pattern: ^refs/heads/main$
            ExcludeMatchedPattern: false
          - Type: ACTOR_ACCOUNT_ID
            Pattern: 12345
            ExcludeMatchedPattern: true
        - - Type: EVENT
            Pattern: PUSH
          - Type: HEAD_REF
            Pattern: ^refs/heads/.*
          - Type: FILE_PATH
            Pattern: READ_ME
            ExcludeMatchedPattern: true
        - - Type: EVENT
            Pattern: PUSH
          - Type: COMMIT_MESSAGE
            Pattern: \[CodeBuild\]
          - Type: FILE_PATH
            Pattern: ^src/.+|^test/.+
        - - Type: EVENT
            Pattern: WORKFLOW_JOB_QUEUED
          - Type: WORKFLOW_NAME
            Pattern: \[CI-CodeBuild\]
```

GitLab 그룹 웹후크

CodeBuild GitLab 그룹 웹후크를 사용하여 GitLab 그룹 내의 모든 리포지토리에서 웹후크 이벤트를 기반으로 빌드를 시작할 수 있습니다. 그룹 웹후크는 기존 GitLab 웹후크 이벤트 유형에서 작동하며 CodeBuild 웹후크를 생성할 때 범위 구성을 추가하여 구성할 수 있습니다. 그룹 웹후크를 사용하여 CodeBuild 내에서 자체 호스팅 GitLab 실행기를 설정하여 단일 프로젝트 내의 여러 리포지토리에서 `WORKFLOW_JOB_QUEUED` 이벤트를 수신할 수도 있습니다.

주제

- 그룹 GitLab 웹후크 설정
- GitLab 그룹 웹후크 이벤트 필터링(콘솔)
- GitLab 그룹 웹후크 이벤트 필터링(CloudFormation)

그룹 GitLab 웹후크 설정

그룹 GitLab 웹후크를 설정하는 상위 단계는 다음과 같습니다. 그룹 GitLab 웹후크에 대한 자세한 내용은 GitLab 그룹 웹후크 섹션을 참조하세요.

1. 프로젝트의 소스 위치를 `CODEBUILD_DEFAULT_WEBHOOK_SOURCE_LOCATION`으로 설정합니다.
2. 웹후크의 범위 구성에서 범위를 `GITLAB_GROUP`으로 설정합니다.
3. 웹후크의 범위 구성의 일부로 이름을 지정합니다. 그룹 웹후크의 경우 그룹 이름입니다.

> **Note**
>
> 프로젝트의 소스 유형이 `GITLAB_SELF_MANAGED`인 경우 웹후크 범위 구성의 일부로 도메인을 지정해야 합니다.

4. (선택 사항) 조직 또는 엔터프라이즈 내의 특정 리포지토리에 대해서만 웹후크 이벤트를 수신하려면 웹후크를 생성할 때 `REPOSITORY_NAME`을 필터로 지정할 수 있습니다.
5. 그룹 웹후크를 생성할 때 CodeBuild에 GitLab 내에서 그룹 수준 웹후크를 생성할 권한이 있는지 확인합니다. 이를 위해 CodeBuild OAuth를 사용할 수 있습니다 AWS CodeConnections. 자세한 내용은 CodeBuild의 GitLab 액세스 섹션을 참조하세요.

   그룹 웹후크는 기존 GitLab 웹후크 이벤트 유형에서 작동합니다.

GitLab 그룹 웹후크 이벤트 필터링(콘솔)

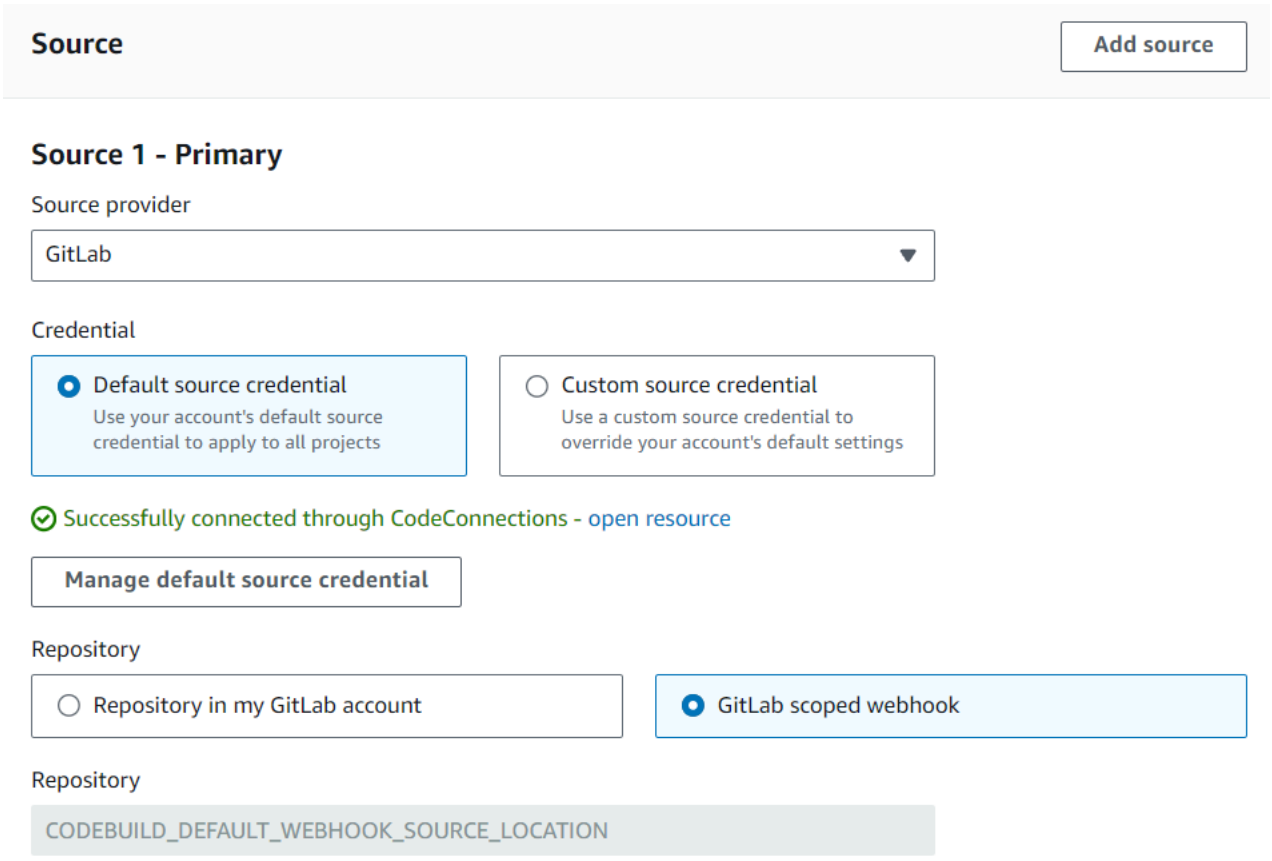
콘솔을 통해 GitLab 프로젝트를 생성할 때 다음 옵션을 선택하여 프로젝트 내에 GitLab 그룹 웹후크를 생성합니다. 그룹 GitLab 웹후크에 대한 자세한 내용은 GitLab 그룹 웹후크 섹션을 참조하세요.

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 빌드 프로젝트를 생성합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 및 빌드 실행(콘솔) 섹션을 참조하세요.

   - 소스에서 다음과 같이 합니다.
     - 소스 공급자에서 GitLab 또는 GitLab Self Managed를 선택합니다.
     - 리포지토리에서 GitLab 범위 웹후크를 선택합니다.

     GitLab 리포지토리는 그룹 웹후크에 필요한 소스 위치인 `CODEBUILD_DEFAULT_WEBHOOK_SOURCE_LOCATION`으로 자동 설정됩니다.

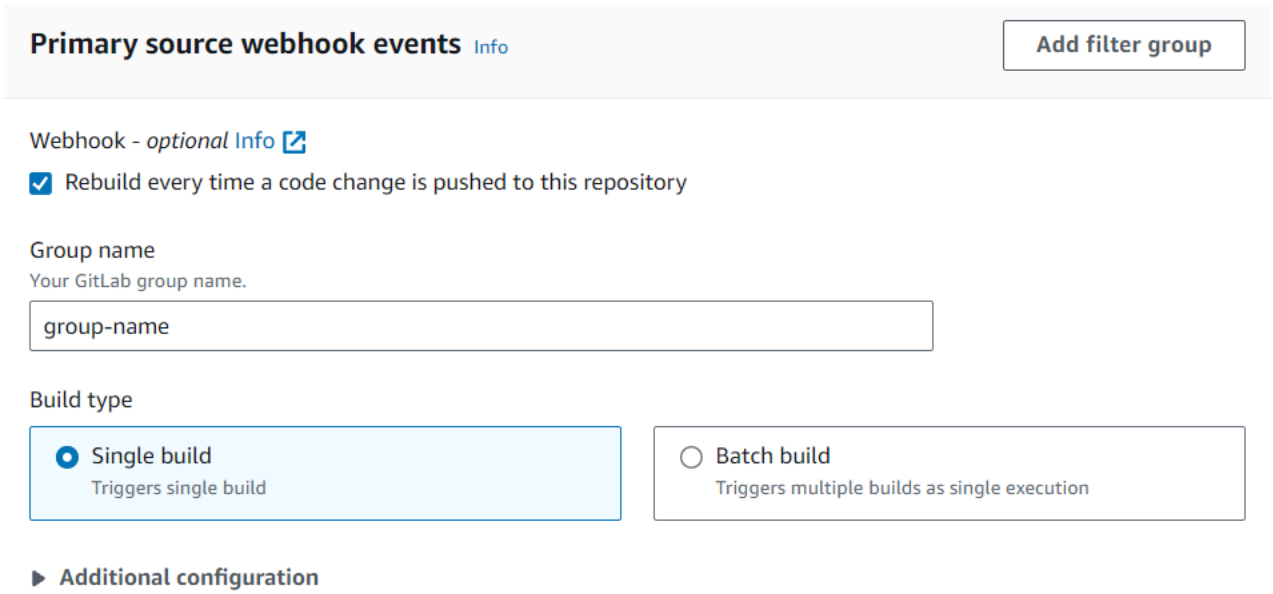
     > **Note**
     >
     > 그룹 웹후크를 사용하는 경우 CodeBuild에 GitLab 내에서 그룹 수준 웹후크를 생성할 수 있는 권한이 있는지 확인합니다. 기존 OAuth 연결을 사용하는 경우 CodeBuild에 이 권한을 부여하려면 연결을 다시 생성해야 할 수 있습니다.



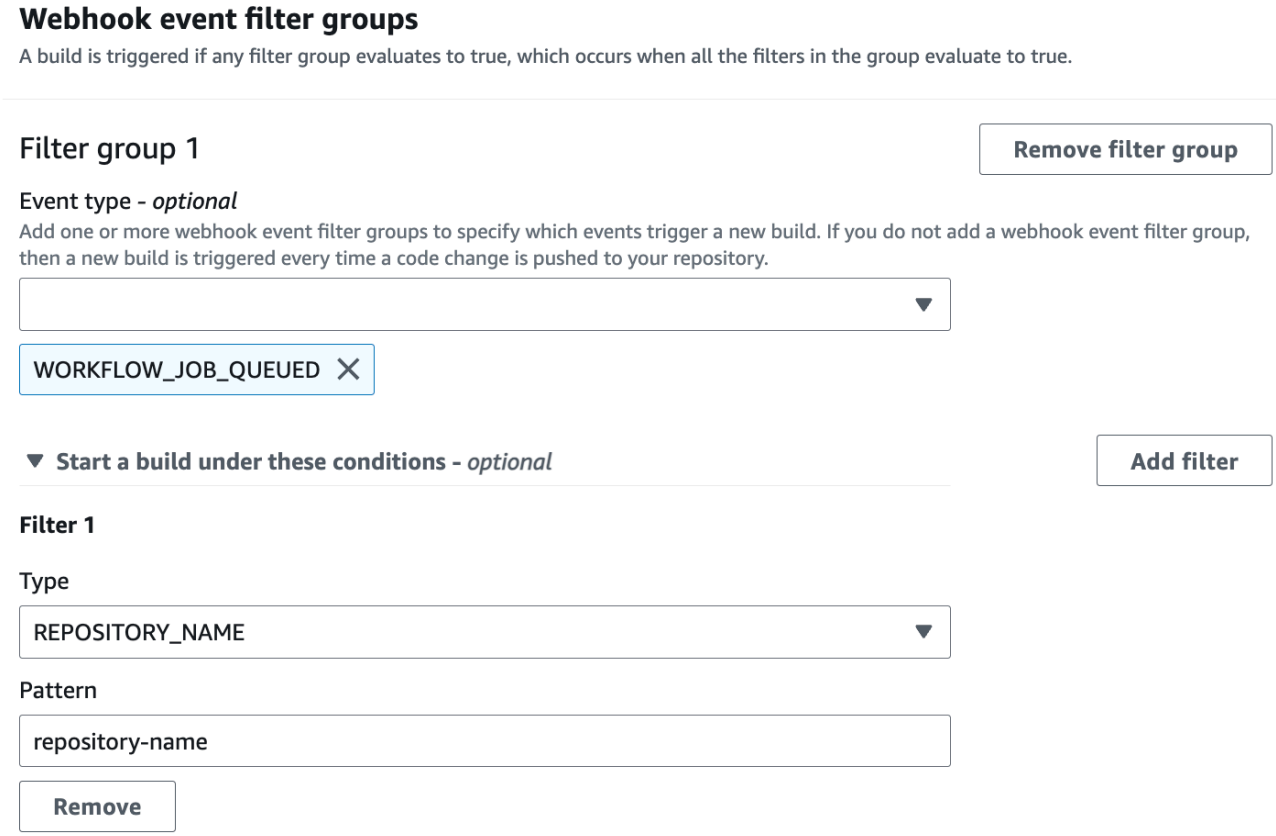
- 기본 소스 웹후크 이벤트에서:
  - 그룹 이름에 그룹 이름을 입력합니다.

    프로젝트의 소스 유형이 `GITLAB_SELF_MANAGED`인 경우 웹후크 그룹 구성의 일부로 도메인을 지정해야 합니다. 예를 들어 그룹의 URL이 **`https://domain.com/group/group-name`**인 경우 도메인은 **`https://domain.com`**입니다.

    > **Note**
    >
    > 웹후크가 생성된 후에는 이 이름을 변경할 수 없습니다. 이름을 변경하려면 웹후크를 삭제하고 다시 생성할 수 있습니다. 웹후크를 완전히 제거하려면 프로젝트 소스 위치를 GitLab 리포지토리로 업데이트할 수도 있습니다.



- (선택 사항) 웹후크 이벤트 필터 그룹에서 새 빌드를 트리거할 이벤트를 지정할 수 있습니다. 특정 리포지토리의 웹후크 이벤트에서만 빌드를 트리거하는 필터로 `REPOSITORY_NAME`을 지정할 수도 있습니다.



이벤트 유형을 WORKFLOW_JOB_QUEUED로 설정하여 자체 호스팅 GitLab 실행기를 설정할 수도 있습니다. 자세한 내용은 의 자체 관리형 GitLab 실행기 AWS CodeBuild 섹션을 참조하세요.

3. 기본값으로 계속 진행한 다음 빌드 프로젝트 생성을 선택합니다.

GitLab 그룹 웹후크 이벤트 필터링(CloudFormation)

CloudFormation 템플릿을 사용하여 그룹 웹후크 이벤트를 필터링하려면 프로젝트의 ScopeConfiguration 속성을 사용합니다 AWS CodeBuild . 그룹 GitLab 웹후크에 대한 자세한 내용은 GitLab 그룹 웹후크 섹션을 참조하세요.

CloudFormation 템플릿의 다음 YAML 형식 부분은 4개의 필터 그룹을 생성합니다. 이들은 하나 또는 모두 true로 평가되면 빌드를 트리거합니다.

- 첫 번째 필터 그룹은 계정 ID가 12345와 일치하지 않는 GitLab 사용자가 정규식 ^refs/heads/main$와 일치하는 Git 참조 이름을 갖는 분기에서 생성 또는 업데이트한 pull 요청을 지정합니다.
- 두 번째 필터 그룹은 정규식 ^refs/heads/.*와 일치하는 Git 참조 이름을 갖는 브랜치에서 정규식 READ_ME와 일치하는 이름을 갖는 파일에 생성되는 push 요청을 지정합니다.
- 세 번째 필터 그룹은 정규식 \[CodeBuild\]와 일치하는 헤드 커밋 메시지에서 push 요청을 지정합니다.
- 네 번째 필터 그룹은 정규식 \[CI-CodeBuild\]와 일치하는 CI/CD 파이프라인 이름이 있는 GitLab CI/CD 파이프라인 작업 요청을 지정합니다.

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Name: MyProject
    ServiceRole: service-role
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      Type: LINUX_CONTAINER
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:5.0
    Source:
      Type: GITLAB
      Location: source-location
```

```
    Triggers:
      Webhook: true
      ScopeConfiguration:
        Name: group-name
        Scope: GITLAB_GROUP
      FilterGroups:
        - - Type: EVENT
            Pattern: PULL_REQUEST_CREATED,PULL_REQUEST_UPDATED
          - Type: BASE_REF
            Pattern: ^refs/heads/main$
            ExcludeMatchedPattern: false
          - Type: ACTOR_ACCOUNT_ID
            Pattern: 12345
            ExcludeMatchedPattern: true
        - - Type: EVENT
            Pattern: PUSH
          - Type: HEAD_REF
            Pattern: ^refs/heads/.*
          - Type: FILE_PATH
            Pattern: READ_ME
            ExcludeMatchedPattern: true
        - - Type: EVENT
            Pattern: PUSH
          - Type: COMMIT_MESSAGE
            Pattern: \[CodeBuild\]
          - Type: FILE_PATH
            Pattern: ^src/.+|^test/.+
        - - Type: EVENT
            Pattern: WORKFLOW_JOB_QUEUED
          - Type: WORKFLOW_NAME
            Pattern: \[CI-CodeBuild\]
```

GitLab 수동 웹후크

CodeBuild가 GitLab 내에서 웹후크를 자동으로 생성하려고 시도하지 않도록 수동 GitLab 웹후크를 구성할 수 있습니다. CodeBuild는 호출의 일부로 페이로드 URL을 반환하여 웹후크를 생성합니다.이 URL은 GitLab 내에서 웹후크를 수동으로 생성하는 데 사용할 수 있습니다. CodeBuild가 GitLab 계정에서 웹후크를 생성할 수 있는 허용 목록에 없는 경우에도 빌드 프로젝트의 웹후크를 수동으로 생성할 수 있습니다.

다음 절차에 따라 GitLab 수동 웹후크를 생성합니다.

GitLab 수동 웹후크를 생성하려면

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 빌드 프로젝트를 생성합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 및 빌드 실행(콘솔) 섹션을 참조하세요.
   - 소스에서 다음과 같이 합니다.
     - 소스 공급자에서 GitLab을 선택합니다.
     - 리포지토리에 대해 내 GitLab 계정의 리포지토리를 선택합니다.
     - 리포지토리 URL에 **https://gitlab.com/*user-name*/*repository-name***을 입력합니다.
   - 기본 소스 웹후크 이벤트에서:
     - 웹후크 - 선택 사항에서 코드 변경이 이 리포지토리로 푸시될 때마다 다시 빌드를 선택합니다.
     - 추가 구성 및 수동 생성 - 선택 사항을 선택하고 GitLab 콘솔에서 이 리포지토리에 대한 웹후크 수동 생성을 선택합니다.
3. 기본값으로 계속 진행한 다음 빌드 프로젝트 생성을 선택합니다. 나중에 사용할 페이로드 URL 및 보안 암호 값을 기록해 둡니다.
4. `https://gitlab.com/`*user-name*`/`*repository-name*`/-/hooks`에서 GitLab 콘솔을 열고 새 웹후크 추가를 선택합니다.
   - URL에 앞서 기록한 페이로드 URL 값을 입력합니다.
   - 보안 암호 토큰에 앞서 기록한 보안 암호 값을 입력합니다.
   - CodeBuild로 웹후크 페이로드를 전송할 개별 이벤트를 구성합니다. 트리거에서 푸시 이벤트, 병합 요청 이벤트. 릴리스 이벤트, 작업 이벤트 중에서 선택합니다. CodeBuild에서 지원하는 이벤트 유형에 대한 자세한 내용은 GitLab 웹후크 이벤트 섹션을 참조하세요.
5. 웹후크 추가를 선택합니다.

GitLab 웹후크 이벤트

웹후크 필터 그룹을 사용하여 어느 GitLab 웹후크 이벤트가 빌드를 트리거할지 지정할 수 있습니다. 예를 들어 특정 분기가 변경된 경우에만 빌드가 트리거되도록 지정할 수 있습니다.

하나 이상의 웹후크 필터 그룹을 생성하여 어느 웹후크 이벤트가 빌드를 트리거할지 지정할 수 있습니다. 필터 그룹이 true로 평가(그룹 내 모든 필터가 true로 평가)되면 빌드가 트리거됩니다. 필터 그룹을 생성할 때 다음을 지정합니다.

이벤트

GitLab의 경우 PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED, PULL_REQUEST_MERGED, PULL_REQUEST_REOPENED, PULL_REQUEST_CLOSED, RELEASED 및 WORKFLOW_JOB_QUEUED 이벤트 가운데 하나 이상을 선택할 수 있습니다.

webhook의 이벤트 유형은 X-GitLab-Event 필드의 헤더에 있습니다. 다음 표에서는 X-GitLab-Event 헤더 값이 이벤트 유형에 매핑되는 방법을 보여 줍니다. Merge Request Hook 웹후크 이벤트의 경우 페이로드의 object_atttributes.action에는 병합 요청 유형에 대한 추가 정보가 포함됩니다.

X-GitLab-Event 헤더 값	**object_atttributes .action**	이벤트 유형
Push Hook	해당 사항 없음	PUSH
Merge Request Hook	open	PULL_REQUEST_CREATED
Merge Request Hook	업데이트	PULL_REQUEST_UPDATED
Merge Request Hook	merge	PULL_REQUEST_MERGED
Merge Request Hook	다시 열기	PULL_REQUEST_REOPENED
Merge Request Hook	해지	PULL_REQUEST_CLOSED
Release Hook	생성, 업데이트	RELEASED
Job Hook	해당 사항 없음	WORKFLOW_JOB_QUEUED

PULL_REQUEST_MERGED의 경우 풀 요청이 스쿼시 전략에 병합되고 풀 요청 분기가 닫히면 원래의 풀 요청 커밋은 더 이상 존재하지 않게 됩니다. 이 경우 CODEBUILD_WEBHOOK_MERGE_COMMIT 환경 변수에는 스쿼시된 병합 커밋의 식별자가 포함됩니다.

하나 이상의 선택적 필터

정규식을 사용하여 필터를 지정합니다. 이벤트가 빌드를 트리거하려면 연결된 그룹 내의 필터가 모두 true로 평가되어야 합니다.

CodeBuild는 미리 보기, 뒤로 보기, 역참조 또는 원자 그룹을 지원하지 않는 RE2 정규 표현식 구문을 사용하여 필터 패턴을 평가합니다. 자세한 내용은 GitHub 웹 사이트의 RE2 구문 페이지를 참조하세요.

`ACTOR_ACCOUNT_ID`(콘솔의 `ACTOR_ID`)

GitLab 계정 ID가 정규식 패턴과 일치하면 웹후크 이벤트가 빌드를 트리거합니다. Webhook 필터 페이로드에 있는 `actor` 객체의 `account_id` 속성에 이 값이 표시됩니다.

`HEAD_REF`

헤드 참조가 정규식 패턴(예: `refs/heads/branch-name` 및 `refs/tags/tag-name`)과 일치하면 webhook 이벤트가 빌드를 트리거합니다. `HEAD_REF` 필터가 브랜치나 태그의 Git 참조 이름을 평가합니다. Webhook 페이로드의 `push` 객체에 있는 `new` 객체의 `name` 필드에 브랜치 또는 태그 이름이 표시됩니다. pull 요청 이벤트의 경우 webhook 페이로드의 `source` 객체에 있는 `branch` 객체의 `name` 필드에 브랜치 이름이 표시됩니다.

`BASE_REF`

베이스 참조가 정규식 패턴과 일치하면 webhook 이벤트가 빌드를 트리거합니다. `BASE_REF` 필터는 풀 요청 이벤트에서만 작동합니다(예: `refs/heads/branch-name`). `BASE_REF` 필터가 브랜치의 Git 참조 이름을 평가합니다. Webhook 페이로드의 `destination` 객체에 있는 `branch` 객체의 `name` 필드에 브랜치 이름이 표시됩니다.

`FILE_PATH`

변경된 파일의 경로가 정규식 패턴과 일치하면 webhook가 빌드를 트리거합니다.

> **Note**
>
> `FILE_PATH` 필터는 푸시 또는 풀 요청에서 처음 100개의 변경된 파일에 대해 평가됩니다. 필터가 의도한 대로 적용되도록 하려면 변경된 파일 수를이 한도 내에서 유지합니다. 풀 요청 이벤트의 경우 풀 요청 빌드 정책을 사용하여 빌드를 시작할 수 있는 풀 요청을 제어하는 것이 좋습니다.

`COMMIT_MESSAGE`

헤드 커밋 메시지가 정규식 패턴과 일치하면 webhook가 빌드를 트리거합니다.

`WORKFLOW_NAME`

워크플로 이름이 정규식 패턴과 일치하면 웹후크가 빌드를 트리거합니다.

> **Note**
>
> GitLab 리포지토리의 웹후크 설정에서 웹후크 페이로드를 찾을 수 있습니다.

주제

- GitLab 웹후크 이벤트 필터링(콘솔)
- GitLab 웹후크 이벤트 필터링(SDK)
- GitLab 웹후크 이벤트 필터링(CloudFormation)

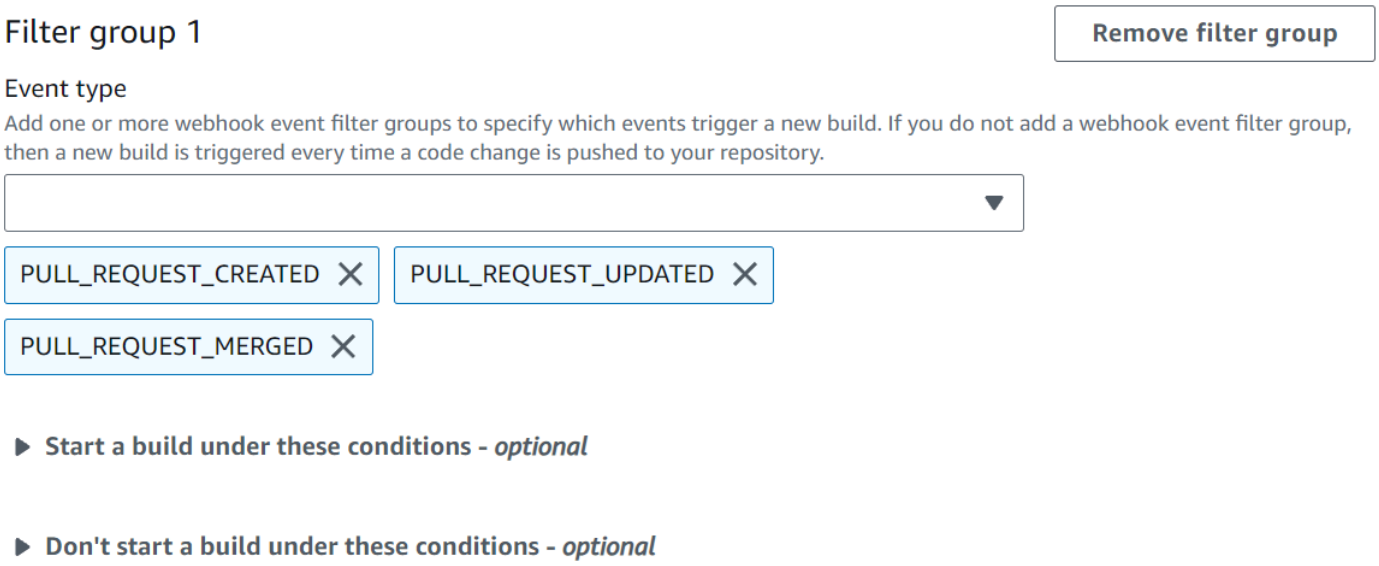
GitLab 웹후크 이벤트 필터링(콘솔)

다음 지침에 따라를 사용하여 웹후크 이벤트를 필터링 AWS Management Console 합니다. GitLab 웹후크 이벤트에 대한 자세한 내용은 GitLab 웹후크 이벤트 섹션을 참조하세요.

1. 프로젝트를 생성할 때 코드 변경이 이 리포지토리로 푸시될 때마다 다시 빌드를 선택합니다.
2. 이벤트 유형에서 하나 이상의 이벤트를 선택합니다.
3. 이벤트가 빌드를 트리거할 때를 필터링하려면 Start a build under these conditions(다음 조건에서 빌드를 시작)에서 하나 이상의 선택적 필터를 추가합니다.
4. 이벤트가 트리거되지 않을 때를 필터링하려면 Don't start a build under these conditions(다음 조건에서 빌드를 시작하지 않음)에서 하나 이상의 선택적 필터를 추가합니다.
5. Add filter group(필터 그룹 추가)를 선택하여 다른 필터 그룹을 추가합니다.

자세한 내용은 빌드 프로젝트 만들기(콘솔) 및 AWS CodeBuild API 참조의 WebHookFilter를 참조하세요.

이 예제에서는 Webhook 필터 그룹이 pull 요청에 대해서만 빌드를 트리거합니다.



두 필터 그룹을 사용하는 예제에서는 하나 또는 두 필터 그룹이 true로 평가되면 빌드가 트리거됩니다.

- 첫 번째 필터 그룹은 정규식 `^refs/heads/main$`와 일치하는 Git 참조 이름과 `^refs/heads/branch1!`와 일치하는 헤드 참조를 갖는 브랜치에서 생성 또는 업데이트된 pull 요청을 지정합니다.
- 두 번째 필터 그룹은 정규식 `^refs/heads/branch1$`와 일치하는 Git 참조 이름을 갖는 브랜치에서 push 요청을 지정합니다.

Webhook event filter groups

A build is triggered if any filter group evaluates to true, which occurs when all the filters in the group evaluate to true.

Filter group 1

Remove filter group

Event type

Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

PULL_REQUEST_CREATED ✕ PULL_REQUEST_UPDATED ✕

▼ **Start a build under these conditions - *optional***

Add filter

Filter 1

Type

HEAD_REF

Pattern

^refs/heads/branch1$

Remove

Filter 2

Type

BASE_REF

Pattern

^refs/heads/main

Remove

▶ **Don't start a build under these conditions - *optional***

Filter group 2

Remove filter group

Event type

Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

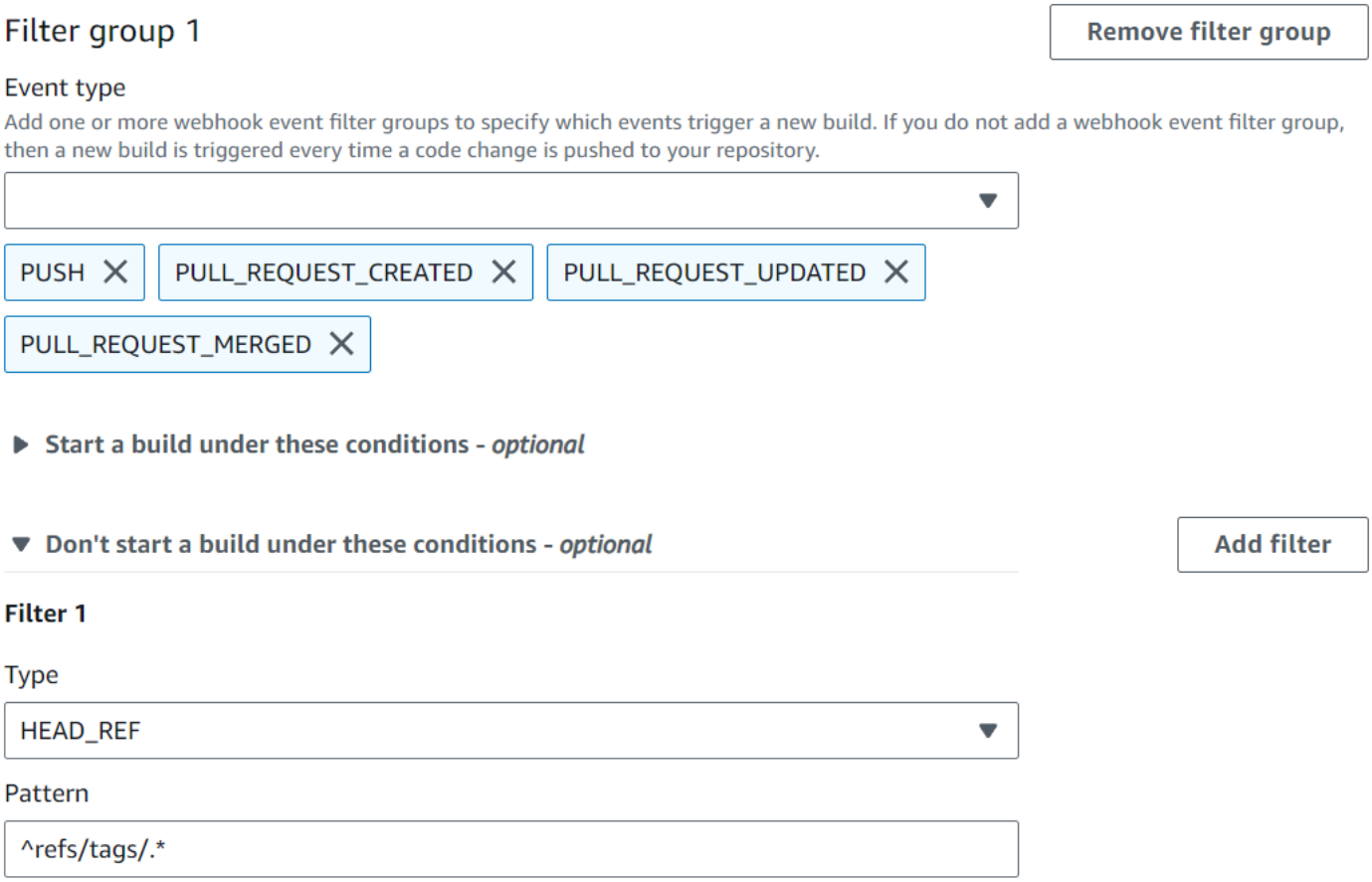
PUSH ✕

▼ **Start a build under these conditions - *optional***

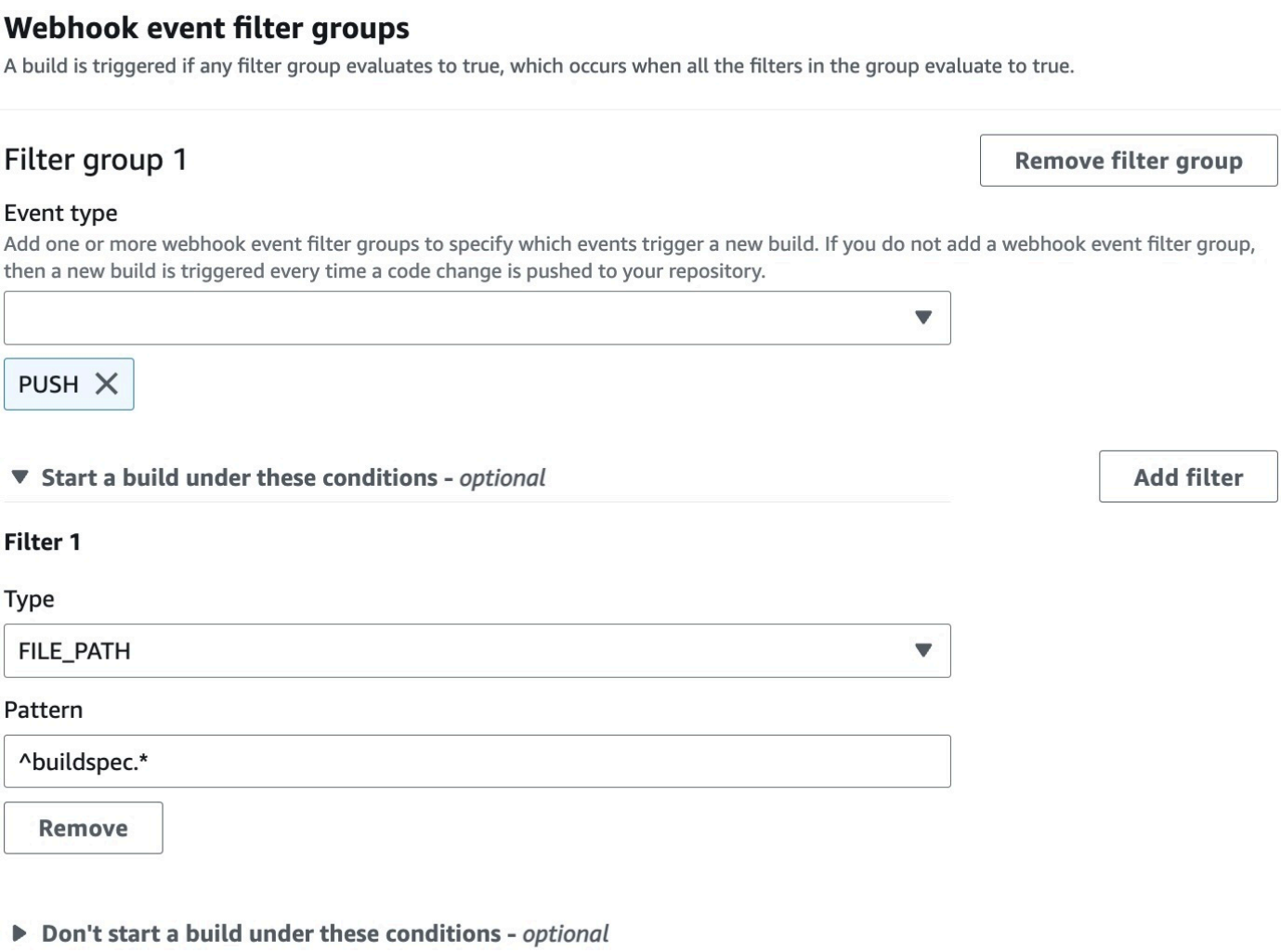
Add filter

Filter 1

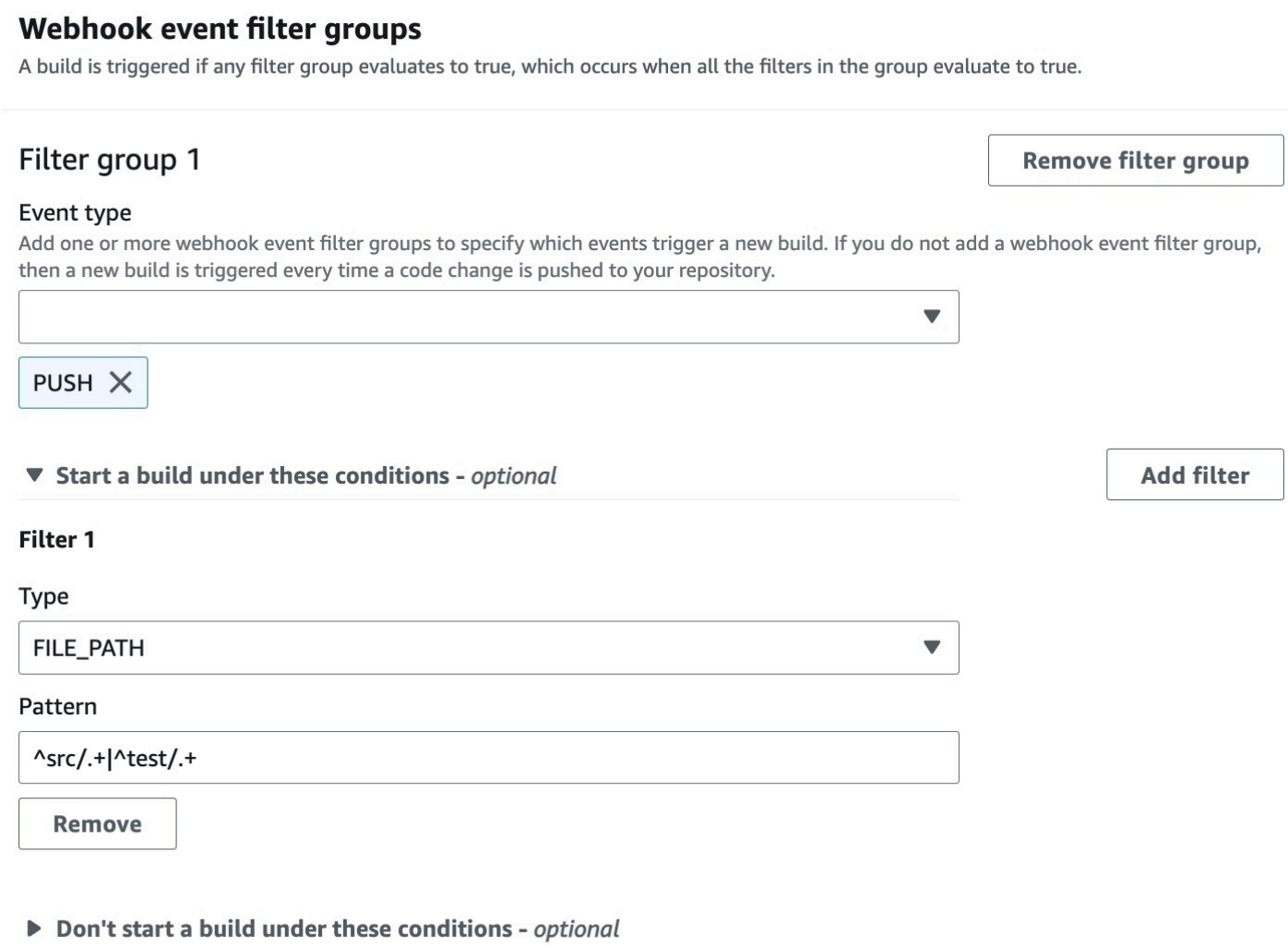
이 예제에서는 Webhook 필터 그룹이 태그 이벤트를 제외한 모든 요청에 대해 빌드를 트리거합니다.



이 예제에서는 Webhook 필터 그룹이 정규식 `^buildspec.*`와 일치하는 이름을 갖는 파일이 변경될 때만 빌드를 트리거합니다.

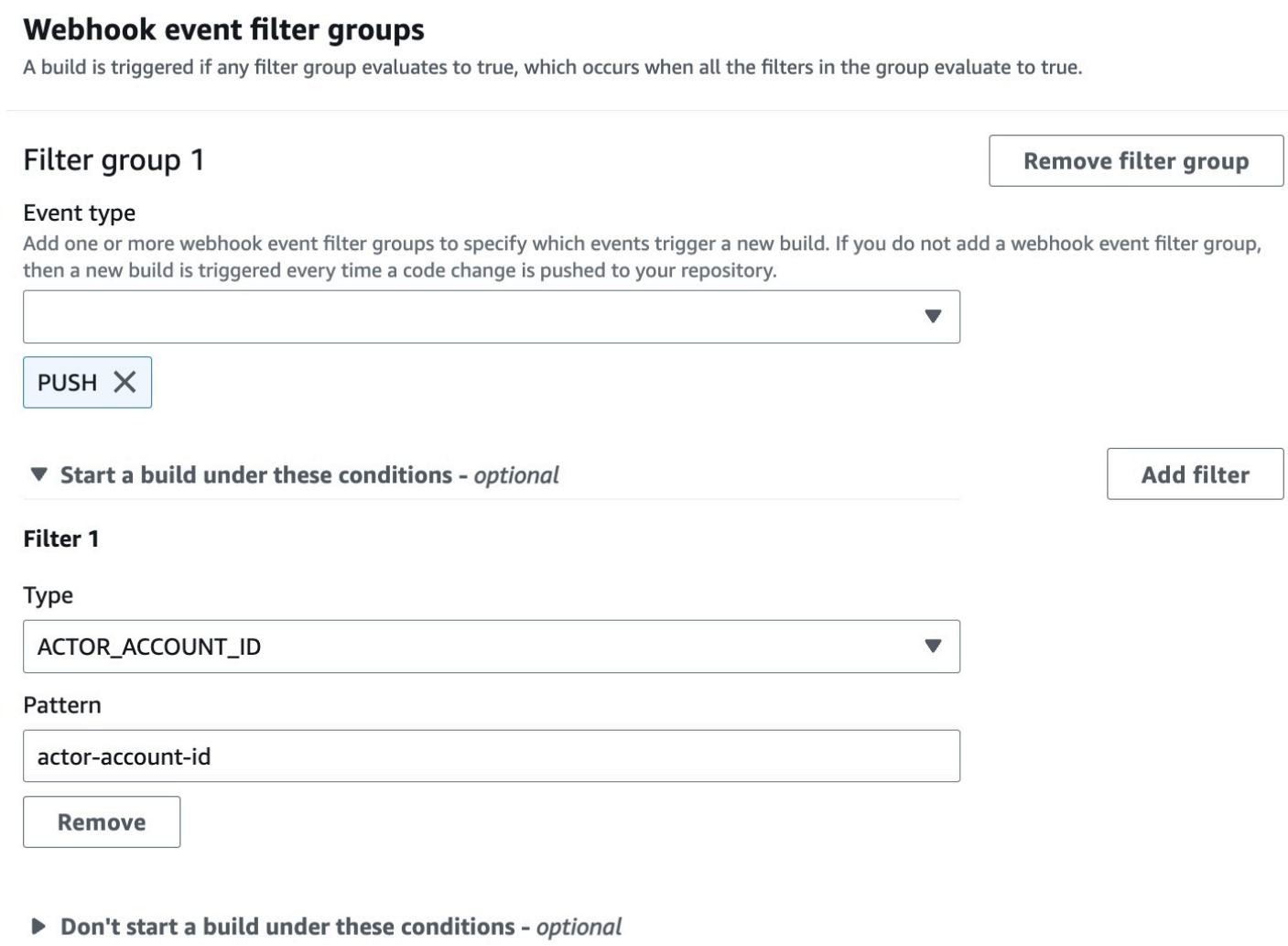


이 예제에서 Webhook 필터 그룹은 파일이 `src` 또는 `test` 폴더에서 변경된 경우에만 빌드를 트리거합니다.

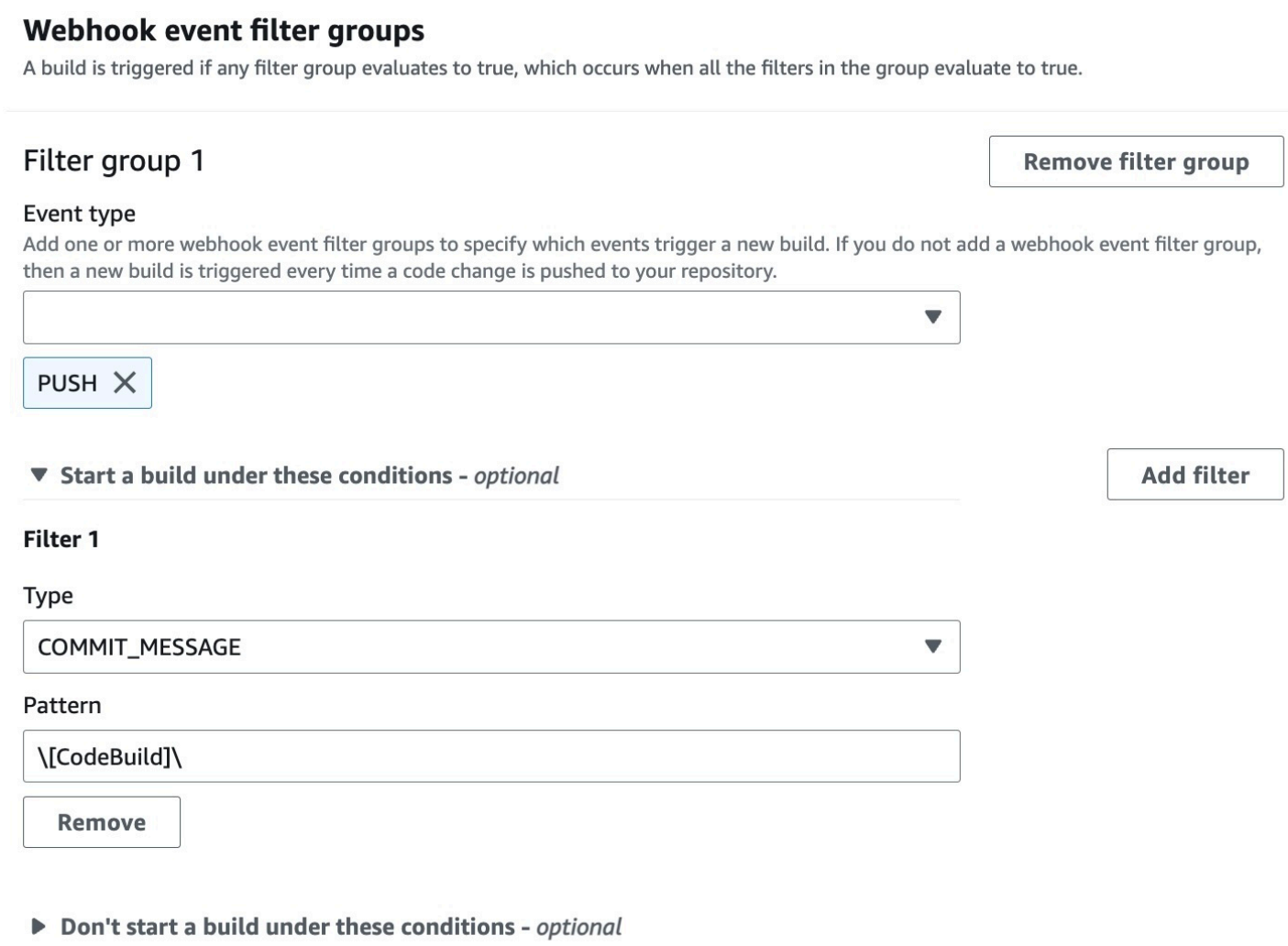


이 예제에서는 웹후크 필터 그룹이 정규식 `actor-account-id`와 일치하는 계정 ID가 없는 GitLab 사용자가 변경을 수행한 경우에만 빌드를 트리거합니다.

> **Note**
>
> GitLab 계정 ID를 확인하는 자세한 방법은 https://api.github.com/users/*user-name*을 참조하십시오. 여기서 *user-name*은 사용자의 GitLab 사용자 이름입니다.



이 예제에서 webhook 필터 그룹은 헤드 커밋 메시지가 정규식 `\[CodeBuild\]`와 일치할 때 푸시 이벤트에 대한 빌드를 트리거합니다.



GitLab 웹후크 이벤트 필터링(SDK)

AWS CodeBuild SDK를 사용하여 웹후크 이벤트를 필터링하려면 `CreateWebhook` 또는 `UpdateWebhook` API 메서드의 요청 구문에서 `filterGroups` 필드를 사용합니다. 자세한 내용은 CodeBuild API 참조의 WebHookFilter를 참조하세요.

GitLab 웹후크 이벤트에 대한 자세한 내용은 GitLab 웹후크 이벤트 섹션을 참조하세요.

pull 요청에 대해서만 빌드를 트리거하는 Webhook 필터를 생성하려면 요청 구문에 다음을 삽입합니다.

```
"filterGroups": [
  [
    {
      "type": "EVENT",
```

```
      "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED, PULL_REQUEST_MERGED"
    }
  ]
]
```

지정된 브랜치에 대해서만 빌드를 트리거하는 Webhook 필터를 생성하려면 `pattern` 파라미터를 사용하여 브랜치 이름을 필터링하는 정규식을 지정합니다. 두 필터 그룹을 사용하는 예제에서는 하나 또는 두 필터 그룹이 true로 평가되면 빌드가 트리거됩니다.

- 첫 번째 필터 그룹은 정규식 `^refs/heads/main$`와 일치하는 Git 참조 이름과 `^refs/heads/myBranch$`와 일치하는 헤드 참조를 갖는 브랜치에서 생성 또는 업데이트된 pull 요청을 지정합니다.
- 두 번째 필터 그룹은 정규식 `^refs/heads/myBranch$`와 일치하는 Git 참조 이름을 갖는 브랜치에서 push 요청을 지정합니다.

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED"
    },
    {
      "type": "HEAD_REF",
      "pattern": "^refs/heads/myBranch$"
    },
    {
      "type": "BASE_REF",
      "pattern": "^refs/heads/main$"
    }
  ],
  [
    {
      "type": "EVENT",
      "pattern": "PUSH"
    },
    {
      "type": "HEAD_REF",
      "pattern": "^refs/heads/myBranch$"
    }
  ]
```

```
]
```

`excludeMatchedPattern` 파라미터를 사용하여 빌드를 트리거하지 않을 이벤트를 지정할 수 있습니다. 이 예제에서는 태그 이벤트를 제외한 모든 요청에 대해 빌드가 트리거됩니다.

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
 PULL_REQUEST_MERGED"
    },
    {
      "type": "HEAD_REF",
      "pattern": "^refs/tags/.*",
      "excludeMatchedPattern": true
    }
  ]
]
```

계정 ID가 `actor-account-id`인 GitLab 사용자가 변경을 수행한 경우에만 빌드를 트리거하는 필터를 생성할 수 있습니다.

> **Note**
>
> GitLab 계정 ID를 확인하는 자세한 방법은 https://api.github.com/users/*user-name*을 참조하십시오. 여기서 *user-name*은 사용자의 GitLab 사용자 이름입니다.

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
 PULL_REQUEST_MERGED"
    },
    {
      "type": "ACTOR_ACCOUNT_ID",
      "pattern": "actor-account-id"
    }
  ]
```

```
]
```

pattern 인수의 정규식과 일치하는 이름을 갖는 파일이 변경되는 경우에만 빌드를 트리거하는 필터를 생성할 수 있습니다. 이 예제에서는 필터 그룹이 정규식 ^buildspec.*와 일치하는 이름을 갖는 파일이 변경될 때만 빌드가 트리거되도록 지정합니다.

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PUSH"
    },
    {
      "type": "FILE_PATH",
      "pattern": "^buildspec.*"
    }
  ]
]
```

이 예제에서 필터 그룹은 파일이 src 또는 test 폴더에서 변경될 때만 빌드가 트리거되도록 지정합니다.

```
"filterGroups": [
    [
        {
            "type": "EVENT",
            "pattern": "PUSH"
        },
        {
            "type": "FILE_PATH",
            "pattern": "^src/.+|^test/.+"
        }
    ]
]
```

헤드 커밋 메시지가 패턴 인수의 정규식과 일치할 때만 빌드를 트리거하는 필터를 생성할 수 있습니다. 이 예제에서 필터 그룹은 푸시 이벤트의 헤드 커밋 메시지가 정규식 \[CodeBuild\]와 일치할 때만 빌드가 트리거되도록 지정합니다.

```
  "filterGroups": [
    [
```

```
      {
        "type": "EVENT",
        "pattern": "PUSH"
      },
      {
        "type": "COMMIT_MESSAGE",
        "pattern": "\[CodeBuild\]"
      }
    ]
  ]
```

GitLab 웹후크 이벤트 필터링(CloudFormation)

CloudFormation 템플릿을 사용하여 웹후크 이벤트를 필터링하려면 AWS CodeBuild 프로젝트의 `FilterGroups` 속성을 사용합니다. GitLab 웹후크 이벤트에 대한 자세한 내용은 GitLab 웹후크 이벤트 섹션을 참조하세요.

다음 CloudFormation 템플릿의 YAML 형식 부분은 두 개의 필터 그룹을 생성합니다. 이들은 하나 또는 둘 모두 true로 평가되면 빌드를 트리거합니다.

- 첫 번째 필터 그룹은 계정 ID가 `12345`와 일치하지 않는 GitLab 사용자가 정규식 `^refs/heads/main$`와 일치하는 Git 참조 이름을 갖는 분기에서 생성 또는 업데이트한 pull 요청을 지정합니다.
- 두 번째 필터 그룹은 정규식 `^refs/heads/.*`와 일치하는 Git 참조 이름을 갖는 브랜치에서 생성되는 push 요청을 지정합니다.
- 세 번째 필터 그룹은 정규식 `\[CodeBuild\]`와 일치하는 헤드 커밋 메시지에서 push 요청을 지정합니다.
- 네 번째 필터 그룹은 정규식 `\[CI-CodeBuild\]`와 일치하는 워크플로 이름을 가진 GitHub Action 워크플로 작업 요청을 지정합니다.

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Name: MyProject
    ServiceRole: service-role
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      Type: LINUX_CONTAINER
      ComputeType: BUILD_GENERAL1_SMALL
```

```
      Image: aws/codebuild/standard:5.0
    Source:
      Type: GITLAB
      Location: source-location
    Triggers:
      Webhook: true
      FilterGroups:
        - - Type: EVENT
            Pattern: PULL_REQUEST_CREATED,PULL_REQUEST_UPDATED
          - Type: BASE_REF
            Pattern: ^refs/heads/main$
            ExcludeMatchedPattern: false
          - Type: ACTOR_ACCOUNT_ID
            Pattern: 12345
            ExcludeMatchedPattern: true
        - - Type: EVENT
            Pattern: PUSH
          - Type: HEAD_REF
            Pattern: ^refs/heads/.*
        - - Type: EVENT
            Pattern: PUSH
          - Type: COMMIT_MESSAGE
            Pattern: \[CodeBuild\]
        - - Type: EVENT
            Pattern: WORKFLOW_JOB_QUEUED
          - Type: WORKFLOW_NAME
            Pattern: \[CI-CodeBuild\]
```

Buildkite 수동 웹후크

현재 CodeBuild에서는 모든 Buildkite 웹후크를 수동으로 생성해야 합니다. CodeBuild는 직접 호출의 일부로 페이로드 URL을 반환하여 웹후크를 생성합니다. 이 URL은 Buildkite 내에서 수동으로 웹후크를 생성하는 데 사용할 수 있습니다.

다음 절차에 따라 Buildkite 수동 웹후크를 생성합니다.

웹후크를 사용하여 CodeBuild 프로젝트를 생성하려면

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 빌드 프로젝트를 생성합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 및 빌드 실행(콘솔) 섹션을 참조하세요.
3. 프로젝트 구성에서 러너 프로젝트를 선택합니다.

러너에서:

- 러너 공급자에서 Buildkite를 선택합니다.
- Buildkite 에이전트 토큰에서 보안 암호 생성 페이지를 사용하여 새 에이전트 토큰 생성을 선택합니다. AWS Secrets Manager에서 위에서 생성한 Buildkite 에이전트 토큰과 동일한 보안 암호 값을 사용하여 새 보안 암호를 생성하라는 메시지가 표시됩니다.
- (선택 사항) 작업에 CodeBuild 관리형 자격 증명을 사용하려면 Buildkite 소스 자격 증명 옵션에서 작업의 소스 리포지토리 공급자를 선택하고 자격 증명이 계정에 대해 구성되어 있는지 확인합니다. 또한, Buildkite 파이프라인이 HTTPS를 사용한 체크아웃을 사용하는지 확인합니다.

4. • 환경에서 다음과 같이 합니다.
   - 지원되는 환경 이미지와 컴퓨팅을 선택합니다. GitHub Action 워크플로 YAML의 레이블을 사용하여 이미지 및 인스턴스 설정을 재정의할 수 있는 옵션이 있습니다. 자세한 내용은 2단계: GitHub Action 워크플로 YAML 업데이트 섹션을 참조하세요.
   - Buildspec에서 다음과 같이 합니다.
     - `buildspec-override:true`가 레이블로 추가되지 않는 한 buildspec은 무시됩니다. 대신 CodeBuild는 자체 호스팅 러너를 설정하는 명령을 사용하도록 이를 재정의합니다.
5. 기본값으로 계속 진행한 다음 빌드 프로젝트 생성을 선택합니다.
6. 웹후크 생성 팝업에서 페이로드 URL 및 보안 암호 값을 저장합니다. 팝업의 지침에 따라 새 Buildkite 조직 웹후크를 생성합니다.

풀 요청 설명 승인

CodeBuild는 풀 요청에 의해 트리거되는 빌드에 대한 추가 제어를 제공하는 풀 요청 빌드 정책을 지원합니다. 변경 사항을 검토할 수 있을 때까지 알 수 없는 사용자의 풀 요청을 자동으로 빌드하지 않을 수 있습니다. 이 기능을 사용하면 팀원 중 한 명이 먼저 코드를 검토한 다음 파이프라인을 실행하도록 요구할 수 있습니다. 이는 일반적으로 알 수 없는 기여자가 제출한 코드를 빌드할 때 보안 조치로 사용됩니다.

풀 요청 빌드 정책을 사용하면 CodeBuild가 기여자의 권한 및 승인 상태에 따라 풀 요청에 대한 빌드를 트리거하는 시기를 제어할 수 있습니다. 이는 퍼블릭 리포지토리 또는 외부 공동 작업자의 기여를 수락하는 리포지토리에 특히 중요합니다.

이 기능을 활성화하면 다음 중 하나에 해당하는 경우에만 풀 요청에 대해 빌드가 트리거됩니다.

- 풀 요청이 신뢰할 수 있는 기여자에 의해 생성됩니다.

- 신뢰할 수 있는 기여자가 특정 설명을 게시하여 풀 요청을 승인합니다.

작동 방식

신뢰할 수 있는 기여자

신뢰할 수 있는 기여자는 풀 요청 기반 정책에서 소스 제어 시스템의 현재 역할이 승인자 역할로 설정된 사용자입니다. 신뢰할 수 있는 기여자가 풀 요청을 생성하면 CodeBuild는 빌드를 자동으로 트리거하여 현재 동작을 유지합니다.

신뢰할 수 없는 기여자

신뢰할 수 없는 기여자는 승인자 역할 목록에 역할이 설정되지 않은 사용자입니다. 신뢰할 수 없는 기여자가 풀 요청을 생성하는 경우:

1. CodeBuild는 "빌드를 시작하는 데 필요한 풀 요청 승인"이라는 메시지와 함께 빌드 상태를 "실패"로 표시합니다.
2. 신뢰할 수 있는 기여자는 변경 사항을 검토하고 `/codebuild_run(<SHA_OF_THE_LATEST_COMMIT>)`과 함께 설명을 게시하여 빌드를 트리거해야 합니다. 예를 들어 `/codebuild_run(046e8b67481d53bdc86c3f6affdd5d1afae6d369)`입니다.
3. CodeBuild는 해설자의 권한을 검증하고 승인된 경우 빌드를 트리거합니다.
4. 빌드 결과는 풀 요청 페이지에 다시 보고됩니다.

설명 승인 구문

신뢰할 수 있는 기여자는 다음 설명 형식을 사용하여 빌드를 승인할 수 있습니다.

- `/codebuild_run(046e8b67481d53bdc86c3f6affdd5d1afae6d369)` - 지정된 커밋 SHA에서 빌드를 트리거합니다.

구성

기본 동작

풀 요청 빌드 정책은 새로 생성된 모든 CodeBuild 프로젝트에 대해 기본적으로 활성화됩니다.

API 파라미터

풀 요청 빌드 정책은 다음 작업의 `PullRequestBuildPolicy` 파라미터를 사용하여 구성됩니다.

- CreateWebhook
- UpdateWebhook

PullRequestBuildPolicy 구조

```
{
    "requiresCommentApproval": "string",
    "approverRoles": ["string", ...]
}
```

requiresCommentApproval

풀 요청에 대한 빌드를 트리거하기 전에 설명 기반 승인이 필요한 시기를 지정합니다. 이 설정은 빌드가 자동으로 실행되는지, 아니면 설명을 통한 명시적 승인이 필요한지를 결정합니다.

유형: 문자열

유효한 값:

- DISABLED - 설명 승인 없이 트리거를 자동으로 빌드합니다.
- FORK_PULL_REQUESTS - 포크된 리포지토리의 풀 요청만 설명 승인이 필요합니다(기여자가 승인자 역할 중 하나가 아닌 경우).
- ALL_PULL_REQUESTS - 모든 풀 요청은 빌드를 실행하기 전에 설명 승인이 필요합니다(기여자가 승인자 역할 중 하나가 아닌 경우). 이것이 기본값입니다.

approverRoles

설명 승인이 필요한 경우 풀 요청 빌드에 대한 승인 권한이 있는 리포지토리 역할 목록입니다. 이러한 역할을 가진 사용자만 유효한 설명 승인을 제공할 수 있습니다. 풀 요청 기여자가 이러한 역할 중 하나인 경우 풀 요청 빌드가 자동으로 트리거됩니다.

유형: 문자열 배열

GitHub 프로젝트의 유효한 값(값은 GitHub 역할에 매핑됨):

- GITHUB_ADMIN - 리포지토리 관리자
- GITHUB_MAINTAIN - 리포지토리 유지 관리자
- GITHUB_WRITE - 쓰기 권한이 있는 사용자
- GITHUB_TRIAGE - 분류 권한이 있는 사용자
- GITHUB_READ - 읽기 권한이 있는 사용자

- 기본값: ["GITHUB_ADMIN", "GITHUB_MAINTAIN", "GITHUB_WRITE"]

GitLab 프로젝트의 유효한 값(값은 GitLab 역할에 매핑됨):

- GITLAB_OWNER - 리포지토리 소유자
- GITLAB_MAINTAINER - 리포지토리 유지 관리자
- GITLAB_DEVELOPER - 개발자 권한이 있는 사용자
- GITLAB_REPORTER - 보고자 권한이 있는 사용자
- GITLAB_PLANNER - 기획자 권한이 있는 사용자
- GITLAB_GUEST - 게스트 권한이 있는 사용자
- 기본값: ["GITLAB_OWNER", "GITLAB_MAINTAINER", "GITLAB_DEVELOPER"]

Bitbucket 프로젝트의 유효한 값(값은 Bitbucket 역할에 매핑됨):

- BITBUCKET_ADMIN - 리포지토리 관리자
- BITBUCKET_WRITE - 쓰기 권한이 있는 사용자
- BITBUCKET_READ - 읽기 권한이 있는 사용자
- 기본값: ["BITBUCKET_ADMIN", "BITBUCKET_WRITE"]

사용자 지정 GitHub Enterprise 역할

CodeBuild는 사용자 지정 역할에 부여된 가장 높은 권한 수준을 기반으로 사용자 지정 GitHub Enterprise(GHE) 리포지토리 역할을 표준 GITHUB_* 값에 매핑합니다. CodeBuild는 가장 높은 권한에서 가장 낮은 권한으로 권한을 평가하며, 활성화된 첫 번째 권한에 따라 매핑된 역할이 결정됩니다.

- admin → GITHUB_ADMIN
- maintain → GITHUB_MAINTAIN
- push (쓰기) → GITHUB_WRITE
- triage → GITHUB_TRIAGE
- pull (읽기) → GITHUB_READ

예를 들어가 보다 높은 권한을 push 가지고 GITHUB_WRITE 있기 때문에 push 및 triage 권한을 모두 부여하는 사용자 지정 역할은에 매핑됩니다triage.

사용자 지정 역할에 인식된 권한(관리자, 유지 관리, 푸시, 분류, 모두 가져오기 비활성화됨)이 없는 경우 사용자는 승인자 역할과 일치하지 않으며 신뢰할 수 없는 기여자로 처리됩니다.

표준 GitHub 역할(관리자, 유지 관리, 쓰기, 분류, 읽기)은 권한 기반 확인 없이 해당 `GITHUB_*` 값에 직접 매핑됩니다.

예제

모든 풀 요청에 대한 설명 승인 활성화

AWS CodeBuild SDK를 사용하여 웹후크에 대한 풀 요청 빌드 정책을 활성화하거나 비활성화하려면 `CreateWebhook` 또는 `UpdateWebhook` API 메서드의 요청 구문에서 `pullRequestBuildPolicy` 필드를 사용합니다. 자세한 내용은 CodeBuild API 참조의 WebHookFilter를 참조하세요.

GitHub 역할 관리, 유지 관리 및 쓰기를 사용하는 사용자는 신뢰할 수 있는 기여자로 처리됩니다.

```
"pullRequestBuildPolicy": {
    "requiresCommentApproval": "ALL_PULL_REQUESTS",
    "approverRoles": ["GITHUB_ADMIN", "GITHUB_MAINTAIN", "GITHUB_WRITE"]
}
```

리포지토리 관리자 및 유지 관리자에 대해서만 설명 승인 활성화

GitHub 역할 관리 및 유지 관리가 있는 사용자는 신뢰할 수 있는 기여자로 처리됩니다.

```
"pullRequestBuildPolicy": {
    "requiresCommentApproval": "FORK_PULL_REQUESTS",
    "approverRoles": ["GITHUB_ADMIN", "GITHUB_MAINTAIN"]
}
```

설명 승인 비활성화

```
"pullRequestBuildPolicy": {
    "requiresCommentApproval": "DISABLED"
}
```

AWS CloudFormation

AWS CloudFormation 템플릿을 사용하여 웹후크에 대한 풀 요청 빌드 정책을 활성화하거나 비활성화하려면 PullRequestBuildPolicy 속성을 사용합니다. AWS CloudFormation 템플릿의 다음 YAML 형식

부분은 모든 풀 요청에 대해 풀 요청 빌드 정책이 활성화된 웹후크가 있는 프로젝트를 생성합니다. 유지 관리 및 관리자 역할은 승인자로 지정됩니다.

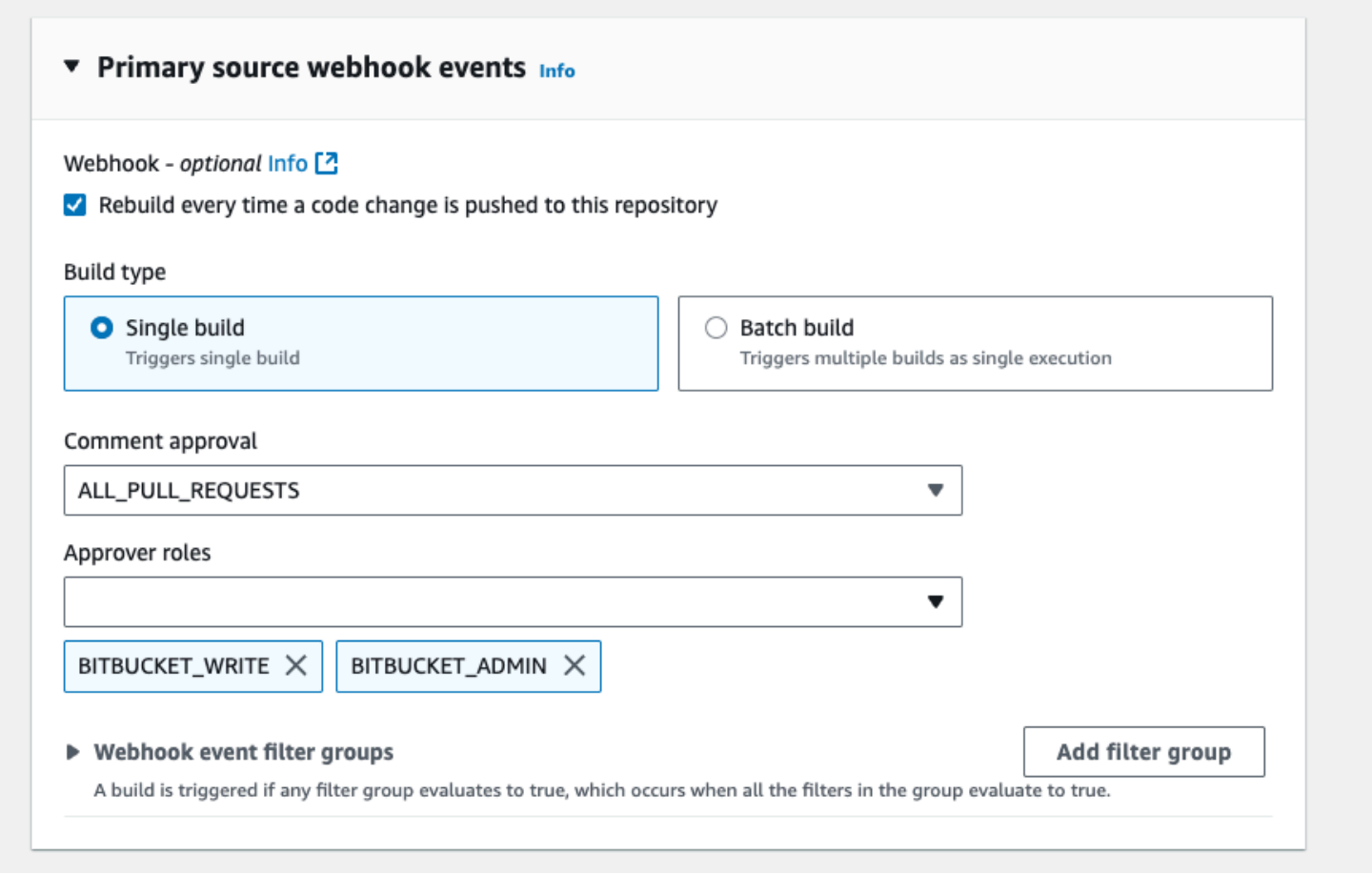
```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Name: MyProject
    ServiceRole: service-role
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      Type: LINUX_CONTAINER
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:5.0
    Source:
      Type: BITBUCKET
      Location: source-location
    Triggers:
      Webhook: true
      FilterGroups:
        - - Type: EVENT
            Pattern: PULL_REQUEST_CREATED,PULL_REQUEST_UPDATED
          - Type: BASE_REF
            Pattern: ^refs/heads/main$
            ExcludeMatchedPattern: false
      PullRequestBuildPolicy:
        RequiresCommentApproval: ALL_PULL_REQUESTS
        ApproverRoles:
          - GITHUB_MAINTAIN
          - GITHUB_ADMIN
```

콘솔 구성

AWS 관리 콘솔을 사용하여 웹후크 이벤트를 필터링하려면:

1. 설명 승인에서 모든 풀 요청(ALL_PULL_REQUEST)에 대해 비활성화 또는 활성화를 선택하거나 포크의 풀 요청(FORK_PULL_REQUEST)에 대해서만 비활성화 또는 활성화를 선택합니다.
2. 승인자 역할에서 설명 승인이 필요할 때 풀 요청 빌드에 대한 승인 권한이 있는 리포지토리 역할을 선택합니다.

자세한 내용은 CodeBuild API 참조의 빌드 프로젝트 만들기(콘솔) 및 WebhookFilter를 참조하세요.



에서 빌드 프로젝트의 세부 정보 보기 AWS CodeBuild

AWS CodeBuild 콘솔 AWS CLI, 또는 AWS SDKs를 사용하여 CodeBuild에서 빌드 프로젝트의 세부 정보를 볼 수 있습니다.

주제

- 빌드 프로젝트 세부 정보 보기(콘솔)
- 빌드 프로젝트 세부 정보 보기(AWS CLI)
- 빌드 프로젝트의 세부 정보(AWS SDKs) 보기

빌드 프로젝트 세부 정보 보기(콘솔)

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 프로젝트 빌드를 선택합니다.

> **Note**
>
> 기본적으로 가장 최근의 빌드 프로젝트 10개만 표시됩니다. 더 많은 빌드 프로젝트를 보려면 기어 아이콘을 선택하고 페이지당 프로젝트 수에서 다른 값을 선택하거나 뒤로 및 앞으로 화살표를 사용합니다.

3. 빌드 프로젝트 목록의 이름 열에서 빌드 프로젝트에 대한 링크를 선택합니다.
4. 프로젝트 빌드: ***project-name*** 페이지에서 빌드 세부 정보를 선택합니다.

빌드 프로젝트 세부 정보 보기(AWS CLI)

batch-get-projects 명령을 실행합니다.

```
aws codebuild batch-get-projects --names names
```

이전 명령에서 다음 자리표시자를 바꿉니다.

- *names*: 세부 정보를 볼 수 있는 하나 이상의 빌드 프로젝트 이름을 나타내는 데 사용되는 필수 문자열입니다. 둘 이상의 빌드 프로젝트를 지정하려면 각 빌드 프로젝트 이름을 공백을 사용하여 구분해야 합니다. 최대 100개의 빌드 프로젝트 이름을 지정할 수 있습니다. 빌드 프로젝트 목록을 가져오려면 빌드 프로젝트 이름 목록 보기(AWS CLI) 단원을 참조하십시오.

예를 들면 다음 명령을 실행하는 경우

```
aws codebuild batch-get-projects --names codebuild-demo-project codebuild-demo-project2
 my-other-demo-project
```

다음과 비슷한 결과가 출력에 나타납니다. 줄임표(`...`)는 간결성을 위해 생략된 데이터를 나타내는 데 사용됩니다.

```
{
  "projectsNotFound": [
    "my-other-demo-project"
  ],
  "projects": [
    {
      ...
```

```
      "name": codebuild-demo-project,
      ...
    },
    {
      ...
      "name": codebuild-demo-project2",
      ...
    }
  ]
}
```

위 출력에서 `projectsNotFound` 배열에는 지정되었지만 찾을 수 없는 모든 빌드 프로젝트 이름이 나열됩니다. `projects` 배열에는 정보가 발견된 각 빌드 프로젝트의 세부 정보가 나열됩니다. 간결성을 위해 이전 출력에서 빌드 프로젝트 세부 정보가 생략되었습니다. 자세한 내용은 빌드 프로젝트 생성(AWS CLI)의 출력을 참조하세요.

batch-get-projects 명령은 특정 속성 값에 대한 필터링을 지원하지 않지만 프로젝트의 속성을 열거하는 스크립트를 작성할 수 있습니다. 예를 들어, 다음 Linux 쉘 스크립트는 현재 계정의 현재 리전에 있는 프로젝트를 열거하고 각 프로젝트에서 사용되는 이미지를 인쇄합니다.

```
#!/usr/bin/sh

# This script enumerates all of the projects for the current account
# in the current region and prints out the image that each project is using.

imageName=""

function getImageName(){
  local environmentValues=(${1//$'\t'/ })
  imageName=${environmentValues[1]}
}

function processProjectInfo() {
  local projectInfo=$1

  while IFS=$'\t' read -r section value; do
    if [[ "$section" == *"ENVIRONMENT"* ]]; then
      getImageName "$value"
    fi
  done <<< "$projectInfo"
}

# Get the list of projects.
```

```
projectList=$(aws codebuild list-projects --output=text)

for projectName in $projectList
do
  if [[ "$projectName" != *"PROJECTS"* ]]; then
    echo "================================================"

    # Get the detailed information for the project.
    projectInfo=$(aws codebuild batch-get-projects --output=text --names
 "$projectName")

    processProjectInfo "$projectInfo"

    printf 'Project "%s" has image "%s"\n' "$projectName" "$imageName"
  fi
done
```

와 AWS CLI 함께를 사용하는 방법에 대한 자세한 내용은 단원을 AWS CodeBuild참조하십시오명령줄 참조.

빌드 프로젝트의 세부 정보(AWS SDKs) 보기

를 SDK와 AWS CodeBuild 함께 사용하는 방법에 대한 자세한 내용은 섹션을 참조하세요AWS SDKs 및 도구 참조. AWS SDKs

에서 빌드 프로젝트 이름 보기 AWS CodeBuild

AWS CodeBuild 콘솔 AWS CLI, 또는 AWS SDKs 사용하여 CodeBuild의 빌드 프로젝트 목록을 볼 수 있습니다.

주제

- 빌드 프로젝트 이름 목록 보기(콘솔)
- 빌드 프로젝트 이름 목록 보기(AWS CLI)
- 빌드 프로젝트 이름(AWS SDKs) 목록 보기

빌드 프로젝트 이름 목록 보기(콘솔)

콘솔에서 AWS 리전의 빌드 프로젝트 목록을 볼 수 있습니다. 정보에는 이름, 소스 공급자, 리포지토리, 최신 빌드 상태 및 설명(있는 경우)이 포함됩니다.

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 프로젝트 빌드를 선택합니다.

> **Note**
>
> 기본적으로 가장 최근의 빌드 프로젝트 10개만 표시됩니다. 더 많은 빌드 프로젝트를 보려면 기어 아이콘을 선택하고 페이지당 프로젝트 수에서 다른 값을 선택하거나 뒤로 및 앞으로 화살표를 사용합니다.

빌드 프로젝트 이름 목록 보기(AWS CLI)

list-projects 명령을 실행합니다.

```
aws codebuild list-projects --sort-by sort-by --sort-order sort-order --next-
token next-token
```

이전 명령에서 다음 자리표시자를 바꿉니다.

- *sort-by*: 빌드 프로젝트 이름을 나열하는 데 사용할 기준을 나타내는 데 사용되는 선택적 문자열입니다. 유효한 값으로는 다음이 포함됩니다.
  - CREATED_TIME: 각 빌드 프로젝트가 생성된 시기를 기준으로 빌드 프로젝트 이름을 나열합니다.
  - LAST_MODIFIED_TIME: 각 빌드 프로젝트에 대한 정보가 마지막으로 변경된 시기를 기준으로 빌드 프로젝트 이름을 나열합니다.
  - NAME: 각 빌드 프로젝트의 이름을 기준으로 빌드 프로젝트 이름을 나열합니다.
- *sort-order*: *sort-by*에 따라 빌드 프로젝트를 나열하는 순서를 나타내는 데 사용되는 선택적 문자열입니다. 유효한 값에는 ASCENDING 및 DESCENDING(이)가 있습니다.
- *next-token*: 선택적 문자열입니다. 이전 실행 중에 목록에 100개가 넘는 항목이 있는 경우 next token이라는 고유한 문자열과 함께 처음 100개 항목만 반환됩니다. 목록에서 다음 항목 배치를 가져오려면 호출에 다음 토큰을 추가하여 이 명령을 다시 실행합니다. 목록에 있는 모든 항목을 가져오려면 다음 토큰이 더 이상 반환되지 않을 때까지 다음 토큰마다 이 명령을 계속 실행합니다.

예를 들면 다음 명령을 실행하는 경우

```
aws codebuild list-projects --sort-by NAME --sort-order ASCENDING
```

다음과 비슷한 결과가 출력에 나타납니다.

```
{
  "nextToken": "Ci33ACF6...The full token has been omitted for brevity...U+AkMx8=",
  "projects": [
    "codebuild-demo-project",
    "codebuild-demo-project2",
    ... The full list of build project names has been omitted for brevity ...
    "codebuild-demo-project99"
  ]
}
```

이 명령을 다시 실행하는 경우:

```
aws codebuild list-projects  --sort-by NAME --sort-order ASCENDING --next-token
 Ci33ACF6...The full token has been omitted for brevity...U+AkMx8=
```

다음과 비슷한 결과가 출력에 나타납니다.

```
{
  "projects": [
    "codebuild-demo-project100",
    "codebuild-demo-project101",
    ... The full list of build project names has been omitted for brevity ...
    "codebuild-demo-project122"
  ]
}
```

빌드 프로젝트 이름(AWS SDKs) 목록 보기

를 SDK와 AWS CodeBuild 함께 사용하는 방법에 대한 자세한 내용은 섹션을 참조하세요AWS SDKs 및 도구 참조. AWS SDKs

에 빌드 AWS CodeBuild

빌드는 입력 아티팩트 세트(예: Java 클래스 파일 모음)를 기반으로 출력 아티팩트(예: JAR 파일)를 생성하기 AWS CodeBuild 위해에서 수행하는 작업 세트를 나타냅니다.

빌드를 여러 개 실행할 경우 다음 규칙이 적용됩니다.

- 가능한 경우 동시에 빌드가 실행됩니다. 동시에 실행할 수 있는 최대 빌드 수는 다를 수 있습니다. 자세한 내용은 에 대한 할당량 AWS CodeBuild 단원을 참조하십시오.
- 빌드 프로젝트에 동시 빌드 제한이 설정되어 있는 경우 실행 중인 빌드 수가 프로젝트의 동시 빌드 제한에 도달하면 빌드에서 오류가 반환됩니다. 자세한 내용은 동시 빌드 제한 활성화를 참조하세요.
- 빌드 프로젝트에 동시 빌드 제한이 설정되어 있지 않은 경우 실행 중인 빌드 수가 플랫폼 및 컴퓨팅 유형의 동시 빌드 제한에 도달하면 빌드가 대기열에 추가됩니다. 대기열의 최대 빌드 수는 동시 빌드 수 한도의 다섯 배입니다. 자세한 내용은 에 대한 할당량 AWS CodeBuild 단원을 참조하십시오.

  제한 시간 값에 지정한 시간(분)이 지난 후 시작되지 않은 대기열의 빌드는 대기열에서 삭제됩니다. 기본 제한 시간 값은 8분입니다. 빌드를 실행할 때 빌드 대기열 제한 시간을 5분에서 8시간까지 원하는 대로 지정할 수 있습니다. 자세한 내용은 수동으로 AWS CodeBuild 빌드 실행 단원을 참조하십시오.

  대기열에 추가된 빌드가 시작되는 순서는 예측할 수 없습니다.

> **Note**
>
> 1년치 빌드의 기록에 액세스할 수 있습니다.

빌드 작업을 수행할 때 이 작업을 수행할 수 있습니다.

주제

- 수동으로 AWS CodeBuild 빌드 실행
- AWS Lambda 컴퓨팅 기반 빌드 실행
- 예약 용량 플릿에서 빌드 실행
- 배치로 빌드 실행
- 배치 빌드에서 병렬 테스트 실행

- 성능을 개선하기 위한 캐시 빌드
- 에서 빌드 디버그 AWS CodeBuild
- 에서 빌드 삭제 AWS CodeBuild
- 에서 수동으로 빌드 재시도 AWS CodeBuild
- 에서 자동으로 빌드 재시도 AWS CodeBuild
- 에서 빌드 중지 AWS CodeBuild
- 에서 배치 빌드 중지 AWS CodeBuild
- 자동으로 AWS CodeBuild 빌드 트리거
- 에서 빌드 세부 정보 보기 AWS CodeBuild
- 에서 빌드 IDs 목록 보기 AWS CodeBuild
- 에서 빌드 프로젝트의 빌드 IDs 목록 보기 AWS CodeBuild

수동으로 AWS CodeBuild 빌드 실행

AWS CodeBuild 콘솔 AWS CLI, 또는 AWS SDKs를 사용하여 CodeBuild에서 빌드를 실행할 수 있습니다.

주제

- AWS CodeBuild 에이전트를 사용하여 로컬에서 빌드 실행
- 빌드 실행(콘솔)
- 빌드 실행(AWS CLI)
- 배치 빌드 실행(AWS CLI)
- 빌드 실행 자동 시작(AWS CLI)
- 빌드 실행 자동 중지(AWS CLI)
- 빌드 실행(AWS SDKs)

AWS CodeBuild 에이전트를 사용하여 로컬에서 빌드 실행

AWS CodeBuild 에이전트를 사용하여 로컬 시스템에서 CodeBuild 빌드를 실행할 수 있습니다. x86_64 및 ARM 플랫폼에 사용할 수 있는 에이전트가 있습니다.

새 버전의 에이전트가 게시되는 시기를 알 수 있도록 알림을 구독할 수도 있습니다.

사전 조건

시작하려면 다음을 수행해야 합니다.

- 로컬 시스템에 Git를 설치합니다.
- 로컬 컴퓨터에 Docker를 설치하고 설정합니다.

빌드 이미지 설정

에이전트를 처음 실행할 때 또는 이미지가 변경된 경우에만 빌드 이미지를 설정해야 합니다.

빌드 이미지를 설정하려면

1. 큐레이션된 Amazon Linux 2 이미지를 사용하려면 다음 명령을 사용하여 https://gallery.ecr.aws/codebuild/amazonlinux-x86_64-standard CodeBuild 퍼블릭 Amazon ECR 리포지토리에서 가져올 수 있습니다.

   ```
   $ docker pull public.ecr.aws/codebuild/amazonlinux-x86_64-standard:4.0
   ```

   다른 Linux 이미지를 사용하려면 다음 단계를 수행합니다.

   a. CodeBuild 이미지 리포지토리를 복제합니다.

      ```
      $ git clone https://github.com/aws/aws-codebuild-docker-images.git
      ```

   b. image 디렉터리로 변경합니다. 이 예에서는 `aws/codebuild/standard:5.0` 이미지를 사용합니다.

      ```
      $ cd aws-codebuild-docker-images/ubuntu/standard/5.0
      ```

   c. 이미지를 빌드합니다. 이 작업은 몇 분 정도 걸릴 수 있습니다.

      ```
      $ docker build -t aws/codebuild/standard:5.0 .
      ```

2. CodeBuild 에이전트를 다운로드합니다.

   x86_64 버전의 에이전트를 다운로드하려면 다음 명령을 실행합니다.

   ```
   $ docker pull public.ecr.aws/codebuild/local-builds:latest
   ```

ARM 버전의 에이전트를 다운로드하려면 다음 명령을 실행합니다.

```
$ docker pull public.ecr.aws/codebuild/local-builds:aarch64
```

3. CodeBuild 에이전트는 https://gallery.ecr.aws/codebuild/local-builds에서 사용할 수 있습니다.

   에이전트의 x86_64 버전에 대한 보안 해시 알고리즘(SHA) 서명은 다음과 같습니다.

   ```
   sha256:ccb19bdd7af94e4dc761e4c58c267e9455c28ec68d938086b4dc1cf8fe6b0940
   ```

   에이전트의 ARM 버전에 대한 SHA 서명은 다음과 같습니다.

   ```
   sha256:7d7b5d35d2ac4e062ae7ba8c662ffed15229a52d09bd0d664a7816c439679192
   ```

   SHA를 사용하여 에이전트 버전을 확인할 수 있습니다. 에이전트의 SHA 서명을 보려면 다음 명령을 실행하고 `RepoDigests`에서 SHA를 찾습니다.

   ```
   $ docker inspect public.ecr.aws/codebuild/local-builds:latest
   ```

CodeBuild 에이전트 실행

CodeBuild 에이전트를 실행하려면

1. 빌드 프로젝트 소스가 들어 있는 디렉터리로 변경합니다.
2. codebuild_build.sh 스크립트를 다운로드합니다.

   ```
   $ curl -O  https://raw.githubusercontent.com/aws/aws-codebuild-docker-images/
   master/local_builds/codebuild_build.sh
   $ chmod +x codebuild_build.sh
   ```

3. `codebuild_build.sh` 스크립트를 실행하고 컨테이너 이미지와 출력 디렉터리를 지정합니다.

   x86_64 빌드를 실행하려면 다음 명령을 실행합니다.

   ```
   $ ./codebuild_build.sh -i <container-image> -a <output directory>
   ```

   ARM 빌드를 실행하려면 다음 명령을 실행합니다.

```
$ ./codebuild_build.sh -i <container-image> -a <output directory> -l
 public.ecr.aws/codebuild/local-builds:aarch64
```

*<container-image>*를 컨테이너 이미지의 이름(예: aws/codebuild/standard:5.0 또는 public.ecr.aws/codebuild/amazonlinux-x86_64-standard:4.0)으로 바꿉니다.

스크립트는 빌드 이미지를 시작하고 현재 디렉터리의 프로젝트에서 빌드를 실행합니다. 빌드 프로젝트의 위치를 지정하려면 스크립트 명령에 -s *<build project directory>* 옵션을 추가합니다.

새 CodeBuild 에이전트 버전에 대한 알림 받기

Amazon SNS 알림을 구독하면 AWS CodeBuild 에이전트의 새 버전이 릴리스될 때 알림을 받을 수 있습니다.

CodeBuild 에이전트 알림을 구독하려면

1. https://console.aws.amazon.com/sns/v3/home에서 Amazon SNS 콘솔을 엽니다.
2. 탐색 모음에서 아직 선택하지 않은 경우 AWS 리전을 미국 동부(버지니아 북부)로 변경합니다. 구독하려는 Amazon SNS 알림이이 AWS 리전에 생성되므로이 리전을 선택해야 합니다.
3. 탐색 창에서 Subscriptions를 선택합니다.
4. 구독 생성을 선택합니다.
5. 구독 생성에서 다음을 수행합니다.
   a. 주제 ARN에 다음 Amazon 리소스 이름(ARN)을 사용합니다.

      ```
      arn:aws:sns:us-east-1:850632864840:AWS-CodeBuild-Local-Agent-Updates
      ```

   b. 프로토콜의 경우 이메일 또는 SMS를 선택합니다.
   c. 엔드포인트의 경우 알림을 수신할 위치(이메일 또는 SMS)를 선택합니다. 지역 번호를 포함한 이메일 주소, 우편 주소 또는 전화번호를 입력합니다.
   d. 구독 생성을 선택합니다.
   e. 구독 사실을 확인하는 이메일을 받으려면 이메일을 선택합니다. 이메일의 지침에 따라 구독을 완료합니다.

      이런 알림을 더 이상 받지 않기를 원하는 경우, 다음 절차를 수행해서 구독을 해제하세요.

CodeBuild 에이전트 알림을 구독 해제하려면

1. https://console.aws.amazon.com/sns/v3/home에서 Amazon SNS 콘솔을 엽니다.
2. 탐색 창에서 구독을 선택합니다.
3. 구독을 선택한 후 작업에서 구독 삭제를 선택합니다. 확인 메시지가 표시되면 [삭제]를 선택합니다.

빌드 실행(콘솔)

AWS CodePipeline 를 사용하여 CodeBuild로 빌드를 실행하려면 다음 단계를 건너뛰고의 지침을 따릅니다CodePipeline에서 CodeBuild 사용.

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 프로젝트 빌드를 선택합니다.
3. 빌드 프로젝트 목록에서 빌드 프로젝트를 선택합니다.
4. 기본 빌드 프로젝트 설정으로 빌드를 실행하거나 이 빌드의 빌드 설정만 재정의할 수 있습니다.

   a. 기본 빌드 프로젝트 설정으로 빌드를 실행하려면 빌드 시작을 선택합니다. 빌드가 즉시 시작됩니다.

   b. 기본 빌드 프로젝트 설정을 재정의하려면 재정의로 빌드 시작을 선택합니다. 빌드 시작 페이지에서 다음을 재정의할 수 있습니다.

      - 빌드 구성
      - 소스
      - 환경 변수 재정의

      고급 재정의를 더 선택해야 하는 경우 고급 빌드 재정의를 선택합니다. 이 페이지에서 다음을 재정의할 수 있습니다.

      - 빌드 구성
      - 소스
      - 환경
      - Buildspec
      - 아티팩트
      - 로그

재정의를 선택했으면 빌드 시작을 선택합니다.

이 빌드에 대한 자세한 정보는 빌드 세부 정보 보기(콘솔) 단원을 참조하십시오.

빌드 실행(AWS CLI)

> **Note**
>
> CodePipeline을 사용하여 로 빌드를 실행하려면 다음 단계를 AWS CodeBuild건너뛰고의 지침을 따릅니다CodeBuild를 사용하는 파이프라인 생성(AWS CLI).
> CodeBuild AWS CLI 에서를 사용하는 방법에 대한 자세한 내용은 섹션을 참조하세요명령줄 참조.

1. 다음 중 한 방법으로 `start-build` 명령을 실행합니다.

   ```
   aws codebuild start-build --project-name <project-name>
   ```

   빌드 입력 결과물의 최신 버전과 빌드 프로젝트의 기존 설정을 사용하는 빌드를 실행하려면 이 방법을 사용합니다.

   ```
   aws codebuild start-build --generate-cli-skeleton
   ```

   이전 버전의 빌드 입력 결과물을 사용하여 빌드를 실행하려는 경우 또는 빌드 출력 결과물, 환경 변수, buildspec 또는 기본 빌드 제한 시간의 설정을 재정의하려는 경우 이 방법을 사용합니다.

2. start-build 명령을 `--project-name` 옵션과 함께 실행하는 경우 *<project-name>*을 빌드 프로젝트의 이름으로 바꾼 다음 이 절차의 6단계로 이동합니다. 빌드 프로젝트 목록을 가져오려면 빌드 프로젝트 이름 보기 단원을 참조하십시오.

3. start-build 명령을 `--idempotency-token` 옵션과 함께 실행하면 고유의 대소문자 구분 식별자 또는 토큰이 `start-build` 요청에 포함됩니다. 토큰은 요청 후 5분 동안 유효합니다. 동일한 토큰을 사용하여 `start-build` 요청을 반복하고 파라미터를 변경하면 CodeBuild는 파라미터 불일치 오류를 반환합니다.

4. start-build 명령을 `--generate-cli-skeleton` 옵션과 함께 실행하는 경우 JSON 형식 데이터가 출력에 표시됩니다. 가 설치된 로컬 컴퓨터 또는 인스턴스의 위치에 있는 파일(예: *start-*

build.json)에 데이터를 복사 AWS CLI 합니다. 복사된 데이터를 다음 형식으로 수정한 다음 결과를 저장합니다.

```
{
  "projectName": "projectName",
  "sourceVersion": "sourceVersion",
  "artifactsOverride": {
    "type": "type",
    "location": "location",
    "path": "path",
    "namespaceType": "namespaceType",
    "name": "artifactsOverride-name",
    "packaging": "packaging"
  },
  "buildspecOverride": "buildspecOverride",
  "cacheOverride": {
    "location": "cacheOverride-location",
    "type": "cacheOverride-type"
  },
  "certificateOverride": "certificateOverride",
  "computeTypeOverride": "computeTypeOverride",
  "environmentTypeOverride": "environmentTypeOverride",
  "environmentVariablesOverride": {
    "name": "environmentVariablesOverride-name",
    "value": "environmentVariablesValue",
    "type": "environmentVariablesOverride-type"
  },
  "gitCloneDepthOverride": "gitCloneDepthOverride",
  "imageOverride": "imageOverride",
  "idempotencyToken": "idempotencyToken",
  "insecureSslOverride": "insecureSslOverride",
  "privilegedModeOverride": "privilegedModeOverride",
  "queuedTimeoutInMinutesOverride": "queuedTimeoutInMinutesOverride",
  "reportBuildStatusOverride": "reportBuildStatusOverride",
  "timeoutInMinutesOverride": "timeoutInMinutesOverride",
  "sourceAuthOverride": "sourceAuthOverride",
  "sourceLocationOverride": "sourceLocationOverride",
  "serviceRoleOverride": "serviceRoleOverride",
  "sourceTypeOverride": "sourceTypeOverride"
}
```

다음과 같이 자리 표시자를 바꿉니다.

- *projectName*: 필수 문자열입니다. 이 빌드에 사용할 빌드 프로젝트의 이름입니다.
- *sourceVersion*: 선택적 문자열입니다. 빌드할 소스 코드의 버전으로, 다음과 같습니다.
  - Amazon S3의 경우, 빌드하려는 입력 ZIP 파일의 버전에 해당하는 버전 ID입니다. *sourceVersion*을 지정하지 않은 경우 최신 버전이 사용됩니다.
  - CodeCommit의 경우, 빌드하려는 소스 코드의 버전에 해당하는 커밋 ID입니다. *sourceVersion*을 지정하지 않은 경우 기본 분기의 HEAD 커밋 ID가 사용됩니다. (*sourceVersion*의 태그 이름은 지정할 수 없지만, 태그의 커밋 ID는 지정할 수 있습니다.)
  - GitHub의 경우, 빌드하려는 소스 코드의 버전에 해당하는 커밋 ID, 풀 요청 ID, 분기 이름 또는 태그 이름입니다. 풀 요청 ID가 지정된 경우 `pr/`*pull-request-ID* 형식을 사용해야 합니다(예: `pr/25`). 분기 이름이 지정되어 있으면 분기의 HEAD 커밋 ID가 사용됩니다. *sourceVersion*을 지정하지 않은 경우 기본 분기의 HEAD 커밋 ID가 사용됩니다.
  - Bitbucket의 경우, 빌드하려는 소스 코드의 버전에 해당하는 커밋 ID, 분기 이름 또는 태그 이름입니다. 분기 이름이 지정되어 있으면 분기의 HEAD 커밋 ID가 사용됩니다. *sourceVersion*을 지정하지 않은 경우 기본 분기의 HEAD 커밋 ID가 사용됩니다.
- 다음 자리 표시자는 `artifactsOverride`용입니다.
  - *type*: 선택 사항입니다. 빌드 프로젝트에서 정의된 빌드 출력 결과물 유형이 아닌 이 빌드에서만 사용되는 빌드 출력 결과물 유형입니다.
  - *location*: 선택 사항입니다. 빌드 프로젝트에서 정의된 빌드 출력 결과물 위치가 아닌 이 빌드에서만 사용되는 빌드 출력 결과물 위치입니다.
  - *path*: 선택 사항입니다. 빌드 프로젝트에서 정의된 빌드 출력 결과물 경로가 아닌 이 빌드에서만 사용되는 빌드 출력 결과물 경로입니다.
  - *namespaceType*: 선택 사항입니다. 빌드 프로젝트에서 정의된 빌드 출력 결과물 유형이 아닌 이 빌드에서만 사용되는 빌드 출력 결과물 유형입니다.
  - *name*: 선택 사항입니다. 빌드 프로젝트에서 정의된 빌드 출력 결과물 이름이 아닌 이 빌드에서만 사용되는 빌드 출력 결과물 이름입니다.
  - *packaging*: 선택 사항입니다. 빌드 프로젝트에서 정의된 빌드 출력 결과물 패키징이 아닌 이 빌드에서만 사용되는 빌드 출력 결과물 패키징입니다.
- *buildspecOverride*: 선택 사항입니다. 빌드 프로젝트에서 정의된 buildspec 선언이 아닌 이 빌드에서만 사용되는 buildspec 선언입니다. 이 값이 설정된 경우 인라인 buildspec 정의, 내장된 `CODEBUILD_SRC_DIR` 환경 변수의 값에 상대적인 대체 buildspec 파일 또는 S3 버킷의 경로가 될 수 있습니다. S3 버킷은 빌드 프로젝트와 동일한 AWS 리전에 있어야 합니다. ARN을 사용하여 buildspec 파일을 지정합니다(예: `arn:aws:s3:::`*<my-codebuild-sample2>*`/buildspec.yml`). 이 값을 제공하지 않거나 빈 문자열로 설정하는 경우 소스 코드에 루트 디렉

터리의 buildspec.yml 파일이 포함되어 있어야 합니다. 자세한 내용은 buildspec 파일 이름 및 스토리지 위치 단원을 참조하십시오.

- 다음 자리 표시자는 cacheOverride용입니다.
  - *cacheOverride-location*: 선택 사항입니다. 빌드 프로젝트에서 지정된 ProjectCache 객체를 재정의하는 이 빌드에 대한 ProjectCache 객체의 위치입니다. cacheOverride는 선택 사항이며 ProjectCache 객체를 가져옵니다. location는 ProjectCache 객체에서 필수입니다.
  - *cacheOverride-type*: 선택 사항입니다. 이 빌드에 대한 ProjectCache 객체의 유형으로, 빌드 프로젝트에서 지정된 ProjectCache 객체를 재정의합니다. cacheOverride는 선택 사항이며 ProjectCache 객체를 가져옵니다. type는 ProjectCache 객체에서 필수입니다.
- *certificateOverride*: 선택 사항입니다. 빌드 프로젝트에서 지정된 인증서를 재정의하는 이 빌드에 대한 인증서의 이름입니다.
- *environmentTypeOverride*: 선택 사항입니다. 빌드 프로젝트에서 지정된 컨테이너를 재정의하는 이 빌드의 컨테이너 유형입니다. 현재 유효한 문자열은 LINUX_CONTAINER입니다.
- 다음 자리 표시자는 environmentVariablesOverride용입니다.
  - *environmentVariablesOverride-name*: 선택 사항입니다. 이 빌드에서 값을 재정의하려는 빌드 프로젝트의 환경 변수 이름입니다.
  - *environmentVariablesOverride-type*: 선택 사항입니다. 이 빌드에서 값을 재정의하려는 빌드 프로젝트의 환경 변수 유형입니다.
  - *environmentVariablesValue*: 선택 사항입니다. 이 빌드에서 재정의하려는 빌드 프로젝트에 정의된 환경 변수 값입니다.
- *gitCloneDepthOverride*: 선택 사항입니다. 이 빌드에서 값을 재정의하려는 빌드 프로젝트의 Git clone depth 값입니다. 소스 유형이 Amazon S3일 경우 이 값이 지원되지 않습니다.
- *imageOverride*: 선택 사항입니다. 빌드 프로젝트에서 지정된 이미지를 재정의하는 이 빌드에 대한 이미지의 이름입니다.
- *idempotencyToken*: 선택 사항입니다. 빌드 요청이 idempotent임을 지정하기 위해 토큰으로 제공되는 문자열입니다. 64자 이하의 모든 문자열을 선택할 수 있습니다. 토큰은 start-build 요청 후 5분 동안 유효합니다. 동일한 토큰을 사용하여 start-build 요청을 반복하고 파라미터를 변경하면 CodeBuild는 파라미터 불일치 오류를 반환합니다.
- *insecureSslOverride*: 빌드 프로젝트에서 지정된 안전하지 않은 TLS 설정을 재정의할지 여부를 지정하는 선택적 부울입니다. 안전하지 않은 TLS 설정은 프로젝트 소스 코드에 연결하는 동안 TLS 경고를 무시할지 여부를 결정합니다. 이 설정은 빌드의 소스가 GitHub Enterprise Server인 경우에만 적용됩니다.

- *privilegedModeOverride*: 선택적 부울입니다. true로 설정하면 빌드는 빌드 프로젝트에서 권한이 있는 모드를 재정의합니다.
- *queuedTimeoutInMinutesOverride*: 빌드 대기 시간이 얼마나 지나야 시간 초과로 처리되는지를 지정하는 정수(분)입니다(선택 사항). 최솟값은 5분이며, 최댓값은 480분(8시간)입니다.
- *reportBuildStatusOverride*: 빌드의 시작 및 완료 상태를 소스 공급자에게 보낼지 여부를 지정하는 선택적 부울입니다. GitHub, GitHub Enterprise Server 또는 Bitbucket이 아닌 소스 공급자로 설정하는 경우, invalidInputException이 발생합니다.
- *sourceAuthOverride*: 선택적 문자열입니다. 빌드 프로젝트에서 정의된 권한 부여를 재정의하는 이 빌드에 대한 권한 부여 유형입니다. 이 재정의는 빌드 프로젝트의 소스가 Bitbucket 또는 GitHub인 경우에만 적용됩니다.
- *sourceLocationOverride*: 선택적 문자열입니다. 빌드 프로젝트에서 정의된 위치의 소스 위치를 이 빌드에 대해 재정의하는 위치입니다.
- *serviceRoleOverride*: 선택 사항 문자열. 빌드 프로젝트에서 지정된 서비스 역할을 재정의하는 이 빌드에 대한 서비스 역할의 이름입니다.
- *sourceTypeOverride*: 선택적 문자열입니다. 빌드 프로젝트에서 정의된 소스 입력을 재정의하는 이 빌드에 대한 소스 입력 유형입니다. 유효한 문자열은 `NO_SOURCE`, `CODECOMMIT`, `CODEPIPELINE`, `GITHUB`, `S3`, `BITBUCKET` 및 `GITHUB_ENTERPRISE`입니다.
- *timeoutInMinutesOverride*: 선택적 숫자입니다. 빌드 프로젝트에서 정의된 빌드 제한 시간(분)이 아닌 이 빌드에서만 사용되는 빌드 제한 시간(분)입니다.

액세스 키 ID, AWS 보안 AWS 액세스 키 또는 암호와 같은 민감한 값을 가진 환경 변수를 Amazon EC2 Systems Manager 파라미터 스토어에 파라미터로 저장하는 것이 좋습니다. CodeBuild는 Amazon EC2 Systems Manager Parameter Store에 저장된 파라미터 이름이 `/CodeBuild/`(예: `/CodeBuild/dockerLoginPassword`)로 시작하는 경우에만 해당 파라미터를 사용할 수 있습니다. CodeBuild 콘솔을 사용하여 Amazon EC2 Systems Manager에서 파라미터를 생성할 수 있습니다. 파라미터 생성을 선택한 후 지침에 따릅니다. 경우에 따라 대화 상자의 KMS 키에 해당 계정의 AWS KMS 키에 대한 ARN을 지정할 수 있습니다. Amazon EC2 Systems Manager는 이 키를 사용하여 저장 시 파라미터의 값을 암호화하고 검색 시 암호를 해독합니다. CodeBuild 콘솔을 사용하여 파라미터를 생성하는 경우 콘솔은 `/CodeBuild/`로 파라미터를 시작합니다. 하지만 Amazon EC2 Systems Manager Parameter Store 콘솔을 사용하여 파라미터를 생성하는 경우, `/CodeBuild/`로 파라미터 이름을 시작해야 하고 유형을 보안 문자열로 설정해야 합니다. 자세한 내용은 Amazon EC2 Systems Manager 사용 설명서의 AWS Systems Manager 파라미터 스토어 및 연습: 문자열 파라미터 생성 및 테스트(콘솔)을 참조하세요.

빌드 프로젝트가 Amazon EC2 Systems Manager Parameter Store에 저장된 파라미터를 참조하는 경우 해당 빌드 프로젝트의 서비스 역할은 `ssm:GetParameters` 작업을 허용해야 합니다. 앞에서 해당 계정에 새로운 서비스 역할 생성을 선택한 경우 CodeBuild는 빌드 프로젝트의 기본 서비스 역할에 이 작업을 자동으로 포함합니다. 하지만 [Choose an existing service role from your account]를 선택한 경우 이 작업을 서비스 역할에 별도로 포함해야 합니다.

사용자가 설정한 환경 변수는 기존 환경 변수를 대체합니다. 예를 들어 도커 이미지에 값이 `my_value`인 `MY_VAR`이라는 환경 변수가 이미 포함되어 있는데, 사용자가 `MY_VAR` 환경 변수의 값을 `other_value`로 설정하면, `my_value`가 `other_value`로 바뀝니다. 마찬가지로, 도커 이미지에 값이 `/usr/local/sbin:/usr/local/bin`인 `PATH`라는 환경 변수가 이미 포함되어 있는데, 사용자가 `PATH` 환경 변수의 값을 `$PATH:/usr/share/ant/bin`으로 설정하면, `/usr/local/sbin:/usr/local/bin`이 `$PATH:/usr/share/ant/bin` 리터럴 값으로 바뀝니다.

`CODEBUILD_`로 시작하는 이름으로 환경 변수를 설정하지 마십시오. 이 접두사는 내부 전용으로 예약되어 있습니다.

여러 위치에서 동일한 이름의 환경 변수가 정의되는 경우, 다음과 같이 환경 변수 값이 결정됩니다.

- 시작 빌드 작업 호출의 값이 가장 높은 우선 순위를 갖습니다.
- 빌드 프로젝트 정의의 값이 다음 우선 순위를 갖습니다.
- buildspec 파일 선언의 값이 가장 낮은 우선 순위를 갖습니다.

이 자리 표시자의 유효한 값에 대한 자세한 정보는 빌드 프로젝트 생성(AWS CLI) 단원을 참조하십시오. 빌드 프로젝트의 최신 설정 목록은 빌드 프로젝트 세부 정보 보기 단원을 참조하십시오.

5. 방금 저장한 파일이 들어 있는 디렉터리로 전환한 다음, `start-build` 명령을 다시 실행합니다.

```
aws codebuild start-build --cli-input-json file://start-build.json
```

6. 이 명령이 제대로 실행되면 빌드를 실행하려면 절차에 설명된 것과 유사한 데이터가 출력에 표시됩니다.

이 빌드에 대한 자세한 정보를 보려면 출력에서 `id` 값을 적어 둔 다음 빌드 세부 정보 보기(AWS CLI) 단원을 참조하십시오.

배치 빌드 실행(AWS CLI)

1. 다음 중 한 방법으로 `start-build-batch` 명령을 실행합니다.

   ```
   aws codebuild start-build-batch --project-name <project-name>
   ```

   빌드 입력 결과물의 최신 버전과 빌드 프로젝트의 기존 설정을 사용하는 빌드를 실행하려면 이 방법을 사용합니다.

   ```
   aws codebuild start-build-batch --generate-cli-skeleton > <json-file>
   ```

   이전 버전의 빌드 입력 결과물을 사용하여 빌드를 실행하려는 경우 또는 빌드 출력 결과물, 환경 변수, buildspec 또는 기본 빌드 제한 시간의 설정을 재정의하려는 경우 이 방법을 사용합니다.
2. start-build-batch 명령을 `--project-name` 옵션과 함께 실행하는 경우 *<project-name>*을 빌드 프로젝트의 이름으로 바꾼 다음 이 절차의 6단계로 이동합니다. 빌드 프로젝트 목록을 가져오려면 빌드 프로젝트 이름 보기 단원을 참조하십시오.
3. start-build-batch 명령을 `--idempotency-token` 옵션과 함께 실행하면 고유의 대소문자 구분 식별자 또는 토큰이 `start-build-batch` 요청에 포함됩니다. 토큰은 요청 후 5분 동안 유효합니다. 동일한 토큰을 사용하여 `start-build-batch` 요청을 반복하고 파라미터를 변경하면 CodeBuild는 파라미터 불일치 오류를 반환합니다.
4. start-build-batch 명령을 `--generate-cli-skeleton` 옵션과 함께 실행하는 경우 JSON 형식 데이터가 *<json-file>* 파일에 출력됩니다. 이 파일은 다음 객체를 추가하여 start-build 명령에서 생성된 스켈레톤과 유사합니다. 일반 객체에 대한 자세한 내용은 빌드 실행(AWS CLI) 섹션을 참조하세요.

   이 파일을 수정하여 빌드 재정의를 추가하고 결과를 저장하세요.

   ```
   "buildBatchConfigOverride": {
     "combineArtifacts": combineArtifacts,
     "restrictions": {
       "computeTypesAllowed": [
         allowedComputeTypes
       ],
       "maximumBuildsAllowed": maximumBuildsAllowed
     },
     "serviceRole": "batchServiceRole",
     "timeoutInMins": batchTimeout
   }
   ```

buildBatchConfigOverride 객체는이 빌드에 대한 배치 빌드 구성 재정의를 포함하는 ProjectBuildBatchConfig 구조입니다.

combineArtifacts

배치 빌드의 빌드 아티팩트를 단일 아티팩트 위치로 결합할지 여부를 지정하는 부울입니다.

allowedComputeTypes

배치 빌드에 허용되는 컴퓨팅 유형을 지정하는 문자열 배열입니다. 이러한 값은 빌드 환경 컴퓨팅 유형를 참조하세요.

maximumBuildsAllowed

허용되는 최대 빌드 수를 지정합니다.

batchServiceRole

배치 빌드 프로젝트에 대한 서비스 역할 ARN을 지정합니다.

batchTimeout

배치 빌드를 완료해야 하는 최대 시간(분)을 지정합니다.

5. 방금 저장한 파일이 들어 있는 디렉터리로 전환한 다음, start-build-batch 명령을 다시 실행합니다.

```
aws codebuild start-build-batch --cli-input-json file://start-build.json
```

6. 성공하면 BuildBatch 객체의 JSON 표현이 콘솔 출력에 나타납니다. 이 데이터의 예는 StartBuildBatch 응답 구문을 참조하세요.

빌드 실행 자동 시작(AWS CLI)

소스 코드가 GitHub 또는 GitHub Enterprise Server 리포지토리에 저장된 경우 코드 변경이 리포지토리로 푸시될 때마다 GitHub 웹후크를 사용하여 소스 코드를 AWS CodeBuild 다시 빌드할 수 있습니다.

다음과 같이 create-webhook 명령을 실행합니다.

```
aws codebuild create-webhook --project-name <project-name>
```

*<project-name>*은 다시 빌드할 소스 코드가 포함되어 있는 빌드 프로젝트의 이름입니다.

GitHub의 경우, 다음과 비슷한 정보가 결과에 나타납니다.

```
{
  "webhook": {
    "url": "<url>"
  }
}
```

*<url>*은 GitHub Webhook에 대한 URL입니다.

GitHub Enterprise Server의 경우, 다음과 비슷한 정보가 결과에 나타납니다.



1. 출력에서 보안 키 및 페이로드 URL을 복사합니다. 이들 값은 GitHub Enterprise Server에 Webhook를 추가할 때 필요합니다.
2. GitHub Enterprise Server에서 CodeBuild 프로젝트가 저장된 리포지토리를 선택합니다. 설정, Hooks & services(후크 및 서비스), Add webhook(webhook 추가)를 차례로 선택합니다.
3. 페이로드 URL 및 보안 키를 입력하고 그 외 필드에 대해서는 기본값을 수락한 다음 [Add webhook]를 선택합니다.

빌드 실행 자동 중지(AWS CLI)

소스 코드가 GitHub 또는 GitHub Enterprise Server 리포지토리에 저장된 경우 코드 변경이 리포지토리로 푸시될 때마다 소스 코드를 AWS CodeBuild 다시 빌드하도록 GitHub 웹후크를 설정할 수 있습니다. 자세한 내용은 빌드 실행 자동 시작(AWS CLI) 단원을 참조하십시오.

이 동작을 활성화한 경우 다음과 같이 `delete-webhook` 명령을 실행하여 해제할 수 있습니다.

```
aws codebuild delete-webhook --project-name <project-name>
```

- *<project-name>*은 다시 빌드할 소스 코드가 포함되어 있는 빌드 프로젝트의 이름입니다.

이 명령이 제대로 실행되면 출력에 정보나 오류가 표시되지 않습니다.

> **Note**
>
> 이렇게 하면 Webhook가 CodeBuild 프로젝트에서만 삭제됩니다. Webhook를 GitHub 또는 GitHub Enterprise Server 리포지토리에서도 삭제해야 합니다.

빌드 실행(AWS SDKs)

CodePipeline을 사용하여 빌드를 실행하려면이 단계를 AWS CodeBuild건너뛰고 와 AWS CodeBuild 함께 AWS CodePipeline 를 사용하여 코드를 테스트하고 빌드를 실행합니다. 대신의 지침을 따르세요.

CodeBuild를 AWS SDKs와 함께 사용하는 방법에 대한 자세한 내용은 섹션을 참조하세요AWS SDKs 및 도구 참조.

AWS Lambda 컴퓨팅 기반 빌드 실행

AWS Lambda 컴퓨팅은 빌드에 최적화된 시작 속도를 제공합니다.는 시작 지연 시간이 짧아 더 빠른 빌드를 AWS Lambda 지원합니다. AWS Lambda 또한는 자동으로 확장되므로 빌드가 실행 대기열에서 대기하지 않습니다. 그러나에서 지원하지 AWS Lambda 않는 몇 가지 사용 사례가 있으며, 이것이 사용자에게 영향을 미치는 경우 EC2 컴퓨팅을 사용합니다. 자세한 내용은 AWS Lambda 컴퓨팅의 제한 사항 단원을 참조하십시오.

주제

- AWS Lambda에서 실행되는 큐레이팅된 런타임 환경 도커 이미지에는 어떤 도구와 런타임이 포함되나요?
- 큐레이팅된 이미지에 필요한 도구가 포함되어 있지 않으면 어떻게 해야 하나요?
- CodeBuild에서 AWS Lambda 컴퓨팅을 지원하는 리전은 어디입니까?
- AWS Lambda 컴퓨팅의 제한 사항
- CodeBuild Lambda Java와 AWS SAM 함께를 사용하여 Lambda 함수 배포
- CodeBuild Lambda Node.js를 사용하여 단일 페이지 React 앱 생성
- CodeBuild Lambda Python으로 Lambda 함수 구성 업데이트

AWS Lambda에서 실행되는 큐레이팅된 런타임 환경 도커 이미지에는 어떤 도구와 런타임이 포함되나요?

AWS Lambda 는 AWS CLI v2, AWS SAM CLI, git, go, Java, Node.js, Python, pip, Ruby 및 .NET 도구를 지원합니다.

큐레이팅된 이미지에 필요한 도구가 포함되어 있지 않으면 어떻게 해야 하나요?

큐레이팅된 이미지에 필요한 도구가 포함되어 있지 않은 경우 필요한 도구가 포함된 사용자 지정 환경 Docker 이미지를 제공할 수 있습니다.

> **Note**
>
> Lambda는 다중 아키텍처 컨테이너 이미지를 사용하는 함수를 지원하지 않습니다. 자세한 내용은 AWS Lambda 개발자 안내서의 컨테이너 이미지를 사용하여 Lambda 함수 생성을 참조하세요.

Lambda 컴퓨팅에 사용자 지정 이미지를 사용하려면 다음 Amazon ECR 권한이 필요합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
```

```
            ],
            "Resource": "arn:aws:ecr:us-east-1:111122223333:repository/image-
repo"
        }
    ]
}
```

또한 사용자 지정 이미지를 사용하려면 `curl` 또는 `wget`을 설치해야 합니다.

CodeBuild에서 AWS Lambda 컴퓨팅을 지원하는 리전은 어디입니까?

CodeBuild에서 AWS Lambda 컴퓨팅은 AWS 리전미국 동부(버지니아 북부), 미국 동부(오하이오), 미국 서부(오레곤), 아시아 태평양(뭄바이), 아시아 태평양(싱가포르), 아시아 태평양(시드니), 아시아 태평양(도쿄), 유럽(프랑크푸르트), 유럽(아일랜드), 남아메리카(상파울루)에서 지원됩니다. CodeBuild가 사용 가능한 AWS 리전 에 대한 자세한 내용은 리전별AWS 서비스를 참조하세요.

AWS Lambda 컴퓨팅의 제한 사항

에서 지원하지 AWS Lambda 않는 몇 가지 사용 사례가 있으며, 이러한 사용 사례가 사용자에게 영향을 미치는 경우 EC2 컴퓨팅을 사용합니다.

- AWS Lambda 는 루트 권한이 필요한 도구를 지원하지 않습니다. `yum` 또는 `rpm` 등의 도구에는 EC2 컴퓨팅 유형이나 루트 권한이 필요하지 않은 기타 도구를 사용하세요.
- AWS Lambda 는 Docker 빌드 또는 실행을 지원하지 않습니다.
- AWS Lambda 는 외부 파일에 쓰기를 지원하지 않습니다`/tmp`. 포함된 패키지 관리자는 기본적으로 이 `/tmp` 디렉터리를 사용하여 패키지를 다운로드하고 참조하도록 구성되어 있습니다.
- AWS Lambda 는 환경 유형을 지원하지 `LINUX_GPU_CONTAINER` 않으며 Windows Server Core 2019에서 지원되지 않습니다.
- AWS Lambda 는 캐싱, 사용자 지정 빌드 제한 시간, 대기열 제한 시간, 빌드 배지, 권한 모드, 사용자 지정 런타임 환경 또는 15분보다 긴 런타임을 지원하지 않습니다.
- AWS Lambda 는 VPC 연결, 고정된 범위의 CodeBuild 소스 IP 주소, EFS, 인증서 설치 또는 Session Manager를 사용한 SSH 액세스를 지원하지 않습니다.

CodeBuild Lambda Java와 AWS SAM 함께를 사용하여 Lambda 함수 배포

AWS Serverless Application Model (AWS SAM)는 서버리스 애플리케이션을 빌드하기 위한 오픈 소스 프레임워크입니다. 자세한 내용은 GitHub의 AWS Serverless Application Model 리포지토리를 참조하

십시오. 다음 Java 샘플은 Gradle을 사용하여 AWS Lambda 함수를 빌드하고 테스트합니다. 그런 다음 AWS SAM CLI를 사용하여 CloudFormation 템플릿과 배포 번들을 배포합니다. CodeBuild Lambda를 사용하면 빌드, 테스트 및 배포 단계가 모두 자동으로 처리되므로 단일 빌드에서 수동 개입 없이 인프라를 빠르게 업데이트할 수 있습니다.

AWS SAM 리포지토리 설정

AWS SAM CLI를 AWS SAM `Hello World` 사용하여 프로젝트를 생성합니다.

AWS SAM 프로젝트를 생성하려면

1. 로컬 시스템에 AWS SAM CLI를 설치하기 위한 AWS Serverless Application Model 개발자 안내서의 지침을 따릅니다.
2. `sam init`을 실행하고 다음 프로젝트 구성을 선택합니다.

```
Which template source would you like to use?: 1 - AWS Quick Start Templates
Choose an AWS Quick Start application template: 1 - Hello World Example
Use the most popular runtime and package type? (Python and zip) [y/N]: N
Which runtime would you like to use?: 8 - java21
What package type would you like to use?: 1 - Zip
Which dependency manager would you like to use?: 1 - gradle
Would you like to enable X-Ray tracing on the function(s) in your application? [y/
N]: N
Would you like to enable monitoring using CloudWatch Application Insights? [y/N]: N
Would you like to set Structured Logging in JSON format on your Lambda functions?
 [y/N]:  N
Project name [sam-app]: <insert project name>
```

3. AWS SAM 프로젝트 폴더를 지원되는 소스 리포지토리에 업로드합니다. 지원되는 소스 유형 목록은 ProjectSource를 참조하세요.

CodeBuild Lambda Java 프로젝트 생성

AWS CodeBuild Lambda Java 프로젝트를 생성하고 빌드에 필요한 IAM 권한을 설정합니다.

CodeBuild Lambda Java 프로젝트를 생성하려면

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. CodeBuild 정보 페이지가 나타나면 빌드 프로젝트 생성을 선택합니다. 그렇지 않을 경우, 탐색 창에서 빌드를 확장한 후 빌드 프로젝트를 선택하고 빌드 프로젝트 생성을 선택합니다.

3. 프로젝트 이름에 이 빌드 프로젝트의 이름을 입력합니다. 빌드 프로젝트 이름은 각 AWS 계정에서 고유해야 합니다. 또한 선택에 따라 빌드 프로젝트에 대한 설명을 포함하여 다른 사용자가 이 프로젝트의 용도를 이해하도록 도울 수 있습니다.
4. 소스에서 AWS SAM 프로젝트가 위치한 소스 리포지토리를 선택합니다.
5. 환경에서 다음과 같이 합니다.
   - 컴퓨팅에서 Lambda를 선택합니다.
   - 런타임에서 Java를 선택합니다.
   - 이미지에서 aws/codebuild/amazonlinux-x86_64-lambda-standard:corretto21을 선택합니다.
   - 서비스 역할의 경우 새 서비스 역할을 선택한 상태로 둡니다. 역할 이름을 기록해 둡니다. 이 샘플의 뒷부분에서 프로젝트의 IAM 권한을 업데이트할 때 필요합니다.
6. 빌드 프로젝트 생성을 선택합니다.
7. IAM 콘솔(https://console.aws.amazon.com/iam/)을 엽니다.
8. 탐색 창에서 역할을 선택하고 프로젝트와 연결된 서비스 역할을 선택합니다. 빌드 프로젝트를 선택하고 편집, 환경, 서비스 역할을 선택하여 CodeBuild에서 프로젝트 역할을 찾을 수 있습니다.
9. 신뢰 관계 탭을 선택한 후 신뢰 정책 편집을 선택합니다.
10. 다음 인라인 정책을 IAM 역할에 추가합니다. 이는 나중에 AWS SAM 인프라를 배포하는 데 사용됩니다. 자세한 내용은 IAM 사용 설명서의 IAM 자격 증명 권한 추가 및 제거를 참조하세요.

    JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Action": [
                "cloudformation:*",
                "lambda:*",
                "iam:*",
                "apigateway:*",
                "s3:*"
            ],
            "Resource": "arn:aws:iam::*:role/Service*"
        }
    ]
```

```
}
```

프로젝트 buildspec 설정

Lambda 함수를 빌드, 테스트 및 배포하기 위해 CodeBuild는 buildspec에서 빌드 명령을 읽고 실행합니다.

프로젝트 buildspec을 설정하려면

1. CodeBuild 콘솔에서 빌드 프로젝트를 선택한 다음 편집 및 Buildspec을 선택합니다.
2. Buildspec에서 빌드 명령 삽입을 선택한 후 편집기로 전환을 선택합니다.
3. 사전 채워진 빌드 명령을 삭제하고 다음 buildspec에 붙여넣습니다.

```
version: 0.2
env:
  variables:
    GRADLE_DIR: "HelloWorldFunction"
phases:
  build:
    commands:
      - echo "Running unit tests..."
      - cd $GRADLE_DIR; gradle test; cd ..
      - echo "Running build..."
      - sam build --template-file template.yaml
      - echo "Running deploy..."
      - sam package --output-template-file packaged.yaml --resolve-s3 --template-
file template.yaml
      - yes | sam deploy
```

4. Update buildspec(buildspec 업데이트)을 선택합니다.

AWS SAM Lambda 인프라 배포

CodeBuild Lambda를 사용하여 Lambda 인프라 자동 배포

Lambda 인프라를 배포하려면

1. 빌드 시작를 선택합니다. 이렇게 하면를 AWS Lambda 사용하여 AWS SAM 애플리케이션을 자동으로 빌드, 테스트 및 배포할 수 있습니다 CloudFormation.

2. 빌드가 완료되면 AWS Lambda 콘솔로 이동하여 AWS SAM 프로젝트 이름에서 새 Lambda 함수를 검색합니다.
3. 함수 개요에서 API Gateway를 선택한 다음 API 엔드포인트 URL을 클릭하여 Lambda 함수를 테스트합니다. `"message": "hello world"` 메시지가 포함된 페이지가 열려 있어야 합니다.

인프라 정리

이 자습서에서 사용한 리소스에 대한 추가 요금이 부과되지 않도록 AWS SAM 템플릿 및 CodeBuild에서 생성한 리소스를 삭제합니다.

인프라를 정리하려면

1. CloudFormation 콘솔로 이동하여를 선택합니다`aws-sam-cli-managed-default`.
2. 리소스에서 배포 버킷 `SamCliSourceBucket`을 비웁니다.
3. `aws-sam-cli-managed-default` 스택을 삭제합니다.
4. AWS SAM 프로젝트와 연결된 CloudFormation 스택을 삭제합니다. 이 스택의 이름은 AWS SAM 프로젝트와 동일해야 합니다.
5. CloudWatch 콘솔로 이동하여 CodeBuild 프로젝트와 연결된 CloudWatch 로그 그룹을 삭제합니다.
6. CodeBuild 콘솔로 이동하여 빌드 프로젝트 삭제를 선택하여 CodeBuild 프로젝트를 삭제합니다.

CodeBuild Lambda Node.js를 사용하여 단일 페이지 React 앱 생성

React 앱 생성은 단일 페이지 React 애플리케이션을 생성하는 방법입니다. 다음 Node.js 샘플은 Node.js를 사용하여 React 앱 생성에서 소스 아티팩트를 빌드하고 빌드 아티팩트를 반환합니다.

소스 리포지토리 및 아티팩트 버킷 설정

Yarn 및 React 앱 생성을 사용하여 프로젝트의 소스 리포지토리를 생성합니다.

소스 리포지토리 및 아티팩트 버킷을 설정하려면

1. 로컬 시스템에서 `yarn create react-app <app-name>`을 실행하여 간단한 React 앱을 생성합니다.
2. React 앱 프로젝트 폴더를 지원되는 소스 리포지토리에 업로드합니다. 지원되는 소스 유형 목록은 ProjectSource를 참조하세요.

CodeBuild Lambda Node.js 프로젝트 생성

AWS CodeBuild Lambda Node.js 프로젝트를 생성합니다.

CodeBuild Lambda Node.js 프로젝트를 생성하려면

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. CodeBuild 정보 페이지가 나타나면 빌드 프로젝트 생성을 선택합니다. 그렇지 않을 경우, 탐색 창에서 빌드를 확장한 후 빌드 프로젝트를 선택하고 빌드 프로젝트 생성을 선택합니다.
3. 프로젝트 이름에 이 빌드 프로젝트의 이름을 입력합니다. 빌드 프로젝트 이름은 각 AWS 계정에서 고유해야 합니다. 또한 선택에 따라 빌드 프로젝트에 대한 설명을 포함하여 다른 사용자가 이 프로젝트의 용도를 이해하도록 도울 수 있습니다.
4. 소스에서 AWS SAM 프로젝트가 위치한 소스 리포지토리를 선택합니다.
5. 환경에서 다음과 같이 합니다.
   - 컴퓨팅에서 Lambda를 선택합니다.
   - 런타임에서 Node.js를 선택합니다.
   - 이미지에서 aws/codebuild/amazonlinux-x86_64-lambda-standard:nodejs20을 선택합니다.
6. 결과물에서 다음과 같이 합니다.
   - 유형에서 Amazon S3를 선택합니다.
   - 버킷 이름에서 이전에 생성한 프로젝트 아티팩트 버킷을 선택합니다.
   - 아티팩트 패키징에서 Zip을 선택합니다.
7. 빌드 프로젝트 생성을 선택합니다.

프로젝트 buildspec 설정

CodeBuild는 React 앱을 빌드하기 위해 buildspec 파일에서 빌드 명령을 읽고 실행합니다.

프로젝트 buildspec을 설정하려면

1. CodeBuild 콘솔에서 빌드 프로젝트를 선택한 다음 편집 및 Buildspec을 선택합니다.
2. Buildspec에서 빌드 명령 삽입을 선택한 후 편집기로 전환을 선택합니다.
3. 사전 채워진 빌드 명령을 삭제하고 다음 buildspec에 붙여넣습니다.

```
version: 0.2
```

```
phases:
  build:
    commands:
      - yarn
      - yarn add --dev jest-junit @babel/plugin-proposal-private-property-in-object
      - yarn run build
      - yarn run test -- --coverage --watchAll=false --testResultsProcessor="jest-
junit" --detectOpenHandles
artifacts:
  name: "build-output"
  files:
    - "**/*"
reports:
  test-report:
    files:
      - 'junit.xml'
    file-format: 'JUNITXML'
  coverage-report:
    files:
      - 'coverage/clover.xml'
    file-format: 'CLOVERXML'
```

4. Update buildspec(buildspec 업데이트)을 선택합니다.

React 앱 빌드 및 실행

CodeBuild Lambda에서 React 앱을 빌드하고, 빌드 아티팩트를 다운로드하고, React 앱을 로컬에서 실행합니다.

React 앱을 빌드하고 실행하려면

1. 빌드 시작를 선택합니다.
2. 빌드가 완료되면 Amazon S3 프로젝트 아티팩트 버킷으로 이동하여 React 앱 아티팩트를 다운로드합니다.
3. 프로젝트 폴더에서 React 빌드 아티팩트 및 `run npm install -g serve && serve -s build`의 압축을 풉니다.
4. `serve` 명령은 로컬 포트의 정적 사이트를 제공하고 터미널로 출력을 인쇄합니다. 터미널 출력의 `Local:`에서 localhost URL을 방문하여 React 앱을 볼 수 있습니다.

React 기반 서버의 배포를 처리하는 방법에 대한 자세한 내용은 React 앱 배포 생성을 참조하세요.

인프라 정리

이 자습서에서 사용한 리소스에 대한 추가 요금을 피하려면 CodeBuild 프로젝트에 대해 생성된 리소스를 삭제합니다.

인프라를 정리하려면

1. 프로젝트 아티팩트 Amazon S3 버킷 삭제
2. CloudWatch 콘솔로 이동하여 CodeBuild 프로젝트와 연결된 CloudWatch 로그 그룹을 삭제합니다.
3. CodeBuild 콘솔로 이동하여 빌드 프로젝트 삭제를 선택하여 CodeBuild 프로젝트를 삭제합니다.

CodeBuild Lambda Python으로 Lambda 함수 구성 업데이트

다음 Python 샘플은 Boto3 및 CodeBuild Lambda Python을 사용하여 Lambda 함수의 구성을 업데이트합니다. 이 샘플을 확장하여 다른 AWS 리소스를 프로그래밍 방식으로 관리할 수 있습니다. 자세한 내용은Boto3 설명서를 참조하십시오.

사전 조건

계정에서 Lambda 함수를 생성하거나 찾습니다.

이 샘플에서는 계정에 Lambda 함수를 이미 생성했다고 가정하고 CodeBuild를 사용하여 Lambda 함수의 환경 변수를 업데이트합니다. CodeBuild를 통해 Lambda 함수를 설정하는 방법에 대한 자세한 내용은 CodeBuild Lambda Java와 AWS SAM 함께를 사용하여 Lambda 함수 배포 샘플을 참조하거나 AWS Lambda를 참조하세요

소스 리포지토리 설정

Boto3 python 스크립트를 저장할 소스 리포지토리를 생성합니다.

소스 리포지토리를 설정하려면

1. 다음 python 스크립트를 `update_lambda_environment_variables.py`라는 새 파일에 복사합니다.

```
import boto3
from os import environ
```

```
def update_lambda_env_variable(lambda_client):
    lambda_function_name = environ['LAMBDA_FUNC_NAME']
    lambda_env_variable = environ['LAMBDA_ENV_VARIABLE']
    lambda_env_variable_value = environ['LAMBDA_ENV_VARIABLE_VALUE']
    print("Updating lambda function " + lambda_function_name + " environment
 variable "
          + lambda_env_variable + " to " + lambda_env_variable_value)
    lambda_client.update_function_configuration(
        FunctionName=lambda_function_name,
        Environment={
            'Variables': {
                lambda_env_variable: lambda_env_variable_value
            }
        },
    )


if __name__ == "__main__":
    region = environ['AWS_REGION']
    client = boto3.client('lambda', region)
    update_lambda_env_variable(client)
```

2. python 파일을 지원되는 소스 리포지토리에 업로드합니다. 지원되는 소스 유형 목록은 ProjectSource를 참조하세요.

CodeBuild Lambda Python 프로젝트 생성

CodeBuild Lambda Python 프로젝트를 생성합니다.

CodeBuild Lambda Java 프로젝트를 생성하려면

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. CodeBuild 정보 페이지가 나타나면 빌드 프로젝트 생성을 선택합니다. 그렇지 않을 경우, 탐색 창에서 빌드를 확장한 후 빌드 프로젝트를 선택하고 빌드 프로젝트 생성을 선택합니다.
3. 프로젝트 이름에 이 빌드 프로젝트의 이름을 입력합니다. 빌드 프로젝트 이름은 각 AWS 계정에서 고유해야 합니다. 또한 선택에 따라 빌드 프로젝트에 대한 설명을 포함하여 다른 사용자가 이 프로젝트의 용도를 이해하도록 도울 수 있습니다.
4. 소스에서 AWS SAM 프로젝트가 위치한 소스 리포지토리를 선택합니다.
5. 환경에서 다음과 같이 합니다.

- 컴퓨팅에서 Lambda를 선택합니다.
- 런타임에서 Python을 선택합니다.
- 이미지에서 aws/codebuild/amazonlinux-x86_64-lambda-standard:python3.12를 선택합니다.
- 서비스 역할의 경우 새 서비스 역할을 선택한 상태로 둡니다. 역할 이름을 기록해 둡니다. 이 샘플의 뒷부분에서 프로젝트의 IAM 권한을 업데이트할 때 필요합니다.

6. 빌드 프로젝트 생성을 선택합니다.
7. IAM 콘솔(https://console.aws.amazon.com/iam/)을 엽니다.
8. 탐색 창에서 역할을 선택하고 프로젝트와 연결된 서비스 역할을 선택합니다. 빌드 프로젝트를 선택하고 편집, 환경, 서비스 역할을 선택하여 CodeBuild에서 프로젝트 역할을 찾을 수 있습니다.
9. 신뢰 관계 탭을 선택한 후 신뢰 정책 편집을 선택합니다.
10. 다음 인라인 정책을 IAM 역할에 추가합니다. 이는 나중에 AWS SAM 인프라를 배포하는 데 사용됩니다. 자세한 내용은 IAM 사용 설명서의 IAM 자격 증명 권한 추가 및 제거를 참조하세요.

    JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "UpdateLambdaPermissions",
            "Effect": "Allow",
            "Action": [
                "lambda:UpdateFunctionConfiguration"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

프로젝트 buildspec 설정

Lambda 함수를 업데이트하기 위해 스크립트는 buildspec에서 환경 변수를 읽어 Lambda 함수의 이름, 환경 변수 이름 및 환경 변수 값을 찾습니다.

프로젝트 buildspec을 설정하려면

1. CodeBuild 콘솔에서 빌드 프로젝트를 선택한 다음 편집 및 Buildspec을 선택합니다.
2. Buildspec에서 빌드 명령 삽입을 선택한 후 편집기로 전환을 선택합니다.
3. 사전 채워진 빌드 명령을 삭제하고 다음 buildspec에 붙여넣습니다.

```
version: 0.2
env:
  variables:
    LAMBDA_FUNC_NAME: "<lambda-function-name>"
    LAMBDA_ENV_VARIABLE: "FEATURE_ENABLED"
    LAMBDA_ENV_VARIABLE_VALUE: "true"
phases:
  install:
    commands:
       - pip3 install boto3
  build:
    commands:
       - python3 update_lambda_environment_variables.py
```

4. Update buildspec(buildspec 업데이트)을 선택합니다.

Lambda 구성 업데이트

CodeBuild Lambda Python을 사용하여 Lambda 함수의 구성을 자동으로 업데이트합니다.

Lambda 함수의 구성을 업데이트하려면

1. 빌드 시작를 선택합니다.
2. 빌드가 완료되면 Lambda 함수로 이동합니다.
3. 구성을 선택한 후 환경 변수를 선택합니다. 키 `FEATURE_ENABLED` 및 값이 `true`인 새 환경 변수가 표시됩니다.

인프라 정리

이 자습서에서 사용한 리소스에 대한 추가 요금을 피하려면 CodeBuild 프로젝트에 대해 생성된 리소스를 삭제합니다.

인프라를 정리하려면

1. CloudWatch 콘솔로 이동하여 CodeBuild 프로젝트와 연결된 CloudWatch 로그 그룹을 삭제합니다.
2. CodeBuild 콘솔로 이동하여 빌드 프로젝트 삭제를 선택하여 CodeBuild 프로젝트를 삭제합니다.
3. 이 샘플을 위해 Lambda 함수를 생성한 경우 작업 및 함수 삭제를 선택하여 Lambda 함수를 정리합니다.

확장 프로그램

AWS CodeBuild Lambda Python을 사용하여 다른 AWS 리소스를 관리하도록이 샘플을 확장하려는 경우:

- Python 스크립트를 업데이트하여 Boto3를 사용하여 새 리소스를 수정합니다.
- 새 리소스에 대한 권한을 갖도록 CodeBuild 프로젝트와 연결된 IAM 역할을 업데이트합니다.
- 새 리소스와 연결된 새 환경 변수를 buildspec에 추가합니다.

예약 용량 플릿에서 빌드 실행

CodeBuild는 다음과 같은 컴퓨팅 플릿을 제공합니다.

- 온디맨드 플릿
- 예약 용량 플릿

CodeBuild는 온디맨드 플릿을 통해 빌드에 맞는 컴퓨팅 기능을 제공합니다. 빌드가 완료되면 머신이 파괴됩니다. 온디맨드 플릿은 완전 관리형이며 수요 급증을 처리할 수 있는 자동 조정 기능을 포함합니다.

> **Note**
>
> 온디맨드 플릿은 macOS를 지원하지 않습니다.

CodeBuild는 또한 CodeBuild에서 유지 관리하는 Amazon EC2 기반 인스턴스를 포함하는 예약 용량 플릿을 제공합니다. 예약 용량 플릿을 사용하면 빌드 환경을 위한 전용 인스턴스 세트를 구성할 수 있습니다. 이러한 시스템은 유휴 상태로 유지되어 빌드 또는 테스트를 즉시 처리하고 빌드 기간을 단축할

준비가 되어 있습니다. 예약 용량 플릿을 사용하면 머신이 항상 실행되며 프로비저닝되는 한 비용이 계속 발생합니다.

> **Important**
>
> 인스턴스 실행 시간에 관계없이 예약 용량 플릿에서는 인스턴스당 초기 요금이 발생하며, 그 이후에는 추가 관련 비용이 발생할 수 있습니다. 자세한 내용은 https://aws.amazon.com/codebuild/pricing/ 단원을 참조하십시오.

주제

- 예약 용량 플릿 생성
- 모범 사례
- 예약 용량 플릿을 여러 CodeBuild 프로젝트에서 공유할 수 있습니까?
- 속성 기반 컴퓨팅은 어떻게 작동하나요?
- 플릿에 Amazon EC2 인스턴스를 수동으로 지정할 수 있나요?
- 예약 용량 플릿을 지원하는 리전은 어디입니까?
- 예약 용량 macOS 플릿을 구성하려면 어떻게 해야 합니까?
- 예약 용량 플릿에 대한 사용자 지정 Amazon Machine Image(AMI)를 구성하려면 어떻게 해야 하나요?
- 예약 용량 플릿의 제한
- 예약 용량 플릿 속성
- 를 사용한 예약 용량 샘플 AWS CodeBuild

예약 용량 플릿 생성

다음 지침에 따라 예약 용량 플릿을 생성합니다.

예약 용량 플릿 생성

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 컴퓨팅 플릿을 선택한 다음, 플릿 생성을 선택합니다.
3. 컴퓨팅 플릿 이름 텍스트 필드에 플릿의 이름을 입력합니다.
4. 운영 체제 드롭다운 메뉴에서 운영 체제를 선택합니다.

5. 아키텍처 드롭다운 메뉴에서 아키텍처를 선택합니다.
6. (선택 사항) Docker 컨테이너 대신 Amazon EC2 인스턴스에서 직접 실행하려면 인스턴스 실행 모드 사용 - 선택 사항을 선택합니다. 그런 다음 메이저 버전과 마이너 버전을 선택합니다.
7. (선택 사항) 추가 구성에서 다음을 수행합니다.

   - VPC 구성 - 선택 사항을 선택하여 사용 중에 프라이빗 리소스에 액세스하기 위해 플릿을 VPC에 연결합니다.
     - VPC 드롭다운 메뉴에서 CodeBuild 플릿이 액세스할 VPC를 선택합니다.
     - 서브넷 드롭다운 메뉴에서 CodeBuild가 VPC 구성을 설정하는 데 사용할 서브넷을 선택합니다.
     - 보안 그룹 드롭다운 메뉴에서 CodeBuild가 VPC와 함께 작동하는 데 사용해야 하는 보안 그룹을 선택합니다.
     - 플릿 서비스 역할 필드에서 기존 서비스 역할을 선택합니다.

     > **Note**
     >
     > 플릿 역할에 필요한 권한이 있는지 확인합니다. 자세한 내용은 사용자가 플릿 서비스 역할에 대한 권한 정책을 추가하도록 허용 단원을 참조하십시오.

     - Amazon Linux 운영 체제를 선택한 경우 프록시 구성 정의 - 선택 사항을 선택하여 예약 용량 인스턴스에 네트워크 액세스 제어를 적용합니다.
     - 기본 동작에서 기본적으로 모든 대상에 대한 발신 트래픽을 허용하거나 거부하도록 선택합니다.
     - 프록시 규칙에서 프록시 규칙 추가를 선택하여 네트워크 액세스 제어를 허용하거나 거부할 대상 도메인 또는 IP를 지정합니다.
   - 사용자 지정 Amazon Machine Image(AMI)를 사용하려면 사용자 지정 AMI 구성 - 선택 사항을 선택합니다.
     - AMI 드롭다운 메뉴에서 플릿의 Amazon Machine Image(AMI)를 선택합니다.
     - 플릿 서비스 역할 필드에서 기존 서비스 역할을 선택합니다.

     > **Note**
     >
     > 플릿 역할에 필요한 권한이 있는지 확인합니다. 자세한 내용은 사용자가 플릿 서비스 역할에 대한 권한 정책을 추가하도록 허용 단원을 참조하십시오.

8. 용량 구성의 컴퓨팅 선택 모드에서 다음 중 하나를 선택합니다.

- 안내형 선택을 선택하는 경우 다음을 수행합니다.
  - 컴퓨팅에서 이 플릿에 포함된 인스턴스 유형을 선택합니다.
  - 용량 텍스트 필드에 플릿의 최소 인스턴스 수를 입력합니다.
  - (선택 사항) 추가 구성에서 다음을 수행합니다.
    - 이 구성을 기반으로 플릿 규모를 자동으로 조정하려면 조정 구성 - 선택 사항을 선택합니다. 조정 모드 - 선택 사항 드롭다운 메뉴에서 수요가 플릿 용량을 초과할 때의 동작을 선택합니다.
- 사용자 지정 인스턴스를 선택하는 경우 다음을 수행합니다.
  - 컴퓨팅 인스턴스 유형 드롭다운 메뉴에서 이 플릿에 포함된 인스턴스 유형을 선택합니다.
  - 추가 EBS 볼륨 크기 - 선택 사항 텍스트 필드에 제공된 64GB의 디스크 공간에 추가된 볼륨을 입력합니다.
  - 용량 텍스트 필드에 플릿의 최소 인스턴스 수를 입력합니다.
  - (선택 사항) 추가 구성에서 다음을 수행합니다.
    - 이 구성을 기반으로 플릿 규모를 자동으로 조정하려면 조정 구성 - 선택 사항을 선택합니다. 조정 모드 - 선택 사항 드롭다운 메뉴에서 수요가 플릿 용량을 초과할 때의 동작을 선택합니다.

9. 컴퓨팅 플릿 생성을 선택합니다.
10. 컴퓨팅 플릿이 생성된 후 새 CodeBuild 프로젝트를 만들거나 기존 프로젝트를 편집합니다. 환경에서 프로비저닝 모델의 예약 용량을 선택한 다음 플릿 이름에서 지정된 플릿을 선택합니다.

모범 사례

예약 용량 플릿을 사용할 때는 다음 모범 사례를 따르는 것이 좋습니다.

- 소스를 캐싱하여 빌드 성능을 향상시키려면 소스 캐시 모드를 사용하는 것이 좋습니다.
- 기존 Docker 계층을 캐싱하여 빌드 성능을 개선하려면 Docker 계층 캐싱을 사용하는 것이 좋습니다.

예약 용량 플릿을 여러 CodeBuild 프로젝트에서 공유할 수 있습니까?

예. 여러 프로젝트에서 플릿을 사용하여 플릿의 용량 활용도를 극대화할 수 있습니다.

> **Important**
>
> 예약 용량 기능을 사용할 때 소스 파일, Docker 계층, 빌드스펙에 지정된 캐시된 디렉터리를 포함하여 플릿 인스턴스에 캐시된 데이터를 동일한 계정 내의 다른 프로젝트에서 액세스할 수 있습니다. 이는 의도적으로 설계된 것이며 동일한 계정 내의 프로젝트가 플릿 인스턴스를 공유할 수 있도록 허용합니다.

속성 기반 컴퓨팅은 어떻게 작동하나요?

플릿의 `computeType`로 `ATTRIBUTE_BASED_COMPUTE`를 선택하면 `computeConfiguration`이라는 새로운 필드에 속성을 지정할 수 있습니다. 이러한 속성에는 vCPU, 메모리, 디스크 공간 및 `machineType`이 포함됩니다. 이 `machineType`는 `GENERAL` 또는 `NVME`입니다. 사용 가능한 속성 중 하나 또는 여러 개를 지정한 후 CodeBuild는 사용 가능한 지원되는 인스턴스 유형에서 컴퓨팅 유형을 완료된 `computeConfiguration`으로 선택합니다.

> **Note**
>
> CodeBuild는 모든 입력 요구 사항과 일치하는 가장 저렴한 인스턴스를 선택합니다. 선택한 인스턴스의 메모리, vCPU 및 디스크 공간은 모두 입력 요구 사항보다 크거나 같습니다. 생성되거나 업데이트된 플릿에서 확인된 `computeConfiguration`을 확인할 수 있습니다.

CodeBuild에서 충족할 수 없는 `computeConfiguration`을 입력하면 검증 예외가 발생합니다. 또한, `computeConfiguration`을 온디맨드로 사용할 수 없는 경우 온디맨드 플릿 오버플로 동작이 대기열 동작으로 재정의된다는 점에 유의하세요.

플릿에 Amazon EC2 인스턴스를 수동으로 지정할 수 있나요?

예. 사용자 지정 인스턴스를 선택하거나 `InstanceType` API 파라미터를 구성하여 콘솔에 원하는 Amazon EC2 인스턴스를 직접 입력할 수 있습니다. 이 필드는 CreateFleet, UpdateFleet, CreateProject, UpdateProject, StartBuild API에서 사용됩니다. 자세한 내용은 Compute instance type 단원을 참조하십시오.

예약 용량 플릿을 지원하는 리전은 어디입니까?

예약 용량 Amazon Linux 및 Windows 플릿은 AWS 리전미국 동부(버지니아 북부), 미국 동부(오하이오), 미국 서부(오레곤), 아시아 태평양(뭄바이), 아시아 태평양(싱가포르), 아시아 태평양(시드니),

아시아 태평양(도쿄), 유럽(프랑크푸르트), 유럽(아일랜드), 남아메리카(상파울루)에서 지원됩니다. CodeBuild가 사용 가능한 AWS 리전 에 대한 자세한 내용은 리전별AWS 서비스를 참조하세요.

예약 용량 macOS Medium 플릿은 AWS 리전미국 동부(버지니아 북부), 미국 동부(오하이오), 미국 서부(오레곤), 아시아 태평양(시드니), 유럽(프랑크푸르트)에서 지원됩니다. 예약 용량 macOS 대규모 플릿은 AWS 리전미국 동부(버지니아 북부), 미국 동부(오하이오), 미국 서부(오레곤), 아시아 태평양(시드니)에서 지원됩니다.

예약 용량 macOS 플릿을 구성하려면 어떻게 해야 합니까?

예약 용량 macOS 플릿을 구성하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 컴퓨팅 플릿을 선택한 다음, 플릿 생성을 선택합니다.
3. 컴퓨팅 플릿 이름 텍스트 필드에 플릿의 이름을 입력합니다.
4. 운영 체제 드롭다운 메뉴에서 macOS를 선택합니다.
5. 컴퓨팅 필드에서 Apple M2, 24GB 메모리, 8 vCPU 또는 Apple M2, 32GB 메모리, 12 vCPU 컴퓨팅 머신 유형 중 하나를 선택합니다.
6. 용량 텍스트 필드에 플릿의 최소 인스턴스 수를 입력합니다.
7. (선택 사항) 플릿에 사용자 지정 이미지를 사용하려면 예약 용량 플릿에 대한 사용자 지정 Amazon Machine Image(AMI)를 구성하려면 어떻게 해야 하나요?의 내용을 참조하여 Amazon Machine Image(AMI)에 필요한 사전 요구 사항이 있는지 확인합니다.
8. (선택 사항) 플릿으로 VPC를 구성하려면 추가 구성에서 다음을 수행합니다.
   - VPC - 선택 사항 드롭다운 메뉴에서 CodeBuild 플릿이 액세스할 VPC를 선택합니다.
   - 서브넷 드롭다운 메뉴에서 CodeBuild가 VPC 구성을 설정하는 데 사용할 서브넷을 선택합니다.
   - 보안 그룹 드롭다운 메뉴에서 CodeBuild가 VPC와 함께 작동하는 데 사용해야 하는 보안 그룹을 선택합니다.
   - 플릿 서비스 역할 필드에서 기존 서비스 역할을 선택합니다.

> **Note**
>
> 플릿 역할에 필요한 권한이 있는지 확인합니다. 자세한 내용은 사용자가 플릿 서비스 역할에 대한 권한 정책을 추가하도록 허용 단원을 참조하십시오.

9. 컴퓨팅 플릿 생성을 선택하고 플릿 인스턴스가 시작될 때까지 기다립니다. 시작되면 용량은 *n*/*n*이며, 여기서 *n*은 제공된 용량입니다.
10. 컴퓨팅 플릿이 시작된 후 새 CodeBuild 프로젝트를 만들거나 기존 프로젝트를 편집합니다. 환경에서 프로비저닝 모델의 예약 용량을 선택한 다음 플릿 이름에서 지정된 플릿을 선택합니다.

예약 용량 플릿에 대한 사용자 지정 Amazon Machine Image(AMI)를 구성하려면 어떻게 해야 하나요?

예약 용량 플릿에 대한 사용자 지정 Amazon Machine Image(AMI)를 구성하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 컴퓨팅 플릿을 선택한 다음, 플릿 생성을 선택합니다.
3. 컴퓨팅 플릿 이름 텍스트 필드에 플릿의 이름을 입력합니다.
4. 플릿에 대해 사용자 지정 이미지를 선택하고 Amazon Machine Image(AMI)에 다음과 같은 사전 조건이 있는지 확인합니다.

   - 환경 유형이 `MAC_ARM`인 경우 AMI 아키텍처가 64비트 `Mac-Arm`인지 확인합니다.
   - 환경 유형이 `LINUX_EC2`인 경우 AMI 아키텍처가 64비트 `x86`인지 확인합니다.
   - 환경 유형이 `ARM_EC2`인 경우 AMI 아키텍처가 64비트 `Arm`인지 확인합니다.
   - 환경 유형이 `WINDOWS_EC2`인 경우 AMI 아키텍처가 64비트 `x86`인지 확인합니다.
   - AMI는 CodeBuild 서비스 조직 ARN을 허용합니다. 조직 ARN 목록은 Amazon Machine Images (AMI) 섹션을 참조하세요.
   - AMI가 AWS KMS 키로 암호화된 경우 AWS KMS 키는 CodeBuild 서비스 조직 ID도 허용해야 합니다. 조직 ID 목록은 Amazon Machine Images (AMI) 섹션을 참조하세요. AWS KMS 키에 대한 자세한 내용은 Amazon EC2 사용 설명서의 조직 및 OUs하세요. CodeBuild 조직에 KMS 키를 사용할 수 있는 권한을 부여하려면 다음 명령문을 키 정책에 추가합니다.

```
{
    "Sid": "Allow access for organization root",
    "Effect": "Allow",
    "Principal": "*",
    "Action": [
        "kms:Describe*",
        "kms:List*",
        "kms:Get*",
```

```
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:CreateGrant"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:PrincipalOrgID": "o-123example"
            }
        }
    }
```

- 플릿 서비스 역할 필드에서 다음 Amazon EC2 권한을 부여합니다.

  JSON

  ```
  {
      "Version":"2012-10-17",
      "Statement": [
          {
              "Effect": "Allow",
              "Action": [
                 "ec2:DescribeImages",
                 "ec2:DescribeSnapshots"
              ],
              "Resource": "*"
          }
      ]
  }
  ```

예약 용량 플릿의 제한

예약 용량 플릿이 지원하지 않는 몇 가지 사용 사례가 있으며, 이로 인해 영향을 받는 경우에는 온디맨드 플릿을 대신 사용하십시오.

- 예약 용량 플릿은 빌드 사용률 지표를 지원하지 않습니다.
- 예약 용량 macOS 플릿은 디버그 세션을 지원하지 않습니다.

제한과 할당량에 대한 자세한 내용은 컴퓨팅 플릿 섹션을 참조하십시오.

예약 용량 플릿 속성

예약 용량 플릿에는 다음 속성이 포함됩니다. 예약 용량 플릿에 대한 자세한 내용은 예약 용량 플릿에서 빌드 실행 섹션을 참조하세요.

운영 체제

운영 체제입니다. 사용할 수 있는 운영 체제는 다음과 같습니다.

- Amazon Linux
- macOS
- Windows Server 2019
- Windows Server 2022

아키텍처

프로세서 아키텍처. 사용할 수 있는 아키텍처는 다음과 같습니다.

- x86_64
- Arm64

환경 유형

Amazon Linux를 선택할 때 사용할 수 있는 환경 유형입니다. 다음은 사용할 수 있는 환경 키입니다.

- Linux EC2
- Linux GPU

컴퓨팅 인스턴스 유형

플릿 인스턴스의 컴퓨팅 구성입니다.

안내형 선택

vCPU, 메모리 및 디스크 공간 설정을 선택하여 다양한 컴퓨팅 유형을 지정합니다. 리전별 컴퓨팅 유형 가용성에 대한 내용은 예약 용량 환경 유형 정보를 참조하세요.

사용자 지정 인스턴스

원하는 인스턴스 유형을 수동으로 지정합니다.

Capacity

플릿에 할당된 초기 머신 수로, 병렬로 실행할 수 있는 빌드 수를 정의합니다.

오버플로 동작

빌드 수가 플릿 용량을 초과할 때의 동작을 정의합니다.

온디맨드

오버플로 빌드는 CodeBuild 온디맨드에서 실행됩니다.

> **Note**
>
> VPC 연결 플릿을 생성하는 동안 오버플로 동작을 온디맨드로 설정하도록 선택한 경우 프로젝트 서비스 역할에 필요한 VPC 권한을 추가해야 합니다. 자세한 내용은 VPC 네트워크 인터페이스를 생성하는 데 필요한 AWS 서비스에 CodeBuild 액세스를 허용하는 정책 설명 예제를 참조하세요.

> **Important**
>
> 오버플로 동작을 온디맨드로 설정하도록 선택하면 온디맨드 Amazon EC2와 마찬가지로 오버플로 빌드에 별도로 요금이 청구됩니다. 자세한 내용은 https://aws.amazon.com/codebuild/pricing/ 단원을 참조하십시오.

대기열

머신을 사용할 수 있을 때까지 빌드 실행이 대기열에 배치됩니다. 이렇게 하면 추가 머신이 할당되지 않으므로 추가 비용이 제한됩니다.

Amazon Machine Image(AMI)

플릿의 Amazon Machine Image(AMI) 속성입니다. CodeBuild에서 다음 속성이 지원됩니다.

AWS 리전	조직 ARN	조직 ID
us-east-1	arn:aws:organizations::851725618577:organization/o-c6wcu152r1	o-c6wcu152r1
us-east-2	arn:aws:organizations::992382780434:organization/o-seufr2suvq	o-seufr2suvq

AWS 리전	조직 ARN	조직 ID
us-west-2	arn:aws:organizations::381491982620:organization/o-0412o99a4r	o-0412o99a4r
ap-northeast-1	arn:aws:organizations::891376993293:organization/o-b6k3sjqavm	o-b6k3sjqavm
ap-south-1	arn:aws:organizations::891376924779:organization/o-krtah1lkeg	o-krtah1lkeg
ap-southeast-1	arn:aws:organizations::654654522137:organization/o-mcn8uvc3tp	o-mcn8uvc3tp
ap-southeast-2	arn:aws:organizations::767398067170:organization/o-6crt0f6bu4	o-6crt0f6bu4
eu-central-1	arn:aws:organizations::590183817084:organization/o-lb2lne3te6	o-lb2lne3te6
eu-west-1	arn:aws:organizations::891376938588:organization/o-ullrrg5qf0	o-ullrrg5qf0

AWS 리전	조직 ARN	조직 ID
sa-east-1	arn:aws:organizations::533267309133:organization/o-db63c45ozw	o-db63c45ozw

추가 구성

VPC - 선택 사항

CodeBuild 플릿이 액세스할 VPC입니다. 자세한 내용은 Amazon Virtual Private Cloud AWS CodeBuild 와 함께 사용 단원을 참조하십시오.

> **Note**
>
> StartBuild API를 직접 호출할 때 플릿 재정의가 지정되면 CodeBuild는 프로젝트 VPC 구성을 무시합니다.

서브넷

CodeBuild가 VPC 구성을 설정하는 데 사용하는 VPC 서브넷입니다. 예약 용량 플릿은 단일 가용 영역에서 하나의 서브넷만 지원합니다. 또한 서브넷에 NAT 게이트웨이가 포함되어 있는지 확인합니다.

보안 그룹

CodeBuild가 VPC와 함께 사용하는 VPC 보안 그룹입니다. 보안 그룹이 아웃바운드 연결을 허용하는지 확인합니다.

플릿 서비스 역할

계정의 기존 서비스 역할에서 플릿의 서비스 역할을 정의합니다.

프록시 구성 정의 - 선택 사항

예약 용량 인스턴스에 네트워크 액세스 제어를 적용하는 프록시 구성입니다. 자세한 내용은 관리형 프록시 서버와 AWS CodeBuild 함께 사용 단원을 참조하십시오.

> **Note**
>
> 프록시 구성은 VPC, Windows 또는 macOS를 지원하지 않습니다.

기본 동작

발신 트래픽의 동작을 정의합니다.

허용

기본적으로 모든 대상으로 전송되는 트래픽을 허용합니다.

거부

기본적으로 모든 대상으로 전송되는 트래픽을 거부합니다.

프록시 규칙

네트워크 액세스 제어를 허용하거나 거부할 대상 도메인 또는 IP를 지정합니다.

를 사용한 예약 용량 샘플 AWS CodeBuild

이 샘플은 CodeBuild 내 예약 용량 플릿을 실험하는 데 사용할 수 있습니다.

주제

- 예약 용량 샘플을 사용한 캐싱

예약 용량 샘플을 사용한 캐싱

캐시는 빌드 환경에서 재사용할 수 있는 정보를 저장하여 여러 빌드에 사용할 수 있습니다. 이 샘플은 예약 용량을 사용하여 빌드 프로젝트 내에서 캐싱을 활성화하는 방법을 보여주었습니다. 자세한 내용은 성능을 개선하기 위한 캐시 빌드 단원을 참조하십시오.

프로젝트 설정에서 하나 이상의 캐시 모드를 지정하여 시작할 수 있습니다.

```
Cache:
        Type: LOCAL
        Modes:
          - LOCAL_CUSTOM_CACHE
```

```
        - LOCAL_DOCKER_LAYER_CACHE
        - LOCAL_SOURCE_CACHE
```

> **Note**
>
> Docker 계층 캐시를 사용하려면 권한 모드를 활성화해야 합니다.

프로젝트 buildspec 설정이 다음과 같아야 합니다.

```
version: 0.2
      phases:
        build:
          commands:
            - echo testing local source cache
            - touch /codebuild/cache/workspace/foobar.txt
            - git checkout -b cached_branch
            - echo testing local docker layer cache
            - docker run alpine:3.14 2>&1 | grep 'Pulling from' || exit 1
            - echo testing local custom cache
            - touch foo
            - mkdir bar && ln -s foo bar/foo2
            - mkdir bar/bar && touch bar/bar/foo3 && touch bar/bar/foo4
            - "[ -f foo ] || exit 1"
            - "[ -L bar/foo2 ] || exit 1"
            - "[ -f bar/bar/foo3 ] || exit 1"
            - "[ -f bar/bar/foo4 ] || exit 1"
      cache:
        paths:
           - './foo'
           - './bar/**/*'
           - './bar/bar/foo3'
```

캐시를 시드하는 새 프로젝트로 빌드를 실행하여 시작할 수 있습니다. 작업이 완료되면 다음과 같이 buildspec을 재정의하는 다른 빌드를 시작해야 합니다.

```
version: 0.2
      phases:
        build:
          commands:
            - echo testing local source cache
            - git branch | if grep 'cached_branch'; then (exit 0); else (exit 1); fi
```

```
            - ls /codebuild/cache/workspace | if grep 'foobar.txt'; then (exit 0); else
(exit 1); fi
            - echo testing local docker layer cache
            - docker run alpine:3.14 2>&1 | if grep 'Pulling from'; then (exit 1); else
(exit 0); fi
            - echo testing local custom cache
            - "[ -f foo ] || exit 1"
            - "[ -L bar/foo2 ] || exit 1"
            - "[ -f bar/bar/foo3 ] || exit 1"
            - "[ -f bar/bar/foo4 ] || exit 1"
      cache:
        paths:
           - './foo'
           - './bar/**/*'
           - './bar/bar/foo3'
```

배치로 빌드 실행

AWS CodeBuild 를 사용하여 배치 빌드가 포함된 프로젝트의 동시 및 조정된 빌드를 실행할 수 있습니다.

주제

- 보안 역할
- 배치 빌드 유형
- 배치 보고서 모드
- 추가 정보

보안 역할

배치 빌드는 배치 구성에 새로운 보안 역할을 도입합니다. CodeBuild가 사용자 대신 `StartBuild`, `StopBuild` 및 `RetryBuild` 작업을 호출하여 배치의 일부로 빌드를 실행할 수 있어야 하므로 이 새 역할이 필요합니다. 고객은 다음과 같은 두 가지 이유로 빌드에 사용하는 것과 동일한 역할이 아닌 새 역할을 사용해야 합니다.

- 빌드 역할 `StartBuild`, `StopBuild` 및 `RetryBuild` 권한을 부여하면 단일 빌드에서 buildspec을 통해 더 많은 빌드를 시작할 수 있습니다.
- CodeBuild 배치 빌드는 배치의 빌드에 사용할 수 있는 빌드 및 컴퓨팅 유형의 수를 제한하는 제한을 제공합니다. 빌드 역할에 이러한 권한이 있는 경우 빌드 자체가 이러한 제한을 우회할 수 있습니다.

배치 빌드 유형

CodeBuild는 다음과 같은 배치 빌드 유형을 지원합니다.

배치 빌드 유형

- 빌드 그래프
- 빌드 목록
- 빌드 매트릭스
- 빌드 팬아웃

빌드 그래프

빌드 그래프는 일괄 처리의 다른 태스크에 종속되는 일련의 태스크를 정의합니다.

다음 예제는 종속성 체인을 생성하는 빌드 그래프를 정의합니다.

```
batch:
  fast-fail: false
  build-graph:
    - identifier: build1
      env:
        variables:
          BUILD_ID: build1
      ignore-failure: false
    - identifier: build2
      buildspec: build2.yml
      env:
        variables:
          BUILD_ID: build2
      depend-on:
        - build1
    - identifier: build3
      env:
        variables:
          BUILD_ID: build3
      depend-on:
        - build2
    - identifier: build4
      env:
        compute-type: ARM_LAMBDA_1GB
```

```
    - identifier: build5
      env:
        fleet: fleet_name
```

이 예제에서는 다음이 적용됩니다.

- 종속성이 없으므로 build1이 먼저 실행됩니다.
- build2는 build1에 종속되어 있으므로 build2는 build1 완료 후에 실행됩니다.
- build3는 build2에 종속되어 있으므로 build3는 build2 완료 후에 실행됩니다.

빌드 그래프 buildspec 구문에 대한 자세한 내용은 batch/build-graph 섹션을 참조하세요.

빌드 목록

빌드 목록은 병렬로 실행되는 여러 태스크를 정의합니다.

다음 예제에서는 빌드 목록을 정의합니다. build1 및 build2 빌드는 병렬로 실행됩니다.

```
batch:
  fast-fail: false
  build-list:
    - identifier: build1
      env:
        variables:
          BUILD_ID: build1
      ignore-failure: false
    - identifier: build2
      buildspec: build2.yml
      env:
        variables:
          BUILD_ID: build2
      ignore-failure: true
    - identifier: build3
      env:
        compute-type: ARM_LAMBDA_1GB
    - identifier: build4
      env:
        fleet: fleet_name
    - identifier: build5
      env:
        compute-type: GENERAL_LINUX_XLAGRE
```

빌드 목록 buildspec 구문에 대한 자세한 내용은 batch/build-list 섹션을 참조하세요.

빌드 매트릭스

빌드 매트릭스는 병렬로 실행되는 다양한 구성의 태스크를 정의합니다. CodeBuild는 가능한 각 구성 조합에 대해 별도의 빌드를 생성합니다.

다음 예제는 buildspec 파일 2개와 환경 변수 값 3개가 포함된 빌드 매트릭스를 보여 줍니다.

```
batch:
  build-matrix:
    static:
      ignore-failure: false
    dynamic:
      buildspec:
        - matrix1.yml
        - matrix2.yml
      env:
        variables:
          MY_VAR:
            - VALUE1
            - VALUE2
            - VALUE3
```

이 예제에서 CodeBuild는 6개의 빌드를 생성합니다.

- $MY_VAR=VALUE1가 있는 matrix1.yml
- $MY_VAR=VALUE2가 있는 matrix1.yml
- $MY_VAR=VALUE3가 있는 matrix1.yml
- $MY_VAR=VALUE1가 있는 matrix2.yml
- $MY_VAR=VALUE2가 있는 matrix2.yml
- matrix2.yml가 있는 $MY_VAR=VALUE3

각 빌드에는 다음과 같은 설정이 있습니다.

- ignore-failure가 false로 설정됨
- env/type이 LINUX_CONTAINER로 설정됨

- env/image이 aws/codebuild/amazonlinux-x86_64-standard:4.0로 설정됨
- env/privileged-mode가 true로 설정됨

이러한 빌드는 병렬로 실행됩니다.

매트릭스 buildspec 구문에 대한 자세한 내용은 batch/build-matrix 섹션을 참조하세요.

빌드 팬아웃

빌드 팬아웃은 배치에서 여러 빌드로 분할되는 작업을 정의합니다. 이는 테스트를 병렬로 실행하는 데 사용할 수 있습니다. CodeBuild는 parallelism 필드에 설정된 값을 기반으로 테스트 사례의 각 샤드에 대해 별도의 빌드를 생성합니다.

다음 예제에서는 병렬로 실행되는 5개의 빌드를 생성하는 빌드 팬아웃을 정의합니다.

```
version: 0.2

batch:
   fast-fail: false
   build-fanout:
     parallelism: 5
     ignore-failure: false

phases:
  install:
    commands:
      - npm install
   build:
    commands:
      - mkdir -p test-results
      - cd test-results
      - |
        codebuild-tests-run \
         --test-command 'npx jest --runInBand --coverage' \
         --files-search "codebuild-glob-search '**/test/**/*.test.js'" \
         --sharding-strategy 'equal-distribution'
```

이 예제에서는 실행해야 하는 테스트가 100개 있다고 가정하면 CodeBuild는 각각 20개의 테스트를 병렬로 실행하는 5개의 빌드를 생성합니다.

빌드 그래프 buildspec 구문에 대한 자세한 내용은 batch/build-fanout 섹션을 참조하세요.

배치 보고서 모드

프로젝트의 소스 공급자가 Bitbucket, GitHub 또는 GitHub Enterprise이고 프로젝트가 소스 공급자에게 빌드 상태를 보고하도록 구성된 경우 배치 빌드 상태를 소스 공급자에게 보내는 방법을 선택할 수 있습니다. 상태를 배치에 대한 단일 집계 상태 보고서로 전송하거나, 배치에 있는 각 빌드의 상태를 개별적으로 보고하도록 선택할 수 있습니다.

자세한 내용은 다음 항목을 참조하세요.

- 빌드 구성(생성)
- 빌드 구성(업데이트)

추가 정보

자세한 내용은 다음 항목을 참조하세요.

- 배치 빌드 buildspec 참조
- 배치 구성
- 배치 빌드 실행(AWS CLI)
- 에서 배치 빌드 중지 AWS CodeBuild

배치 빌드에서 병렬 테스트 실행

AWS CodeBuild 를 사용하여 배치 빌드에서 병렬 테스트를 실행할 수 있습니다. 병렬 테스트 실행은 순차적으로 실행하는 대신 다양한 환경, 시스템 또는 브라우저에서 여러 테스트 사례를 동시에 실행하는 테스트 접근 방식입니다. 이 접근 방식을 사용하면 전체 테스트 실행 시간을 크게 줄이고 테스트 효율성을 개선할 수 있습니다. CodeBuild에서는 테스트를 여러 환경으로 분할하고 동시에 실행할 수 있습니다.

병렬 테스트 실행의 주요 이점은 다음과 같습니다.

1. 실행 시간 단축 - 순차적으로 진행하면 몇 시간이 걸리는 테스트를 몇 분 만에 완료할 수 있습니다.
2. 리소스 사용률 향상 - 사용 가능한 컴퓨팅 리소스를 효율적으로 사용합니다.
3. 이른 피드백 - 테스트 완료 속도가 빨라지면 개발자에게 더 빠른 피드백을 제공할 수 있습니다.
4. 비용 효율성 - 장기적으로 시간과 컴퓨팅 비용을 모두 절감합니다.

병렬 테스트 실행을 구현할 때는 일반적으로 별도의 환경과 멀티스레딩이라는 2가지 주요 접근 방식을 고려합니다. 두 방법 모두 동시 테스트 실행을 달성하는 것을 목표로 하지만, 구현과 효율성은 크게 다릅니다. 별도의 환경은 각 테스트 제품군이 독립적으로 실행되는 격리된 인스턴스를 생성하는 반면, 멀티스레딩은 서로 다른 스레드를 사용하여 동일한 프로세스 공간 내에서 여러 테스트를 동시에 실행합니다.

멀티스레딩이 아닌 별도의 환경 사용 시 얻을 수 있는 주요 이점은 다음과 같습니다.

1. 격리 - 각 테스트는 완전히 격리된 환경에서 실행되어 테스트 간 간섭을 방지합니다.
2. 리소스 충돌 - 멀티스레딩에서 자주 발생하는 공유 리소스에 대한 경쟁이 없습니다.
3. 안정성 - 경쟁 조건 및 동기화 문제에 덜 취약합니다.
4. 더 쉬운 디버깅 - 테스트가 실패하면 각 환경이 독립적이므로 원인을 식별하기가 더 쉽습니다.
5. 상태 관리 - 멀티스레드 테스트에 부정적인 영향을 주는 공유 상태 문제를 편하게 관리할 수 있습니다.
6. 확장성 향상 - 복잡성 없이 더 많은 환경을 쉽게 추가할 수 있습니다.

주제

- 에서 지원 AWS CodeBuild
- 배치 빌드에서 병렬 테스트 실행 활성화
- codebuild-tests-run CLI 명령 사용
- codebuild-glob-search CLI 명령 사용
- 테스트 분할 정보
- 개별 빌드 보고서 자동 병합
- 다양한 테스트 프레임워크 샘플에 대한 병렬 테스트 실행

에서 지원 AWS CodeBuild

AWS CodeBuild 는 별도의 환경 실행을 활용하도록 특별히 설계된 배치 빌드 기능을 통해 병렬 테스트 실행을 강력하게 지원합니다. 이 구현은 격리된 테스트 환경의 이점과 완벽하게 일치합니다.

테스트 배포를 사용한 배치 빌드

CodeBuild의 배치 빌드 기능을 사용하면 동시에 실행되는 여러 빌드 환경을 생성할 수 있습니다. 각 환경은 자체 컴퓨팅 리소스, 런타임 환경 및 종속성과 함께 완전히 격리된 단위로 작동합니다. 배치 빌드 구성을 통해 필요한 병렬 환경 수와 이에 걸쳐 테스트를 배포하는 방법을 지정할 수 있습니다.

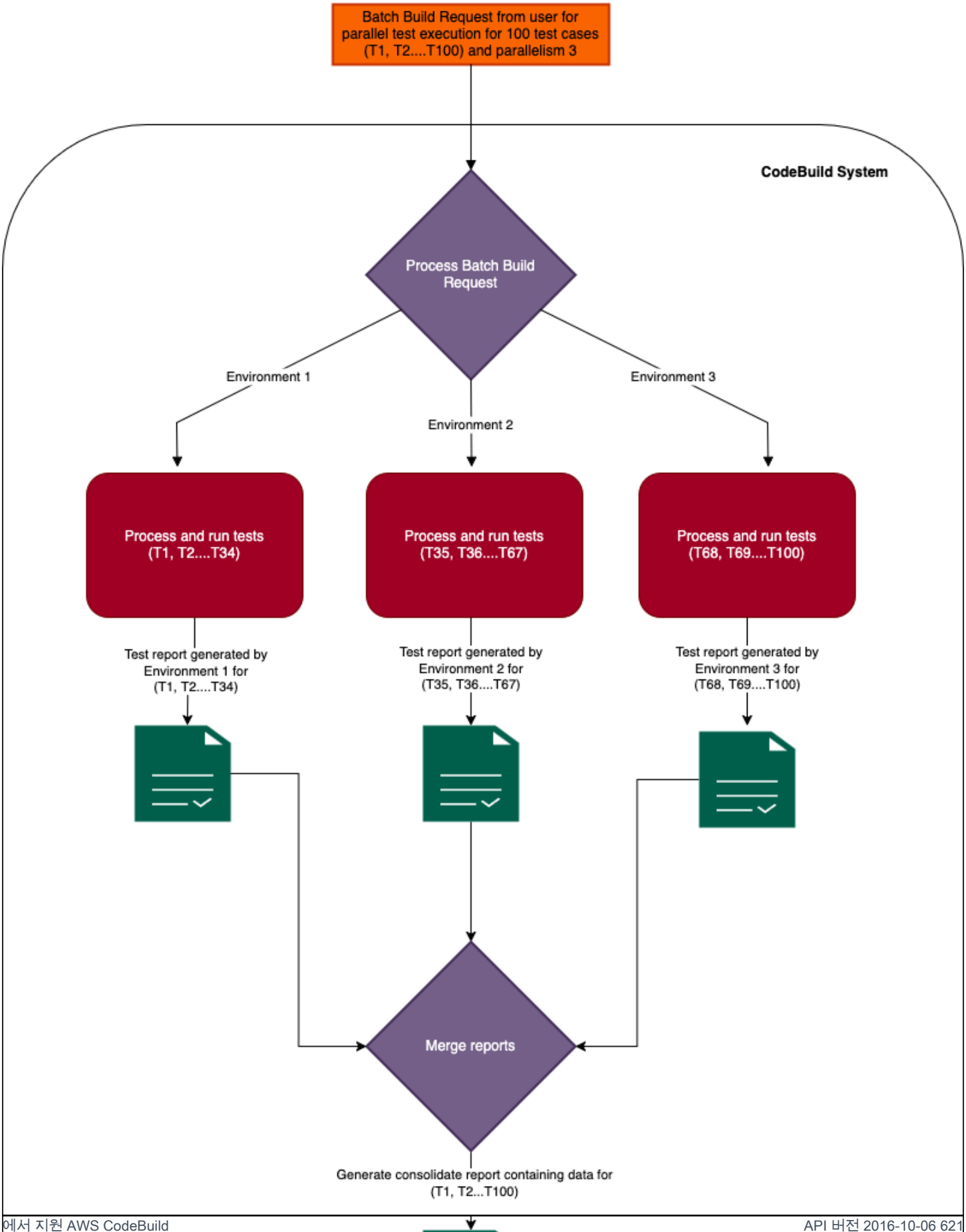
샤딩 CLI 테스트

CodeBuild에는 자동으로 테스트를 다른 환경으로 나누는 CLI 도구인 `codebuild-tests-run`을 통해 내장된 테스트 배포 메커니즘이 포함되어 있습니다.

보고서 집계

CodeBuild 구현의 주요 강점 중 하나는 테스트 결과 집계를 원활하게 처리하는 기능입니다. 별도의 환경에서 테스트가 실행되는 동안 CodeBuild는 각 환경의 테스트 보고서를 자동으로 수집하여 배치 빌드 수준에서 통합 테스트 보고서로 결합합니다. 이 통합은 병렬 실행의 효율성 이점을 유지하면서 테스트 결과에 대한 포괄적인 보기를 제공합니다.

다음 다이어그램은에서 병렬 테스트 실행의 전체 개념을 설명합니다 AWS CodeBuild.


Batch Build Request from user for parallel test execution for 100 test cases (T1, T2....T100) and parallelism 3
CodeBuild System
Process Batch Build Request
Environment 1
Environment 2
Environment 3
Process and run tests (T1, T2....T34)
Process and run tests (T35, T36....T67)
Process and run tests (T68, T69....T100)
Test report generated by Environment 1 for (T1, T2....T34)
Test report generated by Environment 2 for (T35, T36....T67)
Test report generated by Environment 3 for (T68, T69....T100)
Merge reports
Generate consolidate report containing data for (T1, T2...T100)

배치 빌드에서 병렬 테스트 실행 활성화

테스트를 병렬로 실행하려면 아래와 같이 빌드 팬아웃 필드와 병렬 빌드 수를 포함하도록 배치 빌드 buildspec 파일을 업데이트하여 `parallelism` 필드에 테스트 제품군을 분할합니다. `parallelism` 필드는 테스트 제품군을 실행하도록 설정된 독립 실행기 수를 지정합니다.

여러 병렬 실행 환경에서 테스트를 실행하려면 `parallelism` 필드를 0보다 큰 값으로 설정합니다. 아래 예제에서 `parallelism`는 5로 설정되어 있습니다. 즉, CodeBuild는 테스트 제품군의 일부를 병렬로 실행하는 동일한 빌드 5개를 시작합니다.

codebuild-tests-run CLI 명령을 사용하여 테스트를 분할하고 실행할 수 있습니다. 테스트 파일이 분할되고 테스트의 일부가 각 빌드에서 실행됩니다. 이렇게 하면 전체 테스트 제품군을 실행하는 데 걸리는 총 시간이 줄어듭니다. 다음 예제에서는 테스트가 5개로 분할되고 테스트 이름을 기반으로 분할 지점이 계산됩니다.

```
version: 0.2

batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false

phases:
  install:
    commands:
      - npm install jest-junit --save-dev
  pre_build:
    commands:
      - echo 'prebuild'
  build:
    commands:
      - |
        codebuild-tests-run \
         --test-command 'npx jest --runInBand --coverage' \
         --files-search "codebuild-glob-search '**/_tests_/**/*.test.js'" \
         --sharding-strategy 'equal-distribution'

  post_build:
    commands:
      - codebuild-glob-search '**/*.xml'
      - echo "Running post-build steps..."
```

```
      - echo "Build completed on `date`"

reports:
  test-reports:
    files:
      - '**/junit.xml'
    base-directory: .
    discard-paths: yes
    file-format: JUNITXML
```

보고서가 build-fanout 빌드용으로 구성된 경우 각 빌드에 대해 개별적으로 테스트 보고서가 생성되며, 콘솔에서 해당 빌드의 보고서 탭에서 확인할 수 있습니다 AWS CodeBuild .

배치에서 병렬 테스트를 실행하는 방법에 대한 자세한 내용은 다양한 테스트 프레임워크 샘플에 대한 병렬 테스트 실행를 참조하세요.

`codebuild-tests-run` CLI 명령 사용

AWS CodeBuild 는 테스트 명령과 테스트 파일 위치를 입력으로 사용하는 CLI를 제공합니다. 이 입력이 있는 CLI는 테스트 파일 이름을 기반으로 `parallelism` 필드에 지정된 대로 테스트를 샤드 수로 분할합니다. 테스트 파일을 샤드에 할당하는 것은 샤딩 전략에 따라 결정됩니다.

```
codebuild-tests-run \
    --files-search "codebuild-glob-search '**/__tests__/*.js'" \
    --test-command 'npx jest --runInBand --coverage' \
    --sharding-strategy 'equal-distribution'
```

다음 표는 `codebuild-tests-run` CLI 명령의 필드를 설명합니다.

필드 이름	유형	필수 또는 선택	정의
`test-command`	문자열	필수	이 명령은 테스트를 실행하는 데 사용됩니다.
`files-search`	문자열	필수	이 명령은 테스트 파일 목록을 제공합니다. AWS CodeBuild 제공된 codebuild-glob-search CLI 명령 또는 원하는 다른 파일 검색

필드 이름	유형	필수 또는 선택	정의
			도구를 사용할 수 있습니다. Note `files-search` 명령이 각각 새 줄로 구분된 파일 이름을 출력하는지 확인합니다.
`sharding-strategy`	Enum	선택 사항	유효한 값: `equal-distribution` (기본값), `stability` • `equal-distribution` : 테스트 파일 이름을 기반으로 테스트 파일을 균등하게 샤딩합니다. • `stability` : 파일 이름의 일관된 해싱을 사용하여 테스트 파일을 샤딩합니다. 자세한 내용은 테스트 분할 정보 단원을 참조하십시오.

`codebuild-tests-run` CLI는 먼저 `files-search` 파라미터에 제공된 명령을 사용하여 테스트 파일 목록을 식별합니다. 그런 다음 지정된 샤딩 전략을 사용하여 현재 샤드(환경)에 지정된 테스트 파일

의 하위 집합을 결정합니다. 마지막으로 이 테스트 파일의 하위 집합은 공백으로 구분된 목록으로 형식이 지정되고 실행되기 전에 test-command 파라미터에 제공된 명령의 끝에 추가됩니다.

공백으로 구분된 목록을 허용하지 않는 테스트 프레임워크의 경우 codebuild-tests-run CLI는 CODEBUILD_CURRENT_SHARD_FILES 환경 변수를 통해 유연한 대안을 제공합니다. 이 변수에는 현재 빌드 샤드에 지정된 테스트 파일 경로의 줄 바꿈으로 구분된 목록이 포함되어 있습니다. 이 환경 변수를 활용하면 다양한 테스트 프레임워크 요구 사항에 쉽게 적응하여 공백으로 구분된 목록과 다른 입력 형식을 기대하는 바람에 부응할 수 있습니다. 또한, 테스트 프레임워크의 필요에 따라 테스트 파일 이름의 형식을 지정할 수도 있습니다. 다음은 Linux에서 Django 프레임워크와 함께 CODEBUILD_CURRENT_SHARD_FILES를 사용하는 예제입니다. 여기서 CODEBUILD_CURRENT_SHARD_FILES는 Django에서 지원하는 점 표기법 파일 경로를 가져오는 데 사용됩니다.

```
codebuild-tests-run \
    –files-search "codebuild-glob-search '/tests/test_.py'" \
    –test-command 'python3 manage.py test $(echo "$CODEBUILD_CURRENT_SHARD_FILES" | sed
 -E "s/\//__/g; s/\.py$//; s/__/./g")' \
    –sharding-strategy 'equal-distribution'
```

> **Note**
>
> CODEBUILD_CURRENT_SHARD_FILES 환경 변수는 codebuild-tests-run CLI의 범위 내에서만 사용할 수 있습니다.
> 또한, 테스트 명령 내부에서 CODEBUILD_CURRENT_SHARD_FILES를 사용하는 경우 위의 예제와 같이 CODEBUILD_CURRENT_SHARD_FILES를 큰따옴표 안에 넣습니다.

codebuild-glob-search CLI 명령 사용

AWS CodeBuild 는 하나 이상의 glob 패턴을 기반으로 작업 디렉터리의 파일을 검색할 수 codebuild-glob-search 있는 라는 내장 CLI 도구를 제공합니다. 이 도구는 프로젝트 내의 특정 파일 또는 디렉터리에서 테스트를 실행하려는 경우에 특히 유용할 수 있습니다.

용도

codebuild-glob-search CLI에는 다음과 같은 사용 구문이 있습니다.

```
codebuild-glob-search <glob_pattern1> [<glob_pattern2> ...]
```

- *<glob_pattern1>*, *<glob_pattern2>* 등: 작업 디렉터리의 파일과 일치시킬 하나 이상의 glob 패턴입니다.
- *: 모든 문자 시퀀스와 일치합니다(경로 구분자 제외).
- **: 모든 문자 시퀀스와 일치합니다(경로 구분자 포함).

> **Note**
>
> glob 문자열에 따옴표가 있는지 확인합니다. 패턴 일치 결과를 확인하려면 echo 명령을 사용합니다.
>
> ```
> version: 0.2
>
> phases:
>   build:
>     commands:
>       - echo $(codebuild-glob-search '**/__tests__/*.js')
>       - codebuild-glob-search '**/__tests__/*.js' | xargs -n 1 echo
> ```

출력

CLI는 제공된 glob 패턴과 일치하는 파일 경로의 줄 바꿈으로 구분된 목록을 출력합니다. 반환되는 파일 경로는 작업 디렉터리를 기준으로 합니다.

제공된 패턴과 일치하는 파일을 찾을 수 없는 경우 CLI는 파일을 찾을 수 없다는 메시지를 출력합니다.

지정된 패턴으로 인해 발견된 디렉터리는 검색 결과에서 제외됩니다.

예제

`.js` 확장자가 있는 테스트 디렉터리 및 하위 디렉터리 내의 파일만 검색하려면 `codebuild-glob-search` CLI와 함께 다음 명령을 사용할 수 있습니다.

```
codebuild-glob-search '**/__tests__/*.js'
```

이 명령은 패턴으로 표시된 대로 `__tests__` 디렉터리 및 해당 하위 디렉터리 내에 `.js` 확장명이 있는 모든 파일을 검색합니다.

테스트 분할 정보

AWS CodeBuild의 테스트 분할 기능을 사용하면 여러 컴퓨팅 인스턴스에서 테스트 제품군 실행을 병렬화하여 전체 테스트 실행 시간을 줄일 수 있습니다. 이 기능은 CodeBuild 프로젝트 설정의 배치 구성과 buildspec 파일의 `codebuild-tests-run` 유틸리티를 통해 활성화됩니다.

테스트는 지정된 샤딩 전략에 따라 분할됩니다. CodeBuild는 아래와 같이 2가지 샤딩 전략을 제공합니다.

균등 배포

`equal-distribution` 샤딩 전략은 테스트 파일 이름의 알파벳 순서에 따라 병렬 빌드로 테스트를 나눕니다. 이 접근 방식은 먼저 테스트 파일을 정렬한 다음 청크 기반 메서드로 배포하여 유사한 파일이 묶여 테스트되도록 합니다. 비교적 작은 테스트 파일 세트를 처리할 때 권장됩니다. 이 방법은 각 샤드에 최대 차이가 1일 만큼 거의 동일한 수의 파일을 할당하는 것을 목표로 하지만, 안정성을 보장하지는 않습니다. 후속 빌드에서 테스트 파일을 추가하거나 제거하면 기존 파일의 배포가 변경되어 샤드 간에 재할당이 발생할 수 있습니다.

안정성

`stability` 샤딩 전략은 일관된 해싱 알고리즘을 사용하여 샤드 간에 테스트를 분할하므로 파일 배포가 안정적으로 유지됩니다. 새 파일을 추가하거나 제거할 때 이 접근 방식을 사용하면 기존 파일-샤드 할당이 거의 변경되지 않습니다. 대규모 테스트 제품군의 경우 안정성 옵션을 사용하여 샤드 간에 테스트를 균등하게 분산하는 것이 좋습니다. 이 메커니즘은 거의 동일한 배포를 제공하여 각 샤드가 최소한의 분산만으로 비슷한 수의 파일을 수신하도록 하는 것을 목표로 합니다. 안정성 전략은 이상적인 균등 배포를 보장하지는 않지만, 파일이 추가되거나 제거되더라도 빌드 전반의 파일 할당 일관성을 유지하는 거의 균등한 배포를 제공합니다.

테스트 분할을 활성화하려면 CodeBuild 프로젝트 설정에서 배치 섹션을 구성하여 원하는 `parallelism` 수준 및 기타 관련 파라미터를 지정해야 합니다. 또한, 적절한 테스트 명령 및 분할 방법과 함께 buildspec 파일에 `codebuild-tests-run` 유틸리티를 포함해야 합니다.

개별 빌드 보고서 자동 병합

팬아웃 배치 빌드에서는 개별 빌드 보고서를 통합 배치 수준 보고서로 자동 병합할 수 있도록 AWS CodeBuild 지원합니다. 이 기능은 배치 내의 모든 빌드에서 테스트 결과 및 코드 적용 범위를 포괄적으로 보여줍니다.

작동 방식

`fanout` 배치 빌드를 실행할 때 각 개별 빌드는 테스트 보고서를 생성합니다. 그런 다음 CodeBuild는 서로 다른 빌드의 동일한 보고서를 배치 빌드에 연결된 통합 보고서로 자동으로 통합합니다. 이러한 통합 보고서는 BatchGetBuildBatches API의 `reportArns` 필드를 통해 쉽게 액세스할 수 있으며 콘솔의 보고서 탭에서도 볼 수 있습니다. 이 병합 기능은 자동 검색된 보고서로도 확장됩니다.

통합 보고서는 buildspec에 지정되거나 CodeBuild에서 자동으로 검색하는 보고서 그룹에 생성됩니다. 이러한 보고서 그룹에서 바로 병합된 보고서의 추세를 분석하여 동일한 빌드 배치 프로젝트의 과거 빌드에 걸쳐 전체 빌드 성능 및 품질 지표에 대한 귀중한 인사이트를 제공할 수 있습니다.

배치 내의 각 개별 빌드에 대해 CodeBuild는 자동으로 별도의 보고서 그룹을 생성합니다. 이는 특정 명명 규칙을 따르며 배치 빌드 보고서 그룹 이름을 `BuildFanoutShard<shard_number>` 접미사와 결합합니다. 여기서 `shard_number`는 보고서 그룹이 생성되는 샤드의 수를 나타냅니다. 이 조직을 사용하면 통합 및 개별 빌드 수준 모두에서 추세를 추적하고 분석하여 빌드 프로세스를 모니터링하고 평가하는 방법에 유연성을 더할 수 있습니다.

배치 빌드 보고서는 개별 빌드 보고서와 동일한 구조를 따릅니다. 보고서 탭의 다음 키 필드는 배치 빌드 보고서에만 해당됩니다.

배치 빌드 보고서 상태

배치 빌드 보고서의 상태는 보고서 유형에 따라 특정 규칙을 따릅니다.

- 테스트 보고서:
  - 성공: 모든 개별 빌드 보고서가 성공하면 상태가 성공으로 설정됩니다.
  - 실패: 개별 빌드 보고서가 실패한 경우 상태가 실패로 설정됩니다.
  - 미완료: 개별 빌드 보고서가 누락되었거나 미완료 상태인 경우 상태가 미완료로 표시됩니다.
- 코드 적용 범위 보고서:
  - 완료: 모든 개별 빌드 보고서가 완료되면 상태가 완료로 설정됩니다.
  - 실패: 개별 빌드 보고서가 실패한 경우 상태가 실패로 설정됩니다.
  - 미완료: 개별 빌드 보고서가 누락되었거나 미완료 상태인 경우 상태가 미완료로 표시됩니다.

테스트 요약

병합된 테스트 보고서는 모든 개별 빌드 보고서의 다음 필드를 통합합니다.

- duration-in-nano-seconds: 모든 개별 빌드 보고서에서 나노초 단위의 최대 테스트 기간입니다.
- total: 각 빌드의 총 테스트 수를 합산한 모든 테스트 사례의 총 수입니다.

- status-counts: 모든 개별 빌드에서 각 상태 유형의 수를 집계하여 계산된 통과, 실패, 건너뜀 등 테스트 상태에 대한 통합 보기를 제공합니다.

코드 적용 범위 요약

병합된 코드 적용 범위 보고서는 다음 계산을 사용하여 모든 개별 빌드의 필드를 결합합니다.

- branchs-covered: 개별 보고서에서 다루는 모든 브랜치의 합계입니다.
- branches-missed: 개별 보고서에서 누락된 모든 브랜치의 합계입니다.
- branch-coverage-percentage: `(Total covered branches / Total branches) * 100`
- lines-covered: 개별 보고서에서 다루는 모든 줄의 합계입니다.
- lines-missed: 개별 보고서에서 누락된 모든 줄의 합계입니다.
- lines-coverage-percentage: `(Total covered lines / Total lines) * 100`

실행 ID

배치 빌드 ARN입니다.

테스트 사례

병합된 보고서에는 콘솔의 DescribeTestCases API와 배치 빌드 보고서를 통해 액세스할 수 있는 개별 빌드의 모든 테스트 사례에 대한 통합 목록이 포함되어 있습니다.

코드 적용 범위

병합된 코드 적용 범위 보고서는 콘솔의 DescribeCodeCoverages API와 배치 빌드 보고서를 통해 액세스할 수 있는 모든 개별 빌드의 각 파일에 대한 통합 줄 및 브랜치 적용 범위 정보를 제공합니다. 참고: 여러 샤드에 분산된 다양한 테스트 파일이 포함된 파일의 경우 병합된 보고서는 다음 선택 기준을 사용합니다.

1. 기본 선택은 샤드 중에서 가장 높은 줄 적용 범위를 기준으로 합니다.
2. 여러 샤드에서 줄 적용 범위가 같으면 브랜치 적용 범위가 가장 높은 샤드가 선택됩니다.

다양한 테스트 프레임워크 샘플에 대한 병렬 테스트 실행

`codebuild-tests-run` CLI 명령을 사용하여 병렬 실행 환경에서 테스트를 분할하고 실행할 수 있습니다. 다음 섹션에서는 `codebuild-tests-run` 명령 사용을 보여주는 다양한 프레임워크에 대한 `buildspec.yml` 샘플을 제공합니다.

- 아래 각 예제에는 5개의 `parallelism` 수준이 포함되어 있습니다. 즉, 테스트를 분할하기 위해 5개의 동일한 실행 환경이 생성됩니다. `build-fanout` 섹션의 `parallelism` 값을 수정하여 프로젝트에 적합한 `parallelism` 수준을 선택할 수 있습니다.

- 아래 각 예제는 기본값인 테스트 파일 이름으로 분할하도록 테스트를 구성하는 방법을 보여줍니다. 이렇게 하면 병렬 실행 환경에 테스트가 균등하게 분산됩니다.

시작하기 전에 배치 빌드에서 병렬 테스트 실행에서 자세한 내용을 참조하세요.

`codebuild-tests-run` CLI 명령을 사용할 때의 전체 옵션 목록은 `codebuild-tests-run` CLI 명령 사용에서 확인하세요.

주제

- Django를 사용하여 병렬 테스트 구성
- Elixir를 사용하여 병렬 테스트 구성
- Go를 사용하여 병렬 테스트 구성
- Java(Maven)를 사용하여 병렬 테스트 구성
- Javascript(Jest)를 사용하여 병렬 테스트 구성
- Kotlin을 사용하여 병렬 테스트 구성
- PHPUnit을 사용하여 병렬 테스트 구성
- Pytest를 사용하여 병렬 테스트 구성
- Ruby(Cucumber)를 사용하여 병렬 테스트 구성
- Ruby(RSpec)를 사용하여 병렬 테스트 구성

Django를 사용하여 병렬 테스트 구성

다음은 Ubuntu 플랫폼에서 Django를 사용한 병렬 테스트 실행을 보여주는 `buildspec.yml`의 샘플입니다.

```
version: 0.2

batch:
  fast-fail: false
  build-fanout:
    parallelism: 5

phases:
  install:
    commands:
      - echo 'Installing Python dependencies'
      - sudo yum install -y python3 python3-pip
```

```
      - python3 -m ensurepip --upgrade
      - python3 -m pip install django
  pre_build:
    commands:
      - echo 'Prebuild'
  build:
    commands:
      - echo 'Running Django Tests'
      - |
        codebuild-tests-run \
         --test-command 'python3 manage.py test $(echo "$CODEBUILD_CURRENT_SHARD_FILES"
 | sed -E "s/\//__/g; s/\.py$//; s/__/./g")' \
         --files-search "codebuild-glob-search '**/tests/*test_*.py'" \
         --sharding-strategy 'equal-distribution'
  post_build:
    commands:
      - echo 'Test execution completed'
```

위 예제에서는 CODEBUILD_CURRENT_SHARD_FILES 환경 변수를 사용하는 모습을 보여줍니다. 여기서 CODEBUILD_CURRENT_SHARD_FILES는 Django에서 지원하는 점 표기법 파일 경로를 가져오는 데 사용됩니다. 위와 같이 큰따옴표 안에 CODEBUILD_CURRENT_SHARD_FILES를 사용합니다.

Elixir를 사용하여 병렬 테스트 구성

다음은 Ubuntu 플랫폼에서 Elixir를 사용한 병렬 테스트 실행을 보여주는 buildspec.yml의 샘플입니다.

```
version: 0.2

batch:
  fast-fail: false
  build-fanout:
    parallelism: 5

phases:
  install:
    commands:
      - echo 'Installing Elixir dependencies'
      - sudo apt update
      - sudo DEBIAN_FRONTEND=noninteractive apt install -y elixir
      - elixir --version
      - mix --version
  pre_build:
```

```
    commands:
      - echo 'Prebuild'
  build:
    commands:
      - echo 'Running Elixir Tests'
      - |
        codebuild-tests-run \
         --test-command 'mix test' \
         --files-search "codebuild-glob-search '**/test/**/*_test.exs'" \
         --sharding-strategy 'equal-distribution'
  post_build:
    commands:
      - echo "Test execution completed"
```

Go를 사용하여 병렬 테스트 구성

다음은 Linux 플랫폼에서 Go를 사용한 병렬 테스트 실행을 buildspec.yml 보여주는 샘플입니다.

```
version: 0.2

batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false

phases:
  install:
    commands:
      - echo 'Fetching Go version'
      - go version
  pre_build:
    commands:
      - echo 'prebuild'
  build:
    commands:
      - echo 'Running go Tests'
      - go mod init calculator
      - cd calc
      - |
        codebuild-tests-run \
         --test-command "go test -v calculator.go" \
         --files-search "codebuild-glob-search '**/*test.go'"
```

```
  post_build:
    commands:
      - echo "Test execution completed"
```

위 예제에서 `calculator.go` 함수에는 테스트할 간단한 수학 함수가 포함되어 있으며 모든 테스트 파일과 `calculator.go` 파일은 `calc` 폴더 내에 있습니다.

Java(Maven)를 사용하여 병렬 테스트 구성

다음은 Linux 플랫폼에서 Java를 사용한 병렬 테스트 실행을 `buildspec.yml` 보여주는 샘플입니다.

```
version: 0.2

batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false

phases:
  pre_build:
    commands:
      - echo 'prebuild'
  build:
    commands:
      - echo "Running mvn test"
      - |
        codebuild-tests-run \
          --test-command 'mvn test -Dtest=$(echo "$CODEBUILD_CURRENT_SHARD_FILES" | sed
 "s|src/test/java/||g; s/\.java//g; s|/|.|g; s/ /,/g" | tr "\n" "," | sed "s/,$//")' \
          --files-search "codebuild-glob-search '**/test/**/*.java'"

  post_build:
    commands:
      - echo "Running post-build steps..."
      - echo "Test execution completed"
```

주어진 예제에서 `CODEBUILD_CURRENT_SHARD_FILES` 환경 변수는 현재 샤드에 새 줄로 구분된 테스트 파일을 포함합니다. 이러한 파일은 Maven의 `-Dtest` 파라미터에서 허용하는 형식으로 쉼표로 구분된 클래스 이름 목록으로 변환됩니다.

Javascript(Jest)를 사용하여 병렬 테스트 구성

다음은 Ubuntu 플랫폼에서 Javascript를 사용한 병렬 테스트 실행을 보여주는 buildspec.yml의 샘플입니다.

```
version: 0.2

batch:
  fast-fail: true
  build-fanout:
    parallelism: 5
    ignore-failure: false

phases:
  install:
    commands:
      - echo 'Installing Node.js dependencies'
      - apt-get update
      - apt-get install -y nodejs
      - npm install
      - npm install --save-dev jest-junit
  pre_build:
    commands:
      - echo 'prebuild'
  build:
    commands:
      - echo 'Running JavaScript Tests'
      - |
         codebuild-tests-run \
          --test-command "npx jest" \
          --files-search "codebuild-glob-search '**/test/**/*.test.js'" \
          --sharding-strategy 'stability'
    post_build:
      commands:
        - echo 'Test execution completed'
```

Kotlin을 사용하여 병렬 테스트 구성

다음은 Linux 플랫폼에서 Kotlin을 사용한 병렬 테스트 실행을 buildspec.yml 보여주는 샘플입니다.

```
version: 0.2

batch:
```

```
  fast-fail: false
  build-fanout:
    parallelism: 2
    ignore-failure: false

phases:
  install:
    runtime-versions:
      java: corretto11
    commands:
      - echo 'Installing dependencies'
      - KOTLIN_VERSION="1.8.20" # Replace with your desired version
      - curl -o kotlin-compiler.zip -L "https://github.com/JetBrains/kotlin/releases/
download/v${KOTLIN_VERSION}/kotlin-compiler-${KOTLIN_VERSION}.zip"
      - unzip kotlin-compiler.zip -d /usr/local
      - export PATH=$PATH:/usr/local/kotlinc/bin
      - kotlin -version
      - curl -O https://repo1.maven.org/maven2/org/junit/platform/junit-platform-
console-standalone/1.8.2/junit-platform-console-standalone-1.8.2.jar
  pre_build:
    commands:
      - echo 'prebuild'
  build:
    commands:
      - echo 'Running Kotlin Tests'
      - |
        codebuild-tests-run \
          --test-command 'kotlinc src/main/kotlin/*.kt $(echo
 "$CODEBUILD_CURRENT_SHARD_FILES" | tr "\n" " ") -d classes -cp junit-platform-console-
standalone-1.8.2.jar' \
          --files-search "codebuild-glob-search 'src/test/kotlin/*.kt'"
      - |
        codebuild-tests-run \
          --test-command '
            java -jar junit-platform-console-standalone-1.8.2.jar --class-path classes
 \
              $(for file in $CODEBUILD_CURRENT_SHARD_FILES; do
                 class_name=$(basename "$file" .kt)
                 echo "--select-class $class_name"
               done)
          ' \
          --files-search "codebuild-glob-search 'src/test/kotlin/*.kt'"
  post_build:
    commands:
```

```
      - echo "Test execution completed"
```

위 예제에서는 codebuild-tests-run CLI가 두 번 사용됩니다. 첫 번째 실행 중에 kotlinc는 파일을 컴파일합니다. CODEBUILD_CURRENT_SHARD_FILES 변수는 현재 샤드에 할당된 테스트 파일을 검색한 다음 공백으로 구분된 목록으로 변환합니다. 두 번째 실행에서는 JUnit가 테스트를 실행합니다. 다시 CODEBUILD_CURRENT_SHARD_FILES는 현재 샤드에 할당된 테스트 파일을 가져오지만, 이번에는 클래스 이름으로 변환됩니다.

PHPUnit을 사용하여 병렬 테스트 구성

다음은 Linux 플랫폼에서 PHPUnit를 사용한 병렬 테스트 실행을 buildspec.yml 보여주는 샘플입니다.

```
version: 0.2

batch:
   fast-fail: false
   build-fanout:
     parallelism: 5
     ignore-failure: false

phases:
   install:
     commands:
       - echo 'Install dependencies'
       - composer require --dev phpunit/phpunit
   pre_build:
     commands:
       - echo 'prebuild'
   build:
     commands:
       - echo 'Running phpunit Tests'
       - composer dump-autoload
       - |
         codebuild-tests-run \
          --test-command "./vendor/bin/phpunit --debug" \
          --files-search "codebuild-glob-search '**/tests/*Test.php'"
   post_build:
       commands:
         - echo 'Test execution completed'
```

Pytest를 사용하여 병렬 테스트 구성

다음은 Ubuntu 플랫폼에서 Pytest를 사용한 병렬 테스트 실행을 보여주는 buildspec.yml의 샘플입니다.

```
version: 0.2

batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false

phases:
  install:
    commands:
      - echo 'Installing Python dependencies'
      - apt-get update
      - apt-get install -y python3 python3-pip
      - pip3 install --upgrade pip
      - pip3 install pytest
  build:
    commands:
      - echo 'Running Python Tests'
      - |
         codebuild-tests-run \
          --test-command 'python -m pytest' \
          --files-search "codebuild-glob-search 'tests/test_*.py'" \
          --sharding-strategy 'equal-distribution'
  post_build:
    commands:
      - echo "Test execution completed"
```

다음은 Windows 플랫폼에서 Pytest를 사용한 병렬 테스트 실행을 보여주는 buildspec.yml의 샘플입니다.

```
version: 0.2

batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false
```

```
phases:
  install:
    commands:
      - echo 'Installing Python dependencies'
      - pip install pytest
  pre_build:
    commands:
      - echo 'prebuild'
  build:
    commands:
      - echo 'Running pytest'
      - |
        & codebuild-tests-run `
         --test-command 'pytest @("$env:CODEBUILD_CURRENT_SHARD_FILES" -split \"`r?`n
\")'  `
         --files-search "codebuild-glob-search '**/test_*.py' '**/*_test.py'" `
         --sharding-strategy 'equal-distribution'
  post_build:
    commands:
      - echo "Test execution completed"
```

위 예제에서 CODEBUILD_CURRENT_SHARD_FILES 환경 변수는 현재 샤드에 할당되고 pytest 명령에 배열로 전달되는 테스트 파일을 가져오는 데 사용됩니다.

Ruby(Cucumber)를 사용하여 병렬 테스트 구성

다음은 Linux 플랫폼에서 Cucumber를 사용한 병렬 테스트 실행을 buildspec.yml 보여주는 샘플입니다.

```
version: 0.2

batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false

phases:
  install:
    commands:
      - echo 'Installing Ruby dependencies'
      - gem install bundler
```

```
      - bundle install
  pre_build:
    commands:
      - echo 'prebuild'
  build:
    commands:
      - echo 'Running Cucumber Tests'
      - cucumber --init
      - |
        codebuild-tests-run \
         --test-command "cucumber" \
         --files-search "codebuild-glob-search '**/*.feature'"
  post_build:
    commands:
      - echo "Test execution completed"
```

Ruby(RSpec)를 사용하여 병렬 테스트 구성

다음은 Ubuntu 플랫폼에서 RSpec을 사용한 병렬 테스트 실행을 보여주는 `buildspec.yml`의 샘플입니다.

```
version: 0.2

batch:
  fast-fail: false
  build-fanout:
    parallelism: 5
    ignore-failure: false

phases:
  install:
    commands:
      - echo 'Installing Ruby dependencies'
      - apt-get update
      - apt-get install -y ruby ruby-dev build-essential
      - gem install bundler
      - bundle install
  build:
    commands:
      - echo 'Running Ruby Tests'
      - |
         codebuild-tests-run \
          --test-command 'bundle exec rspec' \
```

```
        --files-search "codebuild-glob-search 'spec/**/*_spec.rb'" \
        --sharding-strategy 'equal-distribution'
post_build:
  commands:
    - echo "Test execution completed"
```

성능을 개선하기 위한 캐시 빌드

프로젝트가 빌드될 때 캐시를 사용하여 시간을 절약할 수 있습니다. 캐시는 빌드 환경에서 재사용할 수 있는 정보를 저장하여 여러 빌드에 사용할 수 있습니다. 빌드 프로젝트는 Amazon S3 또는 로컬의 두 캐싱 유형 중 하나를 사용할 수 있습니다. 로컬 캐시를 사용하는 경우 다음 세 캐시 모드에서 하나 이상을 선택해야 합니다. 소스 캐시, Docker 계층 캐시 및 사용자 지정 캐시.

> **Note**
>
> Docker 계층 캐시 모드는 Linux 환경에서만 사용할 수 있습니다. 이 모드를 선택할 경우 권한이 있는 모드에서 빌드를 실행해야 합니다. CodeBuild 프로젝트에서 부여된 권한 모드는 해당 컨테이너에 모든 디바이스에 대한 액세스 권한을 부여합니다. 자세한 내용은 Docker 문서 웹 사이트의 런타임 권한 및 Linux 기능을 참조하십시오.

주제

- Amazon S3 캐싱
- 로컬 캐싱
- 로컬 캐시 지정

Amazon S3 캐싱

Amazon S3 캐싱은 여러 빌드 호스트에 사용 가능한 캐시를 Amazon S3 버킷에 저장합니다. 이는 다운로드보다 빌드가 더 비용이 많이 드는 소형-중형 빌드 아티팩트에 적합한 옵션입니다.

빌드에서 Amazon S3를 사용하려면 `buildspec.yml`에서 캐싱하려는 파일의 경로를 지정할 수 있습니다. CodeBuild는 캐시를 자동으로 저장하고 프로젝트에 구성된 Amazon S3 위치로 업데이트합니다. 파일 경로를 지정하지 않으면 CodeBuild는 빌드 속도를 높이는 데 도움이 되도록 공통 언어 종속성을 캐싱하는 데 최선을 다합니다. 빌드 로그에서 캐시 세부 정보를 볼 수 있습니다.

또한, 여러 버전의 캐시를 사용하려는 경우 `buildspec.yml`에서 캐시 키를 정의할 수 있습니다. CodeBuild는 이 캐시 키의 컨텍스트에 캐시를 저장하고 생성된 후에는 업데이트되지 않는 고유한 캐시

복사본을 생성합니다. 캐시 키는 프로젝트 간에도 공유할 수 있습니다. 동적 키, 캐시 버전 관리, 빌드 간 캐시 공유와 같은 기능은 키가 지정된 경우에만 사용할 수 있습니다.

buildspec 파일의 캐시 구문에 대한 자세한 내용은 buildspec 참조의 cache를 참조하세요.

주제

- 동적 키 생성
- codebuild-hash-files
- 캐시 버전
- 프로젝트 간 캐시 공유
- buildspec 예제

동적 키 생성

캐시 키에는 고유한 셸 명령과 환경 변수가 포함될 수 있으므로 키가 변경될 때 자동 캐시 업데이트가 활성화됩니다. 예를 들어, `package-lock.json` 파일의 해시를 사용하여 키를 정의할 수 있습니다. 해당 파일의 종속성이 변경되면 해시와 캐시 키가 변경되어 새 캐시의 자동 생성이 트리거됩니다.

```
cache:
    key: npm-key-$(codebuild-hash-files package-lock.json)
```

CodeBuild는 `$(codebuild-hash-files package-lock.json)` 표현식을 평가하여 최종 키를 가져옵니다.

```
npm-key-abc123
```

`CODEBUILD_RESOLVED_SOURCE_VERSION`과 같은 환경 변수를 사용하여 캐시 키를 정의할 수도 있습니다. 이렇게 하면 소스가 변경될 때마다 새 키가 생성되어 새 캐시가 자동으로 저장됩니다.

```
cache:
   key: npm-key-$CODEBUILD_RESOLVED_SOURCE_VERSION
```

CodeBuild는 표현식을 평가하고 최종 키를 가져옵니다.

```
npm-key-046e8b67481d53bdc86c3f6affdd5d1afae6d369
```

codebuild-hash-files

`codebuild-hash-files`는 glob 패턴을 사용하여 CodeBuild 소스 디렉터리의 파일 세트에 대한 SHA-256 해시를 계산하는 CLI 도구입니다.

```
codebuild-hash-files <glob-pattern-1> <glob-pattern-2> ...
```

여기에 `codebuild-hash-files`를 사용하는 몇 가지 예가 있습니다.

```
codebuild-hash-files package-lock.json
codebuild-hash-files '**/*.md'
```

캐시 버전

캐시 버전은 캐시되는 디렉터리의 경로에서 생성된 해시입니다. 두 캐시의 버전이 다른 경우 일치하는 프로세스 중에 캐시가 고유한 캐시로 처리됩니다. 예를 들어, 다음 두 캐시는 서로 다른 경로를 참조하기 때문에 서로 다른 것으로 간주됩니다.

```
version: 0.2

phases:
  build:
    commands:
      - pip install pandas==2.2.3 --target pip-dependencies
cache:
  key: pip-dependencies
  paths:
    - "pip-dependencies/**/*"
```

```
version: 0.2

phases:
  build:
    commands:
      - pip install pandas==2.2.3 --target tmp/pip-dependencies
cache:
  key: pip-dependencies
  paths:
    - "tmp/pip-dependencies/**/*"
```

프로젝트 간 캐시 공유

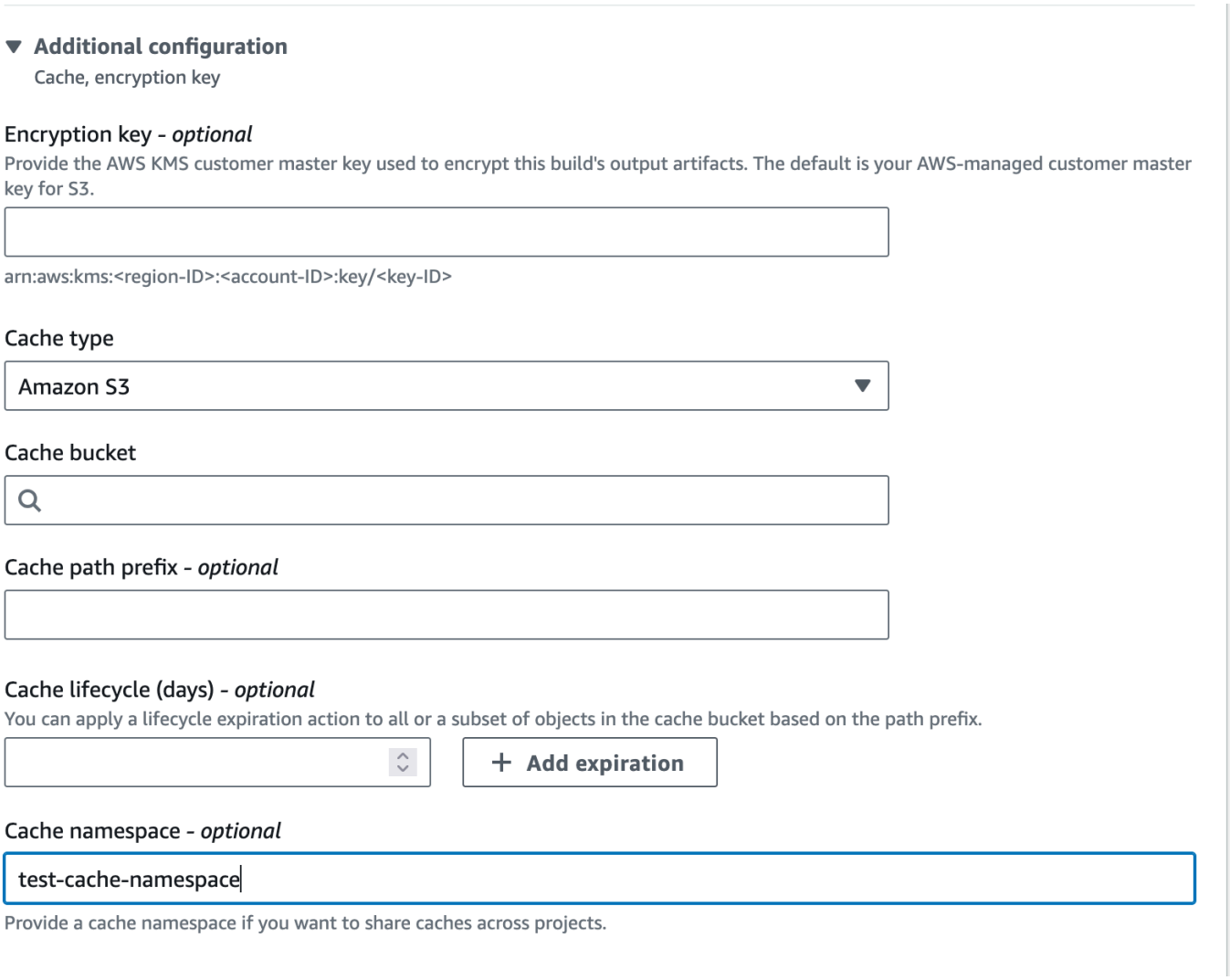
cache 섹션 아래의 cacheNamespace API 필드를 사용하여 여러 프로젝트에서 캐시를 공유할 수 있습니다. 이 필드는 캐시의 범위를 정의합니다. 캐시를 공유하려면 다음을 수행해야 합니다.

- 동일한 cacheNamespace를 사용합니다.
- 동일한 key 캐시를 지정합니다.
- 동일한 캐시 경로를 정의합니다.
- 설정된 경우 동일한 Amazon S3 버킷 및 pathPrefix를 사용합니다.

이렇게 하면 일관성이 보장되고 프로젝트 간에 캐시 공유가 활성화됩니다.

캐시 네임스페이스 지정(콘솔)

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 프로젝트 생성을 선택합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 및 빌드 실행(콘솔) 섹션을 참조하세요.
3. 아티팩트에서 추가 구성을 선택합니다.
4. 캐시 유형에서 Amazon S3를 선택합니다.
5. 캐시 네임스페이스 - 선택 사항에서 네임스페이스 값을 입력합니다.



6. 기본값으로 계속 진행한 다음 빌드 프로젝트 생성을 선택합니다.

캐시 네임스페이스 지정(AWS CLI)

의 `--cache` 파라미터를 사용하여 캐시 네임스페이스를 AWS CLI 지정할 수 있습니다.

```
--cache '{"type": "S3", "location": "your-s3-bucket", "cacheNamespace": "test-cache-namespace"}'
```

buildspec 예제

다음은 일반적인 언어에 대한 몇 가지 buildspec 예제입니다.

주제

- 캐시 Node.js 종속성

- 캐시 Python 종속성
- 캐시 Ruby 종속성
- 캐시 Go 종속성

캐시 Node.js 종속성

프로젝트에 `package-lock.json` 파일이 포함되어 있고 npm를 사용하여 Node.js 종속성을 관리하는 경우, 다음 예제에서 캐싱을 설정하는 방법을 보여줍니다. 기본적으로 npm에서는 `node_modules` 디렉터리에 종속성을 설치합니다.

```
version: 0.2

phases:
  build:
    commands:
      - npm install
cache:
  key: npm-$(codebuild-hash-files package-lock.json)
  paths:
    - "node_modules/**/*"
```

캐시 Python 종속성

프로젝트에 `requirements.txt` 파일이 포함되어 있고 pip를 사용하여 Python 종속성을 관리하는 경우, 다음 예제에서 캐싱을 구성하는 방법을 보여줍니다. 기본적으로 pip는 시스템 `site-packages` 디렉터리에 패키지를 설치합니다.

```
version: 0.2

phases:
  build:
    commands:
      - pip install -r requirements.txt
cache:
  key: python-$(codebuild-hash-files requirements.txt)
  paths:
    - "/root/.pyenv/versions/${python_version}/lib/python${python_major_version}/site-
packages/**/*"
```

또한, 특정 디렉터리에 종속성을 설치하고 해당 디렉터리에 대한 캐싱을 구성할 수 있습니다.

```
version: 0.2

phases:
  build:
    commands:
      - pip install -r requirements.txt --target python-dependencies
cache:
  key: python-$(codebuild-hash-files requirements.txt)
  paths:
    - "python-dependencies/**/*"
```

캐시 Ruby 종속성

프로젝트에 Gemfile.lock 파일이 포함되어 있고 Bundler를 사용하여 gem 종속성을 관리하는 경우, 다음 예제에서 캐싱을 효율적으로 구성하는 방법을 보여줍니다.

```
version: 0.2

phases:
  build:
    commands:
      - bundle install --path vendor/bundle
cache:
  key: ruby-$(codebuild-hash-files Gemfile.lock)
  paths:
    - "vendor/bundle/**/*"
```

캐시 Go 종속성

프로젝트에 go.sum 파일이 포함되어 있고 Go 모듈을 사용하여 종속성을 관리하는 경우, 다음 예제에서 캐싱을 구성하는 방법을 보여줍니다. 기본적으로 Go 모듈은 다운로드되어 ${GOPATH}/pkg/mod 디렉터리에 저장됩니다.

```
version: 0.2

phases:
  build:
    commands:
      - go mod download
cache:
  key: go-$(codebuild-hash-files go.sum)
```

```
  paths:
    - "/go/pkg/mod/**/*"
```

로컬 캐싱

로컬 캐싱은 해당 빌드 호스트에만 사용할 수 있는 캐시를 빌드에 로컬로 저장합니다. 빌드 호스트에서 즉각적으로 캐시를 사용할 수 있으므로 중형-대형 빌드 아티팩트에 적합한 옵션입니다. 빌드가 드문 경우에는 최선의 옵션이 아닙니다. 이는 빌드 성능이 네트워크 전송 시간의 영향을 받지 않는다는 의미입니다.

로컬 캐싱을 선택할 경우 다음 캐시 모드 중 하나 이상을 선택해야 합니다.

- 소스 캐시 모드는 기본 및 보조 소스를 위해 Git 메타데이터를 캐싱합니다. 캐시가 생성되면 이후의 빌드는 커밋 사이의 변경 사항만 끌어옵니다. 이 모드는 클린 작업 디렉터리를 사용하고 소스가 대규모 Git 리포지토리인 프로젝트에 적합한 선택입니다. 이 옵션을 선택하고 프로젝트가 Git 리포지토리(AWS CodeCommit, GitHub, GitHub Enterprise Server 또는 Bitbucket)를 사용하지 않을 경우 이 옵션은 무시됩니다.
- Docker 계층 캐시 모드는 기존 Docker 계층을 캐싱합니다. 이 모드는 대용량 도커 이미지를 빌드하거나 끌어오는 프로젝트에 적합한 선택입니다. 네트워크에서 대용량 도커 이미지를 끌어올 때 발생하는 성능 문제를 방지할 수 있습니다.

  > **Note**
  >
  > - Docker 계층 캐시는 Linux 환경에서만 사용할 수 있습니다.
  > - 프로젝트가 필요한 Docker 권한을 가지도록 `privileged` 플래그를 설정해야 합니다.
  >
  >   기본적으로 비 VPC 빌드에는 Docker 데몬이 활성화됩니다. VPC 빌드에 Docker 컨테이너를 사용하려면 Docker Docs 웹 사이트의 런타임 권한 및 Linux 기능을 참조하고 권한 부여 모드를 활성화합니다. 또한 Windows는 권한 모드를 지원하지 않습니다.
  > - Docker 계층 캐시를 사용하기 전에 보안에 미치는 영향을 고려해야 합니다.

- 사용자 지정 캐시 모드는 buildspec 파일에 지정한 디렉터리를 캐싱합니다. 이 모드는 다른 두 로컬 캐시 모드에 적합하지 않은 빌드 시나리오에 적합한 선택입니다. 사용자 지정 캐시를 사용할 경우
  - 캐싱을 위해 디렉터리만 지정할 수 있습니다. 개별 파일은 지정할 수 없습니다.
  - Symlink를 사용하여 캐싱된 디렉터리를 참조합니다.

- 캐싱된 디렉터리는 프로젝트 소스를 다운로드하기 전에 빌드에 연결됩니다. 캐시된 항목은 이름이 동일한 경우 소스 항목을 재정의합니다. 디렉터리는 buildspec 파일에서 경로를 사용하여 지정합니다. 자세한 내용은 buildspec 구문 단원을 참조하십시오.
- 소스와 캐시에서 동일한 디렉터리 이름은 사용하지 마십시오. 로컬로 캐시된 디렉터리는 소스 리포지토리에서 이름이 같은 디렉터리의 내용을 재정의하거나 삭제할 수 있습니다.

> **Note**
>
> `LINUX_GPU_CONTAINER` 환경 유형 및 `BUILD_GENERAL1_2XLARGE` 컴퓨팅 유형에서는 로컬 캐싱이 지원되지 않습니다. 자세한 내용은 빌드 환경 컴퓨팅 모드 및 유형 단원을 참조하십시오.

> **Note**
>
> VPC에서 사용할 CodeBuild를 구성하는 경우 로컬 캐싱은 지원되지 않습니다. CodeBuild에서 VPC를 사용하는 방법에 대한 자세한 내용은 Amazon Virtual Private Cloud AWS CodeBuild 와 함께 사용 섹션을 참조하세요.

로컬 캐시 지정

AWS CLI, 콘솔, SDK 또는를 사용하여 로컬 캐시를 CloudFormation 지정할 수 있습니다. 로컬 캐싱에 대한 자세한 정보는 로컬 캐싱 섹션을 참조하세요.

주제

- 로컬 캐싱 지정(CLI)
- 로컬 캐싱 지정(콘솔)
- 로컬 캐싱 지정(CloudFormation)

로컬 캐싱 지정(CLI)

의 `--cache` 파라미터를 사용하여 세 가지 로컬 캐시 유형을 각각 AWS CLI 지정할 수 있습니다.

- 소스 캐시를 지정하려면

```
--cache type=LOCAL,mode=[LOCAL_SOURCE_CACHE]
```

- Docker 계층 캐시를 지정하려면

```
--cache type=LOCAL,mode=[LOCAL_DOCKER_LAYER_CACHE]
```

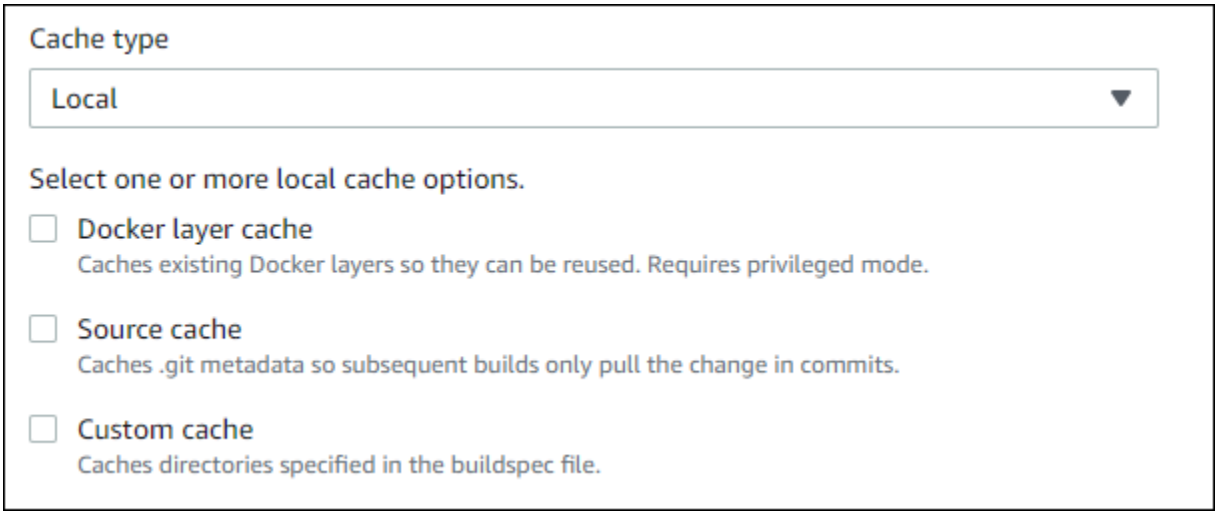
- 사용자 지정 캐시를 지정하려면

```
--cache type=LOCAL,mode=[LOCAL_CUSTOM_CACHE]
```

자세한 내용은 빌드 프로젝트 생성(AWS CLI) 단원을 참조하십시오.

로컬 캐싱 지정(콘솔)

콘솔의 결과물 섹션에서 캐시를 지정합니다. 캐시 유형은 Amazon S3 또는 로컬을 선택합니다. 로컬을 선택한 경우 세 로컬 캐시 옵션 중 하나 이상을 선택합니다.



자세한 내용은 빌드 프로젝트 만들기(콘솔) 단원을 참조하십시오.

로컬 캐싱 지정(CloudFormation)

CloudFormation 를 사용하여 로컬 캐시를 지정하는 경우 `Cache` 속성에서에 대해를 `Type`지정합니다`LOCAL`. 다음 샘플 YAML 형식 CloudFormation 코드는 세 가지 로컬 캐시 유형을 모두 지정합니다. 각 유형을 임의로 조합하여 지정할 수 있습니다. Docker 계층 캐시를 사용하는 경우 `Environment`에서 `PrivilegedMode`를 `true`로 설정하고 `Type`을 `LINUX_CONTAINER`로 설정해야 합니다.

```
CodeBuildProject:
```

```
    Type: AWS::CodeBuild::Project
    Properties:
      Name: MyProject
      ServiceRole: <service-role>
      Artifacts:
        Type: S3
        Location: <bucket-name>
        Name: myArtifact
        EncryptionDisabled: true
        OverrideArtifactName: true
      Environment:
        Type: LINUX_CONTAINER
        ComputeType: BUILD_GENERAL1_SMALL
        Image: aws/codebuild/standard:5.0
        Certificate: <bucket/cert.zip>
        # PrivilegedMode must be true if you specify LOCAL_DOCKER_LAYER_CACHE
        PrivilegedMode: true
      Source:
        Type: GITHUB
        Location: <github-location>
        InsecureSsl: true
        GitCloneDepth: 1
        ReportBuildStatus: false
      TimeoutInMinutes: 10
      Cache:
        Type: LOCAL
        Modes: # You can specify one or more cache mode,
          - LOCAL_CUSTOM_CACHE
          - LOCAL_DOCKER_LAYER_CACHE
          - LOCAL_SOURCE_CACHE
```

> **Note**
>
> 기본적으로 비 VPC 빌드에는 Docker 데몬이 활성화됩니다. VPC 빌드에 Docker 컨테이너를 사용하려면 Docker Docs 웹 사이트의 런타임 권한 및 Linux 기능을 참조하고 권한 부여 모드를 활성화합니다. 또한 Windows는 권한 모드를 지원하지 않습니다.

자세한 내용은 빌드 프로젝트 생성(CloudFormation) 단원을 참조하십시오.

에서 빌드 디버그 AWS CodeBuild

AWS CodeBuild는 개발 및 문제 해결 중에 빌드를 디버깅하는 두 가지 방법을 제공합니다. CodeBuild 샌드박스 환경을 사용하여 문제를 조사하고 실시간으로 수정 사항을 검증하거나 AWS Systems Manager Session Manager를 사용하여 빌드 컨테이너에 연결하고 컨테이너 상태를 볼 수 있습니다.

CodeBuild 샌드박스를 사용하여 빌드 디버그

CodeBuild 샌드박스 환경은 안전하고 격리된 환경에서 대화형 디버그 세션을 제공합니다. AWS Management Console 또는를 통해 환경과 직접 상호 작용하고 AWS CLI, 명령을 실행하고, 단계별로 빌드 프로세스를 검증할 수 있습니다. 비용 효율적인 초당 결제 모델을 사용하며 빌드 환경과 동일한 소스 공급자 및 AWS 서비스와의 기본 통합을 지원합니다. SSH 클라이언트를 사용하거나 통합 개발 환경(IDE)에서 샌드박스 환경에 연결할 수도 있습니다.

CodeBuild 샌드박스 요금에 대한 자세한 내용은 CodeBuild 요금 설명서를 참조하세요. 자세한 지침은 CodeBuild 샌드박스를 사용하여 빌드 디버그 설명서를 참조하세요.

Session Manager를 사용하여 빌드 디버그

AWS Systems Manager Session Manager를 사용하면 실제 실행 환경에서 실행 중인 빌드에 직접 액세스할 수 있습니다. 이 접근 방식을 사용하면 활성 빌드 컨테이너에 연결하고 빌드 프로세스를 실시간으로 검사할 수 있습니다. 파일 시스템을 검사하고, 실행 중인 프로세스를 모니터링하고, 발생하는 문제를 해결할 수 있습니다.

자세한 지침은 Session Manager를 사용하여 빌드 디버그 설명서를 참조하세요.

CodeBuild 샌드박스를 사용하여 빌드 디버그

에서는 CodeBuild 샌드박스를 사용하여 사용자 지정 명령을 실행하고 빌드 문제를 해결하여 빌드를 디버깅할 AWS CodeBuild수 있습니다.

주제

- 사전 조건
- CodeBuild 샌드박스를 사용하여 빌드 디버그(콘솔)
- CodeBuild 샌드박스를 사용하여 빌드 디버그(AWS CLI)
- 자습서: SSH를 사용하여 샌드박스에 연결
- AWS CodeBuild 샌드박스 SSH 연결 문제 해결

사전 조건

CodeBuild 샌드박스를 사용하기 전에 CodeBuild 서비스 역할에 다음 SSM 정책이 있는지 확인합니다.

JSON

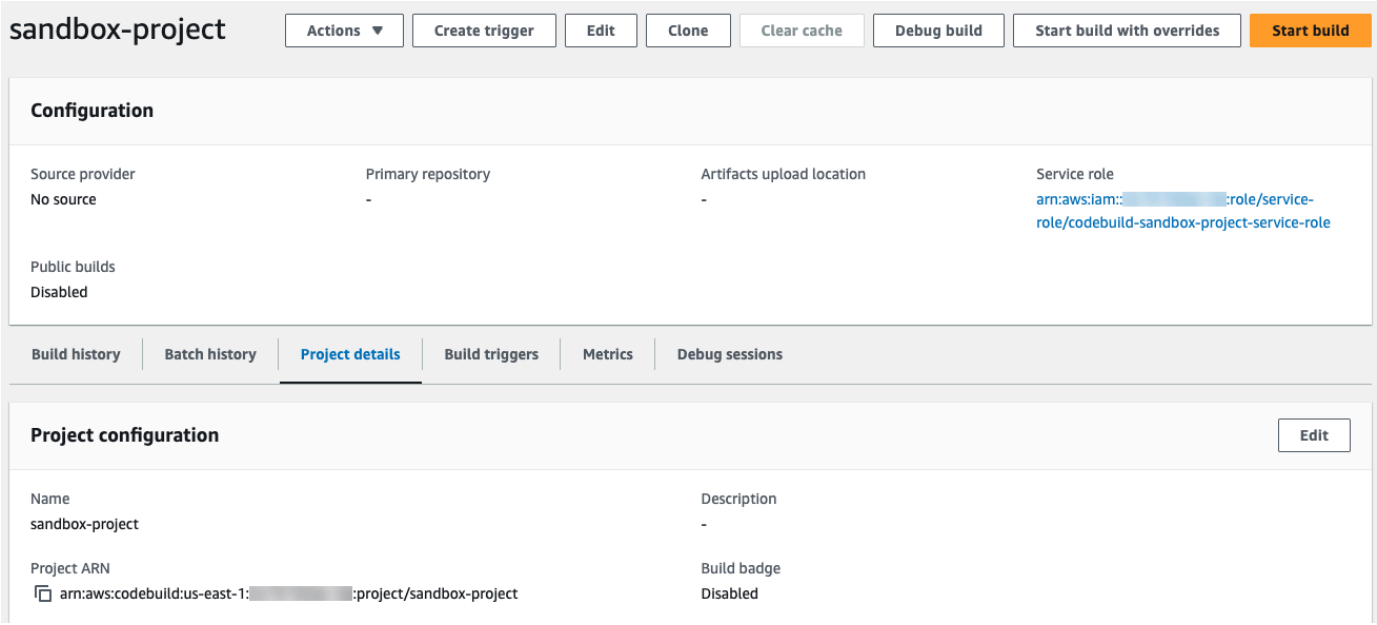
```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssmmessages:CreateControlChannel",
                "ssmmessages:CreateDataChannel",
                "ssmmessages:OpenControlChannel",
                "ssmmessages:OpenDataChannel"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:StartSession"
            ],
            "Resource": [
                "arn:aws:codebuild:us-east-1:111122223333:build/*",
                "arn:aws:ssm:us-east-1::document/AWS-StartSSHSession"
            ]
        }
    ]
}
```

CodeBuild 샌드박스를 사용하여 빌드 디버그(콘솔)

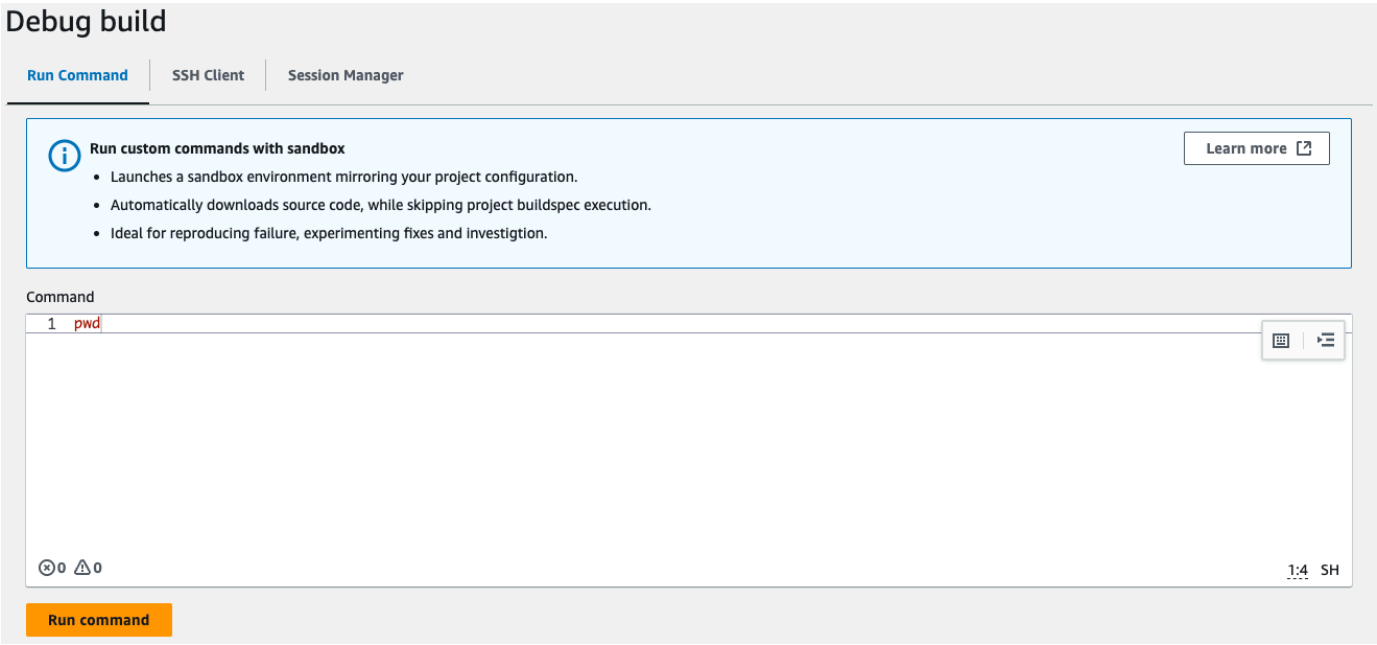
다음 지침에 따라 명령을 실행하고 콘솔에서 SSH 클라이언트를 CodeBuild 샌드박스에 연결합니다.

CodeBuild 샌드박스를 사용하여 명령 실행(콘솔)

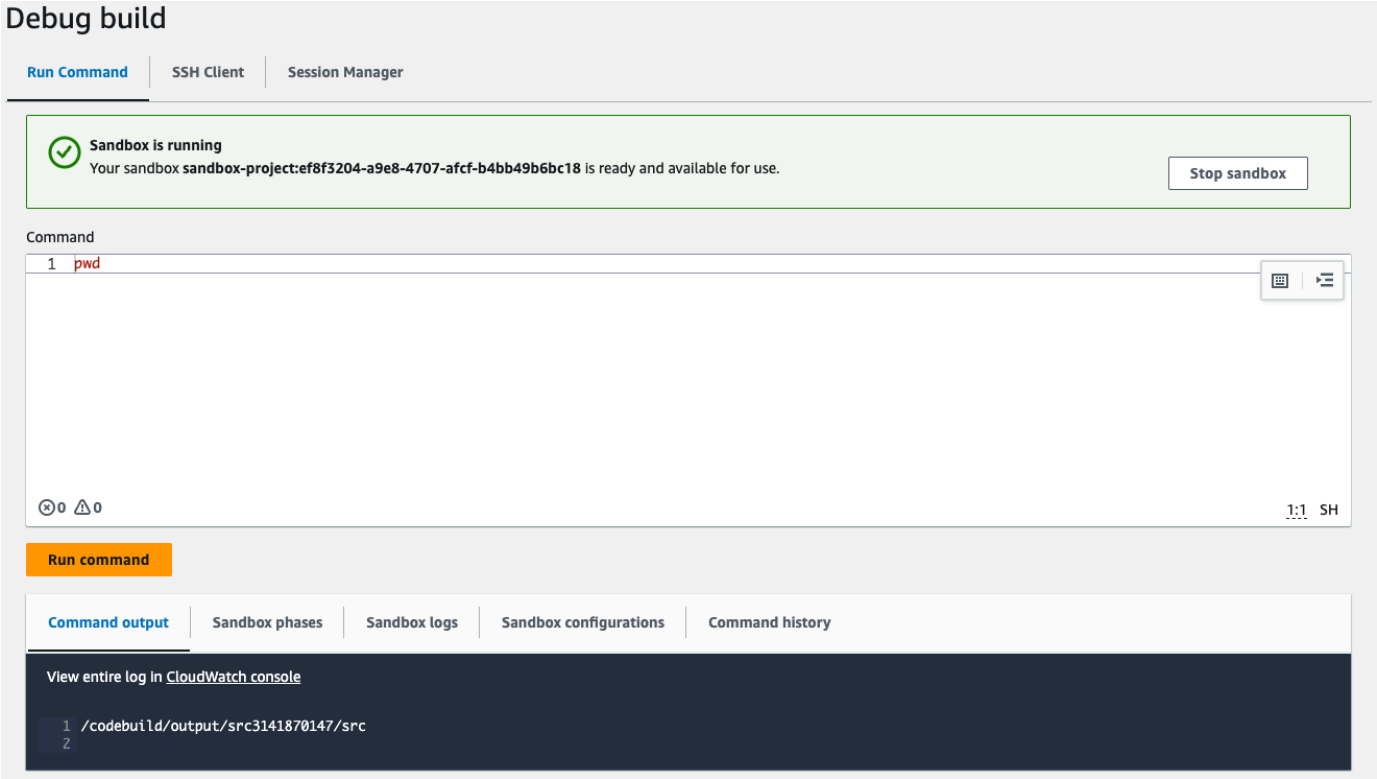
1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 프로젝트 빌드를 선택합니다. 빌드 프로젝트를 선택한 다음 빌드 디버그를 선택합니다.



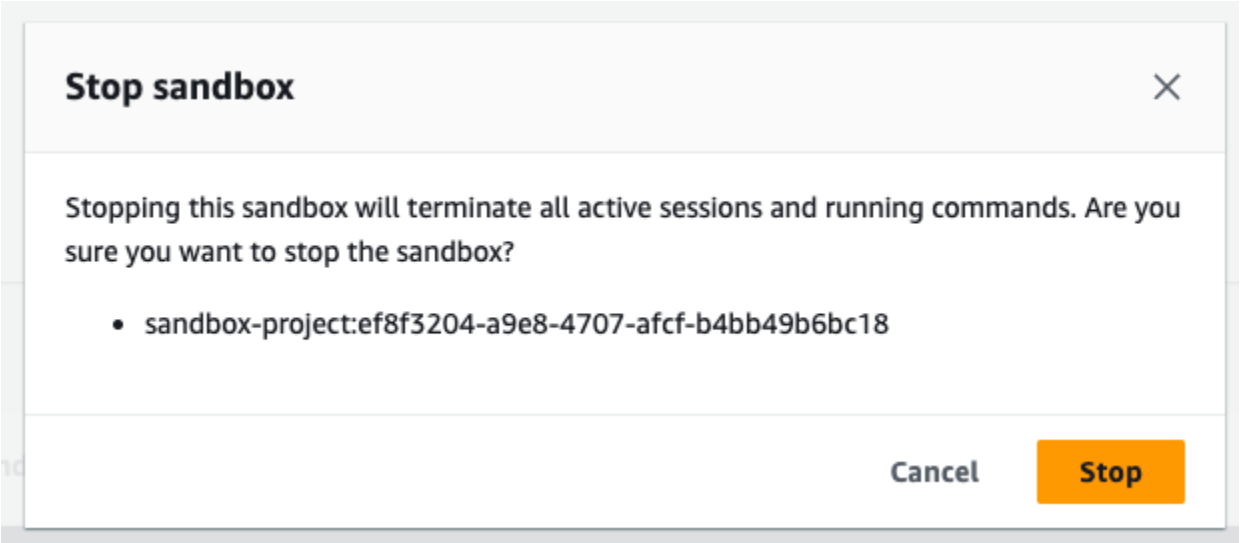
3. 명령 실행 탭에서 사용자 지정 명령을 입력한 다음 명령 실행을 선택합니다.

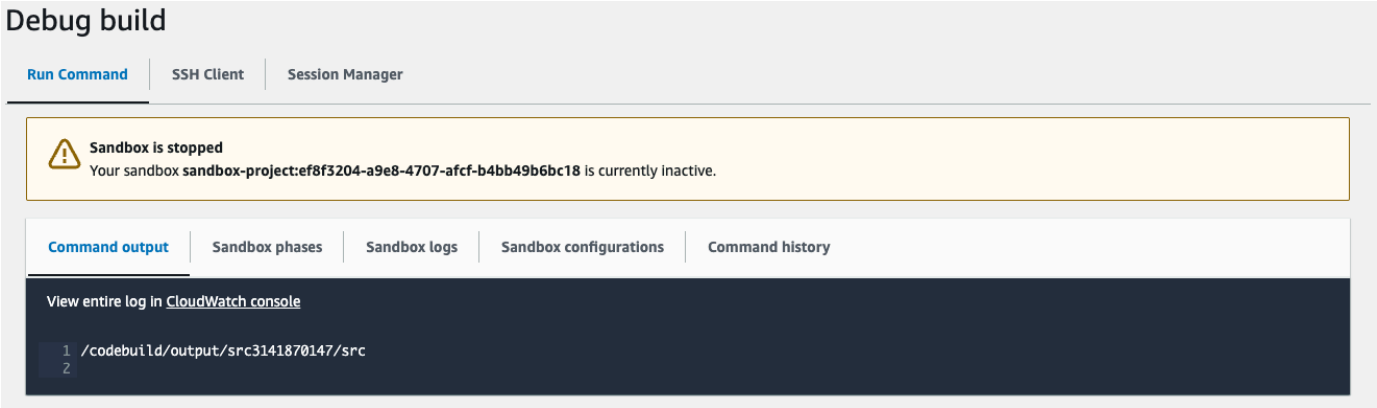


4. 그러면 CodeBuild 샌드박스가 초기화되고 사용자 지정 명령 실행이 시작됩니다. 출력이 완료되면 출력 탭에 출력이 표시됩니다.



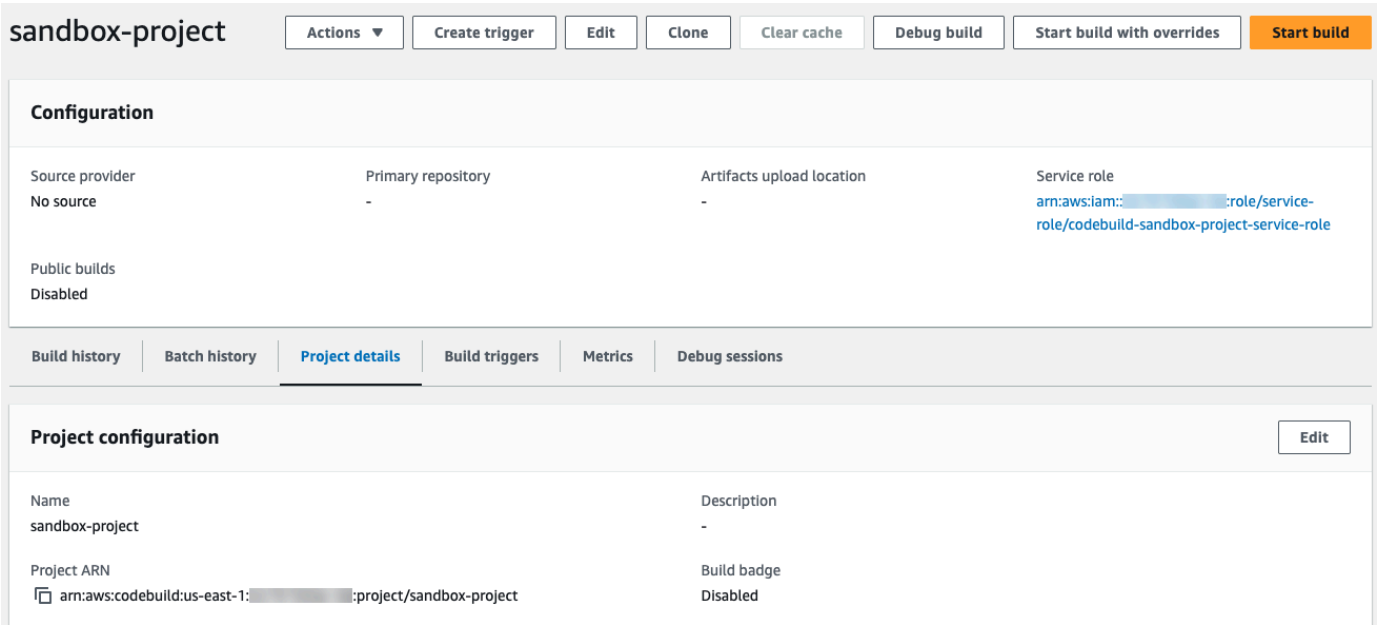
5. 문제 해결이 완료되면 샌드박스 중지를 선택하여 샌드박스를 중지할 수 있습니다. 그런 다음 중지를 선택하여 샌드박스가 중지되는지 확인합니다.



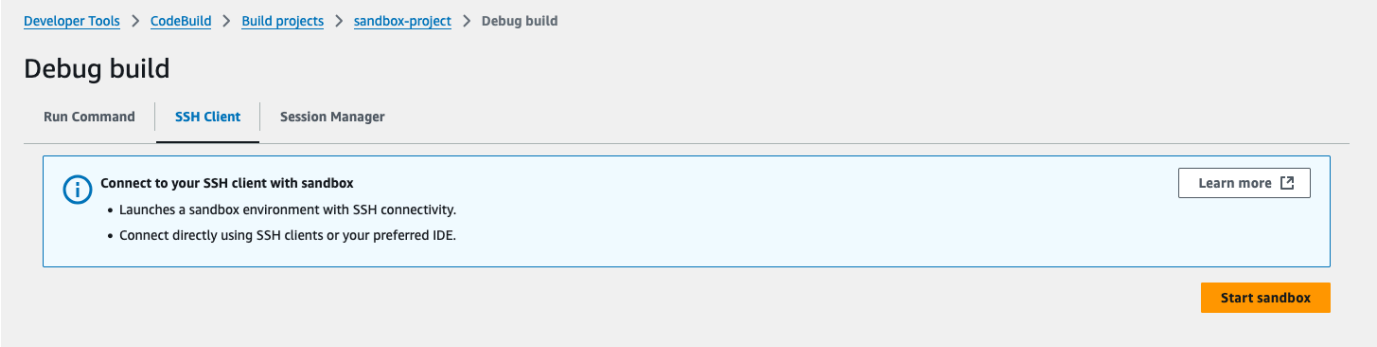


CodeBuild 샌드박스를 사용하여 SSH 클라이언트에 연결(콘솔)

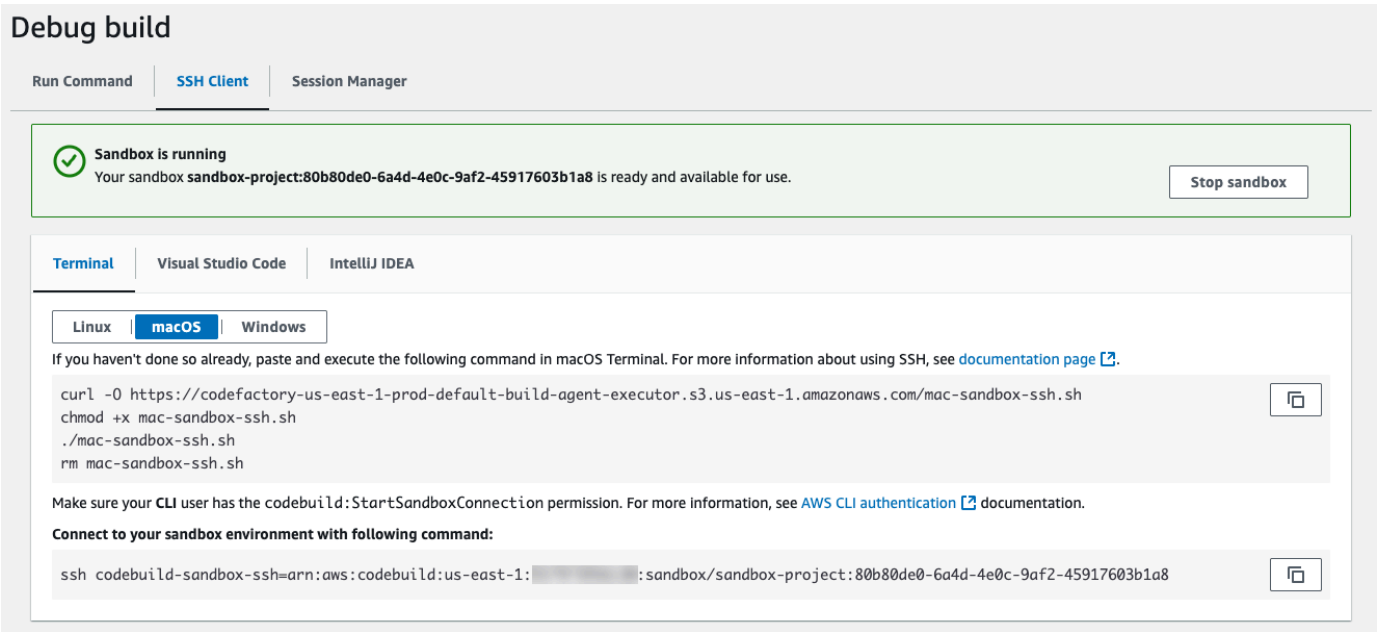
1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 프로젝트 빌드를 선택합니다. 빌드 프로젝트를 선택한 다음 빌드 디버그를 선택합니다.



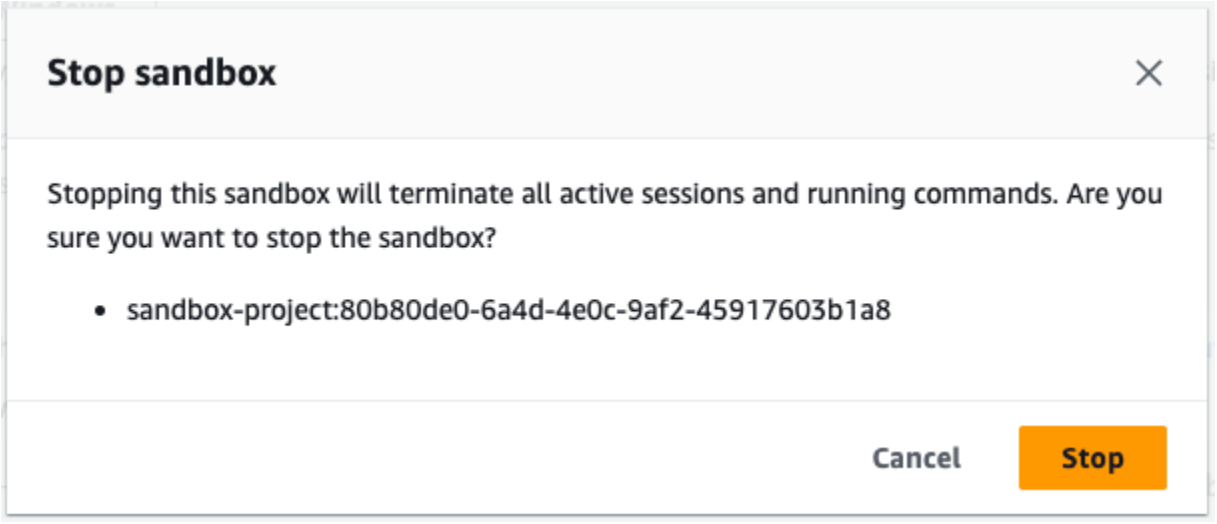
3. SSH 클라이언트 탭에서 샌드박스 시작을 선택합니다.

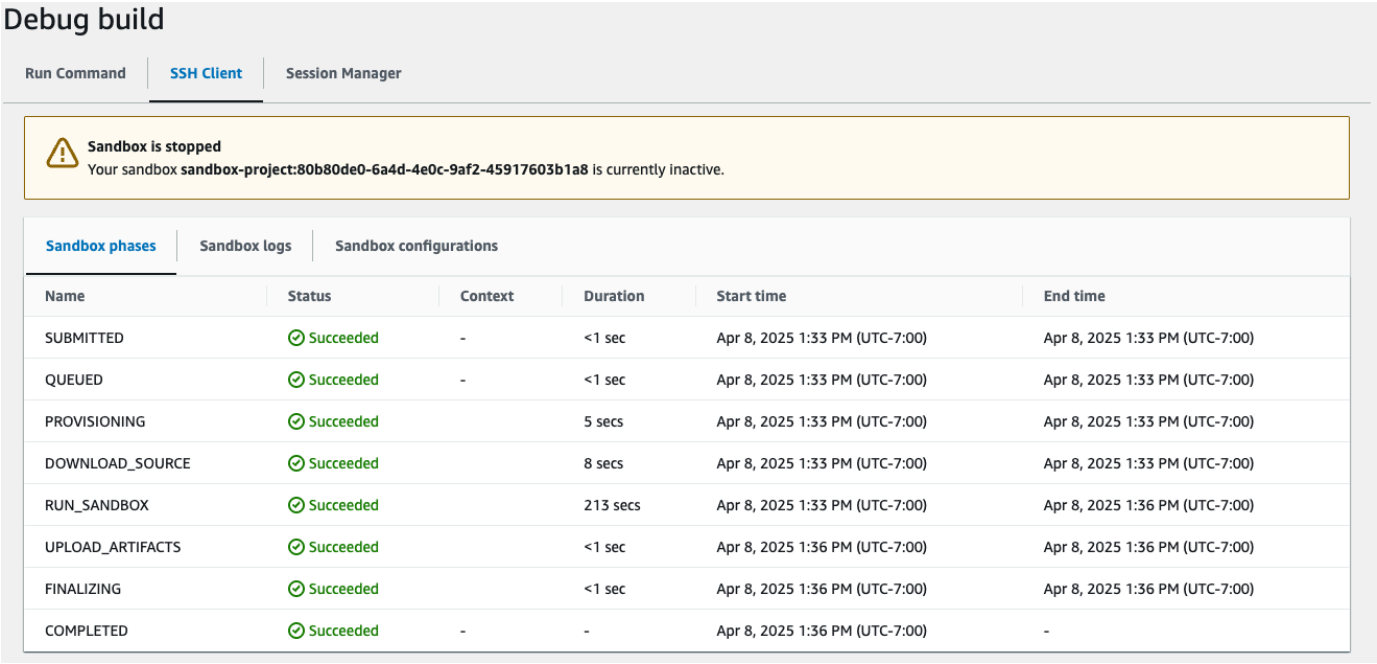


4. CodeBuild 샌드박스가 실행되기 시작하면 콘솔 지침에 따라 SSH 클라이언트를 샌드박스에 연결합니다.



5. 문제 해결이 완료되면 샌드박스 중지를 선택하여 샌드박스를 중지할 수 있습니다. 그런 다음 중지를 선택하여 샌드박스가 중지되는지 확인합니다.





CodeBuild 샌드박스를 사용하여 빌드 디버그(AWS CLI)

다음 지침에 따라 명령을 실행하고 SSH 클라이언트를 CodeBuild 샌드박스에 연결합니다.

CodeBuild 샌드박스 시작(AWS CLI)

CLI command

```
aws codebuild start-sandbox --project-name $PROJECT_NAME
```

- `--project-name`: CodeBuild 프로젝트 이름

Sample request

```
aws codebuild start-sandbox --project-name "project-name"
```

Sample response

```
{
    "id": "project-name",
    "arn": "arn:aws:codebuild:us-west-2:962803963624:sandbox/project-name",
    "projectName": "project-name",
    "requestTime": "2025-02-06T11:24:15.560000-08:00",
    "status": "QUEUED",
    "source": {
        "type": "S3",
        "location": "arn:aws:s3:::cofa-e2e-test-1-us-west-2-beta-default-build-
sources/eb-sample-jetty-v4.zip",
        "insecureSsl": false
    },
    "environment": {
        "type": "LINUX_CONTAINER",
        "image": "aws/codebuild/standard:6.0",
        "computeType": "BUILD_GENERAL1_SMALL",
        "environmentVariables": [{
                "name": "foo",
                "value": "bar",
                "type": "PLAINTEXT"
            },
            {
                "name": "bar",
                "value": "baz",
                "type": "PLAINTEXT"
            }
        ],
        "privilegedMode": false,
        "imagePullCredentialsType": "CODEBUILD"
    },
    "timeoutInMinutes": 10,
    "queuedTimeoutInMinutes": 480,
    "logConfig": {
        "cloudWatchLogs": {
            "status": "ENABLED",
            "groupName": "group",
            "streamName": "stream"
```

```
        },
        "s3Logs": {
            "status": "ENABLED",
            "location": "codefactory-test-pool-1-us-west-2-beta-default-build-logs",
            "encryptionDisabled": false
        }
    },
    "encryptionKey": "arn:aws:kms:us-west-2:962803963624:alias/SampleEncryptionKey",
    "serviceRole": "arn:aws:iam::962803963624:role/BuildExecutionServiceRole",
    "currentSession": {
        "id": "0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
        "currentPhase": "QUEUED",
        "status": "QUEUED",
        "startTime": "2025-02-06T11:24:15.626000-08:00",
        "logs": {
            "groupName": "group",
            "streamName": "stream/0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
            "deepLink": "https://console.aws.amazon.com/cloudwatch/home?
region=us-west-2#logsV2:log-groups/log-group/group/log-events/stream
$252F0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
            "s3DeepLink": "https://s3.console.aws.amazon.com/s3/object/
codefactory-test-pool-1-us-west-2-beta-default-build-logs/0103e0e7-52aa-4a3d-81dd-
bfc27226fa54.gz?region=us-west-2",
            "cloudWatchLogsArn": "arn:aws:logs:us-west-2:962803963624:log-
group:group:log-stream:stream/0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
            "s3LogsArn": "arn:aws:s3:::codefactory-test-pool-1-us-west-2-beta-
default-build-logs/0103e0e7-52aa-4a3d-81dd-bfc27226fa54.gz",
            "cloudWatchLogs": {
                "status": "ENABLED",
                "groupName": "group",
                "streamName": "stream"
            },
            "s3Logs": {
                "status": "ENABLED",
                "location": "codefactory-test-pool-1-us-west-2-beta-default-build-
logs",
                "encryptionDisabled": false
            }
        }
    }
}
```

샌드박스 상태에 대한 정보 가져오기(AWS CLI)

CLI command

```
aws codebuild batch-get-sandboxes --ids $SANDBOX_IDs
```

Sample request

```
aws codebuild stop-sandbox --id "arn:aws:codebuild:us-west-2:962803963624:sandbox/
project-name"
```

- `--ids`: sandboxIds 또는 sandboxArns의 쉼표로 구분된 값 목록.

  샌드박스 ID 또는 샌드박스 ARN을 제공할 수 있습니다.

  - 샌드박스 ID: *<codebuild-project-name>*:*<UUID>*

    예를 들어 `project-name:d25be134-05cb-404a-85da-ac5f85d2d72c`입니다.
  - 샌드박스 ARN: arn:aws:codebuild:*<region>*:*<account-id>*:sandbox/*<codebuild-project-name>*:*<UUID>*

    예를 들어 `arn:aws:codebuild:us-west-2:962803963624:sandbox/project-name:d25be134-05cb-404a-85da-ac5f85d2d72c`입니다.

Sample response

```
{
    "sandboxes": [{
        "id": "project-name",
        "arn": "arn:aws:codebuild:us-west-2:962803963624:sandbox/project-name",
        "projectName": "project-name",
        "requestTime": "2025-02-06T11:24:15.560000-08:00",
        "endTime": "2025-02-06T11:39:21.587000-08:00",
        "status": "STOPPED",
        "source": {
            "type": "S3",
            "location": "arn:aws:s3:::cofa-e2e-test-1-us-west-2-beta-default-build-
sources/eb-sample-jetty-v4.zip",
            "insecureSsl": false
        },
```

```
        "environment": {
            "type": "LINUX_CONTAINER",
            "image": "aws/codebuild/standard:6.0",
            "computeType": "BUILD_GENERAL1_SMALL",
            "environmentVariables": [{
                    "name": "foo",
                    "value": "bar",
                    "type": "PLAINTEXT"
                },
                {
                    "name": "bar",
                    "value": "baz",
                    "type": "PLAINTEXT"
                }
            ],
            "privilegedMode": false,
            "imagePullCredentialsType": "CODEBUILD"
        },
        "timeoutInMinutes": 10,
        "queuedTimeoutInMinutes": 480,
        "logConfig": {
            "cloudWatchLogs": {
                "status": "ENABLED",
                "groupName": "group",
                "streamName": "stream"
            },
            "s3Logs": {
                "status": "ENABLED",
                "location": "codefactory-test-pool-1-us-west-2-beta-default-build-
logs",
                "encryptionDisabled": false
            }
        },
        "encryptionKey": "arn:aws:kms:us-west-2:962803963624:alias/
SampleEncryptionKey",
        "serviceRole": "arn:aws:iam::962803963624:role/BuildExecutionServiceRole",
        "currentSession": {
            "id": "0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
            "currentPhase": "COMPLETED",
            "status": "STOPPED",
            "startTime": "2025-02-06T11:24:15.626000-08:00",
            "endTime": "2025-02-06T11:39:21.600000-08:00",
            "phases": [{
                    "phaseType": "SUBMITTED",
```

```
            "phaseStatus": "SUCCEEDED",
            "startTime": "2025-02-06T11:24:15.577000-08:00",
            "endTime": "2025-02-06T11:24:15.606000-08:00",
            "durationInSeconds": 0
        },
        {
            "phaseType": "QUEUED",
            "phaseStatus": "SUCCEEDED",
            "startTime": "2025-02-06T11:24:15.606000-08:00",
            "endTime": "2025-02-06T11:24:16.067000-08:00",
            "durationInSeconds": 0
        },
        {
            "phaseType": "PROVISIONING",
            "phaseStatus": "SUCCEEDED",
            "startTime": "2025-02-06T11:24:16.067000-08:00",
            "endTime": "2025-02-06T11:24:20.519000-08:00",
            "durationInSeconds": 4,
            "contexts": [{
                "statusCode": "",
                "message": ""
            }]
        },
        {
            "phaseType": "DOWNLOAD_SOURCE",
            "phaseStatus": "SUCCEEDED",
            "startTime": "2025-02-06T11:24:20.519000-08:00",
            "endTime": "2025-02-06T11:24:22.238000-08:00",
            "durationInSeconds": 1,
            "contexts": [{
                "statusCode": "",
                "message": ""
            }]
        },
        {
            "phaseType": "RUNNING_SANDBOX",
            "phaseStatus": "TIMED_OUT",
            "startTime": "2025-02-06T11:24:22.238000-08:00",
            "endTime": "2025-02-06T11:39:21.560000-08:00",
            "durationInSeconds": 899,
            "contexts": [{
                "statusCode": "BUILD_TIMED_OUT",
                "message": "Build has timed out. "
            }]
```

```
                },
                {
                    "phaseType": "COMPLETED",
                    "startTime": "2025-02-06T11:39:21.560000-08:00"
                }
            ],
            "logs": {
                "groupName": "group",
                "streamName": "stream/0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
                "deepLink": "https://console.aws.amazon.com/cloudwatch/
home?region=us-west-2#logsV2:log-groups/log-group/group/log-events/stream
$252F0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
                "s3DeepLink": "https://s3.console.aws.amazon.com/s3/object/
codefactory-test-pool-1-us-west-2-beta-default-build-logs/0103e0e7-52aa-4a3d-81dd-
bfc27226fa54.gz?region=us-west-2",
                "cloudWatchLogsArn": "arn:aws:logs:us-west-2:962803963624:log-
group:group:log-stream:stream/0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
                "s3LogsArn": "arn:aws:s3:::codefactory-test-pool-1-us-west-2-beta-
default-build-logs/0103e0e7-52aa-4a3d-81dd-bfc27226fa54.gz",
                "cloudWatchLogs": {
                    "status": "ENABLED",
                    "groupName": "group",
                    "streamName": "stream"
                },
                "s3Logs": {
                    "status": "ENABLED",
                    "location": "codefactory-test-pool-1-us-west-2-beta-default-
build-logs",
                    "encryptionDisabled": false
                }
            }
        }
    }],
    "sandboxesNotFound": []
}
```


샌드박스 중지(AWS CLI)

CLI command

```
aws codebuild stop-sandbox --id $SANDBOX-ID
```

- `--id`: `sandboxId` 또는 `sandboxArn`

Sample request

```
aws codebuild stop-sandbox --id "arn:aws:codebuild:us-west-2:962803963624:sandbox/
project-name"
```

Sample response

```
{
    "id": "project-name",
    "arn": "arn:aws:codebuild:us-west-2:962803963624:sandbox/project-name",
    "projectName": "project-name",
    "requestTime": "2025-02-06T11:24:15.560000-08:00",
    "status": "STOPPING",
    "source": {
        "type": "S3",
        "location": "arn:aws:s3:::cofa-e2e-test-1-us-west-2-beta-default-build-
sources/eb-sample-jetty-v4.zip",
        "insecureSsl": false
    },
    "environment": {
        "type": "LINUX_CONTAINER",
        "image": "aws/codebuild/standard:6.0",
        "computeType": "BUILD_GENERAL1_SMALL",
        "environmentVariables": [{
                "name": "foo",
                "value": "bar",
                "type": "PLAINTEXT"
            },
            {
                "name": "bar",
                "value": "baz",
                "type": "PLAINTEXT"
            }
        ],
        "privilegedMode": false,
        "imagePullCredentialsType": "CODEBUILD"
    },
    "timeoutInMinutes": 10,
    "queuedTimeoutInMinutes": 480,
    "logConfig": {
```

```
        "cloudWatchLogs": {
            "status": "ENABLED",
            "groupName": "group",
            "streamName": "stream"
        },
        "s3Logs": {
            "status": "ENABLED",
            "location": "codefactory-test-pool-1-us-west-2-beta-default-build-logs",
            "encryptionDisabled": false
        }
    },
    "encryptionKey": "arn:aws:kms:us-west-2:962803963624:alias/SampleEncryptionKey",
    "serviceRole": "arn:aws:iam::962803963624:role/BuildExecutionServiceRole",
    "currentSession": {
        "id": "0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
        "currentPhase": "RUN_SANDBOX",
        "status": "STOPPING",
        "startTime": "2025-02-06T11:24:15.626000-08:00",
        "phases": [{
                "phaseType": "SUBMITTED",
                "phaseStatus": "SUCCEEDED",
                "startTime": "2025-02-08T14:33:26.144000-08:00",
                "endTime": "2025-02-08T14:33:26.173000-08:00",
                "durationInSeconds": 0
            },
            {
                "phaseType": "QUEUED",
                "phaseStatus": "SUCCEEDED",
                "startTime": "2025-02-08T14:33:26.173000-08:00",
                "endTime": "2025-02-08T14:33:26.702000-08:00",
                "durationInSeconds": 0
            },
            {
                "phaseType": "PROVISIONING",
                "phaseStatus": "SUCCEEDED",
                "startTime": "2025-02-08T14:33:26.702000-08:00",
                "endTime": "2025-02-08T14:33:30.530000-08:00",
                "durationInSeconds": 3,
                "contexts": [{
                    "statusCode": "",
                    "message": ""
                }]
            },
            {
```

```
                "phaseType": "DOWNLOAD_SOURCE",
                "phaseStatus": "SUCCEEDED",
                "startTime": "2025-02-08T14:33:30.530000-08:00",
                "endTime": "2025-02-08T14:33:33.478000-08:00",
                "durationInSeconds": 2,
                "contexts": [{
                    "statusCode": "",
                    "message": ""
                }]
            },
            {
                "phaseType": "RUN_SANDBOX",
                "startTime": "2025-02-08T14:33:33.478000-08:00"
            }
        ],
        "logs": {
            "groupName": "group",
            "streamName": "stream/0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
            "deepLink": "https://console.aws.amazon.com/cloudwatch/home?
region=us-west-2#logsV2:log-groups/log-group/group/log-events/stream
$252F0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
            "s3DeepLink": "https://s3.console.aws.amazon.com/s3/object/
codefactory-test-pool-1-us-west-2-beta-default-build-logs/0103e0e7-52aa-4a3d-81dd-
bfc27226fa54.gz?region=us-west-2",
            "cloudWatchLogsArn": "arn:aws:logs:us-west-2:962803963624:log-
group:group:log-stream:stream/0103e0e7-52aa-4a3d-81dd-bfc27226fa54",
            "s3LogsArn": "arn:aws:s3:::codefactory-test-pool-1-us-west-2-beta-
default-build-logs/0103e0e7-52aa-4a3d-81dd-bfc27226fa54.gz",
            "cloudWatchLogs": {
                "status": "ENABLED",
                "groupName": "group",
                "streamName": "stream"
            },
            "s3Logs": {
                "status": "ENABLED",
                "location": "codefactory-test-pool-1-us-west-2-beta-default-build-
logs",
                "encryptionDisabled": false
            }
        }
    }
}
```

명령 실행 시작(AWS CLI)

CLI command

```
aws codebuild start-command-execution --command $COMMAND --type $TYPE --sandbox-id
 $SANDBOX-ID
```

- `--command`: 실행해야 하는 명령
- `--sandbox-id`: `sandboxId` 또는 `sandboxArn`
- `--type`: 명령 유형 `SHELL`

Sample request

```
aws codebuild start-command-execution --command "echo "Hello World"" --type SHELL --
sandbox-id "arn:aws:codebuild:us-west-2:962803963624:sandbox/project-name
```

Sample response

```
{
    "id": "e1c658c2-02bb-42a8-9abb-94835241fcd6",
    "sandboxId": "f7126a4a-b0d5-452f-814c-fea73718f805",
    "submitTime": "2025-02-06T20:12:02.683000-08:00",
    "status": "SUBMITTED",
    "command": "echo \"Hello World\"",
    "type": "SHELL",
    "logs": {
        "groupName": "group",
        "streamName": "stream",
        "deepLink": "https://console.aws.amazon.com/cloudwatch/home?region=us-
west-2#logsV2:log-groups/log-group/group/log-events/stream",
        "s3DeepLink": "https://s3.console.aws.amazon.com/s3/object/codefactory-test-
pool-1-us-west-2-beta-default-build-logs/f7126a4a-b0d5-452f-814c-fea73718f805.gz?
region=us-west-2",
        "cloudWatchLogsArn": "arn:aws:logs:us-west-2:962803963624:log-
group:group:log-stream:stream",
        "s3LogsArn": "arn:aws:s3:::codefactory-test-pool-1-us-west-2-beta-default-
build-logs/f7126a4a-b0d5-452f-814c-fea73718f805.gz",
        "cloudWatchLogs": {
            "status": "ENABLED",
            "groupName": "group",
            "streamName": "stream"
```

```
        },
        "s3Logs": {
            "status": "ENABLED",
            "location": "codefactory-test-pool-1-us-west-2-beta-default-build-logs",
            "encryptionDisabled": false
        }
    }
}
```

명령 실행에 대한 정보 가져오기(AWS CLI)

CLI command

```
aws codebuild batch-get-command-executions --command-execution-ids $COMMAND-IDs --
sandbox-id $SANDBOX-IDs
```

- `--command-execution-ids`: commandExecutionIds의 쉼표로 구분된 값 목록
- `--sandbox-id`: sandboxId 또는 sandboxArn

Sample request

```
aws codebuild batch-get-command-executions --command-execution-
ids"c3c085ed-5a8f-4531-8e95-87d547f27ffd" --sandbox-id "arn:aws:codebuild:us-
west-2:962803963624:sandbox/project-name"
```

Sample response

```
{
    "commandExecutions": [{
        "id": "c3c085ed-5a8f-4531-8e95-87d547f27ffd",
        "sandboxId": "cd71e456-2a4c-4db4-ada5-da892b0bba05",
        "submitTime": "2025-02-10T20:18:17.118000-08:00",
        "startTime": "2025-02-10T20:18:17.939000-08:00",
        "endTime": "2025-02-10T20:18:17.976000-08:00",
        "status": "SUCCEEDED",
        "command": "echo \"Hello World\"",
        "type": "SHELL",
        "exitCode": "0",
        "standardOutputContent": "Hello World\n",
        "logs": {
```

```
            "groupName": "group",
            "streamName": "stream",
            "deepLink": "https://console.aws.amazon.com/cloudwatch/home?region=us-
west-2#logsV2:log-groups/log-group/group/log-events/stream",
            "s3DeepLink": "https://s3.console.aws.amazon.com/s3/object/
codefactory-test-pool-1-us-west-2-beta-default-build-logs/cd71e456-2a4c-4db4-ada5-
da892b0bba05.gz?region=us-west-2",
            "cloudWatchLogsArn": "arn:aws:logs:us-west-2:962803963624:log-
group:group:log-stream:stream",
            "s3LogsArn": "arn:aws:s3:::codefactory-test-pool-1-us-west-2-beta-
default-build-logs/cd71e456-2a4c-4db4-ada5-da892b0bba05.gz",
            "cloudWatchLogs": {
                "status": "ENABLED",
                "groupName": "group",
                "streamName": "stream"
            },
            "s3Logs": {
                "status": "ENABLED",
                "location": "codefactory-test-pool-1-us-west-2-beta-default-build-
logs",
                "encryptionDisabled": false
            }
        }
    }],
    "commandExecutionsNotFound": []
}
```

샌드박스에 대한 명령 실행 나열(AWS CLI)

CLI command

```
aws codebuild list-command-executions-for-sandbox --sandbox-id $SANDBOX-ID --next-
token $NEXT_TOKEN --max-results $MAX_RESULTS --sort-order $SORT_ORDER
```

- `--next-token`: 페이지가 지정된 결과(있는 경우)를 가져오는 다음 토큰. 목록 샌드박스의 이전 실행에서 이 값을 가져옵니다.
- `--max-results`: (선택 사항) 검색할 최대 샌드박스 레코드 수
- `--sort-order`: 샌드박스 레코드를 검색해야 하는 순서

Sample request

```
aws codebuild list-command-executions-for-sandbox --sandbox-id
 "arn:aws:codebuild:us-west-2:962803963624:sandbox/project-name"
```

Sample response

```
{
    "commandExecutions": [{
            "id": "aad6687e-07bc-45ab-a1fd-f5440229b528",
            "sandboxId": "cd71e456-2a4c-4db4-ada5-da892b0bba05",
            "submitTime": "2025-02-10T20:18:35.304000-08:00",
            "startTime": "2025-02-10T20:18:35.615000-08:00",
            "endTime": "2025-02-10T20:18:35.651000-08:00",
            "status": "FAILED",
            "command": "fail command",
            "type": "SHELL",
            "exitCode": "127",
            "standardErrContent": "/codebuild/output/tmp/script.sh: 4: fail: not
 found\n",
            "logs": {
                "groupName": "group",
                "streamName": "stream",
                "deepLink": "https://console.aws.amazon.com/cloudwatch/home?
region=us-west-2#logsV2:log-groups/log-group/group/log-events/stream",
                "s3DeepLink": "https://s3.console.aws.amazon.com/s3/object/
codefactory-test-pool-1-us-west-2-beta-default-build-logs/cd71e456-2a4c-4db4-ada5-
da892b0bba05.gz?region=us-west-2",
                "cloudWatchLogsArn": "arn:aws:logs:us-west-2:962803963624:log-
group:group:log-stream:stream",
                "s3LogsArn": "arn:aws:s3:::codefactory-test-pool-1-us-west-2-beta-
default-build-logs/cd71e456-2a4c-4db4-ada5-da892b0bba05.gz",
                "cloudWatchLogs": {
                    "status": "ENABLED",
                    "groupName": "group",
                    "streamName": "stream"
                },
                "s3Logs": {
                    "status": "ENABLED",
                    "location": "codefactory-test-pool-1-us-west-2-beta-default-
build-logs",
                    "encryptionDisabled": false
                }
```

```
            }
        },
        {
            "id": "c3c085ed-5a8f-4531-8e95-87d547f27ffd",
            "sandboxId": "cd71e456-2a4c-4db4-ada5-da892b0bba05",
            "submitTime": "2025-02-10T20:18:17.118000-08:00",
            "startTime": "2025-02-10T20:18:17.939000-08:00",
            "endTime": "2025-02-10T20:18:17.976000-08:00",
            "status": "SUCCEEDED",
            "command": "echo \"Hello World\"",
            "type": "SHELL",
            "exitCode": "0",
            "standardOutputContent": "Hello World\n",
            "logs": {
                "groupName": "group",
                "streamName": "stream",
                "deepLink": "https://console.aws.amazon.com/cloudwatch/home?
region=us-west-2#logsV2:log-groups/log-group/group/log-events/stream",
                "s3DeepLink": "https://s3.console.aws.amazon.com/s3/object/
codefactory-test-pool-1-us-west-2-beta-default-build-logs/cd71e456-2a4c-4db4-ada5-
da892b0bba05.gz?region=us-west-2",
                "cloudWatchLogsArn": "arn:aws:logs:us-west-2:962803963624:log-
group:group:log-stream:stream",
                "s3LogsArn": "arn:aws:s3:::codefactory-test-pool-1-us-west-2-beta-
default-build-logs/cd71e456-2a4c-4db4-ada5-da892b0bba05.gz",
                "cloudWatchLogs": {
                    "status": "ENABLED",
                    "groupName": "group",
                    "streamName": "stream"
                },
                "s3Logs": {
                    "status": "ENABLED",
                    "location": "codefactory-test-pool-1-us-west-2-beta-default-
build-logs",
                    "encryptionDisabled": false
                }
            }
        }
    ]
}
```

샌드박스 나열(AWS CLI)

CLI command

```
aws codebuild list-sandboxes --next-token $NEXT_TOKEN --max-results $MAX_RESULTS --
sort-order $SORT_ORDER
```

Sample request

```
aws codebuild list-sandboxes
```

Sample response

```
{
    "ids": [
        "s3-log-project-integ-test-temp1739250628149 85d64e0f-7880-41df-9a3c-
fb6597a266d2:827a5243-0841-4b69-a720-4438796f6967",
        "s3-log-project-integ-test-temp1739249999716bbd438dd-8bb8-47bd-
ba6b-0133ac65b3d3:e2fa4eab-73af-42e3-8903-92fddaf9f378",
        "s3-log-project-integ-test-
temp17392474779450fbdacc2-2d6e-4190-9ad5-28f891bb7415:cd71e456-2a4c-4db4-ada5-
da892b0bba05",
        "s3-log-project-integ-test-temp1739224628416430 1421c-5030-4fa1-b4d3-
ca15e44771c5:9e26ab3f-65e4-4896-a19c-56b1a95e630a",
        "s3-log-project-integ-test-temp17392136731949705 6d8d-6d8e-4f5a-a37c-
a62f5686731f:22d91b06-df1e-4e9c-a664-c0abb8d5920b",
        "s3-log-project-integ-test-temp1739213439503f6283f19-390c-4dc8-95a9-
c8480113384a:82cc413e-fc46-47ab-898f-ae23c83a613f",
        "s3-log-project-integ-test-temp1739054385570b1f1ddc2-0a23-4062-
bd0c-24e9e4a99b99:c02562f3-2396-42ec-98da-38e3fe5da13a",
        "s3-log-project-integ-test-temp173905400540237dab1ac-1fde-4dfb-a8f5-
c0114333dc89:d2f30493-f65e-4fa0-a7b6-08a5e77497b9",
        "s3-log-project-integ-test-
temp17390534055719c534090-7bc4-48f1-92c5-34acaec5bf1e:df5f1c8a-f017-43b7-91ba-
ad2619e2c059",
        "s3-log-project-integ-test-temp1739052719086a61813cc-
ebb9-4db4-9391-7f43cc984ee4:d61917ec-8037-4647-8d52-060349272c4a",
        "s3-log-project-integ-test-temp173898670094078b67edb-
c42f-42ed-9db2-4b5c1a5fc66a:ce33dfbc-beeb-4466-8c99-a3734a0392c7",
        "s3-log-project-integ-test-
temp17389863425584d21b7cd-32e2-4f11-9175-72c89ecaffef:046dadf0-1f3a-4d51-a2c0-
e88361924acf",
```

```
        "s3-log-project-integ-test-
temp1738985884273977ccd23-394b-46cc-90d3-7ab94cf764dc:0370dc41-9339-4b0a-91ed-51929761b244",
        "s3-log-project-integ-test-temp1738985365972241b614f-8e41-4387-
bd25-2b8351fbc9e0:076c392a-9630-47d8-85a9-116aa34edfff",
        "s3-log-project-integ-test-
temp1738985043988a51a9e2b-09d6-4d24-9c3c-1e6e21ac9fa8:6ea3949c-435b-4177-
aa4d-614d5956244c",
        "s3-log-project-integ-test-temp1738984123354c68b31ad-49d1-4f4b-981d-
b66c00565ff6:6c3fff6c-815b-48b5-ada3-737400a6dee8",
        "s3-log-project-integ-test-
temp1738977263715d4d5bf6c-370a-48bf-8ea6-905358a6cf92:968a0f54-724a-42d1-9207-6ed854b2fae8",
        "s3-log-project-integ-test-
temp173897358796816ce8d7d-2a5e-41ef-855b-4a94a8d2795d:80f9a7ce-930a-402e-934e-
d8b511d68b04",
        "s3-log-project-integ-test-temp17389730633301af5e452-0966-467c-
b684-4e36d47f568c:cabbe989-2e8a-473c-af25-32edc8c28646",
        "s3-log-project-integ-test-temp1738901503813173fd468-
b723-4d7b-9f9f-82e88d17f264:f7126a4a-b0d5-452f-814c-fea73718f805",
        "s3-log-project-integ-test-temp1738890502472c13616fb-
bd0f-4253-86cc-28b74c97a0ba:c6f197e5-3a53-45b6-863e-0e6353375437",
        "s3-log-project-integ-test-
temp173889034046836l0daf3-8da7-43c6-8580-9978432432ce:d20aa317-8838-4966-
bbfc-85b908213df1",
        "s3-log-project-integ-test-temp173888857196780b5ab8b-e54b-44fd-a222-
c5a374fffe96:ab4b9970-ffae-47a0-b3a8-7b6790008cad",
        "s3-log-project-integ-test-temp1738888336931c11d378d-e74d-49a4-
a723-3b92e6f7daac:4922f0e8-9b7d-4119-9c9f-115cd85e703e",
        "s3-log-project-integ-test-temp17388881717651612a397-c23f-4d88-
ba87-2773cd3fc0c9:be91c3fc-418e-4feb-8a3a-ba58ff8f4e8a",
        "s3-log-project-integ-test-
temp17388879727174c3c62ed-6195-4afb-8a03-59674d0e1187:a48826a8-3c0d-43c5-
a1b5-1c98a0f978e9",
        "s3-log-project-integ-test-temp1738885948597cef305e4-b8b4-46b0-a65b-
e2d0a7b83294:c050e77d-e3f8-4829-9a60-46149628fe96",
        "s3-log-project-integ-test-temp173888561463001a7d2a8-
e4e4-4434-94db-09d3da9a9e17:8c3ac3f5-7111-4297-aec9-2470d3ead873",
        "s3-log-project-integ-test-
temp1738869855076eb19cafd-04fe-41bd-8aa0-40826d0c0d27:d25be134-05cb-404a-85da-
ac5f85d2d72c",
        "s3-project-integ-test-temp1738868157467148eacfc-d39b-49fc-a137-
e55381cd2978:4909557b-c221-4814-b4b6-7d9e93d37c35",
        "s3-project-integ-test-temp1738820926895abec0af2-
e33d-473c-9cf4-2122dd9d6876:8f5cf218-71d6-40a4-a4be-6cacebd7765f",
```

```
        "s3-project-integ-test-temp1738819988775574f969a6-1c2e-4441-b463-
ab175b45ce32:04396851-c901-4986-9117-585528e3877f",
        "s3-project-integ-test-temp17388189812309abd2604-29ba-4cf6-
b6bf-073207b7db9c:540075c7-f5ec-41e8-9341-2233c09247eb",
        "s3-project-integ-test-temp1738818843474d3ea9ac1-b609-461b-
bbdb-2da245c9bc96:865d4c3c-fbfe-4ece-9c92-d0c928341404",
        "s3-project-integ-test-temp1738818542236006e9169-e6d9-4344-9b59-
f557e7aec619:1f9ffa87-da15-4290-83e2-eebdd877497b",
        "s3-project-integ-test-
temp173881809557486ad11fd-7931-48d7-81d5-499cea52a6bc:c4c2efc4-685f-4e13-8b0f-1ef85ec300b1",
        "s3-project-integ-test-temp1738817941033229410 20-3f0b-49c3-b836-
fcd818ec9484:0344cfba-de48-456d-b2a8-6566bd4a5d6e",
        "s3-project-integ-test-temp1738817680747b93d0d0b-ea16-497f-9559-
af25ee6dcfdf:654a3a55-d92a-4dc6-8da8-56fd4d40d7e1",
        "s3-project-integ-test-temp1738817402719125 5c3da-086c-4270-b047-
acac0b7bee0d:b7e82740-2c69-42fc-ab5a-dbf15bc016a1",
        "s3-project-integ-test-temp1738817099799016e7fa3-b9b5-46a2-
bcd5-0888c646743f:8705a6a4-79ff-427a-a1c3-85c4e8fe462e",
        "s3-project-integ-test-temp1738816479281bb0c3606-5ebf-4623-
bed5-12b60e9d3512:f23fc74b-a981-4835-8e28-375fcd4c99e4",
        "s3-project-integ-test-
temp1738816263585c939a133-4d37-482c-9238-1dbff34b7674:ca28e234-0045-4ae6-8732-938b17597f50",
        "s3-project-integ-test-
temp173881580873072d18733-8fe4-43b1-83f7-95f25bb27ccf:c6f0f55b-5736-47c7-
a3aa-1b8461a6d5ed"
    ]
}
```


자습서: SSH를 사용하여 샌드박스에 연결

이 자습서에서는 SSH 클라이언트를 사용하여 CodeBuild 샌드박스에 연결하는 방법을 보여줍니다.

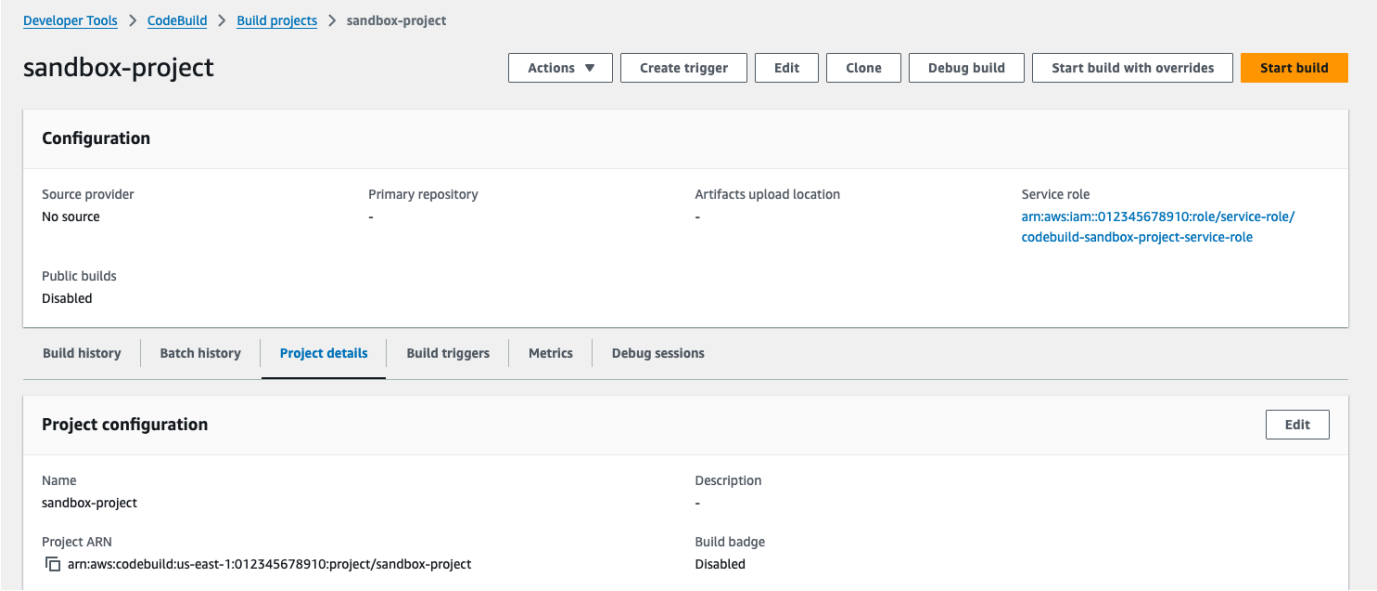
이 자습서를 완료하려면 먼저 다음을 수행해야 합니다.

- 기존 AWS CodeBuild 프로젝트가 있는지 확인합니다.
- CodeBuild 프로젝트 역할에 대해 구성된 적절한 IAM 권한을 설정합니다.
- 로컬 AWS CLI 시스템에를 설치하고 구성합니다.

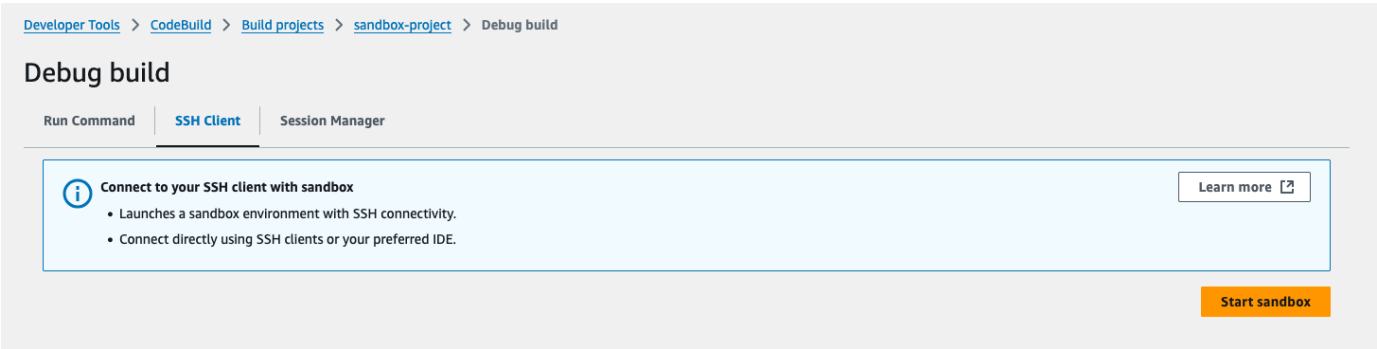
1단계: 샌드박스 시작

콘솔에서 CodeBuild 샌드박스를 시작하려면

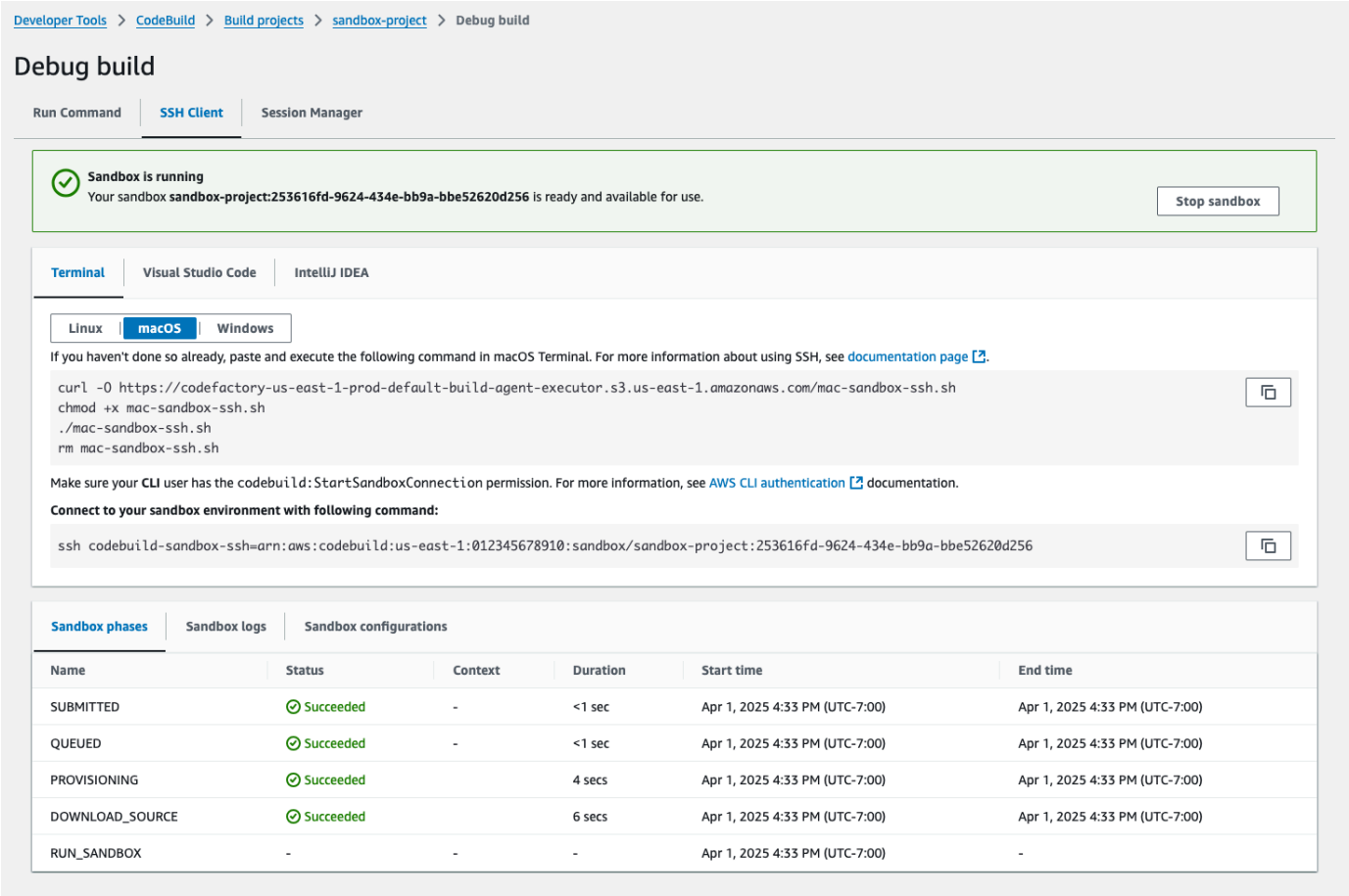
1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 프로젝트 빌드를 선택합니다. 빌드 프로젝트를 선택한 다음 빌드 디버그를 선택합니다.



3. SSH 클라이언트 탭에서 샌드박스 시작을 선택합니다.



4. 샌드박스 초기화 프로세스에 시간이 걸릴 수 있습니다. 상태가 RUN_SANDDBOX로 변경될 때 샌드박스에 연결할 수 있습니다.

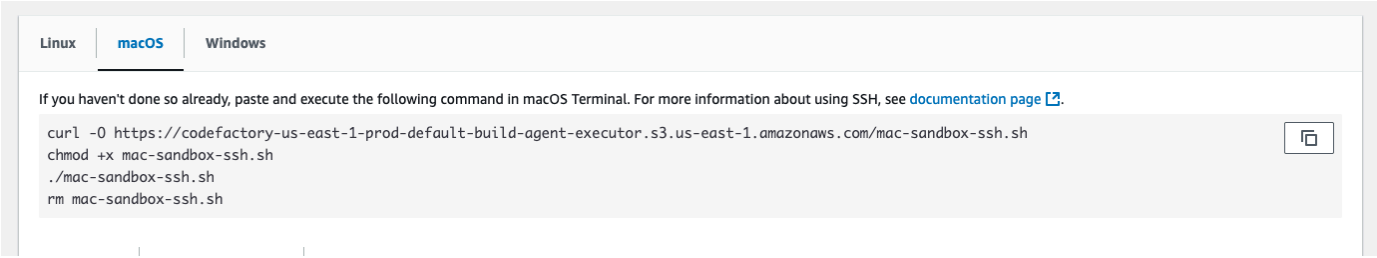


2단계: 로컬 SSH 구성 수정

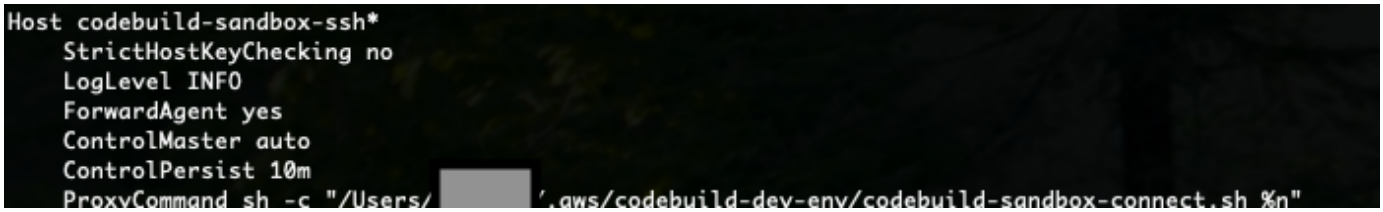
샌드박스에 처음 연결하는 경우 다음 단계를 사용하여 일회성 설정 프로세스를 수행해야 합니다.

콘솔에서 로컬 SSH 구성을 수정하려면

1. 운영 체제의 설정 명령을 찾습니다.
2. 로컬 터미널을 연 다음 제공된 명령을 복사하고 실행하여 로컬 SSH 구성을 설정할 수 있도록 스크립트를 다운로드하고 실행합니다. 예를 들어, 운영 체제가 macOS인 경우 다음 명령을 사용합니다.



3. 구성 스크립트는 샌드박스에 연결하는 데 필요한 구성을 추가합니다. 이러한 변경 사항을 수락하라는 메시지가 표시됩니다.
4. 구성에 성공하면 CodeBuild 샌드박스에 대한 새 SSH 구성 항목이 생성됩니다.



3단계: 샌드박스에 연결

콘솔에서 로컬 SSH 구성을 수정하려면

1. AWS CLI 인증을 구성하고 AWS CLI 사용자에게 `codebuild:StartSandboxConnection` 권한이 있는지 확인합니다. 자세한 내용은 AWS 명령줄 인터페이스 사용 설명서 버전 1의 AWS CLI용 IAM 사용자 자격 증명을 사용하여 인증을 참조하세요.
2. 다음 명령을 사용하여 샌드박스에 연결합니다.

```
ssh codebuild-sandbox-ssh=arn:aws:codebuild:us-east-1:<account-id>:sandbox/<sandbox-id>
```

> **Note**
>
> 연결 실패 문제를 해결하려면 `-v` 플래그를 사용하여 상세 정보 출력을 활성화합니다. 예를 들어 `ssh -v codebuild-sandbox-ssh=arn:aws:codebuild:us-east-1:<account-id>:sandbox/<sandbox-id>`입니다.
> 추가적인 문제 해결 지침은 AWS CodeBuild 샌드박스 SSH 연결 문제 해결를 참조하세요.

4단계: 결과 검토

연결되면 빌드 실패를 디버깅하고, 빌드 명령을 테스트하고, 구성 변경을 실험하고, 샌드박스를 사용하여 환경 변수 및 종속성을 확인할 수 있습니다.

AWS CodeBuild 샌드박스 SSH 연결 문제 해결

이 주제의 정보를 활용하면 CodeBuild 샌드박스 SSH 연결 문제를 식별, 진단 및 해결하는 데 도움이 됩니다.

주제

- StartSandboxConnection InvalidInputException SSH가 CodeBuild 샌드박스 환경으로 전환될 때 오류 발생
- 오류: SSH가 CodeBuild 샌드박스 환경으로 전환될 때 '자격 증명을 찾을 수 없음'
- StartSandboxConnection AccessDeniedException SSH를 CodeBuild 샌드박스 환경으로 전환할 때 오류 발생
- 오류: SSH가 CodeBuild 샌드박스 환경으로 전환될 때 'ssh: 호스트 이름을 확인할 수 없음'

StartSandboxConnection InvalidInputException SSH가 CodeBuild 샌드박스 환경으로 전환될 때 오류 발생

문제: `ssh codebuild-sandbox-ssh=<sandbox-arn>` 명령을 사용하여 CodeBuild 샌드박스 환경에 연결하려고 하면 다음과 같은 `InvalidInputException` 오류가 발생할 수 있습니다.

```
An error occurred (InvalidInputException) when calling the StartSandboxConnection
operation: Failed to start SSM session for {sandbox-arn}
User: arn:aws:sts::<account-ID>:assumed-role/<service-role-name>/AWSCodeBuild-<UUID>
is not authorized to perform: ssm:StartSession on resource.
```

```
An error occurred (InvalidInputException) when calling the StartSandboxConnection
operation: Failed to start SSM session for
sandbox <sandbox-arn>: codebuild:<UUID> is not connected.
```

가능한 원인:

- Amazon EC2 Systems Manager Agent 누락: 빌드 이미지에 SSM 에이전트가 제대로 설치되거나 구성되지 않았습니다.
- 권한 부족: CodeBuild 프로젝트 서비스 역할에 필요한 SSM 권한이 없습니다.

권장 솔루션: 빌드에 사용자 지정 이미지를 사용하는 경우 다음을 수행합니다.

1. SSM Agent 설치 자세한 내용은 에서 Linux용 Amazon EC2 인스턴스에 수동으로 SSM Agent 설치 및 제거를 참조하세요. SSM Agent는 `3.0.1295.0` 이상 버전이어야 합니다.
2. https://github.com/aws/aws-codebuild-docker-images/blob/master/ubuntu/standard/7.0/amazon-ssm-agent.json 파일을 이미지의 `/etc/amazon/ssm/` 디렉터리에 복사합니다. 이렇게 하면 SSM Agent에서 컨테이너 모드가 활성화됩니다.

3. CodeBuild 프로젝트의 서비스 역할에 다음 권한이 있는지 확인한 다음 샌드박스 환경을 다시 시작합니다.

```
{
   "Effect": "Allow",
      "Action": [
         "ssmmessages:CreateControlChannel",
         "ssmmessages:CreateDataChannel",
         "ssmmessages:OpenControlChannel",
         "ssmmessages:OpenDataChannel"
      ],
      "Resource": "*"
 },
 {
    "Effect": "Allow",
    "Action": [
       "ssm:StartSession"
     ],
     "Resource": [
        "arn:aws:codebuild:region:account-id:build/*",
        "arn:aws:ssm:region::document/AWS-StartSSHSession"
     ]
 }
```

오류: SSH가 CodeBuild 샌드박스 환경으로 전환될 때 '자격 증명을 찾을 수 없음'

문제: `ssh codebuild-sandbox-ssh=<sandbox-arn>` 명령을 사용하여 CodeBuild 샌드박스 환경에 연결하려고 하면 다음과 같은 자격 증명 오류가 발생할 수 있습니다.

```
Unable to locate credentials. You can configure credentials by running
"aws configure".
```

가능한 원인: AWS 자격 증명이 로컬 환경에서 제대로 구성되지 않았습니다.

권장 솔루션: AWS 버전 2용 명령줄 인터페이스 사용 설명서의 공식 설명서: 에 대한 설정 구성 AWS CLI에 따라 AWS CLI 자격 증명을 구성합니다.

StartSandboxConnection AccessDeniedException SSH를 CodeBuild 샌드박스 환경으로 전환할 때 오류 발생

문제: `ssh codebuild-sandbox-ssh=<sandbox-arn>` 명령을 사용하여 CodeBuild 샌드박스 환경에 연결하려고 하면 다음과 같은 권한 오류가 발생할 수 있습니다.

```
An error occurred (AccessDeniedException) when calling the StartSandboxConnection
operation:
User: arn:aws:sts::account-id:assumed-role/role-name
is not authorized to perform: codebuild:StartSandboxConnection on resource:
sandbox-arn
because no identity-based policy allows the codebuild:StartSandboxConnection action
```

가능한 원인: 자격 AWS 증명에이 작업을 수행하는 데 필요한 CodeBuild 권한이 없습니다.

권장 솔루션: 자격 AWS CLI 증명과 연결된 IAM 사용자 또는 역할에 다음 권한이 있는지 확인합니다.

```
{
    "Effect": "Allow",
    "Action": [
       "codebuild:StartSandboxConnection"
     ],
     "Resource": [
        "arn:aws:codebuild:region:account-id:sandbox/*"
     ]
}
```

오류: SSH가 CodeBuild 샌드박스 환경으로 전환될 때 'ssh: 호스트 이름을 확인할 수 없음'

문제: `ssh codebuild-sandbox-ssh=<sandbox-arn>` 명령을 사용하여 CodeBuild 샌드박스 환경에 연결하려고 하면 다음과 같은 호스트 이름 확인 오류가 발생합니다.

```
ssh: Could not resolve hostname
```

가능한 원인: 이 오류는 일반적으로 로컬 환경에서 필요한 CodeBuild 샌드박스 연결 스크립트가 제대로 실행되지 않은 경우에 발생합니다.

권장 솔루션

1. CodeBuild 샌드박스 연결 스크립트를 다운로드합니다.

2. 터미널에서 스크립트를 실행하여 필요한 SSH 구성을 설정합니다.
3. 샌드박스 환경에 대한 SSH 연결을 다시 시도합니다.

Session Manager를 사용하여 빌드 디버그

에서 실행 중인 빌드를 일시 중지한 다음 AWS Systems Manager 세션 관리자를 사용하여 빌드 컨테이너에 연결하고 컨테이너의 상태를 볼 AWS CodeBuild수 있습니다.

> **Note**
>
> 이 기능은 Windows 환경에서는 제공되지 않습니다.

주제

- 사전 조건
- 빌드 일시 중지
- 빌드를 시작합니다.
- 빌드 컨테이너에 연결
- 빌드 재개

사전 조건

Session Manager를 빌드 세션과 함께 사용할 수 있게 하려면 빌드의 세션 연결을 활성화해야 합니다. 다음과 같은 두 가지 사전 요구 사항이 있습니다.

- CodeBuild Linux 표준 큐레이트된 이미지에는 이미 SSM Agent가 설치되어 있고 SSM Agent 컨테이너 모드가 활성화되어 있습니다.

  빌드에 사용자 지정 이미지를 사용하는 경우 다음을 수행하세요.

  1. SSM Agent 설치 자세한 내용은 AWS Systems Manager 사용 설명서의 Linux용 EC2 인스턴스에 수동으로 SSM Agent 설치를 참조하세요. SSM Agent는 3.0.1295.0 이상 버전이어야 합니다.
  2. https://github.com/aws/aws-codebuild-docker-images/blob/master/ubuntu/standard/5.0/amazon-ssm-agent.json 파일을 이미지의 `/etc/amazon/ssm/` 디렉터리에 복사합니다. 이렇게 하면 SSM 에이전트에서 컨테이너 모드가 활성화됩니다.

> **Note**
>
> 이 기능이 예상대로 작동하려면 사용자 지정 이미지에 최근 업데이트한 SSM 에이전트가 필요합니다.

- CodeBuild 서비스 역할에는 다음과 같은 SSM 정책이 있어야 합니다.

  JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
     {
       "Effect": "Allow",
       "Action": [
          "ssmmessages:CreateControlChannel",
          "ssmmessages:CreateDataChannel",
          "ssmmessages:OpenControlChannel",
          "ssmmessages:OpenDataChannel"
       ],
       "Resource": "*"
     }
   ]
}
```

  빌드를 시작할 때 CodeBuild 콘솔이 이 정책을 서비스 역할에 자동으로 연결하도록 할 수 있습니다. 또는 이 정책을 서비스 역할에 수동으로 연결할 수도 있습니다.

- Systems Manager 기본 설정에서 감사 및 로깅 세션 활동을 활성화한 경우 CodeBuild 서비스 역할에도 추가 권한이 있어야 합니다. 권한은 로그가 저장되는 위치에 따라 다릅니다.

  CloudWatch Logs

  CloudWatch Logs를 사용하여 로그를 저장하는 경우 CodeBuild 서비스 역할에 다음 권한을 추가합니다.

  JSON

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

```
        {
            "Effect": "Allow",
            "Action": "logs:DescribeLogGroups",
            "Resource": "arn:aws:logs:us-east-1:111122223333:log-group:*:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:us-east-1:111122223333:log-
group:MyLogGroup:*"
        }
    ]
}
```

Amazon S3

Amazon S3를 사용하여 로그를 저장하는 경우 CodeBuild 서비스 역할에 다음 권한을 추가합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetEncryptionConfiguration",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::<bucket-name>",
        "arn:aws:s3:::<bucket-name>/*"
      ]
    }
  ]
}
```

자세한 내용은AWS Systems Manager 사용 설명서의 세션 활동 감사 및 로깅을 참조하세요.

빌드 일시 중지

빌드를 일시 중지하려면 buildspec 파일의 모든 빌드 단계에 codebuild-breakpoint 명령을 삽입합니다. 이 시점에서 빌드가 일시 중지되므로 빌드 컨테이너에 연결하여 컨테이너를 현재 상태로 볼 수 있습니다.

예를 들어, buildspec 파일의 빌드 단계에 다음을 추가합니다.

```
phases:
  pre_build:
    commands:
      - echo Entered the pre_build phase...
      - echo "Hello World" > /tmp/hello-world
      - codebuild-breakpoint
```

이 코드는 `/tmp/hello-world` 파일을 생성한 다음, 이 시점에서 빌드를 일시 중지합니다.

빌드를 시작합니다.

Session Manager를 빌드 세션과 함께 사용할 수 있게 하려면 빌드의 세션 연결을 활성화해야 합니다. 이렇게 하려면 빌드를 시작할 때 다음 절차를 따르세요.

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 프로젝트 빌드를 선택합니다. 빌드 프로젝트를 선택한 다음, 재정의로 빌드 시작을 선택합니다.
3. Advanced build overrides(고급 빌드 재정의)를 선택합니다.
4. 환경 섹션에서 세션 연결 활성화 옵션을 선택합니다. 이 옵션을 선택하지 않으면 codebuild-breakpoint 및 codebuild-resume 명령이 모두 무시됩니다.
5. 다른 사항을 원하는 대로 변경하고 빌드 시작을 선택합니다.
6. 콘솔에서 빌드 상태를 모니터링합니다. 세션을 사용할 수 있게 되면 빌드 상태 섹션에 AWS Session Manager 링크가 나타납니다.

빌드 컨테이너에 연결

다음 두 방법 중 하나로 빌드 컨테이너에 연결할 수 있습니다.

CodeBuild 콘솔

웹 브라우저에서 AWS Session Manager 링크를 열어 빌드 컨테이너에 연결합니다. 빌드 컨테이너를 탐색하고 제어할 수 있는 터미널 세션이 열립니다.

AWS CLI

> **Note**
>
> 이 절차를 수행하려면 로컬 컴퓨터에 Session Manager 플러그인이 설치되어 있어야 합니다. 자세한 내용은 AWS Systems Manager 사용 설명서의 AWS CLI용 세션 관리자 플러그인 설치를 참조하세요.

1. 빌드 ID로 batch-get-builds API를 호출하여 세션 대상 식별자를 비롯한 빌드 관련 정보를 가져옵니다. 세션 대상 식별자 속성 이름은 aws 명령의 출력 유형에 따라 달라집니다. 바로 이 때문에 `--output json`이 명령에 추가된 것입니다.

   ```
   aws codebuild batch-get-builds --ids <buildID> --region <region> --output json
   ```

2. `sessionTarget` 속성 값을 복사합니다. `sessionTarget` 속성 이름은 aws 명령의 출력 유형에 따라 달라질 수 있습니다. 바로 이 때문에 `--output json`이 이전 단계의 명령에 추가된 것입니다.
3. 다음 명령을 사용하여 빌드 컨테이너에 연결합니다.

   ```
   aws ssm start-session --target <sessionTarget> --region <region>
   ```

이 예제에서는 `/tmp/hello-world` 파일이 존재하고 텍스트 `Hello World`가 포함되어 있는지 확인하세요.

빌드 재개

빌드 컨테이너 검사를 마친 후 컨테이너 쉘에서 codebuild-resume 명령을 실행합니다.

```
$ codebuild-resume
```

에서 빌드 삭제 AWS CodeBuild

AWS CLI 또는 AWS SDKs를 사용하여 빌드를 삭제할 수 있습니다 AWS CodeBuild.

주제

- 빌드 삭제(AWS CLI)
- 빌드 삭제(AWS SDKs)

빌드 삭제(AWS CLI)

batch-delete-builds 명령을 실행합니다.

```
aws codebuild batch-delete-builds --ids ids
```

이전 명령에서 다음 자리표시자를 바꿉니다.

- *ids*: 필수 문자열입니다. 삭제할 빌드의 ID입니다. 여러 개의 빌드를 지정하려면 각 빌드 ID를 공백으로 구분합니다. 빌드 ID 목록을 가져오려면 다음 주제를 참조하십시오.
  - 빌드 ID 목록 보기(AWS CLI)
  - 빌드 프로젝트의 빌드 ID 목록 보기(AWS CLI)

성공하면 성공적으로 삭제된 각 빌드의 Amazon 리소스 이름(ARN)이 포함된 buildsDeleted 배열이 출력에 나타납니다. 성공적으로 삭제되지 않은 빌드에 대한 정보는 buildsNotDeleted 배열 내 출력에 표시됩니다.

예를 들면 다음 명령을 실행하는 경우

```
aws codebuild batch-delete-builds --ids my-demo-build-project:f8b888d2-5e1e-4032-8645-
b115195648EX my-other-demo-build-project:a18bc6ee-e499-4887-b36a-8c90349c7eEX
```

다음과 비슷한 정보가 출력에 나타납니다.

```
{
  "buildsNotDeleted": [
    {
      "id": "arn:aws:codebuild:us-west-2:123456789012:build/my-demo-build-
project:f8b888d2-5e1e-4032-8645-b115195648EX",
      "statusCode": "BUILD_IN_PROGRESS"
    }
  ],
  "buildsDeleted": [
    "arn:aws:codebuild:us-west-2:123456789012:build/my-other-demo-build-
project:a18bc6ee-e499-4887-b36a-8c90349c7eEX"
```

```
  ]
}
```

빌드 삭제(AWS SDKs)

를 SDK와 AWS CodeBuild 함께 사용하는 방법에 대한 자세한 내용은 섹션을 참조하세요AWS SDKs 및 도구 참조. AWS SDKs

에서 수동으로 빌드 재시도 AWS CodeBuild

AWS CodeBuild 콘솔 AWS CLI, 또는 AWS SDKs를 사용하여 단일 빌드 또는 배치 빌드를 수동으로 재시도할 수 있습니다 AWS CodeBuild.

주제

- 빌드를 수동으로 재시도(콘솔)
- 빌드를 수동으로 재시도(AWS CLI)
- 빌드 수동 재시AWS SDKs)

빌드를 수동으로 재시도(콘솔)

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 다음 중 하나를 수행하세요.
   - ***build-project-name*:*build-ID*** 페이지가 표시되면 빌드 재시도를 선택합니다.
   - 탐색 창에서 [Build history]를 선택합니다. 빌드 목록에서 해당 빌드의 확인란을 선택한 다음, 빌드 재시도를 선택합니다.
   - 탐색 창에서 프로젝트 빌드를 선택합니다. 빌드 프로젝트 목록의 이름 열에서 빌드 프로젝트 이름에 대한 링크를 선택합니다. 빌드 목록에서 해당 빌드의 확인란을 선택한 다음, 빌드 재시도를 선택합니다.

> **Note**
>
> 기본적으로 가장 최근에 실행한 100개의 최신 빌드 또는 빌드 프로젝트가 표시됩니다. 더 많은 빌드 또는 빌드 프로젝트를 보려면 기어 아이콘을 선택한 다음 페이지당 빌드 수 또는 페이지당 프로젝트 수에서 다른 값을 선택하거나 뒤로 및 앞으로 화살표를 선택합니다.

빌드를 수동으로 재시도(AWS CLI)

- retry-build 명령을 실행합니다.

```
aws codebuild retry-build --id <build-id> --idempotency-token <idempotencyToken>
```

이전 명령에서 다음 자리표시자를 바꿉니다.

- *<build-id>*: 필수 문자열입니다. 재시도할 빌드 또는 배치 빌드의 ID입니다. 빌드 ID 목록을 가져오려면 다음 주제를 참조하십시오.
  - 빌드 ID 목록 보기(AWS CLI)
  - 배치 빌드 ID 목록 보기(AWS CLI)
  - 빌드 프로젝트의 빌드 ID 목록 보기(AWS CLI)
  - 빌드 프로젝트의 배치 빌드 ID 목록 보기(AWS CLI)
- `--idempotency-token`: 선택 사항입니다. retry-build 명령을 옵션과 함께 실행하면 고유의 대소문자 구분 식별자 또는 토큰이 `retry-build` 요청에 포함됩니다. 토큰은 요청 후 5분 동안 유효합니다. 동일한 토큰을 사용하여 `retry-build` 요청을 반복하고 파라미터를 변경하면 CodeBuild는 파라미터 불일치 오류를 반환합니다.

빌드 수동 재시AWS SDKs)

를 SDK와 AWS CodeBuild 함께 사용하는 방법에 대한 자세한 내용은 섹션을 참조하세요AWS SDKs 및 도구 참조. AWS SDKs

에서 자동으로 빌드 재시도 AWS CodeBuild

AWS CodeBuild 콘솔 AWS CLI, 또는 AWS SDKs를 사용하여 빌드를 자동으로 재시도할 수 있습니다 AWS CodeBuild. 자동 재시도를 활성화하면 CodeBuild는 빌드가 실패한 후 프로젝트의 서비스 역할을 사용하여 지정된 한도까지 자동으로 `RetryBuild`를 호출합니다. 예를 들어 자동 재시도 제한이 2로 설정된 경우 CodeBuild는 `RetryBuild` API를 호출하여 빌드를 추가로 최대 2회 자동 재시도합니다.

> **Note**
>
> CodeBuild는 CodePipeline에 대한 자동 재시도를 지원하지 않습니다.

주제

- 빌드 자동 재시도(콘솔)
- 빌드 자동 재시도(AWS CLI)
- 빌드 자동 재시AWS SDKs)

빌드 자동 재시도(콘솔)

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 프로젝트 생성을 선택합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 및 빌드 실행(콘솔) 섹션을 참조하세요.
   - 환경에서 다음과 같이 합니다.
     - 자동 재시도 제한에 빌드 실패 후 원하는 최대 자동 재시도 횟수를 입력합니다.
3. 환경에서 추가 구성을 선택합니다.
4. 기본값으로 계속 진행한 다음 빌드 프로젝트 생성을 선택합니다.

빌드 자동 재시도(AWS CLI)

- create-project 명령을 실행합니다.

```
aws codebuild create-project \
    --name "<project-name>" \
    --auto-retry-limit <auto-retry-limit> \
    --source "<source>" \
    --artifacts {<artifacts>} \
    --environment "{\"type\": \"environment-type>\",\"image\": \"image-type>\",
\"computeType\": \"compute-type>\"}" \
    --service-role "service-role>"
```

이전 명령에서 다음 자리표시자를 바꿉니다.

- *<auto-retry-limit>*: 자동 재시도 제한을 빌드 실패 후 원하는 최대 자동 재시도 횟수로 설정합니다.
- *<project-name>*, *<source>*, *<artifacts>*, *environment-type>*, *image-type>*, *<compute-type>* 및 *<service-role>*: 원하는 프로젝트 구성 설정을 설정합니다.

빌드 자동 재시AWS SDKs)

를 SDK와 AWS CodeBuild 함께 사용하는 방법에 대한 자세한 내용은 섹션을 참조하세요AWS SDKs 및 도구 참조. AWS SDKs

에서 빌드 중지 AWS CodeBuild

AWS CodeBuild 콘솔 AWS CLI, 또는 AWS SDKs를 사용하여에서 빌드를 중지할 수 있습니다 AWS CodeBuild.

주제

- 빌드 중지(콘솔)
- 빌드 중지(AWS CLI)
- 빌드 중지(AWS SDKs)

빌드 중지(콘솔)

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 다음 중 하나를 수행하세요.
   - ***build-project-name*:*build-ID*** 페이지가 표시되면 빌드 중지를 선택합니다.
   - 탐색 창에서 [Build history]를 선택합니다. 빌드 목록에서 해당 빌드의 확인란을 선택한 다음, 빌드 중지를 선택합니다.
   - 탐색 창에서 프로젝트 빌드를 선택합니다. 빌드 프로젝트 목록의 이름 열에서 빌드 프로젝트 이름에 대한 링크를 선택합니다. 빌드 목록에서 해당 빌드의 확인란을 선택한 다음, 빌드 중지를 선택합니다.

> **Note**
>
> 기본적으로 가장 최근에 실행한 100개의 최신 빌드 또는 빌드 프로젝트가 표시됩니다. 더 많은 빌드 또는 빌드 프로젝트를 보려면 기어 아이콘을 선택한 다음 페이지당 빌드 수 또는 페이지당 프로젝트 수에서 다른 값을 선택하거나 뒤로 및 앞으로 화살표를 선택합니다.
> 가 빌드를 성공적으로 중지 AWS CodeBuild 할 수 없는 경우(예: 빌드 프로세스가 이미 완료된 경우) 중지 버튼이 비활성화되었거나 표시되지 않을 수 있습니다.

빌드 중지(AWS CLI)

- stop-build 명령을 실행합니다.

```
aws codebuild stop-build --id id
```

이전 명령에서 다음 자리표시자를 바꿉니다.

- *id*: 필수 문자열입니다. 중지할 빌드의 ID입니다. 빌드 ID 목록을 가져오려면 다음 주제를 참조하십시오.
  - 빌드 ID 목록 보기(AWS CLI)
  - 빌드 프로젝트의 빌드 ID 목록 보기(AWS CLI)

가 빌드를 AWS CodeBuild 성공적으로 중지하면 출력에 있는 `build` 객체의 `buildStatus` 값은 입니다`STOPPED`.

CodeBuild가 빌드를 성공적으로 중지할 수 없는 경우(예: 빌드가 이미 완료된 경우) 출력에서 `build` 객체의 `buildStatus` 값이 최종 빌드 상태(예: `SUCCEEDED`)입니다.

빌드 중지(AWS SDKs)

를 SDK와 AWS CodeBuild 함께 사용하는 방법에 대한 자세한 내용은 섹션을 참조하세요AWS SDKs 및 도구 참조. AWS SDKs

에서 배치 빌드 중지 AWS CodeBuild

AWS CodeBuild 콘솔 AWS CLI, 또는 AWS SDKs를 사용하여에서 배치 빌드를 중지할 수 있습니다 AWS CodeBuild.

> **Note**
>
> 배치 빌드에서 Lambda 컴퓨팅을 사용하는 경우 진행 중인 Lambda 빌드를 중지할 수 없습니다.

주제

- 배치 빌드 중지(콘솔)
- 배치 빌드 중지(AWS CLI)
- 배치 빌드(AWS SDKs) 중지

배치 빌드 중지(콘솔)

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 다음 중 하나를 수행하세요.
   - ***build-project-name*:*build-ID*** 페이지가 표시되면 빌드 중지를 선택합니다.
   - 탐색 창에서 [Build history]를 선택합니다. 빌드 목록에서 해당 빌드의 확인란을 선택한 다음, 빌드 중지를 선택합니다.
   - 탐색 창에서 프로젝트 빌드를 선택합니다. 빌드 프로젝트 목록의 이름 열에서 빌드 프로젝트 이름에 대한 링크를 선택합니다. 빌드 목록에서 해당 빌드의 확인란을 선택한 다음, 빌드 중지를 선택합니다.

> **Note**
>
> 기본적으로 가장 최근에 실행한 100개의 최신 빌드 또는 빌드 프로젝트가 표시됩니다. 더 많은 빌드 또는 빌드 프로젝트를 보려면 기어 아이콘을 선택한 다음 페이지당 빌드 수 또는 페이지당 프로젝트 수에서 다른 값을 선택하거나 뒤로 및 앞으로 화살표를 선택합니다.

배치 빌드 중지(AWS CLI)

- `stop-build-batch` 명령을 실행합니다.

```
aws codebuild stop-build-batch --id <batch-build-id>
```

이전 명령에서 다음 자리표시자를 바꿉니다.

- *<batch-build-id>*: 필수 문자열입니다. 중지할 배치 빌드의 식별자입니다. 배치 빌드 식별자 목록을 가져오려면 다음 주제를 참조하세요.
  - 배치 빌드 ID 목록 보기(AWS CLI)
  - 빌드 프로젝트의 배치 빌드 ID 목록 보기(AWS CLI)

배치 빌드(AWS SDKs) 중지

를 SDK와 AWS CodeBuild 함께 사용하는 방법에 대한 자세한 내용은 섹션을 참조하세요AWS SDKs 및 도구 참조. AWS SDKs

자동으로 AWS CodeBuild 빌드 트리거

프로젝트에 트리거를 생성하여 1시간, 1일 또는 일주일에 한 번씩 빌드를 예약할 수 있습니다. Amazon CloudWatch cron 표현식이 포함된 사용자 지정 규칙을 사용하여 트리거를 편집할 수도 있습니다. 예를 들어 cron 표현식을 사용하여 매주 평일 특정 시간에 빌드를 예약할 수 있습니다. 트리거 생성 및 편집에 대한 자세한 내용은 AWS CodeBuild 트리거 생성 및 AWS CodeBuild 트리거 편집 섹션을 참조하십시오.

주제

- AWS CodeBuild 트리거 생성
- AWS CodeBuild 트리거 편집

AWS CodeBuild 트리거 생성

프로젝트에 트리거를 생성하여 1시간, 1일 또는 일주일에 한 번씩 빌드를 예약할 수 있습니다. Amazon CloudWatch cron 표현식이 포함된 사용자 지정 규칙을 사용하여 트리거를 생성할 수도 있습니다. 예를 들어 cron 표현식을 사용하여 매주 평일 특정 시간에 빌드를 예약할 수 있습니다.

> **Note**
>
> 빌드 트리거, Amazon EventBridge 이벤트 또는 AWS Step Functions 작업에서 배치 빌드를 시작할 수 없습니다.

주제

- AWS CodeBuild 트리거 생성(콘솔)
- 프로그래밍 방식으로 AWS CodeBuild 트리거 생성

AWS CodeBuild 트리거 생성(콘솔)

다음 절차를 통해 AWS Management Console을 사용하여 트리거를 생성합니다.

트리거를 생성하려면

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 프로젝트 빌드를 선택합니다.
3. 트리거를 추가하려는 빌드 프로젝트의 링크를 선택한 후 빌드 트리거 탭을 선택합니다.

> **Note**
>
> 기본적으로 가장 최근의 빌드 프로젝트 100개가 표시됩니다. 더 많은 빌드 프로젝트를 보려면 기어 아이콘을 선택하고 페이지당 프로젝트 수에서 다른 값을 선택하거나 뒤로 및 앞으로 화살표를 사용합니다.

4. 트리거 생성을 선택합니다.
5. 트리거 이름에 이름을 입력합니다.
6. 빈도 드롭다운 목록에서 트리거의 빈도를 선택합니다. cron 표현식으로 빈도를 만들고 싶다면, 사용자 지정을 선택하십시오.
7. 트리거의 빈도에 대한 파라미터를 지정합니다. 텍스트 상자에 선택 항목의 처음 몇 자를 입력하면 드롭다운 메뉴 항목을 필터링할 수 있습니다.

> **Note**
>
> 시작 시간과 분은 0 기준입니다. 시작 분은 0에서 59 사이의 숫자입니다. 시작 시간은 0에서 23 사이의 숫자입니다. 예를 들어, 매일 오후 12시 15분에 시작하는 일일 트리거는 시작 시간이 12이고 시작 분이 15입니다. 매일 자정에 시작하는 일일 트리거는 시작 시간이 0이고 시작 분이 0입니다. 매일 오후 11시 59분에 시작하는 일일 트리거는 시작 시간이 23이고 시작 분이 59입니다.

빈도	필요한 파라미터	세부 정보
시간당	시작 분	Start minute(시작 분) 드롭다운 메뉴를 사용합니다.
일별	시작 분 시작 시간	Start minute(시작 분) 드롭다운 메뉴를 사용합니다.

빈도	필요한 파라미터	세부 정보
		Start hour(시작 시간) 드롭다운 메뉴를 사용합니다.
주별	시작 분 시작 시간 시작 일	Start minute(시작 분) 드롭다운 메뉴를 사용합니다. Start hour(시작 시간) 드롭다운 메뉴를 사용합니다. Start day(시작 일) 드롭다운 메뉴를 사용합니다.
사용자 지정	cron 표현식	cron 표현식에 cron 표현식을 입력합니다. cron 표현식에는 각각 공백으로 구분되는 필수 필드 6개가 있습니다. 필드는 분, 시간, 일, 월, 요일 및 연도의 시작 값을 지정합니다. 와일드카드를 사용하여 범위, 추가 값 등을 지정할 수 있습니다. 예를 들어 cron 표현식 **`0 9 ? * MON-FRI *`**는 매주 평일 오전 9시에 빌드를 예약합니다. 자세한 내용은 Amazon CloudWatch Events 사용 설명서의 cron 표현식을 참조하세요.

8. Enable this trigger(이 트리거 사용)를 선택합니다.
9. (선택 사항) 고급 섹션을 확장합니다. 소스 버전에서 소스의 버전을 입력합니다.

   - Amazon S3의 경우 빌드하려는 입력 아티팩트의 버전에 해당하는 버전 ID를 입력합니다. 소스 버전을 비워 두면 최신 버전이 사용됩니다.
   - 에 커밋 ID를 AWS CodeCommit입력합니다. 소스 버전을 비워 두면 기본 브랜치의 HEAD 커밋 ID가 사용됩니다.

- GitHub 또는 GitHub Enterprise의 경우, 빌드하려는 소스 코드의 버전에 해당하는 커밋 ID, 풀 요청 ID, 분기 이름 또는 태그 이름을 입력합니다. 풀 요청 ID를 지정하는 경우 `pr/pull-request-ID` 형식을 사용해야 합니다(예: `pr/25`). 분기 이름을 지정할 경우 분기의 HEAD 커밋 ID가 사용됩니다. [Source version]이 비어 있으면 기본 분기의 HEAD 커밋 ID가 사용됩니다.
- Bitbucket의 경우, 빌드하려는 소스 코드의 버전에 해당하는 커밋 ID, 분기 이름 또는 태그 이름을 입력하십시오. 분기 이름을 지정할 경우 분기의 HEAD 커밋 ID가 사용됩니다. [Source version]이 비어 있으면 기본 분기의 HEAD 커밋 ID가 사용됩니다.

10. (선택 사항) 제한 시간을 5분에서 2160분(36시간) 사이로 지정합니다. 이 값은 빌드가 중지되기까지가 빌드를 AWS CodeBuild 시도하는 시간을 지정합니다. 시간과 분을 비워 두면 프로젝트에 지정된 기본 제한 시간 값이 사용됩니다.
11. 트리거 생성을 선택합니다.

프로그래밍 방식으로 AWS CodeBuild 트리거 생성

CodeBuild는 빌드 트리거에 대해 Amazon EventBridge 규칙을 사용합니다. EventBridge API를 사용하여 CodeBuild 프로젝트에 대한 빌드 트리거를 프로그래밍 방식으로 생성할 수 있습니다. 자세한 내용은 Amazon EventBridge API 참조를 참조하세요.

AWS CodeBuild 트리거 편집

프로젝트에서 트리거를 편집하여 시간, 일 또는 주에 한 번씩 빌드를 예약할 수 있습니다. Amazon CloudWatch cron 표현식이 포함된 사용자 지정 규칙을 사용하여 트리거를 편집할 수도 있습니다. 예를 들어 cron 표현식을 사용하여 매주 평일 특정 시간에 빌드를 예약할 수 있습니다. 트리거 생성에 대한 자세한 내용은 AWS CodeBuild 트리거 생성 섹션을 참조하세요.

주제

- AWS CodeBuild 트리거 편집(콘솔)
- 프로그래밍 방식으로 AWS CodeBuild 트리거 편집

AWS CodeBuild 트리거 편집(콘솔)

다음 절차를 통해 AWS Management Console을 사용하여 트리거를 편집합니다.

트리거를 편집하려면

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.

2. 탐색 창에서 프로젝트 빌드를 선택합니다.
3. 변경하려는 빌드 프로젝트의 링크를 선택한 다음, 빌드 트리거 탭을 선택합니다.

> **Note**
>
> 기본적으로 가장 최근의 빌드 프로젝트 100개가 표시됩니다. 더 많은 빌드 프로젝트를 보려면 기어 아이콘을 선택하고 페이지당 프로젝트 수에서 다른 값을 선택하거나 뒤로 및 앞으로 화살표를 사용합니다.

4. 변경할 트리거 옆에 있는 라디오 버튼을 선택한 다음, 편집을 선택합니다.
5. 빈도 드롭다운 목록에서 트리거의 빈도를 선택합니다. cron 표현식으로 빈도를 만들고 싶다면, 사용자 지정을 선택하십시오.
6. 트리거의 빈도에 대한 파라미터를 지정합니다. 텍스트 상자에 선택 항목의 처음 몇 자를 입력하면 드롭다운 메뉴 항목을 필터링할 수 있습니다.

> **Note**
>
> 시작 시간과 분은 0 기준입니다. 시작 분은 0에서 59 사이의 숫자입니다. 시작 시간은 0에서 23 사이의 숫자입니다. 예를 들어, 매일 오후 12시 15분에 시작하는 일일 트리거는 시작 시간이 12이고 시작 분이 15입니다. 매일 자정에 시작하는 일일 트리거는 시작 시간이 0이고 시작 분이 0입니다. 매일 오후 11시 59분에 시작하는 일일 트리거는 시작 시간이 23이고 시작 분이 59입니다.

빈도	필요한 파라미터	세부 정보
시간당	시작 분	Start minute(시작 분) 드롭다운 메뉴를 사용합니다.
일별	시작 분 시작 시간	Start minute(시작 분) 드롭다운 메뉴를 사용합니다. Start hour(시작 시간) 드롭다운 메뉴를 사용합니다.
주별	시작 분 시작 시간	Start minute(시작 분) 드롭다운 메뉴를 사용합니다.

빈도	필요한 파라미터	세부 정보
	시작 일	Start hour(시작 시간) 드롭다운 메뉴를 사용합니다. Start day(시작 일) 드롭다운 메뉴를 사용합니다.
사용자 지정	cron 표현식	cron 표현식에 cron 표현식을 입력합니다. cron 표현식에는 각각 공백으로 구분되는 필수 필드 6개가 있습니다. 필드는 분, 시간, 일, 월, 요일 및 연도의 시작 값을 지정합니다. 와일드카드를 사용하여 범위, 추가 값 등을 지정할 수 있습니다. 예를 들어 cron 표현식 **`0 9 ? * MON-FRI *`**는 매주 평일 오전 9시에 빌드를 예약합니다. 자세한 내용은 Amazon CloudWatch Events 사용 설명서의 cron 표현식을 참조하세요.

7. Enable this trigger(이 트리거 사용)를 선택합니다.

> **Note**
>
> https://console.aws.amazon.com/cloudwatch/에서 Amazon CloudWatch 콘솔을 사용하여 소스 버전, 제한 시간 및 AWS CodeBuild에서 사용할 수 없는 기타 옵션을 편집할 수 있습니다.

프로그래밍 방식으로 AWS CodeBuild 트리거 편집

CodeBuild는 빌드 트리거에 대해 Amazon EventBridge 규칙을 사용합니다. EventBridge API를 사용하여 CodeBuild 프로젝트에 대한 빌드 트리거를 프로그래밍 방식으로 편집할 수 있습니다. 자세한 내용은 Amazon EventBridge API 참조를 참조하세요.

에서 빌드 세부 정보 보기 AWS CodeBuild

AWS CodeBuild 콘솔 AWS CLI, 또는 AWS SDKs 사용하여 CodeBuild에서 관리하는 빌드에 대한 세부 정보를 볼 수 있습니다.

주제

- 빌드 세부 정보 보기(콘솔)
- 빌드 세부 정보 보기(AWS CLI)
- 빌드 세부 정보(AWS SDKs) 보기
- 빌드 단계 진행

빌드 세부 정보 보기(콘솔)

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 다음 중 하나를 수행하세요.
   - 탐색 창에서 [Build history]를 선택합니다. 빌드 목록의 빌드 실행 열에서 빌드에 대한 링크를 선택합니다.
   - 탐색 창에서 프로젝트 빌드를 선택합니다. 빌드 프로젝트 목록의 이름 열에서 빌드 프로젝트 이름에 해당하는 링크를 선택합니다. 그런 다음 빌드 목록의 빌드 실행 열에서 빌드에 대한 링크를 선택합니다.

> **Note**
>
> 기본적으로 가장 최근에 실행한 10개의 최신 빌드 또는 빌드 프로젝트가 표시됩니다. 더 많은 빌드 또는 빌드 프로젝트를 보려면 기어 아이콘을 선택한 다음 페이지당 빌드 수 또는 페이지당 프로젝트 수에서 다른 값을 선택하거나 뒤로 및 앞으로 화살표를 선택합니다.

빌드 세부 정보 보기(AWS CLI)

와 AWS CLI 함께를 사용하는 방법에 대한 자세한 내용은 단원을 AWS CodeBuild참조하십시오명령줄 참조.

batch-get-builds 명령을 실행합니다.

```
aws codebuild batch-get-builds --ids ids
```

다음과 같이 자리 표시자를 바꿉니다.

- *ids*: 필수 문자열입니다. 세부 정보를 보려는 하나 이상의 빌드 ID입니다. 둘 이상의 빌드 ID를 지정하려면 각 빌드 ID를 공백을 사용하여 구분해야 합니다. 최대 100개의 빌드 ID를 지정할 수 있습니다. 빌드 ID 목록을 가져오려면 다음 주제를 참조하십시오.
  - 빌드 ID 목록 보기(AWS CLI)
  - 빌드 프로젝트의 빌드 ID 목록 보기(AWS CLI)

예를 들면 다음 명령을 실행하는 경우

```
aws codebuild batch-get-builds --ids codebuild-demo-project:e9c4f4df-3f43-41d2-
ab3a-60fe2EXAMPLE codebuild-demo-project:815e755f-bade-4a7e-80f0-efe51EXAMPLE my-other-
project:813bb6c6-891b-426a-9dd7-6d8a3EXAMPLE
```

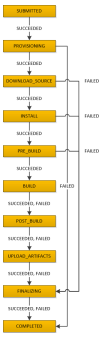
명령이 제대로 실행되면 요약된 빌드 정보를 보려면 단원에 설명된 것과 유사한 데이터가 출력에 표시됩니다.

빌드 세부 정보(AWS SDKs) 보기

를 SDK와 AWS CodeBuild 함께 사용하는 방법에 대한 자세한 내용은 섹션을 참조하세요AWS SDKs 및 도구 참조. AWS SDKs

빌드 단계 진행

의 빌드는 단계적으로 AWS CodeBuild 진행됩니다.



> **Important**
>
> BUILD 단계가 실패하더라도 UPLOAD_ARTIFACTS 단계는 항상 시도됩니다.

에서 빌드 IDs 목록 보기 AWS CodeBuild

AWS CodeBuild 콘솔 AWS CLI, 또는 AWS SDKs 사용하여 CodeBuild에서 관리하는 빌드의 빌드 IDs 목록을 볼 수 있습니다.

주제

- 빌드 ID 목록 보기(콘솔)
- 빌드 ID 목록 보기(AWS CLI)
- 배치 빌드 ID 목록 보기(AWS CLI)
- 빌드 IDs(AWS SDKs) 목록 보기

빌드 ID 목록 보기(콘솔)

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 [Build history]를 선택합니다.

> **Note**
>
> 기본적으로 가장 최근의 빌드 10개만 표시됩니다. 더 많은 빌드를 보려면 기어 아이콘을 선택하고 페이지당 빌드 수에서 다른 값을 선택하거나 뒤로 및 앞으로 화살표를 사용합니다.

빌드 ID 목록 보기(AWS CLI)

CodeBuild AWS CLI 에서를 사용하는 방법에 대한 자세한 내용은 섹션을 참조하세요명령줄 참조.

- list-builds 명령을 실행합니다.

```
aws codebuild list-builds --sort-order sort-order --next-token next-token
```

이전 명령에서 다음 자리표시자를 바꿉니다.

- *sort-order*: 빌드 ID를 나열하는 방법을 나타내는 데 사용되는 선택적 문자열입니다. 유효한 값에는 ASCENDING 및 DESCENDING(이)가 있습니다.
- *next-token*: 선택적 문자열입니다. 이전 실행 중에 목록에 100개가 넘는 항목이 있는 경우 next token이라는 고유한 문자열과 함께 처음 100개 항목만 반환됩니다. 목록에서 다음 항목 배치를 가져오려면 호출에 다음 토큰을 추가하여 이 명령을 다시 실행합니다. 목록에 있는 모든 항목을 가져오려면 다음 토큰이 더 이상 반환되지 않을 때까지 다음 토큰마다 이 명령을 계속 실행합니다.

예를 들면 다음 명령을 실행하는 경우

```
aws codebuild list-builds --sort-order ASCENDING
```

다음과 비슷한 결과가 출력에 나타납니다.

```
{
  "nextToken": "4AEA6u7J...The full token has been omitted for brevity...MzY2OA==",
  "ids": [
    "codebuild-demo-project:815e755f-bade-4a7e-80f0-efe51EXAMPLE"
    "codebuild-demo-project:84a7f3d1-d40e-4956-b4cf-7a9d4EXAMPLE"
    ... The full list of build IDs has been omitted for brevity ...
    "codebuild-demo-project:931d0b72-bf6f-4040-a472-5c707EXAMPLE"
  ]
}
```

이 명령을 다시 실행하는 경우:

```
aws codebuild list-builds --sort-order ASCENDING --next-token 4AEA6u7J...The full
 token has been omitted for brevity...MzY2OA==
```

다음과 비슷한 결과가 출력에 나타납니다.

```
{
  "ids": [
    "codebuild-demo-project:49015049-21cf-4b50-9708-df115EXAMPLE",
    "codebuild-demo-project:543e7206-68a3-46d6-a4da-759abEXAMPLE",
    ... The full list of build IDs has been omitted for brevity ...
```

```
    "codebuild-demo-project:c282f198-4582-4b38-bdc0-26f96EXAMPLE"
  ]
}
```

배치 빌드 ID 목록 보기(AWS CLI)

CodeBuild AWS CLI 에서를 사용하는 방법에 대한 자세한 내용은 섹션을 참조하세요명령줄 참조.

- list-build-batches 명령을 실행합니다.

```
aws codebuild list-build-batches --sort-order sort-order --next-token next-token
```

이전 명령에서 다음 자리표시자를 바꿉니다.

- *sort-order*: 배치 빌드 ID를 나열하는 방법을 나타내는 데 사용되는 선택적 문자열입니다. 유효한 값에는 ASCENDING 및 DESCENDING(이)가 있습니다.
- *next-token*: 선택적 문자열입니다. 이전 실행 중에 목록에 100개가 넘는 항목이 있는 경우 next token이라는 고유한 문자열과 함께 처음 100개 항목만 반환됩니다. 목록에서 다음 항목 배치를 가져오려면 호출에 다음 토큰을 추가하여 이 명령을 다시 실행합니다. 목록에 있는 모든 항목을 가져오려면 다음 토큰이 더 이상 반환되지 않을 때까지 다음 토큰마다 이 명령을 계속 실행합니다.

예를 들면 다음 명령을 실행하는 경우

```
aws codebuild list-build-batches --sort-order ASCENDING
```

다음과 비슷한 결과가 출력에 나타납니다.

```
{
  "nextToken": "4AEA6u7J...The full token has been omitted for brevity...MzY2OA==",
  "ids": [
    "codebuild-demo-project:815e755f-bade-4a7e-80f0-efe51EXAMPLE"
    "codebuild-demo-project:84a7f3d1-d40e-4956-b4cf-7a9d4EXAMPLE"
    ... The full list of build IDs has been omitted for brevity ...
    "codebuild-demo-project:931d0b72-bf6f-4040-a472-5c707EXAMPLE"
  ]
}
```

이 명령을 다시 실행하는 경우:

```
aws codebuild list-build-batches --sort-order ASCENDING --next-token 4AEA6u7J...The
 full token has been omitted for brevity...MzY2OA==
```

다음과 비슷한 결과가 출력에 나타납니다.

```
{
  "ids": [
    "codebuild-demo-project:49015049-21cf-4b50-9708-df115EXAMPLE",
    "codebuild-demo-project:543e7206-68a3-46d6-a4da-759abEXAMPLE",
    ... The full list of build IDs has been omitted for brevity ...
    "codebuild-demo-project:c282f198-4582-4b38-bdc0-26f96EXAMPLE"
  ]
}
```

빌드 IDs(AWS SDKs) 목록 보기

CodeBuild를 AWS SDKs와 함께 사용하는 방법에 대한 자세한 내용은 섹션을 참조하세요AWS SDKs 및 도구 참조.

에서 빌드 프로젝트의 빌드 IDs 목록 보기 AWS CodeBuild

AWS CodeBuild 콘솔 AWS CLI, 또는 AWS SDKs 사용하여 CodeBuild에서 빌드 프로젝트의 빌드 IDs 목록을 볼 수 있습니다.

주제

- 빌드 프로젝트의 빌드 ID 목록 보기(콘솔)
- 빌드 프로젝트의 빌드 ID 목록 보기(AWS CLI)
- 빌드 프로젝트의 배치 빌드 ID 목록 보기(AWS CLI)
- 빌드 프로젝트(AWS SDKs)의 빌드 IDs 목록 보기

빌드 프로젝트의 빌드 ID 목록 보기(콘솔)

1. https://console.aws.amazon.com/codebuild/에서 CodeBuild 콘솔을 엽니다.

2. 탐색 창에서 프로젝트 빌드를 선택합니다. 빌드 프로젝트 목록의 이름 열에서 빌드 프로젝트를 선택합니다.

> **Note**
>
> 기본적으로 가장 최근에 실행한 100개의 최신 빌드 또는 빌드 프로젝트가 표시됩니다. 더 많은 빌드 또는 빌드 프로젝트를 보려면 기어 아이콘을 선택한 다음 페이지당 빌드 수 또는 페이지당 프로젝트 수에서 다른 값을 선택하거나 뒤로 및 앞으로 화살표를 선택합니다.

빌드 프로젝트의 빌드 ID 목록 보기(AWS CLI)

와 AWS CLI 함께를 사용하는 방법에 대한 자세한 내용은 단원을 AWS CodeBuild참조하십시오명령줄 참조.

다음과 같이 list-builds-for-project 명령을 실행합니다.

```
aws codebuild list-builds-for-project --project-name project-name --sort-order sort-order --next-token next-token
```

이전 명령에서 다음 자리표시자를 바꿉니다.

- *project-name*: 빌드 ID를 나열할 빌드 프로젝트의 이름을 나타내는 데 사용되는 필수 문자열입니다. 빌드 프로젝트 목록을 가져오려면 빌드 프로젝트 이름 목록 보기(AWS CLI) 단원을 참조하십시오.
- *sort-order*: 빌드 ID를 나열하는 방법을 나타내는 데 사용되는 선택적 문자열입니다. 유효한 값에는 ASCENDING 및 DESCENDING(이)가 있습니다.
- *next-token*: 선택적 문자열입니다. 이전 실행 중에 목록에 100개가 넘는 항목이 있는 경우 next token이라는 고유한 문자열과 함께 처음 100개 항목만 반환됩니다. 목록에서 다음 항목 배치를 가져오려면 호출에 다음 토큰을 추가하여 이 명령을 다시 실행합니다. 목록에 있는 모든 항목을 가져오려면 다음 토큰이 더 이상 반환되지 않을 때까지 반환되는 다음 토큰마다 이 명령을 계속 실행합니다.

예를 들어 다음과 유사한 명령을 실행하는 경우:

```
aws codebuild list-builds-for-project --project-name codebuild-demo-project --sort-order ASCENDING
```

다음과 같은 결과가 출력에 나타날 수 있습니다.

```
{
  "nextToken": "4AEA6u7J...The full token has been omitted for brevity...MzY2OA==",
  "ids": [
    "codebuild-demo-project:9b175d16-66fd-4e71-93a0-50a08EXAMPLE"
    "codebuild-demo-project:a9d1bd09-18a2-456b-8a36-7d65aEXAMPLE"
    ... The full list of build IDs has been omitted for brevity ...
    "codebuild-demo-project:fe70d102-c04f-421a-9cfa-2dc15EXAMPLE"
  ]
}
```

이 명령을 다시 실행하는 경우:

```
aws codebuild list-builds-for-project --project-name codebuild-demo-project --
sort-order ASCENDING --next-token 4AEA6u7J...The full token has been omitted for
 brevity...MzY2OA==
```

출력에서 다음과 같은 결과를 볼 수 있습니다.

```
{
  "ids": [
    "codebuild-demo-project:98253670-7a8a-4546-b908-dc890EXAMPLE"
    "codebuild-demo-project:ad5405b2-1ab3-44df-ae2d-fba84EXAMPLE"
    ... The full list of build IDs has been omitted for brevity ...
    "codebuild-demo-project:f721a282-380f-4b08-850a-e0ac1EXAMPLE"
  ]
}
```

빌드 프로젝트의 배치 빌드 ID 목록 보기(AWS CLI)

와 AWS CLI 함께를 사용하는 방법에 대한 자세한 내용은 단원을 AWS CodeBuild참조하십시오명령줄 참조.

다음과 같이 list-build-batches-for-project 명령을 실행합니다.

```
aws codebuild list-build-batches-for-project --project-name project-name --sort-
order sort-order --next-token next-token
```

이전 명령에서 다음 자리표시자를 바꿉니다.

- *project-name*: 빌드 ID를 나열할 빌드 프로젝트의 이름을 나타내는 데 사용되는 필수 문자열입니다. 빌드 프로젝트 목록을 가져오려면 빌드 프로젝트 이름 목록 보기(AWS CLI) 단원을 참조하십시오.
- *sort-order*: 빌드 ID를 나열하는 방법을 나타내는 데 사용되는 선택적 문자열입니다. 유효한 값에는 ASCENDING 및 DESCENDING(이)가 있습니다.
- *next-token*: 선택적 문자열입니다. 이전 실행 중에 목록에 100개가 넘는 항목이 있는 경우 next token이라는 고유한 문자열과 함께 처음 100개 항목만 반환됩니다. 목록에서 다음 항목 배치를 가져오려면 호출에 다음 토큰을 추가하여 이 명령을 다시 실행합니다. 목록에 있는 모든 항목을 가져오려면 다음 토큰이 더 이상 반환되지 않을 때까지 반환되는 다음 토큰마다 이 명령을 계속 실행합니다.

예를 들어 다음과 유사한 명령을 실행하는 경우:

```
aws codebuild list-build-batches-for-project --project-name codebuild-demo-project --
sort-order ASCENDING
```

다음과 같은 결과가 출력에 나타날 수 있습니다.

```
{
  "nextToken": "4AEA6u7J...The full token has been omitted for brevity...MzY2OA==",
  "ids": [
    "codebuild-demo-project:9b175d16-66fd-4e71-93a0-50a08EXAMPLE"
    "codebuild-demo-project:a9d1bd09-18a2-456b-8a36-7d65aEXAMPLE"
    ... The full list of build IDs has been omitted for brevity ...
    "codebuild-demo-project:fe70d102-c04f-421a-9cfa-2dc15EXAMPLE"
  ]
}
```

이 명령을 다시 실행하는 경우:

```
aws codebuild list-build-batches-for-project --project-name codebuild-demo-project
 --sort-order ASCENDING --next-token 4AEA6u7J...The full token has been omitted for
 brevity...MzY2OA==
```

출력에서 다음과 같은 결과를 볼 수 있습니다.

```
{
  "ids": [
    "codebuild-demo-project:98253670-7a8a-4546-b908-dc890EXAMPLE"
    "codebuild-demo-project:ad5405b2-1ab3-44df-ae2d-fba84EXAMPLE"
```

```
    ... The full list of build IDs has been omitted for brevity ...
    "codebuild-demo-project:f721a282-380f-4b08-850a-e0ac1EXAMPLE"
  ]
}
```

빌드 프로젝트(AWS SDKs)의 빌드 IDs 목록 보기

를 SDK와 AWS CodeBuild 함께 사용하는 방법에 대한 자세한 내용은 섹션을 참조하세요AWS SDKs 및 도구 참조. AWS SDKs

의 테스트 보고서 AWS CodeBuild

CodeBuild에서 빌드 중에 실행되는 테스트에 대한 세부 정보가 포함된 보고서를 작성할 수 있습니다. 단위 테스트, 구성 테스트 및 기능 테스트와 같은 테스트를 만들 수 있습니다.

지원되는 테스트 보고서 파일 형식은 다음과 같습니다.

- Cucumber JSON(.json)
- JUnit XML(.xml)
- NUnit XML(.xml)
- NUnit3 XML(.xml)
- TestNG XML(.xml)
- Visual Studio TRX(.trx)
- Visual Studio TRX XML(.xml)

> **Note**
>
> `cucumber-js`의 지원되는 최신 버전은 7.3.2입니다.

이러한 형식 중 하나로 보고서 파일을 만들 수 있는 테스트 프레임워크로 테스트 케이스를 만듭니다 (예: Surefire JUnit plugin, TestNG, Cucumber).

테스트 보고서를 만들려면 테스트 케이스에 관한 정보와 함께 빌드 프로젝트의 buildspec 파일에 보고서 그룹 이름을 추가합니다. 빌드 프로젝트를 실행하면 테스트 케이스가 실행되고 테스트 보고서가 만들어집니다. 테스트 케이스가 실행될 때마다 새로운 테스트 보고서가 보고서 그룹에 생성됩니다. 테스트를 실행하기 전에 보고서 그룹을 만들 필요가 없습니다. 보고서 그룹 이름을 지정하는 경우 보고서를 실행할 때 CodeBuild에서 보고서 그룹을 만듭니다. 이미 존재하는 보고서 그룹을 사용하려면 buildspec 파일에서 해당 ARN을 지정합니다.

테스트 보고서를 사용하여 빌드 실행 중에 문제를 해결할 수 있습니다. 빌드 프로젝트의 여러 빌드에 많은 테스트 보고서가 있는 경우 테스트 보고서를 사용하여 추세와 테스트 및 실패율을 보고 빌드를 쉽게 최적화할 수 있습니다.

보고서는 생성되고 30일 후에 만료됩니다. 만료된 테스트 보고서는 볼 수 없습니다. 테스트 보고서를 30일 이상 보관하려면 테스트 결과의 원시 데이터 파일을 Amazon S3 버킷으로 내보내면 됩니다. 내보낸 테스트 파일은 만료되지 않습니다. S3 버킷에 대한 정보는 보고서 그룹을 생성할 때 지정됩니다.

> **Note**
>
> 프로젝트에 지정된 CodeBuild 서비스 역할은 S3 버킷에 업로드하는 권한에 사용됩니다.

주제

- 테스트 보고서 생성
- 코드 범위 보고서 생성
- CodeBuild에서 보고서 자동 검색
- 보고서 그룹
- 테스트 프레임워크
- 테스트 보고서 보기
- 테스트 보고서 권한
- 테스트 보고서 상태

테스트 보고서 생성

테스트 보고서를 만들려면 buildspec 파일에서 1~5개의 보고서 그룹으로 구성된 빌드 프로젝트를 실행합니다. 실행 중에 테스트 보고서가 작성됩니다. 보고서 그룹에 지정된 테스트 케이스의 결과가 포함되어 있습니다. 동일한 buildspec 파일을 사용하는 각 후속 빌드에 대해 새 테스트 보고서가 생성됩니다.

테스트 보고서를 작성하려면

1. 빌드 프로젝트를 생성합니다. 자세한 내용은 에서 빌드 프로젝트 생성 AWS CodeBuild 단원을 참조하세요.
2. 테스트 보고서 정보를 사용하여 프로젝트의 buildspec 파일을 구성합니다.

   a. `reports:` 섹션을 추가하고 기존 보고서 그룹의 ARN 또는 보고서 그룹의 이름을 지정합니다.

      ARN을 지정하는 경우 CodeBuild는 해당 보고서 그룹을 사용합니다.

      이름을 지정하는 경우 CodeBuild는 프로젝트 이름과 *<project-name>-<report-group-name>* 형식으로 지정한 이름을 사용하여 보고서 그룹을 생성합니다. 이름이 지정된 보고서 그룹이 이미 있는 경우 CodeBuild는 해당 보고서 그룹을 사용합니다.

b. 보고서 그룹에서 테스트 결과를 포함하는 파일의 위치를 지정합니다. 둘 이상의 보고서 그룹을 사용하는 경우 각각에 대해 테스트 결과 파일 위치를 지정합니다. 빌드 프로젝트가 실행될 때마다 새 테스트 보고서가 만들어집니다. 자세한 내용은 테스트 파일 지정 단원을 참조하십시오.

c. `build` 또는 `post_build` 시퀀스의 `commands` 섹션에서 보고서 그룹에 대해 지정한 테스트 케이스를 실행하는 명령을 지정합니다. 자세한 내용은 테스트 명령 지정 단원을 참조하십시오.

다음은 buildspec `reports` 섹션의 예입니다.

```
reports:
  php-reports:
    files:
      - "reports/php/*.xml"
    file-format: "JUNITXML"
  nunit-reports:
    files:
      - "reports/nunit/*.xml"
    file-format: "NUNITXML"
```

3. 빌드 프로젝트의 빌드를 실행합니다. 자세한 내용은 수동으로 AWS CodeBuild 빌드 실행 단원을 참조하십시오.
4. 빌드가 완료되면 프로젝트 페이지의 빌드 기록에서 새 빌드 실행을 선택하십시오. 보고서를 선택하여 테스트 보고서를 봅니다. 자세한 내용은 빌드에 대한 테스트 보고서 보기 단원을 참조하십시오.

코드 범위 보고서 생성

CodeBuild를 사용하면 테스트에 대한 코드 범위 보고서를 생성할 수 있습니다. 다음과 같은 코드 범위 보고서가 제공됩니다.

행 범위

행 범위는 테스트에서 다루는 명령문 수를 측정합니다. 명령문은 주석이나 조건문을 포함하지 않는 단일 명령입니다.

```
line coverage = (total lines covered)/(total number of lines)
```

분기 범위

분기 범위는 제어 구조의 가능한 모든 분기(예: `if` 또는 `case` 문) 중에서 테스트에 포함되는 분기 수를 측정합니다.

```
branch coverage = (total branches covered)/(total number of branches)
```

지원되는 코드 범위 보고서 파일 형식은 다음과 같습니다.

- JacoCo XML
- SimpleCov JSON[1]
- Clover XML
- Cobertura XML
- LCOV 정보

[1] CodeBuild는 simplecov-json이 아닌 simplecov에서 생성된 JSON 코드 범위 보고서를 수락합니다.

코드 범위 보고서 생성

코드 범위 보고서를 생성하려면 buildspec 파일에서 1개 이상의 코드 범위 보고서 그룹으로 구성된 빌드 프로젝트를 실행합니다. CodeBuild는 코드 범위 결과를 해석하고 실행에 대한 코드 범위 보고서를 제공합니다. 동일한 buildspec 파일을 사용하는 각 후속 빌드에 대해 새 테스트 보고서가 생성됩니다.

테스트 보고서를 작성하려면

1. 빌드 프로젝트를 생성합니다. 자세한 내용은 에서 빌드 프로젝트 생성 AWS CodeBuild 단원을 참조하세요.
2. 테스트 보고서 정보를 사용하여 프로젝트의 buildspec 파일을 구성합니다.

   a. `reports:` 섹션을 추가하고 보고서 그룹의 이름을 지정합니다. CodeBuild는 프로젝트 이름과 `project-name-report-group-name-in-buildspec` 형식으로 지정한 이름을 사용하여 보고서 그룹을 생성합니다. 사용할 보고서 그룹이 이미 있는 경우 해당 ARN을 지정합니다. ARN 대신 해당 이름을 사용하는 경우 CodeBuild에서 새 보고서 그룹을 생성합니다. 자세한 내용은 Reports syntax in the buildspec file 단원을 참조하십시오.

   b. 보고서 그룹에서 코드 범위 결과를 포함하는 파일의 위치를 지정합니다. 둘 이상의 보고서 그룹을 사용하는 경우 각 보고서 그룹에 대해 결과 파일 위치를 지정합니다. 빌드 프로젝트가 실

행될 때마다 새 코드 범위 보고서가 생성됩니다. 자세한 내용은 테스트 파일 지정 단원을 참조하십시오.

다음은 test-results/jacoco-coverage-report.xml에 있는 JaCoCo XML 결과 파일에 대한 코드 범위 보고서를 생성하는 예제입니다.

```
reports:
  jacoco-report:
    files:
      - 'test-results/jacoco-coverage-report.xml'
    file-format: 'JACOCOXML'
```

c. build 또는 post_build 시퀀스의 commands 섹션에서 코드 범위 분석을 실행하는 명령을 지정합니다. 자세한 내용은 테스트 명령 지정 단원을 참조하십시오.

3. 빌드 프로젝트의 빌드를 실행합니다. 자세한 내용은 수동으로 AWS CodeBuild 빌드 실행 단원을 참조하십시오.

4. 빌드가 완료되면 프로젝트 페이지의 빌드 기록에서 새 빌드 실행을 선택하십시오. 보고서를 선택하여 코드 범위 보고서를 확인합니다. 자세한 내용은 빌드에 대한 테스트 보고서 보기 단원을 참조하십시오.

CodeBuild에서 보고서 자동 검색

자동 검색을 통해 CodeBuild는 빌드 단계가 완료된 후 모든 빌드 파일을 검색하고, 지원되는 보고서 파일 유형을 검색하고, 새 테스트 및 코드 적용 범위 보고서 그룹 및 보고서를 자동으로 생성합니다. 검색된 보고서 유형에 대해 CodeBuild는 다음 패턴으로 새 보고서 그룹을 생성합니다.

```
<project-name>-<report-file-format>-AutoDiscovered
```

> **Note**
>
> 검색된 보고서 파일의 형식 유형이 동일한 경우 동일한 보고서 그룹 또는 보고서에 배치됩니다.

보고서 자동 검색은 프로젝트 환경 변수에 의해 구성됩니다.

`CODEBUILD_CONFIG_AUTO_DISCOVER`

이 변수는 빌드 중에 보고서 자동 검색을 비활성화할지 여부를 결정합니다. 기본적으로 보고서 자동 검색은 모든 빌드에 대해 활성화됩니다. 이 기능을 비활성화하려면 `CODEBUILD_CONFIG_AUTO_DISCOVER`를 `false`로 설정합니다.

`CODEBUILD_CONFIG_AUTO_DISCOVER_DIR`

(선택 사항) 이 변수는 CodeBuild가 잠재적 보고서 파일을 검색할 위치를 결정합니다. CodeBuild는 기본적으로 `**/*`에서 검색합니다.

이러한 환경 변수는 빌드 단계에서 수정할 수 있습니다. 예를 들어 git 브랜치의 빌드에 대한 보고서 자동 검색만 활성화하려는 경우 빌드 프로세스 중에 `main` git 분기를 확인하고 빌드가 `main` 분기에 없는 경우 `CODEBUILD_CONFIG_AUTO_DISCOVER`를 false로 설정할 수 있습니다. 보고서 자동 검색은 콘솔 또는 프로젝트 환경 변수를 사용하여 비활성화할 수 있습니다.

주제

- 콘솔을 사용하여 보고서 자동 검색 구성
- 프로젝트 환경 변수를 사용하여 보고서 자동 검색 구성

콘솔을 사용하여 보고서 자동 검색 구성

다음 절차에 따라 콘솔을 사용하여 보고서 자동 검색을 구성합니다.

콘솔을 사용하여 보고서 자동 검색을 구성하려면

1. 빌드 프로젝트를 생성하거나 편집할 빌드 프로젝트를 선택합니다. 자세한 내용은 에서 빌드 프로젝트 생성 AWS CodeBuild 또는 에서 빌드 프로젝트 설정 변경 AWS CodeBuild 섹션을 참조하세요.
2. 환경에서 추가 구성을 선택합니다.
3. 보고서 자동 검색을 비활성화하려면 보고서 자동 검색에서 보고서 자동 검색 비활성화를 선택합니다.
4. (선택 사항) 디렉터리 자동 검색 - 선택 사항에서 CodeBuild가 지원되는 보고서 형식 파일을 검색할 디렉터리 패턴을 입력합니다. CodeBuild는 기본적으로 `**/*`에서 검색합니다.

프로젝트 환경 변수를 사용하여 보고서 자동 검색 구성

다음 절차에 따라 프로젝트 환경 변수를 사용하여 보고서 자동 검색을 구성합니다.

프로젝트 환경 변수를 사용하여 보고서 자동 검색을 구성하려면

1. 빌드 프로젝트를 생성하거나 편집할 빌드 프로젝트를 선택합니다. 자세한 내용은 에서 빌드 프로젝트 생성 AWS CodeBuild 또는 에서 빌드 프로젝트 설정 변경 AWS CodeBuild 섹션을 참조하세요.
2. 환경 변수에서 다음을 수행합니다.
   a. 보고서 자동 검색을 비활성화하려면 이름에 **CODEBUILD_CONFIG_AUTO_DISCOVER**를 입력하고 값에 **false**를 입력합니다. 이렇게 하면 보고서 자동 검색이 비활성화됩니다.
   b. (선택 사항) 이름에 **CODEBUILD_CONFIG_AUTO_DISCOVER_DIR**을 입력하고 값에 CodeBuild가 지원되는 보고서 형식 파일을 검색해야 하는 디렉터리를 입력합니다. 예를 들어 `output/*xml`은 `output` 디렉터리에서 `.xml` 파일을 검색합니다.

보고서 그룹

보고서 그룹은 그 안에 테스트 보고서가 포함되어 있으며 공유 설정을 지정합니다. buildspec 파일을 사용하여 실행할 테스트 케이스와 빌드할 때 실행할 명령을 지정합니다. 빌드 프로젝트에 구성된 각 보고서 그룹에 대해 빌드 프로젝트를 실행하면 테스트 보고서가 만들어집니다. 보고서 그룹으로 구성된 빌드 프로젝트를 여러 번 실행하면 해당 보고서 그룹에 여러 테스트 보고서가 작성되며, 각각 해당 보고서 그룹에 대해 지정된 동일한 테스트 케이스의 결과가 표시됩니다.

테스트 케이스는 빌드 프로젝트의 buildspec 파일에 있는 보고서 그룹에 대해 지정됩니다. 하나의 빌드 프로젝트에서 최대 5개의 보고서 그룹을 지정할 수 있습니다. 빌드를 실행하면 모든 테스트 케이스가 실행됩니다. 보고서 그룹에 지정된 각 테스트 케이스의 결과와 함께 새 테스트 보고서가 작성됩니다. 새 빌드를 실행할 때마다 테스트 케이스가 실행되고 새 테스트 보고서가 새 테스트 결과와 함께 만들어집니다.

보고서 그룹은 둘 이상의 빌드 프로젝트에서 사용할 수 있습니다. 하나의 보고서 그룹으로 작성된 모든 테스트 보고서는 테스트 보고서가 다른 빌드 프로젝트를 사용하여 작성된 경우에도 내보내기 옵션 및 사용 권한과 같은 동일한 구성을 공유합니다. 여러 빌드 프로젝트에서 하나의 보고서 그룹으로 만든 테스트 보고서에는 서로 다른 테스트 케이스 세트(각 빌드 프로젝트마다 테스트 케이스 세트 하나씩)를 실행한 결과가 포함될 수 있습니다. 이는 각 프로젝트의 buildspec 파일에서 보고서 그룹에 대해 서로 다른 테스트 케이스 파일을 지정할 수 있기 때문입니다. 그 buildspec 파일을 편집하여 빌드 프로젝트

의 보고서 그룹에 대한 테스트 케이스 파일을 변경할 수도 있습니다. 후속 빌드를 실행하면 업데이트된 buildspec에서 테스트 케이스 파일의 결과를 포함하는 새 테스트 보고서가 생성됩니다.

주제

- 보고서 그룹 만들기
- 보고서 그룹 이름 지정
- 보고서 그룹 공유
- 테스트 파일 지정
- 테스트 명령 지정
- 에서 보고서 그룹에 태그 지정 AWS CodeBuild
- 보고서 그룹 업데이트

보고서 그룹 만들기

CodeBuild 콘솔 AWS CLI, 또는 buildspec 파일을 사용하여 보고서 그룹을 생성할 수 있습니다. IAM 역할에는 보고서 그룹을 생성하는 데 필요한 권한이 있어야 합니다. 자세한 내용은 테스트 보고서 권한 단원을 참조하십시오.

주제

- 보고서 그룹 생성(buildspec)
- 보고서 그룹 만들기(콘솔)
- 보고서 그룹 생성(CLI)
- 보고서 그룹 생성(CloudFormation)

보고서 그룹 생성(buildspec)

buildspec을 사용하여 만든 보고서 그룹은 원시 테스트 결과 파일을 내보내지 않습니다. 보고서 그룹을 보고 내보내기 설정을 지정할 수 있습니다. 자세한 내용은 보고서 그룹 업데이트 단원을 참조하십시오.

buildspec 파일을 사용하여 보고서 그룹을 생성하려면

1. AWS 계정의 보고서 그룹과 연결되지 않은 보고서 그룹 이름을 선택합니다.
2. 이 이름으로 buildspec 파일의 `reports` 섹션을 구성합니다. 이 예에서는 보고서 그룹 이름이 `new-report-group`이고, 사용 테스트 케이스는 JUnit 프레임워크로 생성됩니다.

```
reports:
 new-report-group: #surefire junit reports
   files:
     - '**/*'
   base-directory: 'surefire/target/surefire-reports'
```

Buildspec의 환경 변수를 사용하여 보고서 그룹 이름을 지정할 수도 있습니다.

```
version: 0.2
env:
  variables:
    REPORT_GROUP_NAME: "new-report-group"
phases:
  build:
    commands:
      - ...
...
reports:
 $REPORT_GROUP_NAME:
   files:
     - '**/*'
   base-directory: 'surefire/target/surefire-reports'
```

자세한 내용은 테스트 파일 지정 및 Reports syntax in the buildspec file 섹션을 참조하세요.

3. commands 섹션에서, 테스트를 실행할 명령을 지정합니다. 자세한 내용은 테스트 명령 지정 단원을 참조하십시오.
4. 빌드를 실행합니다. 빌드가 완료되면 `project-name-report-group-name` 형식을 사용하는 이름으로 새 보고서 그룹이 만들어집니다. 자세한 내용은 보고서 그룹 이름 지정 단원을 참조하십시오.

보고서 그룹 만들기(콘솔)

다음 절차에서는 AWS Management Console을 사용하여 보고서 그룹을 생성합니다.

보고서 그룹을 생성하려면

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 Report groups(보고서 그룹)을 선택합니다.

3. Create report(보고서 그룹 생성)를 선택합니다.
4. 보고서 그룹 이름은 보고서 그룹의 이름을 입력합니다.
5. (선택 사항) 태그에 지원 AWS 서비스에서 사용할 태그의 이름과 값을 입력합니다. [Add row]를 사용하여 태그를 추가합니다. 최대 50개의 태그를 추가할 수 있습니다.
6. 테스트 보고서 결과의 원시 데이터를 Amazon S3 버킷에 업로드하려면

   a. Amazon S3로 내보내기를 선택합니다.

   b. S3 버킷 이름은 S3 버킷의 이름을 입력합니다.

   c. (선택 사항) S3 버킷 소유자의 경우 S3 버킷을 소유한 계정의 AWS 계정 식별자를 입력합니다. 이 속성을 사용하여 빌드를 실행하는 계정이 아닌 다른 계정이 소유한 Amazon S3 버킷으로 보고서 데이터를 내보낼 수 있습니다.

   d. 경로 접두사는 테스트 결과를 업로드할 S3 버킷의 경로를 입력합니다.

   e. 원시 테스트 결과 데이터 파일을 압축하려면 Compress test result data in a zip file(테스트 결과 데이터를 zip 파일로 압축)을 선택합니다.

   f. 추가 구성을 확장하여 암호화 옵션을 표시합니다. 다음 중 하나를 선택합니다.

      - Amazon S3용를 사용하기 AWS 관리형 키 위한 기본 AWS 관리형 키입니다. 자세한 내용은AWS Key Management Service 사용 설명서의 고객 관리형 CMK를 참조하세요. 이것은 기본 암호화 옵션입니다.
      - 생성하여 구성하는 고객 관리형 키를 사용할 사용자 지정 키를 선택합니다. AWS KMS 암호화 키는 암호화 키의 ARN을 입력합니다. 형식은 `arn:aws:kms:<region-id>:<aws-account-id>:key/<key-id>` 입니다. 자세한 내용을 알아보려면AWS Key Management Service 사용 설명서의 KMS 키 생성을 참조하세요.
      - 아티팩트 암호화를 비활성화하여 암호화를 비활성화합니다. 테스트 결과를 공유하거나 정적 웹사이트에 게시할 경우에 이를 선택할 수 있습니다. (동적 웹사이트에서 테스트 결과를 해독하는 코드를 실행할 수 있습니다.)

      유휴 데이터 암호화에 대한 자세한 내용은 데이터 암호화 단원을 참조하십시오.

   > **Note**
   >
   > 프로젝트에 지정된 CodeBuild 서비스 역할은 S3 버킷에 업로드하는 권한에 사용됩니다.

7. Create report(보고서 그룹 생성)를 선택합니다.

보고서 그룹 생성(CLI)

다음 절차에서는 AWS CLI을 사용하여 보고서 그룹을 생성합니다.

보고서 그룹을 생성하려면

1. `CreateReportGroup.json`이라는 이름의 파일을 만듭니다.
2. 요구 사항에 따라 다음 JSON 코드 조각 중 하나를 `CreateReportGroup.json`에 복사합니다.

   - 다음 JSON을 사용하여 테스트 보고서 그룹이 원시 테스트 결과 파일을 Amazon S3 버킷으로 내보내도록 지정합니다.

```
{
  "name": "<report-name>",
  "type": "TEST",
  "exportConfig": {
    "exportConfigType": "S3",
    "s3Destination": {
      "bucket": "<bucket-name>",
      "bucketOwner": "<bucket-owner>",
      "path": "<path>",
      "packaging": "NONE | ZIP",
      "encryptionDisabled": "false",
      "encryptionKey": "<your-key>"
    },
    "tags": [
      {
        "key": "tag-key",
        "value": "tag-value"
      }
    ]
  }
}
```

   - *<bucket-name>*은 Amazon S3 버킷 이름으로 바꾸고, *<path>*는 파일을 내보낼 버킷의 경로로 바꿉니다.
   - 내보낸 파일을 압축하려면 `packaging`을 `ZIP`로 지정합니다. 아닌 경우에는 `NONE`로 지정합니다.
   - `bucketOwner`는 선택 사항으로 빌드를 실행하는 계정이 아닌 다른 계정이 Amazon S3 버킷을 소유한 경우에만 필요합니다.

- 내보낸 파일을 암호화할지 여부를 지정할 때 encryptionDisabled을 사용합니다. 내보낸 파일을 암호화할 경우에는 고객 관리형 키를 입력합니다. 자세한 내용은 보고서 그룹 업데이트 단원을 참조하십시오.
- 다음 JSON을 사용하여 테스트 보고서가 원시 테스트 파일을 내보내지 않도록 지정합니다.

```
{
  "name": "<report-name>",
  "type": "TEST",
  "exportConfig": {
    "exportConfigType": "NO_EXPORT"
  }
}
```

> **Note**
>
> 프로젝트에 지정된 CodeBuild 서비스 역할은 S3 버킷에 업로드하는 권한에 사용됩니다.

3. 다음 명령을 실행합니다.

```
aws codebuild create-report-group --cli-input-json file://
CreateReportGroupInput.json
```

보고서 그룹 생성(CloudFormation)

다음 지침에 따라 CloudFormation 템플릿을 사용하여 보고서 그룹을 생성합니다.

CloudFormation 템플릿을 사용하여 보고서 그룹을 생성하려면

CloudFormation 템플릿 파일을 사용하여 보고서 그룹을 생성하고 프로비저닝할 수 있습니다. 자세한 내용은 CloudFormation 사용자 안내서를 참조하세요.

다음 CloudFormation YAML 템플릿은 원시 테스트 결과 파일을 내보내지 않는 보고서 그룹을 생성합니다.

```
Resources:
  CodeBuildReportGroup:
    Type: AWS::CodeBuild::ReportGroup
    Properties:
      Name: my-report-group-name
```

```
      Type: TEST
      ExportConfig:
        ExportConfigType: NO_EXPORT
```

다음 CloudFormation YAML 템플릿은 원시 테스트 결과 파일을 Amazon S3 버킷으로 내보내는 보고서 그룹을 생성합니다.

```
Resources:
  CodeBuildReportGroup:
    Type: AWS::CodeBuild::ReportGroup
    Properties:
      Name: my-report-group-name
      Type: TEST
      ExportConfig:
        ExportConfigType: S3
        S3Destination:
          Bucket: amzn-s3-demo-bucket
          Path: path-to-folder-for-exported-files
          Packaging: ZIP
          EncryptionKey: my-KMS-encryption-key
          EncryptionDisabled: false
```

> **Note**
>
> 프로젝트에 지정된 CodeBuild 서비스 역할은 S3 버킷에 업로드하는 권한에 사용됩니다.

보고서 그룹 이름 지정

AWS CLI 또는 AWS CodeBuild 콘솔을 사용하여 보고서 그룹을 생성할 때 보고서 그룹의 이름을 지정합니다. buildspec을 사용하여 새 보고서 그룹을 생성하는 경우 *project-name*-*report-group-name-specified-in-buildspec* 형식을 사용하여 이름이 지정됩니다. 해당 빌드 프로젝트의 빌드를 실행하여 만든 모든 보고서는 새 이름을 가진 새 보고서 그룹에 속합니다.

CodeBuild가 새 보고서 그룹을 생성하지 않게 하려면 빌드 프로젝트의 buildspec 파일에 보고서 그룹의 ARN을 지정합니다. 여러 빌드 프로젝트에서 보고서 그룹의 ARN을 지정할 수 있습니다. 각 빌드 프로젝트가 실행되면 보고서 그룹에는 각 빌드 프로젝트에서 만든 테스트 보고서가 포함됩니다.

예를 들어, 이름 `my-report-group`로 하나의 보고서 그룹을 만든 다음 이름이 `my-project-1`와 `my-project-2`인 서로 다른 두 개의 빌드 프로젝트에서 해당 이름을 사용하고 두 프로젝트의 빌드를

작성하는 경우 두 개의 새 보고서 그룹이 만들어집니다. 그 결과 다음과 같은 이름의 세 개의 보고서 그룹이 만들어집니다.

- `my-report-group`: 테스트 보고서가 없습니다.
- `my-project-1-my-report-group`: 이름이 `my-project-1`인 빌드 프로젝트에서 실행한 테스트 결과가 있는 보고서가 포함되어 있습니다.
- `my-project-2-my-report-group`: 이름이 `my-project-2`인 빌드 프로젝트에서 실행한 테스트 결과가 있는 보고서가 포함되어 있습니다.

두 프로젝트에서 이름이 `my-report-group`로 지정된 보고서 그룹의 ARN을 사용한 후 각 프로젝트의 빌드를 실행해도 보고서 그룹(`my-report-group`)은 한 개입니다. 이 보고서 그룹에는 두 빌드 프로젝트에서 실행한 테스트 결과가 있는 테스트 보고서가 포함되어 있습니다.

AWS 계정의 보고서 그룹에 속하지 않는 보고서 그룹 이름을 선택한 다음 buildspec 파일의 보고서 그룹에 해당 이름을 사용하고 빌드 프로젝트의 빌드를 실행하면 새 보고서 그룹이 만들어집니다. 새 보고서 그룹의 이름 형식은 *`project-name`*`-`*`new-group-name`*입니다. 예를 들어 AWS 계정에 이름이 인 보고서 그룹이 없고 라는 빌드 프로젝트에서 `new-report-group`지정한 경우 빌드 실행`test-project`은 이름이 인 새 보고서 그룹을 생성합니다`test-project-new-report-group`.

보고서 그룹 공유

보고서 그룹 공유를 사용하면 여러 AWS 계정 또는 사용자가 보고서 그룹, 만료되지 않은 보고서 및 보고서의 테스트 결과를 볼 수 있습니다. 이 모델에서는 보고서 그룹을 소유하는 계정(소유자)은 다른 계정(소비자)과 보고서 그룹을 공유합니다. 소비자는 보고서 그룹을 편집할 수 없습니다. 보고서는 생성되고 30일 후에 만료됩니다.

주제

- 보고서 그룹 공유
- 관련 서비스
- 사용자와 공유된 보고서 그룹에 액세스
- 공유 보고서 그룹 공유 해제
- 공유 보고서 그룹 식별
- 공유 보고서 그룹 권한

보고서 그룹 공유

보고서 그룹을 공유하면 소비자에게 보고서 그룹 및 해당 보고서에 대한 읽기 전용 액세스 권한이 부여됩니다. 소비자는 AWS CLI 를 사용하여 보고서 그룹, 보고서 및 각 보고서의 테스트 사례 결과를 볼 수 있습니다. 소비자는 다음을 수행할 수 없습니다.

- CodeBuild 콘솔에서 공유 보고서 그룹 또는 해당 보고서 보기.
- 공유 보고서 그룹 편집하기.
- 프로젝트에서 공유 보고서 그룹의 ARN을 사용하여 보고서 실행하기. 공유 보고서 그룹을 지정하는 프로젝트 빌드가 실패합니다.

CodeBuild 콘솔을 사용하여 기존 리소스 공유에 보고서 그룹을 추가할 수 있습니다. 보고서 그룹을 새 리소스 공유에 추가하려면 먼저 AWS RAM 콘솔에서 보고서 그룹을 만들어야 합니다.

보고서 그룹을 조직 단위 또는 전체 조직과 공유하려면 AWS Organizations와의 공유를 활성화해야 합니다. 자세한 내용은 AWS RAM 사용 설명서에서 AWS Organizations를 사용하여 공유 사용을 참조하세요.

CodeBuild 콘솔, AWS RAM 콘솔 또는 AWS CLI 를 사용하여 소유한 보고서 그룹을 공유할 수 있습니다.

사전 조건

보고서 그룹을 공유하려면 AWS 계정이 보고서 그룹을 소유해야 합니다. 사용자와 공유된 보고서 그룹은 공유할 수 없습니다.

소유한 보고서 그룹을 공유하려면(CodeBuild 콘솔)

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 Report groups(보고서 그룹)을 선택합니다.
3. 공유할 프로젝트를 선택한 다음 공유를 선택합니다. 자세한 내용은AWS RAM 사용 설명서의 리소스 공유 생성을 참조하세요.

소유한 보고서 그룹을 공유하려면(AWS RAM 콘솔)

AWS RAM 사용 설명서에서 리소스 공유 생성을 참조하세요.

소유한 보고서 그룹을 공유하려면(AWS RAM 명령)

create-resource-share 명령을 사용합니다.

소유한 보고서 그룹을 공유하려면(CodeBuild 명령)

put-resource-policy 명령을 사용합니다.

1. 이름이 `policy.json`인 파일을 만들고 다음으로 복사합니다.

   JSON

```
{
   "Version":"2012-10-17",
   "Statement":[{
     "Effect":"Allow",
     "Principal":{
       "AWS":"111122223333"
     },
     "Action":[
       "codebuild:BatchGetReportGroups",
       "codebuild:BatchGetReports",
       "codebuild:ListReportsForReportGroup",
       "codebuild:DescribeTestCases"],
     "Resource":"arn:aws:iam::*:role/Service*"
   }]
 }
```

2. 보고서 그룹 ARN 및 식별자로 `policy.json`을 업데이트하여 공유합니다. 다음 예제에서는 ARN이 있는 보고서 그룹에 대한 읽기 전용 액세스 권한을 `arn:aws:codebuild:us-west-2:123456789012:report-group/my-report-group` Alice와 123456789012로 식별되는 AWS 계정의 루트 사용자에게 부여합니다.

   JSON

```
{
   "Version":"2012-10-17",
   "Statement":[{
     "Effect":"Allow",
     "Principal":{
       "AWS": [
          "arn:aws:iam::123456789012:user/Alice",
          "123456789012"
```

```
        ]
      },
      "Action":[
        "codebuild:BatchGetReportGroups",
        "codebuild:BatchGetReports",
        "codebuild:ListReportsForReportGroup",
        "codebuild:DescribeTestCases"],
      "Resource":"arn:aws:codebuild:us-west-2:123456789012:report-group/my-
report-group"
    }]
 }
```

3. 다음 명령을 실행합니다.

```
aws codebuild put-resource-policy --resource-arn report-group-arn --policy file://
policy.json
```

관련 서비스

보고서 그룹 공유는 모든 AWS 계정 또는를 통해 AWS 리소스를 공유할 수 있는 서비스AWS RAM 인 AWS Resource Access Manager ()와 통합됩니다 AWS Organizations. 를 사용하면 AWS RAM 공유할 리소스와 소비자를 지정하는 리소스 공유를 생성하여 소유한 리소스를 공유할 수 있습니다. 소비자는 개별 AWS 계정,의 조직 단위 AWS Organizations또는의 전체 조직일 수 있습니다 AWS Organizations.

자세한 내용은 AWS RAM 사용 설명서를 참조하십시오.

사용자와 공유된 보고서 그룹에 액세스

공유 보고서 그룹에 액세스하려면 소비자의 IAM 역할에 `BatchGetReportGroups` 권한이 필요합니다. 다음 정책을 해당 IAM 역할에 연결할 수 있습니다.

```
{
    "Effect": "Allow",
    "Resource": [
        "*"
    ],
    "Action": [
        "codebuild:BatchGetReportGroups"
    ]
}
```

자세한 내용은 에 자격 증명 기반 정책 사용 AWS CodeBuild 단원을 참조하십시오.

공유 보고서 그룹 공유 해제

보고서 및 테스트 케이스 결과를 포함해 공유되지 않은 보고서 그룹은 소유자만 액세스할 수 있습니다. 보고서 그룹을 공유 해제하면 이전에 공유한 AWS 계정 또는 사용자가 보고서 그룹, 보고서 또는 보고서의 테스트 사례 결과에 액세스할 수 없습니다.

소유하고 있는 공유된 보고서 그룹의 공유를 해제하려면 리소스 공유에서 제거해야 합니다. AWS RAM 콘솔 또는를 사용하여이 작업을 수행할 수 AWS CLI 있습니다.

소유한 공유 보고서 그룹을 공유 해제하려면(AWS RAM 콘솔)

AWS RAM 사용 설명서에서 리소스 공유 업데이트를 참조하세요.

소유한 공유 보고서 그룹을 공유 해제하려면(AWS RAM 명령)

disassociate-resource-share 명령을 사용합니다.

소유한 보고서 그룹을 공유 해제하려면(CodeBuild 명령)

delete-resource-policy 명령을 실행하고 공유를 해제할 보고서 그룹의 ARN을 지정합니다.

```
aws codebuild delete-resource-policy --resource-arn report-group-arn
```

공유 보고서 그룹 식별

소유자와 소비자는 AWS CLI 를 사용하여 공유 보고서 그룹을 식별할 수 있습니다.

공유 보고서 그룹 및 해당 보고서에 대한 정보를 식별하고 가져오려면 다음 명령을 사용합니다.

- 공유된 보고서 그룹의 ARN을 보려면 list-shared-report-groups을 실행합니다.

  ```
  aws codebuild list-shared-report-groups
  ```

- 보고서 그룹에서 보고서의 ARN을 보려면 보고서 그룹 ARN을 사용하여 list-reports-for-report-group을 실행합니다.

  ```
  aws codebuild list-reports-for-report-group --report-group-arn report-group-arn
  ```

- 보고서의 테스트 케이스에 대한 정보를 보려면 보고서 ARN을 사용하여 describe-test-cases을 실행합니다.

```
aws codebuild describe-test-cases --report-arn report-arn
```

출력은 다음과 같습니다.

```
{
    "testCases": [
        {
            "status": "FAILED",
            "name": "Test case 1",
            "expired": 1575916770.0,
            "reportArn": "report-arn",
            "prefix": "Cucumber tests for agent",
            "message": "A test message",
            "durationInNanoSeconds": 1540540,
            "testRawDataPath": "path-to-output-report-files"
        },
        {
            "status": "SUCCEEDED",
            "name": "Test case 2",
            "expired": 1575916770.0,
            "reportArn": "report-arn",
            "prefix": "Cucumber tests for agent",
            "message": "A test message",
            "durationInNanoSeconds": 1540540,
            "testRawDataPath": "path-to-output-report-files"
        }
    ]
}
```

공유 보고서 그룹 권한

소유자에 대한 권한

보고서 그룹 소유자는 보고서 그룹을 편집하고 프로젝트에서 보고서를 실행할 수 있도록 지정할 수 있습니다.

소비자에 대한 권한

보고서 그룹 소비자는 보고서 그룹, 보고서 및 보고서에 대한 테스트 케이스 결과를 볼 수 있습니다. 소비자는 보고서 그룹이나 해당 보고서를 편집할 수 없으며 이를 사용하여 보고서를 만들 수 없습니다.

테스트 파일 지정

빌드 프로젝트의 buildspec 파일에서 `reports` 섹션에 있는 각 보고서 그룹에 대한 테스트 결과 파일 및 해당 위치를 지정합니다. 자세한 내용은 Reports syntax in the buildspec file 단원을 참조하십시오.

다음은 빌드 프로젝트에 대해 두 개의 보고서 그룹을 지정하는 샘플 `reports` 섹션입니다. 하나는 ARN으로 지정되고 다른 하나는 이름으로 지정됩니다. 이 `files` 섹션에서는 테스트 케이스 결과를 포함하는 파일을 지정합니다. 선택 사항인 `base-directory` 섹션에서는 테스트 케이스 파일이 있는 디렉터리를 지정합니다. 선택 사항인 `discard-paths` 섹션에서는 Amazon S3 버킷에 업로드된 테스트 결과 파일의 경로를 무시할지 여부를 지정합니다.

```
reports:
  arn:aws:codebuild:your-region:your-aws-account-id:report-group/report-group-name-1:
 #surefire junit reports
    files:
      - '**/*'
    base-directory: 'surefire/target/surefire-reports'
    discard-paths: false

  sampleReportGroup: #Cucumber reports from json plugin
    files:
      - 'cucumber-json/target/cucumber-json-report.json'
    file-format: CUCUMBERJSON #Type of the report, defaults to JUNITXML
```

테스트 명령 지정

buildspec 파일의 `commands` 섹션에서 테스트 케이스를 실행하는 명령을 지정합니다. 이 명령은 buildspec 파일의 `reports` 섹션에서 보고서 그룹에 대해 지정된 테스트 케이스를 실행합니다. 다음은 테스트 파일에서 테스트를 실행하는 명령이 포함된 샘플 `commands` 섹션입니다.

```
commands:
    - echo Running tests for surefire junit
    - mvn test -f surefire/pom.xml -fn
    - echo
    - echo Running tests for cucumber with json plugin
    - mvn test -Dcucumber.options="--plugin json:target/cucumber-json-report.json" -f
 cucumber-json/pom.xml -fn
```

자세한 내용은 buildspec 구문 단원을 참조하십시오.

에서 보고서 그룹에 태그 지정 AWS CodeBuild

태그는 사용자 또는가 AWS 리소스에 AWS 할당하는 사용자 지정 속성 레이블입니다. 각 AWS 태그에는 두 부분이 있습니다.

- 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. 보고서 그룹에 포함할 수 있는 태그 수 제한 및 태그 키 및 값에 대한 제한은 Tags 단원을 참조하십시오.

태그는 AWS 리소스를 식별하고 구성하는 데 도움이 됩니다. 많은 AWS 서비스가 태그 지정을 지원하므로 다른 서비스의 리소스에 동일한 태그를 할당하여 리소스가 관련이 있음을 나타낼 수 있습니다. 예를 들어 Amazon S3 버킷에 할당한 것과 동일한 태그를 CodeBuild 보고서 그룹에 할당할 수 있습니다. 태그 사용에 대한 자세한 내용은 태그 지정 모범 사례 백서를 참조하십시오.

CodeBuild에서 기본 리소스는 보고서 그룹 및 프로젝트입니다. CodeBuild 콘솔, AWS CLI, CodeBuild APIs 또는 AWS SDKs 사용하여 보고서 그룹의 태그를 추가, 관리 및 제거할 수 있습니다. 태그로 보고서 그룹을 식별, 구성 및 추적하는 것 외에도 IAM 정책의 태그를 사용하여 보고서 그룹을 보고 상호 작용할 수 있는 사용자를 제어할 수 있습니다. 태그 기반 액세스 정책의 예는 태그를 사용하여 AWS CodeBuild 리소스에 대한 액세스 제어 단원을 참조하세요.

주제

- 보고서 그룹에 태그 추가
- 보고서 그룹의 태그 보기
- 보고서 그룹의 태그 편집
- 보고서 그룹에서 태그 제거

보고서 그룹에 태그 추가

보고서 그룹에 태그를 추가하면 AWS 리소스를 식별 및 구성하고 리소스에 대한 액세스를 관리하는 데 도움이 될 수 있습니다. 먼저 보고서 그룹에 하나 이상의 태그(키-값 페어)를 추가합니다. 보고서 그룹에 태그 수에 대한 제한이 있음을 알아두십시오. 키 및 값 필드에서 사용할 수 있는 문자에 대한 제한이 있습니다. 자세한 내용은 Tags 단원을 참조하십시오. 태그가 생성된 후 해당 태그를 기준으로 보고서

그룹에 대한 액세스를 관리하는 IAM 정책을 생성할 수 있습니다. CodeBuild 콘솔 또는를 사용하여 보고서 그룹에 태그를 AWS CLI 추가할 수 있습니다.

> **Important**
>
> 보고서 그룹에 태그를 추가하면 해당 보고서 그룹에 대한 액세스에 영향을 줄 수 있습니다. 보고서 그룹에 태그를 추가하기 전에 보고서 그룹과 같은 리소스에 대한 액세스를 제어하는 태그를 사용할 수도 있는 모든 IAM 정책을 검토하세요. 태그 기반 액세스 정책의 예는 태그를 사용하여 AWS CodeBuild 리소스에 대한 액세스 제어 단원을 참조하세요.

보고서 그룹을 생성할 때 보고서 그룹에 태그를 추가하는 방법에 대한 자세한 내용은 보고서 그룹 만들기(콘솔) 단원을 참조하십시오.

주제

- 보고서 그룹에 태그 추가(콘솔)
- 보고서 그룹에 태그 추가(AWS CLI)

보고서 그룹에 태그 추가(콘솔)

CodeBuild 콘솔을 사용하여 CodeBuild 보고서 그룹에 하나 이상의 태그를 추가할 수 있습니다.

1. https://console.aws.amazon.com/codebuild/에서 CodeBuild 콘솔을 엽니다.
2. 보고서 그룹에서 태그를 추가할 보고서 그룹의 이름을 선택합니다.
3. 탐색 창에서 설정을 선택합니다.
4. 보고서 그룹에 추가된 태그가 없는 경우 태그 추가를 선택합니다. 편집을 선택한 다음 태그 추가를 선택할 수도 있습니다.
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.

보고서 그룹에 태그 추가(AWS CLI)

보고서 그룹을 생성할 때 보고서 그룹에 태그를 추가하려면 보고서 그룹 생성(CLI) 단원을 참조하십시오. `CreateReportGroup.json`에서 태그를 추가합니다.

기존 보고서 그룹에 태그를 추가하려면 보고서 그룹 업데이트(CLI) 단원을 참조하고 UpdateReportGroupInput.json에서 태그를 추가하십시오.

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

보고서 그룹의 태그 보기

태그는 AWS 리소스를 식별 및 구성하고 리소스에 대한 액세스를 관리하는 데 도움이 될 수 있습니다. 태그 사용에 대한 자세한 내용은 태그 지정 모범 사례 백서를 참조하십시오. 태그 기반 액세스 정책의 예는 Deny or allow actions on report groups based on resource tags 단원을 참조하세요.

보고서 그룹의 태그 보기(콘솔)

CodeBuild 콘솔을 사용하여 CodeBuild 보고서 그룹과 연결된 태그를 볼 수 있습니다.

1. https://console.aws.amazon.com/codebuild/에서 CodeBuild 콘솔을 엽니다.
2. 보고서 그룹에서 태그를 보려는 보고서 그룹의 이름을 선택합니다.
3. 탐색 창에서 설정을 선택합니다.

보고서 그룹의 태그 보기(AWS CLI)

다음 단계에 따라 AWS CLI 를 사용하여 보고서 그룹의 AWS 태그를 봅니다. 태그가 추가되지 않은 경우 반환된 목록은 비어 있습니다.

1. 콘솔 또는 AWS CLI 를 사용하여 보고서 그룹의 ARN을 찾습니다. 해당 ARN을 기록해 둡니다.

   AWS CLI

   다음 명령을 실행합니다.

   ```
   aws list-report-groups
   ```

   이 명령은 다음과 유사한 JSON 형식의 정보를 반환합니다.

   ```
   {
       "reportGroups": [
           "arn:aws:codebuild:region:123456789012:report-group/report-group-1",
           "arn:aws:codebuild:region:123456789012:report-group/report-group-2",
           "arn:aws:codebuild:region:123456789012:report-group/report-group-3"
   ```

```
    ]
}
```

보고서 그룹 ARN은 보고서 그룹의 ARN을 식별하는 데 사용할 수 있는 이름으로 끝납니다.

Console

1. https://console.aws.amazon.com/codebuild/에서 CodeBuild 콘솔을 엽니다.
2. 보고서 그룹에서 보려는 태그가 있는 보고서 그룹의 이름을 선택합니다.
3. 구성에서 보고서 그룹의 ARN을 찾습니다.

2. 다음 명령을 실행합니다. `--report-group-arns` 파라미터에 대해 기록한 ARN을 사용합니다.

```
aws codebuild batch-get-report-groups --report-group-arns
 arn:aws:codebuild:region:123456789012:report-group/report-group-name
```

성공하면 이 명령은 다음과 유사한 `tags` 섹션이 포함된 JSON 형식의 정보를 반환합니다.

```
{
    ...
    "tags": {
        "Status": "Secret",
        "Project": "TestBuild"
    }
    ...
}
```

보고서 그룹의 태그 편집

보고서 그룹과 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 알아두십시오. 자세한 내용은 Tags 단원을 참조하십시오.

> **Important**
>
> 보고서 그룹의 태그를 편집하면 해당 보고서 그룹에 대한 액세스에 영향을 줄 수 있습니다. 보고서 그룹에 대한 태그의 이름(키) 또는 값을 편집하기 전에 보고서 그룹과 같은 리소스에 대한 액세스를 제어하는 태그의 키 또는 값을 사용할 수도 있는 모든 IAM 정책을 검토하세요. 태그

기반 액세스 정책의 예는 Deny or allow actions on report groups based on resource tags 단원을 참조하세요.

보고서 그룹의 태그 편집(콘솔)

CodeBuild 콘솔을 사용하여 CodeBuild 보고서 그룹과 연결된 태그를 편집할 수 있습니다.

1. https://console.aws.amazon.com/codebuild/에서 CodeBuild 콘솔을 엽니다.
2. 보고서 그룹에서 태그를 편집할 보고서 그룹의 이름을 선택합니다.
3. 탐색 창에서 설정을 선택합니다.
4. 편집을 선택합니다.
5. 다음 중 하나를 수행하세요.
   - 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.

보고서 그룹의 태그 편집(AWS CLI)

보고서 그룹에서 태그를 추가, 변경 또는 삭제하려면 보고서 그룹 업데이트(CLI) 단원을 참조하십시오. `UpdateReportGroupInput.json`에서 태그를 업데이트합니다.

보고서 그룹에서 태그 제거

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

> **Important**
>
> 보고서 그룹의 태그를 제거하면 해당 보고서 그룹에 대한 액세스에 영향을 줄 수 있습니다. 보고서 그룹에서 태그를 제거하기 전에 보고서 그룹과 같은 리소스에 대한 액세스를 제어하는 태그의 키 또는 값을 사용할 수도 있는 모든 IAM 정책을 검토하세요. 태그 기반 액세스 정책의 예는 태그를 사용하여 AWS CodeBuild 리소스에 대한 액세스 제어 단원을 참조하세요.

보고서 그룹에서 태그 제거(콘솔)

CodeBuild 콘솔을 사용하면 태그와 CodeBuild 보고서 그룹 간의 연결을 제거할 수 있습니다.

1. https://console.aws.amazon.com/codebuild/에서 CodeBuild 콘솔을 엽니다.
2. 보고서 그룹에서 태그를 제거할 보고서 그룹의 이름을 선택합니다.
3. 탐색 창에서 설정을 선택합니다.
4. 편집을 선택합니다.
5. 제거할 태그를 찾은 다음 태그 제거를 선택합니다.
6. 태그 제거를 마쳤으면 제출을 선택합니다.

보고서 그룹에서 태그 제거(AWS CLI)

다음 단계에 따라 AWS CLI 를 사용하여 CodeBuild 보고서 그룹에서 태그를 제거합니다. 태그를 제거하면 태그는 삭제되지 않고 태그와 보고서 그룹 간의 연결만 제거됩니다.

> **Note**
>
> CodeBuild 보고서 그룹을 삭제하면 삭제된 보고서 그룹에서 모든 태그 연결이 제거됩니다. 보고서 그룹을 삭제하기 전에 태그를 제거할 필요가 없습니다.

보고서 그룹에서 하나 이상의 태그를 삭제하려면 보고서 그룹의 태그 편집(AWS CLI) 단원을 참조하십시오. 삭제하려는 태그가 포함되지 않은 업데이트된 태그 목록으로 JSON 형식 데이터의 `tags` 섹션을 업데이트합니다. 모든 태그를 삭제하려면 `tags` 섹션을 다음과 같이 업데이트하십시오.

```
"tags: []"
```

보고서 그룹 업데이트

보고서 그룹을 업데이트할 때 원시 테스트 결과 데이터를 Amazon S3 버킷의 파일로 내보낼지 여부에 대한 정보를 지정할 수 있습니다. S3 버킷으로 내보내도록 선택하는 경우에는 보고서 그룹에 대해 다음을 지정할 수 있습니다.

- 원시 테스트 결과 파일이 ZIP 파일로 압축되는지 여부.
- 원시 테스트 결과 파일이 암호화되는지 여부. 다음 중 하나를 사용하여 암호화를 지정할 수 있습니다.

- Amazon S3 AWS 관리형 키 용 .
- 직접 생성하고 구성한 고객 관리형 키.

자세한 내용은 데이터 암호화 단원을 참조하십시오.

AWS CLI 를 사용하여 보고서 그룹을 업데이트하는 경우 태그를 업데이트하거나 추가할 수도 있습니다. 자세한 내용은 에서 보고서 그룹에 태그 지정 AWS CodeBuild 단원을 참조하십시오.

> **Note**
>
> 프로젝트에 지정된 CodeBuild 서비스 역할은 S3 버킷에 업로드하는 권한에 사용됩니다.

주제

- 보고서 그룹 업데이트(콘솔)
- 보고서 그룹 업데이트(CLI)

보고서 그룹 업데이트(콘솔)

다음 절차에서는 AWS Management Console를 사용하여 보고서 그룹을 업데이트합니다.

보고서 그룹을 업데이트하려면

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 Report groups(보고서 그룹)을 선택합니다.
3. 업데이트할 보고서 그룹을 선택합니다.
4. 편집을 선택합니다.
5. Amazon S3로 백업을 선택하거나 선택을 취소합니다. 이 옵션을 선택한 경우 다음과 같은 내보내기 설정을 지정합니다.
   a. S3 버킷 이름은 S3 버킷의 이름을 입력합니다.
   b. 경로 접두사는 테스트 결과를 업로드할 S3 버킷의 경로를 입력합니다.
   c. 원시 테스트 결과 데이터 파일을 압축하려면 Compress test result data in a zip file(테스트 결과 데이터를 zip 파일로 압축)을 선택합니다.
   d. 추가 구성을 확장하여 암호화 옵션을 표시합니다. 다음 중 하나를 선택합니다.

- Amazon S3용를 사용하기 AWS 관리형 키 위한 기본 AWS 관리형 키입니다. 자세한 내용은AWS Key Management Service 사용 설명서의 고객 관리형 CMK를 참조하세요. 이것은 기본 암호화 옵션입니다.
- 생성하여 구성하는 고객 관리형 키를 사용할 사용자 지정 키를 선택합니다. AWS KMS 암호화 키는 암호화 키의 ARN을 입력합니다. 형식은 `arn:aws:kms:<region-id>:<aws-account-id>:key/<key-id>` 입니다. 자세한 내용을 알아보려면AWS Key Management Service 사용 설명서의 KMS 키 생성을 참조하세요.
- 아티팩트 암호화를 비활성화하여 암호화를 비활성화합니다. 테스트 결과를 공유하거나 정적 웹사이트에 게시할 경우에 이를 선택할 수 있습니다. (동적 웹사이트에서 테스트 결과를 해독하는 코드를 실행할 수 있습니다.)

보고서 그룹 업데이트(CLI)

다음 절차에서는 AWS CLI를 사용하여 보고서 그룹을 업데이트합니다.

보고서 그룹을 업데이트하려면

1. `UpdateReportGroupInput.json`이라는 이름의 파일을 만듭니다.
2. 다음을 `UpdateReportGroupInput.json`에 복사합니다.

```
{
    "arn": "",
    "exportConfig": {
        "exportConfigType": "S3",
        "s3Destination": {
            "bucket": "bucket-name",
            "path": "path",
            "packaging": "NONE | ZIP",
            "encryptionDisabled": "false",
            "encryptionKey": "your-key"
         }
     },
     "tags": [
        {
            "key": "tag-key",
            "value": "tag-value"
        }
     ]
```

```
}
```

3. arn 줄에 보고서 그룹의 ARN을 입력합니다(예: "arn":"arn:aws:codebuild:*region*:*123456789012*:report-group/*report-group-1*")).
4. 보고서 그룹에 적용할 업데이트 내용으로 UpdateReportGroupInput.json을 업데이트합니다.

   - 원시 테스트 결과 파일을 S3 버킷으로 내보내도록 보고서 그룹을 업데이트하려면 exportConfig 섹션을 업데이트합니다. bucket-name은 S3 버킷 이름으로 바꾸고, path는 파일을 내보낼 S3 버킷의 경로로 바꿉니다. 내보낸 파일을 압축하려면 packaging을 ZIP로 지정합니다. 아닌 경우에는 NONE로 지정합니다. 내보낸 파일을 암호화할지 여부를 지정할 때 encryptionDisabled을 사용합니다. 내보낸 파일을 암호화할 경우에는 고객 관리형 키를 입력합니다.
   - 보고서 그룹을 업데이트하여 원시 테스트 결과 파일을 S3 버킷으로 내보내지 않으려면 exportConfig 섹션을 다음 JSON으로 업데이트합니다.

     ```
     {
       "exportConfig": {
           "exportConfigType": "NO_EXPORT"
       }
     }
     ```

   - 보고서 그룹의 태그를 업데이트하려면 tags 섹션을 업데이트합니다. 태그를 변경, 추가 또는 제거할 수 있습니다. 모든 태그를 제거하려면 다음 JSON으로 업데이트하십시오.

     ```
     "tags": []
     ```

5. 다음 명령을 실행합니다.

   ```
   aws codebuild update-report-group \
   --cli-input-json file://UpdateReportGroupInput.json
   ```

테스트 프레임워크

이 섹션의 주제에서는 다양한 테스트 프레임워크에 AWS CodeBuild 대해에서 테스트 보고를 설정하는 방법을 보여줍니다.

주제

- Jasmine을 사용하여 테스트 보고 설정
- Jest를 사용하여 테스트 보고 설정
- pytest를 사용하여 테스트 보고 설정
- RSpec을 사용하여 테스트 보고 설정

Jasmine을 사용하여 테스트 보고 설정

다음 절차에서는 JasmineBDD 테스트 프레임워크를 AWS CodeBuild 사용하여에서 테스트 보고를 설정하는 방법을 보여줍니다. JasmineBDD

이 절차를 수행하려면 다음 전제 조건이 필요합니다.

- 기존 CodeBuild 프로젝트가 있어야 합니다.
- 프로젝트는 Jasmine 테스트 프레임워크를 사용하도록 설정된 Node.js 프로젝트입니다.

`jasmine-reporters` 패키지를 프로젝트의 `package.json` 파일의 `devDependencies` 섹션에 추가합니다. 이 패키지에는 Jasmine과 함께 사용할 수 있는 JavaScript 리포터 클래스의 컬렉션이 있습니다.

```
npm install --save-dev jasmine-reporters
```

아직 없으면 프로젝트의 `package.json` 파일에 `test` 스크립트를 추가합니다. `test` 스크립트는 npm test가 실행될 때 Jasmine이 호출되도록 합니다.

```
{
  "scripts": {
    "test": "npx jasmine"
  }
}
```

CodeBuild는 다음과 같은 Jasmine 테스트 리포터를 지원합니다.

JUnitXmlReporter

`JunitXml` 형식으로 보고서를 생성하는 데 사용됩니다.

NUnitXmlReporter

NunitXml 형식으로 보고서를 생성하는 데 사용됩니다.

Jasmine과 함께 사용할 수 있는 Node.js 프로젝트에는 기본적으로 Jasmine 구성 및 테스트 스크립트를 포함하는 spec 하위 디렉터리가 있습니다.

JunitXML 형식으로 보고서를 생성하는 Jasmine을 구성하려면 테스트에 다음 코드를 추가하여 JUnitXmlReporter 리포터를 인스턴스화합니다.

```
var reporters = require('jasmine-reporters');

var junitReporter = new reporters.JUnitXmlReporter({
  savePath: <test report directory>,
  filePrefix: <report filename>,
  consolidateAll: true
});

jasmine.getEnv().addReporter(junitReporter);
```

NunitXML 형식으로 보고서를 생성하는 Jasmine을 구성하려면 테스트에 다음 코드를 추가하여 NUnitXmlReporter 리포터를 인스턴스화합니다.

```
var reporters = require('jasmine-reporters');

var nunitReporter = new reporters.NUnitXmlReporter({
  savePath: <test report directory>,
  filePrefix: <report filename>,
  consolidateAll: true
});

jasmine.getEnv().addReporter(nunitReporter)
```

테스트 보고서는 *<test report directory>*/*<report filename>*으로 지정된 파일로 내보내집니다.

buildspec.yml 파일에서 다음 섹션을 추가/업데이트합니다.

```
version: 0.2
```

```
phases:
  pre_build:
    commands:
      - npm install
  build:
    commands:
      - npm build
      - npm test

reports:
  jasmine_reports:
    files:
      - <report filename>
    file-format: JUNITXML
    base-directory: <test report directory>
```

NunitXml 보고서 형식을 사용하는 경우 file-format 값을 다음과 같이 변경합니다.

```
    file-format: NUNITXML
```

Jest를 사용하여 테스트 보고 설정

다음 절차에서는 Jest 테스트 프레임워크를 AWS CodeBuild 사용하여에서 테스트 보고를 설정하는 방법을 보여줍니다. https://jestjs.io/

이 절차를 수행하려면 다음 전제 조건이 필요합니다.

- 기존 CodeBuild 프로젝트가 있어야 합니다.
- 프로젝트는 Jest 테스트 프레임워크를 사용하도록 설정된 Node.js 프로젝트입니다.

jest-junit 패키지를 프로젝트의 package.json 파일의 devDependencies 섹션에 추가합니다. CodeBuild는 이 패키지를 사용하여 JunitXml 형식으로 보고서를 생성합니다.

```
npm install --save-dev jest-junit
```

아직 없으면 프로젝트의 package.json 파일에 test 스크립트를 추가합니다. test 스크립트는 npm test가 실행될 때 Jest가 호출되도록 합니다.

```
{
```

```
  "scripts": {
    "test": "jest"
  }
}
```

Jest 구성 파일에 다음을 추가하여 `JunitXml` 리포터를 사용하도록 Jest를 구성하세요. 프로젝트에 Jest 구성 파일이 없는 경우, 프로젝트 루트에 `jest.config.js`라는 파일을 생성하고 다음을 추가하세요. 테스트 보고서는 *<test report directory>*/*<report filename>*으로 지정된 파일로 내보내집니다.

```
module.exports = {
  reporters: [
    'default',
    [ 'jest-junit', {
      outputDirectory: <test report directory>,
      outputName: <report filename>,
    } ]
  ]
};
```

`buildspec.yml` 파일에서 다음 섹션을 추가/업데이트합니다.

```
version: 0.2

phases:
  pre_build:
    commands:
      - npm install
  build:
    commands:
      - npm build
      - npm test

reports:
  jest_reports:
    files:
      - <report filename>
    file-format: JUNITXML
    base-directory: <test report directory>
```

pytest를 사용하여 테스트 보고 설정

다음 절차에서는 pytest 테스트 프레임워크를 AWS CodeBuild 사용하여에서 테스트 보고를 설정하는 방법을 보여줍니다. https://docs.pytest.org/

이 절차를 수행하려면 다음 전제 조건이 필요합니다.

- 기존 CodeBuild 프로젝트가 있어야 합니다.
- 프로젝트는 pytest 테스트 프레임워크를 사용하도록 설정된 Python 프로젝트입니다.

`buildspec.yml` 파일의 `build` 또는 `post_build` 단계에 다음 항목을 추가합니다. 이 코드는 현재 디렉터리에서 테스트를 자동으로 검색하고 *<test report directory>*/*<report filename>*으로 지정된 파일로 테스트 보고서를 내보냅니다. 보고서는 `JunitXml` 형식을 사용합니다.

```
      - python -m pytest --junitxml=<test report directory>/<report filename>
```

`buildspec.yml` 파일에서 다음 섹션을 추가/업데이트합니다.

```
version: 0.2

phases:
  install:
    runtime-versions:
      python: 3.7
    commands:
      - pip3 install pytest
  build:
    commands:
      - python -m pytest --junitxml=<test report directory>/<report filename>

reports:
  pytest_reports:
    files:
      - <report filename>
    base-directory: <test report directory>
    file-format: JUNITXML
```

RSpec을 사용하여 테스트 보고 설정

다음 절차에서는 RSpec 테스트 프레임워크를 AWS CodeBuild 사용하여에서 테스트 보고를 설정하는 방법을 보여줍니다. RSpec

이 절차를 수행하려면 다음 전제 조건이 필요합니다.

- 기존 CodeBuild 프로젝트가 있어야 합니다.
- 프로젝트는 RSpec 테스트 프레임워크를 사용하도록 설정된 Node.js 프로젝트입니다.

`buildspec.yml` 파일에서 다음을 추가/업데이트합니다. 이 코드는 *<test source directory>* 디렉터리에서 테스트를 실행하고 *<test report directory>*/*<report filename>*으로 지정된 파일로 테스트 보고서를 내보냅니다. 보고서는 `JunitXml` 형식을 사용합니다.

```
version: 0.2

phases:
  install:
    runtime-versions:
      ruby: 2.6
  pre_build:
    commands:
      - gem install rspec
      - gem install rspec_junit_formatter
  build:
    commands:
      - rspec <test source directory>/* --format RspecJunitFormatter --out <test report
 directory>/<report filename>
reports:
    rspec_reports:
        files:
            - <report filename>
        base-directory: <test report directory>
        file-format: JUNITXML
```

테스트 보고서 보기

테스트 케이스에 대한 정보, 통화 및 실패 횟수, 보고서 실행에 걸린 시간 등 테스트 보고서에 대한 세부 정보를 볼 수 있습니다. 빌드 실행, 보고서 그룹 또는 AWS 계정별로 그룹화된 테스트 보고서를 볼 수 있습니다. 콘솔에서 테스트 보고서를 선택하여 세부 정보 및 테스트 케이스의 결과를 확인합니다.

만료되지 않은 테스트 보고서를 볼 수 있습니다. 테스트 보고서는 작성되고 30일 후에 만료됩니다. CodeBuild에서 만료된 보고서를 볼 수 없습니다.

주제

- 빌드에 대한 테스트 보고서 보기
- 보고서 그룹에 대한 테스트 보고서 보기
- AWS 계정에서 테스트 보고서 보기

빌드에 대한 테스트 보고서 보기

빌드에 대한 테스트 보고서를 보려면

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 보려는 빌드를 찾습니다. 테스트 보고서를 만든 빌드를 실행한 프로젝트를 알고 있는 경우:

    1. 탐색 창에서 프로젝트 빌드를 선택한 다음, 보고자 하는 테스트 보고서를 실행한 빌드가 있는 프로젝트를 선택합니다.
    2. 빌드 기록을 선택한 다음, 보고자 하는 보고서를 만들어 실행한 빌드를 선택합니다.

    AWS 계정의 빌드 기록에서도 빌드를 찾을 수 있습니다.

    1. 탐색 창에서 빌드 기록을 선택한 다음, 보고자 하는 테스트 보고서를 만든 빌드를 선택합니다.
3. 빌드 페이지에서 보고서를 선택한 다음 테스트 보고서를 선택하여 세부 정보를 확인합니다.

보고서 그룹에 대한 테스트 보고서 보기

보고서 그룹에서 테스트 보고서를 보려면

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 Report groups(보고서 그룹)을 선택합니다.
3. 보려는 테스트 보고서가 포함된 보고서 그룹을 선택합니다.
4. 테스트 보고서를 선택하여 세부 정보를 확인합니다.

AWS 계정에서 테스트 보고서 보기

AWS 계정에서 테스트 보고서를 보려면

1. https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 보고서 기록을 선택합니다.
3. 테스트 보고서를 선택하여 세부 정보를 확인합니다.

테스트 보고서 권한

이 주제에서는 테스트 보고와 관련된 사용 권한에 대한 중요한 정보를 설명합니다.

주제

- 테스트 보고서의 IAM 역할
- 테스트 보고 작업에 대한 권한
- 테스트 보고 권한 예제

테스트 보고서의 IAM 역할

테스트 보고서를 실행하고 테스트 보고서를 포함하도록 프로젝트를 업데이트하려면 IAM 역할에 다음 권한이 필요합니다. 이러한 권한은 미리 정의된 AWS 관리형 정책에 포함됩니다. 기존 빌드 프로젝트에 테스트 보고를 추가하려면 이러한 권한을 직접 추가해야 합니다.

- `CreateReportGroup`
- `CreateReport`
- `UpdateReport`
- `BatchPutTestCases`

코드 범위 보고서를 실행하려면 IAM 역할에 `BatchPutCodeCoverages` 권한도 포함되어야 합니다.

> **Note**
>
> `BatchPutTestCases`, `CreateReport`, `UpdateReport` 및 `BatchPutCodeCoverages`는 퍼블릭 권한이 아닙니다. 이러한 권한에 대해 해당 AWS CLI 명령 또는 SDK 메서드를 호출할 수 없습니다.

이러한 권한을 가지려면 다음 정책을 IAM 역할에 연결하면 됩니다.

```
{
    "Effect": "Allow",
    "Resource": [
        "*"
    ],
    "Action": [
        "codebuild:CreateReportGroup",
        "codebuild:CreateReport",
        "codebuild:UpdateReport",
        "codebuild:BatchPutTestCases",
        "codebuild:BatchPutCodeCoverages"
    ]
}
```

이 정책은 사용해야 하는 보고서 그룹으로만 제한하는 것이 좋습니다. 다음은 정책에서 ARN이 두 개인 보고서 그룹으로만 권한을 제한합니다.

```
{
    "Effect": "Allow",
    "Resource": [
        "arn:aws:codebuild:your-region:your-aws-account-id:report-group/report-group-
name-1",
        "arn:aws:codebuild:your-region:your-aws-account-id:report-group/report-group-
name-2"
    ],
    "Action": [
        "codebuild:CreateReportGroup",
        "codebuild:CreateReport",
        "codebuild:UpdateReport",
        "codebuild:BatchPutTestCases",
        "codebuild:BatchPutCodeCoverages"
    ]
}
```

다음은 이름이 `my-project`로 지정된 프로젝트의 빌드를 실행하여 만든 보고서 그룹으로만 권한을 제한합니다.

```
{
    "Effect": "Allow",
    "Resource": [
        "arn:aws:codebuild:your-region:your-aws-account-id:report-group/my-project-*"
    ],
```

```
    "Action": [
        "codebuild:CreateReportGroup",
        "codebuild:CreateReport",
        "codebuild:UpdateReport",
        "codebuild:BatchPutTestCases",
        "codebuild:BatchPutCodeCoverages"
    ]
}
```

> **Note**
>
> 프로젝트에 지정된 CodeBuild 서비스 역할은 S3 버킷에 업로드하는 권한에 사용됩니다.

테스트 보고 작업에 대한 권한

다음 테스트 보고 CodeBuild API 작업에 대한 권한을 지정할 수 있습니다.

- `BatchGetReportGroups`
- `BatchGetReports`
- `CreateReportGroup`
- `DeleteReportGroup`
- `DeleteReport`
- `DescribeTestCases`
- `ListReportGroups`
- `ListReports`
- `ListReportsForReportGroup`
- `UpdateReportGroup`

자세한 내용은 AWS CodeBuild 권한 참조 단원을 참조하십시오.

테스트 보고 권한 예제

테스트 보고와 관련된 샘플 정책에 대한 내용은 다음을 참조하십시오.

- 사용자가 보고서 그룹을 변경하도록 허용

- 사용자가 보고서 그룹을 생성하도록 허용
- 사용자가 보고서를 삭제하도록 허용
- 사용자가 보고서 그룹을 삭제하도록 허용
- 사용자가 보고서 그룹에 대한 정보를 가져오도록 허용
- 사용자가 보고서에 대한 정보를 가져오도록 허용
- 사용자가 보고서 그룹 목록을 가져오도록 허용
- 사용자가 보고서 목록을 가져오도록 허용
- 사용자가 보고서 그룹에 대한 보고서 목록을 가져오도록 허용
- 사용자가 보고서에 대한 테스트 케이스 목록을 가져오도록 허용

테스트 보고서 상태

테스트 보고서의 상태는 다음 중 하나일 수 있습니다.

- `GENERATING`: 테스트 케이스의 실행이 아직 진행 중입니다.
- `DELETING`: 테스트 보고서가 삭제 중입니다. 테스트 보고서가 삭제되면 그 테스트 케이스도 삭제됩니다. S3 버킷으로 내보낸 원시 테스트 결과 데이터 파일은 삭제되지 않습니다.
- `INCOMPLETE`: 테스트 보고서가 완료되지 않았습니다. 이 상태는 다음 이유 중 하나 때문에 반환될 수 있습니다.
  - 이 보고서의 테스트 케이스를 지정하는 보고서 그룹의 구성에 문제가 있는 경우. 예를 들어, buildspec 파일의 보고서 그룹 아래의 테스트 케이스 경로가 올바르지 않을 수 있습니다.
  - 빌드를 실행한 IAM 사용자에게 테스트를 실행할 권한이 없는 경우. 자세한 내용은 테스트 보고서 권한 단원을 참조하십시오.
  - 테스트와 관련이 없는 오류로 인해 빌드가 완료되지 않은 경우.
- `SUCCEEDED`: 모든 테스트 케이스가 성공했습니다.
- `FAILED`: 일부 테스트 케이스는 성공하지 못했습니다.

각 테스트 케이스에서 상태를 반환합니다. 테스트 케이스의 상태는 다음 중 하나일 수 있습니다.

- `SUCCEEDED`: 테스트 케이스가 통과되었습니다.
- `FAILED`: 테스트 케이스가 실패했습니다.
- `ERROR`: 테스트 케이스로 인해 예기치 않은 오류가 발생했습니다.
- `SKIPPED`: 테스트 케이스가 실행되지 않았습니다.

- UNKNOWN: 테스트 케이스가 SUCCEEDED, FAILED, ERROR 또는 SKIPPED 이외의 상태를 반환했습니다.

테스트 보고서에는 최대 500개의 테스트 케이스 결과가 있을 수 있습니다. 500개 이상의 테스트 케이스가 실행되는 경우, CodeBuild가 상태 FAILED로 우선 순위를 지정하고 테스트 케이스 결과를 잘라냅니다.

Amazon Virtual Private Cloud AWS CodeBuild 와 함께 사용

일반적으로 AWS CodeBuild 는 VPC의 리소스에 액세스할 수 없습니다. 액세스를 활성화하려면 CodeBuild 프로젝트 구성에 추가적인 VPC별 구성 정보를 제공해야 합니다. 여기에는 VPC ID, VPC 서브넷 ID 및 VPC 보안 그룹 ID가 포함됩니다. 그러면 VPC 활성화 빌드가 VPC 내부의 리소스에 액세스할 수 있습니다. Amazon VPC의 VPC 설정에 대한 자세한 내용은 Amazon VPC 사용 설명서를 참조하세요.

주제

- 사용 사례
- VPC 모범 사례
- VPC의 제한 사항
- CodeBuild 프로젝트에서 Amazon VPC 액세스 허용
- VPC 설정 문제 해결
- VPC 엔드포인트 사용
- 관리형 프록시 서버와 AWS CodeBuild 함께 사용
- 프록시 서버와 AWS CodeBuild 함께 사용
- CloudFormation VPC 템플릿

사용 사례

AWS CodeBuild 빌드의 VPC 연결을 통해 다음을 수행할 수 있습니다.

- 프라이빗 서브넷에서 격리된 Amazon RDS 데이터베이스의 데이터에 대해 빌드에서 통합 테스트를 실행합니다.
- 테스트에서 Amazon ElastiCache 클러스터의 데이터를 직접 쿼리합니다.
- Amazon EC2, Amazon ECS에서 호스팅되는 내부 웹 서비스 또는 내부 Elastic Load Balancing을 사용하는 서비스와 상호 작용합니다.
- Python용 PyPI, Java용 Maven 및 Node.js용 npm과 같은 자체 호스팅된 내부 결과물 리포지토리의 종속성을 검색합니다.
- Amazon VPC 엔드포인트를 통해서만 액세스할 수 있도록 구성된 S3 버킷의 객체에 액세스합니다.
- 서브넷과 연결된 NAT 게이트웨이 또는 NAT 인스턴스의 탄력적 IP 주소를 통해 고정 IP 주소가 필요한 외부 웹 서비스를 쿼리합니다.

빌드는 VPC에서 호스팅되는 모든 리소스에 액세스할 수 있습니다.

VPC 모범 사례

CodeBuild에서 작동하도록 VPC를 설정할 때 이 체크리스트를 사용합니다.

- 퍼블릭 및 프라이빗 서브넷과 NAT 게이트웨이가 있는 VPC를 설정합니다. NAT 게이트웨이는 퍼블릭 서브넷에 상주해야 합니다. 자세한 내용은 Amazon VPC 사용 설명서의 퍼블릭 및 프라이빗 서브넷이 있는 VPC(NAT)를 참조하세요.

  > **Important**
  >
  > CodeBuild와 VPC를 함께 사용하여 CodeBuild가 퍼블릭 엔드포인트에 도달할 수 있게 하려면 NAT 게이트웨이 또는 NAT 인스턴스가 필요합니다(예: 빌드 실행 중 CLI 명령을 실행하는 경우). CodeBuild는 생성하는 네트워크 인터페이스에 탄력적 IP 주소를 할당하는 것을 지원하지 않기 때문에 NAT 게이트웨이 또는 NAT 인스턴스 대신 인터넷 게이트웨이를 사용할 수 없습니다. 또한 Amazon EC2 인스턴스를 시작하지 않고 생성된 네트워크 인터페이스의 경우 Amazon EC2는 퍼블릭 IP 주소 자동 할당을 지원하지 않습니다.

- VPC에 여러 가용 영역을 포함합니다.
- 보안 그룹에 빌드에 허용된 인바운드(수신) 트래픽이 없는지 확인합니다. CodeBuild에는 아웃바운드 트래픽에 대한 특정 요구 사항이 없지만 GitHub 또는 Amazon S3와 같이 빌드에 필요한 모든 인터넷 리소스에 대한 액세스를 허용해야 합니다.

  자세한 내용은 Amazon VPC 사용 설명서의 보안 그룹 규칙을 참조하세요.
- 빌드에 대해 별도의 서브넷을 설정합니다.
- VPC에 액세스하기 위해 CodeBuild 프로젝트를 설정할 때 프라이빗 서브넷만 포함합니다.

Amazon VPC의 VPC 설정에 대한 자세한 내용은 Amazon VPC 사용 설명서를 참조하세요.

CloudFormation 를 사용하여 CodeBuild VPC 기능을 사용하도록 VPC를 구성하는 방법에 대한 자세한 내용은 섹션을 참조하세요CloudFormation VPC 템플릿.

VPC의 제한 사항

- CodeBuild에서 VPC로의 연결은 공유 VPC에 대해 지원되지 않습니다.

CodeBuild 프로젝트에서 Amazon VPC 액세스 허용

VPC 구성에 다음 설정을 포함합니다.

- VPC ID에서 CodeBuild가 사용하는 VPC ID를 선택합니다.
- 서브넷에서 CodeBuild가 사용하는 리소스를 포함하거나 해당 리소스에 라우팅되는 NAT 변환을 통해 프라이빗 서브넷을 선택합니다.
- 보안 그룹에서 CodeBuild가 VPC의 리소스에 대한 액세스를 허용하기 위해 사용하는 보안 그룹을 선택합니다.

콘솔을 사용하여 빌드 프로젝트를 생성하려면 빌드 프로젝트 만들기(콘솔) 단원을 참조하십시오. CodeBuild 프로젝트를 생성하거나 변경할 때 VPC에서 VPC ID, 서브넷 및 보안 그룹을 선택합니다.

AWS CLI 를 사용하여 빌드 프로젝트를 생성하려면 섹션을 참조하세요빌드 프로젝트 생성(AWS CLI). CodeBuild와 AWS CLI 함께를 사용하는 경우 CodeBuild가 IAM 사용자를 대신하여 서비스와 상호 작용하는 데 사용하는 서비스 역할에 정책이 연결되어 있어야 합니다. 자세한 내용은 VPC 네트워크 인터페이스를 생성하는 데 필요한 AWS 서비스에 대한 CodeBuild 액세스 허용 단원을 참조하세요.

vpcConfig 객체는 *vpcId*, *securityGroupIds* 및 *subnets*을 포함해야 합니다.

- *vpcId*: 필수 항목입니다. CodeBuild가 사용하는 VPC ID입니다. 리전의 모든 Amazon VPC ID 목록을 가져오려면 다음 명령을 실행합니다.

```
aws ec2 describe-vpcs
```

- *subnets*: 필수 항목입니다. CodeBuild가 사용하는 리소스가 포함된 서브넷 ID입니다. 이 ID를 얻으려면 다음 명령을 실행합니다.

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=<vpc-id>" --region us-east-1
```

> **Note**
>
> us-east-1을 해당 리전으로 바꿉니다.

- *securityGroupIds*: 필수 항목입니다. CodeBuild가 VPC의 리소스에 대한 액세스를 허용하기 위해 사용하는 보안 그룹 ID입니다. 이 ID를 얻으려면 다음 명령을 실행합니다.

```
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=<vpc-id>" --region us-
east-1
```

> **Note**
>
> us-east-1을 해당 리전으로 바꿉니다.

VPC 설정 문제 해결

오류 메시지에 표시되는 정보를 사용하면 해당 문제를 식별, 진단 및 해결하는 데 도움이 됩니다.

다음은 일반적인 CodeBuild VPC 오류인 `Build does not have internet connectivity. Please check subnet network configuration`의 문제를 해결할 때 도움이 되는 몇 가지 지침입니다.

1. 인터넷 게이트웨이가 VPC에 연결되어 있는지 확인합니다.
2. 퍼블릭 서브넷의 라우팅 테이블이 인터넷 게이트웨이를 가리키는지 확인합니다.
3. 네트워크 ACL이 트래픽의 흐름을 허용하는지 확인합니다.
4. 보안 그룹이 트래픽의 흐름을 허용하는지 확인합니다.
5. NAT 게이트웨이 문제를 해결합니다.
6. 프라이빗 서브넷의 라우팅 테이블이 NAT 게이트웨이를 가리키는지 확인합니다.
7. CodeBuild가 IAM 사용자를 대신하여 서비스와 상호 작용할 때 사용하는 서비스 역할이 이 정책의 권한을 가지고 있어야 합니다. 자세한 내용은 CodeBuild가 다른 AWS 서비스와 상호 작용하도록 허용 단원을 참조하십시오.

   CodeBuild에 사용 권한이 없는 경우, `Unexpected EC2 error: UnauthorizedOperation` 오류가 표시될 수 있습니다. 이 오류는 VPC 작업에 필요한 Amazon EC2 권한이 CodeBuild에 없을 때 발생할 수 있습니다.

VPC 엔드포인트 사용

인터페이스 VPC 엔드포인트를 사용하도록 AWS CodeBuild 를 구성하여 빌드의 보안을 개선할 수 있습니다. 인터페이스 엔드포인트는 프라이빗 IP 주소를 사용하여 Amazon EC2 및 CodeBuild에 비공개로 액세스할 수 있는 기술인 PrivateLink로 구동됩니다. PrivateLink는 관리형 인스턴스, CodeBuild 및

Amazon EC2 간의 모든 네트워크 트래픽을 Amazon 네트워크로 제한합니다. (관리형 인스턴스는 인터넷에 액세스할 수 없음) 또한 인터넷 게이트웨이, NAT 디바이스 또는 가상 프라이빗 게이트웨이가 필요 없습니다. PrivateLink를 구성하는 것이 필수는 아니지만 구성하는 것이 좋습니다. PrivateLink 및 VPC 엔드포인트에 대한 자세한 내용은 란 무엇입니까 AWS PrivateLink?를 참조하세요.

VPC 엔드포인트를 생성하기 전에

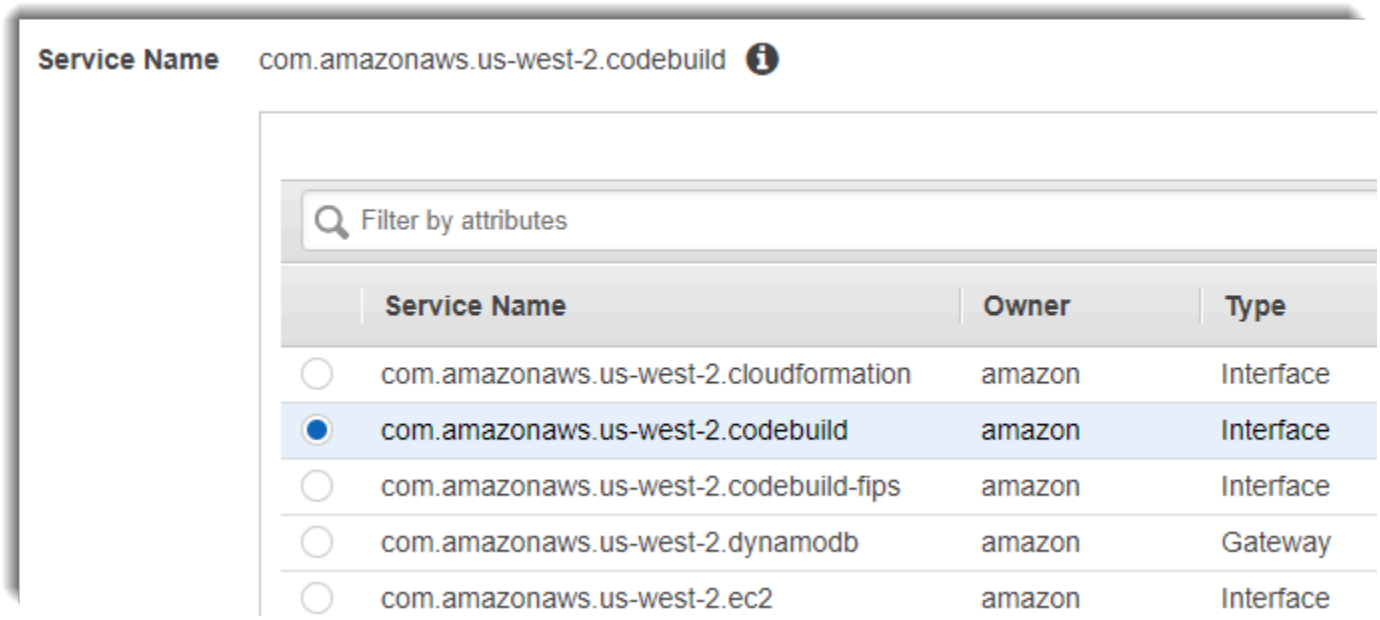
에 대한 VPC 엔드포인트를 구성하기 전에 다음 제한 사항에 유의 AWS CodeBuild하세요.

> **Note**
>
> Amazon VPC PrivateLink 연결을 지원하지 않는 AWS 서비스와 함께 CodeBuild를 사용하려면 NAT 게이트웨이를 사용합니다.

- VPC 엔드포인트는 Amazon Route 53을 통해 Amazon이 제공하는 DNS만 지원합니다. 자체 DNS를 사용하는 경우에는 조건부 DNS 전달을 사용할 수 있습니다. 자세한 정보는 Amazon VPC 사용 설명서의 DHCP 옵션 세트를 참조하세요.
- VPC 엔드포인트는 교차 리전 요청을 현재 지원하지 않습니다. 빌드 입력 및 출력을 저장하는 모든 S3 버킷과 동일한 AWS 리전에서 엔드포인트를 생성해야 합니다. Amazon S3 콘솔 또는 get-bucket-location 명령을 사용하여 버킷 위치를 찾을 수 있습니다. 리전별 Amazon S3 엔드포인트를 사용하여 버킷에 액세스하세요(예: `<bucket-name>.s3-us-west-2.amazonaws.com`). Amazon S3의 리전별 엔드포인트에 대한 자세한 내용은 Amazon Web Services 일반 참조의 Amazon Simple Storage Service를 참조하세요. AWS CLI 를 사용하여 Amazon S3에 요청하는 경우 기본 리전을 버킷이 생성된 리전과 동일한 리전으로 설정하거나 요청에 `--region` 파라미터를 사용합니다.

CodeBuild용 VPC 엔드포인트 생성

인터페이스 엔드포인트 생성의 지침에 따라 엔드포인트 `com.amazonaws.region.codebuild`를 만듭니다. 이는에 대한 VPC 엔드포인트입니다 AWS CodeBuild.



*region*은 미국 동부(오하이오) AWS 리전과 같이 CodeBuild에서 지원하는 리전us-east-2의 리전 식별자를 나타냅니다. 지원되는 AWS 리전 목록은 AWS 일반 참조의 CodeBuild를 참조하세요. 엔드포인트는 로그인 시 지정한 리전으로 미리 채워집니다 AWS. 리전을 변경하면 그에 따라 VPC 엔드포인트가 업데이트됩니다.

CodeBuild에 대한 VPC 엔드포인트 정책 생성

다음을 지정할 수 있는 Amazon VPC 엔드포인트 AWS CodeBuild 에 대한 정책을 생성할 수 있습니다.

- 작업을 수행할 수 있는 위탁자.
- 수행할 수 있는 작업.
- 작업을 수행할 수 있는 리소스.

다음 예제 정책은 모든 보안 주체가 `project-name` 프로젝트에 대한 빌드를 시작하고 볼 수만 있도록 지정합니다.

```
{
    "Statement": [
        {
            "Action": [
                "codebuild:ListBuildsForProject",
                "codebuild:StartBuild",
                "codebuild:BatchGetBuilds"
            ],
```

```
            "Effect": "Allow",
            "Resource": "arn:aws:codebuild:region-ID:account-ID:project/project-name",
            "Principal": "*"
        }
    ]
}
```

자세한 내용은 Amazon VPC 사용 설명서의 VPC 엔드포인트를 통해 서비스에 대한 액세스 제어를 참조하세요.

관리형 프록시 서버와 AWS CodeBuild 함께 사용

관리형 프록시 서버에서 AWS CodeBuild 예약 용량 플릿을 실행하려면 프록시 규칙을 사용하여 외부 사이트와의 트래픽을 허용하거나 거부하도록 프록시 서버를 구성해야 합니다. 관리형 프록시 서버에서 예약 용량 플릿을 실행하는 것은 VPC, Windows 또는 MacOS에서 지원되지 않습니다.

> **Important**
>
> 프록시 구성이 플릿에 존재하는 기간에 따라 추가 비용이 발생합니다. 자세한 내용은 https://aws.amazon.com/codebuild/pricing/ 참조하십시오.

주제

- 예약 용량 플릿에 대한 관리형 프록시 구성
- CodeBuild 예약 용량 플릿 실행

예약 용량 플릿에 대한 관리형 프록시 구성

예약 용량 플릿에 대해 관리형 프록시 서버를 구성하려면 콘솔에서 플릿을 생성하거나 AWS CLI를 사용할 때 이 기능을 활성화해야 합니다. 정의해야 할 몇 가지 속성이 있습니다.

프록시 구성 정의 - 선택 사항

예약 용량 인스턴스에 네트워크 액세스 제어를 적용하는 프록시 구성입니다.

기본 동작

발신 트래픽의 동작을 정의합니다.

허용

기본적으로 모든 대상으로 전송되는 트래픽을 허용합니다.

거부

기본적으로 모든 대상으로 전송되는 트래픽을 거부합니다.

프록시 규칙

네트워크 액세스 제어를 제한할 대상 도메인을 지정합니다.

콘솔에서 프록시 구성을 정의하려면 예약 용량 플릿 생성의 지침을 참조하세요. 를 사용하여 프록시 구성을 정의하려면 다음 JSON 구문을 수정하고 결과를 저장 AWS CLI하면 됩니다.

```
"proxyConfiguration": {
    "defaultBehavior": "ALLOW_ALL" | "DENY_ALL",
    "orderedProxyRules": [
        {
            "type": "DOMAIN" | "IP",
            "effect": "ALLOW" | "DENY",
            "entities": [
                "destination"
            ]
        }
    ]
}
```

JSON 파일은 다음과 비슷합니다.

```
"proxyConfiguration": {
    "defaultBehavior": "DENY_ALL",
    "orderedProxyRules": [
        {
            "type": "DOMAIN",
            "effect": "ALLOW",
            "entities": [
                "github.com"
            ]
        }
    ]
}
```

CodeBuild 예약 용량 플릿 실행

관리형 프록시 서버에서 AWS CodeBuild 예약 용량 플릿을 실행할 때 CodeBuild는 관리형 프록시 주소로 HTTP_PROXY 및 HTTPS_PROXY 환경 변수를 자동으로 설정합니다. 종속 항목 소프트웨어에 자체 구성이 있고 환경 변수를 준수하지 않는 경우 이러한 값을 참조하고 빌드 명령에서 소프트웨어 구성을 업데이트하여 관리형 프록시를 통해 빌드 트래픽을 올바르게 라우팅할 수 있습니다. 자세한 내용은 에서 빌드 프로젝트 생성 AWS CodeBuild 및 에서 빌드 프로젝트 설정 변경 AWS CodeBuild 섹션을 참조하세요.

프록시 서버와 AWS CodeBuild 함께 사용

프록시 서버와 AWS CodeBuild 함께를 사용하여 인터넷과 주고받는 HTTP 및 HTTPS 트래픽을 규제할 수 있습니다. 프록시 서버에서 CodeBuild를 실행하려면 VPC에서 퍼블릭 서브넷에 프록시 서버를 설치하고 가상 서브넷에 CodeBuild를 설치합니다.

프록시 서버에서 CodeBuild를 실행하는 주요 사용 사례는 두 가지입니다.

- VPC에서 NAT 게이트웨이 또는 NAT 인스턴스를 사용할 필요가 없어집니다.
- 프록시 서버 내 인스턴스가 액세스할 수 있는 URL을 지정하고 프록시 서버가 액세스를 거부하는 URL을 지정할 수 있습니다.

CodeBuild를 두 가지 유형의 프록시 서버에 사용할 수 있습니다. 두 유형 모두, 프록시 서버는 퍼블릭 서브넷에서 실행되고 CodeBuild는 프라이빗 서브넷에서 실행됩니다.

- 명시적 프록시: 명시적 프록시 서버를 사용하는 경우 프로젝트 수준에서 CodeBuild에 NO_PROXY, HTTP_PROXY 및 HTTPS_PROXY 환경 변수를 구성해야 합니다. 자세한 내용은 에서 빌드 프로젝트 설정 변경 AWS CodeBuild 및 에서 빌드 프로젝트 생성 AWS CodeBuild 섹션을 참조하세요.
- 투명 프록시: 투명 프록시 서버를 사용하는 경우 특별한 구성이 필요하지 않습니다.

주제

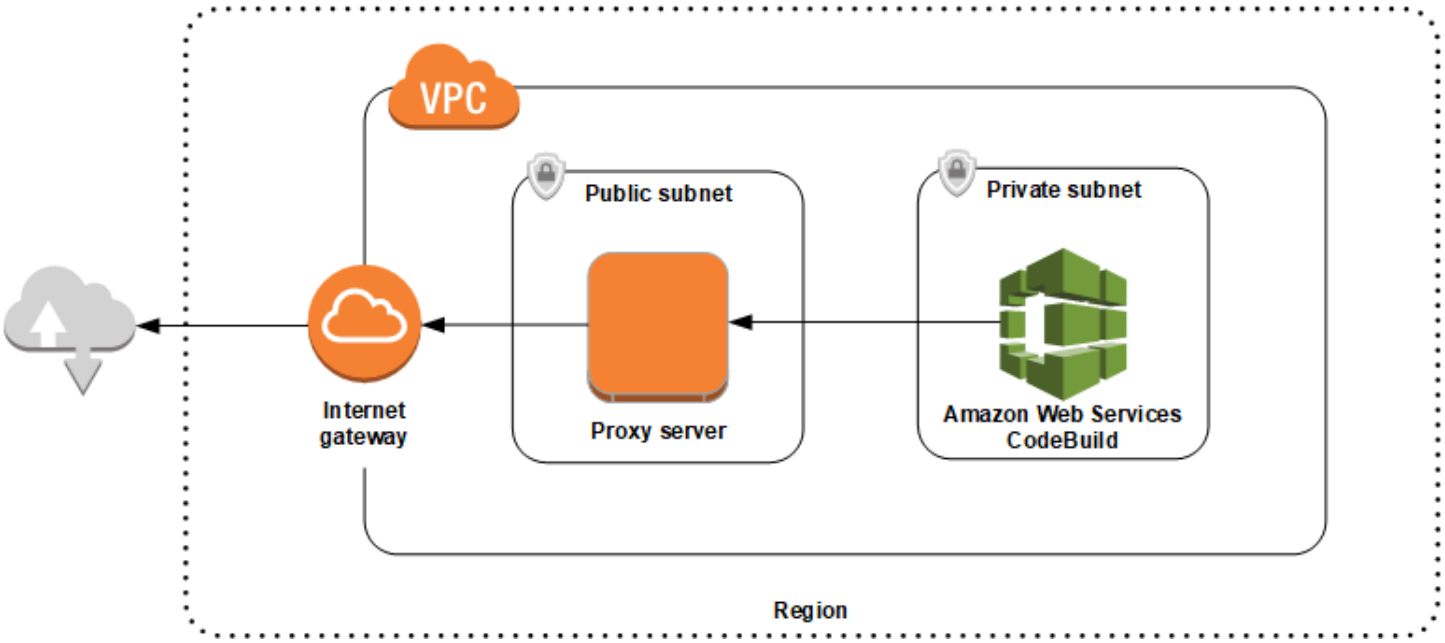
- 프록시 서버에서 CodeBuild를 실행하기 위해 필요한 구성 요소 설정
- 명시적 프록시 서버에서 CodeBuild 실행
- 투명 프록시 서버에서 CodeBuild 실행
- 프록시 서버에서 패키지 관리자 및 기타 도구 실행

프록시 서버에서 CodeBuild를 실행하기 위해 필요한 구성 요소 설정

투명 또는 명시적 프록시 서버에서 실행하려면 다음 구성 요소가 필요합니다 AWS CodeBuild .

- VPC
- 프록시 서버용 VPC에 퍼블릭 서브넷 1개.
- CodeBuild용 VPC에 프라이빗 서브넷 1개.
- VPC와 인터넷 간의 통신을 허용하는 인터넷 게이트웨이입니다.

다음 다이어그램은 구성 요소가 상호 작용하는 방식을 보여 줍니다.



VPC, 서브넷 및 네트워크 게이트웨이 설정

투명 또는 명시적 프록시 서버에서 AWS CodeBuild 를 실행하려면 다음 단계가 필요합니다.

1. VPC를 생성합니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC 생성을 참조하세요.
2. VPC에서 서브넷 2개를 만듭니다. 하나는 프록시 서버가 실행되는 `Public Subnet`이라는 퍼블릭 서브넷입니다. 다른 하나는 CodeBuild가 실행되는 `Private Subnet`이라는 프라이빗 서브넷입니다.

   자세한 내용은 VPC에서 서브넷 만들기 단원을 참조하십시오.
3. 인터넷 게이트웨이를 생성하여 VPC에 연결합니다. 자세한 내용은 인터넷 게이트웨이 생성 및 연결을 참조하세요.

4. 기본 라우팅 테이블에 VPC(0.0.0.0/0)에서 인터넷 게이트웨이로 가는 트래픽을 라우팅하는 규칙을 추가합니다. 자세한 내용은 라우팅 테이블에 경로 추가 및 라우팅 테이블에서 경로 제거 단원을 참조하십시오.
5. VPC의 기본 보안 그룹에 VPC(0.0.0.0/0)로부터 들어오는 SSH 트래픽(TCP 22)을 허용하는 규칙을 추가합니다.
6. Amazon EC2 인스턴스를 시작하려면 Amazon EC2 사용 설명서에서 인스턴스 시작 마법사를 사용하여 인스턴스 시작의 지침을 따르세요. 마법사를 실행할 때 다음 옵션을 선택합니다.
   - 인스턴스 유형 선택에서 Amazon Linux Amazon Machine Image(AMI)를 선택합니다.
   - 서브넷에서 이 주제의 앞부분에서 만든 퍼블릭 서브넷을 선택합니다. 제안된 이름을 사용했다면 퍼블릭 서브넷입니다.
   - [Auto-assign Public IP]에서 [Enable]을 선택합니다.
   - 보안 그룹 구성 페이지의 보안 그룹 할당에서 Select an existing security group(기존 보안 그룹 선택)을 선택합니다. 그런 다음 기본 보안 그룹을 선택합니다.
   - 시작을 선택한 후 기존 키 페어를 하나 선택하거나 생성합니다.

   다른 옵션은 모두 기본 설정을 선택합니다.
7. EC2 인스턴스가 실행되면 원본/대상 확인을 비활성화합니다. 자세한 내용은 Amazon VPC 사용 설명서의 소스/대상 확인 비활성화를 참조하세요.
8. VPC에서 라우팅 테이블을 만듭니다. 라우팅 테이블에 인터넷으로 향하는 트래픽을 프록시 서버로 라우팅하는 규칙을 추가합니다. 이 라우팅 테이블을 프라이빗 서브넷과 연결합니다. 이 단계는 CodeBuild가 실행되는 프라이빗 서브넷 내 인스턴스의 아웃바운드 요청이 항상 프록시 서버를 통해 라우팅되도록 하는 데 필요합니다.

프록시 서버 설치 및 구성

선택할 수 있는 프록시 서버는 많습니다. 오픈 소스 프록시 서버인 Squid는가 프록시 서버에서 어떻게 AWS CodeBuild 실행되는지 보여주는 데 사용됩니다. 동일한 개념을 다른 프록시 서버에도 적용할 수 있습니다.

Squid를 설치하려면 다음 명령을 실행하여 yum repo를 사용합니다.

```
sudo yum update -y
sudo yum install -y squid
```

Squid를 설치한 후 이 주제의 뒷부분에 나오는 지침에 따라 `squid.conf` 파일을 편집합니다.

HTTPS 트래픽에 대해 Squid 구성

HTTPS의 경우, TLS(전송 계층 보안) 연결에서 HTTP 트래픽이 캡슐화된 것입니다. Squid는 SslPeekAndSplice라는 기능을 사용하여 TLS 초기화에서 요청된 인터넷 호스트를 포함하는 SNI(서버 이름 표시)를 검색합니다. 이는 Squid가 HTTPS 트래픽을 해독할 필요가 없도록 하기 위해 필요합니다. SslPeekAndSplice를 활성화하려면 Squid가 인증서를 요구합니다. OpenSSL을 사용하여 이 인증서를 만듭니다.

```
sudo mkdir /etc/squid/ssl
cd /etc/squid/ssl
sudo openssl genrsa -out squid.key 2048
sudo openssl req -new -key squid.key -out squid.csr -subj "/C=XX/ST=XX/L=squid/O=squid/
CN=squid"
sudo openssl x509 -req -days 3650 -in squid.csr -signkey squid.key -out squid.crt
sudo cat squid.key squid.crt | sudo tee squid.pem
```

> **Note**
>
> HTTP의 경우, Squid를 구성할 필요가 없습니다. 모든 HTTP/1.1 요청 메시지에서 요청되는 인터넷 호스를 지정하는 호스트 헤더 필드를 검색할 수 있습니다.

명시적 프록시 서버에서 CodeBuild 실행

명시적 프록시 서버에서 AWS CodeBuild 를 실행하려면 외부 사이트와의 트래픽을 허용하거나 거부하도록 프록시 서버를 구성한 다음 HTTP_PROXY 및 HTTPS_PROXY 환경 변수를 구성해야 합니다.

주제

- Squid를 명시적 프록시 서버로 구성
- CodeBuild 프로젝트 생성
- 명시적 프록시 서버 샘플 squid.conf 파일

Squid를 명시적 프록시 서버로 구성

Squid를 명시적 프록시 서버로 구성하려면 다음과 같이 /etc/squid/squid.conf 파일을 수정해야 합니다.

- 다음 기본 ACL(액세스 제어 목록) 규칙을 제거합니다.

```
acl localnet src 10.0.0.0/8
acl localnet src 172.16.0.0/12
acl localnet src 192.168.0.0/16
acl localnet src fc00::/7
acl localnet src fe80::/10
```

제거한 기본 ACL 규칙 대신 다음 규칙을 추가합니다. 첫 번째 줄은 VPC의 요청을 허용합니다. 다음 두 줄은 프록시 서버에 AWS CodeBuild가 사용할 수 있는 대상 URL에 대한 액세스 권한을 부여합니다. 마지막 줄에서 정규식을 편집하여 AWS 리전에서 S3 버킷 또는 CodeCommit 리포지토리를 지정합니다. 예제:

- 소스가 Amazon S3일 경우 acl download_src dstdom_regex .*s3\.us-west-1\.amazonaws\.com 명령을 사용하여 `us-west-1` 리전 내 S3 버킷에 대한 액세스 권한을 부여합니다.
- 소스가 인 경우 AWS CodeCommit`git-codecommit.<your-region>.amazonaws.com`를 사용하여 허용 목록에 AWS 리전을 추가합니다.

```
acl localnet src 10.1.0.0/16 #Only allow requests from within the VPC
acl allowed_sites dstdomain .github.com #Allows to download source from GitHub
acl allowed_sites dstdomain .bitbucket.com #Allows to download source from Bitbucket
acl download_src dstdom_regex .*\.amazonaws\.com #Allows to download source from
 Amazon S3 or CodeCommit
```

- `http_access allow localnet`을 다음으로 바꿉니다.

```
http_access allow localnet allowed_sites
http_access allow localnet download_src
```

- 빌드가 로그 및 아티팩트를 업로드하도록 하려면 다음 중 하나를 수행하십시오.

  1. `http_access deny all` 문 앞에 다음 문을 삽입합니다. 이를 통해 CodeBuild는 CloudWatch와 Amazon S3에 액세스할 수 있습니다. CodeBuild에서 CloudWatch Logs를 생성하려면 CloudWatch에 대한 액세스 권한이 필요합니다. Amazon S3에 대한 액세스는 아티팩트 및 Amazon S3 캐싱을 업로드하기 위해 필요합니다.

     - ```
       https_port 3130 cert=/etc/squid/ssl/squid.pem ssl-bump intercept
       acl SSL_port port 443
       http_access allow SSL_port
       acl allowed_https_sites ssl::server_name .amazonaws.com
       acl step1 at_step SslBump1
       acl step2 at_step SslBump2
       acl step3 at_step SslBump3
       ```
       ```

```
ssl_bump peek step1 all
ssl_bump peek step2 allowed_https_sites
ssl_bump splice step3 allowed_https_sites
ssl_bump terminate step2 all
```

- `squid.conf`를 저장한 후 다음 명령을 실행합니다.

```
sudo iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --to-port 3130
sudo service squid restart
```

2. buildspec 파일에 `proxy`를 추가합니다. 자세한 내용은 buildspec 구문 단원을 참조하십시오.

```
version: 0.2
proxy:
  upload-artifacts: yes
  logs: yes
phases:
  build:
    commands:
      - command
```

> **Note**
>
> RequestError 시간 초과 오류가 발생할 경우 프록시 서버에서 CodeBuild를 실행할 때 RequestError 시간 초과 오류 단원을 참조하십시오.

자세한 내용은 본 주제의 후반부에서 명시적 프록시 서버 샘플 `squid.conf` 파일을 참조하세요.

CodeBuild 프로젝트 생성

명시적 프록시 서버를 AWS CodeBuild 사용하여를 실행하려면 `HTTP_PROXY` 및 `HTTPS_PROXY` 환경 변수를 프록시 서버에 대해 생성한 EC2 인스턴스의 프라이빗 IP 주소와 프로젝트 수준에서 포트 3128로 설정합니다. 프라이빗 IP 주소는 `http://`*`your-ec2-private-ip-address`*`:3128`과 비슷합니다. 자세한 내용은 에서 빌드 프로젝트 생성 AWS CodeBuild 및 에서 빌드 프로젝트 설정 변경 AWS CodeBuild 섹션을 참조하세요.

다음 명령을 사용하여 Squid 프록시 액세스 로그를 확인합니다.

```
sudo tail -f /var/log/squid/access.log
```

명시적 프록시 서버 샘플 **squid.conf** 파일

다음은 명시적 프록시 서버용으로 구성된 squid.conf 파일의 예입니다.

```
 acl localnet src 10.0.0.0/16 #Only allow requests from within the VPC
 # add all URLS to be whitelisted for download source and commands to be run in build
environment
 acl allowed_sites dstdomain .github.com    #Allows to download source from github
 acl allowed_sites dstdomain .bitbucket.com #Allows to download source from bitbucket
 acl allowed_sites dstdomain ppa.launchpad.net #Allows to run apt-get in build
environment
 acl download_src dstdom_regex .*\.amazonaws\.com #Allows to download source from S3
or CodeCommit
 acl SSL_ports port 443
 acl Safe_ports port 80  # http
 acl Safe_ports port 21  # ftp
 acl Safe_ports port 443  # https
 acl Safe_ports port 70  # gopher
 acl Safe_ports port 210  # wais
 acl Safe_ports port 1025-65535 # unregistered ports
 acl Safe_ports port 280  # http-mgmt
 acl Safe_ports port 488  # gss-http
 acl Safe_ports port 591  # filemaker
 acl Safe_ports port 777  # multiling http
 acl CONNECT method CONNECT
 #
 # Recommended minimum Access Permission configuration:
 #
 # Deny requests to certain unsafe ports
 http_access deny !Safe_ports
 # Deny CONNECT to other than secure SSL ports
 http_access deny CONNECT !SSL_ports
 # Only allow cachemgr access from localhost
 http_access allow localhost manager
 http_access deny manager
 # We strongly recommend the following be uncommented to protect innocent
 # web applications running on the proxy server who think the only
 # one who can access services on "localhost" is a local user
 #http_access deny to_localhost
 #
 # INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS
 #
 # Example rule allowing access from your local networks.
 # Adapt localnet in the ACL section to list your (internal) IP networks
```

```
  # from where browsing should be allowed
  http_access allow localnet allowed_sites
  http_access allow localnet download_src
  http_access allow localhost
  # Add this for CodeBuild to access CWL end point, caching and upload artifacts S3
bucket end point
  https_port 3130 cert=/etc/squid/ssl/squid.pem ssl-bump intercept
  acl SSL_port port 443
  http_access allow SSL_port
  acl allowed_https_sites ssl::server_name .amazonaws.com
  acl step1 at_step SslBump1
  acl step2 at_step SslBump2
  acl step3 at_step SslBump3
  ssl_bump peek step1 all
  ssl_bump peek step2 allowed_https_sites
  ssl_bump splice step3 allowed_https_sites
  ssl_bump terminate step2 all
  # And finally deny all other access to this proxy
  http_access deny all
  # Squid normally listens to port 3128
  http_port 3128
  # Uncomment and adjust the following to add a disk cache directory.
  #cache_dir ufs /var/spool/squid 100 16 256
  # Leave coredumps in the first cache dir
  coredump_dir /var/spool/squid
  #
  # Add any of your own refresh_pattern entries above these.
  #
  refresh_pattern ^ftp:  1440 20% 10080
  refresh_pattern ^gopher: 1440 0% 1440
  refresh_pattern -i (/cgi-bin/|\?) 0 0% 0
  refresh_pattern .  0 20% 4320
```

투명 프록시 서버에서 CodeBuild 실행

투명한 프록시 서버에서 AWS CodeBuild 를 실행하려면 상호 작용하는 웹 사이트 및 도메인에 액세스할 수 있도록 프록시 서버를 구성해야 합니다.

주제

- Squid를 투명 프록시 서버로 구성
- CodeBuild 프로젝트 생성

Squid를 투명 프록시 서버로 구성

프록시 서버를 투명으로 구성하려면 액세스하려는 도메인 및 웹 사이트에 대한 액세스 권한을 부여해야 합니다. 투명한 프록시 서버 AWS CodeBuild 로를 실행하려면에 대한 액세스 권한을 부여해야 합니다amazonaws.com. 또한 CodeBuild가 사용하는 다른 웹 사이트에 대해서도 액세스 권한을 부여해야 합니다. 이러한 웹 사이트는 CodeBuild 프로젝트를 생성하는 방식에 따라 달라집니다. 예를 들어 GitHub, Bitbucket, Yum, Maven 같은 리포지토리용 웹 사이트입니다. Squid에 특정 도메인 및 웹 사이트에 대한 액세스 권한을 부여하려면 다음과 같은 명령을 사용하여 squid.conf 파일을 업데이트합니다. 이 샘플 명령은 amazonaws.com, github.com 및 bitbucket.com에 대한 액세스 권한을 부여합니다. 이 샘플을 편집하여 다른 웹 사이트에 대한 액세스 권한을 부여할 수 있습니다.

```
cat | sudo tee /etc/squid/squid.conf #EOF
visible_hostname squid
#Handling HTTP requests
http_port 3129 intercept
acl allowed_http_sites dstdomain .amazonaws.com
#acl allowed_http_sites dstdomain domain_name [uncomment this line to add another
 domain]
http_access allow allowed_http_sites
#Handling HTTPS requests
https_port 3130 cert=/etc/squid/ssl/squid.pem ssl-bump intercept
acl SSL_port port 443
http_access allow SSL_port
acl allowed_https_sites ssl::server_name .amazonaws.com
acl allowed_https_sites ssl::server_name .github.com
acl allowed_https_sites ssl::server_name .bitbucket.com
#acl allowed_https_sites ssl::server_name [uncomment this line to add another website]
acl step1 at_step SslBump1
acl step2 at_step SslBump2
acl step3 at_step SslBump3
ssl_bump peek step1 all
ssl_bump peek step2 allowed_https_sites
ssl_bump splice step3 allowed_https_sites
ssl_bump terminate step2 all
http_access deny all
EOF
```

프라이빗 서브넷에 있는 인스턴스에서 오는 수신 요청은 Squid 포트로 리디렉션해야 합니다. Squid는 HTTP 트래픽의 경우 포트 3129(80이 아님)에서 수신 대기하고, HTTPS 트래픽의 경우 포트 3130(443이 아님)에서 수신 대기합니다. iptables 명령을 사용하여 트래픽을 라우팅합니다.

```
sudo iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-port 3129
sudo iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --to-port 3130
sudo service iptables save
sudo service squid start
```

CodeBuild 프로젝트 생성

프록시 서버를 구성한 후에는 추가 구성 없이 프라이빗 서브넷 AWS CodeBuild 에서와 함께 사용할 수 있습니다. 모든 HTTP 및 HTTPS 요청이 퍼블릭 프록시 서버를 통과합니다. 다음 명령을 사용하여 Squid 프록시 액세스 로그를 확인합니다.

```
sudo tail -f /var/log/squid/access.log
```

프록시 서버에서 패키지 관리자 및 기타 도구 실행

다음 절차에 따라 프록시 서버에서 패키지 관리자 및 기타 도구를 실행합니다.

프록시 서버에서 패키지 관리자와 같은 도구를 실행하려면

1. `squid.conf` 파일에 문을 추가하여 프록시 서버의 허용 목록에 도구를 추가합니다.
2. 프록시 서버의 프라이빗 엔드포인트를 가리키는 줄을 buildspec 파일에 추가합니다.

다음 예제는 `apt-get`, `curl` 및 `maven`에서 이렇게 하는 방법을 보여줍니다. 다른 도구를 사용하는 경우에도 동일한 원칙이 적용됩니다. CodeBuild가 프록시 서버의 엔드포인트를 인식하도록 `squid.conf` 파일의 허용 목록에 추가하고 buildspec 파일에 명령을 추가합니다.

프록시 서버에서 **`apt-get`**을 실행하려면

1. `squid.conf` 파일에 다음 문을 추가하여 프록시 서버의 허용 목록에 `apt-get`을 추가합니다. 처음 세 줄은 빌드 환경에서 `apt-get`을 실행하도록 허용합니다.

```
acl allowed_sites dstdomain ppa.launchpad.net # Required for apt-get to run in the
 build environment
acl apt_get dstdom_regex .*\.launchpad.net # Required for CodeBuild to run apt-get
 in the build environment
acl apt_get dstdom_regex .*\.ubuntu.com    # Required for CodeBuild to run apt-get
 in the build environment
http_access allow localnet allowed_sites
http_access allow localnet apt_get
```

2. apt-get 명령이 /etc/apt/apt.conf.d/00proxy에서 프록시 구성을 검색하도록 buildspec 파일에 다음 문을 추가합니다.

```
echo 'Acquire::http::Proxy "http://<private-ip-of-proxy-server>:3128";
 Acquire::https::Proxy "http://<private-ip-of-proxy-server>:3128";
 Acquire::ftp::Proxy "http://<private-ip-of-proxy-server>:3128";' > /etc/apt/
apt.conf.d/00proxy
```

프록시 서버에서 **curl**을 실행하려면

1. squid.conf 파일에 다음을 추가하여 빌드 환경의 허용 목록에 curl을 추가합니다.

```
acl allowed_sites dstdomain ppa.launchpad.net # Required to run apt-get in the
 build environment
acl allowed_sites dstdomain google.com # Required for access to a webiste. This
 example uses www.google.com.
http_access allow localnet allowed_sites
http_access allow localnet apt_get
```

2. curl이 프라이빗 프록시 서버를 사용하여 squid.conf에 추가한 웹 사이트에 액세스하도록 buildspec 파일에 다음 문을 추가합니다. 이 예제에서 웹 사이트는 google.com입니다.

```
curl -x <private-ip-of-proxy-server>:3128 https://www.google.com
```

프록시 서버에서 **maven**을 실행하려면

1. squid.conf 파일에 다음을 추가하여 빌드 환경의 허용 목록에 maven을 추가합니다.

```
acl allowed_sites dstdomain ppa.launchpad.net # Required to run apt-get in the
 build environment
acl maven dstdom_regex .*\.maven.org # Allows access to the maven repository in the
 build environment
http_access allow localnet allowed_sites
http_access allow localnet maven
```

2. buildspec 파일에 다음 문을 추가합니다.

```
maven clean install -DproxySet=true -DproxyHost=<private-ip-of-proxy-server> -
DproxyPort=3128
```

CloudFormation VPC 템플릿

CloudFormation 를 사용하면 템플릿 파일을 사용하여 리소스 모음을 단일 단위(스택)로 함께 생성하고 삭제하여 AWS 인프라 배포를 예측 가능하고 반복적으로 생성하고 프로비저닝할 수 있습니다. 자세한 내용은 CloudFormation 사용 설명서를 참조하십시오.

다음은 사용할 VPC를 구성하기 위한 CloudFormation YAML 템플릿입니다 AWS CodeBuild. 이 파일은 samples.zip에서도 사용할 수 있습니다.

```
Description:  This template deploys a VPC, with a pair of public and private subnets
 spread
  across two Availability Zones. It deploys an internet gateway, with a default
  route on the public subnets. It deploys a pair of NAT gateways (one in each AZ),
  and default routes for them in the private subnets.

Parameters:
  EnvironmentName:
    Description: An environment name that is prefixed to resource names
    Type: String

  VpcCIDR:
    Description: Please enter the IP range (CIDR notation) for this VPC
    Type: String
    Default: 10.192.0.0/16

  PublicSubnet1CIDR:
    Description: Please enter the IP range (CIDR notation) for the public subnet in the
 first Availability Zone
    Type: String
    Default: 10.192.10.0/24

  PublicSubnet2CIDR:
    Description: Please enter the IP range (CIDR notation) for the public subnet in the
 second Availability Zone
    Type: String
    Default: 10.192.11.0/24

  PrivateSubnet1CIDR:
    Description: Please enter the IP range (CIDR notation) for the private subnet in
 the first Availability Zone
    Type: String
    Default: 10.192.20.0/24
```

```
  PrivateSubnet2CIDR:
    Description: Please enter the IP range (CIDR notation) for the private subnet in
 the second Availability Zone
    Type: String
    Default: 10.192.21.0/24

Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref VpcCIDR
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
        - Key: Name
          Value: !Ref EnvironmentName

  InternetGateway:
    Type: AWS::EC2::InternetGateway
    Properties:
      Tags:
        - Key: Name
          Value: !Ref EnvironmentName

  InternetGatewayAttachment:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      InternetGatewayId: !Ref InternetGateway
      VpcId: !Ref VPC

  PublicSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 0, !GetAZs '' ]
      CidrBlock: !Ref PublicSubnet1CIDR
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Public Subnet (AZ1)

  PublicSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
```

```
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 1, !GetAZs  '' ]
      CidrBlock: !Ref PublicSubnet2CIDR
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Public Subnet (AZ2)

  PrivateSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 0, !GetAZs  '' ]
      CidrBlock: !Ref PrivateSubnet1CIDR
      MapPublicIpOnLaunch: false
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Subnet (AZ1)

  PrivateSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 1, !GetAZs  '' ]
      CidrBlock: !Ref PrivateSubnet2CIDR
      MapPublicIpOnLaunch: false
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Subnet (AZ2)

  NatGateway1EIP:
    Type: AWS::EC2::EIP
    DependsOn: InternetGatewayAttachment
    Properties:
      Domain: vpc

  NatGateway2EIP:
    Type: AWS::EC2::EIP
    DependsOn: InternetGatewayAttachment
    Properties:
      Domain: vpc

  NatGateway1:
    Type: AWS::EC2::NatGateway
```

```
    Properties:
      AllocationId: !GetAtt NatGateway1EIP.AllocationId
      SubnetId: !Ref PublicSubnet1

  NatGateway2:
    Type: AWS::EC2::NatGateway
    Properties:
      AllocationId: !GetAtt NatGateway2EIP.AllocationId
      SubnetId: !Ref PublicSubnet2

  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Public Routes

  DefaultPublicRoute:
    Type: AWS::EC2::Route
    DependsOn: InternetGatewayAttachment
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway

  PublicSubnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet1

  PublicSubnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet2


  PrivateRouteTable1:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
```

```
      - Key: Name
        Value: !Sub ${EnvironmentName} Private Routes (AZ1)

DefaultPrivateRoute1:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PrivateRouteTable1
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NatGateway1

PrivateSubnet1RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PrivateRouteTable1
    SubnetId: !Ref PrivateSubnet1

PrivateRouteTable2:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Private Routes (AZ2)

DefaultPrivateRoute2:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PrivateRouteTable2
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NatGateway2

PrivateSubnet2RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PrivateRouteTable2
    SubnetId: !Ref PrivateSubnet2

NoIngressSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupName: "no-ingress-sg"
    GroupDescription: "Security group with no ingress rule"
    VpcId: !Ref VPC
```

```
Outputs:
  VPC:
    Description: A reference to the created VPC
    Value: !Ref VPC

  PublicSubnets:
    Description: A list of the public subnets
    Value: !Join [ ",", [ !Ref PublicSubnet1, !Ref PublicSubnet2 ]]

  PrivateSubnets:
    Description: A list of the private subnets
    Value: !Join [ ",", [ !Ref PrivateSubnet1, !Ref PrivateSubnet2 ]]

  PublicSubnet1:
    Description: A reference to the public subnet in the 1st Availability Zone
    Value: !Ref PublicSubnet1

  PublicSubnet2:
    Description: A reference to the public subnet in the 2nd Availability Zone
    Value: !Ref PublicSubnet2

  PrivateSubnet1:
    Description: A reference to the private subnet in the 1st Availability Zone
    Value: !Ref PrivateSubnet1

  PrivateSubnet2:
    Description: A reference to the private subnet in the 2nd Availability Zone
    Value: !Ref PrivateSubnet2

  NoIngressSecurityGroup:
    Description: Security group with no ingress rule
    Value: !Ref NoIngressSecurityGroup
```

에서 로깅 및 모니터링 AWS CodeBuild

로깅 및 모니터링은 AWS CodeBuild 및 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 데 중요한 부분입니다. 다중 지점 장애가 발생할 경우 보다 쉽게 디버깅할 수 있도록 AWS 솔루션의 모든 부분으로부터 모니터링 데이터를 수집해야 합니다.는 CodeBuild 리소스 및 빌드를 모니터링하고 잠재적 인시던트에 대응하기 위한 다음 도구를 AWS 제공합니다.

주제

- 를 사용하여 AWS CodeBuild API 호출 로깅 AWS CloudTrail
- CloudWatch를 사용하여 CodeBuild 빌드 모니터링

를 사용하여 AWS CodeBuild API 호출 로깅 AWS CloudTrail

AWS CodeBuild 는 CodeBuild에서 사용자 AWS CloudTrail, 역할 또는 서비스가 수행한 작업에 대한 레코드를 제공하는 AWS 서비스와 통합됩니다. CloudTrail은 CodeBuild 콘솔의 호출 및 CodeBuild API에 대한 코드 호출을 포함하여 CodeBuild에 대한 모든 API 직접 호출을 이벤트로 캡처합니다. 추적을 생성하면 CodeBuild 이벤트를 포함한 CloudTrail 이벤트를 지속적으로 S3 버킷에 배포할 수 있습니다. 추적을 구성하지 않은 경우에도 이벤트 기록에서 CloudTrail 콘솔의 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 CodeBuild에 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.

주제

- CloudTrail의 AWS CodeBuild 정보
- AWS CodeBuild 로그 파일 항목 정보

CloudTrail의 AWS CodeBuild 정보

AWS 계정을 생성할 때 계정에서 CloudTrail이 활성화됩니다. CodeBuild에서 활동이 발생하면 해당 활동이 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. AWS 계정에서 최근 이벤트를 보고 검색하고 다운로드할 수 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 이벤트 기록으로 이벤트 보기를 참조하세요.

CodeBuild에 대한 이벤트를 포함하여 AWS 계정에 이벤트를 지속적으로 기록하려면 추적을 생성합니다. 추적은 CloudTrail이 S3 버킷으로 로그 파일을 전송할 수 있도록 합니다. 콘솔에서 트레일을 생성하

면 기본적으로 모든 리전에 트레일이 적용됩니다. 추적은 AWS 파티션의 모든 리전에서 이벤트를 로깅하고 지정한 S3 버킷으로 로그 파일을 전송합니다. CloudTrail 로그에서 수집된 이벤트 데이터를 추가로 분석하고 조치를 취하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음을 참조하세요.

- 추적 생성 개요
- CloudTrail 지원 서비스 및 통합
- CloudTrail에 대한 Amazon SNS 알림 구성
- 여러 리전에서 CloudTrail 로그 파일 받기 및 여러 계정에서 CloudTrail 로그 파일 받기

모든 CodeBuild 작업은 CloudTrail에서 로깅되고 CodeBuild API 참조에 기록됩니다. 예를 들어 (, `create-project`), `CreateProject` (AWS CLI, `StartBuild` AWS CLI`start-project`) 및 `UpdateProject` () AWS CLI`update-project`작업에 대한 호출은 CloudTrail 로그 파일에 항목을 생성합니다.

모든 이벤트 또는 로그 항목에는 요청을 생성했던 사용자에 관한 정보가 포함됩니다. 보안 인증 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 아니면 사용자 자격 증명으로 했는지 여부
- 역할 또는 페더레이션 사용자에 대한 임시 보안 인증을 사용하여 요청이 생성되었는지 여부.
- 요청이 다른 AWS 서비스에 의해 이루어졌는지 여부입니다.

자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail userIdentity 요소를 참조하세요.

AWS CodeBuild 로그 파일 항목 정보

추적이란 지정한 S3 버킷에 이벤트를 로그 파일로 입력할 수 있도록 하는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스의 단일 요청을 나타내며 요청된 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보를 포함하고 있습니다. CloudTrail 로그 파일은 퍼블릭 API 직접 호출에 대한 순서 지정된 스택 추적이 아니기 때문에 특정 순서로 표시되지 않습니다.

> **Note**
>
> 중요한 정보를 보호하기 위해 CodeBuild 로그에 다음 항목이 숨겨져 있습니다.

- AWS 액세스 키 IDs. 자세한 내용은AWS Identity and Access Management 사용 설명서에서 IAM 사용자의 액세스 키 관리를 참조하세요.
- 파라미터 스토어를 사용하여 지정된 문자열입니다. 자세한 내용은 Amazon EC2 Systems Manager 사용 설명서의 Systems Manager Parameter Store 및 Systems Manager Parameter Store 콘솔 연습을 참조하세요.
- 를 사용하여 지정된 문자열입니다 AWS Secrets Manager. 자세한 내용은 키 관리 단원을 참조하십시오.

다음은 CodeBuild에서 빌드 프로젝트 만들기 작업을 보여 주는 CloudTrail 로그 항목의 예입니다.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "FederatedUser",
    "principalId": "account-ID:user-name",
    "arn": "arn:aws:sts::account-ID:federated-user/user-name",
    "accountId": "account-ID",
    "accessKeyId": "access-key-ID",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2016-09-06T17:59:10Z"
      },
      "sessionIssuer": {
        "type": "IAMUser",
        "principalId": "access-key-ID",
        "arn": "arn:aws:iam::account-ID:user/user-name",
        "accountId": "account-ID",
        "userName": "user-name"
      }
    }
  },
  "eventTime": "2016-09-06T17:59:11Z",
  "eventSource": "codebuild.amazonaws.com",
  "eventName": "CreateProject",
  "awsRegion": "region-ID",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "user-agent",
  "requestParameters": {
    "awsActId": "account-ID"
```

```
  },
  "responseElements": {
    "project": {
      "environment": {
        "image": "image-ID",
        "computeType": "BUILD_GENERAL1_SMALL",
        "type": "LINUX_CONTAINER",
        "environmentVariables": []
      },
      "name": "codebuild-demo-project",
      "description": "This is my demo project",
      "arn": "arn:aws:codebuild:region-ID:account-ID:project/codebuild-demo-
project:project-ID",
      "encryptionKey": "arn:aws:kms:region-ID:key-ID",
      "timeoutInMinutes": 10,
      "artifacts": {
        "location": "arn:aws:s3:::codebuild-region-ID-account-ID-output-bucket",
        "type": "S3",
        "packaging": "ZIP",
        "outputName": "MyOutputArtifact.zip"
      },
      "serviceRole": "arn:aws:iam::account-ID:role/CodeBuildServiceRole",
      "lastModified": "Sep 6, 2016 10:59:11 AM",
      "source": {
        "type": "GITHUB",
        "location": "https://github.com/my-repo.git"
      },
      "created": "Sep 6, 2016 10:59:11 AM"
    }
  },
  "requestID": "9d32b228-745b-11e6-98bb-23b67EXAMPLE",
  "eventID": "581f7dd1-8d2e-40b0-aeee-0dbf7EXAMPLE",
  "eventType": "AwsApiCall",
  "recipientAccountId": "account-ID"
}
```

CloudWatch를 사용하여 CodeBuild 빌드 모니터링

Amazon CloudWatch를 사용하여 빌드를 관찰하고, 문제 발생 시 보고하고, 적절한 경우 자동 조치를 취할 수 있습니다. 두 수준에서 빌드를 모니터링할 수 있습니다.

프로젝트 수준

이러한 지표는 지정된 프로젝트의 모든 빌드에 대한 것입니다. 프로젝트의 지표를 보려면 CloudWatch의 차원에 `ProjectName`을 지정합니다.

AWS 계정 수준

이러한 지표는 한 계정의 모든 빌드에 대한 것입니다. AWS 계정 수준에서 지표를 보려면 CloudWatch에 차원을 입력하지 마세요. 빌드 리소스 사용률 지표는 AWS 계정 수준에서 사용할 수 없습니다.

CloudWatch 지표는 일정 기간 동안의 빌드의 양상을 보여 줍니다. 예를 들면, 다음을 모니터링할 수 있습니다.

- 시간 경과에 따라 빌드 프로젝트 또는 AWS 계정에서 시도된 빌드 수입니다.
- 시간 경과에 따른 빌드 프로젝트 또는 AWS 계정에서 성공한 빌드 수입니다.
- 시간 경과에 따라 빌드 프로젝트 또는 AWS 계정에서 실패한 빌드 수입니다.
- CodeBuild가 시간 경과에 따라 빌드 프로젝트 또는 AWS 계정에서 빌드를 실행하는 데 소요한 시간입니다.
- 빌드 또는 전체 빌드 프로젝트의 빌드 리소스 사용률입니다. 빌드 리소스 사용률 지표에는 CPU, 메모리, 스토리지 사용률과 같은 지표가 포함됩니다.

자세한 내용은 CodeBuild 지표 보기 단원을 참조하십시오.

CodeBuild CloudWatch 지표

AWS 계정 또는 빌드 프로젝트별로 다음 지표를 추적할 수 있습니다. CloudWatch를 CodeBuild와 함께 사용하는 자세한 방법은 CloudWatch를 사용하여 CodeBuild 빌드 모니터링 섹션을 참조하세요.

BuildDuration

빌드의 `BUILD` 단계 지속 시간을 측정합니다.

단위: 초

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

빌드

트리거된 빌드 수를 측정합니다.

단위: 개

유효한 CloudWatch 통계: 합계

DownloadSourceDuration

빌드의 DOWNLOAD_SOURCE 단계 지속 시간을 측정합니다.

단위: 초

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

지속 시간

일정 기간 동안 모든 빌드 시간을 측정합니다.

단위: 초

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

FailedBuilds

클라이언트 오류 또는 시간 초과로 인해 실패한 빌드 수를 측정합니다.

단위: 개

유효한 CloudWatch 통계: 합계

FinalizingDuration

빌드의 FINALIZING 단계 지속 시간을 측정합니다.

단위: 초

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

InstallDuration

빌드의 INSTALL 단계 지속 시간을 측정합니다.

단위: 초

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

PostBuildDuration

빌드의 POST_BUILD 단계 지속 시간을 측정합니다.

단위: 초

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

PreBuildDuration

빌드의 PRE_BUILD 단계 지속 시간을 측정합니다.

단위: 초

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

ProvisioningDuration

빌드의 PROVISIONING 단계 지속 시간을 측정합니다.

단위: 초

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

QueuedDuration

빌드의 QUEUED 단계 지속 시간을 측정합니다.

단위: 초

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

SubmittedDuration

빌드의 SUBMITTED 단계 지속 시간을 측정합니다.

단위: 초

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

SucceededBuilds

성공한 빌드 수를 측정합니다.

단위: 개

유효한 CloudWatch 통계: 합계

UploadArtifactsDuration

빌드의 UPLOAD_ARTIFACTS 단계 지속 시간을 측정합니다.

단위: 초

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

CodeBuild CloudWatch 리소스 사용률 지표

> **Note**
>
> CodeBuild 리소스 사용률 지표는 다음 리전에서만 사용할 수 있습니다.
>
> - 아시아 태평양(도쿄) 리전
> - Asia Pacific (Seoul) Region
> - Asia Pacific (Mumbai) Region
> - Asia Pacific (Singapore) Region
> - Asia Pacific (Sydney) Region
> - 캐나다(중부) 리전
> - Europe (Frankfurt) Region
> - 유럽(아일랜드) 리전
> - Europe (London) Region
> - Europe (Paris) Region
> - South America (São Paulo) Region
> - US East (N. Virginia) Region
> - US East (Ohio) Region
> - US West (N. California) Region
> - 미국 서부(오리건) 리전

다음 리소스 사용률 지표를 추적할 수 있습니다. CloudWatch를 CodeBuild와 함께 사용하는 자세한 방법은 CloudWatch를 사용하여 CodeBuild 빌드 모니터링 섹션을 참조하세요.

CpuUtilized

빌드 컨테이너에서 사용하는 할당된 처리의 CPU 단위 수입니다.

단위: CPU 단위

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

CPUUtilizedPercent

빌드 컨테이너에서 사용된 할당된 처리의 비율입니다.

단위: 백분율

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

MemoryUtilized

빌드 컨테이너가 사용된 메모리 크기(메가바이트)입니다.

단위: 메가바이트

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

MemoryUtilizedPercent

빌드 컨테이너에서 사용된 할당된 메모리의 비율입니다.

단위: 백분율

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

StorageReadBytes

빌드 컨테이너에서 사용된 스토리지 읽기 속도입니다.

단위: 바이트/초

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

StorageWriteBytes

빌드 컨테이너에서 사용된 스토리지 쓰기 속도입니다.

단위: 바이트/초

유효한 CloudWatch 통계: 평균(권장), 최대, 최소

CodeBuild CloudWatch 차원

CodeBuild는 다음과 같은 CloudWatch 지표 차원을 제공합니다. 이 중 아무 것도 지정하지 않으면 지표는 현재 AWS 계정에 대한 것입니다.

BuildId, BuildNumber, ProjectName

빌드 식별자, 빌드 번호, 프로젝트 이름에 대한 지표가 제공됩니다.

ProjectName

프로젝트 이름에 대한 지표가 제공됩니다.

CodeBuild CloudWatch 경보

CloudWatch 콘솔을 사용하여 CodeBuild 지표를 기준으로 경보를 생성함으로써 빌드에 문제가 생길 경우 조치를 취할 수 있습니다. 경보에 가장 유용한 두 가지 지표는 다음 글머리 기호에 설명되어 있습니다. CloudWatch를 CodeBuild와 함께 사용하는 자세한 방법은 CloudWatch를 사용하여 CodeBuild 빌드 모니터링 섹션을 참조하세요.

- `FailedBuild`. 지정한 시간(초) 내에 특정 수의 실패한 빌드 수가 감지될 경우 트리거되는 경보를 만들 수 있습니다. CloudWatch에서 경보를 발생시킬 실패 빌드 수와 시간(초)을 지정합니다.
- `Duration`. 빌드가 예상한 것보다 오래 걸릴 경우 트리거되는 경보를 만들 수 있습니다. 빌드가 시작된 후 완료되기까지 걸리는 시간(초)을 지정하여 이 시간을 초과할 경우 경보를 생성하도록 지정합니다.

CodeBuild 지표에 대한 경보를 만드는 방법은 CloudWatch 경보를 사용한 CodeBuild 빌드 모니터링 섹션을 참조하세요. 경보에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch 경보 생성을 참조하세요.

CodeBuild 지표 보기

AWS CodeBuild 는 사용자를 대신하여 함수를 모니터링하고 Amazon CloudWatch를 통해 지표를 보고합니다. 이러한 지표에는 총 빌드, 실패한 빌드, 성공한 빌드, 빌드 기간 등이 포함됩니다.

CodeBuild 콘솔 또는 CloudWatch 콘솔을 사용하여 CodeBuild에 대한 지표를 모니터링할 수 있습니다. 다음 절차에서는 지표에 보는 방법을 보여 줍니다.

주제

- 빌드 지표 보기(CodeBuild 콘솔)
- 빌드 지표 보기(Amazon CloudWatch 콘솔)

빌드 지표 보기(CodeBuild 콘솔)

> **Note**
>
> CodeBuild 콘솔에서는 지표를 표시하는 데 사용되는 지표나 그래프를 사용자 지정할 수 없습니다. 디스플레이를 사용자 지정하려면 Amazon CloudWatch 콘솔을 사용하여 빌드 지표를 확인합니다.

계정 수준 지표

AWS 계정 수준 지표를 보려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 계정 지표를 선택합니다.

프로젝트 수준 지표

프로젝트 수준 지표를 보려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 프로젝트 빌드를 선택합니다.
3. 빌드 프로젝트 목록의 이름 열에서 지표를 보려는 프로젝트를 선택합니다.
4. 지표 탭을 선택합니다.

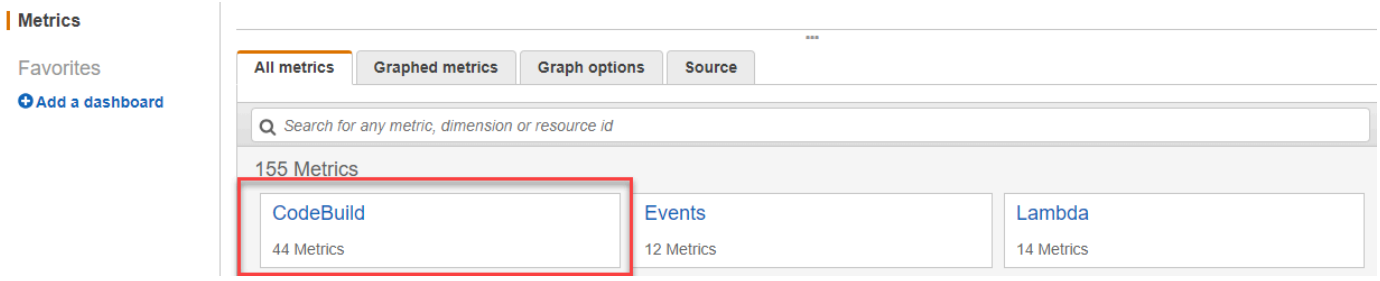
빌드 지표 보기(Amazon CloudWatch 콘솔)

CloudWatch 콘솔에서 지표 및 지표를 표시하는 데 사용되는 그래프를 사용자 지정할 수 있습니다.

계정 수준 지표

계정 수준 지표를 보려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/cloudwatch/ CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표(Metrics)를 선택합니다.
3. 모든 지표 탭에서 CodeBuild를 선택합니다.



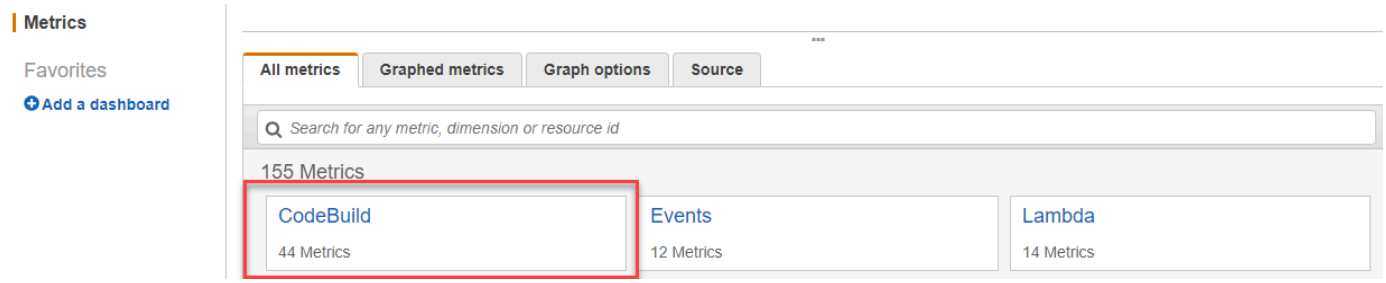
4. 계정 지표를 선택합니다.

5. 하나 이상의 프로젝트 및 지표를 선택합니다. 각 프로젝트에 대해 SucceededBuilds, FailedBuilds, Builds 및 Duration 지표를 선택할 수 있습니다. 선택한 모든 프로젝트 및 지표 조합이 페이지의 그래프에 표시됩니다.

프로젝트 수준 지표

프로젝트 수준 지표를 보려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/cloudwatch/ CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표(Metrics)를 선택합니다.
3. 모든 지표 탭에서 CodeBuild를 선택합니다.



4. 프로젝트별을 선택합니다.
5. 하나 이상의 프로젝트 및 지표 조합을 선택합니다. 각 프로젝트에 대해 SucceededBuilds, FailedBuilds, Builds 및 Duration 지표를 선택할 수 있습니다. 선택한 모든 프로젝트 및 지표 조합이 페이지의 그래프에 표시됩니다.
6. (선택 사항) 지표 및 그래프를 사용자 지정할 수 있습니다. 예를 들어 통계 열의 드롭다운 목록에서 표시할 다른 통계를 선택할 수 있습니다. 또는 기간 열의 드롭다운 메뉴에서 지표를 모니터링하는 데 사용할 다른 기간을 선택할 수 있습니다.

   자세한 내용은 Amazon CloudWatch 사용 설명서의 그래프 지표 및 사용 가능한 지표 보기를 참조하세요.

CodeBuild 리소스 사용률 지표 보기

AWS CodeBuild 는 사용자를 대신하여 빌드 리소스 사용률을 모니터링하고 Amazon CloudWatch를 통해 지표를 보고합니다. 여기에는 CPU, 메모리, 스토리지 사용률과 같은 지표가 포함됩니다.

> **Note**
>
> CodeBuild 리소스 사용률 지표는 1분 넘게 실행되는 빌드에 대해서만 기록됩니다.

CodeBuild 콘솔 또는 CloudWatch 콘솔을 사용하여 CodeBuild에 대한 리소스 사용률 지표를 모니터링할 수 있습니다.

> **Note**
>
> CodeBuild 리소스 사용률 지표는 다음 리전에서만 사용할 수 있습니다.
>
> - 아시아 태평양(도쿄) 리전
> - Asia Pacific (Seoul) Region
> - Asia Pacific (Mumbai) Region
> - Asia Pacific (Singapore) Region
> - Asia Pacific (Sydney) Region
> - 캐나다(중부) 리전
> - Europe (Frankfurt) Region
> - 유럽(아일랜드) 리전
> - Europe (London) Region
> - Europe (Paris) Region
> - South America (São Paulo) Region
> - US East (N. Virginia) Region
> - US East (Ohio) Region
> - US West (N. California) Region
> - 미국 서부(오리건) 리전

다음 절차에서는 리소스 사용률 지표에 액세스하는 방법을 보여 줍니다.

주제

- 리소스 사용률 지표 액세스(CodeBuild 콘솔)
- 리소스 사용률 지표 액세스(Amazon CloudWatch 콘솔)

리소스 사용률 지표 액세스(CodeBuild 콘솔)

> **Note**
>
> CodeBuild 콘솔에서는 지표를 표시하는 데 사용되는 지표나 그래프를 사용자 지정할 수 없습니다. 디스플레이를 사용자 지정하려면 Amazon CloudWatch 콘솔을 사용하여 빌드 지표를 확인합니다.

프로젝트 수준 리소스 사용률 지표

프로젝트 수준 리소스 사용률 지표에 액세스하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 프로젝트 빌드를 선택합니다.
3. 빌드 프로젝트 목록의 이름 열에서 사용률 지표를 보려는 프로젝트를 선택합니다.
4. 지표 탭을 선택합니다. 리소스 사용률 지표는 리소스 사용률 지표 섹션에 표시됩니다.
5. CloudWatch 콘솔에서 프로젝트 수준 리소스 사용률 지표를 보려면 리소스 사용률 지표 섹션에서 CloudWatch에서 보기를 선택합니다.

빌드 수준의 리소스 사용률 지표

빌드 수준 리소스 사용률 지표에 액세스하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codesuite/codebuild/home AWS CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 [Build history]를 선택합니다.
3. 빌드 목록의 빌드 실행 열에서 사용률 지표를 보려는 빌드를 선택합니다.
4. 리소스 사용률 탭을 선택합니다.
5. CloudWatch 콘솔에서 빌드 수준 리소스 사용률 지표를 보려면 리소스 사용률 지표 섹션에서 CloudWatch에서 보기를 선택합니다.

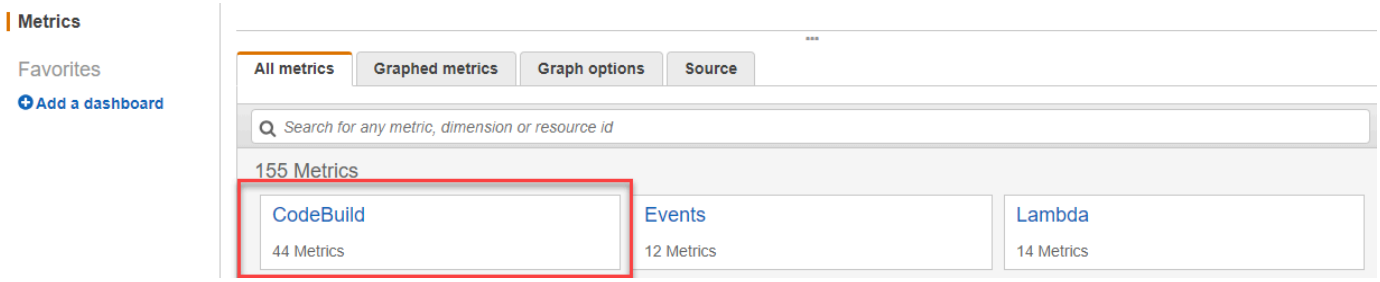
리소스 사용률 지표 액세스(Amazon CloudWatch 콘솔)

Amazon CloudWatch 콘솔을 사용하여 CodeBuild 리소스 사용률 지표에 액세스할 수 있습니다.

프로젝트 수준 리소스 사용률 지표

프로젝트 수준 리소스 사용률 지표에 액세스하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/cloudwatch/ CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표(Metrics)를 선택합니다.
3. 모든 지표 탭에서 CodeBuild를 선택합니다.



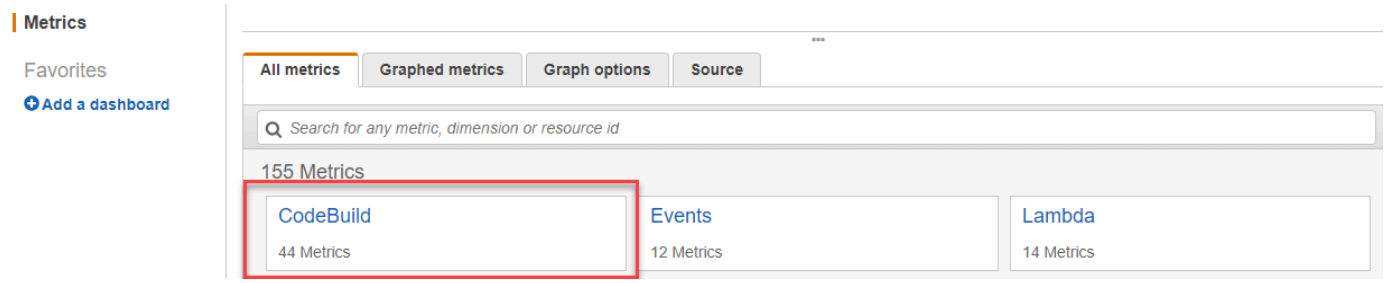
4. 프로젝트별을 선택합니다.
5. 그래프에 추가할 하나 이상의 프로젝트와 지표 조합을 선택합니다. 선택한 모든 프로젝트 및 지표 조합이 페이지의 그래프에 표시됩니다.
6. (선택 사항) 그래프로 표시된 지표 탭에서 지표와 그래프를 사용자 지정할 수 있습니다. 예를 들어 통계 열의 드롭다운 목록에서 표시할 다른 통계를 선택할 수 있습니다. 또는 기간 열의 드롭다운 메뉴에서 지표를 모니터링하는 데 사용할 다른 기간을 선택할 수 있습니다.

   자세한 내용은 Amazon CloudWatch 사용 설명서의 그래프 지표 및 사용 가능한 지표 보기를 참조하세요.

빌드 수준의 리소스 사용률 지표

빌드 수준 리소스 사용률 지표에 액세스하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/cloudwatch/ CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표(Metrics)를 선택합니다.
3. 모든 지표 탭에서 CodeBuild를 선택합니다.



4. BuildId, BuildNumber, ProjectName을 선택합니다.
5. 그래프에 추가할 하나 이상의 빌드와 지표 조합을 선택합니다. 선택한 모든 빌드 및 지표 조합이 페이지의 그래프에 표시됩니다.
6. (선택 사항) 그래프로 표시된 지표 탭에서 지표와 그래프를 사용자 지정할 수 있습니다. 예를 들어 통계 열의 드롭다운 목록에서 표시할 다른 통계를 선택할 수 있습니다. 또는 기간 열의 드롭다운 메뉴에서 지표를 모니터링하는 데 사용할 다른 기간을 선택할 수 있습니다.

   자세한 내용은 Amazon CloudWatch 사용 설명서의 그래프 지표 및 사용 가능한 지표 보기를 참조하세요.

CloudWatch 경보를 사용한 CodeBuild 빌드 모니터링

빌드에 대한 CloudWatch 경보를 생성할 수 있습니다. 경보는 지정한 기간에 단일 메트릭을 감시하고 여러 기간에 지정된 임계값에 대한 메트릭 값을 기준으로 작업을 하나 이상 수행합니다. 기본 CloudWatch 경보 기능을 사용하면 임계값 초과 시 CloudWatch에서 지원하는 모든 작업을 지정할 수 있습니다. 예를 들어, 15분 이내에 계정의 빌드가 4개 이상 실패할 경우 Amazon SNS 알림이 전송되도록 지정할 수 있습니다.

CodeBuild 지표에 대한 CloudWatch 경보를 생성하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/cloudwatch/ CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 경보(Alarms)를 선택하세요.
3. 경보 생성을 선택합니다.
4. 범주별 CloudWatch 지표에서 CodeBuild 지표를 선택합니다. 프로젝트 수준 지표만 사용하려는 경우 프로젝트별을 선택합니다. 계정 수준 지표만 사용하려는 경우 계정 지표를 선택합니다.
5. 경보 생성에서 아직 선택하지 않은 경우 지표 선택을 선택합니다.
6. 경보를 생성하려는 지표를 선택합니다. 옵션은 프로젝트별 또는 계정 지표입니다.

7. 다음 또는 경보 정의를 선택한 다음, 경보를 생성합니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch 경보 생성을 참조하세요. 경보가 트리거될 때 Amazon SNS 알림을 설정하는 방법에 대한 자세한 내용은 Amazon SNS 개발자 안내서의 Amazon SNS 알림 설정을 참조하세요.
8. 경보 생성을 선택합니다.

의 보안 AWS CodeBuild

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 이를 클라우드의 보안과 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS 는에서 AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다 AWS 클라우드. AWS 또한는 안전하게 사용할 수 있는 서비스를 제공합니다. 서드 파티 감사자는 정기적으로 AWS 규정 준수 프로그램의 일환으로 보안 효과를 테스트하고 검증합니다. 에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 AWS 규정 준수 프로그램 제공 범위 내 서비스를 AWS CodeBuild참조하세요.
- 클라우드의 보안 - 사용자의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 귀하는 데이터의 민감도, 회사 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 CodeBuild를 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 주제에서는 보안 및 규정 준수 목표에 맞게 CodeBuild를 구성하는 방법을 보여줍니다. 또한 CodeBuild 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법을 알아봅니다.

주제

- 의 데이터 보호 AWS CodeBuild
- 의 자격 증명 및 액세스 관리 AWS CodeBuild
- 에 대한 규정 준수 검증 AWS CodeBuild
- 의 복원력 AWS CodeBuild
- 의 인프라 보안 AWS CodeBuild
- CodeBuild에서 소스 공급자에 액세스
- 교차 서비스 혼동된 대리인 방지

의 데이터 보호 AWS CodeBuild

AWS 공동 책임 모델 은 AWS CodeBuild의 데이터 보호에 적용됩니다. 이 모델에 설명된 대로 AWS 는 모든를 실행하는 글로벌 인프라를 보호할 책임이 있습니다 AWS 클라우드. 사용자는 이 인프라에 호스

팅되는 콘텐츠에 대한 통제 권한을 유지할 책임이 있습니다. 사용하는 AWS 서비스 의 보안 구성과 관리 태스크에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 일반 데이터 보호 규정(GDPR) 센터를 참조하세요.

데이터 보호를 위해 자격 증명을 보호하고 AWS 계정 AWS IAM Identity Center 또는 AWS Identity and Access Management (IAM)를 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용합니다.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다 AWS CloudTrail. CloudTrail 추적을 사용하여 AWS 활동을 캡처하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 추적 작업을 참조하세요.
- 내부의 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용합니다 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용합니다.
- 명령줄 인터페이스 또는 API를 AWS 통해에 액세스할 때 FIPS 140-3 검증 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 연방 정보 처리 표준(FIPS) 140-3을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 형식 텍스트 필드에 입력하지 않는 것이 좋습니다. 여기에는 CodeBuild 또는 기타 AWS 서비스 에서 콘솔 AWS CLI, API 또는 AWS SDKs를 사용하여 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 보안 인증 정보를 URL에 포함시켜서는 안 됩니다.

중요한 정보를 보호하기 위해 CodeBuild 로그에 다음 항목이 숨겨져 있습니다.

- CodeBuild 프로젝트 환경 변수 또는 buildspec `env/parameter-store` 섹션의 Parameter Store를 사용하여 지정된 문자열. 자세한 내용은 Amazon EC2 Systems Manager 사용 설명서의 Systems Manager Parameter Store 및 Systems Manager Parameter Store 콘솔 연습을 참조하세요.
- AWS Secrets Manager CodeBuild 프로젝트 환경 변수 또는 buildspec `env/secrets-manager` 섹션에서를 사용하여 지정된 문자열입니다. 자세한 내용은 키 관리 섹션을 참조하세요.

> **Note**
>
> 마스킹은 Amazon EC2 Systems Manager 파라미터 스토어 또는에 저장된 정확한 값과 일치합니다 AWS Secrets Manager. 로그에 표시되기 전에 빌드 명령이 보안 암호를 변환하는 경우 결과는 마스킹 처리되지 않은 다른 문자열입니다. 예를 들어 Base64로 인코딩하거나, 되돌리거나, 사례를 변경하거나, 다른 텍스트와 연결하면 빌드 로그에 표시될 수 있는 다른 값이 생성됩니다.
>
> 값을 마스킹된 상태로 유지하려면 빌드에서 사용하는 형식으로 저장합니다. 예를 들어, 빌드에 Base64-encoded 값이 필요한 경우 Base64-encoded 문자열을 AWS Secrets Manager 또는 Amazon EC2 Systems Manager Parameter Store에 직접 저장합니다. 이렇게 하면 로깅된 값이 저장된 값과 일치합니다.

데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

주제

- 데이터 암호화
- 키 관리
- 트래픽 개인 정보 보호

데이터 암호화

암호화는 CodeBuild 보안의 중요한 부분입니다. 일부 암호화(예: 전송 중인 데이터 암호화)는 기본으로 제공되며 어떠한 것도 필요하지 않습니다. 기타 암호화(예: 유휴 상태의 데이터 암호화)는 프로젝트나 빌드 생성 시 구성할 수 있습니다.

- 유휴 데이터 암호화 - 캐시, 로그, 내보낸 원시 테스트 보고서 데이터 파일, 빌드 결과와 같은 빌드 아티팩트는 기본적으로를 사용하여 암호화됩니다 AWS 관리형 키. 이러한 KMS 키를 사용하지 않으려면 고객 관리형 키를 생성하고 구성해야 합니다. KMS 키 생성 및 AWS 키 관리 서비스 개념에 대한 자세한 내용은AWS Key Management Service 사용 설명서를 참조하세요.
  - CodeBuild가 빌드 출력 아티팩트를 암호화하는 데 사용하는 AWS KMS 키의 식별자를 `CODEBUILD_KMS_KEY_ID` 환경 변수에 저장할 수 있습니다. 자세한 내용은 빌드 환경의 환경 변수 섹션을 참조하세요.
  - 빌드 프로젝트 생성 시 고객 관리형 키를 지정할 수 있습니다. 자세한 내용은 Set the Encryption Key Using the Console 및 CLI를 사용하여 암호화 키 설정을 참조하세요.

빌드 플릿의 Amazon Elastic Block Store 볼륨은 기본적으로를 사용하여 암호화됩니다 AWS 관리형 키.

- 전송 중 데이터 암호화 - 고객과 CodeBuild 간 및 CodeBuild와 다운스트림 종속성 간의 모든 통신은 서명 버전 4 서명 프로세스를 사용하여 서명된 TLS 연결을 사용하여 보호됩니다. 모든 CodeBuild 엔드포인트는에서 관리하는 SHA-256 인증서를 사용합니다 AWS Private Certificate Authority. 자세한 내용은 서명 버전 4 서명 프로세스 및 ACM PCA란 무엇입니까를 참조하십시오.
- 빌드 아티팩트 암호화 - 빌드 프로젝트와 관련된 CodeBuild 서비스 역할이 빌드 출력 아티팩트를 암호화하려면 KMS 키에 대한 액세스 권한이 필요합니다. 기본적으로 CodeBuild는 AWS 계정에서 Amazon S3 AWS 관리형 키 용를 사용합니다. 이 AWS 관리형 키를 사용하지 않으려면 고객 관리형 키를 생성 및 구성해야 합니다. 자세한 내용은 AWS KMS 개발자 안내서의 빌드 출력 암호화 및 키 생성을 참조하세요.

키 관리

암호화를 통해 콘텐츠의 무단 사용을 방지할 수 있습니다. 암호화 키를에 저장 AWS Secrets Manager 한 다음 빌드 프로젝트와 연결된 CodeBuild 서비스 역할에 Secrets Manager 계정에서 암호화 키를 가져올 수 있는 권한을 부여합니다. 자세한 내용은 고객 관리형 키를 사용하여 빌드 출력 암호화, 에서 빌드 프로젝트 생성 AWS CodeBuild, 수동으로 AWS CodeBuild 빌드 실행 및 자습서: 보안 암호 저장 및 검색을 참조하십시오.

빌드 명령에서 `CODEBUILD_KMS_KEY_ID` 환경 변수를 사용하여 AWS KMS 키 식별자를 가져옵니다. 자세한 내용은 빌드 환경의 환경 변수 섹션을 참조하세요.

Secrets Manager를 사용하여 런타임 환경에 사용된 도커 이미지를 저장하는 프라이빗 레지스트리의 보안 인증을 보호할 수 있습니다. 자세한 내용은 CodeBuild용 AWS Secrets Manager 샘플이 포함된 프라이빗 레지스트리 섹션을 참조하세요.

트래픽 개인 정보 보호

인터페이스 VPC 엔드포인트를 사용하도록 CodeBuild를 구성하여 빌드의 보안을 향상할 수 있습니다. 이를 위해 인터넷 게이트웨이, NAT 디바이스 또는 가상 프라이빗 게이트웨이가 필요 없습니다. 또한 PrivateLink를 구성하는 것이 필수는 아니지만 구성하는 것이 좋습니다. 자세한 내용은 VPC 엔드포인트 사용 섹션을 참조하세요. PrivateLink 및 VPC 엔드포인트에 대한 자세한 내용은 AWS PrivateLink 및 PrivateLink를 통한 AWS 서비스 액세스를 참조하세요.

의 자격 증명 및 액세스 관리 AWS CodeBuild

에 액세스하려면 자격 증명이 AWS CodeBuild 필요합니다. 이러한 자격 증명에는 S3 버킷에 빌드 아티팩트를 저장 및 검색하고 빌드에 대한 Amazon CloudWatch Logs를 보는 등 AWS 리소스에 액세스할 수 있는 권한이 있어야 합니다. 다음 섹션에서는 AWS Identity and Access Management(IAM) 및 CodeBuild를 사용하여 리소스에 대한 액세스를 보호할 수 있는 방법에 대한 세부 정보를 제공합니다.

AWS CodeBuild 리소스에 대한 액세스 권한 관리 개요

모든 AWS 리소스는 AWS 계정이 소유하며, 리소스를 생성하거나 액세스할 수 있는 권한은 권한 정책에 의해 관리됩니다. 계정 관리자는 IAM ID(사용자, 그룹 및 역할)에 권한 정책을 연결할 수 있습니다.

> **Note**
>
> 계정 관리자 또는 관리자 사용자는 관리자 권한이 있는 사용자입니다. 자세한 설명은 IAM 사용자 가이드의 IAM 모범 사례 섹션을 참조하십시오.

권한을 부여할 때는 권한을 부여 받을 사용자, 사용자가 액세스할 수 있는 리소스 및 해당 리소스에 수행할 수 있는 작업을 결정합니다.

주제

- AWS CodeBuild 리소스 및 작업
- 리소스 소유권 이해
- 리소스 액세스 관리
- 정책 요소 지정: 작업, 효과, 보안 주체

AWS CodeBuild 리소스 및 작업

에서 AWS CodeBuild기본 리소스는 빌드 프로젝트입니다. 정책에서 Amazon 리소스 이름(ARN)을 사용하여 정책이 적용되는 리소스를 식별합니다. 빌드도 리소스이며 빌드에는 이와 연결된 ARN이 들어 있습니다. 자세한 내용은의 Amazon 리소스 이름(ARN) 및 AWS 서비스 네임스페이스를 참조하세요Amazon Web Services 일반 참조.

리소스 유형	ARN 형식
빌드 프로젝트	arn:aws:codebuild: *region-ID* :*account-ID* :project/ *project-name*
빌드	arn:aws:codebuild: *region-ID* :*account-ID* :build/*build-ID*
보고서 그룹	arn:aws:codebuild: *region-ID* :*account-ID* :report-group/ *report-group-name*
Report	arn:aws:codebuild: *region-ID* :*account-ID* :report/*report-ID*
플릿	arn:aws:codebuild: *region-ID* :*account-ID* :fleet/*fleet-ID*
모든 CodeBuild 리소스	arn:aws:codebuild:*
지정된 AWS 리전에서 지정된 계정이 소유한 모든 CodeBuild 리소스	arn:aws:codebuild: *region-ID* :*account-ID* :*

Important

예약 용량 기능을 사용할 때 소스 파일, Docker 계층, 빌드스펙에 지정된 캐시된 디렉터리를 포함하여 플릿 인스턴스에 캐시된 데이터를 동일한 계정 내의 다른 프로젝트에서 액세스할 수 있습니다. 이는 의도적으로 설계된 것이며 동일한 계정 내의 프로젝트가 플릿 인스턴스를 공유할 수 있도록 허용합니다.

Note

대부분의 AWS 서비스는 콜론(:) 또는 슬래시(/)를 ARNs. 그러나 CodeBuild는 리소스 패턴 및 규칙에서 정확한 일치를 사용합니다. 따라서 이벤트 패턴을 만들 때 리소스에서 ARN 구문이 일치하도록 정확한 문자를 사용해야 합니다.

예를 들어, 명령문에서 다음과 같이 ARN을 사용하여 특정 빌드 프로젝트(*myBuildProject*)를 지정할 수 있습니다.

```
"Resource": "arn:aws:codebuild:us-east-2:123456789012:project/myBuildProject"
```

모든 리소스를 지정해야 하거나, API 작업이 ARN을 지원하지 않는 경우 다음과 같이 Resource 요소에 와일드카드 문자(*)를 사용합니다.

```
"Resource": "*"
```

일부 CodeBuild API 작업에서는 여러 리소스 사용을 허용합니다(예: BatchGetProjects). 명령문 하나에 여러 리소스를 지정하려면 다음과 같이 각 ARN을 쉼표로 구분합니다.

```
"Resource": [
  "arn:aws:codebuild:us-east-2:123456789012:project/myBuildProject",
  "arn:aws:codebuild:us-east-2:123456789012:project/myOtherBuildProject"
]
```

CodeBuild는 CodeBuild 리소스를 처리하기 위한 작업 모음을 제공합니다. 목록을 보려면 AWS CodeBuild 권한 참조 섹션을 참조하세요.

리소스 소유권 이해

AWS 계정은 누가 리소스를 생성했는지에 관계없이 계정에 생성된 리소스를 소유합니다. 특히 리소스 소유자는 리소스 생성 요청을 인증하는 AWS 보안 주체 엔터티(즉, 루트 계정, 사용자 또는 IAM 역할)의 계정입니다. 다음 예제에서는 이러한 작동 방법을 설명합니다.

- AWS 계정의 루트 계정 자격 증명을 사용하여 규칙을 생성하는 경우 AWS 계정은 CodeBuild 리소스의 소유자입니다.
- AWS 계정에서 사용자를 생성하고 해당 사용자에게 CodeBuild 리소스를 생성할 수 있는 권한을 부여하는 경우 사용자는 CodeBuild 리소스를 생성할 수 있습니다. 하지만 사용자가 속한 AWS 계정이 CodeBuild 리소스를 소유합니다.
- AWS 계정에서 CodeBuild 리소스를 생성할 권한이 있는 IAM 역할을 생성하는 경우 역할을 수임할 수 있는 사람은 누구나 CodeBuild 리소스를 생성할 수 있습니다. 역할이 속한 AWS 계정이 CodeBuild 리소스를 소유합니다.

리소스 액세스 관리

권한 정책은 누가 어떤 리소스에 액세스 할 수 있는지를 나타냅니다.

> **Note**
>
> 이 섹션에서는 AWS CodeBuild에서 IAM을 사용하는 방법에 대해 설명하며, IAM 서비스에 대한 자세한 내용은 다루지 않습니다. IAM 설명서의 전체 내용은 IAM 사용자 안내서의 IAM이란?을 참조하세요. IAM 정책 구문과 설명에 대한 자세한 내용은 IAM 사용 설명서의 AWS IAM 정책 참조 섹션을 참조하세요.

IAM 보안 인증에 연결된 정책을 보안 인증 기반 정책(IAM 정책)이라고 합니다. 리소스에 연결된 정책을 리소스 기반 정책이라고 합니다. CodeBuild는 계정 리소스 공유를 위해 자격 증명 기반 정책과 특정 읽기 전용 API에 대한 리소스 기반 정책을 지원합니다.

S3 버킷에 대한 보안 액세스

IAM 역할에 다음 권한을 포함시켜 CodeBuild 프로젝트와 연결된 S3 버킷이 사용자 또는 사용자가 신뢰할 수 있는 사람의 소유인지 확인할 것을 적극 권장합니다. 이러한 권한은 AWS 관리형 정책 및 역할에 포함되지 않습니다. 사용자가 직접 추가해야 합니다.

- `s3:GetBucketAcl`
- `s3:GetBucketLocation`

프로젝트에서 사용되는 S3 버킷 소유자가 변경될 경우에는 사용자가 해당 버킷을 아직도 소유하고 있는지 확인하고 아닐 경우에는 사용자의 IAM 역할에 권한을 업데이트해야 합니다. 자세한 내용은 사용자가 CodeBuild와 상호 작용하도록 허용 및 CodeBuild가 다른 AWS 서비스와 상호 작용하도록 허용 섹션을 참조하세요.

정책 요소 지정: 작업, 효과, 보안 주체

각 AWS CodeBuild 리소스에 대해 서비스는 API 작업 세트를 정의합니다. 이러한 API 작업에 대한 권한을 부여하기 위해 CodeBuild에서는 정책에서 지정할 수 있는 작업을 정의합니다. 일부 API 작업에서는 API 작업을 수행하기 위해 복수의 작업에 대한 권한이 필요할 수 있습니다. 자세한 내용은 AWS CodeBuild 리소스 및 작업 및 AWS CodeBuild 권한 참조 섹션을 참조하세요.

다음은 기본 정책 요소입니다.

- 리소스 – Amazon 리소스 이름(ARN)을 사용하여 정책을 적용할 리소스를 식별합니다.
- 작업 – 작업 키워드를 사용하여 허용 또는 거부할 리소스 작업을 식별합니다. 예를 들어, `codebuild:CreateProject` 권한은 사용자에게 `CreateProject` 작업 수행 권한을 제공합니다.
- 효과 – 사용자가 작업을 요청하는 경우 효과(허용 또는 거부)를 지정합니다. 명시적으로 리소스에 대한 액세스 권한을 부여(허용)하지 않는 경우, 액세스는 묵시적으로 거부됩니다. 리소스에 대한 액세스를 명시적으로 거부할 수도 있습니다. 다른 정책에서 액세스 권한을 부여하더라도 사용자가 해당 리소스에 액세스할 수 없도록 하려고 할 때 이러한 작업을 수행할 수 있습니다.
- 보안 주체 – 자격 증명 기반 정책(IAM 정책)에서 정책이 연결되는 사용자는 암시적인 보안 주체입니다. 리소스 기반 정책의 경우 사용자, 계정, 서비스 또는 권한의 수신자인 기타 엔터티를 지정합니다.

IAM 정책 구문과 설명에 대한 자세한 내용은 IAM 사용자 안내서의 AWS IAM 정책 참조를 참조하세요.

모든 CodeBuild API 작업과 해당 작업이 적용되는 리소스를 보여 주는 표는 AWS CodeBuild 권한 참조 섹션을 참조하세요.

에 자격 증명 기반 정책 사용 AWS CodeBuild

이 주제에서는 자격 증명 기반 정책의 예를 통해 계정 관리자가 IAM 자격 증명(사용자, 그룹, 역할)에 권한 정책을 연결해 AWS CodeBuild 리소스에 대한 작업 수행 권한을 부여하는 방법을 보여줍니다.

> **Important**
>
> CodeBuild 리소스에 대한 액세스 관리를 위해 제공되는 기본 개념과 옵션 설명에 대한 소개 주제 부분을 먼저 읽어 보세요. 자세한 내용은 AWS CodeBuild 리소스에 대한 액세스 권한 관리 개요 섹션을 참조하세요.

주제

- AWS CodeBuild 콘솔을 사용하는 데 필요한 권한
- 가 Amazon Elastic Container Registry AWS CodeBuild 에 연결하는 데 필요한 권한
- AWS CodeBuild 콘솔이 소스 공급자에 연결하는 데 필요한 권한
- AWS 에 대한 관리형(미리 정의된) 정책 AWS CodeBuild
- CodeBuild 관리형 정책 및 알림
- AWS 관리형 정책에 대한 CodeBuild 업데이트
- 고객 관리형 정책 예제

다음은 us-east-2라는 이름으로 시작하는 빌드 프로젝트에 대해 사용자가 123456789012 계정의 my 리전에 있는 빌드 프로젝트에 대해서만 정보를 가져오도록 허용하는 권한 정책의 예입니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetProjects",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:project/my*"
    }
  ]
}
```

AWS CodeBuild 콘솔을 사용하는 데 필요한 권한

AWS CodeBuild 콘솔을 사용하는 사용자에게는 AWS 계정의 다른 AWS 리소스를 설명할 수 있는 최소 권한 집합이 있어야 합니다. 사용자에게는 다음 서비스에 대한 권한이 있어야 합니다.

- AWS CodeBuild
- Amazon CloudWatch
- CodeCommit(AWS CodeCommit 리포지토리에 소스 코드를 저장하는 경우)
- Amazon Elastic Container Registry(Amazon ECR)(Amazon ECR 리포지토리의 도커 이미지를 사용하는 빌드 환경을 사용하는 경우)

> **Note**
>
> 2022년 7월 26일부로 기본 IAM 정책이 업데이트되었습니다. 자세한 내용은 가 Amazon Elastic Container Registry AWS CodeBuild 에 연결하는 데 필요한 권한 섹션을 참조하세요.

- Amazon Elastic Container Service(Amazon ECS)(Amazon ECR 리포지토리의 도커 이미지를 사용하는 빌드 환경을 사용하는 경우)
- AWS Identity and Access Management (IAM)
- AWS Key Management Service (AWS KMS)
- Amazon Simple Storage Service(Amazon S3)

최소 필수 권한보다 더 제한적인 IAM 정책을 만들면 콘솔이 의도대로 작동하지 않습니다.

가 Amazon Elastic Container Registry AWS CodeBuild 에 연결하는 데 필요한 권한

2022년 7월 26일부터 AWS CodeBuild 는 Amazon ECR 권한에 대한 기본 IAM 정책을 업데이트했습니다. 다음 권한은 기본 정책에서 제거되었습니다.

```
"ecr:PutImage",
"ecr:InitiateLayerUpload",
"ecr:UploadLayerPart",
"ecr:CompleteLayerUpload"
```

2022년 7월 26일 이전에 생성된 CodeBuild 프로젝트의 경우 다음 Amazon ECR 정책으로 정책을 업데이트하는 것이 좋습니다.

```
"Action": [
    "ecr:BatchCheckLayerAvailability",
    "ecr:GetDownloadUrlForLayer",
    "ecr:BatchGetImage"
]
```

정책 업데이트에 대한 자세한 내용은 사용자가 CodeBuild와 상호 작용하도록 허용 섹션을 참조하세요.

AWS CodeBuild 콘솔이 소스 공급자에 연결하는 데 필요한 권한

AWS CodeBuild 콘솔은 다음 API 작업을 사용하여 소스 공급자(예: GitHub 리포지토리)에 연결합니다.

- `codebuild:ListConnectedOAuthAccounts`
- `codebuild:ListRepositories`
- `codebuild:PersistOAuthToken`
- `codebuild:ImportSourceCredentials`

AWS CodeBuild 콘솔을 사용하여 소스 공급자(예: GitHub 리포지토리)를 빌드 프로젝트와 연결할 수 있습니다. 이렇게 하려면 먼저 AWS CodeBuild 콘솔에 액세스하는 데 사용하는 사용자와 연결된 IAM 액세스 정책에 이전 API 작업을 추가해야 합니다.

`ListConnectedOAuthAccounts`, `ListRepositories` 및 `PersistOAuthToken` API 작업은 코드로 호출되는 것이 아닙니다. 따라서 이러한 API 작업은 AWS CLI 및 AWS SDKs에 포함되지 않습니다.

AWS 에 대한 관리형(미리 정의된) 정책 AWS CodeBuild

AWS 는에서 생성하고 관리하는 독립 실행형 IAM 정책을 제공하여 많은 일반적인 사용 사례를 처리합니다 AWS. 이러한 AWS 관리형 정책은 일반적인 사용 사례에 필요한 권한을 부여하므로 필요한 권한을 조사할 필요가 없습니다. CodeBuild의 관리형 정책은 해당 정책을 부여받은 사용자의 책임에 필요한 IAM, Amazon EC2 AWS CodeCommit, Amazon ECR, Amazon SNS 및 Amazon CloudWatch Events와 같은 다른 서비스에서 작업을 수행할 수 있는 권한도 제공합니다. 예를 들면 `AWSCodeBuildAdminAccess` 정책은 관리 수준의 사용자 정책으로, 이 정책을 통해 사용자가 프로젝트 관련 이벤트(이름에 `arn:aws:codebuild:`라는 접두사가 붙은 주제)에 대한 알림을 위한 프로젝트 빌드 및 Amazon SNS 주제에 대한 CloudWatch 이벤트 규칙을 생성 및 관리하고 CodeBuild에서 프로젝트 및 보고서 그룹을 관리할 수 있습니다. 자세한 내용은 IAM 사용자 안내서의 AWS 관리형 정책을 참조하세요.

계정의 사용자에게 연결할 수 있는 다음 AWS 관리형 정책은에 고유합니다 AWS CodeBuild.

AWSCodeBuildAdminAccess

CodeBuild 빌드 프로세스를 관리하는 데 필요한 권한을 비롯하여 CodeBuild에 대한 전체 액세스 권한을 제공합니다.

AWSCodeBuildDeveloperAccess

CodeBuild에 대한 액세스 권한을 제공하지만 빌드 프로젝트 관리는 허용하지 않습니다.

AWSCodeBuildReadOnlyAccess

CodeBuild에 대한 읽기 전용 액세스 권한을 제공합니다.

CodeBuild가 생성하는 빌드 출력 아티팩트에 액세스하려면 `AmazonS3ReadOnlyAccess`라는 AWS 관리형 정책도 연결해야 합니다.

CodeBuild 서비스 역할을 생성하고 관리하려면 라는 AWS 관리형 정책도 연결해야 합니다`IAMFullAccess`.

CodeBuild 작업 및 리소스에 대한 권한을 허용하는 고유의 사용자 지정 IAM 정책을 생성할 수도 있습니다. 해당 권한이 필요한 사용자 또는 그룹에 이러한 사용자 지정 정책을 연결할 수 있습니다.

주제

- AWSCodeBuildAdminAccess
- AWSCodeBuildDeveloperAccess
- AWSCodeBuildReadOnlyAccess

AWSCodeBuildAdminAccess

AWSCodeBuildAdminAccess 정책은 CodeBuild 빌드 프로세스를 관리하는 데 필요한 권한을 비롯하여 CodeBuild에 대한 전체 액세스 권한을 제공합니다. 이 정책은 관리자 수준 사용자에게만 적용하여 프로젝트 및 보고서 그룹을 삭제하는 기능을 포함하여 AWS 계정의 CodeBuild 프로젝트, 보고서 그룹 및 관련 리소스에 대한 전체 제어 권한을 부여합니다.

전체 관리형 정책은 IAM 관리형 정책 참조의 AWSCodeBuildAdminAccess를 참조하세요.

AWSCodeBuildDeveloperAccess

AWSCodeBuildDeveloperAccess 정책은 CodeBuild의 모든 기능과 프로젝트 및 보고서 그룹 관련 리소스에 액세스할 수 있도록 합니다. 이 정책은 사용자가 CodeBuild 프로젝트나 보고서 그룹 또는 CloudWatch Events와 같은 다른 AWS 서비스의 관련 리소스를 삭제하는 것을 허용하지 않습니다. 이 정책은 대부분의 사용자에게 적용하는 것이 좋습니다.

전체 관리형 정책은 IAM 관리형 정책 참조의 AWSCodeBuildDeveloperAccess를 참조하세요.

AWSCodeBuildReadOnlyAccess

이 AWSCodeBuildReadOnlyAccess 정책은 CodeBuild 및 다른 AWS 서비스의 관련 리소스에 대한 읽기 전용 액세스 권한을 부여합니다. 빌드를 보고 실행하고 프로젝트를 보고 보고서 그룹을 볼 수 있지만 변경할 수 없는 사용자에게 이 정책을 적용하십시오.

전체 관리형 정책은 IAM 관리형 정책 참조의 AWSCodeBuildReadOnlyAccess를 참조하세요.

CodeBuild 관리형 정책 및 알림

CodeBuild에서는 프로젝트를 빌드하는 데 중요한 변경 사항을 사용자에게 알릴 수 있는 알림을 지원합니다. CodeBuild에 대한 관리형 정책에는 알림 기능에 대한 정책 설명이 포함되어 있습니다. 자세한 내용은 알림이란 무엇입니까?를 참조하세요.

읽기 전용 관리형 정책의 알림과 관련된 권한

AWSCodeBuildReadOnlyAccess 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" : {
            "ArnLike" : {"codestar-notifications:NotificationsForResource" :
"arn:aws:codebuild:*:*:project/*"}
        }
    },
    {
        "Sid": "CodeStarNotificationsListAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:ListNotificationRules",
            "codestar-notifications:ListEventTypes",
            "codestar-notifications:ListTargets"
        ],
        "Resource": "*"
    }
```

다른 관리형 정책의 알림과 관련된 권한

AWSCodeBuildDeveloperAccess 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:Subscribe",
            "codestar-notifications:Unsubscribe"
        ],
        "Resource": "*",
        "Condition" : {
            "ArnLike" : {"codestar-notifications:NotificationsForResource" :
"arn:aws:codebuild:*:*:project/*"}
        }
    },
    {
        "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 알림에 대한 Identity and Access Management를 참조하세요.

AWS 관리형 정책에 대한 CodeBuild 업데이트

이 서비스가 이러한 변경 사항을 추적하기 시작한 이후부터 CodeBuild의 AWS 관리형 정책 업데이트에 대한 세부 정보를 봅니다. 이 페이지의 변경 사항에 대한 자동 알림을 받아보려면 AWS CodeBuild 사용 설명서 문서 기록에서 RSS 피드를 구독하세요.

변경	설명	Date
AWSCodeBuildAdminAccess 및 AWSCodeBuildDeveloperAccess – 기존 정책에 대한 업데이트	CodeBuild는 세션 관리자 대화형 빌드 디버깅을 지원하기 위해 이러한 정책에 ssmmessag	2025년 12월 1일

변경	설명	Date
	es:OpenDataChannel 권한을 추가했습니다. 이제 AWSCodeBuildAdminAccess 및 AWSCodeBuildDeveloperAccess 정책에 Session Manager 세션 리소스(arn:aws:ssm:*:*:session/*)에 대한 ssmmessages:OpenDataChannel 작업이 포함되어이 WebSocket API에서 SigV4 적용을 지원합니다.	
AWSCodeBuildAdminAccess , AWSCodeBuildDeveloperAccess 및 AWSCodeBuildReadOnlyAccess - 기존 정책 업데이트	CodeBuild는 리소스를 이러한 정책으로 업데이트했습니다. AWSCodeBuildAdminAccess , AWSCodeBuildDeveloperAccess , AWSCodeBuildReadOnlyAccess 정책이 기존 리소스를 업데이트하도록 변경되었습니다. 원본 리소스 arn:aws:codebuild:* 이 arn:aws:codebuild:*:*:project/* 로 업데이트되었습니다.	2024년 11월 15일

변경	설명	Date
AWSCodeBuildAdminAccess , AWSCodeBuildDeveloperAccess 및 AWSCodeBuildReadOnlyAccess - 기존 정책 업데이트	CodeBuild는 AWS CodeConnections 리브랜딩을 지원하기 위해 이러한 정책에 리소스를 추가했습니다. AWSCodeBuildAdminAccess , AWSCodeBuildDeveloperAccess 및 AWSCodeBuildReadOnlyAccess 정책이 arn:aws:codeconnections:*:*:* 리소스를 추가하도록 변경되었습니다.	2024년 4월 18일
AWSCodeBuildAdminAccess 및 AWSCodeBuildDeveloperAccess - 기존 정책 업데이트	CodeBuild는 채팅 애플리케이션에서 Amazon Q Developer를 사용하는 추가 알림 유형을 지원하기 위해 이러한 정책에 권한을 추가했습니다. AWSCodeBuildAdminAccess 및 AWSCodeBuildDeveloperAccess 정책이 권한, chatbot:ListMicrosoftTeamsChannelConfigurations 를 추가하도록 변경되었습니다.	2023년 5월 16일
CodeBuild가 변경 내용 추적을 시작함	CodeBuild가 AWS 관리형 정책에 대한 변경 사항 추적을 시작했습니다.	2021년 5월 16일

고객 관리형 정책 예제

이 섹션에서는 AWS CodeBuild 작업에 대한 권한을 부여하는 사용자 정책의 예를 제공합니다. 이러한 정책은 CodeBuild API, AWS SDKs 또는를 사용할 때 작동합니다 AWS CLI. 콘솔을 사용하는 경우 콘솔별 추가 권한을 부여해야 합니다. 자세한 내용은 AWS CodeBuild 콘솔을 사용하는 데 필요한 권한 섹션을 참조하세요.

다음 샘플 IAM 정책을 사용하여 사용자 및 역할에 대한 CodeBuild 액세스를 제한할 수 있습니다.

주제

- 사용자가 빌드 프로젝트에 대한 정보를 가져오도록 허용
- 사용자가 플릿에 대한 정보를 가져오도록 허용
- 사용자가 보고서 그룹에 대한 정보를 가져오도록 허용
- 사용자가 보고서에 대한 정보를 가져오도록 허용
- 사용자가 빌드 프로젝트를 생성하도록 허용
- 사용자가 플릿을 생성하도록 허용
- 사용자가 보고서 그룹을 생성하도록 허용
- 사용자가 플릿을 삭제하도록 허용
- 사용자가 보고서 그룹을 삭제하도록 허용
- 사용자가 보고서를 삭제하도록 허용
- 사용자가 빌드 프로젝트를 삭제하도록 허용
- 사용자가 빌드 프로젝트 이름 목록을 가져오도록 허용
- 사용자가 빌드 프로젝트에 대한 정보를 변경하도록 허용
- 사용자가 플릿을 변경하도록 허용
- 사용자가 보고서 그룹을 변경하도록 허용
- 사용자가 빌드에 대한 정보를 가져오도록 허용
- 사용자가 빌드 프로젝트의 빌드 ID 목록을 가져오도록 허용
- 사용자가 빌드 ID 목록을 가져오도록 허용
- 사용자가 플릿 목록을 가져오도록 허용
- 사용자가 보고서 그룹 목록을 가져오도록 허용
- 사용자가 보고서 목록을 가져오도록 허용

- 사용자가 보고서 그룹에 대한 보고서 목록을 가져오도록 허용
- 사용자가 보고서에 대한 테스트 케이스 목록을 가져오도록 허용
- 사용자가 빌드 실행을 시작하도록 허용
- 사용자가 빌드 중지를 시도하도록 허용
- 사용자가 빌드 삭제를 시도하도록 허용
- 사용자가 CodeBuild에서 관리하는 도커 이미지에 대한 정보를 가져오도록 허용
- 사용자가 플릿 서비스 역할에 대한 권한 정책을 추가하도록 허용
- VPC 네트워크 인터페이스를 생성하는 데 필요한 AWS 서비스에 대한 CodeBuild 액세스 허용
- 거부 문을 사용하여 AWS CodeBuild 가 소스 공급자와의 연결을 끊지 않도록 방지

사용자가 빌드 프로젝트에 대한 정보를 가져오도록 허용

다음은 us-east-2라는 이름으로 시작하는 빌드 프로젝트에 대해 사용자가 123456789012 계정의 my 리전에 있는 빌드 프로젝트에 대한 정보를 가져오도록 허용하는 정책 설명의 예입니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetProjects",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:project/my*"
    }
  ]
}
```

사용자가 플릿에 대한 정보를 가져오도록 허용

다음 예제 정책 설명을 통해 사용자는 123456789012 계정의 us-east-2 리전에서 플릿에 대한 정보를 가져올 수 있습니다.

JSON

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetFleets",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:fleet/*"
    }
  ]
}
```

사용자가 보고서 그룹에 대한 정보를 가져오도록 허용

다음 예제 정책 설명을 통해 사용자는 계정 123456789012의 us-east-2 리전에서 보고서 그룹에 대한 정보를 가져올 수 있습니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetReportGroups",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:report-group/*"
    }
  ]
}
```

사용자가 보고서에 대한 정보를 가져오도록 허용

다음 예제 정책 설명을 통해 사용자는 계정 123456789012의 us-east-2 리전에서 보고서에 대한 정보를 가져올 수 있습니다.

JSON

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

```
      "Effect": "Allow",
      "Action": "codebuild:BatchGetReports",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:report-group/*"
    }
  ]
}
```

사용자가 빌드 프로젝트를 생성하도록 허용

다음은 사용자에게 모든 이름의 빌드 프로젝트를 생성하도록 허용하지만, 123456789012 계정의 us-east-2 리전에만 있어야 하며, 지정된 CodeBuild 서비스 역할만 사용해야 하는 정책 설명의 예입니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:CreateProject",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:project/*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::111122223333:role/CodeBuildServiceRole"
    }
  ]
}
```

다음은 사용자에게 모든 이름의 빌드 프로젝트를 생성하도록 허용하지만, 123456789012 계정의 us-east-2 리전에만 있어야 하며, 지정된 CodeBuild 서비스 역할만 사용해야 하는 정책 설명의 예입니다. 또한 사용자가 다른 서비스가 AWS CodeBuild 아닌 에서만 지정된 AWS 서비스 역할을 사용할 수 있도록 강제합니다.

JSON

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:CreateProject",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:project/*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::111122223333:role/CodeBuildServiceRole",
      "Condition": {
          "StringEquals": {"iam:PassedToService": "codebuild.amazonaws.com"}
      }
    }
  ]
}
```

사용자가 플릿을 생성하도록 허용

다음 예제 정책 설명을 통해 사용자는 123456789012 계정의 us-east-2 리전에서 플릿을 생성할 수 있습니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:CreateFleet",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:fleet/*"
    }
  ]
}
```

사용자가 보고서 그룹을 생성하도록 허용

다음 예제 정책 설명을 통해 사용자는 계정 123456789012의 us-east-2 리전에서 보고서 그룹을 생성할 수 있습니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:CreateReportGroup",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:report-group/*"
    }
  ]
}
```

사용자가 플릿을 삭제하도록 허용

다음 예제 정책 설명을 통해 사용자는 123456789012 계정의 us-east-2 리전에서 플릿을 삭제할 수 있습니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:DeleteFleet",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:fleet/*"
    }
  ]
}
```

사용자가 보고서 그룹을 삭제하도록 허용

다음 예제 정책 설명을 통해 사용자는 계정 123456789012의 us-east-2 리전에서 보고서 그룹을 삭제할 수 있습니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:DeleteReportGroup",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:report-group/*"
    }
  ]
}
```

사용자가 보고서를 삭제하도록 허용

다음 예제 정책 설명을 통해 사용자는 계정 123456789012의 us-east-2 리전에서 보고서를 삭제할 수 있습니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:DeleteReport",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:report-group/*"
    }
  ]
}
```

사용자가 빌드 프로젝트를 삭제하도록 허용

다음은 us-east-2라는 이름으로 시작하는 빌드 프로젝트에 대해 사용자가 123456789012 계정의 my 리전에 있는 빌드 프로젝트를 삭제하도록 허용하는 정책 설명의 예입니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:DeleteProject",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:project/my*"
    }
  ]
}
```

사용자가 빌드 프로젝트 이름 목록을 가져오도록 허용

다음은 사용자가 동일한 계정의 빌드 프로젝트 이름 목록을 가져오도록 허용하는 정책 설명의 예입니다.

JSON

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

사용자가 빌드 프로젝트에 대한 정보를 변경하도록 허용

다음은 사용자에게 모든 이름의 빌드 프로젝트에 대한 정보를 변경하도록 허용하지만, 123456789012 계정의 us-east-2 리전에만 있어야 하며, 지정된 AWS CodeBuild 서비스 역할만 사용해야 하는 정책 설명의 예입니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:UpdateProject",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:project/*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::111122223333:role/CodeBuildServiceRole"
    }
  ]
}
```

사용자가 플릿을 변경하도록 허용

다음 예제 정책 설명을 통해 사용자는 123456789012 계정의 us-east-2 리전에서 플릿을 변경할 수 있습니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:UpdateFleet",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:fleet/*"
    }
  ]
}
```

사용자가 보고서 그룹을 변경하도록 허용

다음 예제 정책 설명을 통해 사용자는 계정 123456789012의 us-east-2 리전에서 보고서 그룹을 변경할 수 있습니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:UpdateReportGroup",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:report-group/*"
    }
  ]
}
```

사용자가 빌드에 대한 정보를 가져오도록 허용

다음은 us-east-2 및 123456789012라는 이름의 빌드 프로젝트에 대해 사용자가 my-build-project 계정의 my-other-build-project 리전에 있는 빌드에 대한 정보를 가져오도록 허용하는 정책 설명의 예입니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetBuilds",
      "Resource": [
        "arn:aws:codebuild:us-east-2:111122223333:project/my-build-project",
        "arn:aws:codebuild:us-east-2:111122223333:project/my-other-build-project"
      ]
    }
  ]
}
```

사용자가 빌드 프로젝트의 빌드 ID 목록을 가져오도록 허용

다음은 us-east-2 및 123456789012라는 이름의 빌드 프로젝트에 대해 사용자가 my-build-project 계정의 my-other-build-project 리전에 있는 빌드 ID 목록을 가져오도록 허용하는 정책 설명의 예입니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:ListBuildsForProject",
      "Resource": [
        "arn:aws:codebuild:us-east-2:111122223333:project/my-build-project",
        "arn:aws:codebuild:us-east-2:111122223333:project/my-other-build-project"
      ]
    }
  ]
}
```

사용자가 빌드 ID 목록을 가져오도록 허용

다음은 사용자가 동일한 계정의 모든 빌드 ID 목록을 가져오도록 허용하는 정책 설명의 예입니다.

JSON

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

사용자가 플릿 목록을 가져오도록 허용

다음 예제 정책 설명을 통해 사용자는 123456789012 계정의 us-east-2 리전에서 플릿 목록을 가져올 수 있습니다.

JSON

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

사용자가 보고서 그룹 목록을 가져오도록 허용

다음 예제 정책 설명을 통해 사용자는 계정 123456789012의 us-east-2 리전에서 보고서 그룹 목록을 가져올 수 있습니다.

JSON

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

사용자가 보고서 목록을 가져오도록 허용

다음 예제 정책 설명을 통해 사용자는 계정 123456789012의 us-east-2 리전에서 보고서 목록을 가져올 수 있습니다.

JSON

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

사용자가 보고서 그룹에 대한 보고서 목록을 가져오도록 허용

다음 예제 정책 설명을 통해 사용자는 계정 123456789012의 us-east-2 리전에서 보고서 그룹에 대한 보고서 목록을 가져올 수 있습니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:ListReportsForReportGroup",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:report-group/*"
    }
  ]
}
```

사용자가 보고서에 대한 테스트 케이스 목록을 가져오도록 허용

다음 예제 정책 설명을 통해 사용자는 계정 123456789012의 us-east-2 리전에서 보고서에 대한 테스트 케이스 목록을 가져올 수 있습니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:DescribeTestCases",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:report-group/*"
    }
  ]
}
```

사용자가 빌드 실행을 시작하도록 허용

다음은 us-east-2라는 이름으로 시작하는 빌드 프로젝트에 대해 사용자가 123456789012 계정의 my 리전에 있는 빌드를 실행하도록 허용하는 정책 설명의 예입니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:StartBuild",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:project/my*"
    }
  ]
}
```

사용자가 빌드 중지를 시도하도록 허용

다음은 us-east-2라는 이름으로 시작하는 빌드 프로젝트에 대해 사용자가 123456789012 계정의 my 리전에 있는 빌드만 실행 중지를 시도하도록 허용하는 정책 설명의 예입니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:StopBuild",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:project/my*"
    }
  ]
}
```

사용자가 빌드 삭제를 시도하도록 허용

다음은 이름이 my로 시작하는 빌드 프로젝트에 대해 사용자가 123456789012 계정의 us-east-2 리전에 있는 빌드만 삭제하려고 시도하는 것을 허용하는 정책 설명의 예입니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchDeleteBuilds",
      "Resource": "arn:aws:codebuild:us-east-2:111122223333:project/my*"
    }
  ]
}
```

사용자가 CodeBuild에서 관리하는 도커 이미지에 대한 정보를 가져오도록 허용

다음은 사용자가 CodeBuild가 관리하는 모든 도커 이미지에 대한 정보를 가져오도록 허용하는 정책 설명의 예입니다.

JSON

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

사용자가 플릿 서비스 역할에 대한 권한 정책을 추가하도록 허용

다음 리소스 정책 문 예제에서는 사용자가 플릿 서비스 역할에 대한 VPC 권한 정책을 추가할 수 있습니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "CodeBuildFleetVpcCreateNI",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface"
            ],
            "Resource": [
                "arn:aws:ec2:us-west-2:111122223333:subnet/subnet-id-1",
                "arn:aws:ec2:us-west-2:111122223333:security-group/security-
group-id-1",
                "arn:aws:ec2:us-west-2:111122223333:network-interface/*"
            ]
```

```
        },
        {
            "Sid": "CodeBuildFleetVpcPermission",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:DeleteNetworkInterface"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CodeBuildFleetVpcNIPermission",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterfacePermission"
            ],
            "Resource": "arn:aws:ec2:us-west-2:111122223333:network-interface/*",
            "Condition": {
                "StringEquals": {
                    "ec2:Subnet": [
                        "arn:aws:ec2:us-west-2:111122223333:subnet/subnet-id-1"
                    ]
                }
            }
        }
    ]
}
```

다음 예제 리소스 정책 문을 사용하면 사용자가 폴릿 서비스 역할에 대한 사용자 지정 Amazon Managed Image(AMI) 권한 정책을 추가할 수 있습니다.

JSON

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

```
        {
            "Effect": "Allow",
            "Action": "ec2:DescribeImages",
            "Resource": "*"
        }
    ]
}
```

다음 신뢰 정책 문 예제에서는 사용자가 플릿 서비스 역할에 대한 권한 정책을 추가할 수 있습니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "CodeBuildFleetVPCTrustPolicy",
      "Effect": "Allow",
      "Principal": {
        "Service": "codebuild.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "111122223333"
        }
      }
    }
  ]
}
```

VPC 네트워크 인터페이스를 생성하는 데 필요한 AWS 서비스에 대한 CodeBuild 액세스 허용

다음 예제 정책 설명은 두 개의 서브넷이 있는 VPC에서 네트워크 인터페이스를 생성할 수 있는 AWS CodeBuild 권한을 부여합니다.

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeVpcs"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterfacePermission"
            ],
            "Resource": "arn:aws:ec2:us-west-2:111122223333:network-interface/*",
            "Condition": {
                "StringEquals": {
                    "ec2:AuthorizedService": "codebuild.amazonaws.com"
                },
                "ArnEquals": {
                    "ec2:Subnet": [
                        "arn:aws:ec2:us-west-2:111122223333:subnet/subnet-id-1",
                        "arn:aws:ec2:us-west-2:111122223333:subnet/subnet-id-2"
                    ]
                }
            }
        }
    ]
}
```

거부 문을 사용하여 AWS CodeBuild 가 소스 공급자와의 연결을 끊지 않도록 방지

다음 예제 정책 명령문은 거부문을 사용하여 AWS CodeBuild 가 소스 공급자 연결을 해제하지 않도록 합니다. `codebuild:PersistOAuthToken` 및 `codebuild:ImportSourceCredentials`의 역인 `codebuild:DeleteOAuthToken`을 사용하여 소스 공급자와 연결합니다. 자세한 내용은 AWS CodeBuild 콘솔이 소스 공급자에 연결하는 데 필요한 권한 섹션을 참조하세요.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "codebuild:DeleteOAuthToken",
      "Resource": "*"
    }
  ]
}
```

AWS CodeBuild 권한 참조

AWS CodeBuild 정책에서 AWS전체 조건 키를 사용하여 조건을 표시할 수 있습니다. 목록은 IAM 사용 설명서에서 사용 가능한 키를 참조하세요.

정책의 `Action` 필드에 작업을 지정합니다. 작업을 지정하려면 `codebuild:` 접두사 다음에 API 작업 이름을 사용합니다(예: `codebuild:CreateProject` 및 `codebuild:StartBuild`). 문장 하나에 여러 작업을 지정하려면 쉼표로 구분합니다(예: `"Action": [ "codebuild:CreateProject", "codebuild:StartBuild" ]`).

와일드카드 문자 사용

정책의 `Resource` 필드에 리소스 값으로 와일드카드 문자(*)를 사용하거나 사용하지 않고 ARN을 지정합니다. 와일드카드를 사용하여 여러 작업 또는 리소스를 지정할 수 있습니다. 예를 들어, `codebuild:*`는 모든 CodeBuild 작업을 지정하고, `codebuild:Batch*`는 `Batch`라는 단어로 시작하는 모든 CodeBuild 작업을 지정합니다. 다음 정책은 이름이 `my`로 시작되는 모든 빌드 프로젝트에 대한 액세스 권한을 부여합니다.

```
arn:aws:codebuild:us-east-2:123456789012:project/my*
```

CodeBuild API 작업 및 작업에 필요한 권한

BatchDeleteBuilds

작업: `codebuild:BatchDeleteBuilds`

빌드를 삭제하는 데 필요합니다.

리소스: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

BatchGetBuilds

작업: codebuild:BatchGetBuilds

빌드에 대한 정보를 가져오는 권한이 필요합니다.

리소스: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

BatchGetProjects

작업: codebuild:BatchGetProjects

빌드 프로젝트에 대한 정보를 가져오는 권한이 필요합니다.

리소스: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

BatchGetReportGroups

작업: codebuild:BatchGetReportGroups

보고서 그룹에 대한 정보를 가져오는 권한이 필요합니다.

리소스: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

BatchGetReports

작업: codebuild:BatchGetReports

보고서에 대한 정보를 가져오는 권한이 필요합니다.

리소스: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

BatchPutTestCases [1]

작업: codebuild:BatchPutTestCases

테스트 보고서를 생성하거나 업데이트하는 권한이 필요합니다.

리소스: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

CreateProject

작업: codebuild:CreateProject, iam:PassRole

빌드 프로젝트를 생성하는 권한이 필요합니다.

리소스:

- arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*
- arn:aws:iam::*account-ID*:role/*role-name*

CreateReport [1]

작업: codebuild:CreateReport

테스트 보고서를 생성하는 권한이 필요합니다.

리소스: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

CreateReportGroup

작업: codebuild:CreateReportGroup

보고서 그룹을 만드는 권한이 필요합니다.

리소스: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

CreateWebhook

작업: codebuild:CreateWebhook

Webhook를 생성하는 데 필요합니다.

리소스: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

DeleteProject

작업: codebuild:DeleteProject

CodeBuild 프로젝트를 삭제하는 데 필요합니다.

리소스: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

DeleteReports

작업: codebuild:DeleteReport

규칙을 삭제하는 권한이 필요합니다.

리소스: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

DeleteReportGroup

작업: codebuild:DeleteReportGroup

보고서 그룹을 삭제하는 권한이 필요합니다.

리소스: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

DeleteSourceCredentials

작업: codebuild:DeleteSourceCredentials

GitHub나 GitHub Enterprise Server, Bitbucket 리포지토리의 자격 증명 정보가 포함된 일단의 SourceCredentialsInfo 객체를 삭제하는 데 필요합니다.

리소스: *

DeleteWebhook

작업: codebuild:DeleteWebhook

Webhook를 생성하는 데 필요합니다.

리소스: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

DescribeTestCases

작업: codebuild:DescribeTestCases

테스트 케이스의 페이지 매김 목록을 반환하는 권한이 필요합니다.

리소스: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

ImportSourceCredentials

작업: codebuild:ImportSourceCredentials

GitHub나 GitHub Enterprise Server, Bitbucket 리포지토리의 자격 증명 정보가 포함된 일단의 SourceCredentialsInfo 객체를 가져오는 데 필요합니다.

리소스: *

InvalidateProjectCache

작업: `codebuild:InvalidateProjectCache`

프로젝트용 캐시를 재설정하는 데 필요합니다.

리소스: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:project/`*`project-name`*

ListBuildBatches

작업: `codebuild:ListBuildBatches`

빌드 배치 ID 목록을 가져오는 권한이 필요합니다.

리소스: *

ListBuildBatchesForProject

작업: `codebuild:ListBuildBatchesForProject`

특정 프로젝트의 빌드 배치 ID 목록을 가져오는 권한이 필요합니다.

리소스: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:project/`*`project-name`*

ListBuilds

작업: `codebuild:ListBuilds`

빌드 ID 목록을 가져오는 권한이 필요합니다.

리소스: *

ListBuildsForProject

작업: `codebuild:ListBuildsForProject`

빌드 프로젝트의 빌드 ID 목록을 가져오는 권한이 필요합니다.

리소스: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:project/`*`project-name`*

ListCuratedEnvironmentImages

작업: `codebuild:ListCuratedEnvironmentImages`

AWS CodeBuild가 관리하는 모든 도커 이미지에 대한 정보를 가져오는 권한이 필요합니다.

리소스: *(필요, 단, 주소 지정 가능한 AWS 리소스는 제외)

ListProjects

작업: `codebuild:ListProjects`

빌드 프로젝트 이름 목록을 가져오는 권한이 필요합니다.

리소스: *

ListReportGroups

작업: `codebuild:ListReportGroups`

보고서 그룹 목록을 가져오는 권한이 필요합니다.

리소스: *

ListReports

작업: `codebuild:ListReports`

보고서 목록을 가져오는 권한이 필요합니다.

리소스: *

ListReportsForReportGroup

작업: `codebuild:ListReportsForReportGroup`

보고서 그룹에 대한 보고서 목록을 가져오는 권한이 필요합니다.

리소스: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:report-group/`*`report-group-name`*

RetryBuild

작업: `codebuild:RetryBuild`

빌드를 재시도하는 데 필요합니다.

리소스: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:project/`*`project-name`*

StartBuild

작업: `codebuild:StartBuild`

빌드 실행을 시작하는 권한이 필요합니다.

리소스: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:project/`*`project-name`*

StopBuild

작업: `codebuild:StopBuild`

빌드 실행 중지를 시도할 수 있는 권한이 필요합니다.

리소스: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:project/`*`project-name`*

UpdateProject

작업: `codebuild:UpdateProject`, `iam:PassRole`

빌드에 대한 정보를 변경하는 권한이 필요합니다.

리소스:

- `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:project/`*`project-name`*
- `arn:aws:iam::`*`account-ID`*`:role/`*`role-name`*

UpdateProjectVisibility

작업: `codebuild:UpdateProjectVisibility`, `iam:PassRole`

프로젝트 빌드의 퍼블릭 가시성을 변경하는 데 필요합니다.

리소스:

- `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:project/`*`project-name`*
- `arn:aws:iam::`*`account-ID`*`:role/`*`role-name`*

UpdateReport [1]

작업: `codebuild:UpdateReport`

테스트 보고서를 생성하거나 업데이트하는 권한이 필요합니다.

리소스: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:report-group/`*`report-group-name`*

UpdateReportGroup

작업: `codebuild:UpdateReportGroup`

보고서 그룹을 업데이트하는 권한이 필요합니다.

리소스: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:report-group/`*`report-group-name`*

UpdateWebhook

작업: `codebuild:UpdateWebhook`

Webhook를 업데이트하는 데 필요합니다.

리소스: `arn:aws:codebuild:region-ID:account-ID:project/project-name`

[1] 권한에만 사용됩니다. 이 작업에 대한 API가 없습니다.

태그를 사용하여 AWS CodeBuild 리소스에 대한 액세스 제어

IAM 정책 문의 조건은 CodeBuild 프로젝트 기반 작업에 대한 권한을 지정하는 데 사용할 수 있는 구문의 일부입니다. 해당 프로젝트와 연결된 태그를 기준으로 프로젝트에 대한 작업을 허용하거나 거부하는 정책을 생성한 다음, 사용자를 관리하기 위해 구성하는 IAM 그룹에 해당 정책을 적용할 수 있습니다. 콘솔을 사용하여 프로젝트에 태그를 적용하는 방법에 대한 자세한 내용은 섹션을 AWS CLI참조하세요에서 빌드 프로젝트 생성 AWS CodeBuild. CodeBuild SDK를 사용하여 태그를 적용하는 방법에 대한 자세한 내용은 CodeBuild API 참조의 CreateProject 및 태그를 참조하세요. 태그를 사용하여 AWS 리소스에 대한 액세스를 제어하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 AWS 리소스 태그를 사용하여 리소스에 대한 액세스 제어를 참조하세요.

> **Important**
>
> 예약 용량 기능을 사용할 때 소스 파일, Docker 계층, 빌드스펙에 지정된 캐시된 디렉터리를 포함하여 플릿 인스턴스에 캐시된 데이터를 동일한 계정 내의 다른 프로젝트에서 액세스할 수 있습니다. 이는 의도적으로 설계된 것이며 동일한 계정 내의 프로젝트가 플릿 인스턴스를 공유할 수 있도록 허용합니다.

Example예 1: 리소스 태그를 기준으로 CodeBuild 프로젝트 작업 제한

다음 예제에서는 키 `Environment` 및 키 값 `Production`으로 태그 지정된 프로젝트에 대한 모든 `BatchGetProjects` 작업을 거부합니다. 고객의 관리자는 권한 없는 사용자에게 관리형 사용자 정책 외에도 IAM 정책을 연결해야 합니다. `aws:ResourceTag` 조건 키는 태그를 기반으로 리소스에 대한 액세스를 제어하는 데 사용됩니다.

JSON

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codebuild:BatchGetProjects"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:ResourceTag/Environment": "Production"
        }
      }
    }
  ]
}
```

Example예 2: 요청 태그를 기준으로 CodeBuild 프로젝트 작업 제한

다음 정책은 요청에 키 Environment 및 키 값 Production을 사용하는 태그가 포함된 경우 CreateProject 작업에 대한 사용자 권한을 거부합니다. 또한 이 정책은 요청에 키 Environment를 사용하는 태그가 포함된 경우 UpdateProject를 허용하지 않도록 이러한 권한 없는 사용자가 aws:TagKeys 조건 키를 사용하여 프로젝트를 수정하는 것을 방지합니다. 관리자는 이러한 작업을 수행할 권한이 없는 사용자에게 관리형 사용자 정책 외에도 이 IAM 정책을 연결해야 합니다. aws:RequestTag 조건 키는 IAM 요청에서 전달할 수 있는 태그를 제어하는 데 사용됩니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codebuild:CreateProject"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:RequestTag/Environment": "Production"
        }
```

```
      }
    },
    {
      "Effect": "Deny",
      "Action": [
        "codebuild:UpdateProject"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEqualsIgnoreCase": {
          "aws:TagKeys": ["Environment"]
        }
      }
    }
  ]
}
```

Example예 3: 리소스 태그를 기반으로 보고서 그룹에 대한 작업 거부 또는 허용

해당 리소스와 연결된 AWS 태그를 기반으로 CodeBuild 리소스(프로젝트 및 보고서 그룹)에 대한 작업을 허용하거나 거부하는 정책을 생성한 다음 사용자를 관리하기 위해 구성한 IAM 그룹에 해당 정책을 적용할 수 있습니다. 예를 들어 AWS 태그 키`Status`와 키 값이 인 보고서 그룹에 대한 모든 CodeBuild 작업을 거부하는 정책을 생성한 `Secret`다음 일반 개발자(`###`)를 위해 생성한 IAM 그룹에 해당 정책을 적용할 수 있습니다. 그런 다음, 태그가 지정된 보고서 그룹에 대해 작업하는 개발자가 일반 `###` 그룹에 속하지 않고, 대신에 제한 정책이 적용되지 않는 다른 IAM 그룹(`SecretDevelopers`)에 속하게 해야 합니다.

다음 예제에서는 키 `Status` 및 키 값 `Secret`으로 태그가 지정된 보고서 그룹에 대한 모든 CodeBuild 작업을 거부합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codebuild:BatchGetReportGroups",
```

```
        "codebuild:CreateReportGroup",
        "codebuild:DeleteReportGroup",
        "codebuild:ListReportGroups",
        "codebuild:ListReportsForReportGroup",
        "codebuild:UpdateReportGroup"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aws:RequestedRegion": "us-east-1"
        }
      }
    }
  ]
}
```

Example예 4: 리소스 태그를 기준으로 CodeBuild 작업을 AWSCodeBuildDeveloperAccess로 제한

특정 태그가 지정되지 않은 모든 보고서 그룹 및 프로젝트에 대한 CodeBuild 작업을 허용하는 정책을 생성할 수 있습니다. 예를 들어 다음 정책은 특정 태그로 태그 지정된 보고서 그룹 및 프로젝트를 제외한 모든 보고서 그룹 및 프로젝트에 대해 AWSCodeBuildDeveloperAccess 권한과 동등한 권한을 허용합니다.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "codebuild:StartBuild",
            "codebuild:StopBuild",
            "codebuild:BatchGet*",
            "codebuild:GetResourcePolicy",
            "codebuild:DescribeTestCases",
            "codebuild:List*",
            "codecommit:GetBranch",
            "codecommit:GetCommit",
            "codecommit:GetRepository",
            "codecommit:ListBranches",
```

```
            "cloudwatch:GetMetricStatistics",
            "events:DescribeRule",
            "events:ListTargetsByRule",
            "events:ListRuleNamesByTarget",
            "logs:GetLogEvents",
            "s3:GetBucketLocation",
            "s3:ListAllMyBuckets"
         ],
         "Resource": "*",
         "Condition": {
            "StringNotEquals": {
               "aws:ResourceTag/Status": "Secret",
               "aws:ResourceTag/Team": "Saanvi"
            }
         }
      }
   ]
}
```

콘솔에서 리소스 보기

AWS CodeBuild 콘솔에는 로그인한 AWS 리전의 AWS 계정에 대한 리포지토리 목록을 표시할 수 있는 `ListRepositories` 권한이 필요합니다. 또한 콘솔에는 리소스의 대/소문자를 구분하지 않고 빠르게 검색할 수 있는 Go to resource(리소스로 이동) 기능이 포함되어 있습니다. 이 검색은 로그인한 AWS 리전의 계정 AWS 에서 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 Deny가 있으면 일부 리소스가 반환되지 않습니다.

에 대한 규정 준수 검증 AWS CodeBuild

새로운 AWS 가입 환경은 규제 워크로드용으로 설계되지 않았습니다. 새로운 AWS 가입 환경을 사용하고 있지만 규제 대상 워크로드 AWS 에를 사용하려는 경우 AWS 환경에 가입 AWS (고급)하거나 고급 기능을 활성화할 수 있습니다.

AWS 서비스 가 특정 규정 준수 프로그램의 범위 내에 있는지 알아보려면 AWS 서비스 규정 준수 프로그램 범위 내 참조하고 관심 있는 규정 준수 프로그램을 선택합니다. 일반 정보는 AWS 규정 준수 프로그램.

를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다 AWS Artifact. 자세한 내용은 Downloading Reports inDownloading AWS Artifact을 참조하세요.

사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표, 관련 법률 및 규정에 따라 AWS 서비스 결정됩니다. 사용 시 규정 준수 책임에 대한 자세한 내용은 AWS 보안 설명서를 AWS 서비스참조하세요.

의 복원력 AWS CodeBuild

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다. AWS 리전은 물리적으로 분리되고 격리된 여러 가용 영역을 제공하며,이 가용 영역은 짧은 지연 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워킹으로 연결됩니다. 가용 영역을 사용하면 중단 없이 가용 영역 간에 자동으로 장애 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 복수 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

의 인프라 보안 AWS CodeBuild

관리형 서비스인는 AWS 글로벌 네트워크 보안으로 보호 AWS CodeBuild 됩니다. AWS 보안 서비스 및가 인프라를 AWS 보호하는 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하세요. 인프라 보안 모범 사례를 사용하여 환경을 설계하려면 보안 원칙 AWS Well-Architected Framework의 인프라 보호를 참조하세요 AWS .

AWS 에서 게시한 API 호출을 사용하여 네트워크를 통해 CodeBuild에 액세스합니다. 클라이언트는 다음을 지원해야 합니다.

- Transport Layer Security(TLS). TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

CodeBuild에서 소스 공급자에 액세스

GitHub 또는 GitHub Enterprise Server에서는 개인 액세스 토큰, Secrets Manager 보안 암호, 연결 또는 OAuth 앱을 사용하여 소스 공급자에 액세스합니다. Bitbucket의 경우 액세스 토큰, 앱 암호, API 토큰, Secrets Manager 보안 암호, 연결 또는 OAuth 앱을 사용하여 소스 공급자에 액세스합니다.

주제

- Secrets Manager 보안 암호에 토큰 생성 및 저장
- CodeBuild의 GitHub 및 GitHub Enterprise Server 액세스
- CodeBuild의 Bitbucket 액세스
- CodeBuild의 GitLab 액세스

Secrets Manager 보안 암호에 토큰 생성 및 저장

Secrets Manager를 사용하여 액세스 토큰을 저장하도록 선택한 경우 기존 보안 암호 연결을 사용하거나 새 보안 암호를 생성할 수 있습니다. 새 보안 암호를 생성하려면 다음을 수행합니다.

AWS Management Console

에서 Secrets Manager 보안 암호를 생성하려면 AWS Management Console

1. 소스 공급자에서 Bitbucket, GitHub 또는 GitHub Enterprise를 선택합니다.
2. 자격 증명에서 다음 중 하나를 수행합니다.
   - 기본 소스 자격 증명을 선택하여 계정의 기본 소스 자격 증명을 사용하여 모든 프로젝트에 적용합니다.

     a. 소스 공급자에 연결되지 않은 경우 기본 소스 자격 증명 관리를 선택합니다.

     b. 자격 증명 유형에서 이외의 자격 증명 유형을 선택합니다AWS CodeConnections.

c. 서비스에서 Secrets Manager를 선택하고 보안 암호에서 새 보안 암호를 선택합니다.

d. 보안 암호 이름에 대해 보안 암호 이름을 입력합니다.

e. 보안 암호 설명- 선택 사항에 보안 암호에 대한 설명을 입력합니다.

f. 선택한 소스 공급자에 따라 토큰 또는 사용자 이름 및 앱 암호를 입력하고 저장을 선택합니다. Bitbucket API 토큰의 경우 Atlassian 계정 이메일 주소와 API 토큰을 입력합니다.

- 사용자 지정 소스 자격 증명을 선택하여 사용자 지정 소스 자격 증명을 사용하여 계정의 기본 설정을 재정의합니다.

a. 자격 증명 유형에서 이외의 자격 증명 유형을 선택합니다AWS CodeConnections.

b. 연결에서 보안 암호 생성을 선택합니다.

c. 보안 암호 이름에 대해 보안 암호 이름을 입력합니다.

d. 보안 암호 설명- 선택 사항에 보안 암호에 대한 설명을 입력합니다.

e. 선택한 소스 공급자에 따라 토큰 또는 사용자 이름 및 앱 암호를 입력하고 생성을 선택합니다. Bitbucket API 토큰의 경우 Atlassian 계정 이메일 주소와 API 토큰을 입력합니다.

AWS CLI

에서 Secrets Manager 보안 암호를 생성하려면 AWS CLI

- 터미널(Linux, macOS, Unix) 또는 명령 프롬프트(Windows)를 엽니다. AWS CLI 를 사용하여 Secrets Manager create-secret 명령을 실행합니다.

```
aws secretsmanager create-secret --region <aws-region> \
            --name '<secret-name>' \
            --description '<secret-description>' \
            --secret-string '{
                "ServerType":"<server-type>",
                "AuthType":"<auth-type>",
                "Token":"<token>"
                }' \
            --tags Key=codebuild:source,Value='' \
                Key=codebuild:source:type,Value=<type> \
                Key=codebuild:source:provider,Value=<provider>
```

CodeBuild가 수락하는 Secrets Manager 보안 암호는 CodeBuild 프로젝트와 동일한 계정 및 AWS 리전에 있어야 하며 다음 JSON 형식이어야 합니다.

```
{
            "ServerType": ServerType,
            "AuthType: AuthType,
            "Token": string,
            "Username": string // Optional and is only used for Bitbucket app
 password or API token
        }
```

Field	유효값	설명
ServerType	GITHUB GITHUB_ENTERPRISE BITBUCKET	Secrets Manager 보안 암호의 타사 소스 공급자입니다.
AuthType	PERSONAL_ACCESS_TOKEN BASIC_AUTH	자격 증명에서 사용하는 액세스 토큰의 유형입니다. GitHub의 경우 PERSONAL_ACCESS_TOKEN만 유효합니다. BASIC_AUTH는 Bitbucket 앱 암호 또는 API 토큰에만 유효합니다.
토큰	*string*	GitHub 또는 GitHub Enterprise에서는 개인용 액세스 토큰을 말합니다. Bitbucket의 경우 액세스 토큰, Bitbucket 앱 암호 또는 Bitbucket API 토큰입니다.

Field	유효값	설명
사용자 이름	*string*	AuthType이 BASIC_AUTH일 경우 Bitbucket 사용자 이름입니다. Bitbucket API 토큰을 사용하는 경우 Atlassian 계정과 연결된 이메일 주소입니다. 이 파라미터는 다른 유형의 소스 공급자에서는 유효하지 않습니다.

또한 CodeBuild는 보안 암호에 다음 리소스 태그를 사용하여 프로젝트를 생성하거나 편집할 때 보안 암호를 쉽게 선택할 수 있도록 합니다.

태그 키	태그 값	설명
codebuild:source:provider	GitHub github_enterprise bitbucket	CodeBuild에 이 보안 암호가 사용될 공급자를 알려줍니다.
codebuild:source:type	personal_access_token basic_auth	CodeBuild에 이 보안 암호의 액세스 토큰 유형을 알려줍니다.

CodeBuild의 GitHub 및 GitHub Enterprise Server 액세스

GitHub의 경우 개인 액세스 토큰, OAuth 앱, Secrets Manager 보안 암호 또는 GitHub 앱 연결을 사용하여 소스 공급자에 액세스할 수 있습니다. GitHub Enterprise Server에서는 개인 액세스 토큰, Secrets Manager 암호 또는 GitHub 앱 연결을 사용하여 소스 공급자에 액세스할 수 있습니다.

주제

- GitHub 및 GitHub Enterprise Server에 대한 GitHub 앱 연결
- GitHub 및 GitHub Enterprise Server 액세스 토큰

- GitHub OAuth 앱

GitHub 및 GitHub Enterprise Server에 대한 GitHub 앱 연결

GitHub 앱을 사용하여 CodeBuild에 연결할 수 있습니다. GitHub 앱 연결은 AWS CodeConnections를 통해 지원됩니다

소스 공급자 액세스를 사용하면 CreateWebhook를 사용하여 GitHub Webhook 이벤트에 가입하여 빌드를 트리거하거나 CodeBuild에서 자습서: CodeBuild 호스팅 GitHub Action 실행기 구성를 사용할 수 있습니다.

> **Note**
>
> AWS CodeConnections 는 CodeBuild보다 적은 리전에서 사용할 수 있습니다. CodeBuild에서 리전 간 연결을 사용할 수 있습니다. 옵트인 리전에서 생성된 연결은 다른 리전에서 사용할 수 없습니다. 자세한 내용은 AWS CodeConnections 엔드포인트 및 할당량을 참조하세요.

주제

- 1단계: GitHub 앱에 대한 연결 생성(콘솔)
- 2단계: CodeBuild 프로젝트에 연결을 사용할 IAM 역할 액세스 권한 부여
- 3단계: 새 연결을 사용하도록 CodeBuild 구성
- GitHub 앱을 사용한 문제 해결

1단계: GitHub 앱에 대한 연결 생성(콘솔)

다음 단계를 통해 CodeBuild 콘솔을 사용하여 GitHub의 프로젝트에 대한 연결을 추가할 수 있습니다.

GitHub에 대한 연결을 생성하려면

- 개발자 도구 사용 설명서의 GitHub에 대한 연결 생성을 위한 지침을 따릅니다.

> **Note**
>
> 계정에서 기존 연결을 생성하거나 사용하는 대신 다른 AWS 계정에서 공유된 연결을 사용할 수 있습니다. 자세한 내용은 AWS 계정과 연결 공유를 참조하세요.

2단계: CodeBuild 프로젝트에 연결을 사용할 IAM 역할 액세스 권한 부여

연결에서 제공한 GitHub 토큰을 사용할 수 있는 IAM 역할 액세스 권한을 CodeBuild 프로젝트에 부여할 수 있습니다.

CodeBuild 프로젝트에 IAM 역할 액세스 권한을 부여하려면

1. CodeBuild 프로젝트의 CodeBuild가 다른 AWS 서비스와 상호 작용하도록 허용에 대한 지침에 따라 CodeBuild 프로젝트의 IAM 역할을 생성합니다.
2. 지침에 따라 CodeBuild 프로젝트 역할에 다음 IAM 정책을 추가하여 연결에 대한 액세스 권한을 부여합니다.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codeconnections:GetConnectionToken",
                "codeconnections:GetConnection"
            ],
            "Resource": [
                "arn:aws:iam::*:role/Service*"
            ]
        }
    ]
}
```

3단계: 새 연결을 사용하도록 CodeBuild 구성

연결을 계정 수준 자격 증명으로 구성하고 프로젝트에서 사용할 수 있습니다.

AWS Management Console

에서 연결을 계정 수준 자격 증명으로 구성하려면 AWS Management Console

1. 소스 공급자에서 GitHub를 선택합니다.
2. 자격 증명에서 다음 중 하나를 수행합니다.

- 기본 소스 자격 증명을 선택하여 계정의 기본 소스 자격 증명을 사용하여 모든 프로젝트에 적용합니다.

  a. GitHub에 연결되지 않은 경우 기본 소스 자격 증명 관리를 선택합니다.

  b. 자격 증명 유형에서 GitHub 앱을 선택합니다.

  c. 연결에서 기존 연결을 선택하거나 연결을 새로 생성합니다.

- 사용자 지정 소스 자격 증명을 선택하여 사용자 지정 소스 자격 증명을 사용하여 계정의 기본 설정을 재정의합니다.

  a. 자격 증명 유형에서 GitHub 앱을 선택합니다.

  b. 연결에서 기존 연결을 선택하거나 연결을 새로 생성합니다.

AWS CLI

에서 연결을 계정 수준 자격 증명으로 구성하려면 AWS CLI

- 터미널(Linux, macOS, Unix) 또는 명령 프롬프트(Windows)를 엽니다. AWS CLI 를 사용하여 연결을 `--token` 위해 `--auth-type`, `--server-type`및를 지정하여 import-source-credentials 명령을 실행합니다.

  다음 명령을 사용합니다.

```
aws codebuild import-source-credentials --auth-type CODECONNECTIONS --server-type GITHUB --token <connection-arn>
```

CodeBuild 프로젝트에 여러 토큰을 설정할 수도 있습니다. 자세한 내용은 여러 토큰을 소스 수준 자격 증명으로 구성 섹션을 참조하세요.

GitHub 앱을 사용한 문제 해결

다음 정보는 GitHub App과 관련된 문제를 해결하는 데 도움이 될 수 있습니다.

주제

- 원치 않는 리전에 GitHub용 AWS 커넥터 앱 설치
- GitHub 앱 연결에는 리포지토리에 대한 액세스 권한이 없습니다.
- AWS 서비스의 IAM 역할에 필요한 IAM 권한이 없습니다.

원치 않는 리전에 GitHub용 AWS 커넥터 앱 설치

문제: GitHub Marketplace에서 GitHub용 AWS 커넥터를 설치했지만 원치 않는 리전에서 연결이 생성되었습니다. GitHub 웹 사이트에서 앱을 재구성하려고 하면 앱이 이미 GitHub 계정에 설치되어 있기 때문에 작동하지 않습니다.

가능한 원인: 앱이 GitHub 계정에 이미 설치되어 있으므로 앱 권한만 재구성할 수 있습니다.

권장 솔루션: 원하는 리전의 설치 ID로 새 연결을 생성할 수 있습니다.

1. https://console.aws.amazon.com/codesuite/settings/connections AWS CodeConnections 콘솔을 열고 AWS 콘솔 탐색 모음에서 리전 선택기를 사용하여 원하는 리전으로 이동합니다.
2. 개발자 도구 사용 설명서의 GitHub에 대한 연결 생성을 위한 지침을 따릅니다.

> **Note**
>
> AWS Connector for GitHub 앱을 이미 설치했으므로 새 앱을 설치하는 대신 선택할 수 있습니다.

GitHub 앱 연결에는 리포지토리에 대한 액세스 권한이 없습니다.

문제: CodeBuild 또는 CodePipeline과 같은 연결을 사용하는 AWS 서비스는 리포지토리에 액세스할 수 없거나 리포지토리가 존재하지 않는다고 보고합니다. 몇 가지 가능한 오류 메시지는 다음과 같습니다.

- `Authentication required for primary source.`
- `Unable to create webhook at this time. Please try again later.`
- `Failed to create webhook. GitHub API limit reached. Please try again later.`

가능한 원인: GitHub 앱을 사용하고 웹후크 권한 범위를 부여하지 않았을 수 있습니다.

권장 솔루션: 필요한 권한 범위를 부여하려면 검토하거나 수정하려는 GitHub 앱으로 이동의 지침에 따라 설치된 앱을 구성합니다. 권한 섹션에서 앱에 웹후크 권한이 없는 것을 확인할 수 있으며 새로 요청된 권한을 검토할 수 있는 옵션이 있습니다. 새 권한을 검토하고 수락합니다. 자세한 내용은 GitHub 앱에 대한 업데이트된 권한 승인을 참조하세요.

가능한 원인: 연결이 예상대로 작동했지만 갑자기 리포지토리에 액세스할 수 없습니다.

가능한 솔루션: 먼저 권한 부여 및 설치를 검토하여 시작한 다음 GitHub 앱이 권한 부여 및 설치되었는지 확인합니다. GitHub 앱 설치가 일시 중지된 경우 일시 중지를 해제해야 합니다. GitHub 앱이 UAT(사용자 액세스 토큰) 연결에 대해 승인되지 않았거나 IAT(설치 액세스 토큰) 연결에 설치되지 않은 경우 기존 연결을 더 이상 사용할 수 없으므로 새 연결을 생성해야 합니다. GitHub 앱을 다시 설치해도 이전 설치와 연결된 이전 연결은 복원되지 않습니다.

가능한 솔루션: 연결이 UAT 연결인 경우 여러 CodeBuild 동시 빌드 실행에서 사용되는 것과 같이 연결이 동시에 사용되지 않는지 확인합니다. 이는 GitHub가 연결에 의해 만료되는 토큰이 새로 고쳐지면 이전에 발급된 UAT를 즉시 무효화하기 때문입니다. 여러 동시 CodeBuild 빌드에 UAT 연결을 사용해야 하는 경우 여러 연결을 생성하고 각 연결을 독립적으로 사용할 수 있습니다.

가능한 솔루션: 지난 6개월 동안 UAT 연결을 사용하지 않은 경우 GitHub에서 연결이 무효화됩니다. 이를 수정하려면 새로운 연결을 생성합니다.

가능한 원인: 앱을 설치하지 않고 UAT 연결을 사용했을 수 있습니다.

권장 솔루션: UAT 연결을 생성하면 GitHub 앱 설치와 연결을 연결할 필요가 없지만 리포지토리에 액세스하려면 설치가 필요합니다. 지침에 따라 설치를 검토하여 GitHub 앱이 설치되어 있는지 확인합니다. 설치되지 않은 경우 GitHub 앱 페이지로 이동하여 앱을 설치합니다. UAT의 액세스에 대한 자세한 내용은 사용자 액세스 토큰 정보를 참조하세요.

AWS 서비스의 IAM 역할에 필요한 IAM 권한이 없습니다.

문제: 다음 오류 메시지 중 하나가 표시됩니다.

- `Access denied to connection <connection-arn>`
- `Failed to get access token from <connection-arn>`

권장 솔루션: 일반적으로 CodePipeline 또는 CodeBuild와 같은 AWS 서비스와의 연결을 사용합니다. AWS 서비스에 IAM 역할을 부여하면 AWS 서비스는 역할의 권한을 사용하여 사용자를 대신하여 작업할 수 있습니다. IAM 역할에 필요한 권한이 있는지 확인합니다. 필요한 IAM 권한에 대한 자세한 내용은 개발자 도구 콘솔 사용 설명서의 CodeBuild 프로젝트 IAM 역할에 알림에 대한 연결 및 자격 증명 및 액세스 관리를 사용할 수 있는 액세스 권한 부여를 참조하세요 AWS CodeStarAWS CodeConnections.

GitHub 및 GitHub Enterprise Server 액세스 토큰

액세스 토큰 사전 조건

시작하기 전에 GitHub 액세스 토큰에 적절한 권한 범위를 추가해야 합니다.

GitHub의 경우 개인용 액세스 토큰 범위가 다음과 같아야 합니다.

- repo: 프라이빗 리포지토리의 전체 제어를 부여합니다.
- repo:status: 퍼블릭 및 프라이빗 리포지토리 커밋 상태에 대한 읽기/쓰기 권한을 부여합니다.
- admin:repo_hook: 리포지토리 후크의 전체 제어를 부여합니다. 토큰에 `repo` 범위가 있을 경우 이 범위가 필요하지 않습니다.
- admin:org_hook: 조직 후크를 완전히 제어할 수 있습니다. 이 범위는 조직 웹후크 기능을 사용하는 경우에만 필요합니다.

자세한 내용은 GitHub 웹사이트의 OAuth 앱에 대한 범위 이해를 참조하십시오.

세분화된 개인 액세스 토큰을 사용하는 경우 사용 사례에 따라 개인 액세스 토큰에 다음 권한이 필요할 수 있습니다.

- 목차: 읽기 전용: 프라이빗 리포지토리에 대한 액세스 권한을 부여합니다. 프라이빗 리포지토리를 소스로 사용하는 경우 이 권한이 필요합니다.
- 커밋 상태: 읽기 및 쓰기: 커밋 상태를 생성할 수 있는 권한을 부여합니다. 프로젝트에 웹후크가 설정되어 있거나 보고서 빌드 상태 기능이 활성화된 경우 이 권한이 필요합니다.
- 웹후크: 읽기 및 쓰기: 웹후크를 관리할 수 있는 권한을 부여합니다. 프로젝트에 웹후크가 설정된 경우 이 권한이 필요합니다.
- pull 요청: 읽기 전용: pull 요청에 액세스할 수 있는 권한을 부여합니다. 이 권한은 웹후크에 pull 요청 이벤트에 대한 `FILE_PATH` 필터가 있는 경우 필요합니다.
- 관리: 읽기 및 쓰기: CodeBuild에서 자체 호스팅 GitHub Action 실행기 기능을 사용하는 경우 이 권한이 필요합니다. 자세한 내용은 리포지토리에 대한 등록 토큰 생성 및 자습서: CodeBuild 호스팅 GitHub Action 실행기 구성 섹션을 참조하세요.

> **Note**
>
> 조직 리포지토리에 액세스하려면 조직을 액세스 토큰의 리소스 소유자로 지정해야 합니다.

자세한 내용은 GitHub 웹 사이트의 세분화된 개인 액세스 토큰에 필요한 권한을 참조하세요.

액세스 토큰을 사용하여 GitHub에 연결(콘솔)

콘솔에서 액세스 토큰을 사용하여 프로젝트를 GitHub에 연결하려면 프로젝트를 생성할 때 다음과 같이 합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 섹션을 참조하세요.

1. 소스 공급자에서 GitHub를 선택합니다.
2. 자격 증명에서 다음 중 하나를 수행합니다.
   - 계정 자격 증명을 사용하여 모든 프로젝트에 계정의 기본 소스 자격 증명을 적용하려면 선택합니다.
     a. GitHub에 연결되지 않은 경우 계정 자격 증명 관리를 선택합니다.
     b. 자격 증명 유형에서 개인 액세스 토큰을 선택합니다.
   - 서비스에 계정 수준 자격 증명을 사용하기로 선택한 경우 토큰을 저장하는 데 사용할 서비스를 선택하고 다음을 수행합니다.
     a. Secrets Manager를 사용하도록 선택한 경우 기존 보안 암호 연결을 사용하거나 새 보안 암호를 생성하고 저장을 선택할 수 있습니다. 새 보안 암호를 생성하는 방법에 대한 자세한 내용은 섹션을 참조하세요Secrets Manager 보안 암호에 토큰 생성 및 저장.
     b. CodeBuild를 사용하도록 선택한 경우 GitHub 개인 액세스 토큰을 입력하고 저장을 선택합니다.
   - 사용자 지정 소스 자격 증명을 사용하여 계정의 자격 증명 설정을 재정의하려면 이 프로젝트에 대해서만 자격 증명 재정의 사용을 선택합니다.
     a. 채워진 자격 증명 목록에서 개인 액세스 토큰 아래의 옵션 중 하나를 선택합니다.
     b. 설명에서 새 개인 액세스 토큰 연결 생성을 선택하여 새 개인 액세스 토큰을 생성할 수도 있습니다.

액세스 토큰을 사용하여 GitHub에 연결(CLI)

다음 단계에 따라를 사용하여 액세스 토큰 AWS CLI 을 사용하여 프로젝트를 GitHub에 연결합니다. 와 AWS CLI 함께를 사용하는 방법에 대한 자세한 내용은 단원을 AWS CodeBuild참조하십시오명령줄 참조.

1. import-source-credentials 명령을 실행합니다.

```
aws codebuild import-source-credentials --generate-cli-skeleton
```

JSON 형식 데이터가 출력에 표시됩니다. 가 설치된 로컬 컴퓨터 또는 인스턴스의 위치에 있는 파일(예: *import-source-credentials.json*)에 데이터를 복사 AWS CLI 합니다. 복사된 데이터를 다음과 같이 수정하고 결과를 저장합니다.

```
{
    "serverType": "server-type",
    "authType": "auth-type",
    "shouldOverwrite": "should-overwrite",
    "token": "token",
    "username": "username"
    }
```

다음을 바꿉니다.

- *server-type*: 필수 값. 이 자격 증명에 사용되는 소스 공급자. 유효한 값은 GITHUB, BITBUCKET, GITHUB_ENTERPRISE, GITLAB 및 GITLAB_SELF_MANAGED입니다.
- *auth-type*: 필수 값. 리포지토리에 연결하는 데 사용되는 인증 유형입니다. 유효한 값은 OAUTH, BASIC_AUTH, Personal_ACCESS_TOKEN, CODECONNECTIONS 및 SECRETS_MANAGER입니다. GitHub의 경우 PERSONAL_ACCESS_TOKEN만 허용됩니다. BASIC_AUTH는 Bitbucket 앱 암호 또는 API 토큰에서만 허용됩니다.
- *should-overwrite*: 선택적 값입니다. 리포지토리 소스 자격 증명을 덮어쓰지 않도록 하려면 `false`로 설정합니다. 리포지토리 소스 자격 증명을 덮어쓰려면 `true`로 설정합니다. 기본값은 `true`입니다.
- *token*: 필수 값. GitHub 또는 GitHub Enterprise Server에서는 개인용 액세스 토큰을 말합니다. Bitbucket의 경우 개인 액세스 토큰, 앱 암호 또는 API 토큰입니다. 인증 유형 CODECONNECTIONS의 경우 연결 ARN입니다. 인증 유형 SECRETS_MANAGER의 경우 보안 암호 ARN입니다.
- *username*: 선택 사항 값. GitHub 및 GitHub Enterprise Server 소스 공급자의 경우 이 파라미터가 무시됩니다. Bitbucket 앱 암호의 경우 Bitbucket 사용자 이름입니다. Bitbucket API 토큰의 경우 Atlassian 계정과 연결된 이메일 주소입니다.

2. 액세스 토큰으로 계정에 연결하려면 1단계에서 저장한 `import-source-credentials.json` 파일이 있는 디렉터리로 이동한 후 import-source-credentials 명령을 다시 실행합니다.

```
aws codebuild import-source-credentials --cli-input-json file://import-source-
credentials.json
```

JSON 형식 데이터와 Amazon 리소스 이름(ARN)이 출력에 표시됩니다.

```
{
    "arn": "arn:aws:codebuild:region:account-id:token/server-type"
    }
```

> **Note**
>
> import-source-credentials 명령을 동일한 서버 유형과 인증 유형으로 다시 실행하면 저장된 액세스 토큰이 업데이트됩니다.

계정을 액세스 토큰과 연결한 후 `create-project`를 사용하여 CodeBuild 프로젝트를 생성할 수 있습니다. 자세한 내용은 빌드 프로젝트 생성(AWS CLI) 섹션을 참조하세요.

3. 연결된 액세스 토큰을 보려면 list-source-credentials 명령을 실행합니다.

```
aws codebuild list-source-credentials
```

JSON 형식의 `sourceCredentialsInfos` 객체가 출력에 표시됩니다.

```
{
        "sourceCredentialsInfos": [
            {
                "authType": "auth-type",
                "serverType": "server-type",
                "arn": "arn"
            }
        ]
    }
```

`sourceCredentialsObject`에는 연결된 소스 자격 증명 정보 목록이 포함되어 있습니다.

- `authType`은 자격 증명에서 사용하는 인증 유형입니다. `OAUTH`, `BASIC_AUTH`, `PERSONAL_ACCESS_TOKEN`, `CODECONNECTIONS` 또는 `SECRETS_MANAGER`일 수 있습니다.

- `serverType`은 소스 공급자의 유형입니다. `GITHUB`, `GITHUB_ENTERPRISE`, `BITBUCKET`, `GITLAB` 또는 `GITLAB_SELF_MANAGED`일 수 있습니다.
- `arn`은 토큰의 ARN입니다.

4. 소스 공급자와의 연결을 해제하고 액세스 토큰을 삭제하려면 해당 ARN을 지정하여 delete-source-credentials 명령을 실행합니다.

```
aws codebuild delete-source-credentials --arn arn-of-your-credentials
```

JSON 형식의 데이터가 반환되고 삭제된 자격 증명의 ARN이 표시됩니다.

```
{
    "arn": "arn:aws:codebuild:region:account-id:token/server-type"
    }
```

GitHub OAuth 앱

OAuth를 사용하여 GitHub 연결(콘솔)

콘솔에서 OAuth 앱을 사용하여 프로젝트를 GitHub에 연결하려면 프로젝트를 생성할 때 다음과 같이 합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 섹션을 참조하세요.

1. 소스 공급자에서 GitHub를 선택합니다.
2. 자격 증명에서 다음 중 하나를 수행합니다.
   - 계정 자격 증명을 사용하여 모든 프로젝트에 계정의 기본 소스 자격 증명을 적용하려면 선택합니다.
     a. GitHub에 연결되지 않은 경우 계정 자격 증명 관리를 선택합니다.
     b. 자격 증명 유형에서 OAuth 앱을 선택합니다.
   - 서비스에 계정 수준 자격 증명을 사용하기로 선택한 경우 토큰을 저장하는 데 사용할 서비스를 선택하고 다음을 수행합니다.
     a. Secrets Manager를 사용하도록 선택한 경우 기존 보안 암호 연결을 사용하거나 새 보안 암호를 생성하고 저장을 선택할 수 있습니다. 새 보안 암호를 생성하는 방법에 대한 자세한 내용은 섹션을 참조하세요Secrets Manager 보안 암호에 토큰 생성 및 저장.
     b. CodeBuild를 사용하도록 선택한 경우 저장을 선택합니다.

- 사용자 지정 소스 자격 증명을 사용하여 계정의 자격 증명 설정을 재정의하려면 이 프로젝트에 대해서만 자격 증명 재정의 사용을 선택합니다.

  a. 채워진 자격 증명 목록에서 OAuth 앱 아래의 옵션 중 하나를 선택합니다.

  b. 설명에서 새 OAuth 앱 토큰 연결 생성을 선택하여 새 OAuth 앱 토큰을 생성할 수도 있습니다.

승인된 OAuth 앱을 검토하려면 GitHub의 애플리케이션으로 이동하여 aws-codesuite에서 소유한 `AWS CodeBuild (region)`이라는 애플리케이션이 나열되어 있는지 확인합니다.

CodeBuild의 Bitbucket 액세스

Bitbucket의 경우 액세스 토큰, 앱 암호, API 토큰, OAuth 앱 또는 Bitbucket 연결을 사용하여 소스 공급자에 액세스합니다.

주제

- Bitbucket 앱 연결
- Bitbucket 앱 암호, API 토큰 또는 액세스 토큰
- Bitbucket OAuth 앱

Bitbucket 앱 연결

Bitbucket을 사용하여 CodeBuild에 연결할 수 있습니다. Bitbucket 앱 연결은 AWS CodeConnections를 통해 지원됩니다.

> **Note**
>
> AWS CodeConnections 는 CodeBuild보다 적은 리전에서 사용할 수 있습니다. CodeBuild에서 리전 간 연결을 사용할 수 있습니다. 옵트인 리전에서 생성된 연결은 다른 리전에서 사용할 수 없습니다. 자세한 내용은 AWS CodeConnections 엔드포인트 및 할당량을 참조하세요.

주제

- 1단계:Bitbucket에 대한 연결 생성(콘솔)
- 2단계: CodeBuild 프로젝트에 연결을 사용할 IAM 역할 액세스 권한 부여
- 3단계: 새 연결을 사용하도록 CodeBuild 구성

- Bitbucket OAuth 교체 새로 고침 토큰

1단계:Bitbucket에 대한 연결 생성(콘솔)

다음 단계를 통해 CodeBuild 콘솔을 사용하여 Bitbucket의 프로젝트에 대한 연결을 추가할 수 있습니다.

Bitbucket에 대한 연결을 생성하려면

- 개발자 도구 사용 설명서의 Bitbucket에 대한 연결 생성을 위한 지침을 따릅니다.

> **Note**
>
> 계정에서 기존 연결을 생성하거나 사용하는 대신 다른 AWS 계정에서 공유된 연결을 사용할 수 있습니다. 자세한 내용은 AWS 계정과 연결 공유를 참조하세요.

2단계: CodeBuild 프로젝트에 연결을 사용할 IAM 역할 액세스 권한 부여

연결에서 제공한 Bitbucket 토큰을 사용할 수 있는 IAM 역할 액세스 권한을 CodeBuild 프로젝트에 부여할 수 있습니다.

CodeBuild 프로젝트에 IAM 역할 액세스 권한을 부여하려면

1. CodeBuild 프로젝트의 CodeBuild가 다른 AWS 서비스와 상호 작용하도록 허용에 대한 지침에 따라 CodeBuild 프로젝트의 IAM 역할을 생성합니다.
2. 지침에 따라 CodeBuild 프로젝트 역할에 다음 IAM 정책을 추가하여 연결에 대한 액세스 권한을 부여합니다.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codeconnections:GetConnectionToken",
                "codeconnections:GetConnection"
```

```
            ],
            "Resource": [
                "arn:aws:iam::*:role/Service*"
            ]
        }
    ]
}
```

3단계: 새 연결을 사용하도록 CodeBuild 구성

연결을 계정 수준 자격 증명으로 구성하고 프로젝트에서 사용할 수 있습니다.

AWS Management Console

에서 연결을 계정 수준 자격 증명으로 구성하려면 AWS Management Console

1. 소스 공급자에서 Bitbucket을 선택합니다.
2. 자격 증명에서 다음 중 하나를 수행합니다.
   - 기본 소스 자격 증명을 선택하여 계정의 기본 소스 자격 증명을 사용하여 모든 프로젝트에 적용합니다.
     a. Bitbucket에 연결되지 않은 경우 기본 소스 자격 증명 관리를 선택합니다.
     b. 자격 증명 유형에서를 선택합니다AWS CodeConnections.
     c. 연결에서 기존 연결을 선택하거나 연결을 새로 생성합니다.
   - 사용자 지정 소스 자격 증명을 선택하여 사용자 지정 소스 자격 증명을 사용하여 계정의 기본 설정을 재정의합니다.
     a. 자격 증명 유형에서를 선택합니다AWS CodeConnections.
     b. 연결에서 기존 연결을 선택하거나 연결을 새로 생성합니다.

AWS CLI

에서 연결을 계정 수준 자격 증명으로 구성하려면 AWS CLI

- 터미널(Linux, macOS, Unix) 또는 명령 프롬프트(Windows)를 엽니다. AWS CLI 를 사용하여 연결을 `--token` 위해 `--auth-type`, `--server-type`및를 지정하여 import-source-credentials 명령을 실행합니다.

다음 명령을 사용합니다.

```
aws codebuild import-source-credentials --auth-type CODECONNECTIONS --server-
type BITBUCKET --token <connection-arn>
```

CodeBuild 프로젝트에서 여러 토큰을 설정하는 방법에 대한 자세한 내용은 섹션을 참조하세요여러 토큰을 소스 수준 자격 증명으로 구성.

Bitbucket OAuth 교체 새로 고침 토큰

Atlassian은 이제 Bitbucket OAuth에 대해 일회용 교체 새로 고침 토큰을 적용합니다. 새로 고침 토큰을 사용할 때마다 Bitbucket은 토큰을 무효화하고 새 토큰을 반환합니다. 자세한 내용은 Atlassian 개발자 웹 사이트의 Bitbucket OAuth 일회용 새로 고침 토큰(CHANGE-3052)을 참조하세요.

Secrets Manager 저장 자격 증명에 필요한 작업

Secrets Manager에 Bitbucket OAuth 자격 증명을 저장하는 경우 CodeBuild 서비스 역할에 `secretsmanager:PutSecretValue` 권한을 추가합니다. 이 권한을 사용하면 CodeBuild는 각 토큰 새로 고침 후 업데이트된 새로 고침 토큰을 보안 암호에 다시 쓸 수 있습니다. 이 권한이 없으면 토큰 새로 고침 후 다음 빌드가 실패합니다.

다음 예제 IAM 정책 문은 필요한 권한을 부여합니다.

```
{
    "Effect": "Allow",
    "Action": "secretsmanager:PutSecretValue",
    "Resource": "arn:aws:secretsmanager:*:*:secret:codebuild/*"
}
```

CodeBuild 관리형 옵션을 사용하는 경우 변경할 필요가 없습니다. CodeBuild는 토큰 교체를 자동으로 처리합니다.

로 마이그레이션 AWS CodeConnections (권장)

Bitbucket 통합 AWS CodeConnections 을 위해 로 마이그레이션하는 것이 좋습니다.는 교체 새로 고침 토큰이 영향을 미치지 않는 별도의 권한 부여 메커니즘을 AWS CodeConnections 사용합니다. 토큰을 관리하거나 권한을 추가할 필요가 없습니다.

로 마이그레이션하려면 다음 단계를 AWS CodeConnections완료합니다.

1. 개발자 도구 콘솔을 엽니다. 설정, 연결을 선택한 다음 연결 생성을 선택합니다. Bitbucket을 공급자로 선택합니다.
2. Bitbucket 워크스페이스에서 AWS CodeConnections 앱을 승인합니다. 이를 위해서는 Bitbucket에서 워크스페이스 관리 권한이 필요합니다.
3. CodeBuild 서비스 역할에 `codeconnections:GetConnectionToken` 및 권한을 추가합니다`codeconnections:GetConnection`.
4. 연결을 기본 자격 증명으로 설정합니다.

```
aws codebuild import-source-credentials --auth-type CODECONNECTIONS --server-type
 BITBUCKET --token <connection-arn>
```

5. 기본 자격 증명을 사용하는 모든 프로젝트는 자동으로 전환됩니다. 사용자 지정 프로젝트별 자격 증명을 사용하는 프로젝트를 개별적으로 업데이트합니다. 콘솔 또는 `UpdateProject` API를 사용합니다.

Bitbucket 앱 암호, API 토큰 또는 액세스 토큰

Bitbucket 앱 암호는 더 이상 사용되지 않습니다.

Atlassian은 API 토큰을 위해 Bitbucket 앱 암호를 더 이상 사용하지 않습니다. CodeBuild에서 기본 인증을 계속 사용하려면 Atlassian 계정 설정에서 API 토큰을 생성합니다. CodeBuild에 연결할 때 Atlassian 계정과 연결된 이메일 주소를 사용자 이름으로 사용합니다. CodeBuild는 Git 작업 및 Bitbucket API 호출에 올바른 자격 증명을 자동으로 사용합니다.
자세한 내용은 Atlassian 웹 사이트의 API 토큰 사용을 참조하세요.

Bitbucket OAuth 토큰 교체

Bitbucket OAuth 자격 증명을 사용하는 경우 로 마이그레이션하는 것이 좋습니다Bitbucket 앱 연결. Secrets Manager에서 OAuth를 계속 사용할 때 필요한 변경 사항에 대한 자세한 내용은 섹션을 참조하세요Bitbucket OAuth 앱.

사전 조건

시작하기 전에 Bitbucket 앱 암호, API 토큰 또는 액세스 토큰에 적절한 권한 범위를 추가해야 합니다.

Bitbucket의 경우 앱 암호, API 토큰 또는 액세스 토큰의 범위는 다음과 같아야 합니다.

- repository:read: 권한 있는 사용자가 액세스할 수 있는 모든 리포지토리에 대한 읽기 액세스 권한을 부여합니다.
- pullrequest:read: pull 요청에 대한 읽기 액세스를 부여합니다. 프로젝트에 Bitbucket 웹후크가 있는 경우 앱 암호 또는 액세스 토큰이 이 범위를 가져야 합니다.
- webhook: Webhook에 대한 액세스를 부여합니다. 프로젝트에 웹후크 작업이 있는 경우 앱 암호 또는 액세스 토큰이 범위를 가져야 합니다.
- 계정: 사용자의 계정 정보에 대한 읽기 액세스 권한을 부여합니다.

자세한 내용은 Bitbucket 웹사이트의 Bitbucket 클라우드 REST API에 대한 범위 및 Bitbucket 클라우드의 OAuth를 참조하십시오.

앱 암호 또는 API 토큰을 사용하여 Bitbucket 연결(콘솔)

콘솔을 사용하여 앱 암호 또는 API 토큰을 사용하여 프로젝트를 Bitbucket에 연결하려면 프로젝트를 생성할 때 다음을 수행합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 섹션을 참조하세요.

> **Note**
>
> API 토큰을 사용하는 경우 Atlassian 계정과 연결된 이메일 주소를 사용자 이름으로 입력합니다.

1. 소스 공급자에서 Bitbucket을 선택합니다.
2. 자격 증명에서 다음 중 하나를 수행합니다.
   - 계정 자격 증명을 사용하여 모든 프로젝트에 계정의 기본 소스 자격 증명을 적용하려면 선택합니다.
     a. Bitbucket에 연결되지 않은 경우 계정 자격 증명 관리를 선택합니다.
     b. 자격 증명 유형에서 API 토큰(이전에는 앱 암호로 표시됨)을 선택합니다.
   - 서비스에 계정 수준 자격 증명을 사용하기로 선택한 경우 토큰을 저장하는 데 사용할 서비스를 선택하고 다음을 수행합니다.
     a. Secrets Manager를 사용하도록 선택한 경우 기존 보안 암호 연결을 사용하거나 새 보안 암호를 생성하고 저장을 선택할 수 있습니다. 새 보안 암호를 생성하는 방법에 대한 자세한 내용은 섹션을 참조하세요Secrets Manager 보안 암호에 토큰 생성 및 저장.

b. CodeBuild를 사용하도록 선택한 경우 Bitbucket 사용자 이름과 앱 암호를 입력한 다음 저장을 선택합니다. Bitbucket API 토큰의 경우 Atlassian 계정 이메일 주소와 API 토큰을 입력합니다.

- 사용자 지정 소스 자격 증명을 사용하여 계정의 자격 증명 설정을 재정의하려면 이 프로젝트에 대해서만 자격 증명 재정의 사용을 선택합니다.

  a. 채워진 자격 증명 목록에서 앱 암호 또는 API 토큰 아래의 옵션 중 하나를 선택합니다.

  b. 설명에서 연결 생성 링크를 선택하여 새 앱 암호 또는 API 토큰 연결을 생성할 수도 있습니다.

액세스 토큰을 사용하여 Bitbucket에 연결(콘솔)

콘솔에서 액세스 토큰을 사용하여 프로젝트를 Bitbucket에 연결하려면 프로젝트를 생성할 때 다음과 같이 합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 섹션을 참조하세요.

1. 소스 공급자에서 Bitbucket을 선택합니다.
2. 자격 증명에서 다음 중 하나를 수행합니다.
   - 계정 자격 증명을 사용하여 모든 프로젝트에 계정의 기본 소스 자격 증명을 적용하려면 선택합니다.

     a. Bitbucket에 연결되지 않은 경우 계정 자격 증명 관리를 선택합니다.

     b. 자격 증명 유형에서 개인 액세스 토큰을 선택합니다.
   - 서비스에 계정 수준 자격 증명을 사용하기로 선택한 경우 토큰을 저장하는 데 사용할 서비스를 선택하고 다음을 수행합니다.

     a. Secrets Manager를 사용하도록 선택한 경우 기존 보안 암호 연결을 사용하거나 새 보안 암호를 생성하고 저장을 선택할 수 있습니다. 새 보안 암호를 생성하는 방법에 대한 자세한 내용은 섹션을 참조하세요Secrets Manager 보안 암호에 토큰 생성 및 저장.

     b. CodeBuild를 사용하도록 선택한 경우 Bitbucket 개인 액세스 토큰을 입력하고 저장을 선택합니다.
   - 사용자 지정 소스 자격 증명을 사용하여 계정의 자격 증명 설정을 재정의하려면 이 프로젝트에 대해서만 자격 증명 재정의 사용을 선택합니다.

     a. 채워진 자격 증명 목록에서 개인 액세스 토큰 아래의 옵션 중 하나를 선택합니다.

     b. 설명에서 새 개인 액세스 토큰 연결 생성을 선택하여 새 개인 액세스 토큰을 생성할 수도 있습니다.

앱 암호 또는 액세스 토큰을 사용하여 Bitbucket 연결(CLI)

다음 단계에 따라 AWS CLI 를 사용하여 앱 암호 또는 액세스 토큰을 사용하여 프로젝트를 Bitbucket에 연결합니다. 와 AWS CLI 함께를 사용하는 방법에 대한 자세한 내용은 단원을 AWS CodeBuild참조하십시오명령줄 참조.

1. import-source-credentials 명령을 실행합니다.

```
aws codebuild import-source-credentials --generate-cli-skeleton
```

JSON 형식 데이터가 출력에 표시됩니다. 가 설치된 로컬 컴퓨터 또는 인스턴스의 위치에 있는 파일(예: *import-source-credentials.json*)에 데이터를 복사 AWS CLI 합니다. 복사된 데이터를 다음과 같이 수정하고 결과를 저장합니다.

```
{
    "serverType": "BITBUCKET",
    "authType": "auth-type",
    "shouldOverwrite": "should-overwrite",
    "token": "token",
    "username": "username"
    }
```

다음을 바꿉니다.

- *server-type*: 필수 값. 이 자격 증명에 사용되는 소스 공급자. 유효한 값은 GITHUB, BITBUCKET, GITHUB_ENTERPRISE, GITLAB 및 GITLAB_SELF_MANAGED입니다.
- *auth-type*: 필수 값. 리포지토리에 연결하는 데 사용되는 인증 유형입니다. 유효한 값은 OAUTH, BASIC_AUTH, Personal_ACCESS_TOKEN, CODECONNECTIONS 및 SECRETS_MANAGER입니다. GitHub의 경우 PERSONAL_ACCESS_TOKEN만 허용됩니다. BASIC_AUTH는 Bitbucket 앱 암호 또는 API 토큰에서만 허용됩니다.
- *should-overwrite*: 선택적 값입니다. 리포지토리 소스 자격 증명을 덮어쓰지 않도록 하려면 `false`로 설정합니다. 리포지토리 소스 자격 증명을 덮어쓰려면 `true`로 설정합니다. 기본값은 `true`입니다.
- *token*: 필수 값. GitHub 또는 GitHub Enterprise Server에서는 개인용 액세스 토큰을 말합니다. Bitbucket의 경우 개인 액세스 토큰, 앱 암호 또는 API 토큰입니다. 인증 유형 CODECONNECTIONS의 경우 연결 ARN입니다. 인증 유형 SECRETS_MANAGER의 경우 보안 암호 ARN입니다.

- *username*: 선택 사항 값. GitHub 및 GitHub Enterprise Server 소스 공급자의 경우 이 파라미터가 무시됩니다. Bitbucket 앱 암호의 경우 Bitbucket 사용자 이름입니다. Bitbucket API 토큰의 경우 Atlassian 계정과 연결된 이메일 주소입니다.

2. 앱 암호, API 토큰 또는 액세스 토큰으로 계정을 연결하려면 1단계에서 저장한 `import-source-credentials.json` 파일이 포함된 디렉터리로 전환하고 import-source-credentials 명령을 다시 실행합니다.

```
aws codebuild import-source-credentials --cli-input-json file://import-source-credentials.json
```

JSON 형식 데이터와 Amazon 리소스 이름(ARN)이 출력에 표시됩니다.

```
{
    "arn": "arn:aws:codebuild:region:account-id:token/server-type"
    }
```

> **Note**
>
> import-source-credentials 명령을 동일한 서버 유형과 인증 유형으로 다시 실행하면 저장된 액세스 토큰이 업데이트됩니다.

계정이 앱 암호 또는 API 토큰으로 연결된 후를 사용하여 CodeBuild 프로젝트를 `create-project` 생성할 수 있습니다. 자세한 내용은 빌드 프로젝트 생성(AWS CLI) 섹션을 참조하세요.

3. 연결된 앱 암호 또는 액세스 토큰을 보려면 list-source-credentials 명령을 실행합니다.

```
aws codebuild list-source-credentials
```

JSON 형식의 `sourceCredentialsInfos` 객체가 출력에 표시됩니다.

```
{
        "sourceCredentialsInfos": [
            {
                "authType": "auth-type",
                "serverType": "BITBUCKET",
                "arn": "arn"
            }
        ]
```

```
    }
```

`sourceCredentialsObject`에는 연결된 소스 자격 증명 정보 목록이 포함되어 있습니다.

- `authType`은 자격 증명에서 사용하는 인증 유형입니다. `OAUTH`, `BASIC_AUTH`, `PERSONAL_ACCESS_TOKEN`, `CODECONNECTIONS` 또는 `SECRETS_MANAGER`일 수 있습니다.
- `serverType`은 소스 공급자의 유형입니다. `GITHUB`, `GITHUB_ENTERPRISE`, `BITBUCKET`, `GITLAB` 또는 `GITLAB_SELF_MANAGED`일 수 있습니다.
- `arn`은 토큰의 ARN입니다.

4. 소스 공급자와의 연결을 해제하고 앱 암호 또는 액세스 토큰을 삭제하려면 해당 ARN을 지정하여 delete-source-credentials 명령을 실행합니다.

```
aws codebuild delete-source-credentials --arn arn-of-your-credentials
```

JSON 형식의 데이터가 반환되고 삭제된 자격 증명의 ARN이 표시됩니다.

```
{
    "arn": "arn:aws:codebuild:region:account-id:token/server-type"
    }
```

Bitbucket OAuth 앱

⚠ Bitbucket OAuth 토큰 교체에 필요한 작업

Atlassian은 이제 Bitbucket OAuth에 대해 일회용 교체 새로 고침 토큰을 적용합니다. 토큰 관리를 완전히 제거할 Bitbucket 앱 연결수 있도록 로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 지침은 섹션을 참조하세요로 마이그레이션 AWS CodeConnections (권장).
Secrets Manager에서 OAuth를 계속 사용하려면 CodeBuild 서비스 역할에 `secretsmanager:PutSecretValue` 권한을 추가합니다. 자세한 내용은 Secrets Manager 저장 자격 증명에 필요한 작업 섹션을 참조하세요.
CodeBuild 관리형 옵션을 사용하는 경우 변경할 필요가 없습니다. CodeBuild는 토큰 교체를 자동으로 처리합니다.

OAuth를 사용하여 Bitbucket 연결(콘솔)

콘솔에서 OAuth 앱을 사용하여 프로젝트를 Bitbucket에 연결하려면 프로젝트를 생성할 때 다음과 같이 합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 섹션을 참조하세요.

1. 소스 공급자에서 Bitbucket을 선택합니다.
2. 자격 증명에서 다음 중 하나를 수행합니다.
   - 계정 자격 증명을 사용하여 모든 프로젝트에 계정의 기본 소스 자격 증명을 적용하려면 선택합니다.
     a. Bitbucket에 연결되지 않은 경우 계정 자격 증명 관리를 선택합니다.
     b. 자격 증명 유형에서 OAuth 앱을 선택합니다.
   - 서비스에 계정 수준 자격 증명을 사용하기로 선택한 경우 토큰을 저장하는 데 사용할 서비스를 선택하고 다음을 수행합니다.
     a. Secrets Manager를 사용하도록 선택한 경우 기존 보안 암호 연결을 사용하거나 새 보안 암호를 생성하고 저장을 선택할 수 있습니다. 새 보안 암호 생성에 대한 자세한 내용은 섹션을 참조하세요Secrets Manager 보안 암호에 토큰 생성 및 저장.
     b. CodeBuild를 사용하도록 선택한 경우 저장을 선택합니다.
   - 사용자 지정 소스 자격 증명을 사용하여 계정의 자격 증명 설정을 재정의하려면 이 프로젝트에 대해서만 자격 증명 재정의 사용을 선택합니다.
     a. 채워진 자격 증명 목록에서 OAuth 앱 아래의 옵션 중 하나를 선택합니다.
     b. 설명에서 새 OAuth 앱 토큰 연결 생성을 선택하여 새 OAuth 앱 토큰을 생성할 수도 있습니다.

승인된 OAuth 앱을 검토하려면 Bitbucket에서 애플리케이션 권한 부여로 이동하여 `AWS CodeBuild (region)` 이름이 지정된 애플리케이션이 나열되었는지 확인합니다.

CodeBuild의 GitLab 액세스

GitLab의 경우 GitLab 연결을 사용하여 소스 공급자에 액세스합니다.

주제

- GitLab에 CodeBuild 연결

GitLab에 CodeBuild 연결

연결을 사용하면를 사용하여 타사 공급자를 AWS 리소스와 연결하는 구성을 승인하고 설정할 수 있습니다 AWS CodeConnections. 타사 리포지토리를 빌드 프로젝트의 소스로 연결하려면 연결을 사용합니다.

CodeBuild에서 GitLab 또는 GitLab Self Managed 소스 공급자를 추가하려면 다음 중 하나를 선택할 수 있습니다.

- CodeBuild 콘솔 빌드 프로젝트 생성 마법사 또는 소스 편집 페이지를 사용하여 GitLab 또는 GitLab Self Managed 공급자 옵션을 선택합니다. 소스 공급자를 추가하려면 GitLab에 대한 연결 생성(콘솔) 섹션을 참조하세요. 콘솔을 사용하면 연결 리소스를 만들 수 있습니다.
- 연결 리소스를 생성하려면 GitLab에 대한 연결 생성(CLI) 섹션을 참조하여 CLI를 사용하여 연결 리소스를 생성합니다.

> **Note**
>
> 설정의 개발자 도구 콘솔을 사용하여 연결을 생성할 수도 있습니다. 연결 생성을 참조하세요.

> **Note**
>
> GitLab에서 이 연결 설치를 승인하면 계정에 액세스하여 데이터를 처리할 수 있는 권한을 서비스에 부여하고 언제든지 애플리케이션을 제거하여 권한을 취소할 수 있습니다.

GitLab에 대한 연결 생성

이 섹션에서는 GitLab을 CodeBuild에 연결하는 방법을 설명합니다. GitLab 연결에 대한 자세한 정보는 GitLab에 CodeBuild 연결 섹션을 참조하세요.

시작하기 전:

- GitLab 계정이 이미 생성되어 있어야 합니다.

> **Note**
>
> 연결은 연결을 만들고 권한을 부여하는 데 사용된 계정이 소유한 리포지토리에 대한 액세스 권한만 제공합니다.

> **Note**
>
> GitLab에서 소유자 역할을 가진 리포지토리에 대한 연결을 생성한 다음 CodeBuild와 같은 리소스가 있는 리포지토리에서 해당 연결을 사용할 수 있습니다. 그룹 내 리포지토리의 경우 그룹 소유자가 아니어도 됩니다.

- 빌드 프로젝트에 대한 소스를 지정하려면 GitLab에서 리포지토리가 이미 생성되어 있어야 합니다.

주제

- GitLab에 대한 연결 생성(콘솔)
- GitLab에 대한 연결 생성(CLI)

GitLab에 대한 연결 생성(콘솔)

다음 단계를 사용하여 CodeBuild 콘솔을 사용하여 GitLab의 프로젝트(리포지토리)에 대한 연결을 추가할 수 있습니다.

> **Note**
>
> 계정에서 기존 연결을 생성하거나 사용하는 대신 다른 AWS 계정에서 공유된 연결을 사용할 수 있습니다. 자세한 내용은 AWS 계정과 연결 공유를 참조하세요.

빌드 프로젝트를 생성하거나 편집하려면

1. CodeBuild 콘솔에 로그인합니다.
2. 다음 중 하나를 선택합니다.
   - 빌드 프로젝트 생성을 선택합니다. 빌드 프로젝트 만들기(콘솔)의 단계에 따라 첫 번째 화면을 완료하고 소스 섹션의 소스 공급자에서 GitLab을 선택합니다.

- 기존 빌드 프로젝트를 편집하려면 선택합니다. 편집을 선택한 다음 소스를 선택합니다. 소스 편집 페이지의 소스 공급자에서 GitLab을 선택합니다.

3. 다음 중 하나를 선택합니다.

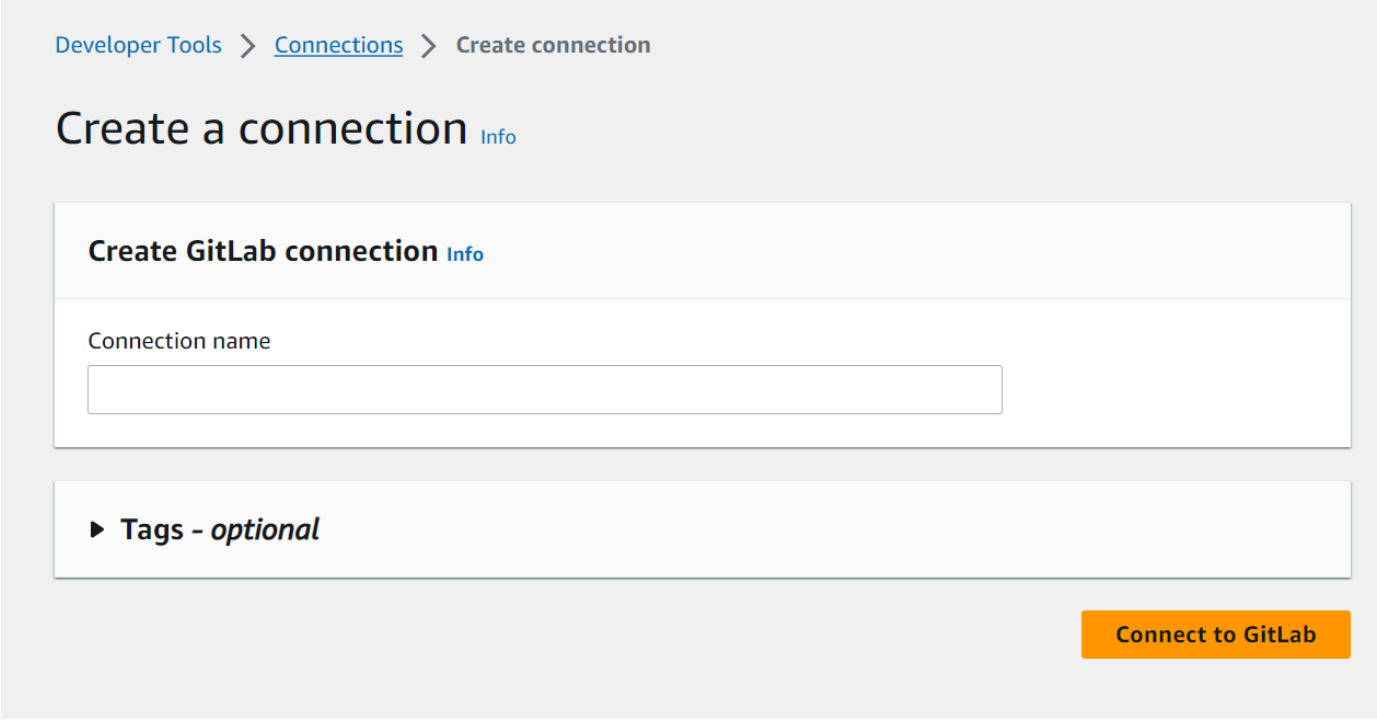
- 연결에서 기본 연결을 선택합니다. 기본 연결은 모든 프로젝트에서 기본 GitLab 연결을 적용합니다.
- 연결에서 사용자 지정 연결을 선택합니다. 사용자 지정 연결은 계정의 기본 설정을 재정의하는 사용자 지정 GitLab 연결을 적용합니다.

4. 다음 중 하나를 수행하세요.

- 기본 연결 또는 사용자 지정 연결에서 공급자와의 연결을 아직 생성하지 않은 경우 새 GitLab 연결 생성을 선택합니다. 5단계로 이동하여 연결을 생성합니다.
- 연결에서 공급자와의 연결을 이미 생성한 경우 연결을 선택합니다. 10단계로 이동합니다.

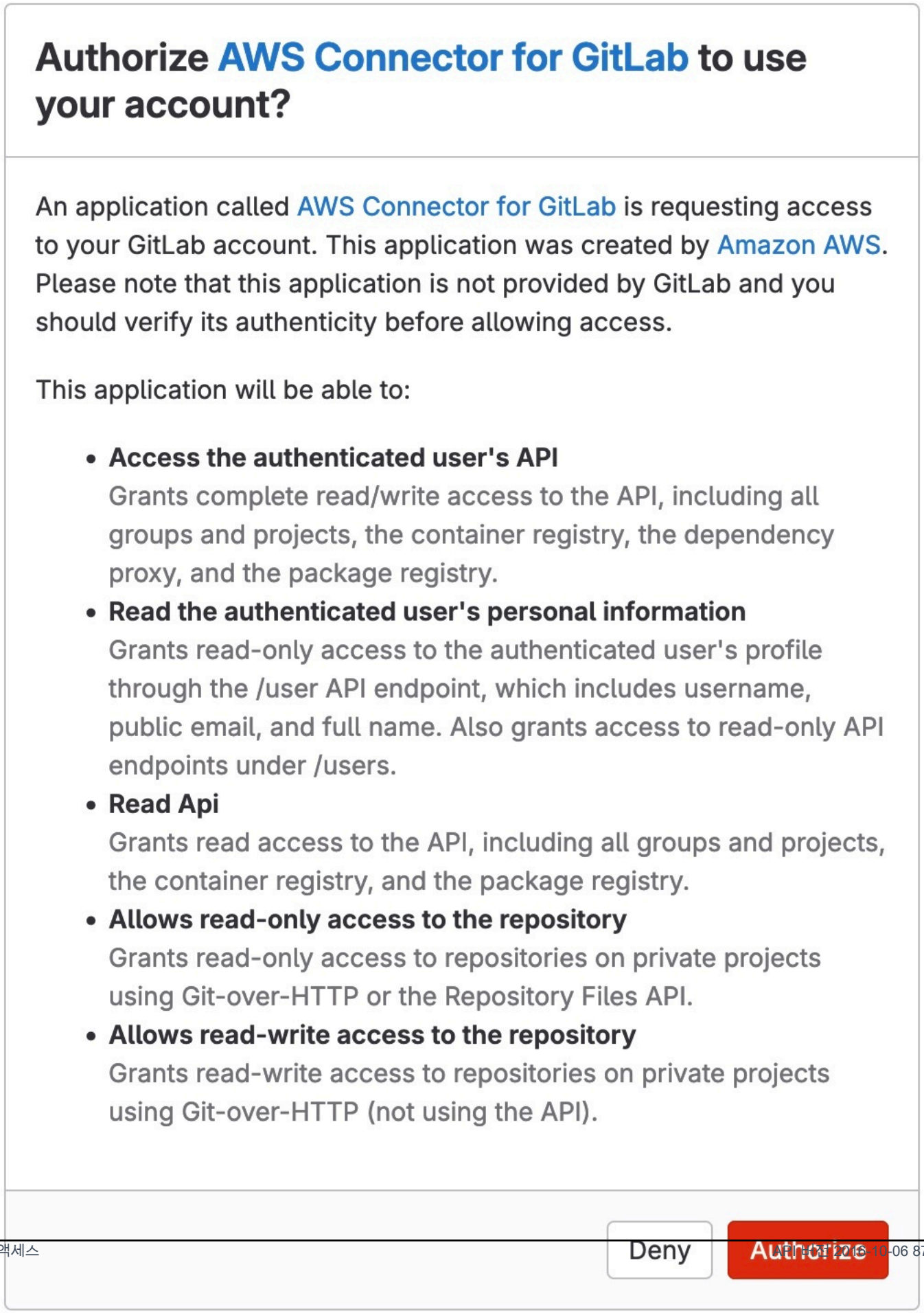
> **Note**
>
> GitLab 연결이 생성되기 전에 팝업 창을 닫으면 페이지를 새로 고쳐야 합니다.

5. GitLab 리포지토리에 대한 연결을 생성하려면 공급자 선택에서 GitLab을 선택합니다. [연결 이름(Connection name)]에 생성하려는 연결의 이름을 입력합니다. GitLab에 연결을 선택합니다.



6. GitLab의 로그인 페이지가 표시되면 자격 증명을 사용하여 로그인한 다음 로그인을 선택합니다.
7. 처음 연결을 승인하는 경우에는 GitLab 계정에 액세스하기 위한 연결 승인을 요청하는 메시지와 함께 권한 부여 페이지가 표시됩니다.

   Authorize를 선택합니다.


Authorize AWS Connector for GitLab to use your account?
An application called AWS Connector for GitLab is requesting access to your GitLab account. This application was created by Amazon AWS. Please note that this application is not provided by GitLab and you should verify its authenticity before allowing access.
This application will be able to:
Access the authenticated user's API
Grants complete read/write access to the API, including all groups and projects, the container registry, the dependency proxy, and the package registry.
Read the authenticated user's personal information
Grants read-only access to the authenticated user's profile through the /user API endpoint, which includes username, public email, and full name. Also grants access to read-only API endpoints under /users.
Read Api
Grants read access to the API, including all groups and projects, the container registry, and the package registry.
Allows read-only access to the repository
Grants read-only access to repositories on private projects using Git-over-HTTP or the Repository Files API.
Allows read-write access to the repository
Grants read-write access to repositories on private projects using Git-over-HTTP (not using the API).
Deny
Authorize

8. 브라우저가 연결 콘솔 페이지로 돌아갑니다. GitLab 연결 설정에서 연결 이름에 새 연결이 표시됩니다.
9. 연결을 선택합니다.

   GitLab 연결이 성공적으로 생성되면 성공 배너가 상단에 표시됩니다.
10. 빌드 프로젝트 생성 페이지의 기본 연결 또는 사용자 지정 연결 드롭다운 목록에서 연결 ARN이 나열되어 있는지 확인합니다. 나열되지 않은 경우 새로 고침 버튼을 선택하여 표시합니다.
11. 리포지토리에서 네임스페이스와 함께 프로젝트 경로를 지정하여 GitLab의 프로젝트 이름을 선택합니다. 예를 들어 그룹 수준 리포지토리의 경우 리포지토리 이름을 `group-name/repository-name` 형식으로 입력합니다. 경로와 네임스페이스에 대한 자세한 내용은 https://docs.gitlab.com/ee/api/projects.html#get-single-project의 `path_with_namespace` 필드를 참조하세요. GitLab의 네임스페이스에 대한 자세한 내용은 https://docs.gitlab.com/ee/user/namespace/를 참조하세요.

    > **Note**
    >
    > GitLab에 있는 그룹의 경우 네임스페이스를 사용하여 프로젝트 경로를 수동으로 지정해야 합니다. 예를 들어 그룹 `mygroup`의 `myrepo`라는 리포지토리의 경우 `mygroup/myrepo` 형식으로 입력합니다. GitLab의 URL에서 네임스페이스를 사용하여 프로젝트 경로를 찾을 수 있습니다.

12. 소스 버전 - 선택 사항에서 pull 요청, 분기, 커밋 ID, 태그 또는 참조와 커밋 ID를 입력합니다. 자세한 내용은 를 사용한 소스 버전 샘플 AWS CodeBuild 섹션을 참조하세요.

    > **Note**
    >
    > `811dd1ba1aba14473856cee38308caed7190c0d` 또는 `5392f7`과 같이 커밋 ID처럼 보이지 않는 Git 분기 이름을 선택하는 것이 좋습니다. 이렇게 하면 실제 커밋과 Git 체크아웃이 충돌하는 것을 방지할 수 있습니다.

13. Git 복제 깊이 - 선택 사항에서 이력이 지정된 커밋 수로 잘린 부분 복제를 생성할 수 있습니다. 전체 복제가 필요할 경우 전체를 선택합니다.
14. 빌드 상태 - 선택 사항에서 빌드 시작 및 완료 상태가 소스 공급자에게 보고되도록 하려면 빌드가 시작되고 완료될 때 소스 공급자에게 빌드 상태 보고를 선택합니다.

소스 공급자에게 빌드 상태를 보고하려면 소스 공급자와 연결된 사용자에게 리포지토리에 대한 쓰기 권한이 있어야 합니다. 사용자에게 쓰기 권한이 없는 경우 빌드 상태를 업데이트할 수 없습니다. 자세한 내용은 소스 공급자 액세스 섹션을 참조하세요.

GitLab에 대한 연결 생성(CLI)

AWS Command Line Interface (AWS CLI)를 사용하여 연결을 생성할 수 있습니다.

이렇게 하려면 create-connection 명령을 사용하세요.

> **Important**
>
> AWS CLI 또는를 통해 생성된 연결 AWS CloudFormation 은 기본적으로 PENDING 상태입니다. CLI 또는를 사용하여 연결을 생성한 후 콘솔을 CloudFormation사용하여 연결을 편집하여 상태를 로 설정합니다AVAILABLE.

연결을 생성하는 방법

- GitLab(CLI)에 대한 연결 생성을 위한 개발자 도구 콘솔 사용 설명서의 지침을 따릅니다.

교차 서비스 혼동된 대리인 방지

혼동된 대리자 문제는 작업을 수행할 권한이 없는 엔터티가 권한이 더 많은 엔터티에게 작업을 수행하도록 강요할 수 있는 보안 문제입니다. 에서 AWS교차 서비스 가장은 혼동된 대리자 문제를 초래할 수 있습니다. 교차 서비스 가장은 한 서비스(직접 호출하는 서비스)가 다른 서비스(직접 호출되는 서비스)를 직접 호출할 때 발생할 수 있습니다. 직접 호출하는 서비스는 다른 고객의 리소스에 대해 액세스 권한이 없는 방식으로 작동하게 권한을 사용하도록 조작될 수 있습니다. 이를 방지하기 위해 AWS 에서는 계정의 리소스에 대한 액세스 권한이 부여된 서비스 위탁자를 사용하여 모든 서비스에 대한 데이터를 보호하는 데 도움이 되는 도구를 제공합니다.

리소스 정책에서 aws:SourceArn 및 aws:SourceAccount 전역 조건 컨텍스트 키를 사용하여 리소스에 다른 서비스를 AWS CodeBuild 제공하는 권한을 제한하는 것이 좋습니다. 하나의 리소스만 교차 서비스 액세스와 연결되도록 허용하려는 경우 aws:SourceArn을 사용하세요. 해당 계정의 모든 리소스가 교차 서비스 사용과 연결되도록 허용하려는 경우 aws:SourceAccount을(를) 사용합니다.

혼동된 대리자 문제로부터 보호하는 가장 효과적인 방법은 리소스의 전체 ARN이 포함된 aws:SourceArn 전역 조건 컨텍스트 키를 사용하는 것입니다. 리소스의 전체

ARN을 모르거나 여러 리소스를 지정하는 경우, ARN의 알 수 없는 부분에 대해 와일드 카드 문자(*)를 포함한 `aws:SourceArn` 글로벌 조건 컨텍스트 키를 사용합니다. 예: `arn:aws:codebuild:*:123456789012:*`

만약 `aws:SourceArn` 값에 Amazon S3 버킷 ARN과 같은 계정 ID가 포함되어 있지 않은 경우, 권한을 제한하려면 두 글로벌 조건 컨텍스트 키를 모두 사용해야 합니다.

`aws:SourceArn`의 값은 CodeBuild 프로젝트 ARN이어야 합니다.

다음 예는 CodeBuild에서 `aws:SourceArn` 및 `aws:SourceAccount` 전역 조건 컨텍스트 키를 사용하여 혼동된 대리자 문제를 방지하는 방법을 보여 줍니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "codebuild.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceArn": "arn:aws:codebuild:us-
east-1:111122223333:project/MyProject"
                }
            }
        }
    ]
}
```

고급 주제

이 단원에서는 더욱 숙련된 AWS CodeBuild 사용자에게 유용한 몇 가지 고급 주제를 다룹니다.

주제

- 사용자가 CodeBuild와 상호 작용하도록 허용
- CodeBuild가 다른 AWS 서비스와 상호 작용하도록 허용
- 고객 관리형 키를 사용하여 빌드 출력 암호화
- 를 사용하여 CodeBuild와 상호 작용 AWS CLI
- 에 대한 명령줄 참조 AWS CodeBuild
- AWS SDKs 및 도구 참조 AWS CodeBuild
- AWS SDK에서이 서비스 사용
- AWS CodeBuild 엔드포인트 지정
- 와 AWS CodeBuild 함께 AWS CodePipeline 를 사용하여 코드를 테스트하고 빌드를 실행합니다.
- Codecov AWS CodeBuild 와 함께 사용
- Jenkins와 AWS CodeBuild 함께 사용
- 서버리스 애플리케이션 AWS CodeBuild 과 함께 사용
- Windows AWS CodeBuild 용에 대한 타사 알림
- CodeBuild 조건 키를 IAM 서비스 역할 변수로 사용하여 빌드 액세스 제어
- AWS CodeBuild 조건 키

사용자가 CodeBuild와 상호 작용하도록 허용

의 단계에 따라 AWS CodeBuild 에 처음 콘솔을 사용하여 시작하기 액세스하는 경우이 주제의 정보가 필요하지 않을 가능성이 높습니다. 그러나 CodeBuild를 계속 사용하면 조직의 다른 사용자 및 그룹에 CodeBuild와 상호 작용할 수 있는 기능을 부여하는 등의 작업을 수행할 수 있습니다.

IAM 사용자 또는 그룹이 상호 작용하도록 허용하려면 CodeBuild에 대한 액세스 권한을 부여 AWS CodeBuild해야 합니다. 이 섹션에서는 IAM 콘솔이나 AWS CLI를 사용하여 이를 수행하는 방법을 설명합니다.

AWS 루트 계정(권장되지 않음) 또는 계정의 관리자 사용자로 CodeBuild에 액세스하는 경우 다음 지침을 따를 필요가 AWS 없습니다.

AWS 루트 계정 및 관리자 사용자에 대한 자세한 내용은 사용 설명서의 AWS 계정 루트 사용자 및 첫 번째 AWS 계정 루트 사용자 및 그룹 생성을 참조하세요.

IAM 그룹 또는 사용자에게 CodeBuild 액세스 권한을 추가하려면(콘솔)

1. IAM 콘솔(https://console.aws.amazon.com/iam/)을 엽니다.

   다음 중 하나를 사용하여 AWS Management Console 에 이미 로그인했어야 합니다.

   - AWS 루트 계정. 이는 권장하지 않습니다. 자세한 내용은 사용 설명서의 AWS 계정 루트 사용자를 참조하세요.
   - AWS 계정의 관리자 사용자입니다. 자세한 내용은 사용 설명서의 첫 번째 AWS 계정 루트 사용자 및 그룹 생성을 참조하세요.
   - AWS 계정에서 다음과 같은 최소 작업 세트를 수행할 수 있는 권한이 있는 사용자:

   ```
   iam:AttachGroupPolicy
   iam:AttachUserPolicy
   iam:CreatePolicy
   iam:ListAttachedGroupPolicies
   iam:ListAttachedUserPolicies
   iam:ListGroups
   iam:ListPolicies
   iam:ListUsers
   ```

   자세한 내용은 사용 설명서의 IAM 정책 개요를 참조하세요.
2. 탐색 창에서 Policies를 선택합니다.
3. IAM 그룹 또는 IAM 사용자에게 사용자 지정 AWS CodeBuild 액세스 권한 세트를 추가하려면이 절차의 4단계로 건너뜁니다.

   IAM 그룹이나 IAM 사용자에게 기본 CodeBuild 액세스 권한 세트를 추가하려면 정책 유형, AWS 관리형을 선택한 후 다음을 수행합니다.

   - CodeBuild에 대한 전체 액세스 권한을 추가하려면 AWSCodeBuildAdminAccess 확인란을 선택하고 정책 작업을 선택한 다음, 연결을 선택합니다. 대상 IAM 그룹 및 사용자 옆의 확인란을 선택하고 정책 연결을 선택합니다. [AmazonS3ReadOnlyAccess] 및 [IAMFullAccess]라는 정책에 대해서도 이 절차를 반복합니다.
   - 빌드 프로젝트 관리를 제외한 모든 항목에 CodeBuild에 대한 액세스 권한을 추가하려면 AWSCodeBuildDeveloperAccess 확인란을 선택하고 정책 작업을 선택한 다음, 연결을

선택합니다. 대상 IAM 그룹 및 사용자 옆의 확인란을 선택하고 정책 연결을 선택합니다. [AmazonS3ReadOnlyAccess] 정책에 대해서도 이 절차를 반복합니다.

- CodeBuild에 대해 읽기 전용 액세스 권한을 추가하려면 AWSCodeBuildReadOnlyAccess 확인란을 선택합니다. 대상 IAM 그룹 및 사용자 옆의 확인란을 선택하고 정책 연결을 선택합니다. [AmazonS3ReadOnlyAccess] 정책에 대해서도 이 절차를 반복합니다.

이제 IAM 그룹 또는 사용자에게 기본 CodeBuild 액세스 권한 세트가 추가되었습니다. 이 절차의 나머지 단계는 건너뜁니다.

4. 정책 생성을 선택합니다.
5. [Create Policy] 페이지에서 [Create Your Own Policy] 옆의 [Select]를 선택합니다.
6. 정책 검토 페이지의 정책 이름에 정책 이름을 입력합니다(예: **CodeBuildAccessPolicy**). 다른 이름을 사용하는 경우 이 절차 전체에서 해당 이름을 사용해야 합니다.
7. 정책 문서에 다음을 입력한 다음 정책 생성을 선택합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "CodeBuildAccessPolicy",
            "Effect": "Allow",
            "Action": [
                "codebuild:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CodeBuildRolePolicy",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::111122223333:role/role-name"
        },
        {
            "Sid": "CloudWatchLogsAccessPolicy",
            "Effect": "Allow",
            "Action": [
```

```
                "logs:FilterLogEvents",
                "logs:GetLogEvents"
            ],
            "Resource": "*"
        },
        {
            "Sid": "S3AccessPolicy",
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:GetObject",
                "s3:List*",
                "s3:PutObject"
            ],
            "Resource": "*"
        },
        {
            "Sid": "S3BucketIdentity",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketAcl",
                "s3:GetBucketLocation"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 이 정책은 모든 CodeBuild 작업과 잠재적으로 많은 AWS 리소스에 대한 액세스를 허용합니다. 특정 CodeBuild 작업으로 권한을 제한하려면 CodeBuild 정책 명령에서 `codebuild:*`의 값을 변경합니다. 자세한 내용은 ID 및 액세스 관리 섹션을 참조하세요. 특정 AWS 리소스에 대한 액세스를 제한하려면 `Resource` 객체의 값을 변경합니다. 자세한 내용은 ID 및 액세스 관리 섹션을 참조하세요.

8. 탐색 창에서 [Groups] 또는 [Users]를 선택합니다.
9. 그룹 또는 사용자 목록에서 CodeBuild 액세스 권한을 추가하려는 IAM 그룹 또는 IAM 사용자의 이름을 선택합니다.
10. 그룹의 경우 그룹 설정 페이지의 권한 탭에서 관리형 정책을 확장한 다음 정책 연결을 선택합니다.

사용자의 경우 사용자 설정 페이지의 [Permissions] 탭에서 [Add permissions]를 선택합니다.

11. 그룹의 경우 정책 연결 페이지에서 CodeBuildAccessPolicy를 선택한 후 정책 연결을 선택합니다.

    사용자의 경우 권한 추가 페이지에서 기존 정책 직접 연결을 선택합니다. CodeBuildAccessPolicy, 다음: 검토를 차례로 선택한 후 권한 추가를 선택합니다.

CodeBuild 액세스 권한을 IAM 그룹 또는 사용자에게 추가하려면(AWS CLI)

1. 이전 절차에서 설명한 대로 IAM 엔터티 중 하나에 해당하는 AWS 액세스 키 및 AWS 보안 액세스 키 AWS CLI 로를 구성했는지 확인합니다. 자세한 내용은AWS Command Line Interface 사용 설명서의 AWS Command Line Interface설정을 참조하세요.
2. IAM 그룹 또는 IAM 사용자에게 사용자 지정 AWS CodeBuild 액세스 권한 세트를 추가하려면이 절차의 3단계로 건너뜁니다.

   IAM 그룹 또는 IAM 사용자에게 기본 CodeBuild 액세스 권한 세트를 추가하려면 다음을 수행합니다.

   IAM 그룹 또는 사용자에게 권한을 추가할 것인지에 따라 다음 명령 중 하나를 실행합니다.

   ```
   aws iam attach-group-policy --group-name group-name --policy-arn policy-arn

   aws iam attach-user-policy --user-name user-name --policy-arn policy-arn
   ```

   group-name 또는 *user-name*을 IAM 그룹 이름 또는 사용자 이름으로 변경하고 다음 정책 Amazon 리소스 이름(ARN)마다 한 번씩 *policy-arn*을 변경하는 등 명령을 3번 실행해야 합니다.

   - CodeBuild에 전체 액세스 권한을 추가하려면 다음 정책 ARN을 사용합니다.
     - `arn:aws:iam::aws:policy/AWSCodeBuildAdminAccess`
     - `arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess`
     - `arn:aws:iam::aws:policy/IAMFullAccess`
   - CodeBuild에 빌드 프로젝트 관리를 제외한 전체 액세스 권한을 추가하려면 다음 정책 ARN을 사용합니다.
     - `arn:aws:iam::aws:policy/AWSCodeBuildDeveloperAccess`
     - `arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess`
   - CodeBuild에 읽기 전용 액세스 권한을 추가하려면 다음 정책 ARN을 사용합니다.

- arn:aws:iam::aws:policy/AWSCodeBuildReadOnlyAccess
- arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess

이제 IAM 그룹 또는 사용자에게 기본 CodeBuild 액세스 권한 세트가 추가되었습니다. 이 절차의 나머지 단계는 건너뜁니다.

3. AWS CLI 가 설치된 로컬 워크스테이션 또는 인스턴스의 빈 디렉터리에서 put-group-policy.json 또는 라는 파일을 생성합니다put-user-policy.json. 다른 파일 이름을 선택하는 경우 이 절차 전체에서 해당 이름을 사용해야 합니다.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "CodeBuildAccessPolicy",
            "Effect": "Allow",
            "Action": [
                "codebuild:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CodeBuildRolePolicy",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::111122223333:role/role-name"
        },
        {
            "Sid": "CloudWatchLogsAccessPolicy",
            "Effect": "Allow",
            "Action": [
                "logs:FilterLogEvents",
                "logs:GetLogEvents"
            ],
            "Resource": "*"
        },
        {
```

```
            "Sid": "S3AccessPolicy",
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:GetObject",
                "s3:List*",
                "s3:PutObject"
            ],
            "Resource": "*"
        },
        {
            "Sid": "S3BucketIdentity",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketAcl",
                "s3:GetBucketLocation"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 이 정책은 모든 CodeBuild 작업과 잠재적으로 많은 AWS 리소스에 대한 액세스를 허용합니다. 특정 CodeBuild 작업으로 권한을 제한하려면 CodeBuild 정책 명령에서 `codebuild:*`의 값을 변경합니다. 자세한 내용은 ID 및 액세스 관리 섹션을 참조하세요. 특정 AWS 리소스에 대한 액세스를 제한하려면 관련 `Resource` 객체의 값을 변경합니다. 자세한 정보는 ID 및 액세스 관리 또는 특정 AWS 서비스의 보안 설명서를 참조하십시오.

4. 파일을 저장한 디렉터리로 전환한 다음, 다음 명령 중 하나를 실행합니다. `CodeBuildGroupAccessPolicy` 및 `CodeBuildUserAccessPolicy`에 다른 값을 사용할 수도 있습니다. 다른 값을 사용하는 경우 여기에서 사용해야 합니다.

   IAM 그룹의 경우:

   ```
   aws iam put-group-policy --group-name group-name --policy-name
    CodeBuildGroupAccessPolicy --policy-document file://put-group-policy.json
   ```

   사용자의 경우:

```
aws iam put-user-policy --user-name user-name --policy-name
 CodeBuildUserAccessPolicy --policy-document file://put-user-policy.json
```

앞의 명령에서 *group-name* 또는 *user-name*을 대상 IAM 그룹 또는 사용자의 이름으로 변경합니다.

CodeBuild가 다른 AWS 서비스와 상호 작용하도록 허용

의 단계에 따라 AWS CodeBuild 에 처음 콘솔을 사용하여 시작하기 액세스하는 경우이 주제의 정보가 필요하지 않을 가능성이 높습니다. 그러나 CodeBuild를 계속 사용하면 CodeBuild가 다른 AWS 서비스와 상호 작용하도록 허용하는 등의 작업을 수행할 수 있습니다.

CodeBuild가 사용자를 대신하여 종속 AWS 서비스와 상호 작용하도록 허용하려면 AWS CodeBuild 서비스 역할이 필요합니다. CodeBuild 또는 AWS CodePipeline 콘솔을 사용하면 CodeBuild 서비스 역할을 생성할 수 있습니다. 자세한 내용은 다음을 참조하세요.

- 빌드 프로젝트 만들기(콘솔)
- CodeBuild를 사용하는 파이프라인 생성(CodePipeline 콘솔)
- CodeBuild 빌드 작업을 파이프라인에 추가(CodePipeline 콘솔)
- 빌드 프로젝트 설정 변경(콘솔)

이러한 콘솔을 사용하지 않을 사용자를 위해 이 섹션에서는 IAM 콘솔 또는 AWS CLI를 사용하여 CodeBuild 서비스 역할을 생성하는 방법을 설명합니다.

> **Important**
>
> CodeBuild에서는 사용자를 대신하여 수행되는 모든 작업에 대해 서비스 역할을 사용합니다. 역할에 사용자에게 불필요한 권한이 포함되어 있는 경우 사용자의 권한을 실수로 에스컬레이션할 수 있습니다. 이 역할은 최소 권한을 부여해야 합니다.
> 이 페이지에서 설명하는 서비스 역할에는 CodeBuild를 사용하기 위해 필요한 최소한의 권한을 부여하는 정책이 포함되어 있습니다. 사용 사례에 따라 다른 권한을 추가해야 할 수 있습니다.

CodeBuild 서비스 역할을 생성하려면(콘솔)

1. IAM 콘솔(https://console.aws.amazon.com/iam/)을 엽니다.

다음 중 하나를 사용하여 콘솔에 이미 로그인되어 있어야 합니다.

- AWS 루트 계정. 이는 권장하지 않습니다. 자세한 내용은 사용 설명서의 AWS 계정 루트 사용자를 참조하세요.
- AWS 계정의 관리자 사용자입니다. 자세한 내용은 사용 설명서의 첫 번째 AWS 계정 루트 사용자 및 그룹 생성을 참조하세요.
- AWS 계정에서 다음과 같은 최소 작업 세트를 수행할 수 있는 권한이 있는 사용자:

```
iam:AddRoleToInstanceProfile
iam:AttachRolePolicy
iam:CreateInstanceProfile
iam:CreatePolicy
iam:CreateRole
iam:GetRole
iam:ListAttachedRolePolicies
iam:ListPolicies
iam:ListRoles
iam:PassRole
iam:PutRolePolicy
iam:UpdateAssumeRolePolicy
```

자세한 내용은 사용 설명서의 IAM 정책 개요를 참조하세요.

2. 탐색 창에서 Policies를 선택합니다.
3. 정책 생성을 선택합니다.
4. [Create Policy] 페이지에서 [JSON]을 선택합니다.
5. JSON 정책에 대해 다음을 입력한 다음 정책 검토를 선택합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "CloudWatchLogsPolicy",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
```

```
        "logs:PutLogEvents"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CodeCommitPolicy",
      "Effect": "Allow",
      "Action": [
        "codecommit:GitPull"
      ],
      "Resource": "*"
    },
    {
      "Sid": "S3GetObjectPolicy",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": "*"
    },
    {
      "Sid": "S3PutObjectPolicy",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "*"
    },
    {
      "Sid": "ECRPullPolicy",
      "Effect": "Allow",
      "Action": [
        "ecr:BatchCheckLayerAvailability",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage"
      ],
      "Resource": "*"
    },
    {
      "Sid": "ECRAuthPolicy",
      "Effect": "Allow",
      "Action": [
        "ecr:GetAuthorizationToken"
```

```
      ],
      "Resource": "*"
    },
    {
      "Sid": "S3BucketIdentity",
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl",
        "s3:GetBucketLocation"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> 이 정책에는 잠재적으로 많은 수의 AWS 리소스에 대한 액세스를 허용하는 문이 포함되어 있습니다. 가 특정 AWS 리소스에 액세스 AWS CodeBuild 하도록 제한하려면 `Resource` 배열의 값을 변경합니다. 자세한 내용은 AWS 서비스에 대한 보안 설명서를 참조하세요.

6. 정책 검토 페이지의 정책 이름에 정책의 이름(예: **`CodeBuildServiceRolePolicy`**)을 입력한 후 정책 생성을 선택합니다.

> **Note**
>
> 다른 이름을 사용하는 경우 이 절차 전체에서 해당 이름을 사용해야 합니다.

7. 탐색 창에서 Roles를 선택합니다.
8. 역할 생성을 선택합니다.
9. 역할 생성 페이지에서 AWS 서비스가 이미 선택되어 있으면 CodeBuild를 선택하고 다음: 권한을 선택합니다.
10. 권한 정책 연결 페이지에서 CodeBuildServiceRolePolicy를 선택한 후 다음: 검토를 선택합니다.
11. 역할 생성 및 검토 페이지의 역할 이름에 역할의 이름(예: **`CodeBuildServiceRole`**)을 입력한 후 역할 생성을 선택합니다.

CodeBuild 서비스 역할을 생성하려면(AWS CLI)

1. 이전 절차에서 설명한 대로 IAM 엔터티 중 하나에 해당하는 AWS 액세스 키 및 AWS 보안 액세스 키 AWS CLI 로를 구성했는지 확인합니다. 자세한 내용은AWS Command Line Interface 사용 설명서의 AWS Command Line Interface설정을 참조하세요.
2. AWS CLI 가 설치된 로컬 워크스테이션 또는 인스턴스의 빈 디렉터리에서 `create-role.json` 및 라는 두 개의 파일을 생성합니다`put-role-policy.json`. 다른 파일 이름을 선택하는 경우 이 절차 전체에서 해당 이름으로 바꿉니다.

   `create-role.json`:

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "codebuild.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

> **Note**
>
> 혼동된 대리자 문제로부터 자신을 보호하기 위하여 `aws:SourceAccount` 및 `aws:SourceArn` 조건 키를 사용할 것을 권장합니다. 예를 들어 다음 조건 블록을 사용해서 이전 신뢰 정책을 편집할 수 있습니다. `aws:SourceAccount`는 CodeBuild 프로젝트의 소유자이고 `aws:SourceArn`은 CodeBuild 프로젝트 ARN입니다.

서비스 역할을 AWS 계정으로 제한하려는 경우는 다음과 비슷할 `create-role.json` 수 있습니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "codebuild.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": [
                        "111122223333"
                    ]
                }
            }
        }
    ]
}
```

서비스 역할을 특정 CodeBuild 프로젝트로 제한하려는 경우 `create-role.json`이 다음과 비슷할 수 있습니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "codebuild.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
```

```
                    "aws:SourceArn": "arn:aws:codebuild:us-
east-1:111122223333:project/MyProject"
                }
            }
        }
    ]
}
```

> **Note**
>
> CodeBuild 프로젝트의 이름을 모르거나 아직 결정하지 않은 상태에서 특정 ARN 패턴에 대한 신뢰 정책 제한을 적용하려면 ARN의 해당 부분을 와일드카드(*)로 바꿀 수 있습니다. 프로젝트를 생성한 후 신뢰 정책을 업데이트할 수 있습니다.

`put-role-policy.json`:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "CloudWatchLogsPolicy",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CodeCommitPolicy",
      "Effect": "Allow",
      "Action": [
        "codecommit:GitPull"
      ],
      "Resource": "*"
    },
```

```
    {
      "Sid": "S3GetObjectPolicy",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": "*"
    },
    {
      "Sid": "S3PutObjectPolicy",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "*"
    },
    {
      "Sid": "S3BucketIdentity",
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl",
        "s3:GetBucketLocation"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> 이 정책에는 잠재적으로 많은 수의 AWS 리소스에 대한 액세스를 허용하는 문이 포함되어 있습니다. 가 특정 AWS 리소스에 액세스 AWS CodeBuild 하도록 제한하려면 `Resource` 배열의 값을 변경합니다. 자세한 내용은 AWS 서비스에 대한 보안 설명서를 참조하세요.

3. 앞의 파일을 저장한 디렉터리로 전환한 다음, 다음 두 가지 명령을 다음 순서로 한 번에 하나씩 실행합니다. `CodeBuildServiceRole`과 `CodeBuildServiceRolePolicy`에 다른 값을 사용하려면 여기에서 사용해야 합니다.

```
aws iam create-role --role-name CodeBuildServiceRole --assume-role-policy-document
 file://create-role.json
```

```
aws iam put-role-policy --role-name CodeBuildServiceRole --policy-name
 CodeBuildServiceRolePolicy --policy-document file://put-role-policy.json
```

고객 관리형 키를 사용하여 빌드 출력 암호화

의 단계에 따라 AWS CodeBuild 에 처음 콘솔을 사용하여 시작하기 액세스하는 경우이 주제의 정보가 필요하지 않을 가능성이 높습니다. 그러나 CodeBuild를 계속 사용하면 빌드 아티팩트 암호화와 같은 작업을 수행할 수 있습니다.

AWS CodeBuild 가 빌드 출력 아티팩트를 암호화하려면 KMS 키에 대한 액세스 권한이 필요합니다. 기본적으로 CodeBuild는 AWS 계정의 Amazon S3 AWS 관리형 키 용를 사용합니다.

를 사용하지 않으려면 고객 관리형 키를 직접 생성하고 구성 AWS 관리형 키해야 합니다. 이 섹션에서는 IAM 콘솔을 사용하여 이를 수행하는 방법을 설명합니다.

고객 관리형 키에 대한 자세한 내용은 AWS KMS 개발자 가이드의 AWS Key Management Service 개념 및 키 생성을 참조하세요.

CodeBuild에서 사용할 고객 관리형 키를 구성하려면 AWS KMS 개발자 안내서의 키 정책 수정의 “키 정책 수정 방법” 섹션에 있는 지침을 따르세요. 그런 다음 키 정책에 다음 명령문(*### BEGIN ADDING STATEMENTS HERE ###*과 *### END ADDING STATEMENTS HERE ###* 사이)을 추가합니다. 간결하게 나타내고 명령문 추가 위치를 알 수 있도록 줄임표(`...`)가 사용되었습니다. 어떤 명령문도 제거하지 않아야 하며, 이러한 줄임표는 키 정책에 입력하지 않아야 합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
```

```
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "s3.us-east-1.amazonaws.com",
          "kms:CallerAccount": "111122223333"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    }
  ]
}
```

- *region-ID*는 CodeBuild와 연결된 Amazon S3 버킷이 위치한 AWS 리전의 ID를 나타냅니다(예: `us-east-1`).
- *account-ID*는 고객 관리형 키를 소유하는 AWS 계정의 ID를 나타냅니다.
- *CodeBuild-service-role*은 이 주제의 앞부분에서 생성하거나 식별한 CodeBuild 서비스 역할 이름을 나타냅니다.

> **Note**
>
> IAM 콘솔을 통해 고객 관리형 키를 생성하거나 구성하려면 먼저 다음 중 하나를 사용하여 AWS Management Console 에 로그인해야 합니다.
>
> - AWS 루트 계정. 이는 권장하지 않습니다. 자세한 내용은 사용 설명서의 계정 루트 사용자를 참조하세요.

- AWS 계정의 관리자 사용자입니다. 자세한 내용은 사용 설명서의 첫 번째 AWS 계정 루트 사용자 및 그룹 생성을 참조하세요.
- AWS 계정에서 고객 관리형 키를 생성하거나 수정할 수 있는 권한이 있는 사용자입니다. 자세한 내용은 AWS KMS 개발자 안내서의 AWS KMS 콘솔 사용에 필요한 권한을 참조하세요.

를 사용하여 CodeBuild와 상호 작용 AWS CLI

의 단계에 따라 AWS CodeBuild 에 처음 콘솔을 사용하여 시작하기 액세스하는 경우이 주제의 정보가 필요하지 않을 가능성이 높습니다. 그러나 CodeBuild를 계속 사용하면 사용자가 CodeBuild 콘솔, CodePipeline 콘솔 또는 SDK 대신(또는 이에 추가하여)를 사용하여 CodeBuild와 상호 작용 AWS CLI 하도록 허용하는 등의 작업을 수행할 수 있습니다. AWS SDKs

를 설치하고 구성하려면 AWS Command Line Interface 사용 설명서의 시작하기 AWS Command Line Interface를 AWS CLI참조하세요.

설치 후 다음 작업을 AWS CLI완료합니다.

1. 다음 명령을 실행하여 CodeBuild에 AWS CLI 를 설치할 수 있는지 여부를 확인합니다.

   ```
   aws codebuild list-builds
   ```

   이 명령이 제대로 실행되면 다음과 비슷한 정보가 출력에 표시됩니다.

   ```
   {
     "ids": []
   }
   ```

   빈 대괄호는 실행한 빌드가 아직 없다는 의미입니다.
2. 오류가 출력되면 현재 AWS CLI 버전을 제거한 다음 최신 버전을 설치해야 합니다. 자세한 내용은 AWS Command Line Interface 사용 설명서의 AWS CLI제거 및 AWS Command Line Interface설치를 참조하세요.

에 대한 명령줄 참조 AWS CodeBuild

는 자동화를 위한 명령을 AWS CLI 제공합니다 AWS CodeBuild. 이 주제의 정보를 AWS Command Line Interface 사용 설명서 및 AWS CodeBuild에 대한AWS CLI 참조의 추가 자료로 사용하세요.

찾는 항목이 보이지 않습니까? AWS SDKs CodeBuild를 호출하려면 섹션을 참조하세요AWS SDKs 및 도구 참조.

이 주제의 정보를 사용하려면에 설명된 대로 CodeBuild AWS CLI 와 함께 사용하도록를 이미 설치하고 구성해야 합니다를 사용하여 CodeBuild와 상호 작용 AWS CLI.

AWS CLI 를 사용하여 CodeBuild의 엔드포인트를 지정하려면 섹션을 참조하세요AWS CodeBuild 엔드포인트 지정(AWS CLI).

이 명령을 실행하여 CodeBuild 명령 목록을 가져옵니다.

```
aws codebuild help
```

이 명령을 실행하여 CodeBuild 명령에 대한 정보를 가져옵니다. 여기서 *command-name*은 명령의 이름입니다.

```
aws codebuild command-name help
```

CodeBuild 명령에는 다음이 포함됩니다.

- `batch-delete-builds`: CodeBuild에서 하나 이상의 빌드를 삭제합니다. 자세한 내용은 빌드 삭제(AWS CLI) 단원을 참조하십시오.
- `batch-get-builds`: CodeBuild의 여러 빌드에 대한 정보를 가져옵니다. 자세한 내용은 빌드 세부 정보 보기(AWS CLI) 단원을 참조하십시오.
- `batch-get-projects`: 하나 이상의 지정된 빌드 프로젝트에 대한 정보를 가져옵니다. 자세한 내용은 빌드 프로젝트 세부 정보 보기(AWS CLI) 단원을 참조하십시오.
- `create-project`: 빌드 프로젝트를 생성합니다. 자세한 내용은 빌드 프로젝트 생성(AWS CLI) 단원을 참조하십시오.
- `delete-project`: 빌드 프로젝트를 삭제합니다. 자세한 내용은 빌드 프로젝트 삭제(AWS CLI) 단원을 참조하십시오.
- `list-builds`: CodeBuild의 빌드를 위한 Amazon 리소스 이름(ARN)을 표시합니다. 자세한 내용은 빌드 ID 목록 보기(AWS CLI) 단원을 참조하십시오.
- `list-builds-for-project`: 지정된 빌드 프로젝트에 연결된 빌드 ID 목록을 가져옵니다. 자세한 내용은 빌드 프로젝트의 빌드 ID 목록 보기(AWS CLI) 단원을 참조하십시오.
- `list-curated-environment-images`: 빌드에 사용할 수 있는 CodeBuild 관리형 도커 이미지 목록을 가져옵니다. 자세한 내용은 CodeBuild가 제공하는 도커 이미지 단원을 참조하십시오.

- `list-projects`: 빌드 프로젝트 이름 목록을 가져옵니다. 자세한 내용은 빌드 프로젝트 이름 목록 보기(AWS CLI) 단원을 참조하십시오.
- `start-build`: 빌드 실행을 시작합니다. 자세한 내용은 빌드 실행(AWS CLI) 단원을 참조하십시오.
- `stop-build`: 지정된 빌드의 실행을 중지하려고 시도합니다. 자세한 내용은 빌드 중지(AWS CLI) 단원을 참조하십시오.
- `update-project`: 지정된 빌드 프로젝트에 대한 정보를 변경합니다. 자세한 내용은 빌드 프로젝트 설정 변경(AWS CLI) 단원을 참조하십시오.

AWS SDKs 및 도구 참조 AWS CodeBuild

AWS SDKs 또는 도구 중 하나를 사용하여 자동화하려면 다음 리소스를 AWS CodeBuild참조하세요.

를 사용하여 CodeBuild AWS CLI 를 실행하려면 섹션을 참조하세요명령줄 참조.

AWS SDKs 및 도구 AWS CodeBuild

다음 AWS SDKs 및 도구는 CodeBuild를 지원합니다.

- C++용AWS SDK. 자세한 내용은 C++용AWS SDK API 참조의 Aws::CodeBuild 네임스페이스 섹션을 참조하세요.
- Go용AWS SDK. 자세한 내용은 Go용AWS SDK API 참조의 codebuild 섹션을 참조하세요.
- Java용AWS SDK. 자세한 내용은 Java용AWS SDK API 참조의 `com.amazonaws.services.codebuild` 및 `com.amazonaws.services.codebuild.model` 섹션을 참조하세요.
- 브라우저에서 JavaScript용AWS SDK 사용 및 Node.js에서 JavaScript용AWS SDK 사용. 자세한 내용은 클래스:를 참조하세요 AWS.AWS JavaScript SDK for JavaScript API 참조의 CodeBuild 섹션.
- .NET용AWS SDK. 자세한 내용은 .NET용AWS SDK API 참조의 Amazon.CodeBuild 및 Amazon.CodeBuild.Model 네임스페이스 섹션을 참조하세요.
- PHP용AWS SDK. 자세한 내용은 PHP용AWS SDK API 참조의 네임스페이스 Aws\CodeBuild 섹션을 참조하세요.
- Python용AWS SDK(Boto3). 자세한 내용은 Boto 3 설명서의 CodeBuild 섹션을 참조하세요.
- Ruby용AWS SDK. 자세한 내용은 Ruby용AWS SDK API 참조의 모듈: Aws::CodeBuild 섹션을 참조하세요.
- PowerShell용AWS 도구. 자세한 내용은 PowerShell Cmdlet용AWS 도구 참조의 AWS CodeBuild 섹션을 참조하세요.

AWS SDK에서이 서비스 사용

AWS 소프트웨어 개발 키트(SDKs)는 널리 사용되는 여러 프로그래밍 언어에 사용할 수 있습니다. 각 SDK는 개발자가 선호하는 언어로 애플리케이션을 쉽게 구축할 수 있도록 하는 API, 코드 예제 및 설명서를 제공합니다.

SDK 설명서	코드 예제
AWS SDK for C++	AWS SDK for C++ 코드 예제
AWS CLI	AWS CLI 코드 예제
AWS SDK for Go	AWS SDK for Go 코드 예제
AWS SDK for Java	AWS SDK for Java 코드 예제
AWS SDK for JavaScript	AWS SDK for JavaScript 코드 예제
AWS SDK for Kotlin	AWS SDK for Kotlin 코드 예제
AWS SDK for .NET	AWS SDK for .NET 코드 예제
AWS SDK for PHP	AWS SDK for PHP 코드 예제
AWS Tools for PowerShell	AWS Tools for PowerShell 코드 예제
AWS Python v4용 SDK	AWS SDK for Python v4 코드 예제
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 코드 예제
AWS SDK for Ruby	AWS SDK for Ruby 코드 예제
AWS SDK for Rust	AWS SDK for Rust 코드 예제
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP 코드 예제
AWS SDK for Swift	AWS SDK for Swift 코드 예제

이 서비스 관련 예제는 AWS SDKs를 사용한 CodeBuild 코드 예제를 참조하세요.

> **예제 사용 가능 여부**
>
> 필요한 예제를 찾을 수 없습니까? 이 페이지 하단의 피드백 제공 링크를 사용하여 코드 예시를 요청하세요.

AWS CodeBuild 엔드포인트 지정

AWS Command Line Interface (AWS CLI) 또는 AWS SDKs 중 하나를 사용하여에서 사용하는 엔드포인트를 지정할 수 있습니다 AWS CodeBuild. CodeBuild를 사용할 수 있는 각 리전에 대한 엔드포인트가 있습니다. 4개 리전에는 1개의 리전 엔드포인트 외에 Federal Information Processing Standards(FIPS) 엔드포인트도 있습니다. FIPS 엔드포인트에 대한 자세한 내용은 FIPS 140-2 개요를 참조하세요.

엔드포인트 지정은 선택 사항입니다. CodeBuild에 사용할 엔드포인트를 명시적으로 알리지 않으면 서비스는 AWS 계정에서 사용하는 리전과 연결된 엔드포인트를 사용합니다. CodeBuild의 기본 엔드포인트는 FIPS 엔드포인트가 아닙니다. FIPS 엔드포인트를 사용하려면 다음 방법 중 하나를 사용하여 CodeBuild를 이 엔드포인트와 연결해야 합니다.

> **Note**
>
> 별칭 또는 리전 이름을 사용하여 AWS SDK를 사용하여 엔드포인트를 지정할 수 있습니다. 를 사용하는 경우 전체 엔드포인트 이름을 사용해야 AWS CLI합니다.

CodeBuild에 사용할 수 있는 엔드포인트에 대해서는 CodeBuild 리전 및 엔드포인트를 참조하세요.

주제

- AWS CodeBuild 엔드포인트 지정(AWS CLI)
- AWS CodeBuild 엔드포인트 지정(AWS SDK)

AWS CodeBuild 엔드포인트 지정(AWS CLI)

AWS CLI 를 사용하여 CodeBuild 명령에서 `--endpoint-url` 인수를 사용하여 AWS CodeBuild 에 액세스할 엔드포인트를 지정할 수 있습니다. 예를 들어 다음 명령을 실행하여 미국 동부(버지니아 북부) 리전에서 Federal Information Processing Standards(FIPS) 엔드포인트를 통해 프로젝트 빌드 이름 목록을 가져옵니다.

```
aws codebuild list-projects --endpoint-url https://codebuild-fips.us-
east-1.amazonaws.com
```

엔드포인트 시작 `https://` 부분에를 포함합니다.

`--endpoint-url` AWS CLI 인수는 모든 AWS 서비스에서 사용할 수 있습니다. 이 인수 및 기타 AWS CLI 인수에 대한 자세한 내용은 AWS CLI 명령 참조를 참조하세요.

AWS CodeBuild 엔드포인트 지정(AWS SDK)

AWS SDK를 사용하여에 액세스할 엔드포인트를 지정할 수 AWS CodeBuild 있습니다. 이 예제에서는 AWS SDK for Java를 사용하지만 다른 AWS SDKs.

AWSCodeBuild 클라이언트를 구성할 때는 이 `withEndpointConfiguration` 메서드를 사용합니다. 사용할 형식은 다음과 같습니다.

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("endpoint",
 "region")).
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).
    build();
```

`AWSCodeBuildClientBuilder`에 대한 자세한 내용은 클래스 AWSCodeBuildClientBuilder를 참조하세요.

`withCredentials`에서 사용되는 보안 인증은 `AWSCredentialsProvider` 형식이어야 합니다. 자세한 내용은 자격 AWS 증명 작업을 참조하세요.

엔드포인트의 시작 `https://` 부분에를 포함하지 마십시오.

비 FIPS 엔드포인트를 지정하려는 경우 실제 엔드포인트 대신 리전을 사용할 수 있습니다. 예를 들어 미국 동부(버지니아 북부) 리전에서 엔드포인트를 지정하려면 전체 엔드포인트 이름 `codebuild.us-east-1.amazonaws.com` 대신 `us-east-1`을 사용할 수 있습니다.

FIPS 엔드포인트를 지정하려는 경우 별칭을 사용하여 코드를 단순화할 수 있습니다. FIPS 엔드포인트에만 별칭이 있습니다. 다른 엔드포인트는 해당 리전 또는 전체 이름을 사용하여 지정해야 합니다.

다음 표에는 사용 가능한 FIPS 엔드포인트 4개 각각의 별칭이 나열되어 있습니다.

리전 이름	리전	엔드포인트	Alias
미국 동부(버지니아 북부)	us-east-1	codebuild-fips.us-east-1.amazonaws.com	us-east-1-fips
미국 동부(오하이오)	us-east-2	codebuild-fips.us-east-2.amazonaws.com	us-east-2-fips
미국 서부(캘리포니아 북부)	us-west-1	codebuild-fips.us-west-1.amazonaws.com	us-west-1-fips
미국 서부(오리건)	us-west-2	codebuild-fips.us-west-2.amazonaws.com	us-west-2-fips

별칭을 사용하여 미국 서부(오레곤) 리전에서 FIPS 엔드포인트 사용을 지정하려면:

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("us-west-2-
fips", "us-west-2")).
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).
    build();
```

미국 동부(버지니아 북부) 리전에서 비 FIPS 엔드포인트 사용을 지정하려면:

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("us-east-1",
 "us-east-1")).
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).
    build();
```

아시아 태평양(뭄바이) 리전에서 비 FIPS 엔드포인트 사용을 지정하려면:

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("ap-south-1",
 "ap-south-1")).
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).
    build();
```

와 AWS CodeBuild 함께 AWS CodePipeline 를 사용하여 코드를 테스트하고 빌드를 실행합니다.

를 사용하여 코드를 테스트하고를 사용하여 빌드를 실행하여 릴리스 프로세스를 자동화 AWS CodePipeline 할 수 있습니다 AWS CodeBuild.

다음 표에는 태스크 및 태스크 수행에 사용할 수 있는 방법이 나와 있습니다. AWS SDK를 사용하여 이러한 태스크를 수행하는 내용은 본 주제에서 다루지 않습니다.

Task	사용 가능한 접근 방식	이 주제에 설명된 접근 방식
CodeBuild로 빌드를 자동화하는 CodePipeline으로 지속적 배포(CD) 파이프라인을 생성합니다.	• CodePipeline 콘솔 • AWS CLI • AWS SDKs	• CodePipeline 콘솔 사용 • AWS CLI사용 • 이 주제의 정보를 AWS SDK를 사용하도록 조정할 수 있습니다. 자세한 내용은 Amazon Web Services용 도구의 SDK 섹션에서 프로그래밍 언어에 대한 `create-pipeline` 작업 설명서를 참조하거나AWS CodePipeline API 참조에서 `CreatePipeline` 을 참조하세요.
CodeBuild를 사용한 테스트 및 빌드 자동화를 CodePipeline의 기존 파이프라인에 추가	• CodePipeline 콘솔 • AWS CLI • AWS SDKs	• CodePipeline 콘솔을 사용하여 빌드 자동화 추가 • CodePipeline 콘솔을 사용하여 테스트 자동화 추가 • 의 경우이 주제의 정보를 조정하여 CodeBuild 빌드 작업 또는 테스트 작업이 포함된 파이프라인을 생성할 AWS CLI수 있습니다. 자세한 내용은AWS CodePipeline 사용 설명서의 파이프라인 편집 (AWS CLI) 및 CodePipeline 파이프라인 구조 참조를 참조하세요. • 이 주제의 정보를 AWS SDK를 사용하도록 조정할 수 있습니다. 자세한 내용은 Amazon Web Services용 도구의

Task	사용 가능한 접근 방식	이 주제에 설명된 접근 방식
		SDK 섹션을 통해 프로그래밍 언어에 대한 `update-pipeline` 작업 설명서를 참조하거나AWS CodePipeline API 참조에서 `UpdatePipeline` 을 참조하세요.

주제

- 사전 조건
- CodeBuild를 사용하는 파이프라인 생성(CodePipeline 콘솔)
- CodeBuild를 사용하는 파이프라인 생성(AWS CLI)
- CodeBuild 빌드 작업을 파이프라인에 추가(CodePipeline 콘솔)
- CodeBuild 테스트 작업을 파이프라인에 추가(CodePipeline 콘솔)

사전 조건

1. 빌드 계획 섹션의 질문에 답하십시오.
2. 사용자를 사용하여 AWS 루트 계정 또는 관리자 사용자 대신 CodePipeline에 액세스하는 경우 라는 관리형 정책을 `AWSCodePipelineFullAccess` 사용자(또는 사용자가 속한 IAM 그룹)에 연결합니다. AWS 루트 계정 사용은 권장되지 않습니다. 이 정책은 사용자에게 CodePipeline에서 파이프라인을 생성할 권한을 부여합니다. 자세한 내용은 사용 설명서의 관리형 정책 연결을 참조하세요.

   > **Note**
   >
   > 정책을 사용자(또는 사용자가 속한 IAM 그룹)에 연결하는 IAM 엔터티는 IAM에서 정책을 연결할 수 있는 권한이 있어야 합니다. 자세한 내용은 사용 설명서의 IAM 사용자, 그룹 및 보안 인증 관리 권한 위임을 참조하세요.

3. AWS 계정에서 사용할 수 있는 CodePipeline 서비스 역할이 없는 경우 생성합니다. CodePipeline은이 서비스 역할을 사용하여 사용자를 대신하여를 포함한 다른 AWS 서비스와 상호 작용 AWS CodeBuild합니다. 예를 들어 AWS CLI 를 사용하여 CodePipeline 서비스 역할을 생성하려면 IAM `create-role` 명령을 실행합니다.

   Linux, macOS, Unix의 경우:

```
aws iam create-role --role-name AWS-CodePipeline-CodeBuild-
Service-Role --assume-role-policy-document '{"Version":
 "2012-10-17",       "Statement":{"Effect":"Allow","Principal":
{"Service":"codepipeline.amazonaws.com"},"Action":"sts:AssumeRole"}}'
```

Windows의 경우:

```
aws iam create-role --role-name AWS-CodePipeline-CodeBuild-Service-Role --assume-
role-policy-document "{\"Version\":\"2012-10-17\",       \"Statement\":{\"Effect
\":\"Allow\",\"Principal\":{\"Service\":\"codepipeline.amazonaws.com\"},\"Action\":
\"sts:AssumeRole\"}}"
```

> **Note**
>
> 이 CodePipeline 서비스 역할을 생성하는 IAM 엔터티는 IAM에서 서비스 역할을 생성할 수 있는 권한이 있어야 합니다.

4. CodePipeline 서비스 역할을 생성하거나 기존 서비스 역할을 식별한 후에는 해당 역할에 정책이 아직 포함되지 않은 경우AWS CodePipeline 사용 설명서의 기본 CodePipeline 서비스 역할 정책 검토에 설명된 대로 서비스 역할에 기본 CodePipeline 서비스 역할 정책을 추가해야 합니다.

> **Note**
>
> 이 CodePipeline 서비스 역할 정책을 추가하는 IAM 엔터티는 IAM에서 서비스 역할에 서비스 역할 정책을 추가할 수 있는 권한이 있어야 합니다.

5. 소스 코드를 생성하고 CodeCommit, Amazon S3, Bitbucket, GitHub 등과 같이 CodeBuild와 CodePipeline에서 지원하는 리포지토리 유형으로 업로드합니다. 소스 코드에 빌드 사양 파일을 포함해야 하지만, 이 주제의 후반부에서 빌드 프로젝트를 정의할 때 빌드 사양 파일을 선언할 수 있습니다. 자세한 내용은 buildspec 참조 단원을 참조하십시오.

> **Important**
>
> 파이프라인을 사용하여 빌드 소스 코드를 배포하려는 경우, 빌드 출력 아티팩트와 사용할 배포 시스템이 호환 가능해야 합니다.

- 자세한 OpsWorks내용은 OpsWorks 사용 설명서의 애플리케이션 소스 및 에서 CodePipeline 사용을 OpsWorks 참조하세요.

CodeBuild를 사용하는 파이프라인 생성(CodePipeline 콘솔)

다음 절차에 따라 CodeBuild를 사용하여 소스 코드를 빌드하고 배포하는 파이프라인을 생성하세요.

소스 코드만 테스트하는 파이프라인을 생성하려면:

- 다음 절차를 사용하여 파이프라인을 생성한 다음, 파이프라인에서 빌드 및 베타 단계를 삭제합니다. 그런 다음, 이 주제의 CodeBuild 테스트 작업을 파이프라인에 추가(CodePipeline 콘솔) 절차를 사용하여 CodeBuild를 사용하는 테스트 작업을 파이프라인에 추가합니다.
- 이 주제의 다른 절차 중 하나를 사용하여 파이프라인을 생성한 다음, 이 주제의 CodeBuild 테스트 작업을 파이프라인에 추가(CodePipeline 콘솔) 절차를 사용하여 CodeBuild를 사용하는 테스트 작업을 파이프라인에 추가합니다.

CodePipeline의 파이프라인 생성 마법사를 사용하여 CodeBuild를 사용하는 파이프라인을 생성하려면

1. 다음을 사용하여 AWS Management Console 에 로그인합니다.
   - AWS 루트 계정. 이는 권장하지 않습니다. 자세한 내용은 사용 설명서의 계정 루트 사용자를 참조하세요.
   - AWS 계정의 관리자 사용자입니다. 자세한 내용은 사용 설명서의 첫 번째 AWS 계정 루트 사용자 및 그룹 생성을 참조하세요.
   - AWS 계정에서 다음과 같은 최소 작업 세트를 사용할 수 있는 권한이 있는 사용자:

```
codepipeline:*
iam:ListRoles
iam:PassRole
s3:CreateBucket
s3:GetBucketPolicy
s3:GetObject
s3:ListAllMyBuckets
s3:ListBucket
s3:PutBucketPolicy
codecommit:ListBranches
codecommit:ListRepositories
```

```
codedeploy:GetApplication
codedeploy:GetDeploymentGroup
codedeploy:ListApplications
codedeploy:ListDeploymentGroups
elasticbeanstalk:DescribeApplications
elasticbeanstalk:DescribeEnvironments
lambda:GetFunctionConfiguration
lambda:ListFunctions
opsworks:DescribeStacks
opsworks:DescribeApps
opsworks:DescribeLayers
```

2. https://console.aws.amazon.com/codesuite/codepipeline/home AWS CodePipeline 콘솔을 엽니다.
3. AWS 리전 선택기에서 빌드 프로젝트 AWS 리소스가 있는 AWS 리전을 선택합니다. CodeBuild가 지원되는 AWS 리전이어야 합니다. 자세한 내용은 Amazon Web Services 일반 참조의 AWS CodeBuild 섹션을 참조하세요.
4. 파이프라인 생성. CodePipeline 정보 페이지가 표시되면 파이프라인 생성을 선택합니다. 파이프라인 페이지가 표시되면 파이프라인 생성을 선택합니다.
5. 1단계: 파이프라인 설정 선택 페이지의 파이프라인 이름에 파이프라인 이름(예: **CodeBuildDemoPipeline**)을 입력합니다. 다른 이름을 선택하는 경우 이 절차 전체에서 해당 이름을 사용해야 합니다.
6. 역할 이름의 경우 다음 중 하나를 수행합니다.

   새 서비스 역할을 선택하고 역할 이름에 새 서비스 역할의 이름을 입력합니다.

   기존 서비스 역할을 선택한 다음, 이 주제의 필수 조건의 일부로 생성하거나 식별한 CodePipeline 서비스 역할을 선택합니다.
7. 아티팩트 스토어에서 다음 중 하나를 수행합니다.

   - 파이프라인에 대해 선택한 AWS 리전의 파이프라인에 대해 기본값으로 지정된 S3 아티팩트 버킷과 같은 기본 아티팩트 스토어를 사용하려면 기본 위치를 선택합니다.
   - 파이프라인과 동일한 AWS 리전에 S3 아티팩트 버킷과 같이 생성한 기존 아티팩트 스토어가 이미 있는 경우 사용자 지정 위치를 선택합니다.

> **Note**
>
> 이는 파이프라인 소스 코드에 대한 소스 버킷이 아닙니다. 이 파이프라인은 아티팩트 스토어입니다. 파이프라인과 동일한 AWS 리전의 각 파이프라인에는 S3 버킷과 같은 별도의 아티팩트 스토어가 필요합니다.

8. 다음을 선택합니다.
9. 2단계: 소스 단계 추가 페이지에서 소스 공급자에 대해 다음 중 하나를 수행하세요.

   - 소스 코드가 S3 버킷에 저장되어 있는 경우 Amazon S3를 선택합니다. 버킷에서 소스 코드를 포함하는 S3 버킷을 선택합니다. S3 객체 키에 소스 코드가 포함된 파일의 이름(예: *file-name*.zip)을 입력합니다. 다음을 선택합니다.
   - 소스 코드가 AWS CodeCommit 리포지토리에 저장된 경우 CodeCommit을 선택합니다. 리포지토리 이름의 경우 소스 코드가 포함된 리포지토리의 이름을 선택합니다. [Branch name]에서 빌드하려는 소스 코드의 버전이 포함된 브랜치 이름을 선택합니다. 다음을 선택합니다.
   - 소스 코드가 GitHub 리포지토리에 저장되어 있는 경우 GitHub를 선택합니다. GitHub에 연결을 선택하고 지침에 따라 GitHub에서 인증을 받습니다. 리포지토리의 경우 소스 코드가 포함된 리포지토리의 이름을 선택합니다. [Branch]에서 빌드하려는 소스 코드의 버전이 포함된 브랜치 이름을 선택합니다.

   다음을 선택합니다.
10. 3단계: 빌드 단계 추가 페이지에서 빌드 공급자에 대해 CodeBuild를 선택합니다.
11. 사용하려는 빌드 프로젝트가 이미 있는 경우 프로젝트 이름에서 빌드 프로젝트의 이름을 선택하고 이 절차의 다음 단계로 건너뜁니다.

    새 CodeBuild 빌드 프로젝트를 생성해야 하는 경우 빌드 프로젝트 만들기(콘솔)의 지침을 따르고 이 절차로 돌아갑니다.

    기존 빌드 프로젝트를 선택하는 경우 CodePipeline에서 재정의되더라도 빌드 출력 아티팩트 설정이 이미 정의되어 있어야 합니다. 자세한 내용은 빌드 프로젝트 설정 변경(콘솔) 단원을 참조하십시오.

> **Important**
>
> CodeBuild 프로젝트에 대해 webhook를 활성화하고 해당 프로젝트가 CodePipeline의 빌드 단계로 사용되는 경우 각 커밋에 대해 두 개의 동일한 빌드가 생성됩니다. 하나의 빌드는 webhook를 통해 트리거되고 다른 하나는 CodePipeline을 통해 트리거됩니다. 빌드 기준으로 요금이 청구되므로 두 빌드 모두에 대해 요금이 청구됩니다. 따라서 CodePipeline을 사용하는 경우 CodeBuild에서 webhook를 비활성화하는 것이 좋습니다. AWS CodeBuild 콘솔에서 Webhook 상자를 해제합니다. 자세한 내용은 빌드 프로젝트 설정 변경(콘솔) 단원을 참조하십시오.

12. 4단계: 배포 단계 추가 페이지에서 다음 중 하나를 수행합니다.

    - 빌드 출력 아티팩트를 배포하지 않으려면 건너뛰기를 선택하고 메시지가 표시되면 이 옵션을 확인합니다.
    - 빌드 출력 아티팩트를 배포하려는 경우 배포 공급자에 대해 배포 공급자를 선택한 다음, 메시지가 표시되면 설정을 지정합니다.

    다음을 선택합니다.

13. 검토 페이지에서 선택 사항을 검토한 다음, 파이프라인 생성을 선택합니다.
14. 파이프라인이 성공적으로 실행되면 빌드 출력 아티팩트를 가져올 수 있습니다. CodePipeline 콘솔에 파이프라인이 표시된 상태에서 빌드 작업에서 도구 설명을 선택합니다. 출력 아티팩트(예: MyAppBuild)의 값을 적어 놓습니다.

> **Note**
>
> CodeBuild 콘솔의 빌드 세부 정보 페이지에서 빌드 아티팩트 링크를 선택하여 빌드 출력 아티팩트를 가져올 수도 있습니다. 이 페이지로 이동하려면 이 절차의 나머지 단계를 건너뛰고 빌드 세부 정보 보기(콘솔) 섹션을 참조하세요.

15. https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
16. 버킷 목록에서 파이프라인에서 사용하는 버킷을 엽니다. 버킷의 이름은 `codepipeline-`*`region-ID`*`-`*`random-number`* 형식을 따릅니다. AWS CLI 를 사용하여 CodePipeline get-pipeline 명령을 실행하여 버킷의 이름을 가져올 수 있습니다. 여기서 *`my-pipeline-name`*은 파이프라인의 표시 이름입니다.

```
aws codepipeline get-pipeline --name my-pipeline-name
```

출력에서 `pipeline` 객체는 `artifactStore` 객체를 포함하며, 이 객체에는 버킷 이름의 `location` 값이 들어 있습니다.

17. 파이프라인의 이름과 일치하는 폴더를 열고(파이프라인의 이름 길이에 따라 폴더 이름이 잘릴 수 있음) 앞에서 적어 둔 출력 아티팩트 값과 일치하는 폴더를 엽니다.
18. 파일 내용의 압축을 풉니다. 해당 폴더에 파일이 여러 개 있는 경우 가장 최근의 마지막 수정 시간 타임스탬프를 사용하여 파일의 내용을 추출합니다. (시스템의 ZIP 유틸리티에서 작업할 수 있도록 파일에 `.zip` 확장자를 지정해야 할 수도 있습니다.) 빌드 출력 아티팩트는 파일의 추출된 내용에 있습니다.
19. CodePipeline에 빌드 출력 아티팩트를 배포하도록 지시한 경우 배포 공급자의 지침을 사용하여 배포 대상의 빌드 출력 아티팩트로 이동합니다.

CodeBuild를 사용하는 파이프라인 생성(AWS CLI)

다음 절차에 따라 CodeBuild를 사용하여 소스 코드를 빌드하는 파이프라인을 생성하세요.

AWS CLI 를 사용하여 빌드된 소스 코드를 배포하거나 소스 코드만 테스트하는 파이프라인을 생성하려면 파이프라인 편집(AWS CLI)의 지침과 AWS CodePipeline 사용 설명서의 CodePipeline 파이프라인 구조 참조를 조정할 수 있습니다.

1. CodeBuild에서 빌드 프로젝트를 생성하거나 식별합니다. 자세한 내용은 빌드 프로젝트 생성 단원을 참조하십시오.

   > **Important**
   >
   > CodePipeline에서 재정의되더라도 빌드 프로젝트는 빌드 출력 아티팩트 설정을 정의해야 합니다. 자세한 내용은 빌드 프로젝트 생성(AWS CLI)의 `artifacts` 설명을 참조하십시오.

2. 이 주제에 설명된 IAM 엔터티 중 하나에 해당하는 AWS 액세스 키 및 AWS 보안 액세스 키 AWS CLI 로를 구성했는지 확인합니다. 자세한 내용은AWS Command Line Interface 사용 설명서의 AWS Command Line Interface설정을 참조하세요.
3. 파이프라인 구조를 나타내는 JSON 형식의 파일을 생성합니다. 파일 이름을 `create-pipeline.json` 또는 비슷한 이름으로 지정합니다. 예를 들어, 다음 JSON 형식 구조는

CodeBuild를 사용하는 빌드 작업 및 S3 입력 버킷을 참조하는 소스 작업으로 파이프라인을 생성합니다.

```
{
  "pipeline": {
    "roleArn": "arn:aws:iam::<account-id>:role/<AWS-CodePipeline-service-role-name>",
    "stages": [
      {
        "name": "Source",
        "actions": [
          {
            "inputArtifacts": [],
            "name": "Source",
            "actionTypeId": {
              "category": "Source",
              "owner": "AWS",
              "version": "1",
              "provider": "S3"
            },
            "outputArtifacts": [
              {
                "name": "MyApp"
              }
            ],
            "configuration": {
              "S3Bucket": "<bucket-name>",
              "S3ObjectKey": "<source-code-file-name.zip>"
            },
            "runOrder": 1
          }
        ]
      },
      {
        "name": "Build",
        "actions": [
          {
            "inputArtifacts": [
              {
                "name": "MyApp"
              }
            ],
            "name": "Build",
```

```
            "actionTypeId": {
              "category": "Build",
              "owner": "AWS",
              "version": "1",
              "provider": "CodeBuild"
            },
            "outputArtifacts": [
              {
                "name": "default"
              }
            ],
            "configuration": {
              "ProjectName": "<build-project-name>"
            },
            "runOrder": 1
          }
        ]
      }
    ],
    "artifactStore": {
      "type": "S3",
      "location": "<CodePipeline-internal-bucket-name>"
    },
    "name": "<my-pipeline-name>",
    "version": 1
  }
}
```

이 JSON 형식의 데이터에서는 다음이 적용됩니다.

- `roleArn`의 값은 사용자가 생성했거나 사전 요구 사항의 일부로 식별한 CodePipeline 서비스 역할의 ARN과 일치해야 합니다.
- `configuration`의 `S3Bucket` 및 `S3ObjectKey` 값은 소스 코드가 S3 버킷에 저장되어 있다고 가정합니다. 다른 소스 코드 리포지토리 유형에 대한 설정은AWS CodePipeline 사용 설명서의 CodePipeline 파이프라인 구조 참조를 참조하세요.
- `ProjectName`의 값은 이 절차의 앞부분에서 생성한 CodeBuild 빌드 프로젝트의 이름입니다.
- `location`의 값은 이 파이프라인에서 사용하는 S3 버킷의 이름입니다. 자세한 내용은AWS CodePipeline 사용 설명서의 CodePipeline의 아티팩트 스토어로 사용할 S3 버킷에 대한 정책 생성을 참조하세요.
- `name`의 값은 이 파이프라인의 이름입니다. 모든 파이프라인 이름은 계정에서 고유해야 합니다.

이 데이터는 소스 작업과 빌드 작업만 설명하지만 테스트, 빌드 출력 아티팩트 배포, AWS Lambda 함수 호출 등과 관련된 활동에 대한 작업을 추가할 수 있습니다. 자세한 내용은AWS CodePipeline 사용 설명서의 AWS CodePipeline 파이프라인 구조 참조를 참조하세요.

4. JSON 파일이 들어 있는 폴더로 전환한 다음, 파일 이름을 지정하여 create-pipeline CodePipeline 명령을 실행합니다.

```
aws codepipeline create-pipeline --cli-input-json file://create-pipeline.json
```

> **Note**
>
> CodeBuild가 지원되는 AWS 리전에서 파이프라인을 생성해야 합니다. 자세한 내용은 Amazon Web Services 일반 참조의 AWS CodeBuild 섹션을 참조하세요.

JSON 형식의 데이터가 출력에 나타나고 CodePipeline이 파이프라인을 생성합니다.

5. 파이프라인 상태에 대한 정보를 가져오려면 파이프라인 이름을 지정하여 CodePipeline get-pipeline-state 명령을 실행합니다.

```
aws codepipeline get-pipeline-state --name <my-pipeline-name>
```

출력에서 빌드가 성공했음을 확인하는 정보를 찾습니다. 간결하게 나타내기 위해 생략된 데이터를 표시하는 데 줄임표(...)가 사용됩니다.

```
{
  ...
  "stageStates": [
    ...
    {
      "actionStates": [
        {
          "actionName": "CodeBuild",
          "latestExecution": {
            "status": "SUCCEEDED",
            ...
          },
          ...
        }
```

```
      ]
    }
  ]
}
```

이 명령을 너무 일찍 실행하면 빌드 작업에 대해 어떤 정보도 표시되지 않을 수 있습니다. 파이프라인에서 빌드 작업 실행이 완료될 때까지 이 명령을 여러 번 실행해야 할 수도 있습니다.

6. 빌드에 성공하면 다음 지침에 따라 빌드 출력 아티팩트를 가져오세요. https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.

> **Note**
>
> CodeBuild 콘솔의 관련 빌드 세부 정보 페이지에서 빌드 아티팩트 링크를 선택하여 빌드 출력 아티팩트를 가져올 수도 있습니다. 이 페이지로 이동하려면 이 절차의 나머지 단계를 건너뛰고 빌드 세부 정보 보기(콘솔) 섹션을 참조하세요.

7. 버킷 목록에서 파이프라인에서 사용하는 버킷을 엽니다. 버킷의 이름은 `codepipeline-<region-ID>-<random-number>` 형식을 따릅니다. `create-pipeline.json` 파일에서 버킷 이름을 가져오거나 CodePipeline get-pipeline 명령을 실행하여 버킷 이름을 가져올 수 있습니다.

```
aws codepipeline get-pipeline --name <pipeline-name>
```

출력에서 `pipeline` 객체는 `artifactStore` 객체를 포함하며, 이 객체에는 버킷 이름의 `location` 값이 들어 있습니다.

8. 파이프라인 이름(예: *<pipeline-name>*)과 일치하는 폴더를 엽니다.
9. 해당 폴더에서 이름이 `default`인 폴더를 엽니다.
10. 파일 내용의 압축을 풉니다. 해당 폴더에 파일이 여러 개 있는 경우 가장 최근의 마지막 수정 시간 타임스탬프를 사용하여 파일의 내용을 추출합니다. (시스템의 ZIP 유틸리티에서 작업할 수 있도록 파일에 `.zip` 확장자를 지정해야 할 수도 있습니다.) 빌드 출력 아티팩트는 파일의 추출된 내용에 있습니다.

CodeBuild 빌드 작업을 파이프라인에 추가(CodePipeline 콘솔)

1. 다음을 사용하여 AWS Management Console 에 로그인합니다.

- AWS 루트 계정. 이는 권장하지 않습니다. 자세한 내용은 사용 설명서의 계정 루트 사용자를 참조하세요.
- AWS 계정의 관리자 사용자입니다. 자세한 내용은 사용 설명서의 첫 번째 AWS 계정 루트 사용자 및 그룹 생성을 참조하세요.
- AWS 계정에서 다음과 같은 최소 작업 세트를 수행할 수 있는 권한이 있는 사용자:

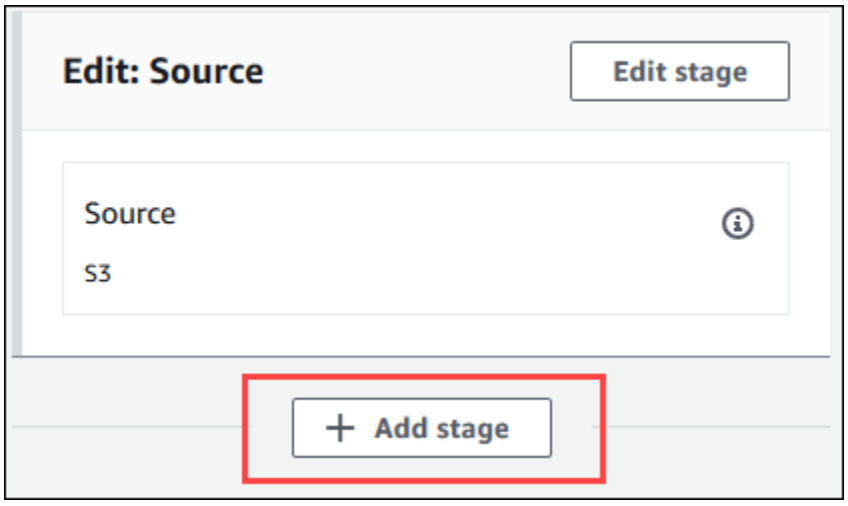
```
codepipeline:*
iam:ListRoles
iam:PassRole
s3:CreateBucket
s3:GetBucketPolicy
s3:GetObject
s3:ListAllMyBuckets
s3:ListBucket
s3:PutBucketPolicy
codecommit:ListBranches
codecommit:ListRepositories
codedeploy:GetApplication
codedeploy:GetDeploymentGroup
codedeploy:ListApplications
codedeploy:ListDeploymentGroups
elasticbeanstalk:DescribeApplications
elasticbeanstalk:DescribeEnvironments
lambda:GetFunctionConfiguration
lambda:ListFunctions
opsworks:DescribeStacks
opsworks:DescribeApps
opsworks:DescribeLayers
```

2. https://console.aws.amazon.com/codesuite/codepipeline/home에서 CodePipeline 콘솔을 엽니다.
3. AWS 리전 선택기에서 파이프라인이 위치한 AWS 리전을 선택합니다. 이는 CodeBuild가 지원되는 리전이어야 합니다. 자세한 내용은 Amazon Web Services 일반 참조의 CodeBuild를 참조하세요.
4. 파이프라인 페이지에서 파이프라인의 이름을 선택합니다.
5. 파이프라인 세부 정보 페이지의 소스 작업에서 도구 설명을 선택합니다. 출력 아티팩트(예: MyApp)의 값을 적어 놓습니다.

> **Note**
>
> 이 절차에서는 소스와 베타 단계 사이의 빌드 단계에 빌드 작업을 추가하는 방법을 보여 줍니다. 빌드 작업을 다른 위치에 추가하려면 빌드 작업을 추가할 위치 바로 앞에 있는 작업의 도구 설명을 선택하고 출력 아티팩트 값을 기록해 둡니다.

6. 편집을 선택합니다.
7. 소스 단계와 베타 단계 사이에서 단계 추가를 선택합니다.

> **Note**
>
> 이 절차에서는 소스와 베타 단계 사이의 빌드 단계를 파이프라인에 추가하는 방법을 보여줍니다. 기존 단계에 빌드 작업을 추가하려면 단계에서 단계 편집을 선택한 다음, 이 절차의 8단계로 건너뜁니다. 빌드 단계를 다른 위치에 추가하려면 원하는 위치에서 단계 추가를 선택합니다.



8. 단계 이름에 빌드 단계 이름(예: **`Build`**)을 입력합니다. 다른 이름을 선택하는 경우 이 절차 전체에서 해당 이름을 사용합니다.
9. 선택한 단계 내에서 작업 추가를 선택합니다.

> **Note**
>
> 이 절차에서는 빌드 단계 내에 빌드 작업을 추가하는 방법을 보여줍니다. 빌드 작업을 다른 위치에 추가하려면 원하는 위치에서 작업 추가를 선택합니다. 먼저 빌드 작업을 추가하려는 기존 단계에서 단계 편집을 선택해야 할 수 있습니다.

10. 작업 편집의 작업 이름에 작업 이름을 입력합니다(예: **`CodeBuild`**). 다른 이름을 선택하는 경우 이 절차 전체에서 해당 이름을 사용합니다.
11. 작업 공급자의 경우 CodeBuild를 선택합니다.
12. 사용하려는 빌드 프로젝트가 이미 있는 경우 프로젝트 이름에서 빌드 프로젝트의 이름을 선택하고 이 절차의 다음 단계로 건너뜁니다.

    새 CodeBuild 빌드 프로젝트를 생성해야 하는 경우 빌드 프로젝트 만들기(콘솔)의 지침을 따르고 이 절차로 돌아갑니다.

    기존 빌드 프로젝트를 선택하는 경우 CodePipeline에서 재정의되더라도 빌드 출력 아티팩트 설정이 이미 정의되어 있어야 합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 또는 빌드 프로젝트 설정 변경(콘솔)의 아티팩트 설명을 참조하세요.

    > **Important**
    >
    > CodeBuild 프로젝트에 대해 webhook를 활성화하고 해당 프로젝트가 CodePipeline의 빌드 단계로 사용되는 경우 각 커밋에 대해 두 개의 동일한 빌드가 생성됩니다. 하나의 빌드는 webhook를 통해 트리거되고 다른 하나는 CodePipeline을 통해 트리거됩니다. 빌드 기준으로 요금이 청구되므로 두 빌드 모두에 대해 요금이 청구됩니다. 따라서 CodePipeline을 사용하는 경우 CodeBuild에서 webhook를 비활성화하는 것이 좋습니다. CodeBuild 콘솔에서 Webhook 상자를 선택 취소합니다. 자세한 내용은 빌드 프로젝트 설정 변경(콘솔) 섹션을 참조하세요.

13. 입력 아티팩트에서 이 절차의 앞에서 적어 둔 출력 아티팩트를 선택합니다.
14. 출력 아티팩트의 경우 출력 아티팩트의 이름(예: **`MyAppBuild`**)을 입력합니다.
15. 작업 추가를 선택합니다.
16. 저장을 선택한 다음, 저장을 선택하여 변경 사항을 파이프라인에 저장합니다.
17. 변경 사항 릴리스를 선택합니다.

18. 파이프라인이 성공적으로 실행되면 빌드 출력 아티팩트를 가져올 수 있습니다. CodePipeline 콘솔에 파이프라인이 표시된 상태에서 빌드 작업에서 도구 설명을 선택합니다. 출력 아티팩트(예: MyAppBuild)의 값을 적어 놓습니다.

    > **Note**
    >
    > CodeBuild 콘솔의 빌드 세부 정보 페이지에서 빌드 아티팩트 링크를 선택하여 빌드 출력 아티팩트를 가져올 수도 있습니다. 이 페이지로 이동하려면 빌드 세부 정보 보기(콘솔)을 참조하고 이 절차의 31단계로 건너뛰세요.

19. https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
20. 버킷 목록에서 파이프라인에서 사용하는 버킷을 엽니다. 버킷의 이름은 `codepipeline-`*`region-ID`*`-`*`random-number`* 형식을 따릅니다. AWS CLI 를 사용하여 CodePipeline get-pipeline 명령을 실행하여 버킷의 이름을 가져올 수 있습니다.

    ```
    aws codepipeline get-pipeline --name my-pipeline-name
    ```

    출력에서 `pipeline` 객체는 `artifactStore` 객체를 포함하며, 이 객체에는 버킷 이름의 `location` 값이 들어 있습니다.
21. 파이프라인의 이름과 일치하는 폴더를 열고(파이프라인의 이름 길이에 따라 폴더 이름이 잘릴 수 있음) 이 절차의 앞에서 적어 둔 출력 아티팩트 값과 일치하는 폴더를 엽니다.
22. 파일 내용의 압축을 풉니다. 해당 폴더에 파일이 여러 개 있는 경우 가장 최근의 마지막 수정 시간 타임스탬프를 사용하여 파일의 내용을 추출합니다. (시스템의 ZIP 유틸리티에서 작업할 수 있도록 파일에 `.zip` 확장자를 지정해야 할 수도 있습니다.) 빌드 출력 아티팩트는 파일의 추출된 내용에 있습니다.
23. CodePipeline에 빌드 출력 아티팩트를 배포하도록 지시한 경우 배포 공급자의 지침을 사용하여 배포 대상의 빌드 출력 아티팩트로 이동합니다.

CodeBuild 테스트 작업을 파이프라인에 추가(CodePipeline 콘솔)

1. 다음을 사용하여 AWS Management Console 에 로그인합니다.

    - AWS 루트 계정. 이는 권장하지 않습니다. 자세한 내용은 사용 설명서의 계정 루트 사용자를 참조하세요.
    - AWS 계정의 관리자 사용자입니다. 자세한 내용은 사용 설명서의 첫 번째 AWS 계정 루트 사용자 및 그룹 생성을 참조하세요.

- AWS 계정에서 다음과 같은 최소 작업 세트를 수행할 수 있는 권한이 있는 사용자:

```
codepipeline:*
iam:ListRoles
iam:PassRole
s3:CreateBucket
s3:GetBucketPolicy
s3:GetObject
s3:ListAllMyBuckets
s3:ListBucket
s3:PutBucketPolicy
codecommit:ListBranches
codecommit:ListRepositories
codedeploy:GetApplication
codedeploy:GetDeploymentGroup
codedeploy:ListApplications
codedeploy:ListDeploymentGroups
elasticbeanstalk:DescribeApplications
elasticbeanstalk:DescribeEnvironments
lambda:GetFunctionConfiguration
lambda:ListFunctions
opsworks:DescribeStacks
opsworks:DescribeApps
opsworks:DescribeLayers
```

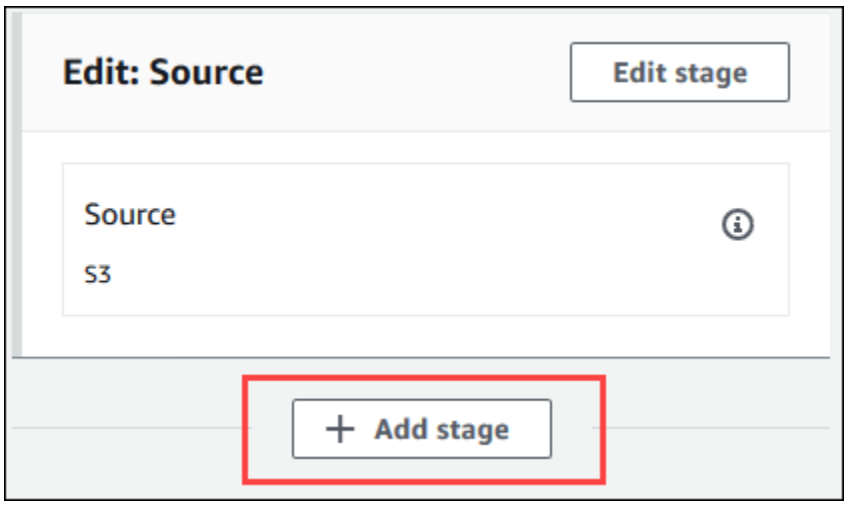
2. https://console.aws.amazon.com/codesuite/codepipeline/home에서 CodePipeline 콘솔을 엽니다.
3. AWS 리전 선택기에서 파이프라인이 위치한 AWS 리전을 선택합니다. CodeBuild가 지원되는 AWS 리전이어야 합니다. 자세한 내용은 Amazon Web Services 일반 참조의 AWS CodeBuild 섹션을 참조하세요.
4. 파이프라인 페이지에서 파이프라인의 이름을 선택합니다.
5. 파이프라인 세부 정보 페이지의 소스 작업에서 도구 설명을 선택합니다. 출력 아티팩트(예: MyApp)의 값을 적어 놓습니다.

> **Note**
>
> 이 절차에서는 소스와 베타 단계 사이의 테스트 단계에 테스트 작업을 추가하는 방법을 보여 줍니다. 테스트 작업을 다른 위치에 추가하려면 마우스 포인터를 바로 앞에 있는 작업에 놓고 출력 아티팩트의 값을 기록해 둡니다.

6. 편집을 선택합니다.

7. 소스 단계 바로 다음에 단계 추가를 선택합니다.

> **Note**
>
> 또한 이 절차에서는 소스 단계 바로 다음의 테스트 단계를 파이프라인에 추가하는 방법을 보여줍니다. 기존 단계에 테스트 작업을 추가하려면 단계에서 단계 편집을 선택한 다음, 이 절차의 8단계로 건너뜁니다. 테스트 단계를 다른 위치에 추가하려면 원하는 위치에서 단계 추가를 선택합니다.



8. 단계 이름에 테스트 단계 이름(예: **Test**)을 입력합니다. 다른 이름을 선택하는 경우 이 절차 전체에서 해당 이름을 사용합니다.
9. 선택한 단계에서 작업 추가를 선택합니다.

> **Note**
>
> 이 절차에서는 테스트 단계에 테스트 작업을 추가하는 방법을 보여 줍니다. 테스트 작업을 다른 위치에 추가하려면 원하는 위치에서 작업 추가를 선택합니다. 테스트 작업을 추가하려는 기존 단계에서 먼저 편집을 선택해야 할 수도 있습니다.

10. 작업 편집의 작업 이름에 작업 이름을 입력합니다(예: **Test**). 다른 이름을 선택하는 경우 이 절차 전체에서 해당 이름을 사용합니다.
11. 작업 제공자의 경우 테스트에서 CodeBuild를 선택합니다.
12. 사용하려는 빌드 프로젝트가 이미 있는 경우 프로젝트 이름에서 빌드 프로젝트의 이름을 선택하고 이 절차의 다음 단계로 건너뜁니다.

새 CodeBuild 빌드 프로젝트를 생성해야 하는 경우 빌드 프로젝트 만들기(콘솔)의 지침을 따르고 이 절차로 돌아갑니다.

> **Important**
>
> CodeBuild 프로젝트에 대해 webhook를 활성화하고 해당 프로젝트가 CodePipeline의 빌드 단계로 사용되는 경우 각 커밋에 대해 두 개의 동일한 빌드가 생성됩니다. 하나의 빌드는 webhook를 통해 트리거되고 다른 하나는 CodePipeline을 통해 트리거됩니다. 빌드 기준으로 요금이 청구되므로 두 빌드 모두에 대해 요금이 청구됩니다. 따라서 CodePipeline을 사용하는 경우 CodeBuild에서 webhook를 비활성화하는 것이 좋습니다. CodeBuild 콘솔에서 Webhook 상자를 선택 취소합니다. 자세한 내용은 빌드 프로젝트 설정 변경(콘솔) 섹션을 참조하세요.

13. 입력 아티팩트에 이 절차의 앞에서 적어 둔 출력 아티팩트 값을 입력합니다.
14. (선택 사항) 테스트 작업에서 출력 결과물을 생성하도록 하려고 하며 이에 맞게 빌드 사양을 설정했다면, 출력 아티팩트에 출력 결과물에 지정하려는 값을 입력합니다.
15. 저장을 선택합니다.
16. 변경 사항 릴리스를 선택합니다.
17. 파이프라인이 성공적으로 실행되면 테스트 결과를 얻을 수 있습니다. 파이프라인의 테스트 단계에서 CodeBuild 하이퍼링크를 선택하여 CodeBuild 콘솔에서 관련 빌드 프로젝트 페이지를 엽니다.
18. 빌드 프로젝트 페이지의 빌드 이력에서 빌드 실행 하이퍼링크를 선택합니다.
19. 빌드 실행 페이지의 빌드 로그에서 전체 로그 보기 하이퍼링크를 선택하여 Amazon CloudWatch 콘솔에서 빌드 로그를 엽니다.
20. 빌드 로그를 스크롤하여 테스트 결과를 확인합니다.

Codecov AWS CodeBuild 와 함께 사용

Codecov는 코드의 테스트 범위를 측정하는 도구입니다. Codecov는 코드에서 테스트되지 않은 메서드와 문을 식별합니다. 결과를 사용하여 코드의 품질을 향상시키기 위해 테스트를 작성할 위치를 결정합니다. Codecov는 CodeBuild에서 지원되는 3개 소스 리포지토리인 GitHub, GitHub Enterprise Server 및 Bitbucket을 사용할 수 있습니다. 빌드 프로젝트가 GitHub Enterprise Server를 사용하는 경우 Codecov Enterprise를 사용해야 합니다.

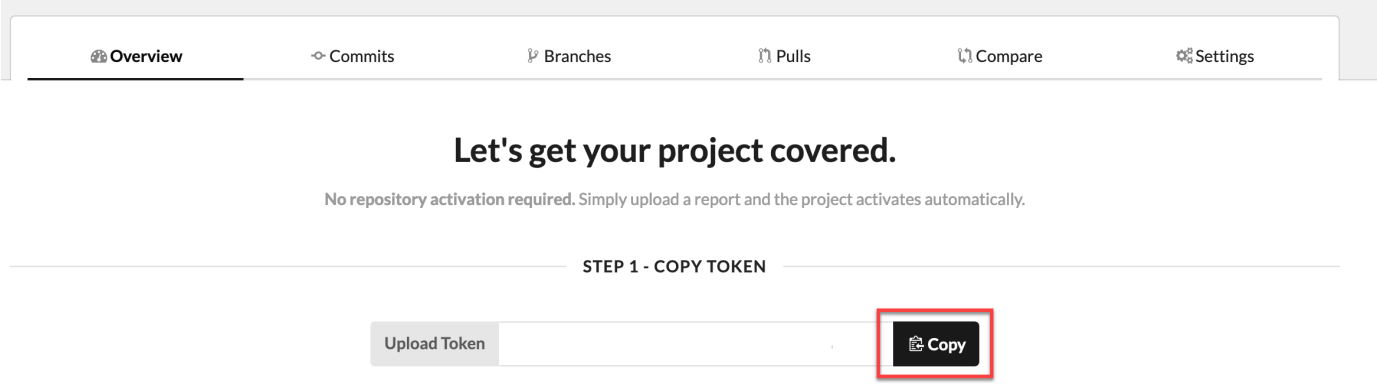
Codecov와 통합된 CodeBuild 프로젝트의 빌드를 실행하면 리포지토리의 코드를 분석하는 Codecov 보고서가 Codecov에 업로드됩니다. 빌드 로그에는 보고서로 연결되는 링크가 있습니다. 이 샘플은 Python과 Java 빌드 프로젝트를 Codecov와 통합하는 방법을 보여줍니다. Codecov에서 지원하는 언어 목록은 Codecov 웹사이트의 Codecov 지원 언어를 참조하세요.

Codecov를 빌드 프로젝트와 통합

다음 절차에 따라 Codecov를 빌드 프로젝트에 통합합니다.

Codecov를 빌드 프로젝트에 통합

1. https://codecov.io/signup으로 이동하여 GitHub 또는 Bitbucket 소스 리포지토리에 등록합니다. GitHub Enterprise를 사용하는 경우 Codecov 웹사이트의 Codecov Enterprise를 참조하세요.
2. Codecov에서 적용 범위를 원하는 리포지토리를 추가합니다.
3. 토큰 정보가 표시되면 복사(Copy)를 선택합니다.



4. 복사된 토큰을 빌드 프로젝트에 이름이 `CODECOV_TOKEN`인 환경 변수로 추가합니다. 자세한 내용은 빌드 프로젝트 설정 변경(콘솔) 단원을 참조하십시오.
5. 리포지토리에서 `my_script.sh`(이)라는 텍스트 파일을 생성합니다. 다음을 파일에 입력합니다.

```
#/bin/bash
bash <(curl -s https://codecov.io/bash) -t $CODECOV_TOKEN
```

6. 빌드 프로젝트 사용에 적합한 Python 또는 Java 탭을 선택하고 다음 단계를 수행합니다.

   Java

   1. 다음 JaCoCo 플러그인을 리포지토리의 `pom.xml`에 추가합니다.

```
<build>
  <plugins>
```

```
    <plugin>
      <groupId>org.jacoco</groupId>
      <artifactId>jacoco-maven-plugin</artifactId>
      <version>0.8.2</version>
      <executions>
          <execution>
              <goals>
                  <goal>prepare-agent</goal>
              </goals>
          </execution>
          <execution>
              <id>report</id>
              <phase>test</phase>
              <goals>
                  <goal>report</goal>
              </goals>
          </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```

2. buildspec 파일에 다음 명령을 입력합니다. 자세한 내용은 buildspec 구문 단원을 참조하십시오.

```
build:
  - mvn test -f pom.xml -fn
postbuild:
  - echo 'Connect to CodeCov'
  - bash my_script.sh
```

Python

buildspec 파일에 다음 명령을 입력합니다. 자세한 내용은 buildspec 구문 단원을 참조하십시오.

```
build:
  - pip install coverage
  - coverage run -m unittest discover
postbuild:
  - echo 'Connect to CodeCov'
```

```
  - bash my_script.sh
```

7. 빌드 프로젝트의 빌드를 실행합니다. 프로젝트에 대해 생성된 Codecov 보고서에 대한 링크가 빌드 로그에 나타납니다. Codecov 보고서를 보려면 링크를 사용합니다. 자세한 내용은 수동으로 AWS CodeBuild 빌드 실행 및 를 사용하여 AWS CodeBuild API 호출 로깅 AWS CloudTrail 섹션을 참조하세요. 빌드 로그의 Codecov 정보는 다음과 같습니다.

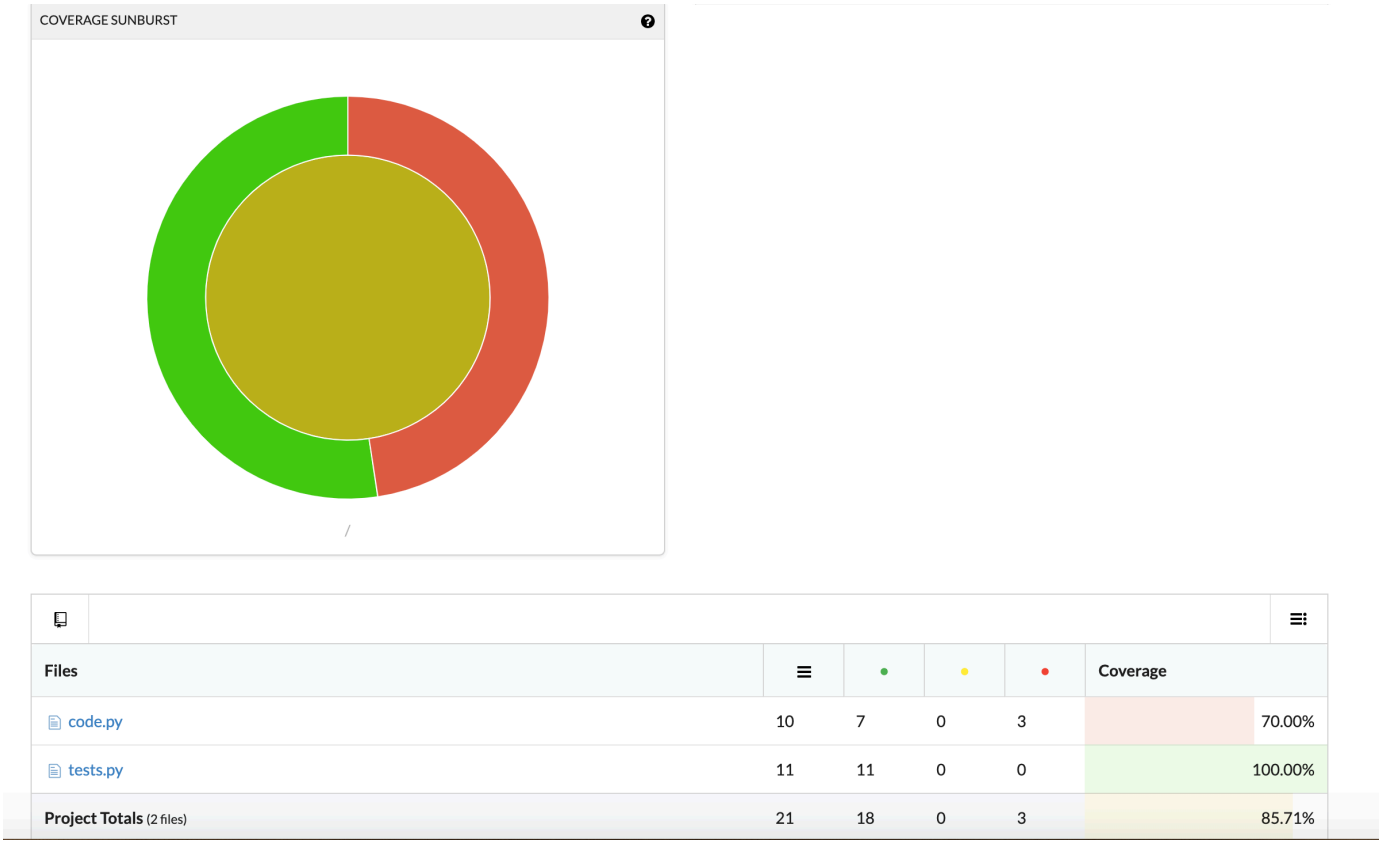
```
[Container] 2020/03/09 16:31:04 Running command bash my_script.sh


  _____          _
 / ____|        | |
| |     ___   __| | ___  ___ _____   __
| |    / _ \ / _` |/ _ \/ __/ _ \ \ / /
| |___| (_) | (_| |  __/ (_| (_) \ V /
 \_____\___/ \__,_|\___|\___\___/ \_/
                              Bash-20200303-bc4d7e6

·[0;90m==>·[0m AWS CodeBuild detected.
... The full list of Codecov log entries has been omitted for brevity ...
    ·
    ·[0;32m->·[0m View reports at ·[0;36mhttps://codecov.io/github/user/test_py/
commit/commit-id·[0m

[Container] 2020/03/09 16:31:07 Phase complete: POST_BUILD State: SUCCEEDED
```

보고서는 다음과 같습니다.



Jenkins와 AWS CodeBuild 함께 사용

용 Jenkins 플러그인 AWS CodeBuild 을 사용하여 CodeBuild를 Jenkins 빌드 작업과 통합할 수 있습니다. 빌드 작업을 Jenkins 빌드 노드로 보내는 대신, 이 플러그인을 사용하여 CodeBuild로 빌드 작업을 전송합니다. 따라서 Jenkins 빌드 노드를 프로비저닝, 구성 및 관리할 필요가 없습니다.

주제

- Jenkins 설정
- 플러그인 설치
- 플러그인 사용

Jenkins 설정

AWS CodeBuild 플러그인을 사용하여 Jenkins를 설정하고 플러그인 소스 코드를 다운로드하는 방법에 대한 자세한 내용은 https://github.com/awslabs/aws-codebuild-jenkins-plugin 참조하십시오.

플러그인 설치

Jenkins 서버를 이미 설정했고 AWS CodeBuild 플러그인만 설치하려는 경우 Jenkins 인스턴스의 Plugin Manager에서 **`CodeBuild Plugin for Jenkins`**를 검색합니다.

플러그인 사용

VPC 외부의 소스와 AWS CodeBuild 함께를 사용하려면

1. CodeBuild 콘솔에서 프로젝트를 생성합니다. 자세한 내용은 빌드 프로젝트 만들기(콘솔) 단원을 참조하십시오.
   - 빌드를 실행할 AWS 리전을 선택합니다.
   - (선택 사항) CodeBuild 빌드 컨테이너가 VPC의 리소스에 액세스할 수 있도록 Amazon VPC 구성을 설정합니다.
   - 프로젝트 이름을 적어 둡니다. 3단계에서 이 이름이 필요합니다.
   - (선택 사항) CodeBuild에서 소스 리포지토리를 기본적으로 지원하지 않는 경우 Amazon S3를 프로젝트의 입력 소스 유형으로 설정할 수 있습니다.
2. IAM 콘솔에서 Jenkins 플러그인에서 사용할 사용자를 생성합니다.
   - 사용자에 대한 보안 인증을 생성할 때는 프로그래밍 방식 액세스를 선택합니다.
   - 다음과 유사한 정책을 만든 다음, 정책을 사용자에게 연결합니다.

     JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:logs:us-east-1:111122223333:log-group:/aws/
codebuild/{{projectName}}:*"
            ],
            "Action": [
                "logs:GetLogEvents"
            ]
        },
        {
            "Effect": "Allow",
```

```
            "Resource": [
                "arn:aws:s3:::{{inputBucket}}"
            ],
            "Action": [
                "s3:GetBucketVersioning"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::{{inputBucket}}/{{inputObject}}"
            ],
            "Action": [
                "s3:PutObject"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::{{outputBucket}}/*"
            ],
            "Action": [
                "s3:GetObject"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:codebuild:us-east-1:111122223333:project/
{{projectName}}"
            ],
            "Action": [
                "codebuild:StartBuild",
                "codebuild:BatchGetBuilds",
                "codebuild:BatchGetProjects"
            ]
        }
    ]
}
```

3. Jenkins에서 자유형 프로젝트를 생성합니다.

   - 구성 페이지에서 빌드 단계 추가를 선택한 다음, CodeBuild에서 빌드 실행을 선택합니다.

- 빌드 단계를 구성합니다.
  - 리전, 보안 인증, 프로젝트 이름에 값을 입력합니다.
  - 프로젝트 소스 사용을 선택합니다.
  - 구성을 저장하고 Jenkins에서 빌드를 실행합니다.

4. 소스 코드 관리에서 원하는 소스 검색 방법을 선택합니다. Jenkins 서버에 GitHub 플러그인(또는 소스 리포지토리 공급자용 Jenkins 플러그인)을 설치해야 할 수 있습니다.

- 구성 페이지에서 빌드 단계 추가를 선택한 다음, AWS CodeBuild에서 빌드 실행을 선택합니다.
- 빌드 단계를 구성합니다.
  - 리전, 보안 인증, 프로젝트 이름에 값을 입력합니다.
  - Jenkins 소스 사용을 선택합니다.
  - 구성을 저장하고 Jenkins에서 빌드를 실행합니다.

Jenkins 파이프라인 AWS CodeBuild 플러그인과 함께 플러그인을 사용하려면

- Jenkins 파이프라인 프로젝트 페이지에서 스니펫 생성기를 사용하여 CodeBuild를 파이프라인의 단계로 추가하는 파이프라인 스크립트를 생성합니다. 다음과 유사한 스크립트가 생성됩니다.

```
awsCodeBuild projectName: 'project', credentialsType: 'keys', region: 'us-west-2',
 sourceControlType: 'jenkins'
```

서버리스 애플리케이션 AWS CodeBuild 과 함께 사용

AWS Serverless Application Model (AWS SAM)는 서버리스 애플리케이션을 빌드하기 위한 오픈 소스 프레임워크입니다. 자세한 내용은 GitHub에서 AWS Serverless Application Model 리포지토리를 참조하십시오.

AWS CodeBuild 를 사용하여 표준에 따라 서버리스 애플리케이션을 패키징하고 배포할 수 있습니다 AWS SAM . 배포 단계에는 CodeBuild에서 AWS CloudFormation을 사용할 수 있습니다. CodeBuild 및 를 사용하여 서버리스 애플리케이션의 빌드 및 배포를 자동화하려면를 사용하면 CloudFormation됩니다 AWS CodePipeline.

자세한 내용을 알아보려면 AWS Serverless Application Model 개발자 안내서의 서버리스 애플리케이션 배포를 참조하세요.

관련 리소스

- 시작하기에 대한 자세한 내용은 단원을 AWS CodeBuild참조하십시오콘솔 AWS CodeBuild 사용 시작하기.
- CodeBuild의 문제 해결에 대한 자세한 내용은 문제 해결 AWS CodeBuild 섹션을 참조하세요.
- CodeBuild의 할당량에 대한 자세한 내용은 에 대한 할당량 AWS CodeBuild 섹션을 참조하세요.

Windows AWS CodeBuild 용에 대한 타사 알림

Windows 빌드용 CodeBuild를 사용하면 타사 패키지 및 모듈을 사용하여 빌드된 애플리케이션을 Microsoft Windows 운영 체제에서 실행하고 타사 제품과 상호 작용할 수 있게 해주는 옵션이 제공됩니다. 다음 목록에는 지정된 타사 패키지 및 모듈의 사용에 적용되는 타사 법률 약관이 포함되어 있습니다.

주제

- 1) 기본 도커 이미지 - windowsservercore
- 2) Windows 기반 도커 이미지 - choco
- 3) Windows 기반 도커 이미지 – git --버전 2.16.2
- 4) Windows 기반 도커 이미지 - microsoft-build-tools --버전 15.0.26320.2
- 5) Windows 기반 도커 이미지 - nuget.commandline --버전 4.5.1
- 7) Windows 기반 도커 이미지 - netfx-4.6.2-devpack
- 8) Windows 기반 도커 이미지 - visualfsharptools, v 4.0
- 9) Windows 기반 도커 이미지 - netfx-pcl-reference-assemblies-4.6
- 10) Windows 기반 도커 이미지 - visualcppbuildtools v 14.0.25420.1
- 11) Windows 기반 도커 이미지 - microsoft-windows-netfx3-ondemand-package.cab
- 12) Windows 기반 도커 이미지 - dotnet-sdk

1) 기본 도커 이미지 - windowsservercore

(라이선스 약관은 https://hub.docker.com/_/microsoft-windows-servercore에서 확인할 수 있습니다)

라이선스: 이 Windows 컨테이너용 컨테이너 OS 이미지를 요청하고 사용할 경우 다음 추가 프로그램 라이선스 약관을 인정하고 이해하며 동의하는 것으로 간주됩니다.

MICROSOFT 소프트웨어 추가 프로그램 라이선스 약관

컨테이너 OS 이미지

Microsoft Corporation(또는 사는 지역을 기반으로 한 Microsoft 계열사 중 하나)(“우리”, “당사” 또는 “Microsoft”로 지칭)에서 이 컨테이너 OS 이미지 추가 프로그램(“추가 프로그램”)의 사용을 허가합니다. 귀하는 호스트 소프트웨어의 컨테이너 기능 실행을 지원하는 용도로만 이 추가 프로그램을 기본 호스트 운영 체제 소프트웨어(“호스트 소프트웨어”)와 함께 사용할 수 있습니다. 호스트 소프트웨어 라이선스 약관이 이 추가 프로그램의 사용에도 적용됩니다. 호스트 소프트웨어에 대한 라이선스가 없는 경우 이 소프트웨어를 사용할 수 없습니다. 유효한 라이선스가 부여된 각 호스트 소프트웨어 사본과 함께 이 추가 프로그램을 사용할 수 있습니다.

추가 라이선스 요구 사항 및/또는 사용 권한

이전 단락에 명시된 대로 추가 프로그램을 사용하면 특정 추가 프로그램 구성 요소를 포함하는 컨테이너 이미지가 생성되거나 수정될 수 있습니다. 명확히 설명하자면 컨테이너 이미지는 가상 머신 또는 가상 어플라이언스 이미지와는 별개로 구분됩니다. 본 라이선스 조건에 따라 당사는 다음 조건에서 해당 추가 프로그램 구성 요소를 재배포할 수 있는 제한된 권리를 사용자에게 부여합니다.

(i) 추가 프로그램 구성 요소는 컨테이너 이미지에 사용된 용도로만 사용할 수 있으며 컨테이너 이미지의 일부로만 사용할 수 있습니다.

(ii) 추가 프로그램과 실질적으로 분리되고 별개인 중요한 주요 기능이 컨테이너 이미지에 있는 한, 컨테이너 이미지에 이러한 추가 프로그램 구성 요소를 사용할 수 있습니다.

(iii) 귀하는 최종 사용자가 추가 프로그램 구성 요소를 사용할 수 있도록 하기 위해 컨테이너 이미지에 이러한 라이선스 약관(또는 당사 또는 호스팅 업체가 요구하는 유사한 조건)을 포함시키는 데 동의합니다.

당사는 여기에서 명시적으로 부여되지 않은 기타 모든 권리를 보유합니다.

이 추가 프로그램을 사용하면 이 약관에 동의하는 것으로 간주됩니다. 동의하지 않는 경우 이 추가 프로그램을 사용하지 마세요.

이 Windows 컨테이너용 컨테이너 OS 이미지에 대한 추가 라이선스 약관의 일부로 https://www.microsoft.com/en-us/useterms에 제공되는 기본 Windows Server 호스트 소프트웨어 라이선스 약관도 적용됩니다.

2) Windows 기반 도커 이미지 - choco

(라이선스 약관은 https://github.com/chocolatey/choco/blob/master/LICENSE 확인할 수 있음)

Copyright 2011 - Present RealDimensions Software, LLC

Apache 라이선스, 버전 2.0(이하 “라이선스”)에 따라 라이선스가 부여됩니다. 이 라이선스를 준수하는 경우를 제외하고는 이러한 파일을 사용할 수 없습니다. 라이선스 사본은 다음 사이트에서 구할 수 있습니다.

http://www.apache.org/licenses/LICENSE-2.0

관련 법률에서 요구하거나 서면으로 합의한 경우를 제외하고, 라이선스에 따라 배포된 소프트웨어는 명시적이든 묵시적이든 어떠한 종류의 보증이나 조건 없이 “있는 그대로” 배포됩니다. 라이선스에 따른 구체적인 언어 관리 권한과 제한 사항은 라이선스를 참조하십시오.

3) Windows 기반 도커 이미지 – git --버전 2.16.2

(라이선스 약관 참조: https://chocolatey.org/packages/git/2.16.2)

GNU 일반 공개 라이선스, 버전 2에 따라 사용이 허가되어 https://www.gnu.org/licenses/old-licenses/gpl-2.0.html에서 참조할 수 있습니다.

4) Windows 기반 도커 이미지 - microsoft-build-tools --버전 15.0.26320.2

(라이선스 약관 참조: https://www.visualstudio.com/license-terms/mt171552/)

MICROSOFT VISUAL STUDIO 2015 EXTENSIONS, VISUAL STUDIO SHELLS 및 C++ 재배포 가능 패키지

본 라이선스 약관은 Microsoft Corporation(또는 사는 지역을 기반으로 한 Microsoft 계열사 중 하나)과 귀하 사이의 계약입니다. 이 조건은 위에 명시된 소프트웨어에 적용됩니다. 본 조건은 추가 조건이 있는 경우를 제외하고 소프트웨어의 모든 Microsoft 서비스 또는 업데이트에도 적용됩니다.

본 라이선스 약관을 준수하는 경우 다음에 대해 권리가 있습니다.

1. 설치 및 사용 권한. 원하는 수만큼 소프트웨어 사본을 설치하고 사용할 수 있습니다.
2. 특정 구성 요소에 대한 조건.
   a. 유틸리티. 소프트웨어에는 유틸리티 목록(https://docs.microsoft.com/en-us/visualstudio/productinfo/2015-redistribution-vs)의 일부 항목이 포함될 수 있습니다. 소프트웨어에 포함된 경우 해당 항목을 사용자 또는 타사 컴퓨터에 복사하고 설치하여 소프트웨어로 개발한 애플리케이

션 및 데이터베이스를 디버그 및 배포할 수 있습니다. 유틸리티는 임시 사용을 위해 설계되었으므로 Microsoft가 나머지 소프트웨어와는 별도로 유틸리티를 패치하거나 업데이트하지 못할 수 있으며 일부 유틸리티는 그 특성상 다른 사용자가 해당 유틸리티가 설치된 컴퓨터에 액세스할 수 있다는 점에 유의하세요. 따라서 애플리케이션 및 데이터베이스의 디버깅 또는 배포를 마친 후에는 설치한 유틸리티를 모두 삭제해야 합니다. Microsoft는 사용자가 컴퓨터에 설치한 유틸리티의 제3자 사용 또는 액세스에 대해 책임을 지지 않습니다.

b. Microsoft 플랫폼. 소프트웨어에는 Microsoft Windows, Microsoft Windows Server, Microsoft SQL Server, Microsoft Exchange, Microsoft Office 및 Microsoft SharePoint의 구성 요소가 포함될 수 있습니다. 이러한 구성 요소에는 해당 구성 요소의 설치 디렉터리 또는 소프트웨어와 함께 제공되는 “Licenses” 폴더에 있는 라이선스 약관에 설명된 대로 별도의 계약 및 자체 제품 지원 정책이 적용됩니다.

c. 타사 구성 요소. 소프트웨어에는 소프트웨어와 함께 제공되는 ThirdPartyNotices 파일에 설명된 대로 별도의 법적 고지가 있거나 다른 계약의 적용을 받는 타사 구성 요소가 포함될 수 있습니다. 이러한 구성 요소가 다른 계약의 적용을 받는 경우에도 아래의 고지 사항, 손해 배상 제한 및 제외 조항이 함께 적용됩니다. 소프트웨어에는 소스 코드 가용성 의무가 있는 오픈 소스 라이선스에 따라 라이선스가 부여된 구성 요소도 포함될 수 있습니다. 해당하는 경우 해당 라이선스의 사본은 ThirdPartyNotices 파일에 포함되어 있습니다. 관련 오픈 소스 라이선스에 따라 필요한 경우, Microsoft Corporation, 1 Microsoft Way, Redmond, WA 98052의 Source Code Compliance Team으로 5.00 USD의 우편환 또는 수표를 보내면 Microsoft에서 이 소스 코드를 받을 수 있습니다. 아래 나열된 구성 요소 중 하나 이상의 소스 코드를 결제 메모란에 적어주세요.

   - Visual Studio 2015용 원격 도구
   - Visual Studio 2015용 독립 실행형 프로파일러
   - IntelliTraceCollector for Visual Studio 2015
   - Microsoft VC++ 재배포 가능 2015
   - Visual Studio 2015용 멀티바이트 MFC 라이브러리
   - Microsoft Build Tools 2015
   - Feedback Client
   - Visual Studio 2015 통합 쉘 또는
   - Visual Studio 2015 격리 쉘.

   소스 코드의 사본이 http://thirdpartysource.microsoft.com에 제공될 수도 있습니다.

3. 데이터. 소프트웨어에서 사용자 및 사용자의 소프트웨어 사용에 대한 정보를 수집하여 Microsoft에 보낼 수 있습니다. Microsoft는 이 정보를 사용하여 서비스를 제공하고 제품 및 서비스를 개선할 수 있습니다. 제품 설명서에 설명된 대로 이러한 시나리오를 대부분 옵트아웃할 수 있지만 전부

는 아닙니다. 또한 소프트웨어에는 애플리케이션 사용자로부터 데이터를 수집할 수 있는 몇 가지 기능이 있습니다. 이러한 기능을 사용하여 애플리케이션에서 데이터를 수집할 수 있게 하는 경우 애플리케이션 사용자에게 적절한 고지를 제공하는 것을 포함하여 관련 법률을 준수해야 합니다. 데이터 수집 및 사용에 대한 자세한 내용은 도움말 설명서 및 https://privacy.microsoft.com/en-us/privacystatement의 개인 정보 보호 정책에서 확인할 수 있습니다. 소프트웨어를 사용하면 이러한 정책에 동의하는 것으로 간주됩니다.

4. 라이선스 범위. 본 소프트웨어는 판매되는 것이 아니라 그 사용이 허여되는 것입니다. 본 계약은 소프트웨어를 사용할 수 있는 일부 권리만 제공합니다. Microsoft는 기타 모든 권리를 보유합니다. 이러한 제한과 관계없이 관련 법률에서 귀하에게 더 많은 권한을 부여하지 않는 한, 이 계약에서 명시적으로 허용된 대로만 소프트웨어를 사용할 수 있습니다. 이 과정에서 귀하는 특정 방식으로만 사용할 수 있도록 하는 소프트웨어의 모든 기술적 제한 사항을 준수해야 합니다. 다음과 같은 행위는 허용되지 않습니다.
   - 소프트웨어의 모든 기술적 제한 사항을 해결하는 행위
   - 소프트웨어를 리버스 엔지니어링, 디컴파일 또는 디스어셈블하거나 그러한 시도를 하는 행위(소프트웨어에 포함될 수 있는 특정 오픈 소스 구성 요소의 사용을 규제하는 제3자 라이선스 약관에서 요구하는 경우 제외)
   - 소프트웨어에서 Microsoft 또는 해당 공급자의 알림을 제거, 최소화, 차단 또는 수정하는 행위
   - 법률에 위배되는 방식으로 소프트웨어를 사용하는 행위 또는
   - 소프트웨어를 공유, 게시, 대여 또는 임대하거나 다른 사람이 사용할 수 있도록 소프트웨어를 독립 실행형으로 호스팅된 솔루션으로 제공하는 행위
5. 수출 제한. 목적지, 최종 사용자 및 최종 사용에 대한 제한 사항을 포함하여 소프트웨어에 적용되는 모든 국내 및 국제 수출 법률 및 규정을 준수해야 합니다. 수출 제한에 대한 자세한 내용은 aka.ms/exporting을 참조하세요.
6. 지원 서비스. 이 소프트웨어는 “있는 그대로” 제공되므로 본 소프트웨어에 대한 지원 서비스가 제공되지 않을 수 있습니다.
7. 완전 합의. 본 계약 및 귀하가 이용하는 추가 구성 요소, 업데이트, 인터넷 기반 서비스 및 지원 서비스에 대한 조항은 소프트웨어 및 지원 서비스에 대한 완전 합의입니다.
8. 관련 법률. 미국에서 본 소프트웨어를 구입한 경우 워싱턴주 법이 본 계약의 위반에 대한 해석 및 클레임에 적용되며, 귀하가 거주하는 지역의 법률이 다른 모든 클레임에 적용됩니다. 다른 국가에서 본 소프트웨어를 구입한 경우 해당 국가의 법률이 적용됩니다.
9. 소비자 권리, 지역 또는 국가별 다양성. 본 계약은 특정 법적 권리에 대해 기술하고 있습니다. 귀하는 거주하는 국가 또는 시/도의 법률에 따라 소비자 권리를 비롯한 기타 권리를 보유할 수 있습니다. Microsoft와 맺은 관계와 별도로, 귀하는 소프트웨어를 판매한 당사자와 관련해서 권리를 보유할 수도 있습니다. 귀하가 거주하고 있는 국가 또는 시/도의 법에서 권리 변경을 허용하지 않는 경우 본 계

약은 이러한 기타 권리를 변경하지 않습니다. 예를 들어 아래 지역 중 하나에서 소프트웨어를 구입했거나 필수 국가 법률이 적용되는 경우 다음 조항이 적용됩니다.

a. 오스트레일리아. 귀하는 오스트레일리아 소비자 보호법에 따라 법적 보장을 받으며 본 계약서의 어떠한 내용도 이러한 권리에 영향을 미쳐서는 안 됩니다.

b. 캐나다. 캐나다에서 본 소프트웨어를 구입한 경우 자동 업데이트 기능을 끄거나, 인터넷에서 디바이스 연결을 끊거나(단, 인터넷에 다시 연결하면 소프트웨어가 업데이트 확인 및 설치를 다시 시작함), 소프트웨어를 제거하여 업데이트 수신을 중지할 수 있습니다. 제품 설명서(있는 경우)에서 특정 디바이스 또는 소프트웨어의 업데이트를 끄는 방법을 지정할 수도 있습니다.

c. 독일 및 오스트리아.

i. 보증. 적절하게 사용이 허가된 소프트웨어는 대체로 본 소프트웨어와 함께 제공되는 Microsoft 자료에 설명된 대로 작동합니다. 그러나 Microsoft는 사용이 허가된 소프트웨어와 관련하여 어떠한 계약 보장도 제공하지 않습니다.

ii. 책임의 제한. 의도적인 행위, 중과실, 제조물 책임법에 따른 청구 및 개인 또는 신체 부상이나 사망의 경우 Microsoft는 성문법에 따라 책임을 집니다. Microsoft는 앞 조항 (ii)에 따라 이와 같은 중요한 계약 의무를 위반하는 경우에만 약간의 과실에 대해 책임집니다. 해당 의무를 다하면 본 계약의 적절한 이행을 촉진하고, 위반하면 본 계약의 목적이 위태롭게 되며, 준수하면 당사자가 “가장 중요한 의무”라는 의무를 지속해서 신뢰할 수 있습니다. 이 외의 경우에는 약간의 과실에 대해 책임지지 않습니다.

10보증의 부인. 이 소프트웨어는 “있는 그대로” 라이선스가 허가됩니다. 해당 사용으로 인해 발생하는 모든 위험은 귀하의 책임입니다. Microsoft는 어떠한 명시적 보증, 보장 또는 조건도 제시하지 않습니다. 귀하가 거주하는 지역의 법규가 허용하는 범위 내에서 MICROSOFT는 상업성, 특정 목적에의 적합성 및 비침해성과 관련된 묵시적 보증을 배제합니다.

11.손해의 제한 및 배제. MICROSOFT와 공급업체에서 최대 $5.00의 직접적인 손해에 대해서만 보상받을 수 있습니다. 결과적 손해, 이익 손실, 특별, 간접 또는 부수적 손해를 포함한 기타 모든 손해에 대해서는 보상을 받을 수 없습니다. 이러한 제한은 (a) 소프트웨어, 서비스, 타사 인터넷 사이트의 콘텐츠(코드 포함) 또는 타사 애플리케이션 (b) 계약, 보증, 보장 또는 조건의 불이행, 무과실 책임, 과실 또는 불법 행위로 인한 청구나 관련 법률에서 허용하는 범위 내의 기타 사례로 청구에 적용됩니다.

또한 본 제한 사항은 Microsoft가 손해 가능성을 알고 있었거나 알았어야 하는 경우에도 적용됩니다. 귀하가 거주하고 있는 국가나 지역에서 부수적, 파생적 또는 기타 손해의 배제나 제한을 허용하지 않는 경우에는 위의 제한이나 배제가 적용되지 않을 수 있습니다.

EULA ID: VS2015_Update3_ShellsRedist_<ENU>

5) Windows 기반 도커 이미지 - nuget.commandline --버전 4.5.1

(라이선스 약관 참조: https://github.com/NuGet/Home/blob/dev/LICENSE.txt)

Copyright (c) .NET Foundation. All rights reserved.

Apache 라이선스, 버전 2.0(이하 “라이선스”)에 따라 라이선스가 부여됩니다. 이 라이선스를 준수하는 경우를 제외하고는 이러한 파일을 사용할 수 없습니다. 라이선스 사본은 다음 사이트에서 구할 수 있습니다.

http://www.apache.org/licenses/LICENSE-2.0

관련 법률에서 요구하거나 서면으로 합의한 경우를 제외하고, 라이선스에 따라 배포된 소프트웨어는 명시적이든 묵시적이든 어떠한 종류의 보증이나 조건 없이 “있는 그대로” 배포됩니다. 라이선스에 따른 구체적인 언어 관리 권한과 제한 사항은 라이선스를 참조하십시오.

7) Windows 기반 도커 이미지 - netfx-4.6.2-devpack

MICROSOFT 소프트웨어 추가 프로그램 라이선스 약관

MICROSOFT WINDOWS 운영 체제용 .NET FRAMEWORK 및 관련 언어 팩

Microsoft Corporation(또는 사는 지역을 기반으로 한 Microsoft 계열사 중 하나)에서 이 추가 프로그램의 사용을 허가합니다. Microsoft Windows 운영 체제 소프트웨어(이하 “소프트웨어”)를 사용할 수 있는 라이선스를 받은 경우 이 추가 프로그램을 사용할 수 있습니다. 소프트웨어에 대한 라이선스가 없는 경우 이 소프트웨어를 사용할 수 없습니다. 유효한 라이선스가 부여된 각 소프트웨어 사본과 함께 이 추가 프로그램을 사용할 수 있습니다.

다음 라이선스 약관은 이 추가 프로그램의 추가 사용 조건을 기술합니다. 본 조건 및 소프트웨어 라이선스 약관은 추가 프로그램의 사용에 적용됩니다. 상충되는 부분이 있는 경우 이러한 추가 프로그램의 라이선스 약관이 적용됩니다.

이 추가 프로그램을 사용하면 이 약관에 동의하는 것으로 간주됩니다. 동의하지 않는 경우 이 추가 프로그램을 사용하지 마세요.

본 라이선스 약관을 준수하는 경우 다음에 대해 권리가 있습니다.

1. 배포 가능 코드. 이 추가 프로그램은 배포 가능 코드로 구성되어 있습니다. “배포 가능 코드”는 아래 조건을 준수하는 경우 개발 중인 프로그램에 배포할 수 있는 코드입니다.

a. 사용 및 배포 권한.

- 이 추가 프로그램의 개체 코드 양식을 복사하여 배포할 수 있습니다.
- 제3자 배포. 프로그램 배포자가 해당 프로그램의 일부로 배포 가능 코드를 복사하고 배포하도록 허용할 수 있습니다.

b. 배포 요구 사항. 배포하는 모든 배포 가능 코드에 대해 다음을 수행해야 합니다.

- 프로그램에 중요한 기본 기능을 추가합니다.
- 파일 확장명이 .lib인 배포 가능한 코드의 경우, 프로그램과 함께 링커를 통해 해당 배포 가능한 코드를 실행한 결과만 배포합니다.
- 설치 프로그램에 포함된 배포 가능 코드를 수정 없이 설치 프로그램의 일부로만 배포합니다.
- 배포자와 외부 최종 사용자에게 최소한 본 계약만큼 해당 코드를 보호하는 조항에 동의하도록 요구합니다.
- 프로그램에 유효한 저작권 고지를 표시합니다. 그리고
- 프로그램의 배포나 사용과 관련하여 변호사 비용을 포함한 모든 청구에 대해 Microsoft를 면책하고 해를 입히지 않으며 방어해야 합니다.

c. 배포 제한 사항. 다음과 같은 행위는 허용되지 않습니다.

- 배포 가능 코드의 저작권, 상표 또는 특허 고지를 변경하는 행위
- 귀하의 애플리케이션이 Microsoft에서 제공했거나 승인했다고 제안하는 방식으로 Microsoft의 상표를 귀하의 애플리케이션 이름에 사용하는 행위
- Windows 플랫폼 이외의 플랫폼에서 실행되도록 배포 가능 코드를 배포하는 행위
- 악성, 기만성 또는 불법 프로그램에 배포 가능 코드를 포함시키는 행위 또는
- 배포 가능 코드의 일부분에 예외적 라이선스가 적용되도록 배포 가능 코드의 소스 코드를 수정하거나 배포하는 행위 예외적 라이선스는 사용, 수정 또는 배포의 조건으로서,
  - 코드가 소스 코드 형식으로 공개 또는 배포되어야 하거나
  - 다른 사람에게 해당 소프트웨어를 수정할 수 있는 권리가 있어야 하는 라이선스입니다.

2. 추가 프로그램 지원 서비스. Microsoft는 www.support.microsoft.com/common/international.aspx에 설명된 대로 이 소프트웨어에 대한 지원 서비스를 제공합니다.

8) Windows 기반 도커 이미지 - visualfsharptools, v 4.0

(라이선스 약관 참조: https://github.com/dotnet/fsharp/blob/main/License.txt)

Copyright (c) Microsoft Corporation. All rights reserved.

Apache 라이선스, 버전 2.0(이하 “라이선스”)에 따라 라이선스가 부여됩니다. 이 라이선스를 준수하는 경우를 제외하고는 이러한 파일을 사용할 수 없습니다. 라이선스 사본은 다음 사이트에서 구할 수 있습니다.

http://www.apache.org/licenses/LICENSE-2.0

관련 법률에서 요구하거나 서면으로 합의한 경우를 제외하고, 라이선스에 따라 배포된 소프트웨어는 명시적이든 묵시적이든 어떠한 종류의 보증이나 조건 없이 “있는 그대로” 배포됩니다. 라이선스에 따른 구체적인 언어 관리 권한과 제한 사항은 라이선스를 참조하십시오.

9) Windows 기반 도커 이미지 - netfx-pcl-reference-assemblies-4.6

MICROSOFT 소프트웨어 라이선스 약관

MICROSOFT .NET 이식 가능한 클래스 라이브러리 참조 어셈블리 - 4.6

본 라이선스 약관은 Microsoft Corporation(또는 사는 지역을 기반으로 한 Microsoft 계열사 중 하나)과 귀하 사이의 계약입니다. 읽어보세요. 이 조건은 위에 명시된 소프트웨어에 적용됩니다. 또한 다른 조건이 있는 경우를 제외하고는 이 소프트웨어에 대한 Microsoft

- 업데이트,
- 추가 프로그램,
- 인터넷 기반 서비스 및
- 지원 서비스

등에도 적용됩니다. 다른 조건이 있는 경우 해당 조건이 적용됩니다.

소프트웨어를 사용하면 본 약관에 동의하는 것으로 간주됩니다. 이 조건에 동의하지 않는 경우에는 소프트웨어를 사용하지 마세요.

본 라이선스 약관을 준수하는 경우 아래와 같은 영구적 권한을 갖습니다.

1. 설치 및 사용 권한. 프로그램을 설계, 개발 및 테스트하기 위해 이 소프트웨어의 사본을 원하는 수만큼 설치하고 사용할 수 있습니다.
2. 추가 라이선스 요구 사항 및/또는 사용 권한.

a. 배포 가능 코드. 아래 조건을 준수하는 경우 개발하는 개발자 도구 프로그램에 소프트웨어를 배포하여 프로그램 고객이 모든 디바이스 또는 운영 체제에서 사용할 수 있는 이식 가능한 라이브러리를 개발하도록 할 수 있습니다.

i. 사용 및 배포 권한. 소프트웨어는 “배포 가능 코드”입니다.

- 배포 가능 코드. 이 소프트웨어의 개체 코드 양식을 복사하여 배포할 수 있습니다.
- 제3자 배포. 프로그램 배포자가 해당 프로그램의 일부로 배포 가능 코드를 복사하고 배포하도록 허용할 수 있습니다.

ii. 배포 요구 사항. 배포하는 모든 배포 가능 코드에 대해 다음을 수행해야 합니다.

- 프로그램에 중요한 기본 기능을 추가합니다.
- 배포자와 고객에게 최소한 본 계약만큼 해당 코드를 보호하는 조항에 동의하도록 요구합니다.
- 프로그램에 유효한 저작권 고지를 표시합니다. 그리고
- 프로그램의 배포나 사용과 관련하여 변호사 비용을 포함한 모든 청구에 대해 Microsoft를 면책하고 해를 입히지 않으며 방어해야 합니다.

iii. 배포 제한 사항. 다음과 같은 행위는 허용되지 않습니다.

- 배포 가능 코드의 저작권, 상표 또는 특허 고지를 변경하는 행위
- 귀하의 애플리케이션이 Microsoft에서 제공했거나 승인했다고 제안하는 방식으로 Microsoft의 상표를 귀하의 애플리케이션 이름에 사용하는 행위
- 악성, 기만성 또는 불법 프로그램에 배포 가능 코드를 포함시키는 행위 또는
- 배포 가능 코드의 일부분에 예외적 라이선스가 적용되도록 배포 가능 코드를 수정하거나 배포하는 행위 예외적 라이선스는 사용, 수정 또는 배포의 조건으로서,
  - 코드가 소스 코드 형식으로 공개 또는 배포되어야 하거나
  - 다른 사람에게 해당 소프트웨어를 수정할 수 있는 권리가 있어야 하는 라이선스입니다.

3. 라이선스 범위. 본 소프트웨어는 판매되는 것이 아니라 그 사용이 허여되는 것입니다. 본 계약은 소프트웨어를 사용할 수 있는 일부 권리만 제공합니다. Microsoft는 기타 모든 권리를 보유합니다. 이러한 제한과 관계없이 관련 법률에서 귀하에게 더 많은 권한을 부여하지 않는 한, 이 계약에서 명시적으로 허용된 대로만 소프트웨어를 사용할 수 있습니다. 이 과정에서 귀하는 특정 방식으로만 사용할 수 있도록 하는 소프트웨어의 모든 기술적 제한 사항을 준수해야 합니다. 다음과 같은 행위는 허용되지 않습니다.

- 소프트웨어의 모든 기술적 제한 사항을 해결하는 행위
- 이러한 제한에도 불구하고 관련 법률에서 명시적으로 허용하는 경우를 제외한 소프트웨어의 리버스 엔지니어링, 디컴파일 또는 디스어셈블 작업을 수행하는 행위

- 다른 사람이 복사할 수 있도록 소프트웨어를 게시하는 행위
- 소프트웨어를 임대, 대여 또는 대부하는 행위

4. 피드백. 소프트웨어에 대한 피드백을 제공할 수 있습니다. Microsoft에 소프트웨어에 대한 피드백을 제공하는 경우 귀하는 피드백을 어떠한 방식과 목적으로든 체험으로 사용, 공유 및 상품화할 수 있는 권리를 Microsoft에 제공하는 것입니다. 또한 피드백이 포함된 Microsoft 소프트웨어 또는 서비스의 일부를 사용하거나 상호 작용하기 위해 제품, 기술 및 서비스에 필요한 특허권을 제3자에게 무료로 제공합니다. Microsoft가 귀하의 피드백을 포함함으로써 제3자에게 소프트웨어 또는 문서의 사용을 허가해야 하는 라이선스가 적용되는 피드백을 제공하지 않습니다. 이러한 권리는 이 계약에서 효력을 유지합니다.
5. 제3자에게 양도. 소프트웨어의 최초 사용자는 소프트웨어 및 본 계약을 제3자에게 직접 양도할 수 있습니다. 양도하기 전에 해당 당사자는 본 계약이 소프트웨어의 전송 및 사용에 적용된다는 데 동의해야 합니다. 최초 사용자는 소프트웨어를 디바이스와 별도로 양도하기 전에 먼저 소프트웨어를 제거해야 합니다. 최초 사용자에게 소프트웨어 사본이 없을 수도 있습니다.
6. 수출 제한. 소프트웨어는 미국 수출 법률 및 규정의 적용을 받습니다. 귀하는 소프트웨어에 적용되는 모든 국내 및 국제 수출법과 규정을 준수해야 합니다. 이러한 법규에는 목적지, 최종 사용자 및 최종 용도에 대한 제한이 포함됩니다. 자세한 내용은 www.microsoft.com/exporting을 참조하세요.
7. 지원 서비스. 이 소프트웨어는 “있는 그대로” 제공되므로 본 소프트웨어에 대한 지원 서비스가 제공되지 않을 수 있습니다.
8. 완전 합의. 본 계약 및 귀하가 이용하는 추가 구성 요소, 업데이트, 인터넷 기반 서비스 및 Microsoft에서 제공하는 지원 서비스에 대한 조항은 소프트웨어 및 지원 서비스에 대한 완전 합의입니다.
9. 관련 법률.
   a. 미국. 소프트웨어를 미국에서 구입한 경우, 국제사법 원칙과 관계없이 본 계약의 해석은 워싱턴 주법을 따르며 계약 위반에 대한 청구 발생 시에도 워싱턴 주법이 적용됩니다. 소비자 보호법, 불공정거래법 및 기타 불법 행위 관련 법규의 적용을 받는 청구가 발생한 경우 사용자가 거주하고 있는 주의 주법이 적용됩니다.
   b. 미국 외 지역. 다른 국가에서 소프트웨어를 구입한 경우 해당 국가의 법률이 적용됩니다.
10. 법적 효력. 본 계약은 특정 법적 권리에 대해 기술하고 있습니다. 귀하는 귀하가 거주하고 있는 국가의 법규가 보장하는 다른 권리를 보유할 수 있습니다. 또한 귀하가 소프트웨어를 구입한 당사자와 관련된 권리를 보유할 수도 있습니다. 귀하가 거주하고 있는 국가의 법에서 권리 변경을 허용하지 않는 경우 본 계약은 해당 권리를 변경하지 않습니다.
11. 보증의 부인. 이 소프트웨어는 “있는 그대로” 라이선스가 허가됩니다. 해당 사용으로 인해 발생하는 모든 위험은 귀하의 책임입니다. Microsoft는 어떠한 명시적 보증, 보장 또는 조건도 제시하지 않습니다. 고객은 현지 법에 따라 본 라이선스로 변경될 수 없는 추가적인 소비자 권리를 보유할 수 있습

니다. 귀하가 거주하는 지역의 법규가 허용하는 범위 내에서 MICROSOFT는 상업성, 특정 목적에의 적합성 및 비침해성과 관련된 묵시적 보증을 배제합니다.

오스트레일리아의 경우 - 귀하는 오스트레일리아 소비자 보호법에 따라 법적 보장을 받으며 본 계약서의 어떠한 내용도 이러한 권리에 영향을 미쳐서는 안 됩니다.

12.손해 및 보상의 제한 및 배제. MICROSOFT와 공급업체에서 최대 $5.00의 직접적인 손해에 대해서만 보상받을 수 있습니다. 결과적 손해, 이익 손실, 특별, 간접 또는 부수적 손해를 포함한 기타 모든 손해에 대해서는 보상을 받을 수 없습니다.

이러한 제한은

- 소프트웨어, 서비스, 타사 인터넷 사이트의 콘텐츠(코드 포함) 또는 타사 애플리케이션
- 계약, 보증, 보장 또는 조건의 불이행, 무과실 책임, 과실 또는 불법 행위로 인한 청구나 관련 법률에서 허용하는 범위 내의 기타 사례에 대한 청구에 적용됩니다.

또한 본 제한 사항은 Microsoft가 손해 가능성을 알고 있었거나 알았어야 하는 경우에도 적용됩니다. 귀하가 거주하고 있는 국가나 지역에서 부수적, 파생적 또는 기타 손해의 배제나 제한을 허용하지 않는 경우에는 위의 제한이나 배제가 적용되지 않을 수 있습니다.

10) Windows 기반 도커 이미지 - visualcppbuildtools v 14.0.25420.1

(라이선스 약관 참조: https://www.visualstudio.com/license-terms/mt644918/)

MICROSOFT VISUAL C++ 빌드 도구

MICROSOFT 소프트웨어 라이선스 약관

MICROSOFT VISUAL C++ 빌드 도구

본 라이선스 약관은 Microsoft Corporation(또는 사는 지역을 기반으로 한 Microsoft 계열사 중 하나)과 사용자 사이의 계약입니다. 이 조건은 위에 명시된 소프트웨어에 적용됩니다. 본 조건은 다른 조건이 있는 경우를 제외하고 소프트웨어의 모든 Microsoft 서비스 또는 업데이트에도 적용됩니다.

본 라이선스 약관을 준수하는 경우 다음에 대해 권리가 있습니다.

1. 설치 및 사용 권한.

a. 한 명의 사용자가 소프트웨어 사본을 사용하여 애플리케이션을 개발하고 테스트할 수 있습니다.

2. 데이터. 소프트웨어에서 사용자 및 사용자의 소프트웨어 사용에 대한 정보를 수집하여 Microsoft에 보낼 수 있습니다. Microsoft는 이 정보를 사용하여 서비스를 제공하고 제품 및 서비스를 개선할 수 있습니다. 제품 설명서에 설명된 대로 이러한 시나리오를 대부분 옵트아웃할 수 있지만 전부는 아닙니다. 또한 소프트웨어에는 애플리케이션 사용자로부터 데이터를 수집할 수 있는 몇 가지 기능이 있습니다. 이러한 기능을 사용하여 애플리케이션에서 데이터를 수집할 수 있게 하는 경우 애플리케이션 사용자에게 적절한 고지를 제공하는 것을 포함하여 관련 법률을 준수해야 합니다. 데이터 수집 및 사용에 대한 자세한 내용은 도움말 설명서 및 http://go.microsoft.com/fwlink/?LinkID=528096의 개인 정보 보호 정책에서 확인할 수 있습니다. 소프트웨어를 사용하면 이러한 정책에 동의하는 것으로 간주됩니다.

3. 특정 구성 요소에 대한 조건.

   a. 빌드 서버. 소프트웨어에는 BuildServer.TXT 파일에 나열된 일부 빌드 서버 구성 요소 및/또는 이 Microsoft 소프트웨어 라이선스 약관에 따라 BuildServer 목록에 나열된 모든 파일이 포함될 수 있습니다. 소프트웨어에 포함된 경우 해당 항목을 빌드 컴퓨터에 복사하여 설치할 수 있습니다. 사용자 및 조직의 다른 사용자는 애플리케이션을 컴파일, 빌드, 확인 및 보관하거나 빌드 프로세스의 일부로 품질 또는 성능 테스트를 실행할 목적으로만 빌드 시스템에서 이러한 항목을 사용할 수 있습니다.

   b. Microsoft 플랫폼. 소프트웨어에는 Microsoft Windows, Microsoft Windows Server, Microsoft SQL Server, Microsoft Exchange, Microsoft Office 및 Microsoft SharePoint의 구성 요소가 포함될 수 있습니다. 이러한 구성 요소에는 해당 구성 요소의 설치 디렉터리 또는 소프트웨어와 함께 제공되는 “Licenses” 폴더에 있는 라이선스 약관에 설명된 대로 별도의 계약 및 자체 제품 지원 정책이 적용됩니다.

   c. 타사 구성 요소. 소프트웨어에는 소프트웨어와 함께 제공되는 ThirdPartyNotices 파일에 설명된 대로 별도의 법적 고지가 있거나 다른 계약의 적용을 받는 타사 구성 요소가 포함될 수 있습니다. 이러한 구성 요소가 다른 계약의 적용을 받는 경우에도 아래의 고지 사항, 손해 배상 제한 및 제외 조항이 함께 적용됩니다.

   d. 패키지 관리자. 소프트웨어에는 애플리케이션과 함께 사용할 다른 Microsoft 및 타사 소프트웨어 패키지를 다운로드할 수 있는 옵션을 제공하는 Nuget과 같은 패키지 관리자가 포함될 수 있습니다. 이러한 패키지에는 자체 라이선스가 적용되며 본 계약이 적용되지 않습니다. Microsoft는 타사 패키지를 배포하거나 라이선스를 부여하거나 보증을 제공하지 않습니다.

4. 라이선스 범위. 본 소프트웨어는 판매되는 것이 아니라 그 사용이 허여되는 것입니다. 본 계약은 소프트웨어를 사용할 수 있는 일부 권리만 제공합니다. Microsoft는 기타 모든 권리를 보유합니다. 이러한 제한과 관계없이 관련 법률에서 귀하에게 더 많은 권한을 부여하지 않는 한, 이 계약에서 명시적으로 허용된 대로만 소프트웨어를 사용할 수 있습니다. 이 과정에서 귀하는 특정 방식으로만 사용할 수 있도록 하는 소프트웨어의 모든 기술적 제한 사항을 준수해야 합니다. 자세한 내용은 https://

docs.microsoft.com/en-us/legal/information-protection/software-license-terms#1-installation-and-use-rights를 참조하세요. 다음과 같은 행위는 허용되지 않습니다.

- 소프트웨어의 모든 기술적 제한 사항을 해결하는 행위
- 소프트웨어를 리버스 엔지니어링, 디컴파일 또는 디스어셈블하거나 그러한 시도를 하는 행위(소프트웨어에 포함될 수 있는 특정 오픈 소스 구성 요소의 사용을 규제하는 제3자 라이선스 약관에서 요구하는 경우 제외)
- Microsoft 또는 해당 공급자의 알림을 제거, 최소화, 차단 또는 수정하는 행위
- 법률에 위배되는 방식으로 소프트웨어를 사용하는 행위 또는
- 소프트웨어를 공유, 게시, 대여 또는 임대하거나 다른 사람이 사용할 수 있도록 소프트웨어를 독립 실행형으로 호스팅된 솔루션으로 제공하는 행위

5. 수출 제한. 목적지, 최종 사용자 및 최종 사용에 대한 제한 사항을 포함하여 소프트웨어에 적용되는 모든 국내 및 국제 수출 법률 및 규정을 준수해야 합니다. 수출 제한에 대한 자세한 내용은 aka.ms/exporting을 참조하세요.
6. 지원 서비스. 이 소프트웨어는 “있는 그대로” 제공되므로 본 소프트웨어에 대한 지원 서비스가 제공되지 않을 수 있습니다.
7. 완전 합의. 본 계약 및 귀하가 이용하는 추가 구성 요소, 업데이트, 인터넷 기반 서비스 및 지원 서비스에 대한 조항은 소프트웨어 및 지원 서비스에 대한 완전 합의입니다.
8. 관련 법률. 미국에서 본 소프트웨어를 구입한 경우 워싱턴주 법이 본 계약의 위반에 대한 해석 및 클레임에 적용되며, 귀하가 거주하는 지역의 법률이 다른 모든 클레임에 적용됩니다. 다른 국가에서 본 소프트웨어를 구입한 경우 해당 국가의 법률이 적용됩니다.
9. 소비자 권리, 지역 또는 국가별 다양성. 본 계약은 특정 법적 권리에 대해 기술하고 있습니다. 귀하는 거주하는 국가 또는 시/도의 법률에 따라 소비자 권리를 비롯한 기타 권리를 보유할 수 있습니다. Microsoft와 맺은 관계와 별도로, 귀하는 소프트웨어를 판매한 당사자와 관련해서 권리를 보유할 수도 있습니다. 귀하가 거주하고 있는 국가 또는 시/도의 법에서 권리 변경을 허용하지 않는 경우 본 계약은 이러한 기타 권리를 변경하지 않습니다. 예를 들어 아래 지역 중 하나에서 소프트웨어를 구입했거나 필수 국가 법률이 적용되는 경우 다음 조항이 적용됩니다.
   - 오스트레일리아. 귀하는 오스트레일리아 소비자 보호법에 따라 법적 보장을 받으며 본 계약서의 어떠한 내용도 이러한 권리에 영향을 미쳐서는 안 됩니다.
   - 캐나다. 캐나다에서 본 소프트웨어를 구입한 경우 자동 업데이트 기능을 끄거나, 인터넷에서 디바이스 연결을 끊거나(단, 인터넷에 다시 연결하면 소프트웨어가 업데이트 확인 및 설치를 다시 시작함), 소프트웨어를 제거하여 업데이트 수신을 중지할 수 있습니다. 제품 설명서(있는 경우)에서 특정 디바이스 또는 소프트웨어의 업데이트를 끄는 방법을 지정할 수도 있습니다.
   - 독일 및 오스트리아.

- 보증. 적절하게 사용이 허가된 소프트웨어는 대체로 본 소프트웨어와 함께 제공되는 Microsoft 자료에 설명된 대로 작동합니다. 그러나 Microsoft는 사용이 허가된 소프트웨어와 관련하여 어떠한 계약 보장도 제공하지 않습니다.
- 책임의 제한. 의도적인 행위, 심각한 부주의, 제조물 책임법에 따른 클레임 및 개인 또는 신체 부상이나 사망의 경우 Microsoft는 성문법에 따라 책임을 집니다.

  앞 조항 (ii)에 따라 Microsoft는 Microsoft가 이러한 중요한 계약 의무를 위반하거나 본 계약의 적절한 수행을 가능하게 하는 사항을 이행하지 않거나 이행을 위반하거나 본 계약의 목적 및 당사자가 지속적으로 신뢰할 수 있도록 하는 규정 준수 사항을 위반하는 경우에만 약간의 부주의에 대해 책임집니다. 이 외의 경우에는 약간의 과실에 대해 책임지지 않습니다.

10.법적 효력. 본 계약은 특정 법적 권리에 대해 기술하고 있습니다. 귀하는 귀하가 거주하고 있는 지역 또는 국가의 법규가 보장하는 다른 권리를 보유할 수 있습니다. 귀하가 거주하고 있는 지역 또는 국가의 법에서 권리 변경을 허용하지 않는 경우 본 계약은 해당 권리를 변경하지 않습니다. 이전 조항에 대한 제한 없이, 오스트레일리아의 경우 귀하는 오스트레일리아 소비자 보호법에 따라 법적 보장을 받으며 본 계약서의 어떠한 내용도 이러한 권리에 영향을 미쳐서는 안 됩니다.

11.보증의 부인. 이 소프트웨어는 “있는 그대로” 라이선스가 허가됩니다. 해당 사용으로 인해 발생하는 모든 위험은 귀하의 책임입니다. Microsoft는 어떠한 명시적 보증, 보장 또는 조건도 제시하지 않습니다. 귀하가 거주하는 지역의 법규가 허용하는 범위 내에서 MICROSOFT는 상업성, 특정 목적에의 적합성 및 비침해성과 관련된 묵시적 보증을 배제합니다.

12.손해의 제한 및 배제. MICROSOFT와 공급업체에서 최대 $5.00의 직접적인 손해에 대해서만 보상받을 수 있습니다. 결과적 손해, 이익 손실, 특별, 간접 또는 부수적 손해를 포함한 기타 모든 손해에 대해서는 보상을 받을 수 없습니다.

  이러한 제한은 (a) 소프트웨어, 서비스, 타사 인터넷 사이트의 콘텐츠(코드 포함) 또는 타사 애플리케이션 (b) 계약, 보증, 보장 또는 조건의 불이행, 무과실 책임, 과실 또는 불법 행위로 인한 청구나 관련 법률에서 허용하는 범위 내의 기타 사례로 청구에 적용됩니다.

  또한 본 제한 사항은 Microsoft가 손해 가능성을 알고 있었거나 알았어야 하는 경우에도 적용됩니다. 귀하가 거주하고 있는 국가나 지역에서 부수적, 파생적 또는 기타 손해의 배제나 제한을 허용하지 않는 경우에는 위의 제한이나 배제가 적용되지 않을 수 있습니다.

11) Windows 기반 도커 이미지 - microsoft-windows-netfx3-ondemand-package.cab

MICROSOFT 소프트웨어 추가 프로그램 라이선스 약관

MICROSOFT WINDOWS 운영 체제용 MICROSOFT .NET FRAMEWORK 3.5 SP1

Microsoft Corporation(또는 사는 지역을 기반으로 한 Microsoft 계열사 중 하나)에서 이 추가 프로그램의 사용을 허가합니다. Microsoft Windows 운영 체제 소프트웨어(본 추가 프로그램이 적용되는)(이하 “소프트웨어”)를 사용할 수 있는 라이선스를 받은 경우 이 추가 프로그램을 사용할 수 있습니다. 소프트웨어에 대한 라이선스가 없는 경우 이 소프트웨어를 사용할 수 없습니다. 유효한 라이선스가 부여된 각 소프트웨어 사본과 함께 이 추가 프로그램 사본을 사용할 수 있습니다.

다음 라이선스 약관은 이 추가 프로그램의 추가 사용 조건을 기술합니다. 본 조건 및 소프트웨어 라이선스 약관은 추가 프로그램의 사용에 적용됩니다. 상충되는 부분이 있는 경우 이러한 추가 프로그램의 라이선스 약관이 적용됩니다.

이 추가 프로그램을 사용하면 이 약관에 동의하는 것으로 간주됩니다. 동의하지 않는 경우 이 추가 프로그램을 사용하지 마세요.

본 라이선스 약관을 준수하는 경우 다음에 대해 권리가 있습니다.

1. 추가 프로그램 지원 서비스. Microsoft는 www.support.microsoft.com/common/international.aspx에 설명된 대로 이 소프트웨어에 대한 지원 서비스를 제공합니다.
2. MICROSOFT .NET 벤치마크 테스트. 소프트웨어에는 Windows 운영 체제의 .NET Framework, Windows Communication Foundation, Windows Presentation Foundation 및 Windows Workflow Foundation 구성 요소(.NET 구성 요소)가 포함됩니다. .NET 구성 요소의 내부 벤치마크 테스트를 수행할 수 있습니다. http://go.microsoft.com/fwlink/?LinkID=66406에 명시된 조건을 준수하는 경우 .NET 구성 요소의 모든 벤치마크 테스트 결과를 공개할 수 있습니다.

   Microsoft와 체결한 다른 계약에도 불구하고 그러한 벤치마크 테스트 결과를 공개하면 Microsoft는 http://go.microsoft.com/fwlink/?LinkID=66406에 명시된 것과 동일한 조건을 준수하는 경우 해당 .NET 구성 요소와 경쟁하는 제품에 대해 수행한 벤치마크 테스트 결과를 공개할 권리가 있습니다.

12) Windows 기반 도커 이미지 - dotnet-sdk

(https://github.com/dotnet/core/blob/main/LICENSE.TXT에서 사용 가능)

MIT 라이선스(MIT)

Copyright (c) Microsoft Corporation

이 소프트웨어 및 관련 문서 파일(“소프트웨어”) 사본을 획득하는 사람에게 다음 조건에 따라 소프트웨어의 사본을 사용, 복사, 수정, 병합, 게시, 배포, 재사용 허가 및/또는 판매하고, 소프트웨어를 제공 받은 사람이 이렇게 할 수 있도록 허용하기 위한 권한을 제한 없이 포함하여 소프트웨어를 자유롭게 거래할 수 있는 권한이 무료로 허여됩니다.

위의 저작권 알림 및 이 권한은 소프트웨어의 모든 사본 또는 중요한 부분에 포함되어야 합니다.

이 소프트웨어는 상품성, 특정한 목적을 위한 적합성 및 비침해에 관한 보증을 포함하되 이에 제한하지 않고 명시적 또는 묵시적인 어떠한 보증도 없이 “그대로” 제공됩니다. 어떤 경우에도 작성자 또는 저작권 소유자는 계약상 행위, 불법 또는 그 밖의 문제로 인한 것인지와 관계없이, 본 소프트웨어나 소프트웨어의 사용 또는 기타 거래로 인해 발생하는 손해 배상, 손해 또는 기타 문제에 대한 책임을 지지 않습니다.

CodeBuild 조건 키를 IAM 서비스 역할 변수로 사용하여 빌드 액세스 제어

CodeBuild 빌드 ARN을 사용하면 컨텍스트 키로 CodeBuild 서비스 역할의 리소스 액세스 범위를 축소하여 빌드 리소스 액세스를 제한할 수 있습니다. CodeBuild의 경우 빌드 액세스 동작을 제어하는 데 사용할 수 있는 키는 `codebuild:buildArn` 및 `codebuild:projectArn`입니다. 빌드 프로젝트 ARN을 사용하면 리소스에 대한 직접 호출이 특정 빌드 프로젝트에서 비롯되었는지 확인할 수 있습니다. 이를 확인하려면 IAM ID 기반 정책에서 `codebuild:buildArn` 또는 `codebuild:projectArn` 조건 키를 사용합니다.

정책에서 `codebuild:buildArn` 또는 `codebuild:projectArn` 조건 키를 사용하려면 이를 ARN 조건 연산자와 함께 조건으로 포함해야 합니다. 키 값은 유효한 ARN으로 확인되는 IAM 변수여야 합니다. 아래 예제 정책에서는 `${codebuild:projectArn}` IAM 변수에 대한 프로젝트 ARN을 가진 빌드 프로젝트에만 액세스가 허용됩니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "s3:PutObject",
```

```
            "Resource": "arn:aws:s3:::bucket-name/${codebuild:projectArn}/*"
        }
    ]
}
```

AWS CodeBuild 조건 키

AWS CodeBuild 는 IAM 정책에서 프로젝트 및 플릿과 같은 CodeBuild 리소스에 조직 정책을 적용하는 데 사용할 수 있는 조건 키 세트를 제공합니다. 조건 키는 네트워크 설정, 자격 증명 구성 및 컴퓨팅 제한을 포함하여 대부분의 CodeBuild API 요청 컨텍스트를 다룹니다.

주제

- 프로젝트 및 플릿에 VPC 연결 설정 적용
- 프로젝트 buildspec에 대한 무단 수정 방지
- 빌드의 컴퓨팅 유형 제한
- 제어 환경 변수 설정
- 조건 키 이름에 변수 사용
- API 요청에 속성이 있는지 확인

프로젝트 및 플릿에 VPC 연결 설정 적용

이 정책은 CodeBuild 프로젝트 및 플릿을 생성할 때 호출자가 선택한 VPC, 서브넷 및 보안 그룹을 사용할 수 있도록 허용합니다. 다중 값 컨텍스트 키에 대한 자세한 내용은 단일 값 컨텍스트 키와 다중 값 컨텍스트 키 비교를 참조하세요.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "codebuild:CreateProject",
            "codebuild:CreateFleet"
        ],
        "Resource": "*",
        "Condition": {
```

```
            "ForAnyValue:StringEquals": {
                "codebuild:vpcConfig.vpcId": [
                    "vpc-01234567890abcdef",
                    "vpc-abcdef01234567890"
                ],
                "codebuild:vpcConfig.subnets": [
                    "subnet-1234abcd",
                    "subnet-5678abcd"
                ],
                "codebuild:vpcConfig.securityGroupIds": [
                    "sg-12345678abcdefghij",
                    "sg-01234567abcdefghij"
                ]
            }
        }
    }]
}
```

프로젝트 buildspec에 대한 무단 수정 방지

이 정책은 호출자가 `buildspecOverride` 필드의 buildspec을 재정의하도록 허용하지 않습니다.

> **Note**
>
> `codebuild:source.buildspec` 조건 키는 Null 연산자만 지원하여 API 필드의 존재 여부를 확인합니다. buildspec의 내용은 평가하지 않습니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": "codebuild:StartBuild",
        "Resource": "*"
    }, {
        "Effect": "Deny",
        "Action": "codebuild:StartBuild",
        "Resource": "*",
```

```
        "Condition": {
            "Null": {
                "codebuild:source.buildspec": "false"
            }
        }
    }]
}
```

빌드의 컴퓨팅 유형 제한

이 정책으로 `c5.large` 또는 `m5.large` 컴퓨팅 인스턴스 유형으로만 빌드할 수 있는 플릿을 생성할 수 있습니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": "codebuild:CreateFleet",
        "Resource": "*",
        "Condition": {
            "ForAnyValue:StringEquals": {
                "codebuild:computeConfiguration.instanceType": ["c5.large",
 "m5.large"]
            }
        }
    }]
}
```

제어 환경 변수 설정

이 정책은 호출자가 `STAGE` 환경 변수를 `BETA` 또는 `GAMMA`로 재정의하도록 허용합니다. 또한, `STAGE`를 `PRODUCTION`으로 재정의하는 것을 명시적으로 거부하고 `MY_APP_VERSION` 환경 변수를 재정의하는 것을 거부합니다. 다중 값 컨텍스트 키에 대해서는 단일 값 컨텍스트 키와 다중 값 컨텍스트 키 비교를 참조하세요.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:StartBuild"
            ],
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "codebuild:environment.environmentVariables/STAGE.value": [
                        "BETA",
                        "GAMMA"
                    ]
                }
            }
        },
        {
            "Effect": "Deny",
            "Action": [
                "codebuild:StartBuild"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "codebuild:environment.environmentVariables/STAGE.value":
 "PRODUCTION"
                },
                "ForAnyValue:StringEquals": {
                    "codebuild:environment.environmentVariables.name": [
                        "MY_APP_VERSION"
                    ]
                }
            }
        }
    ]
}
```

조건 키 이름에 변수 사용

secondarySources/${sourceIdentifier}.location 및 secondaryArtifacts/${artifactIdentifier}.location과 같은 조건 키 이름에 변수를 사용할 수 있습니다. IAM 정책에서 보조 소스 또는 보조 아티팩트 식별자를 지정할 수 있습니다. 아래 정책은 호출자가 보조 소스 mySecondSource의 특정 소스 위치가 있는 프로젝트를 생성하도록 허용합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codebuild:CreateProject",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "codebuild:secondarySources/mySecondSource.location": "my-
source-location"
                }
            }
        }
    ]
}
```

API 요청에 속성이 있는지 확인

CodeBuild는 조건 키를 지원하여 API 요청에 일부 필드가 있는지 확인합니다. 이 정책은 프로젝트를 생성하거나 업데이트할 때 VPC 요구 사항을 적용합니다.

```
{
    "Version": "2012-10-17"             ,
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "codebuild:CreateProject",
            "codebuild:UpdateProject"
        ],
```

```
        "Resource": "*",
        "Condition": {
            "Null": {
                "codebuild:vpcConfig.vpcId": "false"
            }
        }
    }]
}
```

AWS SDKs를 사용한 CodeBuild 코드 예제

다음 코드 예제에서는 CodeBuild를 AWS 소프트웨어 개발 키트(SDK)와 함께 사용하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

코드 예시

- AWS SDKs를 사용한 CodeBuild의 기본 예제
  - AWS SDKs를 사용한 CodeBuild 작업
    - AWS SDK 또는 CLI와 CreateProject 함께 사용
    - AWS SDK 또는 CLI와 ListBuilds 함께 사용
    - AWS SDK 또는 CLI와 ListProjects 함께 사용
    - AWS SDK 또는 CLI와 StartBuild 함께 사용

AWS SDKs를 사용한 CodeBuild의 기본 예제

다음 코드 예제에서는 AWS CodeBuild SDKs에서의 기본 사항을 AWS 사용하는 방법을 보여줍니다.

예제

- AWS SDKs를 사용한 CodeBuild 작업
  - AWS SDK 또는 CLI와 CreateProject 함께 사용
  - AWS SDK 또는 CLI와 ListBuilds 함께 사용
  - AWS SDK 또는 CLI와 ListProjects 함께 사용
  - AWS SDK 또는 CLI와 StartBuild 함께 사용

AWS SDKs를 사용한 CodeBuild 작업

다음 코드 예제에서는 AWS SDKs를 사용하여 개별 CodeBuild 작업을 수행하는 방법을 보여줍니다. 각 예시에는 GitHub에 대한 링크가 포함되어 있습니다. 여기에서 코드 설정 및 실행에 대한 지침을 찾을 수 있습니다.

다음 예제에는 가장 일반적으로 사용되는 작업만 포함되어 있습니다. 전체 목록은 AWS CodeBuild API 참조를 참조하세요.

예제

- AWS SDK 또는 CLI와 CreateProject 함께 사용
- AWS SDK 또는 CLI와 ListBuilds 함께 사용
- AWS SDK 또는 CLI와 ListProjects 함께 사용
- AWS SDK 또는 CLI와 StartBuild 함께 사용

AWS SDK 또는 CLI와 **CreateProject** 함께 사용

다음 코드 예시는 CreateProject의 사용 방법을 보여 줍니다.

CLI

AWS CLI

예제 1: AWS CodeBuild 빌드 프로젝트 생성

다음 create-project 예시에서는 S3 버킷의 소스 파일을 사용하여 CodeBuild 빌드 프로젝트를 생성합니다.

```
aws codebuild create-project \
    --name "my-demo-project" \
    --source "{\"type\": \"S3\",\"location\": \"codebuild-us-west-2-123456789012-
input-bucket/my-source.zip\"}" \
    --artifacts {"\"type\": \"S3\",\"location\": \"codebuild-us-
west-2-123456789012-output-bucket\""} \
    --environment "{\"type\": \"LINUX_CONTAINER\",\"image\": \"aws/codebuild/
standard:1.0\",\"computeType\": \"BUILD_GENERAL1_SMALL\"}" \
    --service-role "arn:aws:iam::123456789012:role/service-role/my-codebuild-
service-role"
```

출력:

```
{
    "project": {
        "arn": "arn:aws:codebuild:us-west-2:123456789012:project/my-demo-
project",
        "name": "my-cli-demo-project",
```

```
        "encryptionKey": "arn:aws:kms:us-west-2:123456789012:alias/aws/s3",
        "serviceRole": "arn:aws:iam::123456789012:role/service-role/my-codebuild-
service-role",
        "lastModified": 1556839783.274,
        "badge": {
            "badgeEnabled": false
        },
        "queuedTimeoutInMinutes": 480,
        "environment": {
            "image": "aws/codebuild/standard:1.0",
            "computeType": "BUILD_GENERAL1_SMALL",
            "type": "LINUX_CONTAINER",
            "imagePullCredentialsType": "CODEBUILD",
            "privilegedMode": false,
            "environmentVariables": []
        },
        "artifacts": {
            "location": "codebuild-us-west-2-123456789012-output-bucket",
            "name": "my-cli-demo-project",
            "namespaceType": "NONE",
            "type": "S3",
            "packaging": "NONE",
            "encryptionDisabled": false
        },
        "source": {
            "type": "S3",
            "location": "codebuild-us-west-2-123456789012-input-bucket/my-
source.zip",
            "insecureSsl": false
        },
        "timeoutInMinutes": 60,
        "cache": {
            "type": "NO_CACHE"
        },
        "created": 1556839783.274
    }
}
```

예제 2: 파라미터에 대한 JSON 입력 파일을 사용하여 AWS CodeBuild 빌드 프로젝트 생성

다음 `create-project` 예시에서는 필요한 파라미터를 모두 JSON 입력 파일에 전달하여 CodeBuild 빌드 프로젝트를 생성합니다. `--generate-cli-skeleton parameter`만 포함하여 명령을 실행하여 입력 파일 템플릿을 생성합니다.

```
aws codebuild create-project --cli-input-json file://create-project.json
```

입력 JSON 파일 `create-project.json`에는 다음 콘텐츠가 포함되어 있습니다.

```
{
    "name": "codebuild-demo-project",
    "source": {
        "type": "S3",
        "location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"
    },
    "artifacts": {
        "type": "S3",
        "location": "codebuild-region-ID-account-ID-output-bucket"
    },
    "environment": {
        "type": "LINUX_CONTAINER",
        "image": "aws/codebuild/standard:1.0",
        "computeType": "BUILD_GENERAL1_SMALL"
    },
    "serviceRole": "serviceIAMRole"
}
```

출력:

```
{
    "project": {
        "name": "codebuild-demo-project",
        "serviceRole": "serviceIAMRole",
        "tags": [],
        "artifacts": {
            "packaging": "NONE",
            "type": "S3",
            "location": "codebuild-region-ID-account-ID-output-bucket",
            "name": "message-util.zip"
        },
        "lastModified": 1472661575.244,
        "timeoutInMinutes": 60,
        "created": 1472661575.244,
        "environment": {
            "computeType": "BUILD_GENERAL1_SMALL",
            "image": "aws/codebuild/standard:1.0",
            "type": "LINUX_CONTAINER",
```

```
            "environmentVariables": []
        },
        "source": {
            "type": "S3",
            "location": "codebuild-region-ID-account-ID-input-bucket/
MessageUtil.zip"
        },
        "encryptionKey": "arn:aws:kms:region-ID:account-ID:alias/aws/s3",
        "arn": "arn:aws:codebuild:region-ID:account-ID:project/codebuild-demo-
project"
    }
}
```

자세한 내용은 AWS CodeBuild 사용 설명서의 빌드 프로젝트 생성(AWS CLI)을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 CreateProject를 참조하세요.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

프로젝트 생성

```
import {
  ArtifactsType,
  CodeBuildClient,
  ComputeType,
  CreateProjectCommand,
  EnvironmentType,
  SourceType,
} from "@aws-sdk/client-codebuild";

// Create the AWS CodeBuild project.
export const createProject = async (
  projectName = "MyCodeBuilder",
  roleArn = "arn:aws:iam::xxxxxxxxxxxx:role/CodeBuildAdmin",
```

```
  buildOutputBucket = "xxxx",
  githubUrl = "https://...",
) => {
  const codeBuildClient = new CodeBuildClient({});

  const response = await codeBuildClient.send(
    new CreateProjectCommand({
      artifacts: {
        // The destination of the build artifacts.
        type: ArtifactsType.S3,
        location: buildOutputBucket,
      },
      // Information about the build environment. The combination of
 "computeType" and "type" determines the
      // requirements for the environment such as CPU, memory, and disk space.
      environment: {
        // Build environment compute types.
        // https://docs.aws.amazon.com/codebuild/latest/userguide/build-env-ref-
compute-types.html
        computeType: ComputeType.BUILD_GENERAL1_SMALL,
        // Docker image identifier.
        // See https://docs.aws.amazon.com/codebuild/latest/userguide/build-env-
ref-available.html
        image: "aws/codebuild/standard:7.0",
        // Build environment type.
        type: EnvironmentType.LINUX_CONTAINER,
      },
      name: projectName,
      // A role ARN with permission to create a CodeBuild project, write to the
 artifact location, and write CloudWatch logs.
      serviceRole: roleArn,
      source: {
        // The type of repository that contains the source code to be built.
        type: SourceType.GITHUB,
        // The location of the repository that contains the source code to be
 built.
        location: githubUrl,
      },
    }),
  );
  console.log(response);
  //   {
  //     '$metadata': {
  //       httpStatusCode: 200,
```

```
  //       requestId: 'b428b244-777b-49a6-a48d-5dffedced8e7',
  //       extendedRequestId: undefined,
  //       cfId: undefined,
  //       attempts: 1,
  //       totalRetryDelay: 0
  //     },
  //     project: {
  //       arn: 'arn:aws:codebuild:us-east-1:xxxxxxxxxxxx:project/MyCodeBuilder',
  //       artifacts: {
  //         encryptionDisabled: false,
  //         location: 'xxxxxx-xxxxxxx-xxxxxx',
  //         name: 'MyCodeBuilder',
  //         namespaceType: 'NONE',
  //         packaging: 'NONE',
  //         type: 'S3'
  //       },
  //       badge: { badgeEnabled: false },
  //       cache: { type: 'NO_CACHE' },
  //       created: 2023-08-18T14:46:48.979Z,
  //       encryptionKey: 'arn:aws:kms:us-east-1:xxxxxxxxxxxx:alias/aws/s3',
  //       environment: {
  //         computeType: 'BUILD_GENERAL1_SMALL',
  //         environmentVariables: [],
  //         image: 'aws/codebuild/standard:7.0',
  //         imagePullCredentialsType: 'CODEBUILD',
  //         privilegedMode: false,
  //         type: 'LINUX_CONTAINER'
  //       },
  //       lastModified: 2023-08-18T14:46:48.979Z,
  //       name: 'MyCodeBuilder',
  //       projectVisibility: 'PRIVATE',
  //       queuedTimeoutInMinutes: 480,
  //       serviceRole: 'arn:aws:iam::xxxxxxxxxxxx:role/CodeBuildAdmin',
  //       source: {
  //         insecureSsl: false,
  //         location: 'https://...',
  //         reportBuildStatus: false,
  //         type: 'GITHUB'
  //       },
  //       timeoutInMinutes: 60
  //     }
  //   }
  return response;
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하세요.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 CreateProject를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **ListBuilds** 함께 사용

다음 코드 예시는 ListBuilds의 사용 방법을 보여 줍니다.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
//! List the CodeBuild builds.
/*!
  \param sortType: 'SortOrderType' type.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::CodeBuild::listBuilds(Aws::CodeBuild::Model::SortOrderType sortType,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);

    Aws::CodeBuild::Model::ListBuildsRequest listBuildsRequest;
    listBuildsRequest.SetSortOrder(sortType);

    Aws::String nextToken; // Used for pagination.

    do {
        if (!nextToken.empty()) {
            listBuildsRequest.SetNextToken(nextToken);
```

```
        }

        Aws::CodeBuild::Model::ListBuildsOutcome listBuildsOutcome =
codeBuildClient.ListBuilds(
                listBuildsRequest);

        if (listBuildsOutcome.IsSuccess()) {
            const Aws::Vector<Aws::String> &ids =
listBuildsOutcome.GetResult().GetIds();
            if (!ids.empty()) {

                std::cout << "Information about each build:" << std::endl;
                Aws::CodeBuild::Model::BatchGetBuildsRequest getBuildsRequest;
                getBuildsRequest.SetIds(listBuildsOutcome.GetResult().GetIds());
                Aws::CodeBuild::Model::BatchGetBuildsOutcome getBuildsOutcome =
codeBuildClient.BatchGetBuilds(
                        getBuildsRequest);

                if (getBuildsOutcome.IsSuccess()) {
                    const Aws::Vector<Aws::CodeBuild::Model::Build> &builds =
getBuildsOutcome.GetResult().GetBuilds();
                    std::cout << builds.size() << " build(s) found." <<
std::endl;
                    for (auto val: builds) {
                        std::cout << val.GetId() << std::endl;
                    }
                } else {
                    std::cerr << "Error getting builds"
                              << getBuildsOutcome.GetError().GetMessage() <<
std::endl;
                    return false;
                }
            } else {
                std::cout << "No builds found." << std::endl;
            }

            // Get the next token for pagination.

            nextToken = listBuildsOutcome.GetResult().GetNextToken();
        } else {
            std::cerr << "Error listing builds"
                      << listBuildsOutcome.GetError().GetMessage()
                      << std::endl;
            return false;
```

```
        }

    } while (!nextToken.

            empty()

            );

    return true;
}
```

- API에 대한 세부 정보는 AWS SDK for C++ API 참조의 ListBuilds를 참조하세요.

CLI

AWS CLI

AWS CodeBuild 빌드 IDs 목록을 가져오는 방법.

다음 `list-builds` 예제에서는 오름차순으로 정렬된 CodeBuild ID 목록을 가져옵니다.

```
aws codebuild list-builds --sort-order ASCENDING
```

출력에는 사용 가능한 출력이 더 있음을 나타내는 `nextToken` 값이 포함됩니다.

```
{
    "nextToken": "4AEA6u7J...The full token has been omitted for
 brevity...MzY2OA==",
    "ids": [
        "codebuild-demo-project:815e755f-bade-4a7e-80f0-efe51EXAMPLE"
        "codebuild-demo-project:84a7f3d1-d40e-4956-b4cf-7a9d4EXAMPLE"
            ... The full list of build IDs has been omitted for brevity ...
        "codebuild-demo-project:931d0b72-bf6f-4040-a472-5c707EXAMPLE"
    ]
}
```

이 명령을 다시 실행하고 이전 응답의 `nextToken` 값을 파라미터로 제공하여 출력의 다음 부분을 가져옵니다. 응답에서 `nextToken` 값을 받지 못할 때까지 반복합니다.

```
aws codebuild list-builds --sort-order ASCENDING --next-
token 4AEA6u7J...The full token has been omitted for brevity...MzY2OA==
```

출력의 다음 부분:

```
{
    "ids": [
        "codebuild-demo-project:49015049-21cf-4b50-9708-df115EXAMPLE",
        "codebuild-demo-project:543e7206-68a3-46d6-a4da-759abEXAMPLE",
            ... The full list of build IDs has been omitted for brevity ...
        "codebuild-demo-project:c282f198-4582-4b38-bdc0-26f96EXAMPLE"
    ]
}
```

자세한 내용은 AWS CodeBuild 사용 설명서의 빌드 IDs(AWS CLI)를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListBuilds 섹션을 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 `ListProjects` 함께 사용

다음 코드 예시는 `ListProjects`의 사용 방법을 보여 줍니다.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
//! List the CodeBuild projects.
/*!
  \param sortType: 'SortOrderType' type.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
```

```
bool AwsDoc::CodeBuild::listProjects(Aws::CodeBuild::Model::SortOrderType
 sortType,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);

    Aws::CodeBuild::Model::ListProjectsRequest listProjectsRequest;
    listProjectsRequest.SetSortOrder(sortType);

    Aws::String nextToken; // Next token for pagination.
    Aws::Vector<Aws::String> allProjects;

    do {
        if (!nextToken.empty()) {
            listProjectsRequest.SetNextToken(nextToken);
        }

        Aws::CodeBuild::Model::ListProjectsOutcome outcome =
 codeBuildClient.ListProjects(
                listProjectsRequest);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::String> &projects =
 outcome.GetResult().GetProjects();
            allProjects.insert(allProjects.end(), projects.begin(),
 projects.end());
            nextToken = outcome.GetResult().GetNextToken();
        }

        else {
            std::cerr << "Error listing projects" <<
 outcome.GetError().GetMessage()
                      << std::endl;
        }

    } while (!nextToken.empty());

    std::cout << allProjects.size() << " project(s) found." << std::endl;
    for (auto project: allProjects) {
        std::cout << project << std::endl;
    }

    return true;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 ListProjects를 참조하세요.

CLI

AWS CLI

AWS CodeBuild 빌드 프로젝트 이름 목록을 가져옵니다.

다음 `list-projects` 예제에서는 이름을 기준으로 정렬된 CodeBuild 빌드 프로젝트 목록을 오름차순으로 가져옵니다.

```
aws codebuild list-projects --sort-by NAME --sort-order ASCENDING
```

출력에는 사용 가능한 출력이 더 있음을 나타내는 nextToken 값이 포함됩니다.

```
{
    "nextToken": "Ci33ACF6...The full token has been omitted for brevity...U
+AkMx8=",
    "projects": [
        "codebuild-demo-project",
        "codebuild-demo-project2",
            ... The full list of build project names has been omitted for
 brevity ...
        "codebuild-demo-project99"
    ]
}
```

이 명령을 다시 실행하고 이전 응답의 nextToken 값을 파라미터로 제공하여 출력의 다음 부분을 가져옵니다. 응답에서 nextToken 값을 받지 못할 때까지 반복합니다.

```
aws codebuild list-projects  --sort-by NAME --sort-order ASCENDING --next-
token Ci33ACF6...The full token has been omitted for brevity...U+AkMx8=

{
    "projects": [
        "codebuild-demo-project100",
        "codebuild-demo-project101",

  ... The full list of build project names has been omitted for brevity ...
        "codebuild-demo-project122"
```

```
  ]
}
```

자세한 내용은 AWS CodeBuild 사용 설명서의 빌드 프로젝트 이름 목록(AWS CLI) 보기를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListProjects를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **StartBuild** 함께 사용

다음 코드 예시는 StartBuild의 사용 방법을 보여 줍니다.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
//! Start an AWS CodeBuild project build.
/*!
  \param projectName: A CodeBuild project name.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::CodeBuild::startBuild(const Aws::String &projectName,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);

    Aws::CodeBuild::Model::StartBuildRequest startBuildRequest;
    startBuildRequest.SetProjectName(projectName);

    Aws::CodeBuild::Model::StartBuildOutcome outcome =
 codeBuildClient.StartBuild(
```

```
            startBuildRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully started build" << std::endl;
        std::cout << "Build ID: " << outcome.GetResult().GetBuild().GetId()
                  << std::endl;
    }

    else {
        std::cerr << "Error starting build" << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 StartBuild를 참조하세요.

CLI

AWS CLI

AWS CodeBuild 빌드 프로젝트의 빌드 실행을 시작합니다.

다음 `start-build` 예제에서는 지정된 CodeBuild 프로젝트의 빌드를 시작합니다. 빌드는 시간이 초과되기 전에 빌드가 대기열에 대기할 수 있는 분 수와 프로젝트의 아티팩트 설정 모두에 대해 프로젝트의 설정을 재정의합니다.

```
aws codebuild start-build \
    --project-name "my-demo-project" \
    --queued-timeout-in-minutes-override 5 \
    --artifacts-override {"\"type\": \"S3\",\"location\":
 \"arn:aws:s3:::artifacts-override\",\"overrideArtifactName\":true"}
```

출력:

```
{
    "build": {
        "serviceRole": "arn:aws:iam::123456789012:role/service-role/my-codebuild-
service-role",
        "buildStatus": "IN_PROGRESS",
```

```
        "buildComplete": false,
        "projectName": "my-demo-project",
        "timeoutInMinutes": 60,
        "source": {
            "insecureSsl": false,
            "type": "S3",
            "location": "codebuild-us-west-2-123456789012-input-bucket/my-
source.zip"
        },
        "queuedTimeoutInMinutes": 5,
        "encryptionKey": "arn:aws:kms:us-west-2:123456789012:alias/aws/s3",
        "currentPhase": "QUEUED",
        "startTime": 1556905683.568,
        "environment": {
            "computeType": "BUILD_GENERAL1_MEDIUM",
            "environmentVariables": [],
            "type": "LINUX_CONTAINER",
            "privilegedMode": false,
            "image": "aws/codebuild/standard:1.0",
            "imagePullCredentialsType": "CODEBUILD"
        },
        "phases": [
            {
                "phaseStatus": "SUCCEEDED",
                "startTime": 1556905683.568,
                "phaseType": "SUBMITTED",
                "durationInSeconds": 0,
                "endTime": 1556905684.524
            },
            {
                "startTime": 1556905684.524,
                "phaseType": "QUEUED"
            }
        ],
        "logs": {
            "deepLink": "https://console.aws.amazon.com/cloudwatch/home?
region=us-west-2#logEvent:group=null;stream=null"
        },
        "artifacts": {
            "encryptionDisabled": false,
            "location": "arn:aws:s3:::artifacts-override/my-demo-project",
            "overrideArtifactName": true
        },
        "cache": {
```

```
            "type": "NO_CACHE"
        },
        "id": "my-demo-project::12345678-a1b2-c3d4-e5f6-11111EXAMPLE",
        "initiator": "my-aws-account-name",
        "arn": "arn:aws:codebuild:us-west-2:123456789012:build/my-demo-
project::12345678-a1b2-c3d4-e5f6-11111EXAMPLE"
    }
}
```

자세한 내용은 AWS CodeBuild 사용 설명서의 빌드 실행(AWS CLI)을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 StartBuild 섹션을 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

퍼블릭 빌드 API

퍼블릭 빌드 API는 AWS CodeBuild 콘솔 및 퍼블릭 빌드 결과 웹 사이트에서 퍼블릭 빌드 프로젝트에 대한 정보를 얻는 데 사용됩니다. 자세한 내용은 the section called “퍼블릭 빌드 프로젝트 URL 가져오기” 단원을 참조하십시오.

> **Note**
>
> CodeBuild 퍼블릭 빌드 API는 AWS CLI AWS SDKs에 포함되지 않습니다.

주제

- 퍼블릭 빌드 작업
- 퍼블릭 빌드 데이터 형식

퍼블릭 빌드 작업

> **Note**
>
> CodeBuild 퍼블릭 빌드 API는 AWS CLI AWS SDKs에 포함되지 않습니다.

CodeBuild 퍼블릭 빌드 API에서 지원하는 작업은 다음과 같습니다.

작업

- DescribeBuildBatchesForPublicProject
- DescribeBuildsForPublicProject
- GetCloudWatchLogsForPublicBuild
- GetPresignedUrlsForPublicBuild
- GetPublicBuild
- GetPublicBuildBatch
- GetPublicProject

DescribeBuildBatchesForPublicProject

> **Note**
>
> 이 API 요소는 AWS CLI AWS SDKs에 포함되지 않습니다.

구문 요청

```
{
   "filter": {
      "status": "string"
   },
   "maxResults": number,
   "nextToken": "string",
   "publicProjectAlias": "string",
   "sortOrder": "string"
}
```

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 자세한 설명은 공통 파라미터를 참조하세요.

요청은 JSON 형식으로 다음 데이터를 받습니다.

> **Note**
>
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

publicProjectAlias

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이는 1024입니다.

패턴: `^[0-9a-zA-Z%+=]+$`

필수 여부: 예

필터

배치 빌드를 검색할 때 필터를 지정합니다.

유형: BuildBatchFilter 객체

필수 여부: 아니요

maxResults

타입: 정수

유효 범위: 최소값 1. 최댓값은 100입니다.

필수 여부: 아니요

nextToken

유형: 문자열

필수 항목 여부: 아니요

sortOrder

유형: 문자열

유효한 값: `ASCENDING | DESCENDING`

필수 항목 여부: 아니요

응답 구문

```
{
   "nextToken": "string",
   "publicBuildBatches": [
      {
         "buildBatchNumber": number,
         "buildBatchStatus": "string",
         "endTime": number,
         "publicBuildBatchAlias": "string",
         "sourceVersion": "string",
         "startTime": number
      }
   ]
```

```
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

nextToken

유형: 문자열

publicBuildBatches

타입: BuildBatchForDescribeBuildBatchesPublic 객체 배열

오류

모든 작업에 공통적인 오류에 대한 자세한 내용은 공통 오류를 참조하세요.

InvalidInputException

HTTP 상태 코드: 400

ResourceNotFoundException

HTTP 상태 코드: 400

DescribeBuildsForPublicProject

> **Note**
>
> 이 API 요소는 AWS CLI AWS SDKs에 포함되지 않습니다.

구문 요청

```
{
   "maxResults": number,
   "nextToken": "string",
   "publicProjectAlias": "string",
   "sortOrder": "string"
}
```

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 자세한 설명은 공통 파라미터를 참조하세요.

요청은 JSON 형식으로 다음 데이터를 받습니다.

> **Note**
>
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

publicProjectAlias

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이는 1024입니다.

패턴: `^[0-9a-zA-Z%+=]+$`

필수 여부: 예

maxResults

타입: 정수

유효 범위: 최소값 1. 최댓값은 100입니다.

필수 여부: 아니요

nextToken

유형: 문자열

필수 항목 여부: 아니요

sortOrder

유형: 문자열

유효한 값: ASCENDING | DESCENDING

필수 항목 여부: 아니요

응답 구문

```
{
   "builds": [
      {
         "artifacts": {
            "artifactIdentifier": "string",
            "packaging": "string",
            "type": "string"
         },
         "buildComplete": boolean,
         "buildNumber": number,
         "buildStatus": "string",
         "endTime": number,
         "environment": {
            "computeType": "string",
            "environmentVariables": [
               {
                  "name": "string",
                  "type": "string",
                  "value": "string"
               }
            ],
            "image": "string",
            "type": "string"
         },
         "id": "string",
         "initiator": "string",
```

```
      "logsStatus": {
         "cloudWatchLogsStatus": "string",
         "s3LogsStatus": "string"
      },
      "phases": [
         {
            "contexts": [
               {
                  "message": "string",
                  "statusCode": "string"
               }
            ],
            "durationInSeconds": number,
            "endTime": number,
            "phaseStatus": "string",
            "phaseType": "string",
            "startTime": number
         }
      ],
      "projectName": "string",
      "queuedTimeoutInMinutes": number,
      "resolvedSourceVersion": "string",
      "secondaryArtifacts": [
         {
            "artifactIdentifier": "string",
            "packaging": "string",
            "type": "string"
         }
      ],
      "secondarySources": [
         {
            "buildspec": "string",
            "gitCloneDepth": number,
            "gitSubmodulesConfig": {
               "fetchSubmodules": boolean
            },
            "location": "string",
            "sourceIdentifier": "string",
            "type": "string"
         }
      ],
      "secondarySourceVersions": [
         {
            "sourceIdentifier": "string",
```

```
            "sourceVersion": "string"
         }
      ],
      "source": {
         "buildspec": "string",
         "gitCloneDepth": number,
         "gitSubmodulesConfig": {
            "fetchSubmodules": boolean
         },
         "location": "string",
         "sourceIdentifier": "string",
         "type": "string"
      },
      "sourceVersion": "string",
      "startTime": number,
      "timeoutInMinutes": number
   }
],
"nextToken": "string"
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

빌드

타입: PublicBuild객체 배열

nextToken

유형: 문자열

오류

모든 작업에 공통적인 오류에 대한 자세한 내용은 공통 오류를 참조하세요.

InvalidInputException

HTTP 상태 코드: 400

ResourceNotFoundException

HTTP 상태 코드: 400

GetCloudWatchLogsForPublicBuild

> **Note**
>
> 이 API 요소는 AWS CLI AWS SDKs에 포함되지 않습니다.

구문 요청

```
{
   "maxResult": number,
   "nextToken": "string",
   "publicBuildAlias": "string"
}
```

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 자세한 설명은 공통 파라미터를 참조하세요.

요청은 JSON 형식으로 다음 데이터를 받습니다.

> **Note**
>
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

publicBuildAlias

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이는 1024입니다.

패턴: `^[0-9a-zA-Z%+=]+:\p{XDigit}{8}(-\p{XDigit}{4}){3}-\p{XDigit}{12}$`

필수 여부: 예

maxResult

타입: 정수

유효한 범위: 최소값 1. 최대값은 10,000입니다.

필수 여부: 아니요

nextToken

유형: 문자열

필수사항: 아니요

응답 구문

```
{
   "logs": [ "string" ],
   "nextToken": "string"
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

로그

유형: 문자열 배열

nextToken

유형: 문자열

길이 제약: 최소 길이 1.

오류

모든 작업에 공통적인 오류에 대한 자세한 내용은 공통 오류를 참조하세요.

InvalidInputException

HTTP 상태 코드: 400

ResourceNotFoundException

HTTP 상태 코드: 400

GetPresignedUrlsForPublicBuild

> **Note**
>
> 이 API 요소는 AWS CLI AWS SDKs에 포함되지 않습니다.

구문 요청

```
{
   "publicBuildAlias": "string"
}
```

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 자세한 설명은 공통 파라미터를 참조하세요.

요청은 JSON 형식으로 다음 데이터를 받습니다.

> **Note**
>
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

publicBuildAlias

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이는 1024입니다.

패턴: `^[0-9a-zA-Z%+=]+:\p{XDigit}{8}(-\p{XDigit}{4}){3}-\p{XDigit}{12}$`

필수 여부: 예

응답 구문

```
{
   "artifacts": {
```

```
      "expiredAt": number,
      "identifier": "string",
      "md5Checksum": "string",
      "presignedUrl": "string",
      "s3Arn": "string",
      "sha256Checksum": "string"
   },
   "log": {
      "expiredAt": number,
      "identifier": "string",
      "md5Checksum": "string",
      "presignedUrl": "string",
      "s3Arn": "string",
      "sha256Checksum": "string"
   },
   "secondaryArtifacts": [
      {
         "expiredAt": number,
         "identifier": "string",
         "md5Checksum": "string",
         "presignedUrl": "string",
         "s3Arn": "string",
         "sha256Checksum": "string"
      }
   ]
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

아티팩트

유형: S3Downloadable객체

로그

유형: S3Downloadable객체

secondaryArtifacts

타입: S3Downloadable 객체 배열

오류

모든 작업에 공통적인 오류에 대한 자세한 내용은 공통 오류를 참조하세요.

InvalidInputException

HTTP 상태 코드: 400

ResourceNotFoundException

HTTP 상태 코드: 400

GetPublicBuild

> **Note**
>
> 이 API 요소는 AWS CLI AWS SDKs에 포함되지 않습니다.

구문 요청

```
{
   "publicBuildAlias": "string"
}
```

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 자세한 설명은 공통 파라미터를 참조하세요.

요청은 JSON 형식으로 다음 데이터를 받습니다.

> **Note**
>
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

publicBuildAlias

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이는 1024입니다.

패턴: `^[0-9a-zA-Z%+=]+:\p{XDigit}{8}(-\p{XDigit}{4}){3}-\p{XDigit}{12}$`

필수 여부: 예

응답 구문

```
{
   "build": {
      "artifacts": {
         "artifactIdentifier": "string",
         "packaging": "string",
```

```
      "type": "string"
   },
   "buildComplete": boolean,
   "buildNumber": number,
   "buildStatus": "string",
   "endTime": number,
   "environment": {
      "computeType": "string",
      "environmentVariables": [
         {
            "name": "string",
            "type": "string",
            "value": "string"
         }
      ],
      "image": "string",
      "type": "string"
   },
   "id": "string",
   "initiator": "string",
   "logsStatus": {
      "cloudWatchLogsStatus": "string",
      "s3LogsStatus": "string"
   },
   "phases": [
      {
         "contexts": [
            {
               "message": "string",
               "statusCode": "string"
            }
         ],
         "durationInSeconds": number,
         "endTime": number,
         "phaseStatus": "string",
         "phaseType": "string",
         "startTime": number
      }
   ],
   "projectName": "string",
   "queuedTimeoutInMinutes": number,
   "resolvedSourceVersion": "string",
   "secondaryArtifacts": [
      {
```

```
            "artifactIdentifier": "string",
            "packaging": "string",
            "type": "string"
         }
      ],
      "secondarySources": [
         {
            "buildspec": "string",
            "gitCloneDepth": number,
            "gitSubmodulesConfig": {
               "fetchSubmodules": boolean
            },
            "location": "string",
            "sourceIdentifier": "string",
            "type": "string"
         }
      ],
      "secondarySourceVersions": [
         {
            "sourceIdentifier": "string",
            "sourceVersion": "string"
         }
      ],
      "source": {
         "buildspec": "string",
         "gitCloneDepth": number,
         "gitSubmodulesConfig": {
            "fetchSubmodules": boolean
         },
         "location": "string",
         "sourceIdentifier": "string",
         "type": "string"
      },
      "sourceVersion": "string",
      "startTime": number,
      "timeoutInMinutes": number
   }
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

빌드

타입: PublicBuild 객체

오류

모든 작업에 공통적인 오류에 대한 자세한 내용은 공통 오류를 참조하세요.

InvalidInputException

HTTP 상태 코드: 400

ResourceNotFoundException

HTTP 상태 코드: 400

GetPublicBuildBatch

> **Note**
>
> 이 API 요소는 AWS CLI AWS SDKs에 포함되지 않습니다.

구문 요청

```
{
   "publicBuildBatchAlias": "string"
}
```

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 자세한 설명은 공통 파라미터를 참조하세요.

요청은 JSON 형식으로 다음 데이터를 받습니다.

> **Note**
>
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

publicBuildBatchAlias

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이는 1024입니다.

패턴: `^[0-9a-zA-Z%+=]+:[a-z0-9]{8}(-[a-z0-9]{4}){3}-[a-z0-9]{12}$`

필수 여부: 예

응답 구문

```
{
   "publicBuildBatch": {
      "artifacts": {
         "artifactIdentifier": "string",
         "packaging": "string",
```

```
            "type": "string"
         },
         "buildBatchConfig": {
            "combineArtifacts": boolean,
            "restrictions": {
               "computeTypesAllowed": [ "string" ],
               "maximumBuildsAllowed": number
            },
            "timeoutInMins": number
         },
         "buildBatchNumber": number,
         "buildBatchStatus": "string",
         "buildGroups": [
            {
               "currentBuildSummary": {
                  "buildStatus": "string",
                  "primaryArtifact": {
                     "identifier": "string",
                     "location": "string",
                     "type": "string"
                  },
                  "publicBuildAlias": "string",
                  "requestedOn": number,
                  "secondaryArtifacts": [
                     {
                        "identifier": "string",
                        "location": "string",
                        "type": "string"
                     }
                  ]
               },
               "dependsOn": [ "string" ],
               "identifier": "string",
               "ignoreFailure": boolean,
               "priorBuildSummaryList": [
                  {
                     "buildStatus": "string",
                     "primaryArtifact": {
                        "identifier": "string",
                        "location": "string",
                        "type": "string"
                     },
                     "publicBuildAlias": "string",
                     "requestedOn": number,
```

```
                "secondaryArtifacts": [
                   {
                      "identifier": "string",
                      "location": "string",
                      "type": "string"
                   }
                ]
             }
          ]
       }
    ],
    "buildTimeoutInMinutes": number,
    "complete": boolean,
    "currentPhase": "string",
    "endTime": number,
    "environment": {
       "computeType": "string",
       "environmentVariables": [
          {
             "name": "string",
             "type": "string",
             "value": "string"
          }
       ],
       "image": "string",
       "type": "string"
    },
    "id": "string",
    "initiator": "string",
    "logsStatus": {
       "cloudWatchLogsStatus": "string",
       "s3LogsStatus": "string"
    },
    "phases": [
       {
          "contexts": [
             {
                "message": "string",
                "statusCode": "string"
             }
          ],
          "durationInSeconds": number,
          "endTime": number,
          "phaseStatus": "string",
```

```
            "phaseType": "string",
            "startTime": number
         }
      ],
      "projectName": "string",
      "publicBuildBatchAlias": "string",
      "queuedTimeoutInMinutes": number,
      "resolvedSourceVersion": "string",
      "secondaryArtifacts": [
         {
            "artifactIdentifier": "string",
            "packaging": "string",
            "type": "string"
         }
      ],
      "secondarySources": [
         {
            "buildspec": "string",
            "gitCloneDepth": number,
            "gitSubmodulesConfig": {
               "fetchSubmodules": boolean
            },
            "location": "string",
            "sourceIdentifier": "string",
            "type": "string"
         }
      ],
      "secondarySourceVersions": [
         {
            "sourceIdentifier": "string",
            "sourceVersion": "string"
         }
      ],
      "source": {
         "buildspec": "string",
         "gitCloneDepth": number,
         "gitSubmodulesConfig": {
            "fetchSubmodules": boolean
         },
         "location": "string",
         "sourceIdentifier": "string",
         "type": "string"
      },
      "sourceVersion": "string",
```

```
        "startTime": number
    }
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

publicBuildBatch

타입: PublicBuildBatch 객체

오류

모든 작업에 공통적인 오류에 대한 자세한 내용은 공통 오류를 참조하세요.

InvalidInputException

HTTP 상태 코드: 400

ResourceNotFoundException

HTTP 상태 코드: 400

GetPublicProject

> **Note**
>
> 이 API 요소는 AWS CLI AWS SDKs에 포함되지 않습니다.

구문 요청

```
{
   "publicProjectAlias": "string"
}
```

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 자세한 설명은 공통 파라미터를 참조하세요.

요청은 JSON 형식으로 다음 데이터를 받습니다.

> **Note**
>
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

publicProjectAlias

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이는 1024입니다.

패턴: `^[0-9a-zA-Z%+=]+$`

필수 여부: 예

응답 구문

```
{
   "project": {
      "artifacts": {
         "artifactIdentifier": "string",
         "location": "string",
```

```
      "type": "string"
   },
   "buildBatchConfig": {
      "combineArtifacts": boolean,
      "restrictions": {
         "computeTypesAllowed": [ "string" ],
         "maximumBuildsAllowed": number
      },
      "timeoutInMins": number
   },
   "concurrentBuildLimit": number,
   "description": "string",
   "environment": {
      "computeType": "string",
      "environmentVariables": [
         {
            "name": "string",
            "type": "string",
            "value": "string"
         }
      ],
      "image": "string",
      "type": "string"
   },
   "name": "string",
   "queuedTimeoutInMinutes": number,
   "secondaryArtifacts": [
      {
         "artifactIdentifier": "string",
         "location": "string",
         "type": "string"
      }
   ],
   "secondarySources": [
      {
         "buildspec": "string",
         "gitCloneDepth": number,
         "gitSubmodulesConfig": {
            "fetchSubmodules": boolean
         },
         "location": "string",
         "sourceIdentifier": "string",
         "type": "string"
      }
```

```
      ],
      "secondarySourceVersions": [
         {
            "sourceIdentifier": "string",
            "sourceVersion": "string"
         }
      ],
      "source": {
         "buildspec": "string",
         "gitCloneDepth": number,
         "gitSubmodulesConfig": {
            "fetchSubmodules": boolean
         },
         "location": "string",
         "sourceIdentifier": "string",
         "type": "string"
      },
      "sourceVersion": "string",
      "timeoutInMinutes": number,
      "webhook": {
         "branchFilter": "string",
         "buildType": "string",
         "filterGroups": [
            [
               {
                  "excludeMatchedPattern": boolean,
                  "pattern": "string",
                  "type": "string"
               }
            ]
         ],
         "payloadUrl": "string",
         "url": "string"
      }
   }
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

프로젝트

타입: PublicProject 객체

오류

모든 작업에 공통적인 오류에 대한 자세한 내용은 공통 오류를 참조하세요.

InvalidInputException

HTTP 상태 코드: 400

ResourceNotFoundException

HTTP 상태 코드: 400

퍼블릭 빌드 데이터 형식

> **Note**
> CodeBuild 퍼블릭 빌드 API는 AWS CLI AWS SDKs에 포함되지 않습니다.

CodeBuild 퍼블릭 빌드 API에서 지원하는 데이터 유형은 다음과 같습니다.

데이터 타입

- BuildBatchForDescribeBuildBatchesPublic
- PublicBuild
- PublicBuildArtifacts
- PublicBuildBatch
- PublicBuildGroup
- PublicBuildSummary
- PublicLogsStatus
- PublicProject
- PublicProjectArtifacts
- PublicProjectBuildBatchConfig

- PublicProjectEnvironment
- PublicProjectSource
- PublicWebhook
- S3Downloadable

BuildBatchForDescribeBuildBatchesPublic

> **Note**
>
> 이 API 요소는 AWS CLI AWS SDKs에 포함되지 않습니다.

내용

> **Note**
>
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

buildBatchNumber

유형: Long

필수 여부: 아니요

buildBatchStatus

유형: 문자열

유효한 값: `PENDING | SUCCEEDED | FAILED | FAULT | TIMED_OUT | IN_PROGRESS | STOPPED`

필수 여부: 아니요

endTime

유형: 타임스탬프

필수 여부: 아니요

publicBuildBatchAlias

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이는 1024입니다.

패턴: `^[0-9a-zA-Z%+=]+:[a-z0-9]{8}(-[a-z0-9]{4}){3}-[a-z0-9]{12}$`

필수 여부: 아니요

sourceVersion

유형: 문자열

길이 제약: 최소 길이 1.

필수 여부: 아니요

startTime

유형: 타임스탬프

필수 여부: 아니요

PublicBuild

> **Note**
>
> 이 API 요소는 AWS CLI AWS SDKs에 포함되지 않습니다.

내용

> **Note**
>
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

artifacts

유형: PublicBuildArtifacts객체

필수 여부: 아니요

buildComplete

유형: 부울

필수 항목 여부: 아니요

buildNumber

유형: Long

필수 여부: 아니요

buildStatus

유형: 문자열

유효한 값: `PENDING | SUCCEEDED | FAILED | FAULT | TIMED_OUT | IN_PROGRESS | STOPPED`

필수 여부: 아니요

endTime

유형: 타임스탬프

필수 여부: 아니요

환경

유형: PublicProjectEnvironment객체

필수 여부: 아니요

id

유형: 문자열

길이 제약: 최소 길이 1.

필수 여부: 아니요

이니시에이터

유형: 문자열

필수 항목 여부: 아니요

logsStatus

유형: PublicLogsStatus객체

필수 여부: 아니요

단계

유형: BuildPhase 객체 배열

필수 여부: 아니요

projectName

유형: 문자열

길이 제약: 최소 길이 1.

필수 여부: 아니요

queuedTimeoutInMinutes

유형: 정수

필수 항목 여부: 아니요

resolvedSourceVersion

유형: 문자열

길이 제약: 최소 길이 1.

필수 여부: 아니요

secondaryArtifacts

타입: PublicBuildArtifacts객체 배열

필수: 아니요

secondarySources

타입: PublicProjectSource객체 배열

배열 구성원: 최소수는 0개입니다. 최대 항목 수는 12개입니다.

필수 여부: 아니요

secondarySourceVersions

유형: ProjectSourceVersion 객체 배열

배열 구성원: 최소수는 0개입니다. 최대 항목 수는 12개입니다.

필수 여부: 아니요

source

유형: PublicProjectSource객체

필수 여부: 아니요

sourceVersion

유형: 문자열

길이 제약: 최소 길이 1.

필수 여부: 아니요

startTime

유형: 타임스탬프

필수 여부: 아니요

timeoutInMinutes

유형: 정수

필수 항목 여부: 아니요

PublicBuildArtifacts

> **Note**
>
> 이 API 요소는 AWS CLI AWS SDKs에 포함되지 않습니다.

내용

> **Note**
>
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

artifactIdentifier

유형: 문자열

필수 항목 여부: 아니요

패키징

유형: 문자열

필수 항목 여부: 아니요

type

유형: 문자열

유효한 값: `CODEPIPELINE | S3 | NO_ARTIFACTS | DOCKER_IMAGE`

필수 여부: 아니요

PublicBuildBatch

> **Note**
>
> 이 API 요소는 AWS CLI AWS SDKs에 포함되지 않습니다.

내용

> **Note**
>
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

artifacts

유형: PublicBuildArtifacts객체

필수 여부: 아니요

buildBatchConfig

유형: PublicProjectBuildBatchConfig객체

필수 여부: 아니요

buildBatchNumber

유형: Long

필수 여부: 아니요

buildBatchStatus

유형: 문자열

유효한 값: `PENDING | SUCCEEDED | FAILED | FAULT | TIMED_OUT | IN_PROGRESS | STOPPED`

필수 여부: 아니요

buildGroups

타입: PublicBuildGroup객체 배열

필수: 아니요

buildTimeoutInMinutes

유형: 정수

필수 항목 여부: 아니요

완료

유형: 부울

필수 항목 여부: 아니요

currentPhase

유형: 문자열

필수 항목 여부: 아니요

endTime

유형: 타임스탬프

필수 여부: 아니요

환경

유형: PublicProjectEnvironment객체

필수 여부: 아니요

id

유형: 문자열

길이 제약: 최소 길이 1.

필수 여부: 아니요

이니시에이터

유형: 문자열

필수 항목 여부: 아니요

logsStatus

유형: PublicLogsStatus객체

필수 여부: 아니요

단계

유형: BuildBatchPhase 객체 배열

필수 여부: 아니요

projectName

유형: 문자열

길이 제약: 최소 길이 1.

필수 여부: 아니요

publicBuildBatchAlias

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이는 1024입니다.

패턴: `^[0-9a-zA-Z%+=]+:[a-z0-9]{8}(-[a-z0-9]{4}){3}-[a-z0-9]{12}$`

필수 여부: 아니요

queuedTimeoutInMinutes

유형: 정수

필수 항목 여부: 아니요

resolvedSourceVersion

유형: 문자열

길이 제약: 최소 길이 1.

필수 여부: 아니요

secondaryArtifacts

타입: PublicBuildArtifacts객체 배열

필수: 아니요

secondarySources

타입: PublicProjectSource객체 배열

배열 구성원: 최소수는 0개입니다. 최대 항목 수는 12개입니다.

필수 여부: 아니요

secondarySourceVersions

유형: ProjectSourceVersion 객체 배열

배열 구성원: 최소수는 0개입니다. 최대 항목 수는 12개입니다.

필수 여부: 아니요

source

유형: PublicProjectSource객체

필수 여부: 아니요

sourceVersion

유형: 문자열

길이 제약: 최소 길이 1.

필수 여부: 아니요

startTime

유형: 타임스탬프

필수 여부: 아니요

PublicBuildGroup

> **Note**
>
> 이 API 요소는 AWS CLI AWS SDKs에 포함되지 않습니다.

내용

> **Note**
>
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

currentBuildSummary

유형: PublicBuildSummary객체

필수 여부: 아니요

dependsOn

유형: 문자열 배열

길이 제약: 최소 길이 1.

필수 여부: 아니요

identifier

유형: 문자열

필수 항목 여부: 아니요

ignoreFailure

유형: 부울

필수 항목 여부: 아니요

priorBuildSummaryList

타입: PublicBuildSummary객체 배열

필수: 아니요

PublicBuildSummary

> **Note**
>
> 이 API 요소는 AWS CLI AWS SDKs에 포함되지 않습니다.

내용

> **Note**
>
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

buildStatus

유형: 문자열

필수 항목 여부: 아니요

primaryArtifact

해결된 빌드 아티팩트를 나타냅니다. 해결된 아티팩트는 Amazon S3와 같은 대상에 빌드 및 배포되는 아티팩트입니다.

유형: ResolvedArtifact 객체

필수 여부: 아니요

publicBuildAlias

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이는 1024입니다.

패턴: `^[0-9a-zA-Z%+=]+:\p{XDigit}{8}(-\p{XDigit}{4}){3}-\p{XDigit}{12}$`

필수 여부: 아니요

requestedOn

유형: 타임스탬프

필수 여부: 아니요

secondaryArtifacts

유형: ResolvedArtifact 객체 배열

필수 여부: 아니요

PublicLogsStatus

> **Note**
>
> 이 API 요소는 AWS CLI AWS SDKs에 포함되지 않습니다.

내용

> **Note**
>
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

cloudWatchLogsStatus

유형: 문자열

유효한 값: `ENABLED | DISABLED`

필수 여부: 아니요

s3LogsStatus

유형: 문자열

유효한 값: `ENABLED | DISABLED`

필수 여부: 아니요

PublicProject

> **Note**
>
> 이 API 요소는 AWS CLI AWS SDKs에 포함되지 않습니다.

내용

> **Note**
>
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

artifacts

유형: PublicProjectArtifacts객체

필수 여부: 아니요

buildBatchConfig

유형: PublicProjectBuildBatchConfig객체

필수 여부: 아니요

concurrentBuildLimit

유형: 정수

필수 항목 여부: 아니요

description

유형: 문자열

길이 제약 조건: 최소 길이는 0입니다. 최대 길이는 255입니다.

필수 여부: 아니요

환경

유형: PublicProjectEnvironment객체

필수 여부: 아니요

이름

유형: 문자열

길이 제한: 최소 길이는 2. 최대 길이 150.

패턴: `[A-Za-z0-9][A-Za-z0-9\-_]{1,254}`

필수 여부: 아니요

queuedTimeoutInMinutes

유형: 정수

유효한 범위: 최소값은 5. 최대값은 480입니다.

필수 여부: 아니요

secondaryArtifacts

타입: PublicProjectArtifacts객체 배열

배열 구성원: 최소수는 0개입니다. 최대 항목 수는 12개입니다.

필수 여부: 아니요

secondarySources

타입: PublicProjectSource객체 배열

배열 구성원: 최소수는 0개입니다. 최대 항목 수는 12개입니다.

필수 여부: 아니요

secondarySourceVersions

유형: ProjectSourceVersion 객체 배열

배열 구성원: 최소수는 0개입니다. 최대 항목 수는 12개입니다.

필수 여부: 아니요

source

유형: PublicProjectSource객체

필수 여부: 아니요

sourceVersion

유형: 문자열

길이 제약: 최소 길이 1.

필수 여부: 아니요

timeoutInMinutes

유형: 정수

유효한 범위: 최소값은 5. 최대값은 480입니다.

필수 여부: 아니요

웹후크

유형: PublicWebhook객체

필수 여부: 아니요

PublicProjectArtifacts

> **Note**
>
> 이 API 요소는 AWS CLI AWS SDKs에 포함되지 않습니다.

내용

> **Note**
>
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

type

유형: 문자열

유효한 값: `CODEPIPELINE | S3 | NO_ARTIFACTS | DOCKER_IMAGE`

필수 항목 여부: 예

artifactIdentifier

유형: 문자열

필수 항목 여부: 아니요

location

유형: 문자열

필수 항목 여부: 아니요

PublicProjectBuildBatchConfig

> **Note**
>
> 이 API 요소는 AWS CLI AWS SDKs에 포함되지 않습니다.

내용

> **Note**
>
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

combineArtifacts

유형: 부울

필수 항목 여부: 아니요

제한 사항

배치 빌드에 대한 제한을 지정합니다.

유형: BatchRestrictions 객체

필수 여부: 아니요

timeoutInMins

유형: 정수

필수 항목 여부: 아니요

PublicProjectEnvironment

> **Note**
>
> 이 API 요소는 AWS CLI AWS SDKs에 포함되지 않습니다.

내용

> **Note**
>
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

computeType

유형: 문자열

유효한 값: `BUILD_GENERAL1_SMALL | BUILD_GENERAL1_MEDIUM | BUILD_GENERAL1_LARGE | BUILD_GENERAL1_2XLARGE`

필수 여부: 아니요

environmentVariables

유형: EnvironmentVariable 객체 배열

필수 여부: 아니요

image

유형: 문자열

길이 제약: 최소 길이 1.

필수 여부: 아니요

type

유형: 문자열

유효한 값: `WINDOWS_CONTAINER | WINDOWS_SERVER_2019_CONTAINER | LINUX_CONTAINER | LINUX_GPU_CONTAINER | ARM_CONTAINER | MAC`

필수 여부: 아니요

PublicProjectSource

> **Note**
>
> 이 API 요소는 AWS CLI AWS SDKs에 포함되지 않습니다.

내용

> **Note**
>
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

type

유형: 문자열

유효한 값: `NO_SOURCE | CODECOMMIT | CODEPIPELINE | GITHUB | S3 | BITBUCKET | GITHUB_ENTERPRISE | GITLAB | GITLAB_SELF_MANAGED`

필수 항목 여부: 예

buildspec

유형: 문자열

필수 항목 여부: 아니요

gitCloneDepth

유형: 정수

유효 범위: 최소값 0.

필수 여부: 아니요

gitSubmodulesConfig

AWS CodeBuild 빌드 프로젝트의 Git 하위 모듈 구성에 대한 정보입니다.

유형: GitSubmodulesConfig 객체

필수 여부: 아니요

location

유형: 문자열

필수 항목 여부: 아니요

sourceIdentifier

유형: 문자열

필수 항목 여부: 아니요

PublicWebhook

> **Note**
>
> 이 API 요소는 AWS CLI AWS SDKs에 포함되지 않습니다.

내용

> **Note**
>
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

branchFilter

유형: 문자열

필수 항목 여부: 아니요

buildType

유형: 문자열

필수 항목 여부: 아니요

filterGroups

유형: WebhookFilter 객체의 배열 배열

필수 여부: 아니요

payloadUrl

유형: 문자열

길이 제약: 최소 길이 1.

필수 여부: 아니요

url

유형: 문자열

길이 제약: 최소 길이 1.

필수 여부: 아니요

S3Downloadable

> **Note**
>
> 이 API 요소는 AWS CLI AWS SDKs에 포함되지 않습니다.

내용

> **Note**
>
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

expiredAt

유형: 타임스탬프

필수 여부: 아니요

identifier

유형: 문자열

필수 항목 여부: 아니요

md5Checksum

유형: 문자열

필수 항목 여부: 아니요

presignedUrl

유형: 문자열

필수 항목 여부: 아니요

s3Arn

유형: 문자열

필수 항목 여부: 아니요

sha256Checksum

유형: 문자열

필수 항목 여부: 아니요

문제 해결 AWS CodeBuild

이 주제의 정보를 활용하면 문제를 식별, 진단 및 해결하는 데 도움이 됩니다. 문제를 해결하기 위해 CodeBuild 빌드를 기록하고 모니터링하는 방법은 로깅 및 모니터링 섹션을 참조하세요.

주제

- Apache Maven 빌드가 잘못된 리포지토리의 아티팩트를 참조함
- 기본적으로 루트로 실행되는 빌드 명령
- 파일 이름에 미국 영어 이외의 문자가 있으면 빌드가 실패할 수 있음
- Amazon EC2 Parameter Store에서 파라미터를 가져올 때 빌드가 실패할 수 있음
- CodeBuild 콘솔에서 브랜치 필터에 액세스할 수 없음
- 빌드 성공 또는 실패 여부를 볼 수 없음
- 빌드 상태가 소스 공급자에게 보고되지 않음
- Windows Server Core 2019 플랫폼의 기본 이미지를 찾고 선택할 수 없습니다.
- buildspec 파일의 앞에 있는 명령을 나중에 있는 명령에서 인식하지 못함
- 오류: 캐시를 다운로드하려고 할 때 “Access denied”라는 메시지가 표시됨
- 오류: 사용자 지정 빌드 이미지를 사용할 때 "BUILD_CONTAINER_UNABLE_TO_PULL_IMAGE"라는 메시지가 표시됨
- 오류: “빌드를 완료하기 전에 빌드 컨테이너가 종료된 것으로 나타났습니다. 빌드 컨테이너가 메모리가 부족하거나 도커 이미지가 지원되지 않기 때문에 종료되었습니다. ErrorCode: 500"
- 오류: 빌드 실행 시 "도커 데몬에 연결할 수 없음"
- 오류: 빌드 프로젝트를 생성 또는 업데이트할 때 "CodeBuild is not authorized to perform: sts:AssumeRole"오류가 표시됨
- 오류: "GetBucketAcl 호출 중 오류 발생: 버킷 소유자가 변경되었거나 해당 서비스 역할에 s3:GetBucketAcl이라는 이름의 권한이 없습니다"
- 오류: 빌드를 실행할 때 "Failed to upload artifacts: Invalid arn"이라는 메시지가 표시됨
- 오류: "Git Clone Failed: unable to access 'your-repository-URL': SSL certificate problem: self signed certificate"
- 오류: 빌드를 실행할 때 "The bucket you are attempting to access must be addressed using the specified endpoint..."라는 메시지가 표시됨
- 오류: "이 빌드 이미지는 하나 이상의 런타임 버전을 선택해야 합니다."

- 오류: 빌드 대기열의 빌드가 실패할 때 "QUEUED: INSUFFICIENT_SUBNET"이라는 메시지가 표시됨
- 오류: "Unable to download cache: RequestError: Send request failed caused by: x509: Failed to load system roots and no roots provided"
- 오류: "Unable to download certificate from S3. AccessDenied"
- 오류: "Unable to locate credentials"
- 프록시 서버에서 CodeBuild를 실행할 때 RequestError 시간 초과 오류
- 빌드 이미지에 있어야 하는 Bourne 쉘(sh)
- 경고: 빌드 실행 시 “런타임 설치를 건너뜁니다. 런타임 버전 선택은 이 빌드 이미지에서 지원되지 않습니다.”가 표시됨
- CodeBuild 콘솔을 열 때 오류: “JobWorker ID를 확인할 수 없음”이 표시됨
- 빌드를 시작하지 못함
- 로컬로 캐시된 빌드의 GitHub 메타데이터에 액세스
- AccessDenied: 보고서 그룹의 버킷 소유자가 S3 버킷 소유자와 일치하지 않습니다.
- 오류:를 사용하여 CodeBuild 프로젝트를 생성할 때 "자격 증명에 필요한 권한 범위가 하나 이상 없음" AWS CodeConnections
- 오류: Ubuntu 설치 명령을 사용하여 빌드 시 '죄송합니다. 터미널이 전혀 요청되지 않았습니다. 입력을 가져올 수 없습니다'
- Secrets Manager OAuth 사용 후 인증 오류와 함께 Bitbucket 빌드 실패

Apache Maven 빌드가 잘못된 리포지토리의 아티팩트를 참조함

문제: Maven을 AWS CodeBuild제공 Java 빌드 환경과 함께 사용하는 경우 Maven은 https://repo1.maven.org/maven2 보안 중앙 Maven 리포지토리에서 빌드 및 플러그인 종속성을 가져옵니다. 빌드 프로젝트의 `pom.xml` 파일에 대신 사용할 다른 위치가 명시적으로 선언되어 있는 경우에도 이러한 문제가 발생합니다.

가능한 원인: CodeBuild 제공 Java 빌드 환경에 빌드 환경의 `/root/.m2` 디렉터리에 사전 설치되어 있는 `settings.xml`이라는 파일이 포함되어 있습니다. 이 `settings.xml` 파일에는 다음 선언이 포함되어 있는데, 이는 Maven이 항상 https://repo1.maven.org/maven2에 있는 보안된 중앙 Maven 리포지토리로부터 빌드 및 플러그인 종속성을 가져오도록 지시합니다.

```
<settings>
```

```
  <activeProfiles>
    <activeProfile>securecentral</activeProfile>
  </activeProfiles>
  <profiles>
    <profile>
      <id>securecentral</id>
      <repositories>
        <repository>
          <id>central</id>
          <url>https://repo1.maven.org/maven2</url>
          <releases>
            <enabled>true</enabled>
          </releases>
        </repository>
      </repositories>
      <pluginRepositories>
        <pluginRepository>
          <id>central</id>
          <url>https://repo1.maven.org/maven2</url>
          <releases>
            <enabled>true</enabled>
          </releases>
        </pluginRepository>
      </pluginRepositories>
    </profile>
  </profiles>
</settings>
```

권장 솔루션: 다음을 실행합니다.

1. 소스 코드에 `settings.xml` 파일을 추가합니다.
2. 이 `settings.xml` 파일에서, 이전 `settings.xml` 형식을 참고하여 Maven이 빌드 및 플러그인 종속성을 가져오게 하려는 다른 리포지토리를 선언합니다.
3. 빌드 프로젝트의 `install` 단계에서 CodeBuild가 사용자가 만든 `settings.xml` 파일을 빌드 환경의 `/root/.m2` 디렉터리에 복사하도록 명령을 설정합니다. 예를 들어 이러한 작업을 수행하는 `buildspec.yml` 파일의 다음 코드 조각을 고려해 보십시오.

```
version 0.2

phases:
  install:
    commands:
```

```
      - cp ./settings.xml /root/.m2/settings.xml
```

기본적으로 루트로 실행되는 빌드 명령

Issue: AWS CodeBuild 루트 사용자로 빌드 명령을 실행합니다. 관련 빌드 이미지의 Dockerfile이 USER 명령을 다른 사용자에게 설정하는 경우에도 이러한 문제가 발생합니다.

원인: CodeBuild는 기본적으로 루트 사용자로 모든 빌드 명령을 실행합니다.

권장 솔루션: 없음.

파일 이름에 미국 영어 이외의 문자가 있으면 빌드가 실패할 수 있음

문제: 영어 이외의 문자(예: 중국어 문자)가 포함된 파일 이름을 사용하는 빌드를 실행하면 빌드가 실패합니다.

가능한 원인: AWS CodeBuild 가 제공하는 빌드 환경은 기본 로캘이 POSIX로 설정되어 있습니다. POSIX 현지화 설정은 CodeBuild 및 영어 이외의 문자가 포함된 파일 이름과 호환되지 않으므로 관련 빌드가 실패할 수 있습니다.

권장 솔루션: 다음 명령을 buildspec 파일의 pre_build 섹션에 추가합니다. 이 명령을 사용하면 빌드 환경에서 현지화 설정에 영어(미국) UTF-8을 사용하게 되므로 CodeBuild 및 영어 이외의 문자가 포함된 파일 이름과 더 호환됩니다.

Ubuntu 기반의 빌드 환경:

```
pre_build:
  commands:
    - export LC_ALL="en_US.UTF-8"
    - locale-gen en_US en_US.UTF-8
    - dpkg-reconfigure -f noninteractive locales
```

Amazon Linux 기반의 빌드 환경:

```
pre_build:
  commands:
    - export LC_ALL="en_US.utf8"
```

Amazon EC2 Parameter Store에서 파라미터를 가져올 때 빌드가 실패할 수 있음

문제: 빌드가 Amazon EC2 Parameter Store에 저장된 하나 이상의 파라미터 값을 가져오려고 하면 `Parameter does not exist`라는 오류가 표시되면서 DOWNLOAD_SOURCE 단계에서 빌드가 실패합니다.

가능한 원인: 빌드 프로젝트가 사용하는 서비스 역할에 `ssm:GetParameters` 작업을 호출할 권한이 없거나 빌드 프로젝트가에서 생성 AWS CodeBuild 되고 `ssm:GetParameters` 작업 호출을 허용하는 서비스 역할을 사용하지만 파라미터에는 로 시작하지 않는 이름이 있습니다`/CodeBuild/`.

권장 솔루션:

- 서비스 역할이 CodeBuild로 생성되지 않은 경우 CodeBuild가 `ssm:GetParameters` 작업을 호출할 수 있게 허용하도록 정의를 업데이트합니다. 예를 들어 다음 정책 설명은 `ssm:GetParameters` 작업 호출을 허용하여 `/CodeBuild/`로 시작하는 이름을 가진 파라미터를 가져옵니다.

  JSON

  ```
  {
      "Version":"2012-10-17",
      "Statement": [
          {
              "Action": "ssm:GetParameters",
              "Effect": "Allow",
              "Resource": "arn:aws:ssm:us-east-1:111122223333:parameter/
  CodeBuild/*"
          }
      ]
  }
  ```

- 서비스 역할이 CodeBuild로 생성되지 않은 경우 CodeBuild가 Amazon EC2 Parameter Store에서 `/CodeBuild/`로 시작하는 이름이 아닌 다른 이름의 파라미터에 액세스할 수 있게 허용하도록 정의를 업데이트합니다. 예를 들어 다음 정책 설명은 `ssm:GetParameters` 작업 호출을 허용하여 지정된 이름을 가진 파라미터를 가져옵니다.

  JSON

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

```
        {
            "Action": "ssm:GetParameters",
            "Effect": "Allow",
            "Resource": "arn:aws:ssm:us-
east-1:111122223333:parameter/PARAMETER_NAME"
        }
    ]
}
```

CodeBuild 콘솔에서 브랜치 필터에 액세스할 수 없음

문제: AWS CodeBuild 프로젝트를 생성하거나 업데이트할 때 콘솔에서 브랜치 필터 옵션을 사용할 수 없습니다.

가능한 원인: 브랜치 필터 옵션이 사용되지 않습니다. 이 옵션은 CodeBuild에서 새 빌드를 트리거하는 Webhook 이벤트에 대해 더 다양한 제어를 제공하는 Webhook 필터 그룹으로 대체되었습니다.

권장 솔루션: Webhook 필터 도입 이전에 생성된 브랜치 필터를 마이그레이션하려면 정규식 `^refs/heads/branchName$`를 사용하여 HEAD_REF 필터를 포함하는 Webhook 필터 그룹을 생성합니다. 예를 들어 브랜치 필터 정규식이 `^branchName$`이었다면 HEAD_REF 필터에 삽입하는 업데이트된 정규식은 `^refs/heads/branchName$`입니다. 자세한 내용은 Bitbucket Webhook 이벤트 및 GitHub Webhook 이벤트 필터링(콘솔) 섹션을 참조하세요.

빌드 성공 또는 실패 여부를 볼 수 없음

문제: 재시도한 빌드의 성공 또는 실패 여부를 볼 수 없습니다.

가능한 원인: 빌드 상태를 보고하는 옵션이 활성화되어 있지 않습니다.

권장 솔루션: CodeBuild 프로젝트를 생성하거나 업데이트할 때 보고서 빌드 상태를 활성화하세요. 이 옵션은 CodeBuild에 빌드가 트리거되었을 때 상태를 다시 보고하도록 지시합니다. 자세한 내용은 AWS CodeBuild API 참조의 reportBuildStatus를 참조하세요.

빌드 상태가 소스 공급자에게 보고되지 않음

문제: GitHub 또는 Bitbucket과 같은 소스 공급자에게 빌드 상태 보고를 허용한 후 빌드 상태가 업데이트되지 않습니다.

가능한 원인: 소스 공급자와 연결된 사용자에게 리포지토리에 대한 쓰기 권한이 없습니다.

권장 솔루션: 소스 공급자에게 빌드 상태를 보고하려면 소스 공급자와 연결된 사용자에게 리포지토리에 대한 쓰기 권한이 있어야 합니다. 사용자에게 쓰기 권한이 없는 경우 빌드 상태를 업데이트할 수 없습니다. 자세한 내용은 소스 공급자 액세스 섹션을 참조하세요.

Windows Server Core 2019 플랫폼의 기본 이미지를 찾고 선택할 수 없습니다.

문제: Windows Server Core 2019 플랫폼의 기본 이미지를 찾거나 선택할 수 없습니다.

가능한 원인:이 이미지를 지원하지 않는 AWS 리전을 사용하고 있습니다.

권장 솔루션: Windows Server Core 2019 플랫폼의 기본 이미지가 지원되는 다음 AWS 리전 중 하나를 사용합니다.

- 미국 동부(버지니아 북부)
- 미국 동부(오하이오)
- 미국 서부(오리건)
- 유럽(아일랜드)

buildspec 파일의 앞에 있는 명령을 나중에 있는 명령에서 인식하지 못함

문제: buildspec 파일에 있는 하나 이상의 명령 결과를 동일한 buildspec 파일의 나중에 있는 명령에서 인식하지 못합니다. 예를 들어, 명령에서 로컬 환경 변수를 설정했지만 나중에 명령이 실행될 때 해당 로컬 환경 변수 값을 가져오지 못할 수 있습니다.

가능한 원인: buildspec 파일 버전 0.1에서 AWS CodeBuild 는 빌드 환경에 있는 기본 쉘의 서로 다른 인스턴스에서 각 명령을 실행합니다. 즉, 각 명령이 다른 모든 명령과 독립적으로 실행됩니다. 그러면 기본적으로 이전 명령의 상태에 따라 실행 여부가 결정되는 단일 명령을 실행할 수 없습니다.

권장 솔루션:이 문제를 해결하는 buildspec 버전 0.2를 사용하는 것이 좋습니다. buildspec 버전 0.1을 사용해야 하는 경우, 쉘 명령 결합 연산자(예: Linux의 &&)를 사용하여 여러 명령을 하나의 명령으로 결합하는 것이 좋습니다. 또는 여러 명령을 포함하는 쉘 스크립트를 소스 코드에 포함한 다음, buildspec 파일에서 단일 명령으로 해당 쉘 스크립트를 호출합니다. 자세한 내용은 빌드 환경의 쉘 및 명령 및 빌드 환경의 환경 변수 섹션을 참조하세요.

오류: 캐시를 다운로드하려고 할 때 “Access denied”라는 메시지가 표시됨

문제: 캐시가 활성화된 빌드 프로젝트에서 캐시를 다운로드하려고 하면 `Access denied` 오류가 표시됩니다.

가능한 원인:

- 방금 빌드 프로젝트의 일부로 캐싱을 구성했습니다.
- 최근 `InvalidateProjectCache` API를 통해 캐시가 무효화되었습니다.
- CodeBuild에서 사용 중인 서비스 역할에는 캐시를 보유하고 있는 S3 버킷에 대한 `s3:GetObject` 및 `s3:PutObject` 권한이 없습니다.

권장 솔루션: 처음으로 사용하는 경우 일반적으로 캐시 구성을 업데이트한 직후에 확인할 수 있습니다. 이러한 오류가 계속되는 경우 서비스 역할에 캐시를 보유하고 있는 S3 버킷에 대한 `s3:GetObject` 및 `s3:PutObject` 권한이 있는지 확인해야 합니다. 자세한 내용을 알아보려면 Amazon S3 개발자 안내서의 S3 권한 지정을 참조하세요.

오류: 사용자 지정 빌드 이미지를 사용할 때 "BUILD_CONTAINER_UNABLE_TO_PULL_IMAGE"라는 메시지가 표시됨

문제: 사용자 지정 빌드 이미지를 사용하는 빌드를 실행하려고 할 때 `BUILD_CONTAINER_UNABLE_TO_PULL_IMAGE`라는 오류가 표시되며 빌드가 실패합니다.

가능한 원인: 빌드 이미지의 압축되지 않은 전체 크기는 빌드 환경 컴퓨팅 유형의 사용 가능한 디스크 공간보다 큽니다. 빌드 이미지의 크기를 확인하려면 Docker를 사용하여 *`docker images REPOSITORY:TAG`* 명령을 실행합니다. 컴퓨팅 유형별로 사용 가능한 디스크 공간 목록은 빌드 환경 컴퓨팅 모드 및 유형 단원을 참조하십시오.

권장 솔루션: 사용 가능한 디스크 공간이 많은 더 큰 컴퓨팅 유형을 사용하거나 사용자 지정 빌드 이미지의 크기를 줄입니다.

가능한 원인: AWS CodeBuild Amazon Elastic Container Registry(Amazon ECR)에서 빌드 이미지를 가져올 권한이 없습니다.

권장 솔루션: CodeBuild가 사용자 지정 빌드 이미지를 빌드 환경으로 가져올 수 있도록 Amazon ECR에서 리포지토리의 권한을 업데이트합니다. 자세한 내용은 Amazon ECR 샘플 섹션을 참조하세요.

가능한 원인: 요청한 Amazon ECR 이미지를 AWS 계정이 사용 중인 AWS 리전에서 사용할 수 없습니다.

권장 솔루션: AWS 계정에서 사용 중인 리전과 동일한 AWS 리전에 있는 Amazon ECR 이미지를 사용합니다.

가능한 원인: 퍼블릭 인터넷 액세스가 불가능한 VPC에서 프라이빗 레지스트리를 사용하고 있습니다. CodeBuild는 VPC의 프라이빗 IP 주소에서 이미지를 끌어올 수 없습니다. 자세한 내용은 CodeBuild용 AWS Secrets Manager 샘플이 포함된 프라이빗 레지스트리 섹션을 참조하세요.

권장 솔루션: VPC에서 프라이빗 레지스트리를 사용하는 경우 VPC가 퍼블릭 인터넷에 액세스할 수 있는지 확인합니다.

가능한 원인: 오류 메시지에 "toomanyrequests“이 포함되어 있고 Docker Hub에서 이미지를 가져온 경우 이 오류는 Docker Hub 풀 제한에 도달했음을 의미합니다.

권장 솔루션: Docker Hub 프라이빗 레지스트리를 사용하거나 Amazon ECR에서 이미지를 가져옵니다. 프라이빗 레지스트리 사용에 대한 자세한 내용은 CodeBuild용 AWS Secrets Manager 샘플이 포함된 프라이빗 레지스트리 섹션을 참조하세요. Amazon ECR 사용에 대한 자세한 내용은 CodeBuild용 Amazon ECR 샘플 섹션을 참조하세요.

오류: “빌드를 완료하기 전에 빌드 컨테이너가 종료된 것으로 나타났습니다. 빌드 컨테이너가 메모리가 부족하거나 도커 이미지가 지원되지 않기 때문에 종료되었습니다. ErrorCode: 500"

문제:에서 Microsoft Windows 또는 Linux 컨테이너를 사용하려고 하면 프로비저닝 단계에서 AWS CodeBuild이 오류가 발생합니다.

가능한 원인:

- CodeBuild에서 컨테이너 OS 버전을 지원하지 않습니다.
- HTTP_PROXY, HTTPS_PROXY 또는 둘 다 컨테이너에서 지정됩니다.

권장 솔루션:

- Microsoft Windows의 경우, microsoft/windowsservercore:10.0.x 버전의 컨테이너 OS를 설치한 Windows 컨테이너를 사용합니다(예: microsoft/windowsservercore:10.0.14393.2125).
- Linux의 경우, 도커 이미지에서 HTTP_PROXY 및 HTTPS_PROXY 설정을 지우거나 빌드 프로젝트에서 VPC 구성을 지정합니다.

오류: 빌드 실행 시 "도커 데몬에 연결할 수 없음"

문제: 빌드에 실패하여 빌드 로그에 `Cannot connect to the Docker daemon at unix:/var/run/docker.sock. Is the docker daemon running?`과 유사한 오류가 표시됩니다.

가능한 원인: 권한이 있는 모드에서 빌드를 실행하지 않았습니다.

권장 솔루션: 이 오류를 해결하려면 권한 모드를 활성화하고 다음 지침을 사용하여 buildspec을 업데이트해야 합니다.

다음 단계에 따라 권한이 있는 모드에서 빌드를 실행합니다.

1. https://console.aws.amazon.com/codebuild/에서 CodeBuild 콘솔을 엽니다.
2. 탐색 창에서 빌드 프로젝트를 선택한 후 빌드 프로젝트를 선택합니다.
3. Edit(편집)에서 Environment(환경)을 선택합니다.
4. 추가 구성을 선택합니다.
5. 권한 있음에서 Docker 이미지를 빌드하거나 빌드에서 승격된 권한을 얻으려는 경우 이 플래그 활성화를 선택합니다.
6. Update environment(환경 업데이트)를 선택합니다.
7. Start build(빌드 시작)을 선택하여 빌드를 다시 시도합니다.

컨테이너 내에서 Docker 데몬도 시작해야 합니다. Buildspec의 `install` 단계는 이와 유사할 수 있습니다.

```
phases:
  install:
    commands:
      - nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --
host=tcp://127.0.0.1:2375 --storage-driver=overlay2 &
      - timeout 15 sh -c "until docker info; do echo .; sleep 1; done"
```

buildspec 파일에 참조된 OverlayFS 스토리지 드라이버에 대한 자세한 내용은 도커 웹사이트의 OverlayFS 스토리지 드라이버 사용을 참조하십시오.

> **Note**
>
> 기본 운영 체제가 Alpine Linux인 경우 `buildspec.yml`에서 `-t` 인수를 `timeout`에 추가합니다.
>
> ```
> - timeout -t 15 sh -c "until docker info; do echo .; sleep 1; done"
> ```

를 사용하여 도커 이미지를 빌드하고 실행하는 방법에 대한 자세한 내용은 섹션을 AWS CodeBuild참조하세요CodeBuild용 Docker 사용자 지정 이미지 샘플.

오류: 빌드 프로젝트를 생성 또는 업데이트할 때 "CodeBuild is not authorized to perform: sts:AssumeRole"오류가 표시됨

문제: 빌드 프로젝트를 생성하거나 업데이트하려고 하면 `Code:InvalidInputException, Message:CodeBuild is not authorized to perform: sts:AssumeRole on arn:aws:iam::`*`account-ID`*`:role/`*`service-role-name`* 오류가 표시됩니다.

가능한 원인:

- 빌드 프로젝트를 생성하거나 업데이트하려는 AWS 리전에 대해 AWS Security Token Service (AWS STS)가 비활성화되었습니다.
- 빌드 프로젝트와 연결된 AWS CodeBuild 서비스 역할이 존재하지 않거나 CodeBuild를 신뢰할 수 있는 충분한 권한이 없습니다.
- 빌드 프로젝트와 연결된 AWS CodeBuild 서비스 역할 대소문자가 실제 IAM 역할과 일치하지 않습니다.

권장 솔루션:

- 빌드 프로젝트를 생성하거나 업데이트하려는 AWS 리전에 대해 AWS STS 가 활성화되어 있는지 확인합니다. 자세한 내용은 IAM 사용 설명서의 AWS 리전 AWS STS 에서 활성화 및 비활성화를 참조하세요.
- 대상 CodeBuild 서비스 역할이 AWS 계정에 있는지 확인합니다. 콘솔을 사용하고 있지 않은 경우, 빌드 프로젝트를 생성하거나 업데이트할 때 서비스 역할의 Amazon 리소스 이름(ARN)을 잘못 입력

하지 않았는지 확인합니다. IAM 역할은 대소문자를 구분하므로 IAM 역할의 대소문자가 올바른지 확인합니다.

- 대상 CodeBuild 서비스 역할에 CodeBuild를 신뢰할 만큼 충분한 권한이 있는지 확인합니다. 자세한 내용은 CodeBuild가 다른 AWS 서비스와 상호 작용하도록 허용 섹션의 신뢰 관계 정책 설명을 참조하십시오.

오류: "GetBucketAcl 호출 중 오류 발생: 버킷 소유자가 변경되었거나 해당 서비스 역할에 s3:GetBucketAcl이라는 이름의 권한이 없습니다"

문제: 빌드를 실행하면 S3 버킷 및 `GetBucketAcl` 권한 소유자 변동에 관한 오류가 표시됩니다.

가능한 원인: `s3:GetBucketAcl` 및 `s3:GetBucketLocation` 권한을 IAM 역할에 추가했습니다. 이러한 권한은 프로젝트의 S3 버킷을 보호하여 사용자만 액세스할 수 있게 합니다. 이러한 권한을 추가한 후에 S3 버킷 소유자가 변경되었습니다.

권장 솔루션: 사용자가 S3 버킷 소유자인지 확인한 후에 IAM 역할에 다시 권한을 추가합니다. 자세한 내용은 S3 버킷에 대한 보안 액세스 섹션을 참조하세요.

오류: 빌드를 실행할 때 "Failed to upload artifacts: Invalid arn"이라는 메시지가 표시됨

문제: 빌드를 실행하면 `Failed to upload artifacts: Invalid arn` 오류가 표시되면서 `UPLOAD_ARTIFACTS` 빌드 단계가 실패합니다.

가능한 원인: S3 출력 버킷(빌드에서 출력을 AWS CodeBuild 저장하는 버킷)이 CodeBuild 빌드 프로젝트와 다른 AWS 리전에 있습니다.

권장 솔루션: 빌드 프로젝트와 동일한 AWS 리전에 있는 출력 버킷을 가리키도록 빌드 프로젝트의 설정을 업데이트합니다.

오류: "Git Clone Failed: unable to access **'your-repository-URL'**: SSL certificate problem: self signed certificate"

문제: 빌드 프로젝트를 실행하려고 하면 이 오류가 표시되면서 빌드가 실패합니다.

가능한 원인: 소스 리포지토리에 자체 서명된 인증서가 있지만 빌드 프로젝트의 일부로 S3 버킷에서 인증서를 설치하도록 선택하지 않은 것입니다.

권장 솔루션:

- 프로젝트를 편집합니다. 인증서에서 Install certificate from S3(S3에서 인증서 설치)를 선택합니다. Bucket of certificate(인증서 버킷)에 SSL 인증서가 저장된 S3 버킷을 선택합니다. 인증서의 객체 키에 S3 객체 키의 이름을 입력합니다.
- 프로젝트를 편집합니다. GitHub Enterprise Server 프로젝트 리포지토리에 연결되어 있는 동안 SSL을 무시하려면 [Insecure SSL]을 선택합니다.

> **Note**
>
> [Insecure SSL]은 테스트 용도로만 사용하는 것이 좋습니다. 프로덕션 환경에 사용하면 안 됩니다.

오류: 빌드를 실행할 때 "The bucket you are attempting to access must be addressed using the specified endpoint..."라는 메시지가 표시됨

문제: 빌드를 실행하면 `The bucket you are attempting to access must be addressed using the specified endpoint. Please send all future requests to this endpoint` 오류가 표시되면서 `DOWNLOAD_SOURCE` 빌드 단계가 실패합니다.

가능한 원인: 사전 구축된 소스 코드는 S3 버킷에 저장되며 해당 버킷은 빌드 프로젝트와 다른 AWS 리전에 있습니다 AWS CodeBuild .

권장 솔루션: 사전 작성된 소스 코드가 포함되어 있는 버킷을 가리키도록 빌드 프로젝트 설정을 업데이트합니다. 버킷이 빌드 프로젝트와 동일한 AWS 리전에 있는지 확인합니다.

오류: "이 빌드 이미지는 하나 이상의 런타임 버전을 선택해야 합니다."

문제: 빌드를 실행하면 `YAML_FILE_ERROR: This build image requires selecting at least one runtime version` 오류가 표시되면서 `DOWNLOAD_SOURCE` 빌드 단계가 실패합니다.

가능한 원인: 빌드에서 Amazon Linux 2(AL2) 표준 이미지 버전 1.0 이상 또는 Ubuntu 표준 이미지 버전 2.0 이상을 사용하고, buildspec 파일에 런타임이 지정되어 있지 않습니다.

권장 솔루션: `aws/codebuild/standard:2.0` CodeBuild 관리형 이미지를 사용할 경우에는 buildspec 파일의 `runtime-versions` 섹션에서 런타임 버전을 지정해야 합니다. 예를 들어, PHP를 사용하는 프로젝트에는 다음 buildspec 파일을 사용해야 합니다.

```
version: 0.2

phases:
  install:
    runtime-versions:
        php: 7.3
  build:
    commands:
      - php --version
artifacts:
  files:
    -  README.md
```

> **Note**
>
> `runtime-versions` 섹션을 지정하고 Ubuntu 표준 이미지 2.0 이상 또는 Amazon Linux 2(AL2) 표준 이미지 1.0 이상 외의 이미지를 사용하는 경우, 빌드에서 "`Skipping install of runtimes. Runtime version selection is not supported by this build image`" 경고가 발생합니다.

자세한 내용은 Specify runtime versions in the buildspec file 섹션을 참조하세요.

오류: 빌드 대기열의 빌드가 실패할 때 "QUEUED: INSUFFICIENT_SUBNET"이라는 메시지가 표시됨

문제: `QUEUED: INSUFFICIENT_SUBNET`과 유사한 오류로 빌드 대기열의 빌드가 실패합니다.

가능한 원인: VPC에 지정된 IPv4 CIDR 블록이 예약된 IP 주소를 사용합니다. 각 서브넷 CIDR 블록에서 처음 4개의 IP 주소와 마지막 IP 주소는 사용자가 사용할 수 없으므로 인스턴스에 할당할 수 없습니다. 예를 들어 `10.0.0.0/24` CIDR 블록의 서브넷에서는 다음 5개 IP 주소가 예약되어 있습니다.

- `10.0.0.0`:: 네트워크 주소
- `10.0.0.1`: VPC 라우터에 AWS 대해에서 예약했습니다.
- `10.0.0.2`: 예약자입니다 AWS. DNS 서버의 IP 주소는 항상 기본 VPC 네트워크 범위에 2를 더한 주소입니다. 그러나 각 서브넷 범위에 2를 더한 주소도 예약합니다. CIDR 블록이 여러 개인 VPC의 경우, DNS 서버의 IP 주소가 기본 CIDR에 위치합니다. 자세한 내용은 Amazon VPC 사용 설명서의 Amazon DNS 서버를 참조하세요.
- `10.0.0.3`: 향후 사용을 AWS 위해에서 예약했습니다.
- `10.0.0.255`: 네트워크 브로드캐스트 주소. VPC에서는 브로드캐스트를 지원하지 않습니다. 이 주소는 예약되어 있습니다.

권장 솔루션: VPC가 예약된 IP 주소를 사용하는지 확인합니다. 예약된 IP 주소를 예약되지 않은 IP 주소로 바꿉니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC 및 서브넷 크기 조정을 참조하세요.

오류: "Unable to download cache: RequestError: Send request failed caused by: x509: Failed to load system roots and no roots provided"

문제: 빌드 프로젝트를 실행하려고 하면 이 오류가 표시되면서 빌드가 실패합니다.

가능한 원인: 캐시를 빌드 프로젝트의 일부로 구성했으며 만료된 루트 인증서가 포함된 기존 도커 이미지를 사용하고 있습니다.

권장 솔루션: AWS CodeBuild 프로젝트에서 사용 중인 Docker 이미지를 업데이트합니다. 자세한 내용은 CodeBuild가 제공하는 도커 이미지 섹션을 참조하세요.

오류: "Unable to download certificate from S3. AccessDenied"

문제: 빌드 프로젝트를 실행하려고 하면 이 오류가 표시되면서 빌드가 실패합니다.

가능한 원인:

- 인증서에 대해 잘못된 S3 버킷을 선택한 것입니다.
- 인증서에 대해 잘못된 객체 키를 입력한 것입니다.

권장 솔루션:

- 프로젝트를 편집합니다. Bucket of certificate(인증서 버킷)에 SSL 인증서가 저장된 S3 버킷을 선택합니다.
- 프로젝트를 편집합니다. 인증서의 객체 키에 S3 객체 키의 이름을 입력합니다.

오류: "Unable to locate credentials"

문제:를 실행하거나 AWS CLI SDK를 AWS 사용하거나 빌드의 일부로 다른 유사한 구성 요소를 호출하려고 하면 AWS CLI, , AWS SDK 또는 구성 요소와 직접 관련된 빌드 오류가 발생합니다. 예를 들어 `Unable to locate credentials`와 같은 빌드 오류가 발생할 수 있습니다.

가능한 원인:

- 빌드 환경의 AWS CLI, AWS SDK 또는 구성 요소의 버전이와 호환되지 않습니다 AWS CodeBuild.
- Docker를 사용하는 빌드 환경 내에서 Docker 컨테이너를 실행 중이며 컨테이너는 기본적으로 AWS 자격 증명에 액세스할 수 없습니다.

권장 솔루션:

- 빌드 환경에 다음 버전 이상의 AWS CLI, AWS SDK 또는 구성 요소가 있는지 확인합니다.
  - AWS CLI: 1.10.47
  - AWS C++용 SDK: 0.2.19
  - AWS Go용 SDK: 1.2.5
  - AWS Java용 SDK: 1.11.16
  - AWS JavaScript용 SDK: 2.4.7
  - AWS PHP용 SDK: 3.18.28
  - AWS SDK for Python(Boto3): 1.4.0
  - AWS SDK for Ruby: 2.3.22
  - Botocore: 1.4.37
  - CoreCLR: 3.2.6-beta
  - Node.js: 2.4.7
- 빌드 환경에서 Docker 컨테이너를 실행해야 하고 컨테이너에 AWS 자격 증명이 필요한 경우 빌드 환경에서 컨테이너로 자격 증명을 전달해야 합니다. buildspec 파일에서 다음과 같은 도커 `run` 명령을 포함합니다. 이 예에서는 `aws s3 ls` 명령을 사용하여 사용 가능한 S3 버킷을 나열합니다. `-e` 옵션은 컨테이너가 AWS 자격 증명에 액세스하는 데 필요한 환경 변수를 전달합니다.

```
docker run -e AWS_DEFAULT_REGION -e AWS_CONTAINER_CREDENTIALS_RELATIVE_URI your-image-tag aws s3 ls
```

- Docker 이미지를 빌드할 때 빌드에 AWS 자격 증명이 필요한 경우(예: Amazon S3에서 파일을 다운로드하려면) 다음과 같이 빌드 환경에서 Docker 빌드 프로세스로 자격 증명을 전달해야 합니다.

  1. 도커 이미지용 소스 코드의 Dockerfile에서 다음 ARG 지침을 지정합니다.

     ```
     ARG AWS_DEFAULT_REGION
     ARG AWS_CONTAINER_CREDENTIALS_RELATIVE_URI
     ```

  2. buildspec 파일에서 다음과 같은 도커 build 명령을 포함합니다. --build-arg 옵션은 Docker 빌드 프로세스가 AWS 자격 증명에 액세스하는 데 필요한 환경 변수를 설정합니다.

     ```
     docker build --build-arg AWS_DEFAULT_REGION=$AWS_DEFAULT_REGION --build-arg
      AWS_CONTAINER_CREDENTIALS_RELATIVE_URI=$AWS_CONTAINER_CREDENTIALS_RELATIVE_URI -
     t your-image-tag .
     ```

프록시 서버에서 CodeBuild를 실행할 때 RequestError 시간 초과 오류

문제: 다음 중 하나와 비슷한 RequestError 오류가 발생합니다.

- CloudWatch Logs의 RequestError: send request failed caused by: Post https://logs.<your-region>.amazonaws.com/: dial tcp 52.46.158.105:443: i/o timeout
- Amazon S3의 Error uploading artifacts: RequestError: send request failed caused by: Put https://*your-bucket*.s3.*your-aws-region*.amazonaws.com/*: dial tcp 52.219.96.208:443: connect: connection refused

가능한 원인:

- ssl-bump가 제대로 구성되지 않았습니다.
- 조직의 보안 정책이 사용자가 ssl_bump를 사용하는 것을 허용하지 않습니다.
- buildspec 파일에 proxy 요소를 사용하여 지정한 프록시 설정이 없습니다.

권장 솔루션:

- `ssl-bump`가 올바로 구성되었는지 확인하십시오. Squid를 프록시 서버로 사용하는 경우 Squid를 명시적 프록시 서버로 구성 단원을 참조하십시오.
- 다음 단계에 따라 Amazon S3 및 CloudWatch Logs에 대해 프라이빗 엔드포인트를 사용합니다.

  1. 프라이빗 서브넷 라우팅 테이블에서 이전에 추가한 인터넷으로 향하는 트래픽을 프록시 서버로 라우팅하는 규칙을 제거합니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC에서 서브넷 만들기를 참조하세요.
  2. 프라이빗 Amazon S3 엔드포인트 및 CloudWatch Logs 엔드포인트를 생성한 후 Amazon VPC의 프라이빗 서브넷과 연결합니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC 엔드포인트 서비스를 참조하세요.
  3. Amazon VPC에서 프라이빗 DNS 이름 활성화가 선택되었는지 확인합니다. 자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트 생성을 참조하세요.

- 명시적 프록시 서버에 `ssl-bump`를 사용하지 않을 경우 `proxy` 요소를 사용하여 buildspec 파일에 프록시 구성을 추가하십시오. 자세한 내용은 명시적 프록시 서버에서 CodeBuild 실행 및 buildspec 구문 섹션을 참조하세요.

```
version: 0.2
proxy:
  upload-artifacts: yes
  logs: yes
phases:
  build:
    commands:
```

빌드 이미지에 있어야 하는 Bourne 쉘(sh)

문제:에서 제공하지 않는 빌드 이미지를 사용하고 AWS CodeBuild있으며 메시지와 함께 빌드가 실패합니다`Build container found dead before completing the build.`

가능한 원인: Bourne 쉘(sh)이 빌드 이미지에 포함되어 있지 않습니다. 빌드 명령과 스크립트를 실행하려면 CodeBuild에 `sh`가 필요합니다.

권장 솔루션: `sh`가 빌드 이미지에 없는 경우 이미지를 사용하는 빌드를 더 시작하기 전에 이를 포함해야 합니다. (CodeBuild의 빌드 이미지에는 이미 `sh`가 포함되어 있습니다.)

경고: 빌드 실행 시 “런타임 설치를 건너뜁니다. 런타임 버전 선택은 이 빌드 이미지에서 지원되지 않습니다.”가 표시됨

문제: 빌드를 실행하면 빌드 로그에 이 경고가 포함되어 있습니다.

가능한 원인: 빌드에서 Amazon Linux 2(AL2) 표준 이미지 버전 1.0 이상 또는 Ubuntu 표준 이미지 버전 2.0 이상을 사용하지 않고, buildspec 파일의 `runtime-versions` 섹션에 런타임이 지정되어 있습니다.

권장 솔루션: buildspec 파일에 `runtime-versions` 섹션이 포함되지 않게 하십시오. `runtime-versions` 섹션은 Amazon Linux 2(AL2) 표준 이미지 버전 이상 또는 Ubuntu 표준 이미지 버전 2.0 이상을 사용하는 경우에만 필요합니다.

CodeBuild 콘솔을 열 때 오류: “JobWorker ID를 확인할 수 없음”이 표시됨

문제: CodeBuild 콘솔을 열면 “JobWorker ID를 확인할 수 없음” 오류 메시지가 표시됩니다.

가능한 원인: 콘솔 액세스에 사용되는 IAM 역할에 `jobId` 키가 있는 태그가 있습니다. 이 태그 키는 CodeBuild용으로 예약되어 있으며 이 태그 키가 있는 경우 이 오류가 발생합니다.

권장 솔루션: `jobId` 키가 있는 사용자 지정 IAM 역할 태그를 다른 키를 사용하도록 변경합니다(예: `jobIdentifier`).

빌드를 시작하지 못함

문제: 빌드를 시작할 때 빌드를 시작하지 못함 오류 메시지가 나타납니다.

가능한 원인: 동시 빌드 수에 도달했습니다.

권장 솔루션: 다른 빌드가 완료될 때까지 기다리거나 프로젝트의 동시 빌드 제한을 늘리고 빌드를 다시 시작합니다. 자세한 내용은 프로젝트 구성 섹션을 참조하세요.

로컬로 캐시된 빌드의 GitHub 메타데이터에 액세스

문제: 캐시된 빌드의 .git 디렉터리가 디렉터리가 아닌 텍스트 파일인 경우가 있습니다.

가능한 원인: 빌드에 대해 로컬 소스 캐싱이 활성화되면 CodeBuild는 `.git` 디렉터리에 대한 gitlink를 생성합니다. 즉, `.git` 디렉터리는 실제로 디렉터리 경로가 포함된 텍스트 파일입니다.

권장 솔루션: 모든 경우에 다음 명령을 사용하여 Git 메타데이터 디렉터리를 가져오세요. 이 명령은 다음과 같은 `.git` 형식에 관계없이 작동합니다.

```
git rev-parse --git-dir
```

AccessDenied: 보고서 그룹의 버킷 소유자가 S3 버킷 소유자와 일치하지 않습니다.

문제: Amazon S3 버킷에 테스트 데이터를 업로드할 때 CodeBuild가 테스트 데이터를 버킷에 쓸 수 없습니다.

가능한 원인:

- 보고서 그룹 버킷 소유자로 지정된 계정이 Amazon S3 버킷 소유자와 일치하지 않습니다.
- 서비스 역할에는 버킷에 대한 쓰기 권한이 없습니다.

권장 솔루션:

- Amazon S3 버킷 소유자와 일치하도록 보고서 그룹 버킷 소유자를 변경합니다.
- Amazon S3 버킷에 대한 쓰기 액세스 권한을 허용하도록 서비스 역할을 수정합니다.

오류:를 사용하여 CodeBuild 프로젝트를 생성할 때 "자격 증명에 필요한 권한 범위가 하나 이상 없음" AWS CodeConnections

문제:를 사용하여 CodeBuild 프로젝트를 생성할 때 Bitbucket 웹후크를 설치할 권한이 AWS CodeConnections없습니다.

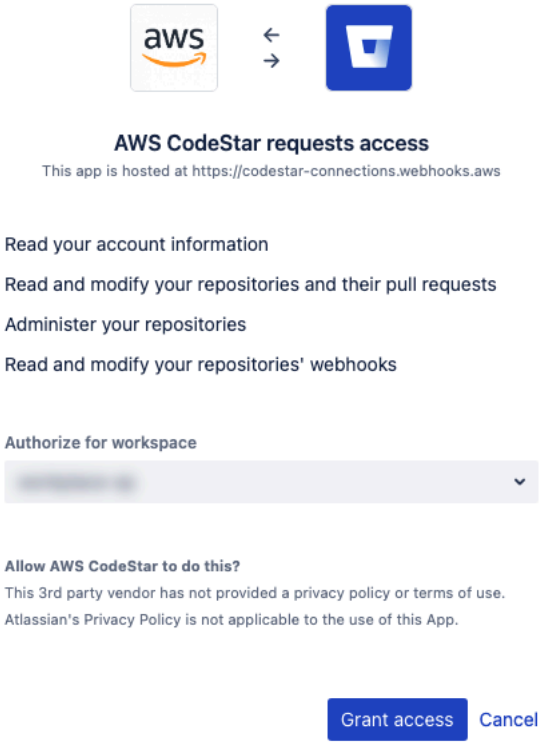
가능한 원인:

- Bitbucket 계정에서 새 권한 범위가 수락되지 않았을 수 있습니다.

권장 솔루션:

- 새 권한을 수락하려면 Bitbucket,에서 보낸 작업 필요 - Scopes for AWS CodeStar라는 제목의 이메일을 수신했어야 합니다`notifications-noreply@bitbucket.org`. 이메일에는 기존 AWS CodeConnections Bitbucket 앱 설치에 웹후크 권한을 부여하는 링크가 포함되어 있습니다.

- 이메일을 찾을 수 없는 경우 `https://bitbucket.org/site/addons/reauthorize?account=<workspace-name>&addon_key=aws-codestar` 또는 `https://bitbucket.org/site/addons/reauthorize?addon_key=aws-codestar`로 이동하고 웹후크 권한을 부여하려는 워크스페이스를 선택하여 권한을 부여할 수 있습니다.



오류: Ubuntu 설치 명령을 사용하여 빌드 시 '죄송합니다. 터미널이 전혀 요청되지 않았습니다. 입력을 가져올 수 없습니다'

문제: GPU 컨테이너 권한 빌드를 실행하는 경우 이 절차에 따라 NVIDIA 컨테이너 툴킷을 설치할 수 있습니다. 최신 CodeBuild 이미지 릴리스에서 CodeBuild는 최신 `amazonlinux` 및 `ubuntu` 큐레이션 이미지에 `nvidia-container-toolkit`을 사용하여 Docker를 사전 설치하고 구성합니다. 이 절차를 따르면 Ubuntu 설치 명령을 사용한 빌드가 실패하고 다음 오류가 발생합니다.

```
Running command curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey | gpg --
dearmor --no-tty -o /usr/share/keyrings/nvidia-container-toolkit-keyring.gpg
gpg: Sorry, no terminal at all requested - can't get input
curl: (23) Failed writing body
```

가능한 원인: gpg 키가 이미 동일한 위치에 있습니다.

권장 솔루션: `nvidia-container-toolkit`이 이미지에 이미 설치되어 있습니다. 이 오류가 표시되면 buildspec에서 Docker 설치 및 재시작 프로세스를 건너뛸 수 있습니다.

Secrets Manager OAuth 사용 후 인증 오류와 함께 Bitbucket 빌드 실패

문제: Bitbucket을 소스 공급자로 사용하는 빌드는 소스 자격 증명 오류와 함께 실패합니다. Bitbucket 연결은와 AWS Secrets Manager 함께 OAuth를 스토리지 서비스로 사용하여 구성되었습니다.

가능한 원인: Atlassian은 Bitbucket OAuth에 대해 일회용 교체 새로 고침 토큰을 적용합니다. 새로 고침 토큰을 사용할 때마다 Bitbucket은 토큰을 무효화하고 새 토큰을 반환합니다. 자세한 내용은 Atlassian 개발자 웹 사이트의 Bitbucket OAuth 일회용 새로 고침 토큰(CHANGE-3052)을 참조하세요.

권장 솔루션: CodeBuild 서비스 역할에 `secretsmanager:PutSecretValue` 권한을 추가합니다. 필요한 IAM 권한에 대한 자세한 내용은 섹션을 참조하세요Secrets Manager 저장 자격 증명에 필요한 작업.

에 대한 할당량 AWS CodeBuild

다음 표에는 현재 할당량이 나열되어 있습니다 AWS CodeBuild. 이러한 할당량은 달리 지정되지 않는 한 각 AWS 계정에서 지원되는 각 AWS 리전에 대해 적용됩니다.

Service Quotas

다음은 AWS CodeBuild 서비스의 기본 할당량입니다.

이름	기본값	조정 가능	설명
프로젝트별 관련 태그	지원되는 각 리전: 50	아니요	빌드 프로젝트와 연결할 수 있는 최대 태그 수
빌드 프로젝트	지원되는 각 리전: 5,000	예	최대 빌드 프로젝트 수
빌드 제한 시간(분)	지원되는 각 리전: 2,160	아니요	최대 빌드 제한 시간(분)
빌드 정보에 대한 동시 요청	지원되는 각 리전: 100	아니요	AWS CLI 또는 AWS SDK를 사용하여 한 번에 정보를 요청할 수 있는 최대 빌드 수입니다.
빌드 프로젝트 정보에 대한 동시 요청	지원되는 각 리전: 100	아니요	AWS CLI 또는 AWS SDK를 사용하여 한 번에 정보를 요청할 수 있는 최대 빌드 프로젝트 수입니다.

이름	기본값	조정 가능	설명
ARM Lambda/10GB 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	ARM Lambda/10GB 환경에서 동시에 실행되는 빌드의 최대 수
ARM Lambda/1GB 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	ARM Lambda/1GB 환경에서 동시에 실행되는 빌드의 최대 수
ARM Lambda/2GB 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	ARM Lambda/2GB 환경에서 동시에 실행되는 빌드의 최대 수
ARM Lambda/4GB 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	ARM Lambda/4GB 환경에서 동시에 실행되는 빌드의 최대 수
ARM Lambda/8GB 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	ARM Lambda/8GB 환경에서 동시에 실행되는 빌드의 최대 수
ARM/2XLarge 환경에서 동시에 실행되는 빌드	각 지원되는 지역: 1	예	ARM/2XLarge 환경에서 동시에 실행되는 빌드의 최대 수
ARM/Large 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	ARM/Large 환경에서 동시에 실행되는 빌드의 최대 수
ARM/Medium 환경에서 동시에 실행되는 빌드	각 지원되는 지역: 1	예	ARM/Medium 환경에서 동시에 실행되는 빌드의 최대 수

이름	기본값	조정 가능	설명
ARM/Small 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	ARM/Small 환경에서 동시에 실행되는 빌드의 최대 수
ARM/XLarge 환경에서 동시에 실행되는 빌드	각 지원되는 지역: 1	예	ARM/XLarge 환경에서 동시에 실행되는 빌드의 최대 수
Linux GPU Large 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 0	예	Linux GPU/Large 환경에서 동시에 실행되는 빌드의 최대 수
Linux GPU Small 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 0	예	Linux GPU/Small 환경에서 동시에 실행되는 빌드의 최대 수
Linux Lambda/10GB 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	Linux Lambda/10GB 환경에서 동시에 실행되는 빌드의 최대 수
Linux Lambda/1GB 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	Linux Lambda/1GB 환경에서 동시에 실행되는 빌드의 최대 수
Linux Lambda/2GB 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	Linux Lambda/2GB 환경에서 동시에 실행되는 빌드의 최대 수
Linux Lambda/4GB 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	Linux Lambda/4GB 환경에서 동시에 실행되는 빌드의 최대 수

이름	기본값	조정 가능	설명
Linux Lambda/8GB 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	Linux Lambda/8GB 환경에서 동시에 실행되는 빌드의 최대 수
Linux/2XLarge 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 0	예	Linux/2XLarge 환경에서 동시에 실행되는 빌드의 최대 수
Linux/Large 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	Linux/Large 환경에서 동시에 실행되는 빌드의 최대 수
Linux/Medium 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	Linux/Medium 환경에서 동시에 실행되는 빌드의 최대 수
Linux/Small 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	Linux/Small 환경에서 동시에 실행되는 빌드의 최대 수
Linux/XLarge 환경 플릿에서 동시에 실행되는 빌드	각 지원되는 지역: 1	예	Linux/XLarge 환경에서 동시에 실행되는 빌드의 최대 수
Windows Server 2019/Large 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	Windows Server 2019/Large 환경에서 동시에 실행되는 빌드의 최대 수
Windows Server 2019/Medium 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	Windows Server 2019/Medium 환경에서 동시에 실행되는 빌드의 최대 수

이름	기본값	조정 가능	설명
Windows Server 2022/2XLarge 환경에서 동시에 실행되는 빌드	각 지원되는 지역: 1	예	Windows Server 2022/2XLarge 환경에서 동시에 실행되는 빌드의 최대 수
Windows Server 2022/Large 환경에서 동시에 실행되는 빌드	각 지원되는 지역: 1	예	Windows Server 2022/Large 환경에서 동시에 실행되는 빌드의 최대 수
Windows Server 2022/Medium 환경에서 동시에 실행되는 빌드	각 지원되는 지역: 1	예	Windows Server 2022/Medium 환경에서 동시에 실행되는 빌드의 최대 수
Windows Server 2022/XLarge 환경에서 동시에 실행되는 빌드	각 지원되는 지역: 1	예	Windows Server 2022/XLarge 환경에서 동시에 실행되는 빌드의 최대 수
Windows/Large 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	Windows/Large 환경에서 동시에 실행되는 빌드의 최대 수
Windows/Medium 환경에서 동시에 실행되는 빌드	지원되는 각 리전: 1	예	Windows/Medium 환경에서 동시에 실행되는 빌드의 최대 수
빌드 제한 시간의 최소 기간(분)	지원되는 각 리전: 5	아니요	최소 빌드 제한 시간(분)
VPC 구성의 보안 그룹	지원되는 각 리전: 5	아니요	VPC 구성에 사용할 수 있는 보안 그룹

이름	기본값	조정 가능	설명
VPC 구성의 서브넷	지원되는 각 리전: 16	아니요	VPC 구성에 사용할 수 있는 서브넷

> **Note**
>
> 내부 지표에 따라 동시 실행 빌드의 기본 할당량이 결정됩니다.

최대 동시 실행 빌드 수 할당량은 컴퓨팅 유형에 따라 다릅니다. 일부 플랫폼 및 컴퓨팅 유형의 경우 기본값은 20입니다. 더 높은 동시 빌드 할당량을 요청하거나 "계정에 X개 이상의 활성 빌드를 가질 수 없음" 오류가 발생하는 경우 위 링크를 사용하여 요청을 수행하세요. 요금에 대한 자세한 내용은 AWS CodeBuild 요금을 참조하세요.

기타 제한

빌드 프로젝트

Resource	기본값
빌드 프로젝트 설명에 허용되는 문자	임의
빌드 프로젝트 이름에 허용되는 문자	글자(A-Z 및 a-z), 숫자(0-9) 및 특수 문자(- 및 _)
빌드 프로젝트 이름의 길이	2~150자(경계값 포함)
빌드 프로젝트 설명의 최대 길이	255자
프로젝트에 추가할 수 있는 최대 보고서 수	5

Resource	기본값
관련된 모든 빌드의 빌드 제한 시간으로 빌드 프로젝트에 지정할 수 있는 시간(분)	5~2160(36시간)

빌드

Resource	기본값
빌드 기록이 유지되는 최대 시간	1년
단일 빌드의 빌드 제한 시간으로 지정할 수 있는 시간(분)	5~2160(36시간)

컴퓨팅 플릿

Resource	기본값
동시 컴퓨팅 플릿 수	10
ARM/Small 환경 플릿에서 동시에 실행되는 인스턴스	1
ARM/Large 환경 플릿에서 동시에 실행되는 인스턴스	1
Linux/Small 환경 플릿에서 동시에 실행되는 인스턴스	1
Linux/Medium 환경 플릿에서 동시에 실행되는 인스턴스	1
Linux/Large 환경 플릿에서 동시에 실행되는 인스턴스	1
Linux/XLarge 환경 플릿에서 동시에 실행되는 인스턴스	1

Resource	기본값
Linux/2XLarge 환경 플릿에서 동시에 실행되는 인스턴스	0
Linux GPU/Small 환경 플릿에서 동시에 실행되는 인스턴스	0
Linux GPU/Large 환경 플릿에서 동시에 실행되는 인스턴스	0
Windows Server 2019/Medium 환경 플릿에서 동시에 실행되는 인스턴스	1
Windows Server 2019/Large 환경 플릿에서 동시에 실행되는 인스턴스	1
Windows Server 2022/Medium 환경 플릿에서 동시에 실행되는 인스턴스	1
Windows Server 2022/Large 환경 플릿에서 동시에 실행되는 인스턴스	1
Mac ARM/Medium 환경 플릿에서 동시에 실행되는 인스턴스	1
Mac ARM/Large 환경 플릿에서 동시에 실행되는 인스턴스	1

Reports

Resource	기본값
테스트 보고서를 만든 후 사용할 수 있는 최대 기간	30일
테스트 사례 메시지의 최대 길이	5,000자

Resource	기본값
테스트 사례 이름의 최대 길이	1,000자
AWS 계정당 최대 보고서 그룹 수	5,000
보고서당 최대 테스트 케이스 수	500

Tags

태그 제한은 CodeBuild 빌드 프로젝트 및 CodeBuild 보고서 그룹 리소스의 태그에 적용됩니다.

Resource	기본값
리소스 태그 키 이름	유니코드 문자, 숫자, 공백 및 UTF-8 형식의 허용되는 문자 조합(1 - 127 문자). 허용되는 문자는 `+ - = . _ : / @`입니다. 태그 키 이름은 고유해야 하며 각 키는 하나의 값만 가질 수 있습니다. 태그 키 이름에는 다음 사항이 적용됩니다. • `aws:`로 시작할 수 없음 • 공백으로만 이루어짐 • 공백으로 끝남 • 이모티콘 또는 다음 문자를 포함할 수 없음: `? ^ * [ \ ~ ! # $ % & * ( ) > < | " ' ` [ ] { } ;`
리소스 태그 값	유니코드 문자, 숫자, 공백 및 UTF-8 형식의 허용되는 문자 조합(0 - 255 문자). 허용되는 문자는 `+ - = . _ : / @`입니다. 키는 하나의 값만 가질 수 있지만 많은 키가 동일한 값을 가질 수 있습니다. 태그 키 값에는 이모티콘 또는 다음 문자를 포함할 수 없습니다.

Resource	기본값
	? ^ * [\ ~ ! # $ % & * ( ) > < | " ' ` [ ] { } ;

AWS CodeBuild 사용 설명서 문서 기록

다음 표에서는의 마지막 릴리스 이후 설명서에 대한 중요한 변경 사항을 설명합니다 AWS CodeBuild. 이 설명서에 대한 업데이트 알림을 받으려면 RSS 피드를 구독하면 됩니다.

- 최신 API 버전: 2016-10-06

변경 사항	설명	날짜
업데이트된 콘텐츠: CodeBuild 표준 이미지에 사용 가능한 런타임	x86_64 표준 이미지에 Node.js 26 런타임 및 Rust 1.97이 추가되었습니다.	2026년 9월 22일
업데이트된 콘텐츠: CodeBuild 표준 이미지에 사용 가능한 런타임	지원되는 런타임에 Amazon Linux 2023 AArch64 표준:4.0 이미지가 추가되었습니다.	2026년 9월 22일
업데이트된 콘텐츠: webhook 필터 정규 표현식 구문	웹후크 필터 패턴은 RE2 정규식 구문을 사용하고 지원되지 않는 구문을 사용함을 명확히 했습니다.	2026년 8월 3일
업데이트 내용: Lambda 컴퓨팅 이미지	.NET 10, , Go 1.25, 및에 대한 Lambda 컴퓨팅 이미지가 추가되었습니다Corretto 25Node.js 24Python 3.14.	2026년 7월 29일
업데이트 내용: Bitbucket 앱 암호, API 토큰 또는 액세스 토큰	더 이상 사용되지 않는 앱 암호를 대체하는 Bitbucket API 토큰에 대한 지원을 추가합니다.	2026년 7월 19일
사용자 지정 GitHub Enterprise 역할 매핑 및 역할 이름 유지 수정	권한 계층 구조를 기반으로 GitHub Enterprise 사용자 지정 리포지토리 역할을 표준 역할에 매핑하는 방법에 대한 설명서가 추가되었습니다. 기본값 및 예제GITHUB_MA	2026년 4월 13일

	INTAIN 에서 유지 수준 역할 이름을 로 수정했습니다.	
업데이트된 콘텐츠: CodeBuild 표준 이미지에 사용 가능한 런타임	사용 가능한 런타임 버전을 업데이트하고 새 이미지를 추가했습니다.	2026년 4월 7일
업데이트 내용: EC2 컴퓨팅 이미지	Amazon Linux 2023 x86_64 표준:6.0 및 Ubuntu 24.04 표준:8.0 컴퓨팅 이미지가 추가되었습니다.	2026년 4월 7일
업데이트된 콘텐츠: 런타임 버전 선택 동작	이전 기본 버전이 수명 종료(EOL)에 도달하면 기본 런타임 버전이 변경될 수 있다는 지침이 추가되었습니다.	2026년 3월 25일
업데이트된 콘텐츠: CodeBuild 표준 이미지에 사용 가능한 런타임	사용 가능한 런타임 버전을 업데이트하고 새 이미지를 추가했습니다.	2026년 3월 25일
에 대한 업데이트된 콘텐츠: AWS 관리형(미리 정의된) 정책 AWS CodeBuild	Session Manager 대화형 빌드 디버깅에 필요한 ssmmessages:OpenDataChannel 권한을 포함하도록 AWSCodeBuildAdminAccess 및 AWSCodeBuildDeveloperAccess 정책이 업데이트되었습니다.	2025년 12월 1일
풀 요청 설명 승인	CodeBuild는 풀 요청에 의해 트리거되는 빌드에 대한 추가 제어를 제공하는 풀 요청 빌드 정책을 지원합니다.	2025년 8월 7일

에 대한 업데이트된 콘텐츠: AWS 관리형(미리 정의된) 정책 AWS CodeBuild	AWSCodeBuildAdminAccess, AWSCodeBuildDeveloperAccess 및 AWSCodeBuildReadOnlyAccess 정책이 업데이트되었습니다. 원본 리소스 `arn:aws:codeconnections:*:*:connection/*` 이 `arn:aws:codeconnections:*:*:*` 로 업데이트되었습니다.	2025년 6월 10일
CodeBuild 조건 키에 대한 새로운 참조	CodeBuild 조건 키 사용에 대한 예제가 포함된 새 참조 페이지가 추가되었습니다. AWS CodeBuild 조건 키를 참조하세요.	2025년 5월 15일
새 콘텐츠: CodeBuild용 Docker 이미지 빌드 서버 샘플	CodeBuild는 이제 Docker 빌드를 관리형 이미지 빌드 서버로 오프로드하는 기능을 지원합니다.	2025년 5월 15일
새 컴퓨팅 유형: `CUSTOM_INSTANCE_TYPE`	이제 CodeBuild에서 `CUSTOM_INSTANCE_TYPE` 을 사용하여 특정 인스턴스 유형으로 예약 용량 플릿을 생성할 수 있습니다.	2025년 4월 23일
CodeBuild 샌드박스에 대한 새로운 지원	새 CodeBuild 샌드박스 사용에 대한 정보가 추가되었습니다. CodeBuild 샌드박스를 사용한 빌드 디버그를 참조하세요.	2025년 4월 7일
새로운 Windows 환경 유형	CodeBuild는 이제 Windows XL 및 2XL 환경 유형을 지원합니다. 자세한 내용은 빌드 환경 컴퓨팅 유형을 참조하십시오.	2025년 3월 31일

업데이트된 Amazon S3 캐싱	CodeBuild는 이제 Amazon S3 캐싱에 대한 새로운 캐싱 동작을 지원합니다.	2025년 3월 28일
새 콘텐츠: GitHub Actions 러너 구성 옵션	CodeBuild는 이제 엔터프라이즈 수준에서 등록을 위해 `CODEBUILD_CONFIG_GITHUB_ACTIONS_ENTERPRISE_REGISTRATION_NAME` 을 지원합니다.	2025년 3월 11일
새 콘텐츠: 새 웹후크 필터 유형 추가	새 웹후크 필터 유형 (`ORGANIZATION_NAME`)에 대한 지원을 추가합니다.	2025년 3월 11일
새 콘텐츠: S3 인증서 스토리지가 포함된 Fastlane을 사용한 Apple 코드 서명 자습서	인증서 스토리지용 S3를 사용하여 CodeBuild에서 Fastlane을 사용한 새 Apple 코드 서명 자습서 추가	2025년 2월 5일
새 콘텐츠: GitHub 인증서 스토리지를 사용하는 Fastlane을 사용한 Apple 코드 서명 자습서	인증서 스토리지용 GitHub를 사용하여 CodeBuild에서 Fastlane을 사용한 새 Apple 코드 서명 자습서 추가	2025년 2월 5일
새 콘텐츠: Buildkite 러너	Buildkite 러너에 대한 새 콘텐츠 추가	2025년 1월 31일
새 콘텐츠: Buildkite 수동 웹후크	Buildkite 수동 웹후크에 대한 지원을 추가합니다.	2025년 1월 31일
새 콘텐츠: 배치 빌드 buildspec 참조	예약 용량 플릿 및 Lambda 환경에서 배치 빌드에 대한 지원을 추가합니다.	2025년 1월 8일
새 콘텐츠: 배치 빌드에서 병렬 테스트 실행	배치 빌드에서 병렬 테스트를 위한 새 콘텐츠를 추가합니다.	2025년 1월 2일

새 콘텐츠: 재시도가 자동으로 빌드됨	CodeBuild는 이제 웹후크 빌드에 대한 자동 재시도를 지원합니다.	2024년 12월 18일
새 콘텐츠: 자체 호스팅 러너에 대한 프라이빗 레지스트리 자격 증명 구성	프라이빗이 아닌 레지스트리의 사용자 지정 이미지를 사용할 때 레지스트리 자격 증명 설정에 대한 지원을 추가합니다.	2024년 12월 13일
새 콘텐츠: GitHub Actions 러너 구성 옵션	CodeBuild GitHub Actions 자체 호스팅 러너를 사용하면 이제 조직 수준에서 러너를 등록하고 특정 러너 그룹 ID를 구성할 수 있습니다.	2024년 12월 12일
새 콘텐츠: 실패 시 속성 RETRY 추가	이제 CodeBuild를 사용하여 buildspec에서 실패 시 속성을 RETRY로 구성할 수 있습니다.	2024년 12월 12일
새 콘텐츠: GitLab 수동 웹후크	GitLab 수동 웹후크에 대한 지원을 추가합니다.	2024년 12월 11일
업데이트된 콘텐츠: 업데이트된 별칭	Linux 기반 표준 런타임 이미지의 별칭을 업데이트합니다.	2024년 11월 22일
업데이트된 콘텐츠: CodeBuild 호스팅 GitLab 러너에서 지원되는 레이블 재정의	GitLab 러너의 사용자 지정 이미지 레이블 재정의에 대한 지원을 추가합니다.	2024년 11월 22일
업데이트된 콘텐츠: CodeBuild 호스팅 GitHub Actions 러너에서 지원되는 레이블 재정의	GitHub Actions 러너의 사용자 지정 이미지 레이블 재정의에 대한 지원을 추가합니다.	2024년 11월 22일

에 대한 업데이트된 콘텐츠: AWS 관리형(미리 정의된) 정책 AWS CodeBuild	AWSCodeBuildAdminAccess, AWSCodeBuildDeveloperAccess 및 AWSCodeBuildReadOnlyAccess 정책이 업데이트되었습니다. 원본 리소스 `arn:aws:codebuild:*` 이 `arn:aws:codebuild:*:*:project/*` 로 업데이트되었습니다.	2024년 11월 15일
업데이트된 콘텐츠: 예약 용량	예약 용량 플릿은 이제 컨테이너가 아닌 빌드인 ARM EC2, Linux EC2 및 Windows EC2를 지원합니다.	2024년 11월 12일
업데이트된 콘텐츠: 예약 용량	예약 용량 플릿은 이제 속성 기반 컴퓨팅을 지원합니다.	2024년 11월 6일
새 콘텐츠: 재시도가 자동으로 빌드됨	이제 CodeBuild를 사용하여 빌드에 대해 자동 재시도를 활성화할 수 있습니다.	2024년 10월 25일
새 콘텐츠: 예약 용량 플릿에 대해 관리형 프록시 서버에서 CodeBuild 실행	예약 용량 플릿에 대한 프록시 구성 지원을 추가합니다.	2024년 10월 15일
새 콘텐츠: 자체 관리형 GitLab 실행기	자체 관리형 GitLab 실행기를 위한 새 콘텐츠 추가	2024년 9월 17일
새 콘텐츠: GitLab 그룹 웹후크	GitLab 그룹 웹후크에 대한 지원을 추가합니다.	2024년 9월 17일
새 콘텐츠: INSTALL, PRE_BUILD 및 POST_BUILD 단계에서 buildspec 명령 실행	`-with-buildspec` 에 대한 지원이 추가되었습니다.	2024년 8월 20일

업데이트된 콘텐츠: 예약 용량	예약 용량 플릿은 이제 macOS를 지원합니다.	2024년 8월 19일
새 콘텐츠: GitHub 앱 연결	GitHub 앱 연결에 대한 지원을 추가합니다.	2024년 8월 14일
새 콘텐츠: Bitbucket 앱 연결	Bitbucket 앱 연결에 대한 지원을 추가합니다.	2024년 8월 14일
새 콘텐츠: CodeBuild의 다중 액세스 토큰	AWS Secrets Manager 또는 AWS CodeConnections 연결을 통해 보안 암호에서 타사 공급자에게 액세스 토큰 소싱에 대한 지원을 추가합니다.	2024년 8월 14일
업데이트된 콘텐츠: 예약 용량	예약 용량 플릿은 이제 ARM Medium, ARM XLarge 및 ARM 2XLarge 컴퓨팅 유형을 지원합니다.	2024년 8월 5일
업데이트된 콘텐츠: 예약 용량	CodeBuild는 이제 Windows에서 예약 용량 플릿에 대한 VPC 연결을 지원합니다.	2024년 8월 1일
새 ARM 컴퓨팅 유형	CodeBuild는 이제 ARM Medium, ARM XLarge 및 ARM 2XLarge 컴퓨팅 유형을 지원합니다. 자세한 내용은 빌드 환경 컴퓨팅 유형을 참조하십시오.	2024년 7월 10일
업데이트된 콘텐츠: SHA 서명	x86_64 및 ARM에 대한 SHA(Secure Hash Algorithm) 서명을 업데이트합니다.	2024년 6월 19일
새 콘텐츠: GitHub 글로벌 및 조직 웹후크	GitHub 글로벌 및 조직 웹후크에 대한 지원을 추가합니다.	2024년 6월 17일

새 콘텐츠: 새 웹후크 필터 유형 추가	새 웹후크 필터 유형(`REPOSITORY_NAME`)에 대한 지원을 추가합니다.	2024년 6월 17일
업데이트된 디스크 공간	이제 `ARM Small` 및 `ARM Large` 컴퓨팅 유형에 디스크 공간이 늘어났습니다.	2024년 6월 4일
새 콘텐츠: GitHub 수동 웹후크	GitHub 수동 웹후크에 대한 지원을 추가합니다.	2024년 5월 23일
업데이트된 콘텐츠: 예약 용량	CodeBuild는 이제 Amazon Linux에서 예약 용량 플릿에 대한 VPC 연결을 지원합니다.	2024년 5월 15일
업데이트 내용: Lambda 컴퓨팅 이미지	.NET 8에 대한 Lambda 지원 추가(`al-lambda/aarch64/dotnet8` 및 `al-lambda/x86_64/dotnet8`)	2024년 5월 8일
업데이트된 할당량: 빌드 제한 시간	최대 빌드 제한 시간 할당량을 2160분(36시간)으로 업데이트합니다.	2024년 5월 1일
에 대한 업데이트된 콘텐츠: AWS 관리형(미리 정의된) 정책 AWS CodeBuild	AWSCodeBuildAdminAccess, AWSCodeBuildDeveloperAccess 및 AWSCodeBuildReadOnlyAccess 정책이 AWS CodeConnections 리브랜딩을 반영하도록 업데이트되었습니다.	2024년 4월 30일
새 콘텐츠: Bitbucket 앱 암호, API 토큰 또는 액세스 토큰	Bitbucket 액세스 토큰에 대한 지원을 추가합니다.	2024년 4월 11일
새 콘텐츠: CodeBuild에서 보고서 자동 검색	CodeBuild는 이제 보고서 자동 검색을 지원합니다.	2024년 4월 4일

새 콘텐츠: 자체 호스팅 GitHub Action 실행기	자체 호스팅 GitHub Action 실행기를 위한 새 콘텐츠 추가	2024년 4월 2일
새 콘텐츠: GitLab 연결	GitLab 및 GitHub Self Managed 연결에 대한 지원을 추가합니다.	2024년 3월 25일
새 콘텐츠: 새 웹후크 이벤트 및 필터 유형 추가	새 웹후크 이벤트(RELEASED 및 PRERELEASED) 및 필터 유형(TAG_NAME 및 RELEASE_NAME)에 대한 지원을 추가합니다.	2024년 3월 15일
새 콘텐츠: 새 웹후크 이벤트 추가: PULL_REQUEST_CLOSED	새 웹후크 이벤트에 대한 지원 추가: PULL_REQUEST_CLOSED .	2024년 2월 20일
업데이트 내용: CodeBuild가 제공하는 도커 이미지	Windows Server 2019에 대한 지원 추가(windows-base:2019-3.0)	2024년 2월 7일
업데이트 내용: CodeBuild가 제공하는 도커 이미지	Amazon Linux 2023의 새 런타임에 대한 지원 추가(al2/aarch64/standard/3.0)	2024년 1월 29일
새 콘텐츠: 예약 용량	CodeBuild는 이제 CodeBuild에서 예약 용량 플릿을 지원합니다.	2024년 1월 18일
새 컴퓨팅 유형	CodeBuild는 이제 Linux XLarge 컴퓨팅 유형을 지원합니다. 자세한 내용은 빌드 환경 컴퓨팅 유형을 참조하십시오.	2024년 1월 8일

업데이트 내용: CodeBuild가 제공하는 도커 이미지	Amazon Linux 2(al2/standard/5.0) 및 Ubuntu(ubuntu/standard/7.0)의 새 런타임에 대한 지원 추가	2023년 12월 14일
업데이트 내용: CodeBuild가 제공하는 도커 이미지	새로운 Lambda 컴퓨팅 이미지에 대한 지원 추가	2023년 12월 8일
새 콘텐츠: AWS Lambda 컴퓨팅	AWS Lambda 컴퓨팅에 대한 새 콘텐츠 추가	2023년 11월 6일
업데이트 내용: CodeBuild가 제공하는 도커 이미지	Amazon Linux 2(al2/standard/5.0)에 대한 지원 추가	2023년 5월 17일
CodeBuild의 관리형 정책에 대한 변경 사항	이제 CodeBuild의 AWS 관리형 정책 업데이트에 대한 세부 정보를 사용할 수 있습니다. 자세한 내용은 AWS 관리형 정책에 대한 CodeBuild 업데이트를 참조하세요.	2023년 5월 16일
업데이트 내용: CodeBuild가 제공하는 도커 이미지	Amazon Linux 2(al2/standard/3.0)에 대한 지원 제거, Amazon Linux 2(al2/standard/corretto8) 및 Amazon Linux 2(al2/standard/corretto11)에 대한 지원 추가	2023년 5월 9일
업데이트 내용: CodeBuild가 제공하는 도커 이미지	Ubuntu 22.04(ubuntu/standard/7.0)에 대한 지원 추가	2023년 4월 13일

업데이트 내용: CodeBuild가 제공하는 도커 이미지	Ubuntu 18.04(`ubuntu/standard/4.0`)) 및 Amazon Linux 2(`al2/aarch64/standard/1.0`)에 대한 지원 제거	2023년 3월 31일
업데이트된 콘텐츠: VPC 제한 제거	다음 제한 사항 제거: VPC에서 작동하도록 CodeBuild를 구성하는 경우 로컬 캐싱은 지원되지 않습니다. 2022년 2월 28일부터 각 빌드에 새 Amazon EC2 인스턴스가 사용되므로 VPC 빌드에 시간이 더 오래 걸립니다.	2023년 3월 1일
업데이트 내용: CodeBuild가 제공하는 도커 이미지	Ubuntu 18.04(`ubuntu/standard/3.0`) 및 Amazon Linux 2(`al2/standard/2.0`)에 대한 지원 제거	2022년 6월 30일
Amazon ECR 샘플: 이미지 액세스 제한	CodeBuild 보안 인증을 사용하여 Amazon ECR 이미지를 가져오는 경우 특정 CodeBuild 프로젝트에 대한 이미지 액세스를 제한할 수 있습니다. 자세한 내용은 Amazon ECR 샘플을 참조하세요.	2022년 3월 10일
리전 지원 추가	이제 아시아 태평양(서울), 캐나다(중부), 유럽(런던), 유럽(파리) 리전에서 `ARM_CONTAINER` 컴퓨팅 유형이 지원됩니다. 자세한 내용은 빌드 환경 컴퓨팅 유형을 참조하십시오.	2022년 3월 10일

새 VPC 제한 사항	VPC에서 작동하도록 CodeBuild를 구성하는 경우 로컬 캐싱은 지원되지 않습니다. 2022년 2월 28일부터 각 빌드에 새 Amazon EC2 인스턴스가 사용되므로 VPC 빌드에 시간이 더 오래 걸립니다.	2022년 2월 25일
배치 보고서 모드	CodeBuild를 사용하면 배치 빌드 상태를 프로젝트의 소스 공급자에게 보내는 방법을 선택할 수 있습니다. 자세한 내용은 배치 보고서 모드를 참조하세요.	2021년 10월 4일
새 컴퓨팅 유형	CodeBuild는 이제 소형 ARM 컴퓨팅 유형을 지원합니다. 자세한 내용은 빌드 환경 컴퓨팅 유형을 참조하십시오.	2021년 9월 13일
퍼블릭 빌드 프로젝트	이제 CodeBuild를 사용하면 AWS 계정에 액세스할 필요 없이 빌드 프로젝트의 빌드 결과를 일반에 제공할 수 있습니다. 자세한 내용은 퍼블릭 빌드 프로젝트를 참조하세요.	2021년 8월 11일
배치 빌드를 위한 세션 디버깅	CodeBuild는 이제 배치 빌드에 대한 세션 디버깅을 지원합니다. 자세한 내용은 build-graph 및 build-list를 참조하세요.	2021년 3월 3일

프로젝트 수준 동시 빌드 제한	이제 CodeBuild를 사용하여 빌드 프로젝트의 동시 빌드 수를 제한할 수 있습니다. 자세한 내용은 프로젝트 구성 및 concurrentBuildLimit를 참조하세요.	2021년 2월 16일
새 buildspec 속성: s3-prefix	CodeBuild는 이제 Amazon S3에 업로드되는 아티팩트의 경로 접두사를 지정할 수 있는 아티팩트에 대한 `s3-prefix` buildspec 속성을 제공합니다. 자세한 내용은 s3-prefix를 참조하세요.	2021년 2월 9일
새 buildspec 속성: on-failure	이제 CodeBuild는 빌드 단계가 실패할 때 발생하는 상황을 확인할 수 있는 빌드 단계에 대한 `on-failure` buildspec 속성을 제공합니다. 자세한 내용은 on-failure를 참조하세요.	2021년 2월 9일
새 buildspec 속성: exclude-paths	CodeBuild는 이제 빌드 아티팩트에서 경로를 제외할 수 있는 아티팩트에 대한 `exclude-paths` buildspec 속성을 제공합니다. 자세한 내용은 exclude-paths를 참조하세요.	2021년 2월 9일
새 buildspec 속성: enable-symlinks	CodeBuild는 이제 ZIP 아티팩트에 심볼릭 링크를 보존할 수 있는 아티팩트에 대한 `enable-symlinks` buildspec 속성을 제공합니다. 자세한 내용은 enable-symlinks를 참조하세요.	2021년 2월 9일

Buildspec 아티팩트 이름 개선	이제 CodeBuild에서 `artifacts/name` 속성에 경로 정보를 포함할 수 있습니다. 자세한 내용은 name을 참조하세요.	2021년 2월 9일
코드 범위 보고	CodeBuild는 이제 코드 범위 보고서를 제공합니다. 자세한 내용은 코드 범위 보고서를 참조하세요.	2020년 7월 30일
배치 빌드	CodeBuild는 이제 프로젝트의 동시 및 조정된 빌드 실행을 지원합니다. 자세한 내용은 CodeBuild의 배치 빌드를 참조하세요.	2020년 7월 30일
Windows Server 2019 이미지	CodeBuild는 이제 Windows Server Core 2019 빌드 이미지를 제공합니다. 자세한 내용은 CodeBuild에서 제공하는 도커 이미지를 참조하세요.	2020년 7월 20일
세션 관리자	CodeBuild를 사용하면 실행 중인 빌드를 일시 중지한 다음 AWS Systems Manager Session Manager를 사용하여 빌드 컨테이너에 연결하고 컨테이너의 상태를 볼 수 있습니다. 자세한 내용은 세션 관리자를 참조하세요.	2020년 7월 20일
업데이트된 주제	CodeBuild에서는 이제 빌드 환경에서 사용할 쉘을 buildspec 파일에 지정할 수 있습니다. 자세한 내용은 빌드 사양 참조를 참조하세요.	2020년 6월 25일

테스트 프레임워크를 사용하여 테스트 보고	여러 테스트 프레임워크를 사용하여 CodeBuild 테스트 보고서를 생성하는 방법을 설명하는 여러 주제가 추가되었습니다. 자세한 내용은 테스트 프레임워크를 사용하여 테스트 보고를 참조하십시오.	2020년 5월 29일
업데이트된 주제	이제 CodeBuild에서 보고서 그룹에 태그를 추가할 수 있습니다. 자세한 내용은 ReportGroup을 참조하십시오.	2020년 5월 21일
테스트 보고 지원	테스트 보고에 대한 CodeBuild 지원이 이제 일반 공급됩니다.	2020년 5월 21일
업데이트된 주제	이제 CodeBuild는 헤드 커밋 메시지가 지정된 표현식과 일치할 때만 빌드를 트리거하는 GitHub 및 Bitbucket에 대한 webhook 필터 생성을 지원합니다. 자세한 내용은 GitHub pull 요청 및 webhook 필터 샘플 및 Bitbucket pull 요청 및 webhook 필터 샘플을 참조하십시오.	2020년 5월 6일
새로운 주제	CodeBuild에서는 현재 빌드 프로젝트 및 보고서 그룹 리소스 공유를 지원합니다. 자세한 내용은 공유 프로젝트 작업 및 공유 보고서 그룹 작업을 참조하십시오.	2019년 12월 13일

신규 및 업데이트된 주제	CodeBuild에서는 현재 빌드 프로젝트를 실행하는 동안 테스트 보고를 지원합니다. 자세한 내용은 테스트 보고 작업, 테스트 보고서 생성 및 AWS CLI 샘플을 사용하여 테스트 보고서 생성을 참조하세요.	2019년 11월 25일
업데이트된 주제	CodeBuild에서는 현재 Linux GPU 및 Arm 환경 유형과 `2xlarge` 컴퓨팅 유형을 지원합니다. 자세한 내용은 빌드 환경 컴퓨팅 유형을 참조하십시오.	2019년 11월 19일
업데이트된 주제	CodeBuild는 이제 모든 빌드의 빌드 번호, 환경 변수 내보내기 및 AWS Secrets Manager 통합을 지원합니다. 자세한 내용은 내보낸 변수 및 Buildspec 구문의 Secrets Manager를 참조하세요.	2019년 11월 6일
새 주제	CodeBuild에서는 이제 알림 규칙을 지원합니다. 알림 규칙을 사용하여 사용자에게 빌드 프로젝트의 중요한 변경 사항을 알릴 수 있습니다. 자세한 내용은 알림 규칙 생성을 참조하세요.	2019년 11월 5일
업데이트된 주제	CodeBuild는 이제 Android 버전 29 및 Go 버전 1.13 런타임을 지원합니다. 자세한 내용은 CodeBuild가 제공하는 도커 이미지 및 Buildspec 구문을 참조하세요.	2019년 9월 10일

업데이트된 주제	이제 프로젝트를 생성할 때 Amazon Linux 2(AL2) 관리형 이미지를 선택할 수 있습니다. 자세한 내용은 CodeBuild가 제공하는 도커 이미지 및 CodeBuild용 buildspec 파일 샘플의 런타임 버전을 참조하세요.	2019년 8월 16일
업데이트된 주제	이제 프로젝트를 생성할 때 S3 로그 암호화를 비활성화할 수 있고, Git 기반 소스 리포지토리를 사용할 경우 Git 하위 모듈을 포함할 수 있습니다. 자세한 내용은 CodeBuild에서 빌드 프로젝트 생성을 참조하세요.	2019년 3월 8일
새 주제	이제 CodeBuild에서 로컬 캐싱을 지원합니다. 빌드를 생성할 때 로컬 캐싱을 네 가지 모드 중 하나 이상 지정할 수 있습니다. 자세한 내용은 CodeBuild의 빌드 캐싱을 참조하세요.	2019년 2월 21일
새로운 주제	이제 CodeBuild가 빌드를 트리거하는 이벤트를 지정하기 위한 Webhook 필터 그룹을 지원합니다. 자세한 내용은 GitHub Webhook 이벤트 필터링 및 Bitbucket Webhook 이벤트 필터링을 참조하십시오.	2019년 2월 8일

새 주제	이제 CodeBuild 사용자 설명서가 프록시 서버에서 CodeBuild를 사용하는 방법을 보여 줍니다. 자세한 내용은 프록시 서버에서 CodeBuild 사용을 참조하세요.	2019년 2월 4일
업데이트된 주제	CodeBuild는 이제 다른 AWS 계정에 있는 Amazon ECR 이미지 사용을 지원합니다. 여러 주제가 CodeBuild용 Amazon ECR 샘플, 빌드 프로젝트 생성 및 CodeBuild 서비스 역할 생성을 포함한 이 변경 사항을 반영하도록 업데이트되었습니다.	2019년 1월 24일
프라이빗 Docker 레지스트리 지원	CodeBuild에서는 이제 런타임 환경으로서 프라이빗 레지스트리에 저장되는 도커 이미지를 사용하도록 지원합니다. 자세한 내용을 알아보려면 AWS Secrets Manager 샘플이 있는 프라이빗 레지스트리를 참조하세요.	2019년 1월 24일
업데이트된 주제	CodeBuild에서 이제 액세스 토큰을 사용한 GitHub(개인 액세스 토큰) 및 Bitbucket(앱 암호) 리포지토리 연결을 지원합니다. 자세한 내용은 빌드 프로젝트 생성(콘솔) 및 소소 공급자에 액세스 토큰 사용을 참조하십시오.	2018년 12월 6일

업데이트된 주제	CodeBuild에서 이제 빌드의 각 단계 지속 시간을 측정하는 새로운 빌드 지표를 지원합니다. 자세한 내용은 CodeBuild CloudWatch 지표를 참조하세요.	2018년 11월 15일
VPC 엔드포인트 정책 주제	CodeBuild용 Amazon VPC 엔드포인트는 이제 정책을 지원합니다. 자세한 내용은 CodeBuild에 대한 VPC 엔드포인트 정책 생성을 참조하세요.	2018년 11월 9일
콘텐츠 업데이트	새 콘솔 환경을 반영하도록 주제가 업데이트되었습니다.	2018년 10월 30일
Amazon EFS 샘플	CodeBuild는 빌드 중 프로젝트 buildspec 파일의 명령을 사용하여 Amazon EFS 파일 시스템을 탑재할 수 있습니다. 자세한 내용은 CodeBuild용 Amazon EFS 샘플을 참조하세요.	2018년 10월 26일
Bitbucket webhook	이제 CodeBuild는 리포지토리로 Bitbucket을 사용할 때 Webhook를 지원합니다. 자세한 내용은 CodeBuild용 Bitbucket 풀 요청 샘플을 참조하세요.	2018년 10월 2일
S3 로그	이제 CodeBuild는 S3 버킷에서 빌드 로그를 지원합니다. 이전에는 CloudWatch Logs를 통해서만 빌드 로그를 수행할 수 있었습니다. 자세한 내용은 프로젝트 생성을 참조하십시오.	2018년 9월 17일

다중 입력 소스 및 다중 출력 아티팩트	CodeBuild가 이제 입력 소스 1개 이상을 사용해 아티팩트 세트 1개 이상을 게시하는 프로젝트를 지원합니다. 자세한 내용은 다중 입력 소스와 출력 아티팩트 샘플 및 CodeBuild와 CodePipeline의 통합 및 다중 입력 소스와 출력 아티팩트 샘플을 참조하세요.	2018년 8월 30일
의미 체계 버전 관리 샘플	CodeBuild 사용 설명서에 빌드할 때 의미 체계 버전 관리를 사용해 아티팩트 이름을 생성하는 방법을 설명하는 사용 사례 기반 샘플이 추가되었습니다. 자세한 내용은 의미 체계 버전 관리를 사용해 빌드 아티팩트 샘플 이름 지정을 참조하십시오.	2018년 8월 14일
새로운 정적 웹 사이트 샘플	CodeBuild 사용 설명서에 빌드 출력을 S3 버킷에서 호스팅하는 방법을 설명하는 사용 사례 기반 샘플이 추가되었습니다. 샘플은 암호화되지 않은 빌드 아티팩트를 지원하는 최근 옵션을 이용합니다. 자세한 내용은 빌드 출력을 S3 버킷에서 호스팅하여 정적 웹사이트 생성을 참조하십시오.	2018년 8월 14일

의미 체계 버전 관리를 사용해 아티팩트 이름을 재정의할 수 있도록 지원하는 옵션	이제는 의미 체계 버전 관리를 사용해 CodeBuild가 빌드 아티팩트의 이름을 지정할 때 사용하는 형식을 지정할 수 있습니다. 하드 코딩된 이름을 가진 빌드 아티팩트는 동일한 하드 코딩 이름을 사용하는 이전 빌드 아티팩트를 덮어쓰기 때문에 이 기능은 유용합니다. 예를 들어 하루에 여러 번 빌드를 트리거하는 경우 결과물 이름에 타임스탬프를 추가할 수 있습니다. 각 빌드 아티팩트 이름은 고유하므로 이전 빌드의 아티팩트를 덮어쓰지 않습니다.	2018년 8월 7일
암호화되지 않은 빌드 아티팩트 지원	CodeBuild는 이제 암호화되지 않은 빌드 아티팩트를 포함한 빌드를 지원합니다. 자세한 내용은 빌드 프로젝트 생성(콘솔)을 참조하세요.	2018년 7월 26일
Amazon CloudWatch 지표 및 경보 지원	CodeBuild는 이제 CloudWatch 지표 및 경보와의 통합을 제공합니다. CodeBuild 또는 CloudWatch 콘솔을 사용하여 프로젝트 및 계정 수준에서 빌드를 모니터링할 수 있습니다. 자세한 내용은 빌드 모니터링을 참조하십시오.	2018년 7월 19일
빌드 상태 보고 지원	이제 CodeBuild는 빌드의 시작 및 완료 상태를 소스 공급자에게 보고할 수 있습니다. 자세한 내용은 CodeBuild에서 빌드 프로젝트 생성을 참조하세요.	2018년 7월 10일

CodeBuild 설명서에 환경 변수가 추가됨	빌드 환경 내의 환경 변수 페이지가 CODEBUILD_BUILD_ID, CODEBUILD_LOG_PATH 및 CODEBUILD_START_TIME 환경 변수와 함께 업데이트되었습니다.	2018년 7월 9일
buildspec 파일에서 `finally` 블록 지원	CodeBuild 설명서에서 빌드 사양 파일의 선택적인 `finally` 블록에 대한 세부 정보가 업데이트되었습니다. 마지막 블록에서의 명령은 항상 해당 명령 블록에서의 명령 후에 실행됩니다. 자세한 내용은 Buildspec 구문을 참조하십시오.	2018년 6월 20일
CodeBuild 에이전트 업데이트 알림	CodeBuild 설명서는 CodeBuild 에이전트의 새 버전이 릴리스될 때 Amazon SNS를 사용하여 알림을 받을 수 있는 방법에 대한 세부 정보와 함께 업데이트되었습니다. 자세한 내용은 새 AWS CodeBuild 에이전트 버전에 대한 알림 수신을 참조하세요.	2018년 15월 6일

이전 업데이트

다음 표에서는 2018년 6월 이전 AWS CodeBuild 사용 설명서의 각 릴리스에서 변경된 중요 사항에 대해 설명합니다.

변경	설명	Date
Windows 빌드 지원	이제 CodeBuild는 Windows의 .NET Core 2.0의 미리 패키징된 빌드 환경을 포함하여	2018년 5월 25일

변경	설명	Date
	Microsoft Windows Server 플랫폼의 빌드를 지원합니다. 자세한 내용은 CodeBuild용 Microsoft Windows 샘플 실행 섹션을 참조하세요.	
빌드 멱등성 지원	AWS Command Line Interface (AWS CLI)를 사용하여 `start-build` 명령을 실행할 때 빌드가 멱등성을 갖도록 지정할 수 있습니다. 자세한 내용은 빌드 실행(AWS CLI) 섹션을 참조하세요.	2018년 5월 15일
더 많은 빌드 프로젝트 설정 재정의 지원	빌드를 생성할 때 이제 더 많은 빌드 프로젝트 설정을 재정의할 수 있습니다. 재정의는 해당 빌드에만 적용됩니다. 자세한 내용은 수동으로 AWS CodeBuild 빌드 실행 섹션을 참조하세요.	2018년 5월 15일
VPC 엔드포인트 지원	이제 VPC 종단점을 사용하여 빌드의 보안을 향상시킬 수 있습니다. 자세한 내용은 VPC 엔드포인트 사용 섹션을 참조하세요.	2018년 3월 18일
트리거의 지원	이제 정기적으로 빌드를 예약하도록 트리거를 생성할 수 있습니다. 자세한 내용은 AWS CodeBuild 트리거 생성 섹션을 참조하세요.	2018년 3월 28일

변경	설명	Date
FIPS 엔드포인트 설명서	이제 AWS Command Line Interface (AWS CLI) 또는 AWS SDK를 사용하여 CodeBuild에 4개의 FIPS(연방 정보 처리 표준) 엔드포인트 중 하나를 사용하도록 지시하는 방법에 대해 알아볼 수 있습니다. 자세한 내용은 AWS CodeBuild 엔드포인트 지정 섹션을 참조하세요.	2018년 3월 28일
AWS CodeBuild 아시아 태평양(뭄바이), 유럽(파리) 및 남아메리카(상파울루)에서 사용 가능	AWS CodeBuild 이제 아시아 태평양(뭄바이), 유럽(파리) 및 남아메리카(상파울루) 리전에서를 사용할 수 있습니다. 자세한 내용은 Amazon Web Services 일반 참조의 AWS CodeBuild 섹션을 참조하세요.	2018년 3월 28일
GitHub Enterprise Server 지원	CodeBuild는 이제 GitHub Enterprise Server 리포지토리에 저장된 소스 코드에서 빌드할 수 있습니다. 자세한 내용은 GitHub Enterprise Server 샘플 실행 섹션을 참조하세요.	2018년 1월 25일
Git 복제 수준 지원	CodeBuild는 이제 이력이 지정된 커밋 수로 잘린 부분 복제의 생성을 지원합니다. 자세한 내용은 빌드 프로젝트 생성 섹션을 참조하세요.	2018년 1월 25일

변경	설명	Date
VPC 지원	이제 VPC 활성화 빌드는 VPC 내부의 리소스에 액세스할 수 있습니다. 자세한 내용은 VPC 지원 섹션을 참조하세요.	2017년 11월 27일
종속성 캐싱 지원	CodeBuild는 이제 종속성 캐싱을 지원합니다. 이를 통해 CodeBuild는 빌드 환경의 재사용 가능한 특정 부분을 캐시에 저장하고 빌드 전반에서 사용할 수 있습니다.	2017년 11월 27일
빌드 배지 지원	CodeBuild는 이제 동적으로 생성되어 프로젝트의 최신 빌드 상태를 표시하는 내장 가능형 이미지(배지)를 제공하는 빌드 배지 사용을 지원합니다. 자세한 내용은 빌드 배지 샘플 섹션을 참조하세요.	2017년 11월 27일
AWS Config 통합	AWS Config 는 이제 CodeBuild를 AWS 리소스로 지원합니다. 즉, 서비스가 CodeBuild 프로젝트를 추적할 수 있습니다. 에 대한 자세한 내용은 단원을 AWS Config참조하십시오AWS Config 샘플.	2017년 10월 20일

변경	설명	Date
GitHub 리포지토리에서 업데이트된 소스 코드를 자동으로 다시 빌드	소스 코드가 GitHub 리포지토리에 저장되는 경우 코드 변경이 리포지토리로 푸시될 때마다 AWS CodeBuild 가 소스 코드를 다시 빌드하도록 활성화할 수 있습니다. 자세한 내용은 GitHub pull 요청 및 웹후크 필터 샘플 실행 섹션을 참조하세요.	2017년 9월 21일
Amazon EC2 Systems Manager Parameter Store에서 중요하거나 규모가 큰 환경 변수를 저장하고 검색하는 새로운 방법	이제 AWS CodeBuild 콘솔 또는를 사용하여 Amazon EC2 Systems Manager Parameter Store에 저장된 민감하거나 큰 환경 변수를 AWS CLI 검색할 수 있습니다. 또한 AWS CodeBuild 콘솔을 사용하여 Amazon EC2 Systems Manager Parameter Store에 이러한 유형의 환경 변수를 저장할 수 있습니다. 이전에는 이러한 유형의 환경 변수를 빌드 사양에 포함하거나 빌드 명령을 실행하여 AWS CLI를 자동화하는 방식으로만 이러한 환경 변수를 검색할 수 있었습니다. Amazon EC2 Systems Manager Parameter Store 콘솔을 사용해서만 이러한 유형의 환경 변수를 저장할 수 있었습니다. 자세한 내용은 빌드 프로젝트 생성, 빌드 프로젝트 설정 변경, 빌드를 수동으로 실행 섹션을 참조하세요.	2017년 9월 14일

변경	설명	Date
빌드 삭제 지원	이제 AWS CodeBuild에서 빌드를 삭제할 수 있습니다. 자세한 내용은 빌드 삭제 섹션을 참조하세요.	2017년 8월 31일
buildspec을 사용하여 Amazon EC2 Systems Manager Parameter Store에 저장된 중요하거나 규모가 큰 환경 변수를 검색하는 방식이 업데이트되었습니다.	AWS CodeBuild 이제를 사용하면 buildspec을 사용하여 Amazon EC2 Systems Manager Parameter Store에 저장된 민감하거나 큰 환경 변수를 더 쉽게 검색할 수 있습니다. 이전에는 빌드 명령을 실행하여 AWS CLI를 자동화하는 방식으로만 이러한 환경 변수를 검색할 수 있었습니다. 자세한 내용은 `parameter-store`의 buildspec 구문 매핑을 참조하세요.	2017년 8월 10일
AWS CodeBuild Bitbucket 지원	CodeBuild는 이제 Bitbucket 리포지토리에 저장된 소스 코드에서 빌드할 수 있습니다. 자세한 내용은 빌드 프로젝트 생성 및 빌드를 수동으로 실행 섹션을 참조하세요.	2017년 8월 10일
AWS CodeBuild 미국 서부(캘리포니아 북부), 유럽(런던) 및 캐나다(중부)에서 사용 가능	AWS CodeBuild 이제 미국 서부(캘리포니아 북부), 유럽(런던) 및 캐나다(중부) 리전에서를 사용할 수 있습니다. 자세한 내용은 Amazon Web Services 일반 참조의 AWS CodeBuild 섹션을 참조하세요.	2017년 6월 29일

변경	설명	Date
대체 빌드 사양 파일 이름 및 지원 위치	이제 소스 코드의 루트에서 `buildspec.yml` 이라는 기본 빌드 사양 파일 대신 빌드 프로젝트에 사용할 빌드 사양 파일의 대체 파일 이름 또는 위치를 지정할 수 있습니다. 자세한 내용은 buildspec 파일 이름 및 스토리지 위치 섹션을 참조하세요.	2017년 6월 27일
업데이트된 빌드 알림 샘플	CodeBuild는 이제 Amazon CloudWatch Events 및 Amazon Simple Notification Service(Amazon SNS)를 통한 파이프라인 상태 변경 알림을 기본적으로 지원합니다. 이전 빌드 알림 샘플 섹션이 업데이트되어 이 새로운 동작을 보여줍니다.	2017년 6월 22일
Docker 사용자 지정 이미지 샘플 추가	CodeBuild 및 사용자 지정 Docker 빌드 이미지를 사용하여 도커 이미지를 빌드하고 실행하는 방법을 보여 주는 샘플이 추가되었습니다. 자세한 내용은 도커 사용자 지정 이미지 샘플 섹션을 참조하십시오.	2017년 6월 7일

변경	설명	Date
GitHub 풀 요청에 대한 소스 코드 가져오기	GitHub 리포지토리에 저장된 소스 코드를 사용하는 CodeBuild를 통해 빌드를 실행하면 이제 빌드할 GitHub 풀 요청 ID를 지정할 수 있습니다. 또한 커밋 ID, 분기 이름 또는 태그 이름을 대신 지정할 수도 있습니다. 자세한 내용은 빌드 실행(콘솔)의 소스 버전 값 및 빌드 실행(AWS CLI)의 `sourceVersion` 값을 참조하세요.	2017년 6월 6일
빌드 사양 버전 업데이트	빌드 사양 형식의 새 버전이 릴리스되었습니다. 버전 0.2는 기본 쉘의 별도 인스턴스에서 각 빌드 명령을 실행하는 CodeBuild 문제를 해결합니다. 또한 버전 0.2에서 `environment_variables` 는 `env`로, `plaintext` 는 `variables` 로 이름이 다시 지정되었습니다. 자세한 내용은 CodeBuild의 빌드 사양 참조 섹션을 참조하세요.	2017년 5월 9일
GitHub에서 사용할 수 있는 빌드 이미지용 Dockerfiles	에서 제공하는 많은 빌드 이미지에 대한 정의 AWS CodeBuild 는 GitHub에서 Dockerfiles로 사용할 수 있습니다. 자세한 내용은 CodeBuild가 제공하는 도커 이미지에 있는 표의 정의 열을 참조하세요.	2017년 5월 2일

변경	설명	Date
AWS CodeBuild 유럽(프랑크푸르트), 아시아 태평양(싱가포르), 아시아 태평양(시드니) 및 아시아 태평양(도쿄)에서 사용 가능	AWS CodeBuild 이제 유럽(프랑크푸르트), 아시아 태평양(싱가포르), 아시아 태평양(시드니) 및 아시아 태평양(도쿄) 리전에서를 사용할 수 있습니다. 자세한 내용은 Amazon Web Services 일반 참조의 AWS CodeBuild 섹션을 참조하세요.	2017년 3월 21일
CodeBuild에 대한 CodePipeline 테스트 작업 지원	이제 CodePipeline의 파이프라인에 CodeBuild를 사용하는 테스트 작업을 추가할 수 있습니다. 자세한 내용은 CodeBuild 테스트 작업을 파이프라인에 추가(CodePipeline 콘솔) 섹션을 참조하세요.	2017년 3월 8일
빌드 사양 파일에서 선택된 최상위 디렉터리 내부로부터 빌드 출력 가져오기 지원	이제 빌드 사양 파일을 사용하여 CodeBuild가 빌드 출력 아티팩트에 포함하도록 할 내용을 지정할 수 있습니다. `base-directory` 매핑을 이용하면 이러한 작업이 가능합니다. 자세한 내용은 buildspec 구문 섹션을 참조하세요.	2017년 2월 8일
내장 환경 변수	AWS CodeBuild 는 빌드에서 사용할 수 있는 추가 내장 환경 변수를 제공합니다. 빌드를 시작한 엔터티, 소스 코드 리포지토리에 대한 URL, 소스 코드의 버전 ID 등을 설명하는 환경 변수가 추가되었습니다. 자세한 내용은 빌드 환경의 환경 변수 섹션을 참조하세요.	2017년 1월 30일

변경	설명	Date
AWS CodeBuild 미국 동부(오하이오)에서 사용 가능	AWS CodeBuild 이제 미국 동부(오하이오) 리전에서를 사용할 수 있습니다. 자세한 내용은 Amazon Web Services 일반 참조의 AWS CodeBuild 섹션을 참조하세요.	2017년 1월 19일
쉘 및 명령 동작 정보	CodeBuild는 사용자가 빌드 환경의 기본 쉘의 개별 인스턴스에서 지정한 각 명령을 실행합니다. 이러한 기본 동작은 명령에 예기치 못한 부작용을 초래할 수 있습니다. 필요하면 이러한 기본 동작을 해결하는 몇 가지 방법을 시도해 보는 것이 좋습니다. 자세한 내용은 빌드 환경의 쉘 및 명령 섹션을 참조하세요.	2016년 9월 12일
환경 변수 정보	CodeBuild는 빌드 명령에서 사용할 수 있는 다양한 환경 변수를 제공합니다. 사용자 고유의 환경 변수를 정의할 수도 있습니다. 자세한 내용은 빌드 환경의 환경 변수 섹션을 참조하세요.	2016년 7월 12일
문제 해결 주제	이제 문제 해결 정보가 제공됩니다. 자세한 내용은 문제 해결 AWS CodeBuild 섹션을 참조하세요.	2016년 5월 12일

변경	설명	Date
Jenkins 플러그인 최초 릴리스	CodeBuild Jenkins 플러그인이 처음으로 릴리스되었습니다. 자세한 내용은 Jenkins와 AWS CodeBuild 함께 사용 섹션을 참조하세요.	2016년 5월 12일
사용 설명서 최초 릴리스	이는 CodeBuild 사용 설명서의 최초 공개 릴리스입니다.	2016년 12월 1일

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