사용자 가이드 AWS CodePipeline



API 버전 2015-07-09

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

AWS CodePipeline: 사용자 가이드

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

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

Table of Contents

CodePipeline이란 무엇인가요?	. 1
지속적 제공 및 지속적 통합	. 1
CodePipeline으로 할 수 있는 작업은 무엇인가요?	2
간단히 보는 CodePipeline	. 2
CodePipeline은 어떻게 시작할 수 있나요?	. 3
개념	. 4
Pipelines	. 4
파이프라인 실행	. 6
스테이지 작업	. 7
작업 실행	. 7
실행 유형	. 8
작업 유형	. 8
아티팩트	. 8
소스 개정	. 8
트리거	. 9
변수	. 9
조건	10
규칙	10
DevOps 파이프라인 예제	10
파이프라인 실행 작동 방식	12
파이프라인 실행 시작 방법	13
파이프라인 실행에서 소스 개정이 처리되는 방법	13
파이프라인 실행을 중지하는 방법	14
SUPERSEDED 모드에서 실행을 처리하는 방법	17
QUEUED 모드에서 실행을 처리하는 방법	18
PARALLEL 모드에서 실행을 처리하는 방법	20
파이프라인 흐름 관리	20
입력 및 출력 아티팩트	23
스테이지 조건은 어떻게 작동하나요?	26
스테이지 조건에 대한 규칙	29
파이프라인 유형	30
나에게 적합한 파이프라인 유형은 무엇인가요?	30
시작	36
1단계: AWS 계정 및 관리 사용자 생성	36

가입 AWS 계정	36
관리자 액세스 권한이 있는 사용자 생성	. 37
2단계: CodePipeline에 대한 관리자 액세스를 위한 관리형 정책 적용	38
3단계: 설치 AWS CLI	39
4단계: CodePipeline 콘솔 열기	40
다음 단계	. 41
제품 및 서비스 통합	42
CodePipeline 작업 유형과의 통합	42
소스 작업 통합	42
빌드 작업 통합	49
테스트 작업 통합	. 51
배포 작업 통합	53
Amazon Simple Notification Service와 승인 작업 통합	60
호출 작업 통합	60
CodePipeline으로 일반 통합	. 61
커뮤니티 예제	. 65
블로그 게시물	. 65
자습서	. 69
자습서: CodePipeline을 사용하여 Amazon EC2 인스턴스에 배포	70
사전 조건	70
1단계: Amazon EC2 Linux 인스턴스 생성	. 71
2단계: EC2 인스턴스 역할에 아티팩트 버킷 권한 추가	. 72
3단계: 리포지토리에 스크립트 파일 추가	73
4단계: 파이프라인 생성	. 74
5단계: 파이프라인 테스트	79
자습서: CodePipeline을 사용하여 Docker 이미지를 빌드하고 Amazon ECR에 푸시(V2 유형)	80
사전 조건	81
1단계: 소스 리포지토리에 Dockerfile 추가	81
2단계: 소스 리포지토리에 imagedefinitions.json 파일 추가	. 83
3단계: 파이프라인 생성	. 84
4단계: 파이프라인 테스트	89
자습서: CodePipeline을 사용하여 Amazon EKS에 배포	90
사전 조건	90
1단계: (선택 사항) Amazon EKS에서 클러스터 생성	91
2단계: Amazon EKS에서 프라이빗 클러스터 구성	. 93
3단계: IAM에서 CodePipeline 서비스 역할 정책 업데이트	. 93

4단계: CodePipeline 서비스 역할에 대한 액세스 항목 생성	97
5단계: 소스 helm chart 리포지토리 생성 및 구성 파일 추가	97
6단계: 파이프라인 생성	98
자습서: 컴퓨팅을 사용하여 명령을 실행하는 파이프라인 생성(V2 유형)	100
사전 조건	101
1단계: 소스 파일 생성 및 GitHub 리포지토리에 푸시	101
2단계: 파이프라인 생성	102
3단계: 파이프라인 실행 및 빌드 명령 확인	105
자습서: Git 태그를 사용하여 파이프라인 시작하기	106
사전 조건	107
1단계: CloudShell을 열고 리포지토리 복제	108
2단계: Git 태그에 트리거할 파이프라인 생성	108
3단계: 릴리스용 커밋에 태그 지정	112
4단계: 변경 사항 릴리스 및 로그 보기	113
자습서: 파이프라인을 시작하기 위한 풀 요청의 브랜치 이름 필터링(V2 유형)	113
사전 조건	114
1단계: 지정된 브랜치에 대한 풀 요청을 시작할 파이프라인 생성	114
2단계: GitHub.com에서 풀 요청을 생성 및 병합하여 파이프라인 실행 시작	117
자습서: 파이프라인 수준 변수 사용	118
사전 조건	119
1단계: 파이프라인 및 빌드 프로젝트 생성	119
2단계: 변경 사항 릴리스 및 로그 보기	122
자습서: 간단한 파이프라인 생성(S3 버킷)	123
S3 버킷 생성	124
Windows Server Amazon EC2 인스턴스 생성 및 CodeDeploy 에이전트 설치	126
CodeDeploy에서 애플리케이션 생성	128
첫 번째 파이프라인 생성	130
다른 단계 추가	132
단계 간 전환 비활성화 및 활성화	139
리소스 정리	140
자습서: 간단한 파이프라인 생성(CodeCommit 리포지토리)	140
CodeCommit 리포지토리 생성	141
코드 다운로드, 커밋, 푸시	142
Amazon EC2 Linux 인스턴스 생성 및 CodeDeploy 에이전트 설치	145
CodeDeploy에서 애플리케이션 생성	147
첫 번째 파이프라인 생성	148

CodeCommit 리포지토리의 코드 업데이트	. 151
리소스 정리	. 153
심소 사료	. 153
사급지: 4년계 파이프다인 경경	. 153
자신 오신 원묘	. 155
파이프다인 경경	. 159
신계 더 우가 리스스 저희	. 100
디소스 영디 자스너·파이포라이 사태 벼경에 대한 이메이 아리은 스시쿼트로 Cloud/Wateb Events 그치 서	. 104
자급지. 파이프다인 영대 인영에 대한 이메일 달림을 구선하도록 CloudWatch Events 뉴릭 걸 저	165
O Amazon SNS록 사용하여 이메익 악림 석정	165
CodePipeline에 대한CloudWatch Events 악림 규칙 생성	166
리소스 정리	168
자습서:를 사용하여 Android 앱 빌드 및 테스트 AWS Device Farm	. 168
Device Farm 테스트를 사용하도록 CodePipeline 구성	. 169
· 자습서:를 사용하여 iOS 앱 테스트 AWS Device Farm	. 174
Device Farm 테스트를 사용하도록 CodePipeline 구성(Amazon S3 예)	. 175
자습서: Service Catalog에 배포하는 파이프라인 생성	. 180
옵션 1: 구성 파일 없이 Service Catalog에 배포	. 181
옵션 2: 구성 파일을 사용하여 Service Catalog에 배포	. 186
자습서:를 사용하여 파이프라인 생성 AWS CloudFormation	. 191
예제 1:를 사용하여 AWS CodeCommit 파이프라인 생성 AWS CloudFormation	. 191
예 2: AWS CloudFormation을 사용하여 Amazon S3 파이프라인 생성	. 193
자습서: AWS CloudFormation 배포 작업의 변수를 사용하는 파이프라인 생성	. 197
사전 조건: AWS CloudFormation 서비스 역할 및 CodeCommit 리포지토리 생성	. 198
1단계: 샘플 AWS CloudFormation 템플릿 다운로드, 편집 및 업로드	. 198
2단계: 파이프라인 생성	. 199
3단계: AWS CloudFormation 배포 작업을 추가하여 변경 세트 생성	. 202
4단계: 수동 승인 작업 추가	. 203
5단계: CloudFormation 배포 작업을 추가하여 변경 세트 실행	. 203
6단계: CloudFormation 배포 작업을 추가하여 스택 삭제	. 204
자습서: CodePipeline을 사용한 Amazon ECS 표준 배포	. 204
사전 조건	. 205
1단계: 소스 리포지토리에 빌드 사양 파일 추가	. 208
2단계: CD(Continuous Deployment) 파이프라인 만들기	. 210
3단계: CodeBuild 역할에 Amazon ECR 권한 추가	. 212

4단계: 파이프라인 테스트	. 212
자습서: Amazon ECR 소스 및 ECS-to-CodeDeploy 배포를 사용하여 파이프라인 생성	. 213
사전 조건	. 215
1단계: 이미지 생성 및 Amazon ECR 리포지토리로 푸시	. 215
2단계: 작업 정의 및 AppSpec 소스 파일 생성과 CodeCommit 리포지토리에 푸시	. 216
3단계: Application Load Balancer 및 대상 그룹 만들기	. 220
4단계: Amazon ECS 클러스터 및 서비스 생성	. 223
5단계: CodeDeploy 애플리케이션 및 배포 그룹 만들기(ECS 컴퓨팅 플랫폼)	. 225
6단계: 파이프라인 생성	. 226
7단계: 파이프라인 변경 및 배포 확인	. 230
자습서: Amazon Alexa Skill을 배포하는 파이프라인 생성	. 231
사전 조건	. 231
1단계: Alexa 개발자 서비스 LWA 보안 프로필 생성	. 232
2단계: Alexa Skill 소스 파일 생성 및 CodeCommit 리포지토리에 푸시	. 232
3단계: ASK CLI 명령을 사용하여 새로 고침 토큰 생성	. 234
4단계: 파이프라인 생성	. 234
5단계: 모든 소스 파일을 변경하고 배포 확인	. 236
자습서: Amazon S3를 배포 공급자로 사용하는 파이프라인 생성	. 237
옵션 1: Amazon S3에 정적 웹 사이트 배포	. 238
옵션 2: S3 소스 버킷에서 Amazon S3에 빌드된 아카이브 파일 배포	. 242
자습서:에 애플리케이션 게시 AWS Serverless Application Repository	. 248
시작하기 전 준비 사항	. 248
1단계: buildspec.yml 파일 생성	. 249
2단계: 파이프라인 생성 및 구성	. 249
3단계: 게시 애플리케이션 배포	. 251
4단계: 게시 작업 생성	252
자습서: Lambda 호출 작업과 함께 변수 사용	. 253
사전 조건	. 253
1단계: Lambda 함수 생성	. 254
2단계: 파이프라인에 Lambda 호출 작업 및 수동 승인 작업 추가	. 256
자습서: AWS Step Functions 호출 작업 사용	. 258
사전 조건: 단순 파이프라인 생성 또는 선택	. 259
1단계: 샘플 상태 시스템 생성	. 259
2단계: 파이프라인에 Step Functions 호출 작업 추가	. 259
자습서: AppConfig를 배포 공급자로 사용하는 파이프라인 생성	. 261
사전 조건	. 261

1단계: your AWS AppConfig 리소스 생성	261
2단계: S3 소스 버킷에 파일 업로드	262
3단계: 파이프라인 생성	263
4단계: 모든 소스 파일을 변경하고 배포 확인	264
자습서: GitHub 파이프라인 소스와 함께 전체 복제 사용	264
사전 조건	266
1단계: README 파일 생성	266
2단계: 파이프라인 및 빌드 프로젝트 생성	266
3단계: 연결을 사용하도록 CodeBuild 서비스 역할 정책 업데이트	270
4단계: 빌드 출력에서 리포지토리 명령 보기	270
자습서: CodeCommit 파이프라인 소스와 함께 전체 복제 사용	270
사전 조건	271
1단계: README 파일 생성	271
2단계: 파이프라인 및 빌드 프로젝트 생성	272
3단계: 리포지토리를 복제하도록 CodeBuild 서비스 역할 정책 업데이트	275
4단계: 빌드 출력에서 리포지토리 명령 보기	275
자습서: AWS CloudFormation StackSets 배포 작업을 사용하여 파이프라인 생성	275
사전 조건	276
1단계: 샘플 AWS CloudFormation 템플릿 및 파라미터 파일 업로드	277
2단계: 파이프라인 생성	278
3단계: 초기 배포 보기	281
4단계: CloudFormationStackInstances 작업 추가	281
5단계: 배포를 위한 스택 세트 리소스 보기	282
6단계: 스택 세트 업데이트	283
입력 조건으로 파이프라인에 대한 변수 확인 규칙 생성	283
사전 조건	284
1단계: 샘플 소스 파일을 생성하고 GitHub 리포지토리에 추가	284
2단계: 파이프라인 생성	285
2단계: 빌드 스테이지를 편집하여 조건 및 규칙 추가	288
3단계: 파이프라인 실행 및 확인된 변수 보기	290
모범 사례 및 사용 사례	293
CodePipeline을 사용하는 방법의 예	293
Amazon S3 AWS CodeCommit, 및에서 CodePipeline 사용 AWS CodeDeploy	293
CodePipeline과 타사 작업 공급자 사용(GitHub 및 Jenkins)	294
CodeBuild에서 CodePipeline을 사용하여 코드 컴파일, 빌드, 테스트	294

CodePipeline과 Amazon ECS를 함께 사용하여 컨테이너 기반 애플리케이션을 클라우드에 지속적으로 제공	295
CodePipeline과 Elastic Beanstalk를 함께 사용하여 웹 애플리케이션을 클라우드에 지속적으 로 제공	295
Lambda 기반 및 서버리스 애플리케이션을 지속적으로 제공 AWS Lambda 하려면와 함께 CodePipeline 사용	295
CodePipeline을 AWS CloudFormation 템플릿과 함께 사용하여 클라우드에 지속적으로 제 고	005
승	295
가용성	290
기 8 8 CodePineline의 V/PC 에드포이트 생성	290
V/PC 서전 무제 해견	207
시테이지 및 잔언을 사용하여 CI/CD 파이프라이 전이	200
파이프라인 스테이지 및 잔언 생성	200
사용자 지정 파이프라인 생성(코속)	301
파이프라인 생성(CLI)	312
정적 템플릿에서 파이프라인 생성	318
파이프라인 편집	324
파이프라인 편집(콘솔)	325
파이프라인 편집(AWS CLI)	328
파이프라인 및 세부 정보 보기	332
파이프라인 보기(콘솔)	333
파이프라인에서 작업 세부 정보 보기(콘솔)	337
파이프라인 ARN 및 서비스 역할 ARN 보기(콘솔)	341
파이프라인 세부 정보 및 이력 보기(CLI)	342
실행 기록에서 스테이지 조건에 대한 규칙 결과 보기	343
파이프라인 삭제	346
파이프라인 삭제(콘솔)	346
파이프라인 삭제(CLI)	346
다른 계정의 리소스를 사용하는 파이프라인 생성	347
필수 조건: AWS KMS 암호화 키 생성	349
1단계: 계정 정책 및 역할 설정	350
2단계: 파이프라인 편집	358
이벤트 기반 변경 감지를 사용하도록 폴링 파이프라인 마이그레이션	361
폴링 파이프라인을 마이그레이션하는 방법	361
계정의 폴링 파이프라인 보기	363

CodeCommit 소스를 사용하여 폴링 파이프라인 마이그레이션	368
이벤트용으로 활성화된 S3 소스를 사용하여 폴링 파이프라인 마이그레이션	389
S3 소스 및 CloudTrail 트레일을 사용하여 폴링 파이프라인 마이그레이션	417
GitHub(OAuth 앱을 통해) 소스 작업에 대한 폴링 파이프라인을 연결로 마이그레이션	452
GitHub(OAuth 앱 사용) 소스 작업에 대한 폴링 파이프라인을 웹후크로 마이그레이션	455
CodePipeline 서비스 역할 생성	. 472
CodePipeline 서비스 역할 생성(콘솔)	473
CodePipeline 서비스 역할 생성(CLI)	473
리소스에 태그 지정	. 475
파이프라인 태그 지정	476
파이프라인 태그 지정(콘솔)	476
파이프라인 태그 지정(CLI)	478
알림 규칙 생성	480
	484
소스 작업을 사용하여 자사 소스 공급자에 연결	486
Amazon ECR 소스 작업 및 EventBridge	486
Amazon ECR 소스에 대한 EventBridge 규칙 생성(콘솔)	. 487
Amazon ECR 소스에 대한 EventBridge 규칙 생성(CLI)	489
Amazon ECR 소스에 대한 EventBridge 규칙 생성(AWS CloudFormation 템플릿)	492
EventBridge 이벤트를 사용하는 Amazon S3 소스 작업에 연결	. 497
이벤트에 대해 활성화된 S3 소스를 사용하여 파이프라인 생성(CLI)	. 498
이벤트에 대해 활성화된 S3 소스를 사용하여 파이프라인 생성(AWS CloudFormation 템플	
릿)	. 502
EventBridge 및를 사용하는 Amazon S3 소스 작업에 연결 AWS CloudTrail	. 525
Amazon S3 소스에 대한 EventBridge 규칙 생성(콘솔)	526
Amazon S3 소스에 대한 EventBridge 규칙 생성(CLI)	529
Amazon S3 소스에 대한 EventBridge 규칙 생성(AWS CloudFormation 템플릿)	535
CodeCommit 소스 작업 및 EventBridge	548
CodeCommit 소스에 대한 EventBridge 규칙 생성(콘솔)	. 548
CodeCommit 소스에 대한 EventBridge 규칙 생성(CLI)	551
CodeCommit 소스에 대한 EventBridge 규칙 생성(AWS CloudFormation 템플릿)	555
CodeConnections를 사용하여 파이프라인에 타사 소스 공급자 추가	564
Bitbucket Cloud 연결	564
Bitbucket Cloud에 대한 연결 생성(콘솔)	566
Bitbucket Cloud에 대한 연결 생성(CLI)	569
GitHub 연결	571

GitHub에 대한 연결 생성(콘솔)	573
GitHub에 대한 연결 생성(CLI)	576
GitHub Enterprise Server 연결	577
GitHub Enterprise Server에 대한 연결 생성(콘솔)	579
GitHub Enterprise Server에 대한 연결 및 호스트 생성(CLI)	583
GitLab.com 연결	586
GitLab.com에 대한 연결 생성(콘솔)	588
GitLab.com에 대한 연결 생성(CLI)	592
GitLab 자체 관리형을 위한 연결	594
GitLab 자체 관리형에 대한 연결 생성(콘솔)	596
호스트를 만들고 GitLab 자체 관리형에 연결(CLI)	598
다른 계정과 공유된 연결 사용	601
트리거 및 필터링을 사용하여 시작 파이프라인 자동화	602
트리거 필터 고려 사항	604
공급자별 트리거에 대한 풀 요청 이벤트	604
트리거 필터의 예	605
필터 없이 코드 푸시 시 트리거 추가	614
코드 푸시 또는 풀 요청 이벤트 유형을 사용하여 트리거 추가	614
푸시 및 풀 요청 이벤트 유형에 대한 필터 추가(콘솔)	615
푸시 및 풀 요청 이벤트 유형에 대한 필터 추가(CLI)	618
푸시 및 풀 요청 이벤트 유형에 대한 필터 추가(AWS CloudFormation 템플릿)	620
변경 감지를 끄는 트리거 추가	622
파이프라인 수동 시작 및 중지	623
CodePipeline에서 파이프라인 시작	623
수동으로 파이프라인 시작	
일정에 따라 파이프라인 시작	626
소스 개정 재정의로 파이프라인 시작	629
파이프라인 실행 중지	632
파이프라인 실행 중지(콘솔)	633
인바운드 실행 중지(콘솔)	635
파이프라인 실행 중지(CLI)	636
인바운드 실행 중지(CLI)	638
파이프라인 실행에 대한 기록 보기 및 모드 설정	639
실행 보기	639
파이프라인 실행 내역 보기(콘솔)	639
실행 상태 보기(콘솔)	641

인바운드 실행 보기(콘솔)	643
파이프라인 실행 소스 개정 보기(콘솔)	. 643
작업 실행 보기(콘솔)	645
작업 아티팩트 및 아티팩트 스토어 정보 보기(콘솔)	. 646
파이프라인 세부 정보 및 이력 보기(CLI)	. 646
파이프라인 실행 모드 설정 또는 변경	. 658
실행 모드 보기에 대한 고려 사항	658
실행 모드 간 전환 시 고려 사항	661
파이프라인 실행 모드 설정 또는 변경(콘솔)	662
파이프라인 실행 모드 설정(CLI)	663
롤백 또는 재시도 스테이지	667
실패한 스테이지 또는 실패한 작업에 대해 스테이지 재시도 구성	667
스테이지 재시도 시 고려 사항	668
실패한 스테이지를 수동으로 재시도	668
실패 시 자동 재시도를 위한 스테이지 구성	. 672
스테이지 롤백 구성	. 678
롤백 고려 사항	678
스테이지를 수동으로 롤백	678
자동 롤백을 위한 스테이지 구성	683
실행 목록에서 롤백 상태 보기	. 687
롤백 상태 세부 정보 보기	690
스테이지에 대한 조건 구성	695
스테이지 조건에 대한 사용 사례	696
스테이지 조건에 대해 구성된 결과 고려 사항	696
스테이지 조건에 대해 구성된 규칙 고려 사항	. 697
입력 조건 생성	. 697
입력 조건 생성 - CloudWatchAlarm 규칙 예제(콘솔)	698
건너뛰기 결과 및 VariableCheck 규칙을 사용하여 입력 조건 생성(콘솔)	. 699
입력 조건 생성(CLI)	701
입력 조건 생성(CFN)	703
실패 시 조건 생성	. 704
실패 시 조건 생성(콘솔)	. 704
재시도 결과 예제를 사용하여 onFailure 조건 생성(콘솔)	705
실패 시 조건 생성(CLI)	706
실패 시 조건 생성(CFN)	708
성공 시 조건 생성	709

성공 시 조건 생성(콘솔)	. 709
성공 시 조건 생성(CLI)	711
성공 시 조건 생성(CFN)	713
스테이지 조건 삭제	. 714
스테이지 조건 재정의	. 715
작업 유형, 사용자 지정 작업 및 승인 작업 사용	717
작업 유형 처리	. 717
작업 유형 요청	. 719
파이프라인에 사용 가능한 작업 유형 추가(콘솔)	725
작업 유형 보기	. 726
작업 유형 업데이트	. 728
파이프라인에 대한 사용자 지정 작업 생성	. 729
사용자 지정 작업 생성	. 731
사용자 지정 작업에 대한 작업자 생성	. 735
파이프라인에 사용자 지정 작업 추가	741
CodePipeline에서 사용자 지정 작업 태그 지정	. 744
사용자 지정 작업에 태그 추가	. 745
사용자 지정 작업에 대한 태그 보기	745
사용자 지정 작업에 대한 태그 편집	746
사용자 지정 작업에서 태그 제거	746
파이프라인에서 Lambda 함수 호출	. 747
1단계: 파이프라인 생성	749
2단계: Lambda 함수 생성	. 750
3단계: CodePipeline 콘솔의 파이프라인에 Lambda 함수 추가	. 754
4단계: Lambda 함수로 파이프라인 테스트	755
5단계: 다음 절차	756
예제 JSON 이벤트	. 757
추가 샘플 함수	. 758
스테이지에 수동 승인 작업 추가	. 771
수동 승인 작업에 대한 구성 옵션	. 771
승인 작업의 설정 및 워크플로우 개요	. 772
CodePipeline에서 IAM 사용자에게 승인 권한 부여	. 773
서비스 역할에 Amazon SNS 권한 부여	776
수동 승인 작업 추가	778
승인 작업 승인 또는 거부	. 781
수동 승인 알림에 대한 JSON 데이터 형식	. 785

파이프라인에 교차 리전 작업 추가	786
파이프라인에서 교차 리전 작업 관리(콘솔)	788
파이프라인에 교차 리전 작업 추가(CLI)	790
파이프라인에 교차 리전 작업 추가(AWS CloudFormation)	795
변수 작업	. 798
변수에 대한 작업 구성	799
출력 변수 보기	803
예: 수동 승인에 변수 사용	805
예: CodeBuild 환경 변수와 함께 BranchName 변수 사용	806
단계 전환 작업	808
전환 비활성화 또는 활성화(콘솔)	808
전환 비활성화 또는 활성화(CLI)	810
Monitoring pipelines	812
CodePipeline 이벤트 모니터링	813
세부 정보 유형	814
파이프라인 수준 이벤트	816
단계 수준 이벤트	. 825
작업 수준 이벤트	. 829
파이프라인 이벤트에 알림을 보내는 규칙 생성	837
이벤트 자리 표시자 버킷 참조	841
리전별 이벤트 자리 표시자 버킷 이름	842
AWS CloudTrail을 사용하여 API 호출 로깅	. 845
CloudTrail의 CodePipeline 정보	846
CodePipeline 로그 파일 항목 이해	847
CodePipeline CloudWatch 지표	849
PipelineDuration	849
FailedPipelineExecutions	849
문제 해결	851
파이프라인 오류: AWS Elastic Beanstalk 으로 구성한 파이프라인이 오류 메시지를 반환	
했습니다. "Deployment failed. The provided role does not have sufficient permissions:	
Service:AmazonElasticLoadBalancing"	852
배포 오류: "DescribeEvents" 권한이 누락되면 AWS Elastic Beanstalk 배포 작업으로 구성한 ፤	타
이프라인이 실패하는 대신 멈춥니다	853
파이프라인 오류: 소스 작업이 권한이 부족하다는 메시지를 반환합니다. "CodeCommit 리포ㅈ	토
리 repository-name에 액세스할 수 없습니다. 파이프라인 IAM 역할에 있는 리포지토리 액	세
스 권한이 부족한지 확인하십시오."	853

	파이프라인 오류: Jenkins 빌드 또는 테스트 작업을 오래 지속했는데 자격 증명 또는 권한이 없이서 실패했습니다.	어 853
	파이프라인 오류: 다른 AWS 리전에서 생성된 버킷을 사용하여 한 AWS 리전에서 생성된 파이프	<u>I</u>
	라인은 코드 "JobFailed"와 함께 "InternalError"를 반환합니다 배포 오류· WAR 파인이 포함되 ZIP 파인이에 성공적으로 배포 AWS Elastic Beanstalk 되었지만	854
	애플리케이션 URL이 404 찾을 수 없음 오류를 보고합니다.	853
	파이프라인 아티팩트 폴더 이름이 잘린 것처럼 보입니다	855
	Bitbucket, GitHub, GitHub Enterprise Server 또는 GitLab.com 연결을 위한 CodeBuild GitClone	•
	권한 주가 CodeCommit 스스 자언에 대한 CodePuild CitClone 권한 초가	855
	파이프라인 오류: CodeDeployToECS 작업을 통해 배포하면 "<소스 아티팩트 이름>에서 작업 정	ਰਹਾ ਤ
	의 아티팩트 파일을 읽으려고 시도하는 동안 예외가 발생했습니다"라는 오류 메시지가 반환됩니	-
	다	858
	GitHub(OAuth 앱 사용) 소스 작업: 리포지토리 목록에 다른 리포지토리가 표시됨	858
	GitHub(GitHub 앱 사용) 소스 작업: 리포시도리에 내한 연결을 완료할 수 없음 Amazon S3 오류: CodePineline 서비스 연한 <arn>에서 S3 버킹(-BucketName>에 대한 S3 앤</arn>	859
	세스가 거부되었습니다.	859
	Amazon S3, Amazon ECR 또는 CodeCommit 소스를 사용하는 파이프라인은 더 이상 자동으로	-
	시작되지 않음	861
	GitHub에 연결할 때 연결 오류 말생: "A problem occurred, make sure cookies are enabled in your browser" 또는 "An organization owner must install the GitHub app"	863
	실행 모드가 QUEUED 또는 PARALLEL 모드로 변경된 파이프라인은 실행 한도에 도달하면 실	000
	패합니다	863
	QUEUED 또는 SUPERSEDED 모드로 변경할 때 편집하면 PARALLEL 모드의 파이프라인에 오	2
	대된 파이프라인 정의가 있습니다 PARALLEL 모드에서 벼겨되 파이프라이이 이저 시해 모드르 포시하니다	864 864
	파일 경로별 트리거 필터링을 사용하는 연결이 있는 파이프라인은 브랜치 생성 시 시작되지 않	004
	을 수 있습니다	864
	파일 경로별 트리거 필터링을 사용하는 연결이 있는 파이프라인은 파일 한도에 도달하면 시작되	<u> </u>
	지 않을 수 있습니다.	865
	PARALLEL 포트의 CodeCommit 또는 33 또스 개칭이 EventBridge 이팬트와 될지하지 않을 두 있습니다.	865
	오류 메시지와 함께 EC2 배포 작업이 실패함 No such file	865
	cluster unreachable 오류 메시지와 함께 EKS 배포 작업이 실패함	866
ш	다른 문제에 도움이 필요하십니까?	867
모	안	868

데이터 보호	. 868
인터네트워크 트래픽 개인 정보	. 869
저장 시 암호화	. 870
전송 중 데이터 암호화	. 870
암호화 키 관리	. 870
CodePipeline의 Amazon S3에 저장된 아티팩트에 대해 서버 측 암호화 구성	871
AWS Secrets Manager 를 사용하여 데이터베이스 암호 또는 타사 API 키 추적	. 874
Identity and Access Management(IAM)	. 874
대상	. 875
ID를 통한 인증	. 875
정책을 사용하여 액세스 관리	. 878
AWS CodePipeline 에서 IAM을 사용하는 방법	. 880
보안 인증 기반 정책 예	. 886
리소스 기반 정책 예제	. 921
문제 해결	. 922
CodePipeline 권한 참조	. 924
CodePipeline 서비스 역할 관리	. 934
인시던트 대응	. 940
규정 준수 확인	. 941
복원성	. 942
인프라 보안	. 942
보안 모범 사례	. 943
파이프라인 구조 참조	. 944
파이프라인 선언	. 947
name	. 949
roleArn	. 950
artifactStore OR artifactStores	. 950
stages	. 951
version	. 952
executionMode	. 952
pipelineType	. 952
variables	. 952
triggers	. 952
metadata	. 957
스테이지 선언	. 957
name	. 960

actions	
conditions	
rules	
작업 선언	
name	
region	
roleArn	
namespace	
actionTypeId	
InputArtifacts	
outputArtifacts	
configuration(작업 공급자별)	
runOrder	970
CodePipeline에서 유효한 작업 공급자	
PollForSourceChanges 파라미터의 유효한 설정	975
각 작업 유형에 대한 유효한 입력 및 출력 아티팩트	977
각 공급자 유형에 유효한 구성 파라미터	979
작업 구조 참조	
Amazon EC2 작업 참조	
작업 유형	
구성 파라미터	
입력 아티팩트	
출력 아티팩트	
EC2 배포 작업에 대한 서비스 역할 정책 권한	
작업 선언	
다음 사항도 참조하세요	
Amazon ECR 소스 작업 참조	
작업 유형	
구성 파라미터	
입력 아티팩트	
출력 아티팩트	
출력 변수	
서비스 역할 권한: Amazon ECR 작업	
작업 선언(Amazon ECR 예제)	
다음 사항도 참조하세요	

ECRBuildAndPublish 빌드 작업 참조	
작업 유형	
구성 파라미터	
입력 아티팩트	1000
출력 아티팩트	1000
출력 변수	1000
서비스 역할 권한: ECRBuildAndPublish 작업	1000
작업 선언	
다음 사항도 참조하세요	1003
Amazon ECS 및 CodeDeploy 블루-그린 배포 작업 참조	1003
작업 유형	
구성 파라미터	1005
입력 아티팩트	1006
출력 아티팩트	1007
서비스 역할 권한: CodeDeployToECS 작업	1007
작업 선언	1009
다음 사항도 참조하세요	
Amazon Elastic Container Service(Amazon ECS) 배포 작업 참조	1011
작업 유형	1012
구성 파라미터	1013
입력 아티팩트	1013
출력 아티팩트	1014
서비스 역할 권한: Amazon ECS 표준 작업	1014
작업 선언	1016
다음 사항도 참조하세요	1017
Amazon Elastic Kubernetes Service EKS 배포 작업 참조	1017
작업 유형	1018
구성 파라미터	1019
입력 아티팩트	1020
출력 아티팩트	1020
환경 변수	1020
출력 변수	1021
서비스 역할 정책 권한	1021
작업 선언	1023
다음 사항도 참조하세요	
Amazon S3 배포 작업 참조	

작업 유형	1025
구성 파라미터	1025
입력 아티팩트	1026
출력 아티팩트	1027
서비스 역할 권한: S3 배포 작업	1027
예제 작업 구성	1028
다음 사항도 참조하세요	1030
Amazon S3 소스 작업 참조	1031
작업 유형	1032
구성 파라미터	1032
입력 아티팩트	1034
출력 아티팩트	1034
출력 변수	1035
서비스 역할 권한: S3 소스 작업	1035
작업 선언	1036
다음 사항도 참조하세요	1037
AWS AppConfig 배포 작업 참조	1037
작업 유형	1038
구성 파라미터	1038
입력 아티팩트	1039
출력 아티팩트	1039
_ · · · · · · · · · · · · · · · · · · ·	1039
예제 작업 구성	1039
다음 사항도 참조하세요.	
AWS CloudFormation 배포 작업 참조	1041
작업 유형	1042
구성 파라미터	1042
인력 아티팩트	1047
축력 아티팩트	1047
같이 가지 다니프 ···································	1047
ᇀ ' ㄷ ' ································	1048
작업 서어	1040
다음 사항도 착조하세요	1050
AWS CloudFormation StackSets	1050
AWS CloudFormation StackSets 작언 잔동 방신	1057
파이프라이에서 StackSets 작업을 구성하는 반번	1052

CloudFormationStackSet 작업	1055
CloudFormationStackInstances 작업	1068
서비스 역할 권한: CloudFormationStackSet 작업	1077
서비스 역할 권한: CloudFormationStackInstances 작업	1078
스택 세트 작업에 대한 권한 모델	1078
템플릿 파라미터 데이터 유형	1079
다음 사항도 참조하세요	1050
AWS CodeBuild 빌드 및 테스트 작업 참조	1080
작업 유형	1081
구성 파라미터	1081
입력 아티팩트	1084
출력 아티팩트	1084
출력 변수	1085
서비스 역할 권한: CodeBuild 작업	1085
작업 선언(CodeBuild 예제)	1085
다음 사항도 참조하세요	1087
AWS CodePipeline 작업 참조 호출	1088
작업 유형	1088
구성 파라미터	1088
입력 아티팩트	1091
출력 아티팩트	1092
CodePipeline 호출 작업에 대한 서비스 역할 정책 권한	1092
작업 선언	1092
다음 사항도 참조하세요	1093
AWS CodeCommit 소스 작업 참조	1093
작업 유형	1094
구성 파라미터	1095
입력 아티팩트	1096
출력 아티팩트	1096
출력 변수	1096
서비스 역할 권한: CodeCommit 작업	1097
예제 작업 구성	1098
다음 사항도 참조하세요	1100
AWS CodeDeploy 배포 작업 참조	1100
작업 유형	1101
구성 파라미터	1101

입력 아티팩트	1101
출력 아티팩트	1102
서비스 역할 권한: AWS CodeDeploy 작업	. 1102
작업 선언	. 1103
다음 사항도 참조하세요	. 1104
Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com 및 GitLab 자체 관리형 작업용	2
CodeStarSourceConnection	1105
작업 유형	. 1108
구성 파라미터	1108
입력 아티팩트	1109
출력 아티팩트	1110
출력 변수	. 1110
서비스 역할 권한: CodeConnections 작업	1111
작업 선언	. 1111
설치 앱 설치 및 연결 생성	. 1113
다음 사항도 참조하세요	. 1113
명령 작업 참조	. 1114
명령 작업에 대한 고려 사항	1115
서비스 역할 정책 권한	1116
작업 유형	. 1116
구성 파라미터	1117
입력 아티팩트	1121
출력 아티팩트	1121
환경 변수	. 1121
서비스 역할 권한: 명령 작업	1122
작업 선언(예)	. 1123
다음 사항도 참조하세요	. 1124
AWS Device Farm 테스트 작업 참조	1125
작업 유형	. 1125
구성 파라미터	1125
입력 아티팩트	1129
출력 아티팩트	1129
서비스 역할 권한: AWS Device Farm 작업	1130
작업 선언	. 1130
다음 사항도 참조하세요	. 1131
Elastic Beanstalk 배포 작업 참조	1132

작업 유형	1132
구성 파라미터	1133
입력 아티팩트	1133
출력 아티팩트	1133
서비스 역할 권한: ElasticBeanstalk 배포 작업	1133
작업 선언	1134
다음 사항도 참조하세요	1135
Amazon Inspector InspectorScan 호출 작업 참조	1136
작업 유형 ID	1136
구성 파라미터	1137
입력 아티팩트	1139
출력 아티팩트	1139
출력 변수	1139
서비스 역할 권한: InspectorScan 작업	1139
작업 선언	1140
다음 사항도 참조하세요	1141
AWS Lambda 작업 참조 호출	1141
작업 유형	1142
구성 파라미터	1142
입력 아티팩트	1142
출력 아티팩트	1143
출력 변수	1143
예제 작업 구성	1143
예제 JSON 이벤트	1144
다음 사항도 참조하세요	1146
AWS OpsWorks 배포 작업 참조	1147
작업 유형	1147
구성 파라미터	1147
입력 아티팩트	1147
출력 아티팩트	1148
서비스 역할 권한: AWS OpsWorks 작업	1148
예제 작업 구성	1148
다음 사항도 참조하세요	1149
AWS Service Catalog배포 작업 참조	1149
작업 유형	1149
구성 파라미터	1150

입력 아티팩트	1150
출력 아티팩트	1150
서비스 역할 권한: Service Catalog 작업	1151
구성 파일 유형별 작업 구성 예제	. 1151
다음 사항도 참조하세요	. 1153
AWS Step Functions	1153
작업 유형	. 1147
구성 파라미터	1154
입력 아티팩트	1155
출력 아티팩트	1155
출력 변수	. 1155
서비스 역할 권한: StepFunctions 작업	. 1155
예제 작업 구성	. 1156
동작	1159
다음 사항도 참조하세요	. 1041
규칙 구조 참조	. 1162
CloudWatchAlarm	1162
규칙 타입	. 1163
구성 파라미터	1163
규칙 구성의 예	. 1163
다음 사항도 참조하세요	. 1164
CodeBuild 규칙	. 1165
서비스 역할 정책 권한	. 1165
규칙 타입	. 1166
구성 파라미터	1166
규칙 구성의 예	. 1168
다음 사항도 참조하세요	. 1169
명령	. 1169
명령 규칙에 대한 고려 사항	1169
서비스 역할 정책 권한	. 1165
규칙 타입	. 1166
구성 파라미터	1166
규칙 구성의 예	. 1168
다음 사항도 참조하세요	. 1169
DeploymentWindow	1174
규칙 타입	. 1175

구성 파라미터	1175
규칙 구성의 예	1177
다음 사항도 참조하세요	1178
LambdaInvoke	1178
규칙 타입	1163
구성 파라미터	1179
규칙 구성의 예	1179
다음 사항도 참조하세요	1180
VariableCheck	1180
규칙 타입	1180
구성 파라미터	1181
규칙 구성의 예	1183
다음 사항도 참조하세요	1184
통합 모델 참조	1185
타사 작업 유형이 통합자와 함께 작동하는 방식	1185
개념	1186
지원되는 통합 모델	1187
Lambda 통합 모델	1189
CodePipeline의 입력을 처리하도록 Lambda 함수를 업데이트하세요	1189
Lambda 함수에서 CodePipeline에 결과 반환	1193
연속 토큰을 사용하여 비동기 프로세스의 결과를 기다립니다	1195
CodePipeline에 런타임 시 통합자 Lambda 함수를 호출할 수 있는 권한을 제공합니	다 1195
작업자 통합 모델	1195
작업자에 대한 권한 관리 전략 선택 및 구성	1196
이미지 정의 파일 참조	1198
Amazon ECS 표준 배포 작업을 위한 imagedefinitions.json 파일	1198
Amazon ECS 블루/그린 배포 작업을 위한 imageDetail.json 파일	1201
변수 참조	1205
개념	1206
변수	1206
네임스페이스	1207
변수 사용 사례	1208
변수 구성	1208
파이프라인 수준에서 변수 구성	1208
작업 수준에서 변수 구성	1209
변수 확인	1211

변수에 대한 규칙	1212
파이프라인 작업에 사용할 수 있는 변수	1213
정의된 변수 키가 있는 작업	1213
사용자가 구성한 변수 키를 사용한 작업	1217
구문에서 glob 패턴 작업	1220
폴링 파이프라인을 권장되는 변경 감지 방법으로 업데이트	1222
GitHub(OAuth 앱 사용) 소스 작업을 GitHub(GitHub 앱 사용) 소스 작업으로 업데이트	1223
1단계: (OAuth 앱을 통해) GitHub 작업 교체	1224
2단계: GitHub에 대한 연결 생성	1225
3단계: GitHub 소스 작업 저장	1225
할당량	1227
부록 A: GitHub(OAuth 앱 사용) 소스 작업	1242
GitHub(OAuth 앱 사용) 소스 작업 추가	1243
GitHub(OAuth 앱 사용) 소스 작업 참조	1244
작업 유형	1245
구성 파라미터	1245
입력 아티팩트	1246
출력 아티팩트	1247
출력 변수	1247
작업 선언(GitHub 예제)	1248
GitHub에 연결(OAuth)	1249
다음 사항도 참조하세요	1249
문서 기록	1250
이전 업데이트	1276
CodePipeline 기능 참조	1286
	lxxxix

AWS CodePipeline란 무엇인가요?

AWS CodePipeline 는 소프트웨어 릴리스에 필요한 단계를 모델링, 시각화 및 자동화하는 데 사용할 수 있는 지속적 제공 서비스입니다. 소프트웨어 릴리스 프로세스를 구성하는 여러 단계를 신속하게 모 델링하고 구성할 수 있습니다. CodePipeline은 소프트웨어 변경 사항을 지속적으로 릴리스하는 데 필 요한 단계를 자동화합니다. CodePipeline에 대한 요금 정보는 <u>요금</u>을 참조하세요.

주제

- 지속적 제공 및 지속적 통합
- <u>CodePipeline으로 할 수 있는 작업은 무엇인가요?</u>
- 간단히 보는 CodePipeline
- CodePipeline은 어떻게 시작할 수 있나요?
- CodePipeline 개념
- DevOps 파이프라인 예제
- 파이프라인 실행 작동 방식
- 입력 및 출력 아티팩트
- 스테이지 조건은 어떻게 작동하나요?
- 파이프라인 유형
- 나에게 적합한 파이프라인 유형은 무엇인가요?

지속적 제공 및 지속적 통합

CodePipeline은 사용자의 소프트웨어를 프로덕션으로 빌드, 테스트, 배포하는 과정을 자동화하는 지속 적 제공 서비스입니다.

지속적 제공은 릴리스 프로세스가 자동화되는 소프트웨어 개발 방법론입니다. 모든 소프트웨어 변경 이 프로덕션으로 자동 빌드, 테스트 및 배포됩니다. 마지막으로 프로덕션에 푸시하기 전에 사람이나 자 동화된 테스트, 혹은 비즈니스 규칙이 마지막 푸시 발생 시점을 결정합니다. 성공적인 모든 소프트웨어 변경 사항은 즉시 지속적 제공으로 프로덕션에 릴리스되지만, 모든 변경 사항을 즉시 릴리스해야 하는 것은 아닙니다.

<u>지속적 통합</u>은 팀원들이 버전 제어 시스템을 이용하고 기본 브랜치와 같은 동일 위치에 자신들의 업무 를 자주 통합하는 소프트웨어 개발 업무입니다. 각 변경 내용을 빌드한 다음 검증을 거쳐 통합 오류를 신속히 감지합니다. 프로덕션까지 전체 소프트웨어 릴리스 프로세스를 자동화하는 지속적 제공과 비 교할 때 지속적 통합은 코드의 자동 빌드 및 테스트에 주력합니다.

자세한 내용은 <u>지속적인 통합 및 지속적 제공 연습 AWS</u>: DevOps를 통한 소프트웨어 제공 가속화를 참조하세요.

CodePipeline 콘솔, AWS Command Line Interface (AWS CLI), AWS SDKs 또는 이들의 조합을 사용 하여 파이프라인을 생성하고 관리할 수 있습니다.

CodePipeline으로 할 수 있는 작업은 무엇인가요?

CodePipeline을 사용하면 클라우드에서 애플리케이션을 자동으로 빌드하고 테스트하며 배포하는 데 도움이 됩니다. 구체적으로 다음 작업이 가능합니다.

- 릴리스 프로세스 자동화: CodePipeline은 빌드, 테스트, 배포를 통해 소스 리포지토리에서 시작해서 말단에서 말단까지 릴리스 프로세스를 완전히 자동화합니다. Source 단계를 제외하고 모든 단계에 서 수동 승인 작업을 포함시키면 파이프라인을 통해 변경 내용이 이동하지 않도록 방지할 수 있습니 다. 인스턴스 한 개나 여러 개에 원하는 시기에 원하는 방식으로 선택한 시스템에서 릴리스할 수 있 습니다.
- 일관된 릴리스 프로세스 확립: 모든 코드 변경에 일관된 단계 세트를 정의합니다. CodePipeline은 기 준에 따라 릴리스의 각 단계를 실행합니다.
- 품질을 개선하면서 제공 속도 증가: 개발자들이 증분 방식으로 코드를 테스트하고 릴리스할 수 있도
 록 릴리스 프로세스를 자동화하며 새 기능을 고객에게 빨리 선보일 수 있습니다.
- 즐겨 찾는 도구 이용: 기존의 소스와 빌드, 배포 도구를 파이프라인에 통합할 수 있습니다.
 CodePipeline에서 현재 지원하는 AWS 서비스 및 타사 도구의 전체 목록은 섹션을 참조하세 요CodePipeline과 제품 및 서비스 통합.
- 진행 상황 한 눈에 보기: 파이프라인의 실시간 상태를 확인하고, 모든 경고의 세부 사항을 점검하며, 실패한 단계 또는 작업을 재시도하고, 각 스테이지에서 최근 파이프라인 실행에 사용한 소수 수정의 세부 내용을 보고, 모든 파이프라인을 수동으로 다시 실행합니다.
- 파이프라인 기록 세부 정보 보기: 시작 및 종료 시간, 실행 기간 및 실행 ID 등의 파이프라인 실행에 대한 세부 정보를 볼 수 있습니다.

간단히 보는 CodePipeline

다음 다이어그램은 CodePipeline을 이용한 릴리스 프로세스의 예입니다.



이 예에서 개발자가 소스 리포지토리에 변경을 가하면 CodePipeline이 자동으로 변경 내용을 감지합니 다. 그러한 변경 내용을 빌드하고 테스트를 구성하는 경우에는 그 테스트를 실행합니다. 테스트를 마친 후 테스트를 위해 빌드된 코드를 스테이징 서버로 배포합니다. 스테이징 서버에서 CodePipeline은 통 합이나 로드 테스트 같은 추가 테스트를 실행합니다. 테스트가 성공적으로 끝나고 파이프라인에 추가 한 수동 승인 작업에 승인을 받고 나면 CodePipeline이 테스트가 완료된 승인받은 코드를 프로덕션 인 스턴스에 배포합니다.

CodePipeline은 CodeDeploy AWS Elastic Beanstalk또는를 사용하여 EC2 인스턴스에 애플리케이션 을 배포할 수 있습니다 AWS OpsWorks Stacks. CodePipeline은 또한 Amazon ECS를 사용하여 컨테 이너 기반 애플리케이션을 서비스에 배포할 수 있습니다. 개발자는 CodePipeline과 함께 제공되는 통 합 포인트를 이용해 빌드 서비스, 테스트 제공자, 다른 배포 대상이나 시스템 같은 다른 도구나 서비스 를 플러그인할 수 있습니다.

파이프라인은 릴리스 프로세스가 요구하는 대로 단순하거나 복잡해집니다.

CodePipeline은 어떻게 시작할 수 있나요?

CodePipeline을 시작하려면

- 1. <u>CodePipeline 개념</u> 섹션을 읽고 CodePipeline의 작동 방식을 알아봅니다.
- 2. <u>CodePipeline 시작하기</u>의 단계에 따라 CodePipeline을 사용할 준비를 하세요.
- 3. <u>CodePipeline 자습서</u> 자습서의 단계에 따라 CodePipeline을 실험하세요.
- 4. <u>파이프라인 스테이지 및 작업 생성</u>의 단계에 따라 새 프로젝트 또는 기존 프로젝트에 CodePipeline 을 사용하세요.

CodePipeline 개념

에 사용된 개념과 용어를 이해하면 자동 릴리스 프로세스를 모델링하고 구성하는 것이 더 쉽습니다 AWS CodePipeline. 다음은 CodePipeline 사용에 관해 알아야 할 개념입니다.

DevOps 파이프라인의 예는 DevOps 파이프라인 예제 단원을 참조하십시오.

CodePipeline에서 사용되는 용어는 다음과 같습니다.

주제

- Pipelines
- 파이프라인 실행
- 스테이지 작업
- 작업 실행
- <u>실행 유형</u>
- 작업 유형
- 아티팩트
- 소스 개정
- <u>트리거</u>
- 변수
- 조건
- 규칙

Pipelines

파이프라인이란 소프트웨어 변경 사항이 릴리스 프로세스에 적용되는 방법을 설명하는 워크플로우 구 성입니다. 각 파이프라인은 일련의 단계로 구성됩니다.

Stages

단계는 환경을 격리하고 해당 환경에서 동시 변경 작업의 수를 제한하는 데 사용할 수 있는 논리 단위 입니다. 각 단계에는 애플리케이션 <u>아티팩트</u>에 대해 수행되는 작업이 포함됩니다. 소스 코드는 아티팩 트의 한 예입니다. 소스 코드가 빌드되고 테스트가 실행되는 빌드 단계가 될 수 있습니다. 또한 코드가 런타임 환경에 배포되는 배포 단계가 될 수도 있습니다. 각 단계는 일련의 직렬 또는 병렬 작업으로 구 성됩니다.

Transitions

전환은 파이프라인 실행이 파이프라인의 다음 단계로 이동하는 지점입니다. 단계의 인바운드 전환을 비활성화하여 실행이 해당 단계로 들어가지 못하도록 한 다음, 전환을 활성화하여 실행을 계속할 수 있 습니다. 둘 이상의 실행이 비활성화된 전환에 도달하면 가장 최신의 실행만 다음 단계로 계속되어 전환 이 활성화됩니다. 즉, 전환이 비활성화된 동안 보다 최신의 실행이 대기 중인 실행을 계속 대체하고 전 환이 활성화된 후에는 계속 진행되는 실행이 대체 실행이 됩니다.



작업

작업은 애플리케이션 코드에서 수행되고 지정된 지점에 파이프라인에서 작업이 실행되도록 구성된 일 련의 작업입니다. 여기에는 코드 변경의 소스 작업, 애플리케이션을 인스턴스에 배포하는 작업 등이 포 함될 수 있습니다. 예를 들어 배포 단계에는 Amazon EC2 또는 AWS Lambda과 같은 컴퓨팅 서비스에 코드를 배포하는 배포 작업이 포함될 수 있습니다.

유효한 CodePipeline 작업 유형은 source, build, test, deploy, approval 및 invoke입니다. 작 업 공급자 목록은 CodePipeline에서 유효한 작업 공급자 단원을 참조하십시오.

작업은 직렬 또는 병렬로 실행할 수 있습니다. 단계의 직렬 및 병렬 작업에 대한 자세한 내용은 <u>작업 구</u> 조 요구 사항의 run0rder 정보를 참조하세요.

파이프라인 실행

실행은 파이프라인에서 릴리스된 변경 집합입니다. 각 파이프라인 실행은 고유한 자체 ID를 가집니다. 실행은 병합된 커밋 또는 최신 커밋의 수동 릴리스와 같은 변경 집합에 해당합니다. 두 실행은 서로 다 른 시간에 동일한 변경 집합을 릴리스할 수 있습니다.

파이프라인은 동시에 여러 실행을 처리할 수 있지만 파이프라인 단계는 한 번에 하나의 실행만 처리합 니다. 이를 위해 실행을 처리하는 동안 단계가 잠깁니다. 두 파이프라인 실행이 동시에 같은 단계를 차 지할 수는 없습니다. 점유 단계에 진입하기 위해 대기 중인 실행을 인바운드 실행이라고 합니다. 인바 운드 실행은 여전히 실패하거나, 대체되거나, 수동으로 중지될 수 있습니다. 인바운드 실행 작동 방식 에 대한 자세한 내용은 인바운드 실행 작동 방식 단원을 참조하세요.

파이프라인 실행은 파이프라인 단계를 순서대로 순회합니다. 파이프라인의 유효한 상태는 InProgress, Stopping, Stopped, Succeeded, Superseded 및 Failed입니다.

자세한 내용은 PipelineExecution을 참조하세요.

중지된 실행

진행 중인 파이프라인 실행이 파이프라인을 통해 계속되지 않도록 파이프라인 실행을 수동으로 중지 할 수 있습니다. 수동으로 중지한 경우 파이프라인 실행은 완전히 중지될 때까지 Stopping 상태로 표 시됩니다. 그런 다음 Stopped 상태로 표시됩니다. Stopped 파이프라인 실행을 다시 시도할 수 있습 니다.

파이프라인 실행을 중지하는 방법은 두 가지입니다.

- 중지 및 대기
- 중지 및 중단

실행 중지의 사용 사례에 대한 자세한 내용과 이러한 옵션에 대한 순서 세부 정보는 <u>파이프라인 실행을</u> 중지하는 방법 단원을 참조하십시오.

실패한 실행

실행이 실패하면 작업이 중지되고 파이프라인을 완전히 순회하지 못합니다. 상태는 FAILED 상태가 되 고 단계는 잠금이 해제됩니다. 보다 최근의 실행은 잠금 해제된 단계를 파악해 잠글 수 있습니다. 실패 한 실행이 대체되었거나 다시 시도할 수 없는 경우가 아니면 실패한 실행을 재시도할 수 있습니다. 실 패한 스테이지를 이전에 성공한 실행으로 롤백할 수 있습니다.

실행 모드

파이프라인을 통해 최신 변경 집합을 전달하기 위해 보다 최근의 실행이 파이프라인을 통해 이미 실행 중인 덜 최근의 실행을 전달 및 대체합니다. 이 경우 이전 실행은 새로운 실행으로 대체됩니다. 실행은 단계 사이의 특정 지점에서 보다 최근의 실행으로 대체될 수 있습니다. SUPERSEDED는 기본 실행 모 드입니다.

SUPERSEDED 모드에서 실행이 잠금 상태의 단계로 진입하기 위해 대기 중인 경우, 보다 최근의 실행 이 이를 파악해 대체할 수 있습니다. 이제 보다 최신의 실행이 단계가 잠금 해제될 때까지 대기하고, 대 체된 실행은 SUPERSEDED 상태로 중지됩니다. 파이프라인 실행이 대체되면 실행이 중지되고 파이프 라인을 완전히 순회하지 않게 됩니다. 이 단계에서 대체된 실행은 이후에 더 이상 재시도할 수 없습니 다. 사용 가능한 다른 실행 모드는 PARALLEL 또는 QUEUED 모드입니다.

실행 모드 및 잠긴 스테이지에 대한 자세한 내용은 <u>SUPERSEDED 모드에서 실행을 처리하는 방법</u> 섹 션을 참조하세요.

스테이지 작업

파이프라인 실행이 스테이지를 통해 실행될 때 스테이지는 그 안의 모든 작업을 완료하는 과정에 있습 니다. 스테이지 작동 방식 및 잠긴 스테이지에 대한 자세한 내용은 <u>SUPERSEDED 모드에서 실행을 처</u> 리하는 방법 섹션을 참조하세요.

단계의 유효한 상태는 InProgress, Stopping, Stopped, Succeeded 및 Failed입니다. 실패 한 단계를 재시도할 수 없는 경우가 아니면 실패한 단계를 재시도할 수 있습니다. 자세한 내용은 <u>StageExecution</u>을 참조하세요. 스테이지를 지정된 이전에 성공한 실행으로 롤백할 수 있습니다. <u>스테</u> 이지 롤백 구성에 설명된 대로 실패 시 자동으로 롤백하도록 스테이지를 구성할 수 있습니다. 자세한 내용은 RollbackStage를 참조하세요.

작업 실행

작업 실행은 지정된 <u>아티팩트</u>에서 작동하는 구성된 작업을 완료하는 프로세스입니다. 입력 아티팩트, 출력 아티팩트 또는 둘 다가 여기에 해당될 수 있습니다. 예를 들어 빌드 작업은 애플리케이션 소스 코 드 컴파일과 같은 입력 아티팩트에서 빌드 명령을 실행할 수 있습니다. 작업 실행 세부 정보에는 작업 실행 ID, 관련된 파이프라인 실행 소스 트리거, 작업의 입력 및 출력 아티팩트가 포함됩니다.

작업에 대한 유효한 상태는 InProgress, Abandoned, Succeeded 또는 Failed입니다. 자세한 내 용은 <u>ActionExecution</u>을 참조하세요.

실행 유형

파이프라인 또는 스테이지 실행은 표준 또는 롤백 실행일 수 있습니다.

표준 유형의 경우 실행에는 고유한 ID가 있으며 전체 파이프라인 실행입니다. 파이프라인 롤백에는 롤 백할 스테이지와 롤백할 대상 실행으로 스테이지에 대한 성공적인 실행이 있습니다. 대상 파이프라인 실행은 재실행할 스테이지의 소스 개정 및 변수를 검색하는 데 사용됩니다.

작업 유형

작업 유형은 CodePipeline에서 선택할 수 있는 사전 구성된 작업입니다. 작업 유형은 소유자, 공급자, 버전 및 범주에 따라 정의됩니다. 작업 유형은 파이프라인에서 작업 태스크를 완료하는 데 사용되는 사 용자 지정 파라미터를 제공합니다.

작업 유형에 따라 파이프라인에 통합할 수 있는 AWS 서비스 및 타사 제품 및 서비스에 대한 자세한 내 용은 섹션을 참조하세요CodePipeline 작업 유형과의 통합.

CodePipeline의 작업 유형에 지원되는 통합 모델에 대한 자세한 내용은 통합 모델 참조을 참조하세요.

타사 공급자가 CodePipeline에서 작업 유형을 설정하고 관리하는 방법에 대한 자세한 내용은 <u>작업 유</u> <u>형 처리</u>을 참조하세요.

아티팩트

아티팩트는 애플리케이션 소스 코드, 빌드된 애플리케이션, 종속성, 정의 파일, 템플릿 등과 같이 파이 프라인 작업을 통해 만들어진 데이터 컬렉션을 의미합니다. 아티팩트는 일부 작업에서 생성되고 다른 작업에서 사용됩니다. 파이프라인에서 아티팩트는 작업에서 만들어진 파일 집합(입력 아티팩트)이거 나 완료된 작업의 업데이트 출력(출력 아티팩트)이 될 수 있습니다.

작업은 파이프라인 아티팩트 버킷을 사용하여 추가 처리를 위해 출력을 다른 작업으로 전달합니다. CodePipeline이 아티팩트를 아티팩트 스토어에 복사하면 작업이 이들을 선택합니다. 아티팩트에 대한 자세한 내용은 입력 및 출력 아티팩트 섹션을 참조하세요.

소스 개정

소스 코드를 변경하면 새 버전이 생성됩니다. 소스 개정은 파이프라인 실행을 트리거하는 소스 변경의 버전입니다. 실행은 소스 개정을 처리합니다. GitHub 및 CodeCommit 리포지토리의 경우 커밋이 여기 에 해당됩니다. S3 버킷 또는 작업의 경우 객체 버전이 여기에 해당됩니다. 지정한 커밋과 같은 소스 개정으로 파이프라인 실행을 시작할 수 있습니다. 실행 시 지정된 개정이 처 리되고 실행에 사용된 개정이 재정의됩니다. 자세한 내용은 <u>소스 개정 재정의로 파이프라인 시작</u> 단원 을 참조하십시오.

트리거

트리거는 파이프라인을 시작하는 이벤트입니다. 파이프라인 수동 시작과 같은 일부 트리거는 파이프 라인의 모든 소스 작업 공급자가 사용할 수 있습니다. 특정 트리거는 파이프라인의 소스 공급자에 따라 달라집니다. 예를 들어, 이벤트 규칙의 대상으로 파이프라인 ARN이 추가된 Amazon CloudWatch의 이 벤트 리소스로 CloudWatch 이벤트를 구성해야 합니다. Amazon CloudWatch Events는 CodeCommit 또는 S3 소스 작업을 사용하는 파이프라인의 자동 변경 감지를 위한 권장 트리거입니다. Webhook는 타사 리포지토리 이벤트용으로 구성된 트리거 유형입니다. 예를 들어 WebhookV2는 Git 태그를 사용 하여 GitHub.com, GitHub Enterprise Server, GitLab.com, GitLab 자체 관리형 또는 Bitbucket Cloud와 같은 타사 소스 공급자와 파이프라인을 시작하는 데 사용할 수 있는 트리거 유형입니다. 파이프라인 구 성에서 푸시 또는 풀 요청과 같은 트리거에 대한 필터를 지정할 수 있습니다. Git 태그, 브랜치 또는 파 일 경로에서 코드 푸시 이벤트를 필터링할 수 있습니다. 이벤트(열림, 업데이트, 닫힘), 브랜치 또는 파 일 경로에 대한 풀 요청 이벤트를 필터링할 수 있습니다.

트리거에 대한 자세한 내용은 <u>CodePipeline에서 파이프라인 시작</u> 주제를 참조하세요. Git 태그를 파이 프라인의 트리거로 사용하는 방법을 안내하는 자습서는 <u>자습서: Git 태그를 사용하여 파이프라인 시작</u> 하기을 참조하세요.

A Important

30일 이상 비활성 상태인 파이프라인은 파이프라인에 대해 폴링이 비활성화됩니다. 자세한 내 용은 파이프라인 구조 참조의 <u>pollingDisabledAt</u>를 참조하세요. 파이프라인을 폴링에서 이벤트 기반 변경 감지로 마이그레이션하는 단계는 변경 감지 방법을 참조하세요.

변수

변수는 파이프라인에서 작업을 동적으로 구성하는 데 사용할 수 있는 값입니다. 변수는 파이프라인 수 준에서 선언하거나 파이프라인의 작업에 의해 내보낼 수 있습니다. 변수 값은 파이프라인 실행 시 확인 되며 실행 기록에서 확인할 수 있습니다. 파이프라인 수준에서 선언된 변수의 경우 파이프라인 구성에 서 기본값을 정의하거나 지정된 실행에 대해 기본값을 재정의할 수 있습니다. 작업에서 내보낸 변수의 경우 작업이 성공적으로 완료된 후에 값을 사용할 수 있습니다. 자세한 내용은 <u>변수 참조</u> 단원을 참조 하십시오.

조건

조건에는 평가되는 규칙 세트가 포함됩니다. 조건의 모든 규칙이 성공하면 조건이 충족됩니다. 기준이 충족되지 않으면 스테이지 실패와 같은 지정된 결과가 적용되도록 조건을 구성할 수 있습니다. 조건은 실행이 스테이지를 통해 시작 및 실행되는 시점을 지정하거나 실행 후 스테이지를 종료할 시점을 지정 할 수 있기 때문에 게이트라고도 합니다. 이는 도로의 차량 행렬이 닫힌 게이트에 모이도록 한 다음 게 이트의 개방을 지정하여 특정 지역으로의 차량 흐름을 허용하는 것과 유사합니다. 조건 유형의 결과에 는 스테이지 실패 또는 스테이지 롤백이 포함됩니다. 조건은 파이프라인 단계에서 이러한 작업이 발생 하는 시기를 지정하는 데 도움이 됩니다. 런타임 시 조건을 재정의할 수 있습니다.

세 가지 유형의 조건이 있습니다. 입력 조건은 "조건에 대한 규칙이 충족되면 스테이지에 들어갑니 다."라는 질문에 답합니다. 실행이 스테이지에 들어가면 스테이지가 잠긴 다음 규칙이 실행됩니다. 실 패 시 조건의 경우, 실패한 스테이지를 롤백한 결과와 함께 스테이지가 실패하면 규칙이 활성화됩니다. 성공 시 조건의 경우 진행하기 전에 경보가 성공적으로 실행되었는지 확인하는 등 스테이지가 성공하 면 규칙이 활성화됩니다. 예를 들어, CloudWatchAlarm 규칙이 배포 환경에 경보가 있는 것을 확인하는 경우 성공 시 조건이 성공 스테이지를 롤백하는 결과로 이어집니다. 자세한 내용은 <u>스테이지 조건은 어</u> 떻게 작동하나요? 단원을 참조하십시오.

규칙

조건은 하나 이상의 사전 구성된 규칙을 사용하여 조건이 충족되지 않을 때 구성된 결과를 포함하는 검 사를 실행하고 수행합니다. 예를 들어 경보 상태 및 배포 창 시간을 확인하는 입력 조건 규칙에 대한 모 든 규칙을 충족하면 모든 검사가 통과된 후 성공 스테이지가 배포됩니다. 자세한 내용은 <u>스테이지 조건</u> <u>은 어떻게 작동하나요?</u> 단원을 참조하십시오.

DevOps 파이프라인 예제

DevOps 파이프라인의 예제로, 2단계 파이프라인에는 소스라는 소스 단계와 프로덕션이라는 두 번째 단계가 있을 수 있습니다. 이 예제에서는 파이프라인이 애플리케이션을 최신 변경 사항으로 업데이 트하고 최신 결과를 지속적으로 배포합니다. 최신 애플리케이션을 배포하기 전에 파이프라인은 웹 애 플리케이션을 빌드 및 테스트합니다. 이 예제에서 개발자 그룹은 인프라 템플릿과 MyRepository라는 GitHub 리포지토리의 웹 애플리케이션용 소스 코드를 설정했습니다.



예를 들어 개발자가 웹 애플리케이션의 인덱스 페이지에 수정 사항을 푸시하면 다음과 같은 상황이 발 생합니다.
- 애플리케이션 소스 코드는 파이프라인에서 GitHub 소스 작업으로 구성된 리포지토리에서 유지 관 리됩니다. 개발자가 커밋을 리포지토리로 푸시하면 CodePipeline은 푸시된 변경 사항을 감지하고 소스 단계에서 파이프라인 실행이 시작됩니다.
- 2. GitHub 소스 작업이 성공적으로 완료됩니다(즉, 최신 변경 사항이 해당 실행에 고유한 아티팩트 버 킷에 다운로드되어 저장됨). 리포지토리에서 나온 애플리케이션 파일인 GitHub 소스 작업에서 생성 된 출력 아티팩트는 다음 단계의 작업에서 처리될 입력 아티팩트로 사용됩니다.
- 파이프라인 실행은 Source Stage(소스 단계)에서 Prod Stage(프로덕션 단계)로 전환됩니다. 프로덕 션 단계의 첫 번째 작업은 CodeBuild에서 생성되고 파이프라인에서 빌드 작업으로 구성된 빌드 프 로젝트를 실행합니다. 빌드 작업은 빌드 환경 이미지를 가져오며 가상 컨테이너에서 웹 애플리케이 션을 빌드합니다.
- 4. 프로덕션 단계의 다음 작업은 CodeBuild에서 생성되고 파이프라인에서 테스트 작업으로 구성된 단 위 테스트 프로젝트입니다.
- 5. 다음 단계로 단위 테스트 코드는 프로덕션 환경에 애플리케이션을 배포하는 Prod Stage(프로덕션 단계)의 배포 작업에서 처리됩니다. 배포 작업이 성공적으로 완료되고 난 후 해당 단계의 최종 작업 은 CodeBuild에서 생성되고 파이프라인에서 테스트 작업으로 구성된 통합 테스트 프로젝트입니다. 테스트 작업은 웹 애플리케이션에서 링크 검사기와 같은 테스트 도구를 설치하고 실행하는 셸 스크 립트를 호출합니다. 성공적으로 완료가 된 후에는 빌드된 웹 애플리케이션 및 테스트 결과 집합이 출력됩니다.

개발자는 각 변경 사항에 대해 빌드 및 테스트를 수행한 이후에 애플리케이션을 배포하거나 추가로 테 스트하는 작업을 파이프라인에 추가할 수 있습니다.

자세한 내용은 파이프라인 실행 작동 방식 단원을 참조하십시오.

파이프라인 실행 작동 방식

이 섹션에서는 CodePipeline이 일련의 변경 사항을 처리하는 방법을 간략히 설명합니다. CodePipeline 은 파이프라인을 수동으로 시작하거나 소스 코드에 변경 사항이 발생할 때 시작되는 각 파이프라인 실 행을 추적합니다. CodePipeline은 다음 실행 모드를 사용하여 각 실행이 파이프라인을 통해 진행되는 방식을 처리합니다. 자세한 내용은 파이프라인 실행 모드 설정 또는 변경 단원을 참조하십시오.

- SUPERSEDED 모드: 보다 최근의 실행이 이전 실행을 추월할 수 있습니다. 이 값이 기본값입니다.
- QUEUED 모드: 실행은 대기열에 있는 순서대로 하나씩 처리됩니다. 이를 위해서는 파이프라인 유형 V2가 필요합니다.

 PARALLEL 모드: PARALLEL 모드에서 실행은 서로 독립적으로 동시에 실행됩니다. 실행은 시작 또 는 완료 전에 다른 실행이 완료될 때까지 기다리지 않습니다. 이를 위해서는 파이프라인 유형 V2가 필요합니다.

A Important

PARALLEL 모드의 파이프라인의 경우 스테이지 롤백을 사용할 수 없습니다. 마찬가지로 롤 백 결과 유형이 있는 실패 조건은 PARALLEL 모드 파이프라인에 추가할 수 없습니다.

파이프라인 실행 시작 방법

소스 코드를 변경하거나 파이프라인을 수동으로 시작할 때 실행을 시작할 수 있습니다. 일정을 잡는 Amazon CloudWatch Events 규칙을 통해 실행을 트리거할 수도 있습니다. 예를 들어 소스 코드 변경 이 파이프라인의 소스 작업으로 구성된 리포지토리로 푸시되면 파이프라인은 변경 사항을 감지하고 실행을 시작합니다.

Note

파이프라인에 여러 개의 소스 작업이 포함되어 있는 경우, 소스 작업 한 개에만 변경이 감지되 더라도 모든 소스 작업이 다시 실행됩니다.

파이프라인 실행에서 소스 개정이 처리되는 방법

소스 코드 변경(소스 개정)으로 시작하는 각 파이프라인 실행에 대해 소스 개정은 다음과 같이 결정됩 니다.

 CodeCommit 소스가 있는 파이프라인의 경우 커밋이 푸시되는 시점에 CodePipeline에 의해 HEAD 가 복제됩니다. 예를 들어 커밋이 푸시되면 실행 1의 파이프라인이 시작됩니다. 두 번째 커밋이 푸시 되는 시점에 실행 2의 파이프라인이 시작됩니다.

Note

CodeCommit 소스가 있는 PARALLEL 모드의 파이프라인의 경우 파이프라인 실행을 트 리거한 커밋에 관계없이 소스 작업은 시작 시 항상 HEAD를 복제합니다. 자세한 내용은 PARALLEL 모드의 CodeCommit 또는 S3 소스 개정이 EventBridge 이벤트와 일치하지 않을 수 있습니다. 단원을 참조하십시오. S3 소스가 있는 파이프라인의 경우 S3 버킷 업데이트에 대한 EventBridge 이벤트가 사용됩니다. 예 를 들어 소스 버킷에서 파일이 업데이트되면 이벤트가 생성되어 실행 1의 파이프라인이 시작됩니다. 두 번째 버킷 업데이트에 대한 이벤트가 수행되는 시점에 실행 2에 대한 파이프라인이 시작됩니다.

Note

S3 소스가 있는 PARALLEL 모드의 파이프라인의 경우 실행을 트리거한 이미지 태그와 관계 없이 소스 작업은 항상 최신 이미지 태그로 시작됩니다. 자세한 내용은 <u>PARALLEL 모드의</u> <u>CodeCommit 또는 S3 소스 개정이 EventBridge 이벤트와 일치하지 않을 수 있습니다.</u> 단원 을 참조하십시오.

 Bitbucket과 같은 연결 소스가 있는 파이프라인의 경우 커밋이 푸시되는 시점에 CodePipeline에 의 해 HEAD가 복제됩니다. 예를 들어 PARALLEL 모드의 파이프라인의 경우 커밋이 푸시되어 실행 1을 위한 파이프라인이 시작되고 두 번째 파이프라인 실행은 두 번째 커밋을 사용합니다.

파이프라인 실행을 중지하는 방법

콘솔을 사용하여 파이프라인 실행을 중지하려면 파이프라인 시각화 페이지, 실행 기록 페이지 또는 세 부 기록 페이지에서 실행 중지를 선택할 수 있습니다. CLI를 사용하여 파이프라인 실행을 중지하려면 stop-pipeline-execution 명령을 사용합니다. 자세한 내용은 <u>CodePipeline에서 파이프라인 실행</u> 중지 단원을 참조하십시오.

파이프라인 실행을 중지하는 방법은 두 가지입니다.

- 중지 및 대기: 진행 중인 모든 작업 실행을 완료할 수 있으며 후속 작업이 시작되지 않습니다. 파이프 라인 실행은 후속 단계로 계속되지 않습니다. 이미 Stopping 상태인 실행에는 이 옵션을 사용할 수 없습니다.
- 중지 및 중단: 진행 중인 모든 작업 실행이 중단되고 완료되지 않으며 후속 작업이 시작되지 않습니
 다. 파이프라인 실행은 후속 단계로 계속되지 않습니다. 이미 Stopping 상태인 실행에 이 옵션을 사용할 수 있습니다.

1 Note

이 옵션을 선택하면 작업이 실패하거나 작업 순서가 뒤바뀔 수 있습니다.

각 옵션에서 파이프라인 및 작업 실행 단계의 순서가 다음과 같이 달라집니다.

옵션 1: 중지 및 대기

중지하고 대기하도록 선택하면 진행 중인 작업이 완료될 때까지 선택한 실행이 계속됩니다. 예를 들어, 빌드 작업이 진행되는 동안 다음 파이프라인 실행이 중지되었습니다.

 파이프라인 보기에 성공 메시지 배너가 표시되고 빌드 작업이 완료될 때까지 계속됩니다. 파이프라 인 실행 상태는 중지 중입니다.

기록 보기에서 빌드 작업과 같은 진행 중인 작업의 상태는 빌드 작업이 완료될 때까지 진행 중입니 다. 작업이 진행 중인 동안 파이프라인 실행 상태는 중지 중입니다.

 중지 프로세스가 완료되면 실행이 중지됩니다. 빌드 작업이 성공적으로 완료되면 상태는 성공이고 파이프라인 실행은 중지됨 상태로 표시됩니다. 후속 작업이 시작되지 않습니다. 재시도 버튼이 활성 화됩니다.

Execution history Info			Stop execution View		
Q					
				< 1 2	
	Execution ID	Status	Source revisions	Duration	
\sim	e3be8a77-	O Stanged	Source –	1 minute 9	
0		O stopped	Added main.css	seconds	

진행 중인 작업이 완료된 후 기록 보기에서 실행 상태는 중지됨입니다.

옵션 2: 중지 및 중단

중지하고 중단하도록 선택하면 선택한 실행은 진행 중인 작업이 완료될 때까지 대기하지 않습니다. 작 업은 중단됩니다. 예를 들어, 빌드 작업이 진행 중인 동안 다음 파이프라인 실행이 중지되고 중단되었 습니다.

 파이프라인 보기에 성공 배너 메시지가 표시되고 빌드 작업은 진행 중 상태로 표시되며 파이프라인 실행은 중지 중 상태로 표시됩니다.

- 파이프라인 실행이 중지된 후 빌드 작업은 중단됨 상태로 표시되고 파이프라인 실행은 중지됨 상태 로 표시됩니다. 후속 작업이 시작되지 않습니다. 재시도 버튼이 활성화됩니다.
- 3. 기록 보기에서 실행 상태는 중지됨입니다.

Execution history Info			Stop execution View d		
Q					
					< 1
	Execution ID	Status	Source revisions	Duration	Completed
0	bf3dc924-	⊖ Stopped	Source – Added main.css	22 seconds	Jan 16, 2020 8:28 AM (UTC-8:00)

파이프라인 실행 중지의 사용 사례

중지 및 대기 옵션을 사용하여 파이프라인 실행을 중지하는 것이 좋습니다. 이 옵션은 파이프라인에서 작업이 실패하거나 작업 순서가 뒤바뀌는 것을 방지하기 때문에 더 안전합니다. CodePipeline에서 작 업이 중단되면 작업 공급자는 해당 작업과 관련된 모든 작업을 계속합니다. AWS CloudFormation 작업 의 경우 파이프라인의 배포 작업이 중단되지만 스택 업데이트가 계속되어 업데이트가 실패할 수 있습 니다.

작업 순서가 뒤바뀔 수 있는 중단된 작업의 예로, S3 배포 작업을 통해 대용량 파일(1GB)을 배포하는 경우 배포가 이미 진행 중인 동안 작업을 중지하고 중단하도록 선택하면 작업은 CodePipeline에서 중 단되지만 Amazon S3에서 계속됩니다. Amazon S3에서는 업로드를 취소하라는 지시가 발생하지 않습 니다. 그런 다음, 아주 작은 파일로 새 파이프라인 실행을 시작하는 경우 이제 두 개의 배포가 진행 중입 니다. 새 실행의 파일 크기가 작기 때문에 이전 배포가 여전히 업로드되는 동안 새 배포가 완료됩니다. 이전 배포가 완료되면 이전 파일이 새 파일을 덮어씁니다.

사용자 지정 작업이 있는 사례에서 중지 및 중단 옵션을 사용하려고 할 수 있습니다. 예를 들어, 버그 수 정을 위한 새 실행을 시작하기 전에 완료할 필요 없는 작업이 포함된 사용자 지정 작업을 중단할 수 있 습니다.

SUPERSEDED 모드에서 실행을 처리하는 방법

실행을 처리하는 기본 모드는 SUPERSEDED 모드입니다. 실행은 해당 실행에서 선택되고 처리되는 일련의 변경 사항으로 구성됩니다. 파이프라인은 동시에 여러 실행을 처리할 수 있습니다. 각 실행은 별도로 파이프 라인을 통해 실행됩니다. 파이프라인은 각 실행을 순서대로 처리하며 이전 실행을 이후 실행으로 대체할 수 있습니다. 다음 규칙은 SUPERSEDED 모드의 파이프라인에서 실행을 처리하는 데 사용됩니다.

규칙 1: 실행이 처리 중일 때 단계가 잠깁니다.

각 단계는 한 번에 하나의 실행만 처리할 수 있으므로 진행 중인 동안에는 단계가 잠깁니다. 실행이 단 계를 완료하면 파이프라인의 다음 단계로 전환됩니다.



이전: Stage 1 is locked as Execution 1 enters. 이후: Stage 2 is locked as Execution 1 enters.

규칙 2: 후속 실행은 단계가 잠금 해제될 때까지 기다립니다.

단계가 잠겨 있는 동안 대기 중인 실행은 잠긴 단계 앞에 유지됩니다. 단계가 완료된 것으로 간주되기 전에 한 단계에 구성한 모든 작업을 성공적으로 끝내야 합니다. 실패하면 단계에서 잠금이 해제됩니다. 실행이 중지되면 단계에서 실행이 계속되지 않고 단계가 잠금 해제됩니다.

Note

실행을 중지하기 전에 단계 앞의 전환을 비활성화하는 것이 좋습니다. 이렇게 하면 실행 중지 로 인해 단계가 잠금 해제될 때 단계에서 후속 파이프라인 실행이 허용되지 않습니다.





규칙 3: 대기 중인 실행이 더 최근의 실행으로 대체됩니다.

실행은 단계 사이에서만 대체됩니다. 잠긴 단계에서는 단계가 완료되기를 기다리는 단계의 앞에 실행 이 유지됩니다. 보다 최근의 실행이 대기 중인 실행을 추월하고 단계의 잠금이 해제되는 즉시 다음 단 계로 계속됩니다. 대체된 실행은 계속되지 않습니다. 이 예제에서는 실행 2가 잠긴 단계를 기다리는 동 안 실행 3으로 대체되었습니다. 실행 3은 다음 단계에 들어갑니다.



이전: 실행 3이 단계 1에 들어가는 동안 실행 2는 단계 사이에서 대기합니다. 이후: 실행 3이 단계 1을 종료합니다. 실행 2는 실행 3으로 대체됩니다.

실행 모드 간의 보기 및 전환 고려 사항에 대한 자세한 내용은 <u>파이프라인 실행 모드 설정 또는 변경</u> 섹 션을 참조하세요. 실행 모드가 있는 할당량에 대한 자세한 내용은 <u>in AWS CodePipeline 할당량</u> 섹션을 참조하세요.

QUEUED 모드에서 실행을 처리하는 방법

QUEUED 모드의 파이프라인의 경우 실행이 처리될 때 스테이지가 잠기지만, 대기 실행은 이미 시작된 실행을 추월하지 않습니다.

대기 실행은 스테이지에 도달하는 순서대로 잠긴 스테이지 진입점에서 수집되어 대기 실행 대기열 을 형성합니다. QUEUED 모드를 사용하면 동일한 파이프라인에 여러 대기열이 있을 수 있습니다. 대 기 중인 실행이 스테이지에 들어가면 스테이지가 잠기고 다른 실행은 입력할 수 없습니다. 이 동작은 SUPERSEDED 모드와 동일하게 유지됩니다. 실행이 스테이지를 완료하면 스테이지가 잠금 해제되고 다음 실행을 위해 준비됩니다.

다음 다이어그램은 QUEUED 모드 파이프라인의 스테이지가 실행을 처리하는 방법을 보여줍니다. 예 를 들어 소스 스테이지에서 실행 5를 처리하는 동안 6 및 7에 대한 실행은 대기열 #1을 형성하고 스테 이지 진입점에서 기다립니다. 대기열의 다음 실행은 스테이지가 잠금 해제된 후 처리됩니다.



MyPipeline

Note: maximum of 50 concurrent executions per pipeline

실행 모드 간의 보기 및 전환 고려 사항에 대한 자세한 내용은 <u>파이프라인 실행 모드 설정 또는 변경</u> 섹 션을 참조하세요. 실행 모드가 있는 할당량에 대한 자세한 내용은 <u>in AWS CodePipeline 할당량</u> 섹션을 참조하세요.

PARALLEL 모드에서 실행을 처리하는 방법

PARALLEL 모드의 파이프라인의 경우 실행은 서로 독립적이며 시작하기 전에 다른 실행이 완료될 때 까지 기다리지 않습니다. 대기열은 없습니다. 파이프라인에서 병렬 실행을 보려면 실행 기록 보기를 사 용합니다.

각 기능에 고유한 기능 브랜치가 있고 다른 사용자가 공유하지 않는 대상에 배포되는 개발 환경에서 PARALLEL 모드를 사용합니다.

실행 모드 간의 보기 및 전환 고려 사항에 대한 자세한 내용은 <u>파이프라인 실행 모드 설정 또는 변경</u> 섹 션을 참조하세요. 실행 모드가 있는 할당량에 대한 자세한 내용은 <u>in AWS CodePipeline 할당량</u> 섹션을 참조하세요.

파이프라인 흐름 관리

파이프라인 실행의 흐름은 다음을 통해 제어할 수 있습니다.

 전환을 통해 단계로의 실행 흐름을 제어합니다. 전환은 활성화 또는 비활성화가 가능합니다. 전환이 비활성화되면 파이프라인 실행이 단계에 들어갈 수 없습니다. 전환이 비활성화된 단계에 들어가기 위해 기다리는 파이프라인 실행을 인바운드 실행이라고 합니다. 전환을 활성화하면 인바운드 실행 이 단계 이동하여 잠깁니다.

잠긴 단계를 기다리는 실행과 마찬가지로 전환이 비활성화된 경우에도 단계에 들어가기를 기다리는 실행은 여전히 새 실행으로 대체될 수 있습니다. 비활성화된 전환이 다시 활성화되면 전환이 비활성 화된 상태에서 이전 실행을 대체한 최신 실행이 해당 단계로 들어갑니다.

 승인 작업은 권한이 허용될 때까지 파이프라인이 다음 작업으로 전환하지 않도록 합니다(예를 들면 권한이 있는 자격 증명에게 수동 승인을 받아서). 예를 들어 파이프라인이 최종 프로덕션 단계로 전 환되는 시간을 제어하려는 경우 승인 작업을 사용할 수 있습니다.

Note

승인 작업이 있는 단계는 승인 작업이 승인 또는 거부되거나 제한 시간이 초과될 때까지 잠 깁니다. 제한 시간이 초과된 승인 작업은 실패한 작업과 동일한 방식으로 처리됩니다.

- 단계의 작업이 성공적으로 완료되지 않을 때 실패가 발생합니다. 개정은 해당 단계의 다음 작업 또는 파이프라인의 다음 단계로 전환하지 않습니다. 다음과 같은 상황이 발생할 수 있습니다.
 - 실패한 작업이 포함된 단계를 수동으로 재시작합니다. 이렇게 하면 실행이 재개됩니다(실패한 작 업을 다시 시도하고 성공하면 단계/파이프라인에서 계속).

• 또 다른 실행은 실패한 단계에 들어가서 실패한 실행을 대체합니다. 이 때 실패한 실행을 다시 시 도할 수 없습니다.

권장 파이프라인 구조

코드 변경이 파이프라인을 통과하는 방식을 결정할 때는 단계가 잠길 때 작업이 모두 동일한 실행을 처 리하도록 단계 내에서 관련 작업을 그룹화하는 것이 가장 좋습니다. 각 애플리케이션 환경 AWS 리전 또는 가용 영역 등에 대한 단계를 생성할 수 있습니다. 단계가 너무 많은(즉, 너무 세분화된) 파이프라 인은 동시 변경을 너무 많이 허용할 수 있으며, 큰 단계(잘 세분화되지 않은)에서 작업의 수가 많은 파 이프라인의 경우 변경 사항을 릴리스하는 데 너무 오랜 시간이 걸릴 수 있습니다.

예를 들어, 동일한 단계의 배포 작업 이후의 테스트 작업은 배포된 것과 동일한 변경 사항을 테스트하 도록 보장됩니다. 이 예제에서는 변경 사항을 테스트 환경에 배포하고 테스트한 다음, 테스트 환경의 최신 변경 사항을 프로덕션 환경에 배포합니다. 권장되는 예제에서 테스트 환경과 프로덕션 환경은 별 도의 단계입니다.



왼쪽: 관련 테스트, 배포 및 승인 작업이 그룹화되어 있습니다(권 장). 오른쪽: 관련 작업들이 별도의 단계에 있습니다(권장하지 않음).

인바운드 실행 작동 방식

인바운드 실행이란 사용할 수 없는 단계, 전환 또는 작업을 사용할 수 있게 될 때까지 기다렸다가 다음 단계로 넘어가는 실행입니다. 다음과 같은 이유로 다음 단계, 전환 또는 작업을 사용하지 못할 수 있습 니다.

- 다른 실행이 이미 다음 단계에 들어가서 잠겼습니다.
- 다음 단계로 들어가기 위한 전환이 비활성화되었습니다.

현재 실행이 후속 단계에서 완료될 시간이 있는지 여부를 제어하거나 특정 시점에서 모든 작업을 중지 하려는 경우 인바운드 실행을 보류하는 전환을 비활성화할 수 있습니다. 인바운드 실행이 있는지 확인 하려면 콘솔에서 파이프라인을 보거나 get-pipeline-state 명령의 출력을 볼 수 있습니다.

인바운드 실행은 다음과 같은 고려 사항에 따라 작동합니다.

- 작업, 전환 또는 잠금 단계를 사용할 수 있게 되면 진행 중인 인바운드 실행이 단계에 진입하여 파이 프라인을 통해 계속됩니다.
- 인바운드 실행이 대기 중인 동안에는 수동으로 중지할 수 있습니다. 인바운드 실행은 InProgress, Stopped 또는 Failed 상태일 수 있습니다.
- 인바운드 실행이 중지되거나 실패한 경우 재시도할 실패한 작업이 없으므로 다시 시도할 수 없습니
 다. 인바운드 실행이 중지되고 전환이 활성화된 경우 중지된 인바운드 실행은 단계로 계속되지 않습니다.

인바운드 실행을 보거나 중지할 수 있습니다.

입력 및 출력 아티팩트

CodePipeline은 개발 도구에 통합되어 코드 변경을 확인한 후 지속적 제공 프로세스의 모든 단계를 통 해 빌드 및 배포합니다. 아티팩트는 애플리케이션 코드가 있는 파일 또는 폴더, 인덱스 페이지 파일, 스 크립트 등과 같이 파이프라인의 작업에 의해 작동되는 파일입니다. 예를 들어 Amazon S3 소스 작업 아티팩트는 파이프라인 소스 작업에 대해 애플리케이션 소스 코드 파일이 제공되는 파일 이름(또는 파 일 경로)입니다. 파일은 ZIP 파일로 제공됩니다SampleApp_Windows.zip. 예를 들어 아티팩트 이 름:. 소스 작업의 출력 아티팩트인 애플리케이션 소스 코드 파일은 해당 작업의 출력 아티팩트이고 빌 드 작업과 같은 다음 작업에 대한 입력 아티팩트입니다. 또 다른 예로, 빌드 작업은 입력 아티팩트, 즉 애플리케이션 소스 코드 파일에 대한 애플리케이션 소스 코드를 컴파일하는 빌드 명령을 실행할 수 있 습니다. CodeBuild 작업의 <u>AWS CodeBuild 빌드 및 테스트 작업 참조</u>과 같은 아티팩트 파라미터에 대 한 자세한 내용은 특정 작업의 작업 구성 도움말 페이지를 참조하세요.

작업은 파이프라인을 생성할 때 선택한 Amazon S3 아티팩트 버킷에 저장된 입력 및 출력 아티팩트를 사용합니다. CodePipeline은 단계의 작업 유형에 적합한 입력 또는 출력 아티팩트 파일을 압축하여 전 송합니다.

Note

아티팩트 버킷은 선택한 소스 작업이 S3인 파이프라인의 소스 파일 위치로 사용되는 버킷과 다릅니다.

예시:

- 1. CodePipeline은 소스 리포지토리에 대한 커밋이 있을 때 파이프라인이 실행되도록 트리거하여 소스 단계의 출력 아티팩트(빌드되는 모든 파일)를 제공합니다.
- 이전 단계의 출력 아티팩트(빌드되는 모든 파일)는 빌드 단계에 대한 입력 아티팩트로서 수집됩니다. 빌드 단계의 출력 아티팩트(빌드된 애플리케이션)는 컨테이너에 빌드되는 업데이트 도커 이미지 또는 업데이트 애플리케이션일 수 있습니다.
- 이전 단계의 출력 아티팩트(빌드된 애플리케이션)는 배포 단계에 대한 입력 아티팩트로서 수집됩니 다(예: AWS 클라우드의 스테이징 또는 프로덕션 환경). 배포 플릿에 애플리케이션을 배포하거나, ECS 클러스터에서 실행되는 작업에 컨테이너 기반 애플리케이션을 배포할 수 있습니다.

작업을 생성하거나 편집할 때 작업의 입력 및 출력 아티팩트(들)를 지정합니다. 예를 들어, 소스 및 배 포 단계가 있는 2단계 파이프라인의 경우에는 작업 편집에서 배포 작업의 입력 아티팩트를 위한 소스 작업의 아티팩트 이름을 선택합니다.

 콘솔을 사용하여 첫 번째 파이프라인을 생성하면 CodePipeline은 동일한에 Amazon S3 버킷을 생성 하고 모든 파이프라인 AWS 계정 AWS 리전 에 대한 항목을 저장합니다. 콘솔을 사용하여 해당 리전 에 파이프라인을 생성할 때마다 CodePipeline은 버킷에 해당 파이프라인에 대한 폴더를 생성합니다. 그리고 자동화된 릴리스 프로세스가 실행되면 그 폴더를 이용해 파이프라인에 대한 아티팩트를 저 장합니다. 이 버킷의 이름은 codepipeline-*region-12345EXAMPLE*이며, 여기서 *region*은 파이프 라인을 생성한 AWS 리전이고 *12345EXAMPLE*은 버킷 이름이 고유하도록 보장하는 12자리 난수입 니다.

Note

파이프라인을 생성하는 리전에 codepipeline-region-으로 시작하는 버킷이 이미 있는 경우 CodePipeline은 해당 버킷을 기본 버킷으로 사용합니다. 또한 사전 순서를 따릅니다. 예를 들어 codepipeline-region-abcexample이 codepipeline-region-defexample보다 먼저 선택됩 니다.

CodePipeline은 아티팩트 이름을 자르기 때문에 일부 버킷 이름이 유사하게 나타날 수 있습니다. 아 티팩트 이름이 잘린 것처럼 보이더라도 CodePipeline은 이름이 잘린 아티팩트의 영향을 받지 않는 방식으로 아티팩트 버킷에 매핑합니다. 파이프라인은 정상적으로 작동할 수 있습니다. 이는 폴더 또 는 결과물에 문제가 되지 않습니다. 파이프라인 이름은 100자 이내로 제한됩니다. 결과물 폴더 이름 이 짧아 보이더라도 여전히 파이프라인에 고유합니다.

파이프라인을 생성하거나 편집할 때는 파이프라인에 아티팩트 버킷 AWS 계정 이 있어야 하며 AWS 리전, 작업을 실행하려는 리전당 하나의 아티팩트 버킷이 있어야 합니다. 콘솔을 사용하여 파이프라 인 또는 교차 리전 작업을 생성하는 경우 기본 아티팩트 버킷은 작업이 있는 리전의 CodePipeline에 의해 구성됩니다.

AWS CLI 를 사용하여 파이프라인을 생성하는 경우 해당 버킷이 파이프라인 AWS 계정 과 동일한 및 에 있는 한 해당 파이프라인의 아티팩트를 Amazon S3 버킷에 저장할 수 AWS 리전 있습니다. 계정 에 허용된 Amazon S3 버킷의 한도를 넘을까 걱정이 된다면 이렇게 할 수 있습니다. AWS CLI 를 사 용하여 파이프라인을 생성하거나 편집하고 교차 리전 작업(파이프라인과 다른 리전의 AWS 공급자 가 있는 작업)을 추가하는 경우 작업을 실행하려는 각 추가 리전에 대해 아티팩트 버킷을 제공해야 합니다.

- 모든 작업에는 유형이 있습니다. 유형에 따라 다음 중 하나 또는 둘 다 갖고 있을 수 있습니다.
 - 작업을 실행하는 동안 소비하거나 작업하는 아티팩트인 입력 아티팩트.
 - 작업의 출력인 출력 아티팩트.

파이프라인의 모든 출력 아티팩트에는 고유의 이름이 있어야 합니다. 하나의 작업에 대한 모든 입력 아티팩트는 앞선 파이프라인 내 작업의 출력 아티팩트와 일치해야 합니다. 한 단계에서 작업 직전에 있었던 작업이든 몇 단계 전의 단계에서 실행된 작업이든 중요하지 않습니다.

아티팩트는 한 가지 이상의 작업에 의해 작동합니다.

스테이지 조건은 어떻게 작동하나요?

규칙을 지정하는 각 조건에 대해 규칙이 실행됩니다. 조건이 실패할 경우 결과가 적용됩니다. 스테이지 는 조건이 실패할 때만 지정된 결과를 수행합니다. 선택적으로 규칙의 일부로서 CodePipeline이 특정 경우에 사용해야 하는 리소스도 지정합니다. 예를 들어 CloudWatchAlarm 규칙은 CloudWatch 경보 리소스를 사용하여 조건에 대한 검사를 실행합니다.

조건은 여러 규칙과 일치할 수 있으며 각 규칙은 세 공급자 중 하나를 지정할 수 있습니다.

다음과 같이 조건을 생성하기 위한 상위 수준 흐름입니다.

- CodePipeline의 사용 가능한 조건 유형에서 조건 유형을 선택합니다. 예를 들어, 스테이지 성공 후 계속 진행하기 전에 일련의 규칙을 사용하여 검사를 실행할 수 있도록 성공 시 조건 유형을 사용하 여 스테이지를 설정합니다.
- [] 규칙을 선택합니다. 예를 들어 CloudWatchAlarm 규칙은 경보를 확인하고 EB를 사용하여 사전 구성된 경보 임계값을 확인합니다. 확인에 성공하고 경보가 임계값 미만인 경우 스테이지를 진행할 수 있습니다.
- 3. 규칙이 실패할 경우 사용할 롤백과 같은 결과를 구성합니다.

조건은 특정 유형의 표현식에 사용되며 각 표현식에는 다음과 같이 사용할 수 있는 결과에 대한 특정 옵션이 있습니다.

- 입력 충족되는 경우 스테이지에 대한 진입을 허용하는 검사 수행 조건입니다. 규칙은 실패 또는 건 너뛰기 결과 옵션과 함께 사용됩니다.
- 실패 시 스테이지가 실패할 때 스테이지를 확인하기 위한 조건입니다. 규칙은 롤백 결과 옵션과 함 께 사용됩니다.
- 성공 시 스테이지가 성공할 때 스테이지를 확인하기 위한 조건입니다. 규칙은 롤백 또는 실패와 같
 은 결과 옵션과 함께 사용됩니다.

다음 다이어그램은 CodePipeline의 입력 조건 유형에 대한 예제 흐름을 보여줍니다. 조건이 충족되지 않으면 어떻게 되나요? 규칙이 실패하나요? 라는 질문에 조건이 답합니다. 다음 흐름에서 입력 조건은 LambdaInvoke 규칙과 CloudWatchAlarm 규칙으로 구성됩니다. 규칙이 실패하면 실패와 같은 구성 된 결과가 적용됩니다.

Stage	configured for entry condition	
Condition: Entry Results: Block or Enter	Override	
	 Rule2 status: Failed (Error), Result: Block Rule2 status: Succeeded, Result: Enter Rule1 status: Failed (Alarm), Result: Block Rule2 status: Succeeded, Result: Enter Qverride results in Stage status: Enter 	

다음 다이어그램은 CodePipeline의 실패 시 조건 유형에 대한 예제 흐름을 보여줍니다. 조건이 충족되 면 어떻게 되나요? 규칙이 모두 검사에 성공하나요? 라는 질문에 조건이 답합니다. 다음 흐름에서 실패 시 조건은 LambdaInvoke 규칙과 CloudWatchAlarm 규칙으로 구성됩니다. 규칙이 성공하면 실패와 같은 구성된 결과가 적용됩니다.

Condition: OnFailure Result: Rollback Rule2 { } CWE Lambda	reafor on Fallure condition	
	e1 { } O	
 Rule2 status: Failed (Error), Result: Rollback Rule2 status: Succeeded, Result: Fail stage Rule1 status: Failed (Alarm), Result: Rollback Rule2 status: Succeeded, Result: Fail stage 	 Rule2 status: Failed (Error), Result: Rollback Rule2 status: Succeeded, Result: Fail stage Rule1 status: Failed (Alarm), Result: Rollback Rule2 status: Succeeded, Result: Fail stage 	

다음 다이어그램은 CodePipeline의 성공 시 조건 유형에 대한 예제 흐름을 보여줍니다. 조건이 충족되 면 어떻게 되나요? 규칙이 모두 검사에 성공하나요? 라는 질문에 조건이 답합니다. 다음 흐름에서 성공 시 조건은 LambdaInvoke 규칙과 CloudWatchAlarm 규칙으로 구성됩니다. 규칙이 성공하면 실패와 같은 구성된 결과가 적용됩니다.

Stage	configured for On Success condition
Condition: Entry Results: Rollback or Fail	Override Image: Comparison of the second secon
	Rule2 status: Failed (Error), Result: Rollback or Fail Rule2 status: Succeeded, Result: Proceed Rule1 status: Failed (Alarm), Result: Rollback or Fail
	 Rule2 status: Succeeded, Result: Proceed Override results in Stage status: Proceed

스테이지 조건에 대한 규칙

스테이지 조건을 구성할 때 사전 정의된 규칙에서 선택하고 규칙에 대한 결과를 지정합니다. 조건의 규 칙 중 하나라도 실패하면 조건 상태가 실패하고 모든 규칙이 성공하면 성공합니다. 실패 시 및 성공 시 조건에 대한 기준이 충족되는 방법은 규칙 유형에 따라 다릅니다.

다음은 스테이지 조건에 추가할 수 있는 관리형 규칙입니다.

- 조건은 명령 규칙을 사용하여 조건에 대한 규칙 기준을 충족하는 명령을 지정할 수 있습니다. 이 규 칙에 대한 자세한 내용은 명령 섹션을 참조하세요.
- 조건은 AWS DeploymentWindow 규칙을 사용하여 배포를 허용하는 승인된 배포 시간을 지정할 수 있습니다. 규칙의 기준은 배포 창에 제공된 cron 표현식으로 측정됩니다. 배포 창의 날짜 및 시간이 규칙의 cron 표현식 기준을 충족하면 규칙이 성공합니다. 이 규칙에 대한 자세한 내용은 DeploymentWindow 섹션을 참조하세요.
- 조건은 AWS Lambda 규칙을 사용하여 구성된 Lambda 함수에서 반환된 오류 상태를 확인할 수 있 습니다. 확인이 Lambda 함수 결과를 수신하면 규칙이 충족됩니다. Lambda 함수의 오류는 실패 시 조건의 기준을 충족합니다. 이 규칙에 대한 자세한 내용은 LambdaInvoke 섹션을 참조하세요.

- 조건은 AWS CloudWatchAlarm 규칙을 사용하여 CloudWatch 이벤트에서 구성된 경보를 확인할 수 있습니다. 확인이 OK, ALARM 또는 INSUFF_DATA의 경보 상태를 반환하면 규칙이 충족됩니다. 성 공 시 조건의 경우 OK 및 INSUFFICIENT_DATA가 기준을 충족합니다. ALARM은 실패 시 조건에 대 한 기준을 충족합니다. 이 규칙에 대한 자세한 내용은 CloudWatchAlarm 섹션을 참조하세요.
- 조건은 VariableCheck 규칙을 사용하여 출력 변수가 제공된 표현식에 대해 확인되는 조건을 생성할 수 있습니다. 규칙은 변수 값이 지정된 출력 변수와 같거나 큰 값과 같은 규칙 기준을 충족할 때 확인 을 전달합니다. 이 규칙에 대한 자세한 내용은 <u>VariableCheck</u> 섹션을 참조하세요.

파이프라인 유형

CodePipeline은 다음의 파이프라인 유형을 제공하고 이러한 볼륨 유형은 특성과 가격이 다르므로 애플 리케이션의 필요에 맞게 파이프라인 특성과 비용을 조정할 수 있습니다.

- V1 유형 파이프라인은 표준 파이프라인, 단계, 작업 수준 파라미터를 포함하는 JSON 구조를 가지고 있습니다.
- V2 유형 파이프라인은 릴리스 안전 및 트리거 구성을 위한 추가 파라미터와 함께 V1 유형과 구조가 동일합니다.

CodePipeline에 대한 요금 정보는 요금을 참조하세요.

각 파이프라인 유형의 파라미터에 대한 자세한 내용은 <u>CodePipeline 파이프라인 구조 참조</u> 페이지를 참조하세요. 선택할 파이프라인 유형에 대한 자세한 내용은 <u>나에게 적합한 파이프라인 유형은 무엇인</u> 가요?을 참조하세요.

나에게 적합한 파이프라인 유형은 무엇인가요?

파이프라인 유형은 각 파이프라인 버전에서 지원하는 특성 및 기능 집합에 따라 결정됩니다.

다음은 각 파이프라인 유형에 사용할 수 있는 사용 사례 및 특성에 대한 요약입니다.

	V1 유형	V2 유형
특성		
사용 사례	• 표준 배포	• 런타임 시 파이프라인 수준 변수를 전달하여 구성한 배 포

	V1 유형	V2 유형
특성		
		• Git 태그에서 시작하도록 파 이프라인이 구성된 배포
<u>작업 수준 변수</u>	지원	지원
병렬 실행 모드	지원되지 않음	지원
<u>파이프라인 수준 변수</u>	지원되지 않음	지원
대기열 실행 모드	지원되지 않음	지원
<u>파이프라인 단계에 대한 롤백</u>	지원되지 않음	지원
<u>소스 개정 재정의</u>	지원되지 않음	지원
<u>스테이지 조건</u>	지원되지 않음	지원
<u>Git 태그, 풀 요청, 브랜치 또는</u> 파일 경로 트리거 및 필터링	지원되지 않음	지원
<u>명령 작업</u>	지원되지 않음	지원
<u>건너뛰기 결과를 사용하여 입</u> 력 조건 생성	지원되지 않음	지원
실패 시 자동 재시도를 위한 스 테이지 구성	지원되지 않음	지원

CodePipeline에 대한 요금 정보는 <u>요금</u>을 참조하세요.

Python 스크립트를 생성하고 실행하여 V1 유형 파이프라인을 V2 유형 파이프라인으로 이동하는 데 드 는 잠재적 비용을 분석할 수 있습니다.

Note

아래 샘플 스크립트는 데모 및 평가 목적으로만 사용됩니다. 이는 견적 도구가 아니며 V2 유형 파이프라인의 실제 사용에 대한 비용을 보장하지 않으며 적용될 수 있는 세금은 포함되지 않습 니다. CodePipeline에 대한 요금 정보는 요금을 참조하세요.

V1 유형 파이프라인을 V2 유형 파이프라인으로 이동하는 비용을 평가할 수 있는 스크립트를 생성하고 실행하려면

- 1. Python을 다운로드하여 설치하세요.
- 2. 터미널 창을 엽니다. 다음 명령을 실행하여 PipelineCostAnalyzer.py라는 새 python 스크립트를 생성합니다.

vi PipelineCostAnalyzer.py

3. 다음 코드를 복사하여 PipelineCostAnalyzer.py 스크립트에 붙여 넣습니다.

```
import boto3
import sys
import math
from datetime import datetime, timedelta, timezone
if len(sys.argv) < 3:
    raise Exception("Please provide region name and pipeline name as arguments.
Example usage: python PipelineCostAnalyzer.py us-east-1 MyPipeline")
session = boto3.Session(profile_name='default', region_name=sys.argv[1])
pipeline = sys.argv[2]
codepipeline = session.client('codepipeline')
def analyze_cost_in_v2(pipeline_name):
    if codepipeline.get_pipeline(name=pipeline)['pipeline']['pipelineType'] ==
 'V2':
        raise Exception("Provided pipeline is already of type V2.")
    total_action_executions = 0
   total_blling_action_executions = 0
    total_action_execution_minutes = 0
    cost = 0.0
   hasNextToken = True
    nextToken = ""
```

```
while hasNextToken:
       if nextToken=="":
            response =
 codepipeline.list_action_executions(pipelineName=pipeline_name)
       else:
            response =
 codepipeline.list_action_executions(pipelineName=pipeline_name,
 nextToken=nextToken)
       if 'nextToken' in response:
            nextToken = response['nextToken']
        else:
            hasNextToken= False
       for action_execution in response['actionExecutionDetails']:
            start_time = action_execution['startTime']
            end_time = action_execution['lastUpdateTime']
            if (start_time < (datetime.now(timezone.utc) - timedelta(days=30))):
                hasNextToken= False
                continue
            total_action_executions += 1
            if (action_execution['status'] in ['Succeeded', 'Failed', 'Stopped']):
                action_owner = action_execution['input']['actionTypeId']['owner']
                action_category = action_execution['input']['actionTypeId']
['category']
                if (action_owner == 'Custom' or (action_owner == 'AWS' and
action_category == 'Approval')):
                    continue
                total_blling_action_executions += 1
                action_execution_minutes = (end_time -
 start_time).total_seconds()/60
                action_execution_cost = math.ceil(action_execution_minutes) * 0.002
                total_action_execution_minutes += action_execution_minutes
                cost = round(cost + action_execution_cost, 2)
   print ("{:<40}".format('Activity in last 30 days:'))</pre>
    print ("| {:<40} | {:<10}".format('______
          '))
    print ("| {:<40} | {:<10}".format('Total action executions:',</pre>
total_action_executions))
    print ("| {:<40} | {:<10}".format('Total billing action executions:',</pre>
total_blling_action_executions))
    print ("| {:<40} | {:<10}".format('Total billing action execution minutes:',</pre>
 round(total_action_execution_minutes, 2)))
```

```
print ("| {:<40} | {:<10}".format('Cost of moving to V2 in $:', cost - 1))</pre>
```

analyze_cost_in_v2(pipeline)

4. 터미널 또는 명령 프롬프트에서 분석기 스크립트를 생성한 디렉터리를 변경합니다.

해당 디렉터리에서 다음 명령을 실행합니다. 여기서 region은 분석하려는 V1 파이프라인을 생성 한 AWS 리전 입니다. 이름을 제공하여 특정 파이프라인을 선택적으로 평가할 수도 있습니다.

python3 PipelineCostAnalyzer.py region --pipelineName

예를 들어 다음 명령을 실행하여 PipelineCostAnalyzer.py라는 Python 스크립트를 실행합니다. 이 예제에서 리전은 us-west-2입니다.

python3 PipelineCostAnalyzer.py us-west-2

Note

이 스크립트는 특정 파이프라인 이름을 지정하지 않는 한 지정된 AWS 리전 의 모든 V1 파 이프라인을 분석합니다.

5. 스크립트의 다음 샘플 출력에서 작업 실행 목록, 청구에 적합한 작업 실행 목록, 이러한 작업 실행 의 총 런타임, V2 파이프라인에서 수행된 이러한 작업의 예상 비용을 볼 수 있습니다.

이 예제에서 마지막 행의 음수 값은 V2 유형 파이프라인으로 이동하여 저장할 수 있는 예상 금액 을 나타냅니다.

Note

비용 및 기타 정보를 보여주는 스크립트 출력 및 관련 예제는 추정치일 뿐입니다. 데모 및 평가 목적으로만 사용되며 실제 절감액을 보장하지 않습니다. CodePipeline에 대한 요금 정보는 요금을 참조하세요.

CodePipeline 시작하기

CodePipeline을 처음 사용하는 경우 이 장의 단계 뒤에 나오는 이 설명서의 자습서를 따라 설정하세요.

CodePipeline 콘솔은 촉소 가능한 패널에 유용한 정보를 담고 있습니다. 이 패널은 해당 페이지의 정보 아이콘이나 정보 링크를 통해 열 수 있습니다.

0

).

언제든지 이 패널을 닫을 수 있습니다.

CodePipeline 콘솔에서 리포지토리, 구축 프로젝트, 배포 애플리케이션 및 파이프라인과 같은 리소스 를 신속하게 검색할 수도 있습니다. 리소스로 이동을 선택하거나 / 키를 누른 후 리소스 이름을 입력하 세요. 목록에 일치 항목이 나타납니다. 검색은 대/소문자를 구분하지 않습니다. 보기 권한이 있는 리소 스만 표시됩니다. 자세한 내용은 콘솔에서 리소스 보기 단원을 참조하십시오.

AWS CodePipeline 를 처음 사용하려면 먼저 AWS 계정 를 생성하고 첫 번째 관리 사용자를 생성해야 합니다.

주제

- 1단계: AWS 계정 및 관리 사용자 생성
- 2단계: CodePipeline에 대한 관리자 액세스를 위한 관리형 정책 적용
- <u>3단계: 설치 AWS CLI</u>
- <u>4단계: CodePipeline 콘솔 열기</u>
- 다음 단계

1단계: AWS 계정 및 관리 사용자 생성

가입 AWS 계정

이 없는 경우 다음 단계를 AWS 계정완료하여 생성합니다.

에 가입하려면 AWS 계정

- 1. https://portal.aws.amazon.com/billing/signup을 엽니다.
- 2. 온라인 지시 사항을 따릅니다.

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

에 가입하면 AWS 계정AWS 계정 루트 사용자이 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스에 액세스할 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 <u>루트 사용자 액세스 권한이 필요한 작업</u>을 수행하는 것 입니다.

AWS 는 가입 프로세스가 완료된 후 확인 이메일을 보냅니다. 언제든지 <u>https://aws.amazon.com/</u>으로 이동하고 내 계정을 선택하여 현재 계정 활동을 보고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

에 가입한 후 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 AWS 계정보호 AWS IAM Identity Center, AWS 계정 루트 사용자활성화 및 생성합니다.

보안 AWS 계정 루트 사용자

1. 루트 사용자를 선택하고 AWS 계정 이메일 주소를 입력하여 계정 소유자<u>AWS Management</u> Console로에 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

루트 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 User Guide의 <u>루트 사용자</u> 로 로그인을 참조하세요.

2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

지침은 IAM 사용 설명서의 <u>AWS 계정 루트 사용자(콘솔)에 대한 가상 MFA 디바이스 활성화를 참</u> 조하세요.

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

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

2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

를 ID 소스 IAM Identity Center 디렉터리 로 사용하는 방법에 대한 자습서는 사용 AWS IAM Identity Center 설명서<u>의 기본값으로 사용자 액세스 구성을 IAM Identity Center 디렉터리</u> 참조하 세요. 관리 액세스 권한이 있는 사용자로 로그인

• IAM IDentity Center 사용자로 로그인하려면 IAM Identity Center 사용자를 생성할 때 이메일 주소 로 전송된 로그인 URL을 사용합니다.

IAM Identity Center 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 사용 설명 서의 AWS 액세스 포털에 로그인을 참조하세요.

추가 사용자에게 액세스 권한 할당

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

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

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

지침은 AWS IAM Identity Center 사용 설명서의 Add groups를 참조하세요.

2단계: CodePipeline에 대한 관리자 액세스를 위한 관리형 정책 적용

CodePipeline과 상호 작용할 수 있는 권한을 부여해야 합니다. 가장 빠른 방법은 AWSCodePipeline_FullAccess 관리형 정책을 관리 사용자에게 적용하는 것입니다.

Note

AWSCodePipeline_FullAccess 정책에는 콘솔 사용자가 IAM 역할을 CodePipeline 또는 다른 AWS 서비스로 전달할 수 있도록 하는 권한이 포함됩니다. 그러면 서비스가 역할을 수 임하고 사용자 대신 작업을 수행할 수 있습니다. 사용자, 역할 또는 그룹에 정책을 연결하면 iam:PassRole 권한이 적용됩니다. 정책이 신뢰할 수 있는 사용자에게만 적용되는지 확인하 십시오. 이러한 권한을 가진 사용자가 콘솔을 사용하여 파이프라인을 생성하거나 편집할 때 다 음 옵션을 사용할 수 있습니다.

- CodePipeline 서비스 역할을 생성하거나 기존 역할을 선택하고 해당 역할을 CodePipeline에 전달합니다.
- 변경 감지에 대한 CloudWatch Events 규칙을 만들고 CloudWatch Events 서비스 역할을 CloudWatch Events로 전달하도록 선택할 수 있습니다.

자세한 내용은 에 역할을 전달할 수 있는 사용자 권한 부여를 참조하세요 AWS 서비스.

Note

AWSCodePipeline_FullAccess 정책에서 IAM 사용자에게 액세스 권한이 있는 모든 CodePipeline 작업 및 리소스에 대한 액세스 권한뿐만 아니라 CodeDeploy, Elastic Beanstalk 또는 Amazon S3를 포함하는 단계를 생성하는 것과 같이 파이프라인에 단계를 생성할 때 사 용할 수 있는 모든 작업에 대한 액세스 권한을 제공합니다. 모범 사례대로 하려면, 개별 사용 자에게 각자의 업무를 수행하는 데 필요한 권한만 부여해야 합니다. IAM 사용자가 한정된 범 위의 CodePipeline 작업 및 리소스만 사용할 수 있도록 제한하는 방법에 대한 자세한 내용은 CodePipeline 서비스 역할에서 권한 제거 단원을 참조하세요.

액세스 권한을 제공하려면 사용자, 그룹 또는 역할에 권한을 추가하세요:

• 의 사용자 및 그룹 AWS IAM Identity Center:

권한 세트를 생성합니다. AWS IAM Identity Center 사용 설명서의 <u>권한 세트 생성</u>의 지침을 따릅니다.

• 보안 인증 공급자를 통해 IAM에서 관리되는 사용자:

ID 페더레이션을 위한 역할을 생성합니다. IAM 사용 설명서의 <u>Create a role for a third-party identity</u> provider (federation)의 지침을 따릅니다.

- IAM 사용자:
 - 사용자가 맡을 수 있는 역할을 생성합니다. IAM 사용 설명서에서 <u>Create a role for an IAM user</u>의 지침을 따릅니다.
 - (권장되지 않음)정책을 사용자에게 직접 연결하거나 사용자를 사용자 그룹에 추가합니다. IAM 사용 설명서에서 사용자(콘솔)에 권한 추가의 지침을 따르세요.

3단계: 설치 AWS CLI

로컬 개발 시스템의 AWS CLI에서 CodePipeline 명령을 호출하려면 AWS CLI를 설치해야 합니다. 이 단계는 CodePipeline 콘솔에 이 안내서의 단계만 사용하여 시작하려는 경우 선택 사항입니다.

를 설치하고 구성하려면 AWS CLI

1. 로컬 시스템에서를 다운로드하여 설치합니다 AWS CLI. 이렇게 하면 명령줄에서 CodePipeline과 상호 작용할 수 있습니다. 자세한 내용은 AWS 명령줄 인터페이스 설정 단원을 참조하십시오.

1 Note

CodePipeline은 AWS CLI 버전 1.7.38 이상에서만 작동합니다. 설치했을 수 AWS CLI 있 는의 버전을 확인하려면 명령을 실행합니다aws --version. 이전 버전의 AWS CLI 를 최 신 버전으로 업그레이드하려면 <u>제거 AWS CLI</u>의 지침을 따른 다음 <u>설치 AWS Command</u> Line Interface의 지침을 따릅니다.

2. 다음과 같이 configure 명령을 AWS CLI 사용하여를 구성합니다.

aws configure

메시지가 표시되면 CodePipeline에서 사용할 IAM 사용자의 AWS 액세스 키와 AWS 보안 액세스 키를 지정합니다. 기본 리전 이름을 입력하라는 메시지가 표시되면, 파이프라인을 생성할 리전을 지정합니다(예: us-east-2). 기본 출력 형식을 묻는 메시지가 표시되면 json을 지정합니다. 예 시:

AWS Access Key ID [None]: Type your target AWS access key ID here, and then press Enter AWS Secret Access Key [None]: Type your target AWS secret access key here, and then press Enter Default region name [None]: Type us-east-2 here, and then press Enter Default output format [None]: Type json here, and then press Enter

Note

IAM, 액세스 키, 보안 키에 등에 대한 자세한 내용은 <u>IAM 사용자의 액세스 키 관리</u> 및 <u>보안</u> <u>인증 정보를 얻는 방법</u>을 참조하세요. CodePipeline에 사용할 수 있는 리전과 엔드포인트에 대한 자세한 내용은 <u>AWS</u> <u>CodePipeline 엔드포인트 및 할당량</u>을 참조하세요.

4단계: CodePipeline 콘솔 열기

• 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> <u>codepipeline/home</u>://https//https//htt

다음 단계

사전 필수 단계를 완료했습니다. CodePipeline 사용을 시작할 수 있습니다. CodePipeline 작업을 시작 하려면 <u>CodePipeline 자습서</u> 단원을 참조하세요.

CodePipeline과 제품 및 서비스 통합

기본적으로 AWS CodePipeline 는 여러 AWS 서비스 및 파트너 제품 및 서비스와 통합됩니다. 다음 섹 션의 정보를 이용해 제품과 서비스를 통합할 CodePipeline을 구성할 수 있습니다.

다음의 관련 리소스는 이 서비스 사용 시 도움이 될 수 있습니다.

주제

- CodePipeline 작업 유형과의 통합
- CodePipeline으로 일반 통합
- 커뮤니티 예제

CodePipeline 작업 유형과의 통합

이 주제에 나오는 통합 정보는 CodePipeline 작업 유형으로 정리한 것입니다.

주제

- <u>소스 작업 통합</u>
- 빌드 작업 통합
- 테스트 작업 통합
- 배포 작업 통합
- Amazon Simple Notification Service와 승인 작업 통합
- 호출 작업 통합

소스 작업 통합

다음 정보는 CodePipeline 작업 유형별로 구성되어 있으며 다음 소스 작업 공급자와 통합되도록 CodePipeline을 구성하는 데 도움이 됩니다.

주제

- Amazon ECR 소스 작업
- <u>Amazon S3 소스 작업</u>
- <u>Bitbucket Cloud, GitHub(GitHub App을 통해), GitHub Enterprise Server, GitLab.com, 및 GitLab 자</u> 체 관리형에 대한 연결

- CodeCommit 소스 작업
- GitHub(OAuth 앱 사용) 소스 작업

Amazon ECR 소스 작업

Amazon ECR은 AWS Docker 이미지 리포지토리 서비스입니다. Docker의 push 명령과 pull 명령을 사용하여 Docker 이미지를 리포지토리에 업로드할 수 있습니다. Amazon ECR 리포지토리 URI와 이미지는 소스 이미지 정보를 참조하기 위해 Amazon ECS 작업 정의에 사용됩니다.

자세히 알아보기:

- 구성 파라미터와 예제 JSON/YAML 코드 조각은 Amazon ECR 소스 작업 참조을 참조하세요.
- 파이프라인 스테이지 및 작업 생성
- 자습서: Amazon ECR 소스 및 ECS-to-CodeDeploy 배포를 사용하여 파이프라인 생성

Amazon S3 소스 작업

<u>Amazon S3</u>는 인터넷 스토리지입니다. Amazon S3를 사용하면 인터넷을 통해 언제 어디서든 원하는 양의 데이터를 저장하고 검색할 수 있습니다. 한 가지 버전의 Amazon S3 버킷을 코드에 대한 소스 작 업으로 사용하도록 CodePipeline을 구성할 수 있습니다.

Note

Amazon S3는 파이프라인에 배포 작업으로 포함될 수 있습니다.

자세히 알아보기:

- 구성 파라미터와 예제 JSON/YAML 코드 조각은 Amazon S3 소스 작업 참조을 참조하세요.
- 1단계: 애플리케이션에 대한 S3 소스 버킷 생성
- 파이프라인 생성(CLI)
- CodePipeline은 Amazon EventBridge(이전의 Amazon CloudWatch Events)를 사용하여 Amazon S3 소스 버킷의 변경 사항을 감지합니다. <u>CodePipeline으로 일반 통합</u>을(를) 참조하세요.

Bitbucket Cloud, GitHub(GitHub App을 통해), GitHub Enterprise Server, GitLab.com, 및 GitLab 자체 관리형에 대한 연결

연결(CodeStarSourceConnection 작업)은 타사 Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com, 또는 GitLab 자체 관리형 리포지토리에 액세스하는 데 사용됩니다.

Note

아시아 태평양(홍콩), 아시아 태평양(하이데라바드), 아시아 태평양(자카르타), 아시아 태 평양(멜버른), 아시아 태평양(오사카), 아프리카(케이프타운), 중동(바레인), 중동(UAE), 유 럽(스페인), 유럽(취리히), 이스라엘(텔아비브) 또는 AWS GovCloud(미국 서부) 리전에서 는이 기능을 사용할 수 없습니다. 사용 가능한 다른 작업을 참조하려면 <u>CodePipeline과 제</u> 품 및 서비스 통합을 참조하세요. 유럽(밀라노) 리전에서 이 조치를 고려할 경우 <u>Bitbucket</u> <u>Cloud, GitHub, GitHub Enterprise Server, GitLab.com 및 GitLab 자체 관리형 작업용</u> CodeStarSourceConnection의 참고 사항을 참조하세요.

Bitbucket Cloud 리포지토리를 코드에 대한 소스로 사용하도록 CodePipeline을 구성할 수 있습니다. 전에 만들어 둔 Bitbucket 계정과 최소 한 개의 Bitbucket Cloud 리포지토리가 있어야 합니다. 파이프라인을 만들거나 기존 파이프라인 을 편집하여 Bitbucket Cloud 리포지토리에 소스 작업을 추가할 수 있습니다.

Note

Bitbucket Cloud 리포지토리에 대한 연결을 생성할 수 있습니다. Bitbucket Server와 같은 설치된 Bitbucket 공급자 유형은 지원되지 않 습니다.

파이프라인이 타사 코드 리포지토리에 액세스할 수 있도록 연결이라는 리소스 를 설정할 수 있습니다. 연결을 생성하는 경우 타사 코드 리포지토리와 함께 커 넥터 앱을 설치하고 생성한 연결과 이를 연결합니다.

Bitbucket Cloud의 경우 콘솔의 Bitbucket 옵션 또는 CLI의 CodestarS ourceConnection 작업을 사용하세요. <u>Bitbucket Cloud 연결</u>을(를) 참조하 세요. 이 작업에 대한 전체 복제 옵션을 사용하여 리포지토리 Git 메타데이터를 참조 하여 다운스트림 작업에서 Git 명령을 직접 수행할 수 있도록 할 수 있습니다. 이 옵션은 CodeBuild 다운스트림 작업에서만 사용할 수 있습니다.

자세히 알아보기:

- 구성 파라미터와 예제 JSON/YAML 코드 조각은 <u>Bitbucket Cloud, GitHub,</u> <u>GitHub Enterprise Server, GitLab.com 및 GitLab 자체 관리형 작업용</u> CodeStarSourceConnection을 참조하세요.
- Bitbucket Cloud 소스로 파이프라인을 생성하는 시작하기 자습서를 보려면 연결 시작하기를 참조하세요.

GitHub 또는 GitHub 리포지토리를 코드에 대한 소스로 사용하도록 CodePipeline을 구성할 GitHub Enterpris 수 있습니다. 전에 만들어 둔 GitHub 계정과 최소 한 개의 GitHub 리포지토리가 있어야 합니다. 파이프라인을 만들거나 기존 파이프라인을 편집하여 GitHub 리 포지토리에 소스 작업을 추가할 수 있습니다.

> 파이프라인이 타사 코드 리포지토리에 액세스할 수 있도록 연결이라는 리소스 를 설정할 수 있습니다. 연결을 생성하는 경우 타사 코드 리포지토리와 함께 커 넥터 앱을 설치하고 생성한 연결과 이를 연결합니다.

> 콘솔에서 GitHub(GitHub 앱 사용) 공급자 옵션을 사용하거나 CLI에서 CodestarSourceConnection 작업을 사용합니다. <u>GitHub 연결</u>을(를) 참 조하세요.

> 이 작업에 대한 전체 복제 옵션을 사용하여 리포지토리 Git 메타데이터를 참조 하여 다운스트림 작업에서 Git 명령을 직접 수행할 수 있도록 할 수 있습니다. 이 옵션은 CodeBuild 다운스트림 작업에서만 사용할 수 있습니다.

자세히 알아보기:

- 구성 파라미터와 예제 JSON/YAML 코드 조각은 <u>Bitbucket Cloud, GitHub,</u> <u>GitHub Enterprise Server, GitLab.com 및 GitLab 자체 관리형 작업용</u> CodeStarSourceConnection을 참조하세요.
- GitHub 리포지토리에 연결하고 전체 복제 옵션을 사용하는 방법을 보여주는 자습서는 <u>자습서: GitHub 파이프라인 소스와 함께 전체 복제 사용</u> 단원을 참 조하세요.

현재 GitHub(GitHub 앱을 통해) 작업은 GitHub의 버전 2 소스 작업입니다. GitHub(OAuth 앱 사용) 작업은 OAuth 토큰 인증으로 관리되는 버전 1GitHub 작업입니다. GitHub(OAuth 앱 사용) 작업을 사용하지 않는 것이 좋지만 GitHub(OAuth 앱 사용) 작업이 있는 기존 파이프라인은 아무런 영향 없이 계속 작동합니다. 이제 GitHub 앱으로 GitHub 소스 작업을 관리하는 Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com 및 GitLab 자체 관리형 작업용 CodeStarSourceConnection 소스 작업을 파이프라인에서 사용할 수 있습니다. GitHub(OAuth 앱 사용) 작업을 사용하는 파이프라인 이 있는 경우에서 GitHub(GitHub 앱 사용) 소스 작업을 GitHub(GitHub 앱 사용) 소스 작업으로 업데이트.

GitHub Enterprise GitHub Enterprise Server 리포지토리를 코드에 대한 소스로 사용하도록 e Server CodePipeline을 구성할 수 있습니다. 전에 만들어 둔 GitHub 계정과 최소 한 개 의 GitHub 리포지토리가 있어야 합니다. 파이프라인을 만들거나 기존 파이프라 인을 편집하여 GitHub Enterprise Server 리포지토리에 소스 작업을 추가할 수 있습니다.

> 파이프라인이 타사 코드 리포지토리에 액세스할 수 있도록 연결이라는 리소스 를 설정할 수 있습니다. 연결을 생성하는 경우 타사 코드 리포지토리와 함께 커 넥터 앱을 설치하고 생성한 연결과 이를 연결합니다.

콘솔의 GitHub Enterprise Server 공급자 옵션 또는 CLI의 CodestarS ourceConnection 작업을 사용하세요. <u>GitHub Enterprise Server 연결</u>을 (를) 참조하세요.

이 작업에 대한 전체 복제 옵션을 사용하여 리포지토리 Git 메타데이터를 참조 하여 다운스트림 작업에서 Git 명령을 직접 수행할 수 있도록 할 수 있습니다. 이 옵션은 CodeBuild 다운스트림 작업에서만 사용할 수 있습니다.

자세히 알아보기:

 구성 파라미터와 예제 JSON/YAML 코드 조각은 <u>Bitbucket Cloud, GitHub,</u> <u>GitHub Enterprise Server, GitLab.com 및 GitLab 자체 관리형 작업용</u> <u>CodeStarSourceConnection</u>을 참조하세요. GitHub 리포지토리에 연결하고 전체 복제 옵션을 사용하는 방법을 보여주는 자습서는 <u>자습서: GitHub 파이프라인 소스와 함께 전체 복제 사용</u> 단원을 참 조하세요.

GitLab.com 이포지토리를 코드에 대한 소스로 사용하도록 CodePipeline을 구 성할 수 있습니다. 전에 만들어 둔 GitLab.com 계정과 최소 한 개의 GitLab.com 리포지토리가 있어야 합니다. 파이프라인을 만들거나 기존 파이프라인을 편집 하여 GitLab.com 리포지토리에 소스 작업을 추가할 수 있습니다.

> 콘솔의 GitLab 공급자 옵션 또는 CLI의 GitLab 공급자와 함께 CodestarS ourceConnection 작업을 사용하세요. <u>GitLab.com 연결</u>을(를) 참조하세 요.

자세히 알아보기:

• 구성 파라미터와 예제 JSON/YAML 코드 조각은 <u>Bitbucket Cloud, GitHub,</u> <u>GitHub Enterprise Server, GitLab.com 및 GitLab 자체 관리형 작업용</u> CodeStarSourceConnection을 참조하세요.

GitLab 자체 관리 GitLab 자체 관리형 설치를 코드 소스로 사용하도록 CodePipeline을 구성할 수 있습니다. 이전에 GitLab 계정을 만들고 자체 관리형 GitLab(엔터프라이즈 에디 션 또는 커뮤니티 에디션)을 구독하고 있어야 합니다. 파이프라인을 만들거나 기존 파이프라인을 편집하여 GitLab 자체 관리형 리포지토리에 소스 작업을 추 가할 수 있습니다.

> 파이프라인이 타사 코드 리포지토리에 액세스할 수 있도록 연결이라는 리소스 를 설정할 수 있습니다. 연결을 생성하는 경우 타사 코드 리포지토리와 함께 커 넥터 앱을 설치하고 생성한 연결과 이를 연결합니다.

> 콘솔의 GitLab 자체 관리형 공급자 옵션 또는 CLI의 CodestarSourceConn ection 작업을 사용하십시오. <u>GitLab 자체 관리형을 위한 연결</u>을(를) 참조하 세요.

이 작업에 대한 전체 복제 옵션을 사용하여 리포지토리 Git 메타데이터를 참조 하여 다운스트림 작업에서 Git 명령을 직접 수행할 수 있도록 할 수 있습니다. 이 옵션은 CodeBuild 다운스트림 작업에서만 사용할 수 있습니다.
자세히 알아보기:

- 구성 파라미터와 예제 JSON/YAML 코드 조각은 <u>Bitbucket Cloud, GitHub,</u> <u>GitHub Enterprise Server, GitLab.com 및 GitLab 자체 관리형 작업용</u> <u>CodeStarSourceConnection</u>을 참조하세요.
- 이 공급자 유형으로 연결을 생성하는 단계는 <u>GitLab 자체 관리형을 위한 연결</u> 섹션을 참조하십시오.

CodeCommit 소스 작업

CodeCommit은 클라우드에서 자산(예: 문서, 소스 코드, 바이너리 파일)을 비공개로 저장하여 관리할 수 있는 버전 관리 서비스입니다. CodeCommit 리포지토리의 브랜치를 코드에 대한 소스로 사용하도 록 CodePipeline을 구성할 수 있습니다. 저장소를 만들고 로컬 시스템의 작업 디렉터리와 연결합니다. 그런 다음 단계에서 소스 작업의 일환으로 브랜치를 사용하는 파이프라인을 만들 수 있습니다. 파이프 라인을 만들거나 기존 파이프라인을 편집하여 CodeCommit 리포지토리에 연결할 수 있습니다.

이 작업에 대한 전체 복제 옵션을 사용하여 리포지토리 Git 메타데이터를 참조하여 다운스트림 작업에 서 Git 명령을 직접 수행할 수 있도록 할 수 있습니다. 이 옵션은 CodeBuild 다운스트림 작업에서만 사 용할 수 있습니다.

자세히 알아보기:

- 구성 파라미터와 예제 JSON/YAML 코드 조각은 CodeCommit 소스 작업 참조을 참조하세요.
- 자습서: 간단한 파이프라인 생성(CodeCommit 리포지토리)
- CodePipeline은 Amazon CloudWatch Events를 사용하여 파이프라인의 소스로 사용된 CodeCommit 리포지토리의 변경을 감지합니다. 각 소스 작업에는 해당 이벤트 규칙이 있습니다. 이 이벤트 규칙은 리포지토리에 변경이 발생할 때 파이프라인을 시작합니다. <u>CodePipeline으로 일반 통</u> 합을(를) 참조하세요.

GitHub(OAuth 앱 사용) 소스 작업

GitHub(OAuth 앱 사용) 작업은 OAuth 앱으로 관리되는 버전 1 GitHub 작업입니다. 사용 가능한 리 전에서 GitHub 앱으로 GitHub 소스 작업을 관리하는 <u>Bitbucket Cloud, GitHub, GitHub Enterprise</u> <u>Server, GitLab.com 및 GitLab 자체 관리형 작업용 CodeStarSourceConnection</u> 소스 작업을 파이프라 인에서 사용할 수도 있습니다. GitHub(OAuth 앱 사용) 작업을 사용하는 파이프라인이 있는 경우에서 GitHub(GitHub 앱 사용) 작업을 사용하도록 업데이트하는 단계를 참조하세요<u>GitHub(OAuth 앱 사용)</u> 소스 작업을 GitHub(GitHub 앱 사용) 소스 작업으로 업데이트.

Note

GitHub(OAuth 앱 사용) 작업을 사용하지 않는 것이 좋지만 GitHub(OAuth 앱 사용) 작업이 있 는 기존 파이프라인은 아무런 영향 없이 계속 작동합니다.

자세히 알아보기:

- 앱 기반 GitHub 액세스와 달리 OAuth 기반 GitHub(OAuth 앱을 통해) 액세스에 대한 자세한 내용은 섹션을 참조하세요<u>https://docs.github.com/en/developers/apps/differences-between-github-apps-</u>and-oauth-apps.
- GitHub(OAuth 앱 사용) 작업 세부 정보가 포함된 부록을 보려면 섹션을 참조하세요<u>부록 A:</u> <u>GitHub(OAuth 앱 사용) 소스 작업</u>.

빌드 작업 통합

다음 정보는 CodePipeline 작업 유형별로 구성되어 있으며 다음 빌드 작업 공급자와 통합되도록 CodePipeline을 구성하는 데 도움이 됩니다.

주제

- <u>CodeBuild 빌드 작업</u>
- CloudBees 빌드 작업
- Amazon ECR 빌드 및 게시 작업
- Jenkins 빌드 작업
- TeamCity 빌드 작업

CodeBuild 빌드 작업

CodeBuild는 소스 코드를 컴파일하고 단위 테스트를 실행하며 배포할 준비가 완료된 아티팩트를 생성 하는 완전 관리형 빌드 서비스입니다.

파이프라인의 빌드 단계에 빌드 작업으로 CodeBuild를 추가할 수 있습니다. 자세한 내용은 <u>AWS</u> <u>CodeBuild 빌드 및 테스트 작업 참조</u>에 대한 CodePipeline 작업 구성 참조를 참조하세요.

Note

빌드 출력이 있든 없든 CodeBuild는 파이프라인에 테스트 작업으로 포함될 수 있습니다.

자세히 알아보기:

- 구성 파라미터와 예제 JSON/YAML 코드 조각은 <u>AWS CodeBuild 빌드 및 테스트 작업 참조</u>을 참조 하세요.
- CodeBuild란 무엇입니까?
- CodeBuild 완전 관리형 빌드 서비스

CloudBees 빌드 작업

CodePipeline이 <u>CloudBees</u>를 써서 파이프라인에서 하나 이상의 작업에 코드를 빌드하거나 테스트하 도록 구성할 수 있습니다.

자세히 알아보기:

re:INVENT 2017: Cloud First with AWS

Amazon ECR 빌드 및 게시 작업

Amazon ECR은 AWS Docker 이미지 리포지토리 서비스입니다. Docker의 push 명령과 pull 명령을 사용하여 Docker 이미지를 리포지토리에 업로드할 수 있습니다.

파이프라인에 ECRBuildAndPublish 작업을 추가하여 이미지 빌드 및 푸시를 자동화할 수 있습니다. 자세한 내용은에 대한 CodePipeline 작업 구성 참조를 참조하세요<u>ECRBuildAndPublish 빌드 작업</u> 참조.

Jenkins 빌드 작업

CodePipeline이 Jenkins CI를 써서 파이프라인에서 하나 이상의 작업에 코드를 빌드하거나 테스 트하도록 구성할 수 있습니다. 전에 만들어 둔 Jenkins 프로젝트가 있고 그 프로젝트에 Jenkins용 CodePipeline 플러그인을 설치하고 구성해 두었어야 합니다. 새 파이프라인을 만들거나 기존 파이프라 인을 편집하면 Jenkins 프로젝트에 연결할 수 있습니다.

프로젝트마다 Jenkins 액세스 권한이 구성되었습니다. CodePipeline과 함께 사용하려는 모든 Jenkins 인스턴스에 Jenkins용 CodePipeline 플러그인을 설치해야 합니다. 또한 Jenkins 프로젝트에 대한 CodePipeline 액세스를 구성해야 합니다. HTTPS/SSL 연결만 수락하도록 구성하여 Jenkins 프로젝트 의 보안을 구현합니다. Jenkins 프로젝트가 Amazon EC2 인스턴스에 설치된 경우 각 인스턴스 AWS CLI 에를 설치하여 AWS 자격 증명을 제공하는 것이 좋습니다. 그런 다음 연결에 사용할 자격 증명을 사용하여 해당 인스턴스에 AWS 프로파일을 구성합니다. 이것은 Jenkins 웹 인터페이스를 통해 추가하 고 저장하는 대안입니다.

자세히 알아보기:

- Jenkins 액세스
- 자습서: 4단계 파이프라인 생성

TeamCity 빌드 작업

CodePipeline이 <u>TeamCity</u>를 써서 파이프라인에서 하나 이상의 작업에 코드를 빌드하고 테스트하도록 구성할 수 있습니다.

자세히 알아보기:

• CodePipeline용 TeamCity Plugin

테스트 작업 통합

다음 정보는 CodePipeline 작업 유형별로 구성되어 있으며 다음 테스트 작업 공급자와 통합되도록 CodePipeline을 구성하는 데 도움이 됩니다.

주제

- CodeBuild 테스트 작업
- AWS Device Farm 테스트 작업
- Ghost Inspector 테스트 작업
- OpenText LoadRunner Cloud 테스트 작업
- 테스트 자동화 반영

CodeBuild 테스트 작업

<u>CodeBuild</u>는 클라우드상의 완전 관리형 빌드 서비스입니다. CodeBuild는 소스 코드를 컴파일하고 단 위 테스트를 실행하며 배포 준비가 완료된 아티팩트를 생성합니다. 테스트 작업으로 파이프라인에 CodeBuild를 추가할 수 있습니다. 자세한 내용은 <u>AWS CodeBuild 빌드</u> 및 테스트 작업 참조에 대한 CodePipeline 작업 구성 참조를 참조하세요.

Note

필수 빌드 출력 아티팩트를 이용해 CodeBuild는 파이프라인에 빌드 작업으로 포함될 수 있습 니다.

자세히 알아보기:

- 구성 파라미터와 예제 JSON/YAML 코드 조각은 <u>AWS CodeBuild 빌드 및 테스트 작업 참조</u>을 참조 하세요.
- CodeBuild란 무엇입니까?

AWS Device Farm 테스트 작업

AWS Device Farm은 실제 휴대폰 및 태블릿에서 Android, iOS 및 웹 애플리케이션을 테스트하고 상 호 작용할 수 있는 앱 테스트 서비스입니다. 파이프라인의 하나 이상의 작업에서 코드를 테스트 AWS Device Farm 하는 데 사용하도록 CodePipeline을 구성할 수 있습니다.를 AWS Device Farm 사용하면 자체 테스트를 업로드하거나 스크립트가 없는 기본 제공 호환성 테스트를 사용할 수 있습니다. 테스트 는 병렬로 수행되기 때문에 여러 디바이스의 테스트가 몇 분 안에 시작됩니다. 테스트가 완료되면 상위 수준 결과, 하위 수준 로그, pixel-to-pixel 스크린샷 및 성능 데이터가 포함된 테스트 보고서가 업데이트 됩니다.는 PhoneGap, Titanium, Xamarin, Unity 및 기타 프레임워크로 생성된 앱을 포함하여 네이티브 및 하이브리드 Android, iOS 및 Fire OS 앱의 테스트를 AWS Device Farm 지원합니다. 테스트 디바이 스와 직접 상호작용할 수 있도록 Android 앱의 원격 액세스를 지원합니다.

자세히 알아보기:

- 구성 파라미터와 예제 JSON/YAML 코드 조각은 AWS Device Farm 테스트 작업 참조을 참조하세요.
- 란 무엇입니까 AWS Device Farm?
- AWS Device Farm CodePipeline 테스트 단계에서 사용

Ghost Inspector 테스트 작업

CodePipeline이 <u>Ghost Inspector</u>를 써서 파이프라인에서 하나 이상의 작업에 코드를 테스트하도록 구 성할 수 있습니다.

자세히 알아보기:

• CodePipeline으로 서비스를 통합하기 위한 Ghost Inspector 설명서

OpenText LoadRunner Cloud 테스트 작업

CodePipeline이 파이프라인에서 하나 이상의 작업에 <u>OpenText LoadRunner Cloud</u>를 사용하도록 구성 할 수 있습니다.

자세히 알아보기:

• CodePipeline으로 통합을 위한 LoadRunner Cloud 설명서

테스트 자동화 반영

Reflect는 테스트를 간소화하고 수동 프로세스의 문제를 해결할 수 있는 AI 기반 테스트 자동화 솔루션 입니다. 코드 없는 테스트 자동화를 통해 Reflect는 테스트 생성, 실행 및 유지 관리를 간소화하므로 기 술 지식 없이도 강력하고 반복 가능한 테스트를 생성할 수 있습니다. 복잡성을 제거하고 워크플로 중단 을 최소화하면 테스트를 가속화하고 매번 고품질 애플리케이션을 자신 있게 제공할 수 있습니다.

자세히 알아보기:

• AWS CodePipeline Reflect와의 통합 테스트

배포 작업 통합

다음 정보는 CodePipeline 작업 유형별로 구성되어 있으며 다음 배포 작업 공급자와 통합되도록 CodePipeline을 구성하는 데 도움이 됩니다.

주제

- Amazon EC2 배포 작업
- Amazon Elastic Kubernetes Service EKS 배포 작업
- Amazon S3 배포 작업
- AWS AppConfig 배포 작업
- <u>AWS CloudFormation 작업 배포</u>
- AWS CloudFormation StackSets 배포 작업

- Amazon ECS 배포 작업
- Elastic Beanstalk 배포 작업
- AWS OpsWorks 작업 배포
- <u>Service Catalog 배포 작업</u>
- <u>Amazon Alexa 배포 작업</u>
- <u>CodeDeploy 배포 작업</u>
- XebiaLabs 배포 작업

Amazon EC2 배포 작업

<u>Amazon EC2</u>를 사용하면 클라우드에서 컴퓨팅을 생성하고 관리할 수 있습니다. Amazon EC2를 인스 턴스에 애플리케이션을 배포하는 배포 공급자로 사용하는 파이프라인에 작업을 추가할 수 있습니다.

자세히 알아보기:

- 의 작업 참조 페이지를 참조하세요Amazon EC2 작업 참조.
- 자습서는 자습서: CodePipeline을 사용하여 Amazon EC2 인스턴스에 배포을 참조하세요.

Amazon Elastic Kubernetes Service EKS 배포 작업

Amazon EKS를 사용하면 kubernetes 클러스터를 생성하고 관리할 수 있습니다. Amazon EKS를 클러 스터에 이미지를 배포하는 배포 공급자로 사용하는 파이프라인에 작업을 추가할 수 있습니다. helm 템 플릿 또는 kubernetes 매니페스트 파일을 사용할 수 있습니다.

자세히 알아보기:

- 의 작업 참조 페이지를 참조하세요Amazon Elastic Kubernetes Service EKS 배포 작업 참조.
- 자습서는 자습서: CodePipeline을 사용하여 Amazon EKS에 배포을 참조하세요.

Amazon S3 배포 작업

Amazon S3는 인터넷 스토리지입니다. Amazon S3를 사용하면 인터넷을 통해 언제 어디서든 원하는 양의 데이터를 저장하고 검색할 수 있습니다. 이제 Amazon S3를 배포 제공자로 사용하는 파이프라인 에 작업을 추가할 수 있습니다.

1 Note

Amazon S3는 파이프라인에 소스 작업으로 포함될 수 있습니다.

자세히 알아보기:

- 파이프라인 스테이지 및 작업 생성
- 자습서: Amazon S3를 배포 공급자로 사용하는 파이프라인 생성

AWS AppConfig 배포 작업

AWS AppConfig는 애플리케이션 구성을 생성, 관리 및 빠르게 배포 AWS Systems Manager 하는의 기 능입니다. EC2 인스턴스, AWS Lambda컨테이너, 모바일 애플리케이션 또는 IoT 디바이스에서 호스팅 되는 애플리케이션과 함께 AppConfig를 사용할 수 있습니다.

자세히 알아보기:

- AWS AppConfig 배포 작업 참조에 대한 CodePipeline 작업 구성 참조
- 자습서: AWS AppConfig를 배포 공급자로 사용하는 파이프라인 생성

AWS CloudFormation 작업 배포

AWS CloudFormation는 개발자와 시스템 관리자에게 템플릿을 사용하여 해당 AWS 리소스를 프로비 저닝하고 업데이트하여 관련 리소스 모음을 쉽게 생성하고 관리할 수 있는 방법을 제공합니다. 서비스 의 샘플 템플릿을 사용하거나 직접 만들 수 있습니다. 템플릿은 애플리케이션을 실행하는 데 필요한 AWS 리소스와 모든 종속성 또는 런타임 파라미터를 설명합니다.

AWS 서버리스 애플리케이션 모델(AWS SAM)은를 확장 AWS CloudFormation 하여 서버리스 애플리 케이션을 정의하고 배포하는 간소화된 방법을 제공합니다. AWS SAM은 Amazon API Gateway APIs, AWS Lambda 함수 및 Amazon DynamoDB 테이블을 지원합니다. CodePipeline을 및 AWS SAM과 함 께 AWS CloudFormation 사용하여 서버리스 애플리케이션을 지속적으로 제공할 수 있습니다.

를 배포 공급자 AWS CloudFormation 로 사용하는 파이프라인에 작업을 추가할 수 있습니다. 를 배포 공급자 AWS CloudFormation 로 사용하면 AWS CloudFormation 스택에 대한 작업을 수행하고 파이프 라인 실행의 일부로 세트를 변경할 수 있습니다.는 파이프라인이 실행될 때 스택 및 변경 세트를 생성, 업데이트, 교체 및 삭제할 수 AWS CloudFormation 있습니다. 따라서 AWS CloudFormation 템플릿 및 파라미터 정의에 제공하는 사양에 따라 파이프라인 실행 중에 사용자 AWS 지정 리소스를 생성, 프로 비저닝, 업데이트 또는 종료할 수 있습니다.

자세히 알아보기:

- AWS CloudFormation 배포 작업 참조에 대한 CodePipeline 작업 구성 참조
- <u>CodePipeline을 사용한 지속적 전달</u> CodePipeline을 사용하여 지속적인 전달 워크플로를 구축하는 방법을 알아봅니다 AWS CloudFormation.
- Lambda 기반 애플리케이션 배포 자동화 AWS 서버리스 애플리케이션 모델을 사용하고 Lambda 기반 애플리케이션을 위한 지속적인 전송 워크플로 AWS CloudFormation 를 구축하는 방법을 알아 봅니다.

AWS CloudFormation StackSets 배포 작업

AWS CloudFormation를 사용하면 여러 계정 및 AWS 리전에 리소스를 배포할 수 있습니다.

CodePipeline AWS CloudFormation 을와 함께 사용하여 스택 세트 정의를 업데이트하고 인스턴스에 업데이트를 배포할 수 있습니다.

파이프라인에 다음 작업을 추가하여 AWS CloudFormation StackSets를 배포 공급자로 사용할 수 있습 니다.

- CloudFormationStackSet
- CloudFormationStackInstances

자세히 알아보기:

- AWS CloudFormation StackSets 배포 작업 참조에 대한 CodePipeline 작업 구성 참조
- 자습서: AWS CloudFormation StackSets 배포 작업을 사용하여 파이프라인 생성

Amazon ECS 배포 작업

Amazon ECS는 확장성과 성능이 뛰어난 컨테이너 관리 서비스로서 AWS 클라우드에서 컨테이너 기반 애플리케이션을 실행할 수 있습니다. 파이프라인을 만들 때 Amazon ECS를 배포 공급자로 선 택할 수 있습니다. 소스 제어 리포지토리의 코드를 변경하면 파이프라인이 새 도커 이미지를 빌드하 고 이를 컨테이너 레지스트리에 푸시한 후 업데이트된 이미지를 Amazon ECS에 배포합니다. 또한 CodePipeline의 ECS(블루/그린) 공급자 작업을 사용하여 CodeDeploy를 통해 트래픽을 Amazon ECS 로 라우팅하고 배포할 수 있습니다. 자세히 알아보기:

- Amazon ECS란 무엇입니까?
- 자습서: CodePipeline을 사용하여 연속 배포
- 파이프라인 스테이지 및 작업 생성
- <u>자습서: Amazon ECR 소스 및 ECS-to-CodeDeploy 배포를 사용하여 파이프라인 생성</u>

Elastic Beanstalk 배포 작업

Elastic Beanstalk는 Java, .NET, PHP, Node.js, Python, Ruby, Go, Docker를 사용하여 개발된 웹 애플 리케이션 및 서비스를 Apache, Nginx, Passenger, IIS와 같은 친숙한 서버에 배포하고 확장하기 위한 서비스입니다. Elastic Beanstalk로 코드를 배포하도록 CodePipeline을 구성할 수 있습니다. 파이프라 인을 생성하기 전 단계나 파이프라인 생성 마법사를 사용할 때 Elastic Beanstalk 애플리케이션 및 환경 을 생성하여 배포 작업에 사용할 수 있습니다.

1 Note

이 기능은 아시아 태평양(하이데라바드), 아시아 태평양(멜버른), 중동(UAE), 유럽(스페인), 유 럽(취리히) 리전에서 사용할 수 없습니다. 사용 가능한 다른 작업을 참조하려면 <u>CodePipeline</u> <u>과 제품 및 서비스 통합</u>을 참조하세요.

자세히 알아보기:

- Elastic Beanstalk 사용 시작하기
- 파이프라인 스테이지 및 작업 생성

AWS OpsWorks 작업 배포

AWS OpsWorks 는 Chef를 사용하여 모든 셰이프 및 크기의 애플리케이션을 구성하고 운영하는 데 도 움이 되는 구성 관리 서비스입니다. 를 사용하면 패키지 설치 AWS OpsWorks Stacks, 소프트웨어 구 성 및 스토리지와 같은 리소스를 포함하여 애플리케이션의 아키텍처와 각 구성 요소의 사양을 정의할 수 있습니다. 에서 사용자 지정 Chef 쿡북 및 애플리케이션과 함께 코드를 배포하는 AWS OpsWorks Stacks 데 사용하도록 CodePipeline을 구성할 수 있습니다 AWS OpsWorks.

• 사용자 지정 Chef 쿡북 - Chef 쿡북을 AWS OpsWorks 사용하여 패키지 설치 및 구성, 애플리케이션 배포와 같은 작업을 처리합니다.

• 애플리케이션 - AWS OpsWorks 애플리케이션은 애플리케이션 서버에서 실행하려는 코드로 구성됩니다. 애플리케이션 코드는 Amazon S3 버킷과 같은 리포지토리에 저장됩니다.

파이프라인을 생성하기 전에 AWS OpsWorks 스택과 계층을 생성합니다. 파이프라인을 생성하기 전에 또는 파이프라인 생성 마법사를 사용할 때 단계의 배포 작업에 사용할 AWS OpsWorks 애플리케이션 을 생성할 수 있습니다.

에 대한 CodePipeline 지원 AWS OpsWorks 은 현재 미국 동부(버지니아 북부) 리전(us-east-1)에서만 사용할 수 있습니다.

자세히 알아보기:

- AWS OpsWorks Stacks에서 CodePipeline 사용
- 쿡북과 레시피
- AWS OpsWorks 앱

Service Catalog 배포 작업

<u>Service Catalog</u>를 사용하면 조직에서 사용이 승인된 제품 카탈로그를 생성하고 관리할 수 있습니다 AWS.

CodePipeline을 구성하여 제품 템플릿의 업데이트 및 버전을 Service Catalog에 배포할 수 있습니다. 배포 작업에 사용할 Service Catalog 제품을 만든 다음 파이프라인 생성 마법사를 사용하여 파이프라 인을 만들 수 있습니다.

자세히 알아보기:

- <u>자습서: Service Catalog에 배포하는 파이프라인 생성</u>
- 파이프라인 스테이지 및 작업 생성

Amazon Alexa 배포 작업

Amazon Alexa Skills Kit</mark>를 통해 클라우드 기반 기술을 구축하고 Alexa 지원 디바이스 사용자에게 배포 할 수 있습니다.

Note

이 기능은 아시아 태평양(홍콩) 또는 유럽(밀라노) 리전에서 사용할 수 없습니다. 해당 리전에 서 사용 가능한 다른 배포 작업을 사용하려면 배포 작업 통합을 참조하세요.

이제 Alexa Skills Kit를 배포 공급자로 사용하는 파이프라인에 작업을 추가할 수 있습니다. 소스 변경이 파이프라인에서 감지되며, 그 다음에 파이프라인은 Alexa 서비스에서 사용자의 Alexa 기술에 대한 업 데이트를 배포합니다.

자세히 알아보기:

• 자습서: Amazon Alexa Skill을 배포하는 파이프라인 생성

CodeDeploy 배포 작업

CodeDeploy는 Amazon EC2/온프레미스 인스턴스, Amazon Elastic Container Service 컴퓨팅 플랫폼 및 서버리스 AWS Lambda 컴퓨팅 플랫폼에 대한 애플리케이션 배포를 조정합니다. CodeDeploy로 코 드를 배포하도록 CodePipeline을 구성할 수 있습니다. 파이프라인을 생성하기 전 단계나 파이프라인 생성 마법사를 사용할 때 배포 작업에 사용할 CodeDeploy 애플리케이션, 배포 및 배포 그룹을 생성할 수 있습니다.

자세히 알아보기:

- 3단계: CodeDeploy에서 애플리케이션 생성
- 자습서: 간단한 파이프라인 생성(CodeCommit 리포지토리)

XebiaLabs 배포 작업

CodePipeline이 <u>XebiaLabs</u>를 써서 파이프라인에서 하나 이상의 작업에 코드를 배포하도록 구성할 수 있습니다.

자세히 알아보기:

• CodePipeline으로 XL Deploy 사용

Amazon Simple Notification Service와 승인 작업 통합

Amazon SNS는 빠르고 유연한 완전관리형 푸시 알림 서비스로서, 이 서비스를 사용하면 개별 메시지 를 전송하거나 대규모의 수신자에게 메시지를 전송할 수 있습니다. Amazon SNS를 사용하면 간편하 고 비용 효과적으로 모바일 디바이스 사용자와 이메일 수신자에게 푸시 알림을 보내거나 다른 배포된 서비스에도 메시지를 보낼 수 있습니다.

CodePipeline에서 수동 승인을 새로 만들 때, 승인 작업을 검토할 준비가 되었다는 알림을 모든 IAM 구 독 사용자가 받을 수 있도록 Amazon SNS에 주제를 게시하는 옵션이 있습니다.

자세히 알아보기:

- Amazon SNS란 무엇인가요?
- CodePipeline 서비스 역할에 Amazon SNS 권한 부여

호출 작업 통합

다음 정보는 CodePipeline 작업 유형별로 구성되어 있으며 다음 호출 작업 공급자와 통합되도록 CodePipeline을 구성하는 데 도움이 됩니다.

주제

- <u>Amazon Inspector 호출 작업</u>
- Lambda 호출 작업
- Step Functions 호출 작업

Amazon Inspector 호출 작업

Amazon Inspector는 자동으로 워크로드를 검색하고 소프트웨어 취약성 및 의도하지 않은 네트워크 노 출을 지속적으로 검사하는 취약성 관리 서비스입니다. Amazon Inspector는 tar 및 war를 포함한 여러 아카이브 형식을 지원하며, Amazon Inspector는 Rust 및 Go 바이너리를 포함한 바이너리를 지원합니 다.

소스 코드 또는 Amazon ECR 이미지 리포지토리의 취약성 스캔을 자동화하도록 CodePipeline InspectorScan 작업을 구성할 수 있습니다.

자세히 알아보기:

• 에 대한 CodePipeline 작업 구성 참조 Amazon Inspector InspectorScan 호출 작업 참조

Lambda 호출 작업

Lambda를 사용하면 서버를 프로비저닝하거나 관리하지 않고 코드를 실행할 수 있습니다. Lambda 함 수를 써서 파이프라인에 유연성과 기능성을 추가하도록 CodePipeline을 구성할 수 있습니다. 파이프라 인을 생성하기 전 단계나 파이프라인 생성 마법사를 사용할 때 Lambda 함수를 생성하여 작업으로 추 가할 수 있습니다.

자세히 알아보기:

- 에 대한 CodePipeline 작업 구성 참조 AWS Lambda 작업 참조 호출
- CodePipeline의 파이프라인에서 AWS Lambda 함수 호출

Step Functions 호출 작업

Step Functions를 사용하면 상태 시스템을 생성하고 구성할 수 있습니다. Step Functions 호출 작업을 사용하여 상태 시스템 실행을 트리거하도록 CodePipeline을 구성할 수 있습니다.

자세히 알아보기:

- AWS Step Functions 작업 참조 호출에 대한 CodePipeline 작업 구성 참조
- <u>자습서: 파이프라인에서 AWS Step Functions 호출 작업 사용</u>

CodePipeline으로 일반 통합

다음 AWS 서비스 통합은 CodePipeline 작업 유형을 기반으로 하지 않습니다.

Amazon CloudWatch	Amazon CloudWatch는 AWS 리소스를 모니터링합니다.
	자세히 알아보기:
	• <u>Amazon CloudWatch란 무엇인가요?</u>
Amazon EventBridge	Amazon EventBridge는 정의한 규칙에 AWS 서비스 따라의 변경 사항을 감지하고 변경 사항이 발생할 AWS 서비스 때 지정된 하나 이상의에서 작업을 호출하는 웹 서비스입니다.
	• 변경이 발생하면 자동으로 파이프라인 실행 시작 - Amazon EventBrid ge에 설정된 규칙의 대상으로 CodePipeline을 구성할 수 있습니다. 그

	러면 다른 서비스가 변경되는 경우 파이프라인이 자동으로 시작됩니 다.
	자세히 알아보기:
	• <u>Amazon EventBridge란 무엇인가요?</u>
	• CodePipeline에서 파이프라인 시작
	• <u>CodeCommit 소스 작업 및 EventBridge</u>
	• 파이프라인 상태가 면경되면 알림 수신 - 파이프라인, 단계 또는 작업 의 실행 상태에서 변경을 감지하고 대응하도록 EventBridge 규칙을 설 정할 수 있습니다.
	자세히 알아보기:
	• CodePipeline 이벤트 모니터링
	• <u>자습서: 파이프라인 상태 변경에 대한 이메일 알림을 수신하도록</u> CloudWatch Events 규칙 설정
AWS Cloud9	AWS Cloud9 는 웹 브라우저를 통해 액세스하는 온라인 IDE입니다. IDE 는 여러 프로그래밍 언어와 런타임 디버거 및 터미널을 갖춘 강력한 코 드 편집 환경을 제공합니다. 백그라운드에서 Amazon EC2 인스턴스는 AWS Cloud9 개발 환경을 호스팅합니다. 자세한 내용은 <u>AWS Cloud9 사</u> <u>용 설명서</u> 를 참조하십시오.
	자세히 알아보기:
	• <u>AWS Cloud9설정</u>
AWS CloudTrail	CloudTrail은 AWS 계정에서 또는 계정을 대신하여 수행한 AWS API 호 출 및 관련 이벤트를 캡처하고 사용자가 지정한 Amazon S3 버킷에 로 그 파일을 전송합니다. CodePipeline 콘솔,의 CodePipeline 명령 AWS CLI및 CodePipeline API에서 CodePipeline API 호출을 캡처하도록 CloudTrail을 구성할 수 있습니다.
	자세히 알아보기:
	• 를 사용하여 CodePipeline API 호출 로깅 AWS CloudTrail

AWS CodeStar 알림 알림을 사용하면 파이프라인 실행이 시작된 것과 같은 중요한 변경 사항 을 사용자에게 알릴 수 있습니다. 자세한 내용은 <u>알림 규칙 생성</u> 단원을 참조하십시오. AWS Key Managemen t Service AWS KMS는 데이터 암호화에 사용하는 암호화 키를 쉽게 생성하고 제 어할 수 있게 해주는 관리형 서비스입니다. 기본적으로 CodePipeline은 AWS KMS 를 사용하여 Amazon S3 버킷에 저장된 파이프라인의 아티팩 트를 암호화합니다.

자세히 알아보기:

- 한 AWS 계정의 소스 버킷, 아티팩트 버킷 및 서비스 역할과 다른 AWS 계정의 CodeDeploy 리소스를 사용하는 파이프라인을 생성하려면 고 객 관리형 KMS 키를 생성하고, 파이프라인에 키를 추가하고, 계정 정 책 및 역할을 설정하여 교차 계정 액세스를 활성화해야 합니다. 자세한 내용은 <u>CodePipeline에서 다른 AWS 계정의 리소스를 사용하는 파이</u> 프라인 생성 단원을 참조하십시오.
- AWS CloudFormation 스택을 다른 AWS 계정에 배포하는 파이프라인 을 한 AWS 계정에서 생성하려면 고객 관리형 KMS 키를 생성하고, 파 이프라인에 키를 추가하고, 스택을 다른 계정에 배포하도록 AWS 계정 정책 및 역할을 설정해야 합니다. 자세한 내용은 <u>CodePipeline을 사용</u> 하여 다른 계정에 AWS CloudFormation 스택을 배포하려면 어떻게 해 야 하나요?를 참조하세요.
- 파이프라인의 S3 아티팩트 버킷에 대한 서버 측 암호화를 구성하려면 기본 AWS 관리형 KMS 키를 사용하거나 고객 관리형 KMS 키를 생성 하고 암호화 키를 사용하도록 버킷 정책을 설정할 수 있습니다. 자세한 내용은 <u>CodePipeline의 Amazon S3에 저장된 아티팩트에 대해 서버</u> <u>측 암호화 구성</u> 단원을 참조하십시오.

AWS KMS key의 경우 키 ID, 키 ARN 또는 별칭 ARN을 사용할 수 있습니다.

Note

별칭은 KMS 키를 생성한 계정에서만 인식됩니다. 교차 계정 작 업의 경우 키 ID 또는 키 ARN만 사용하여 키를 식별할 수 있습니 다. 계정 간 작업에는 다른 계정(AccountB)의 역할을 사용하는 것이 포함되므로 키 ID를 지정하면 다른 계정(AccountB)의 키가 사용됩니다.

커뮤니티 예제

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

1 Note

이러한 링크는 정보 제공 목적으로만 제공되며 예제의 콘텐츠에 대한 포괄적인 목록 또는 보증 으로 간주해서는 안 됩니다. AWS 는 외부 콘텐츠의 콘텐츠 또는 정확성에 대해 책임을 지지 않 습니다.

주제

• 통합 예: 블로그 게시물

통합 예: 블로그 게시물

• 타사 Git 리포지토리에서 AWS CodePipeline 빌드 상태 추적

개발자가 컨텍스트를 전환하지 않고도 쉽게 상태를 추적할 수 있도록 타사 리포지토리에 파이프라 인 및 빌드 작업 상태를 표시할 리소스를 설정하는 방법을 알아보세요.

2021년 3월 발행

• <u>AWS CodeBuildAWS CodeDeploy및 AWS CodeCommit를 사용하여 CI/CD 완료 AWS</u> CodePipeline

CodeCommit, CodePipeline, CodeBuild 및 CodeDeploy 서비스를 사용하여 Amazon EC2 Linux 인 스턴스 세트에 버전 제어 Java 애플리케이션을 컴파일, 빌드 및 설치하는 파이프라인을 설정하는 방 법을 알아봅니다.

2020년 9월 발행

• CodePipeline을 사용하여 GitHub에서 Amazon EC2로 배포하는 방법

개발, 테스트 및 프로덕션 브랜치를 별도의 배포 그룹에 배포하도록 CodePipeline을 처음부터 설정 하는 방법을 알아봅니다. CodePipeline과 함께 IAM 역할, CodeDeploy 에이전트 및 CodeDeploy를 사용하고 구성하는 방법을 알아봅니다.

2020년 4월 발행

• AWS Step Functions에 대한 CI/CD 파이프라인 테스트 및 생성

커뮤니티 예제

Step Functions 상태 머신과 파이프라인을 조정할 리소스를 설정하는 방법을 알아봅니다.

2020년 3월 발행

• CodePipeline을 사용한 DevSecOps 구현

CodePipeline에서 CI/CD 파이프라인을 이용해 예방성 감지 보안 통제를 자동화하는 방법을 알아봅 니다. 이 게시물에서는 파이프라인을 사용하여 간단한 보안 그룹을 생성하고 소스, 테스트 및 프로덕 션 단계에서 보안 검사를 수행하여 AWS 계정의 보안 태세를 개선하는 방법을 다룹니다.

2017년 3월 발행

• <u>CodePipeline, CodeBuild, Amazon ECR 및를 사용하여 Amazon ECS에 지속적으로 배포 AWS</u> CloudFormation

Amazon Elastic Container Service(Amazon ECS)에 대한 지속적 배포 파이프라인을 생성하는 방법 을 알아봅니다. 애플리케이션은 CodePipeline, CodeBuild, Amazon ECR 및 AWS CloudFormation 을 이용한 Docker 컨테이너로 제공됩니다.

• 샘플 AWS CloudFormation 템플릿과 이를 사용하여 GitHub의 <u>ECS 참조 아키텍처: 연속 배포 리</u> <u>포지토리에서 자체 연속 배포</u> 파이프라인을 생성하기 위한 지침을 다운로드합니다.

2017년 1월 발행

• 서버리스 애플리케이션을 위한 지속적 배포

의 모음을 사용하여 서버리스 애플리케이션을 위한 지속적 배포 파이프라인을 AWS 서비스 생성하 는 방법을 알아봅니다. SAM(Serverless Application Model)을 이용해 애플리케이션과 그 리소스를 정의하면 CodePipeline이 애플리케이션 배포를 조율합니다.

• Go with the Gin 프레임워크와 API Gateway 프록시 심에 쓴 샘플 애플리케이션을 보십시오.

2016년 12월 발행

• <u>CodePipeline과 Dynatrace로</u> DevOps 배포 확장

Dynatrace 모니터링 솔루션을 이용해 CodePipeline에서 파이프라인을 확장하고, 코드가 커밋되기 전에 테스트 실행을 자동으로 분석하고, 최적의 리드 타임을 유지하는 법을 알아봅니다.

2016년 11월 발행

• <u>AWS CloudFormation 및 CodeCommit을 사용하여 CodePipeline AWS Elastic Beanstalk 에서에 대</u> 한 파이프라인 생성 CodeCommit AWS Elastic Beanstalk의 애플리케이션용 CodePipeline 파이프라인에서 지속적 제공을 실행하는 방법을 알아봅니다. 모든 AWS 리소스는 템플릿을 사용하여 자동으로 프로비저닝됩니다 AWS CloudFormation . 이 연습에서는 CodeCommit 및 AWS Identity and Access Management (IAM)도 통합합니다.

2016년 5월 발행

• 에서 CodeCommit 및 CodePipeline 자동화 AWS CloudFormation

AWS CloudFormation 를 사용하여 CodeCommit, CodePipeline, CodeDeploy CodeDeploy 및를 사용하는 지속적 전송 파이프라인에 대한 AWS 리소스 프로비저닝을 자동화합니다 AWS Identity and Access Management.

2016년 4월 발행

• 에서 교차 계정 파이프라인 생성 AWS CodePipeline

AWS Identity and Access Management을 이용해 AWS CodePipeline 의 파이프라인에 교차 계정 액 세스의 프로비저닝을 자동화하는 방법을 알아봅니다. AWS CloudFormation 템플릿에 예제를 포함 합니다.

2016년 3월 발행

• ASP.NET Core 파트 2: 지속적 제공 둘러보기

CodeDeploy 및를 사용하여 ASP.NET Core 애플리케이션에 대한 전체 연속 전송 시스템을 생성하는 방법을 알아봅니다 AWS CodePipeline.

2016년 3월 발행

• AWS CodePipeline 콘솔을 사용하여 파이프라인 생성

AWS CodePipeline 콘솔을 사용하여를 기반으로 연습에서 2단계 파이프라인을 생성하는 방법을 알 아봅니다 AWS CodePipeline 자습서: 4단계 파이프라인 생성.

2016년 3월 발행

• <u>를 사용하여 AWS CodePipeline 파이프라인 모의 AWS Lambda</u>

파이프라인이 작동하기 전 CodePipeline 소프트웨어 전달 프로세스를 설계할 때 여기에서 작업 및 스테이지를 시각적으로 보여주는 Lambda 함수 호출 방법을 알아봅니다. 파이프라인 구조를 설계하 면서 Lambda 함수를 이용해 파이프라인이 성공적으로 완료될 것인지 테스트할 수 있습니다. 2016년 2월 발행

• <u>를 사용하여 CodePipeline에서 AWS Lambda 함수 실행 AWS CloudFormation</u>

사용 설명서 작업에 사용되는 모든 AWS 리소스를 프로비저닝하는 AWS CloudFormation 스택을 생성하는 방법을 알아봅니다CodePipeline의 파이프라인에서 AWS Lambda 함수 호출.

2016년 2월 발행

• 에서 사용자 지정 CodePipeline 작업 프로비저닝 AWS CloudFormation

를 사용하여 CodePipeline에서 사용자 지정 작업을 프로비저닝 AWS CloudFormation 하는 방법을 알아봅니다.

2016년 1월 게시

• 를 사용하여 CodePipeline 프로비저닝 AWS CloudFormation

AWS CloudFormation을 이용해 CodePipeline에서 기본적인 지속적 제공 파이프라인을 프로비저닝 하는 방법을 알아봅니다.

2015년 12월 발행

• 사용자 지정 작업 및를 AWS OpsWorks 사용하여 CodePipeline에서 로 배포 AWS Lambda

CodePipeline을 AWS OpsWorks 사용하여에 배포할 파이프라인과 AWS Lambda 함수를 구성하는 방법을 알아봅니다.

2015년 7월 발행

CodePipeline 자습서

의 단계를 완료한 후이 사용 설명서의 AWS CodePipeline 자습서 중 하나를 사용해 <u>CodePipeline 시작</u> 하기볼 수 있습니다.

주제

- 자습서: CodePipeline을 사용하여 Amazon EC2 인스턴스에 배포
- 자습서: CodePipeline을 사용하여 Docker 이미지를 빌드하고 Amazon ECR에 푸시(V2 유형)
- 자습서: CodePipeline을 사용하여 Amazon EKS에 배포
- 자습서: 컴퓨팅을 사용하여 명령을 실행하는 파이프라인 생성(V2 유형)
- 자습서: Git 태그를 사용하여 파이프라인 시작하기
- 자습서: 파이프라인을 시작하기 위한 풀 요청의 브랜치 이름 필터링(V2 유형)
- 자습서: 파이프라인 수준 변수 사용
- 자습서: 간단한 파이프라인 생성(S3 버킷)
- 자습서: 간단한 파이프라인 생성(CodeCommit 리포지토리)
- 자습서: 4단계 파이프라인 생성
- 자습서: 파이프라인 상태 변경에 대한 이메일 알림을 수신하도록 CloudWatch Events 규칙 설정
- 자습서:를 사용하여 Android 앱을 빌드하고 테스트하는 파이프라인 생성 AWS Device Farm
- 자습서:를 사용하여 iOS 앱을 테스트하는 파이프라인 생성 AWS Device Farm
- 자습서: Service Catalog에 배포하는 파이프라인 생성
- 자습서:를 사용하여 파이프라인 생성 AWS CloudFormation
- 자습서: AWS CloudFormation 배포 작업의 변수를 사용하는 파이프라인 생성
- 자습서: CodePipeline을 사용한 Amazon ECS 표준 배포
- 자습서: Amazon ECR 소스 및 ECS-to-CodeDeploy 배포를 사용하여 파이프라인 생성
- 자습서: Amazon Alexa Skill을 배포하는 파이프라인 생성
- <u>자습서: Amazon S3를 배포 공급자로 사용하는 파이프라인 생성</u>
- <u>자습서: 서버리스 애플리케이션을에 게시하는 파이프라인 생성 AWS Serverless Application</u> <u>Repository</u>
- <u>자습서: Lambda 호출 작업과 함께 변수 사용</u>
- 자습서: 파이프라인에서 AWS Step Functions 호출 작업 사용

- 자습서: AWS AppConfig를 배포 공급자로 사용하는 파이프라인 생성
- 자습서: GitHub 파이프라인 소스와 함께 전체 복제 사용
- 자습서: CodeCommit 파이프라인 소스와 함께 전체 복제 사용
- 자습서: AWS CloudFormation StackSets 배포 작업을 사용하여 파이프라인 생성
- 자습서: 입력 조건으로 파이프라인에 대한 변수 확인 규칙 생성

자습서: CodePipeline을 사용하여 Amazon EC2 인스턴스에 배포

이 자습서는 CodePipeline에서 Amazon EC2에서 구성한 인스턴스에 코드를 배포하는 배포 작업을 생 성하는 데 도움이 됩니다.

Note

콘솔에서 파이프라인을 생성하는 과정에서 CodePipeline은 S3 아티팩트 버킷을 아티팩트에 사용합니다. (이는 S3 소스 작업에 사용되는 버킷과 다릅니다.) S3 아티팩트 버킷이 파이프라 인의 계정과 다른 계정에 있는 경우 S3 아티팩트 버킷을 안전하고 신뢰할 수 AWS 계정 있는에 서 소유해야 합니다.

Note

EC2 배포 작업은 V2 유형 파이프라인에서만 사용할 수 있습니다.

사전 조건

이 자습서를 이용하여 CD 파이프라인을 만들려면 먼저 몇 가지 리소스를 갖춰야 합니다. 다음은 시작 하기 위해 필요한 항목입니다.

1 Note

이러한 모든 리소스는 동일한 AWS 리전 내에 생성되어야 합니다.

• 샘플 script.sh 파일을 추가할 소스 제어 리포지토리(이 자습서에서는 GitHub 사용).

 이 작업에 대한 권한으로 업데이트된 기존 CodePipeline 서비스 역할을 사용해야 합니다. 서비스 역 할을 업데이트하려면 섹션을 참조하세요EC2 배포 작업에 대한 서비스 역할 정책 권한.

이러한 사전 조건을 모두 갖췄으면 이제 자습서를 이용하여 CD 파이프라인을 만들 수 있습니다.

1단계: Amazon EC2 Linux 인스턴스 생성

이 단계에서는 샘플 애플리케이션을 배포할 Amazon EC2 인스턴스를 생성합니다. 리소스를 생성하려 는 리전에서 인스턴스 역할을 아직 생성하지 않은 경우이 프로세스의 일부로 IAM에서 인스턴스 역할 을 생성합니다.

인스턴스 역할을 생성하려면

- 1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
- 2. 콘솔 대시보드에서 역할을 선택합니다.
- 3. 역할 생성을 선택합니다.
- 신뢰할 수 있는 유형의 엔터티 선택에서 AWS 서비스를 선택합니다. 사용 사례 선택에서 EC2를 선택합니다. Select your use case(사용 사례 선택) 아래에서 EC2를 선택합니다. 다음: 권한을 선 택합니다.
- 검색하여 AWSSystemsManagerDefaultEC2InstanceManagementRoleeployAction라는 정책을 선택합니다.
- 6. 검색하여 AmazonSSMManagedInstanceCore라는 정책을 선택합니다. 다음: 태그를 선택합니다. 다음: 태그를 선택합니다.
- 7. 다음: 검토를 선택합니다. 역할의 이름을 입력합니다(예: EC2InstanceRole).

Note

다음 단계를 위해 역할 이름을 적어 둡니다. 인스턴스를 생성할 때 이 역할을 선택합니다.

Note

파이프라인 생성 후 파이프라인의 S3 아티팩트 버킷에 대한 액세스를 허용하는 권한을이 역할에 추가합니다.

역할 생성을 선택합니다.

1단계: Amazon EC2 Linux 인스턴스 생성

인스턴스를 시작하려면

- 1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
- 2. 측면 탐색에서 인스턴스를 선택하고 페이지 상단에서 인스턴스 시작을 선택합니다.
- 3. 이름에 MyInstances를 입력합니다. 그러면 인스턴스에 Name의 태그 키와 MyInstances의 태 그 값이 할당됩니다.
- 4. 애플리케이션 및 OS 이미지(Amazon 머신 이미지)에서 AWS 로고가 있는 Amazon Linux AMI 옵 션을 찾아 선택했는지 확인합니다. (이 AMI는 Amazon Linux 2 AMI(HVM)로 기술되며 "프리 티어 가능"이라는 레이블이 지정되어 있습니다.)
- 5. 인스턴스 유형에서 인스턴스의 하드웨어 구성으로 사용할 프리 티어 가능 t2.micro 유형을 선택 합니다.
- 6. 키 페어(로그인)에서 키 페어를 선택하거나 새로 생성합니다.
- 7. 네트워크 설정에서 상태가 활성화인지 확인합니다.
- 8. Advanced details(고급 세부 정보)를 확장합니다. IAM 인스턴스 프로파일에서 이전 절차에서 생성 한 IAM 역할을 선택합니다(예: EC2InstanceRole).

Note

인스턴스 역할을 비워 두지 마세요. 이렇게 하면 기본 역할이 생성되고 생성한 역할을 선 택하지 않기 때문입니다.

- 9. 요약의 인스턴스 수에를 입력합니다2.
- 10. 인스턴스 시작을 선택합니다.
- 11. [Instances] 페이지에서 시작 상태를 볼 수 있습니다. 인스턴스를 시작할 때 초기 상태는 pending입니다. 인스턴스가 시작된 후에는 상태가 running으로 바뀌고 퍼블릭 DNS 이름을 받 습니다. [Public DNS] 열이 표시되지 않으면 [Show/Hide] 아이콘을 선택하고 [Public DNS]를 선택 합니다.

2단계: EC2 인스턴스 역할에 아티팩트 버킷 권한 추가

파이프라인의 아티팩트 버킷에 액세스할 수 있도록 인스턴스에 대해 생성한 EC2 인스턴스 역할을 업 데이트해야 합니다.

Note

인스턴스를 생성할 때 기존 EC2 인스턴스 역할을 생성하거나 사용합니다. Access Denied 오류를 방지하려면 인스턴스 역할에 S3 버킷 권한을 추가하여 인스턴스에 CodePipeline 아티 팩트 버킷에 대한 권한을 부여해야 합니다. 파이프라인 리전의 아티팩트 버킷으로 범위가 지정 된 s3:Get0bject 권한으로 기본 역할을 생성하거나 기존 역할을 업데이트합니다.

- CodePipeline 콘솔에서 파이프라인으로 이동합니다. 설정을 선택합니다. 기존 파이프라인에 대한 아티팩트 스토어의 이름과 위치를 확인합니다. 아티팩트 버킷 Amazon 리소스 이름(ARN)을 기록 해 복사합니다.
- IAM 콘솔로 이동하여 역할(Roles)을 선택합니다. 이 자습서의 1단계에서 생성한 인스턴스 역할을 선택합니다.
- 3. 권한 탭에서 인라인 정책 추가(Add inline policy)를 선택합니다.
- 4. 정책 문서에 다음 JSON을 추가하여 Resource 필드의 값을 버킷 ARN으로 바꿉니다.

```
{
    "Effect": "Allow",
    "Principal": "*",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::BucketName"
}
```

5. 업데이트를 선택합니다.

3단계: 리포지토리에 스크립트 파일 추가

이 샘플 텍스트를 붙여넣어 배포의 스크립트 후 단계를 위한 script.sh 파일을 생성합니다.

echo "Hello World!"

script.sh 파일을 해당 소스 리포지토리에 추가하려면

- 1. 텍스트 편집기를 연 다음 위의 파일을 복사하여 새 파일에 붙여 넣습니다.
- 2. script.sh 파일을 커밋한 후 소스 리포지토리에 푸시합니다.
 - a. 파일을 추가합니다.

git add .

b. 변경 내용을 커밋합니다.

git commit -m "Adding script.sh."

c. 커밋을 푸시합니다.

git push

리포지토리의 경로를 기록해 둡니다.

/MyDemoRepo/test/script.sh

4단계: 파이프라인 생성

CodePipeline 마법사를 사용하여 파이프라인 단계를 생성하고 소스 리포지토리를 연결합니다.

파이프라인을 생성하려면

- 1. https://console.aws.amazon.com/codepipeline/에서 CodePipeline 콘솔을 엽니다.
- [Welcome] 페이지, [Getting started] 페이지 또는 [Pipelines] 페이지에서 Create pipeline(파이프라 인 생성)을 선택합니다.
- 1단계: 생성 옵션 선택 페이지의 생성 옵션에서 사용자 지정 파이프라인 빌드 옵션을 선택합니다. Next(다음)를 선택합니다.
- 4. 2단계: 파이프라인 설정 선택의 파이프라인 이름에 MyPipeline을 입력합니다.
- CodePipeline은 특성과 가격이 다른 V1 및 V2 유형 파이프라인을 제공합니다. V2 유형은 콘 솔에서 선택할 수 있는 유일한 유형입니다. 자세한 내용은 <u>파이프라인 유형</u>을 참조하세요. CodePipeline에 대한 요금 정보는 <u>요금</u>을 참조하세요.
- 서비스 역할에서 기존 서비스 역할 사용을 선택한 다음이 작업에 필요한 권한으로 업데이트된 CodePipeline 서비스 역할을 선택합니다. 이 작업에 대한 CodePipeline 서비스 역할을 구성하려면 섹션을 참조하세요EC2 배포 작업에 대한 서비스 역할 정책 권한.
- 7. [Advanced settings]의 설정은 기본값 그대로 두고, [Next]를 선택합니다.
- 8. 3단계: 소스 스테이지 추가 페이지에서 소스 스테이지를 추가합니다.

- a. 소스 공급자에서 GitHub(GitHub 앱을 통해)를 선택합니다.
- b. 연결에서 기존 연결을 선택하거나 새로 생성합니다. GitHub 소스 작업에 대한 연결을 만들거 나 관리하려면 GitHub 연결을 참조하세요.
- c. 리포지토리 이름에서 GitHub 리포지토리의 이름을 선택합니다.

Next(다음)를 선택합니다.

- 9. 4단계: 빌드 단계 추가 페이지에서 건너뛰기를 선택합니다.
- 10. 5단계: 배포 단계 추가 페이지에서 EC2를 선택합니다.

Х

Instance type

Choose the instance type that you want to deploy to. Supported types are Amazon EC2 instances or AWS Systems Manager (SSM) managed nodes. You must have already created, tagged, and installed the SSM agent on all instances.

EC2

You can add one group of tags for EC2 instances to this deployment group.

One tag group: Any instance identified by the tag group will be deployed to.

Key	
Q Name	X

Q Name

Value

Q my-instances

Matching instances

2 unique matched instances.

Click here for details [2]

Target directory

Specify the location of the target directory you want to deploy to. Use an absolute path like /home/ec2-user/deploy.

/home/ec2-user/testhelloworld

PreScript - optional

Path to the executable script file that runs BEFORE the Deploy phase. It should start from the root directory of your uploaded source artifact. Use an absolute path like uploadDir/preScript.sh.

PostScript

Path to the executable script file that runs AFTER the Deploy phase. It should start from the root directory of your uploaded source artifact. Use an absolute path like uploadDir/postScript.sh.

test/script.sh

Advanced

Max batch - optional

Specify the number or percentage of targets that can deploy in parallel.

targets

O percentage

targets: from 1 to the number of instances you have

a. 대상 디렉터리에와 같이 배포하려는 인스턴스의 디렉터리를 입력합니다/home/ec2-user/ testhelloworld.

Note

인스턴스에서 작업을 사용할 배포 디렉터리를 지정합니다. 작업은 배포의 일부로 인 스턴스에 지정된 디렉터리를 자동으로 생성합니다.

- b. PostScript에와 같이 스크립트의 경로와 파일 이름을 입력합니다test/script.sh.
- c. Next(다음)를 선택합니다.
- 11. 6단계: 검토 페이지에서 파이프라인 구성을 검토하고 파이프라인 생성을 선택하여 파이프라인을 생성합니다.

Source Succeeded Succeeded Pipeline execution ID: <u>e4e931ec-56cb</u>		
Source GitHub (via GitHub App) [2] Succeeded - <u>42 minutes ago</u> <u>f8c490e7</u> View details		
f8c490e7 Source: Edited script.sh Disable transition		
DeploytoEC2 Succeeded Pipeline execution ID: e4e931ec-56cb	St	art rollback
EC2Deploy Amazon EC2 Succeeded - <u>38 minutes ago</u> View details		
f8c490e7 Source: Edited script.sh Disable transition		

12. 파이프라인이 성공적으로 실행되면 세부 정보 보기를 선택하여 작업에 대한 로그를 보고 관리형 컴퓨팅 작업 출력을 확인합니다.

Logs Summary Input Output
Showing the last 38 lines of the build log. <u>View entire log</u>
A Show previous logs
<pre>1 [2025/02/13 00:15:04.521] Describing tag, tag key = Name. tag value = my-instances. 2 [2025/02/13 00:15:04.784] Found 2 instances: i-0145; , i-05866 3 [2025/02/13 00:15:04.873] Processing deploy event BLOCK_TRAFFIC on instances: i- 0145;</pre>
4 [2025/02/13 00:15:04.891] Skipping deploy event BLOCK_TRAFFIC on instances: i-0145 as not specified in action configuration
5 [2025/02/13 00:15:04.918] Processing deploy event DOWNLOAD on instances: i-0145; 6 [2025/02/13 00:15:05.093] Executing commands on instances i-0145, SSM command id 9d57dace-, , commands: mkdir -p /tmp/codepipeline/53
7 aws s3api get-objectbucket codepipeline-us-east-1-2938 rbtest/SourceArti/rwfBtWb /tmp/codepipeline/530
8 unzip -o /tmp/codepipeline/530398 /tmp/codepipeline/5303
<pre>9 rm /tmp/codepipeline/5303985c-+c99-2 10 [2025/02/13 00:15:38.340] Deploy event DOWNLOAD succeeded on instances: i-0145 11 [2025/02/13 00:15:38.397] Processing deploy event BEFORE_DEPLOY on instances: i- 0145c</pre>
12 [2025/02/13 00:15:38.412] Skipping deploy event BEFORE_DEPLOY on instances: i- 0145: as not specified in action configuration
13 [2025/02/13 00:15:38.436] Processing deploy event DEPLOY on instances: i-0145 14 [2025/02/13 00:15:38.523] Executing commands on instances i-0145 , SSM command id
6C/d0e01-(), commands: mkdir -p /home/ec2-user/testhelloworld 15 cp -r /tmp/codepipeline/530 ;/* /home/ec2-user/testhelloworld 16 rm -rf /tmp/codepipeline/530 /*
17 [2025/02/13 00:16:13.616] Deploy event DEPLOY succeeded on instances: i-0145 7. 18 [2025/02/13 00:16:13.673] Processing deploy event AFTER_DEPLOY on instances: i- 0145 0145 0145 0145

Log	gs Summary Input Output
29	aws spape get-objectbucket codepiperine-us-east-i-29900999000key
30	unzip -o /tmp/codepipeline/530
	/tmp/codepipeline/5303
31	rm /tmp/codepipeline/53
32	[2025/02/13 00:17:17.891] Deploy event DOWNLOAD succeeded on instances: i-05866
33	[2025/02/13 00:1/:1/.942] Processing deploy event BEFORE_DEPLOY on instances: 1-
3/1	12800 [2825/82/13 88:17:17 953] Skinning denlow event REFORE DEPLOY on instances: i-
74	058663cf25cc55720 as not specified in action configuration
35	[2025/02/13 00:17:17.974] Processing deploy event DEPLOY on instances: i-05866
36	[2025/02/13 00:17:18.062] Executing commands on instances i-05866 , SSM command id
	d7abd80c- , commands: mkdir -p /home/ec2-user/testhelloworld
37	<pre>cp -r /tmp/codepipeline/JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ</pre>
38	rm -rf /tmp/codepipeline/
39	[2025/02/13 00:17:18.129] Total instances 2, pending instances 0, in progress instances 1.
40	[2025/02/13 00:17:49.738] Deploy event DEPLOY succeeded on instances: i-05866
41	[2025/02/13 00:1/:49./8/] Processing deploy event AFTER_DEPLOY on instances: 1-
42	[2025/02/13 00:17:49 880] Evecuting commands on instances i-058663
74	06365 . commands: chmod u+x /home/ec2-
	user/testhelloworld/test/script.sh
43	/home/ec2-user/testhelloworld/test/script.sh
44	[2025/02/13 00:17:49.938] Total instances 2, pending instances 0, in progress instances 1.
45	[2025/02/13 00:18:20.868] Deploy event AFTER_DEPLOY succeeded on instances: i-
	05866
46	[2025/02/13 00:18:20.921] Processing deploy event UNBLOCK_TRAFFIC on instances: i-
47	05866
47	2023/02/13 00.18:20.935] Skipping deploy event UNBLOCK_TRAFFIC on instances: 1-
48	$[2025/02/13 \ 00:18:21.075]$ Describing tag. tag key = Name. tag value = my-instances
49	[2025/02/13 00:18:21.322] Found 0 more instances:
50	[2025/02/13 00:18:21.415] Deployment SUCCEEDED
51	

5단계: 파이프라인 테스트

파이프라인에는 end-to-end네이티브 AWS 지속적 배포를 실행하기 위한 모든 것이 있어야 합니다. 이 제 소스 리포지토리에 코드 변경을 푸시하여 파이프라인 기능을 테스트해 보겠습니다.

파이프라인을 테스트하려면

- 1. 구성된 소스 리포지토리에 대한 코드를 변경하고 커밋한 후 변경 사항을 푸시합니다.
- 2. https://console.aws.amazon.com/codepipeline/에서 CodePipeline 콘솔을 엽니다.
- 3. 목록에서 파이프라인을 선택합니다.

- 단계를 수행하면서 파이프라인 진행 상황을 관찰합니다. 파이프라인이 완료되고 작업이 인스턴스 에 스크립트를 배포합니다.
- 5. 문제 해결에 대한 자세한 정보는 <u>오류 메시지와 함께 EC2 배포 작업이 실패함 No such file</u> 섹 션을 참조하세요.

자습서: CodePipeline을 사용하여 Docker 이미지를 빌드하고 Amazon ECR에 푸시(V2 유형)

이 자습서는 소스 코드를 변경한 후 Docker 이미지를 실행하고 Amazon ECR에 푸시하는 빌드 작업 을 CodePipeline에서 생성하는 데 도움이 됩니다. 또한이 자습서에서는 푸시된 이미지를 배포하는 Amazon ECS 배포 작업을 추가하는 방법을 보여줍니다.

🛕 Important

콘솔에서 파이프라인을 생성하는 과정에서 CodePipeline은 S3 아티팩트 버킷을 아티팩트에 사용합니다. (이는 S3 소스 작업에 사용되는 버킷과 다릅니다.) S3 아티팩트 버킷이 파이프라 인의 계정과 다른 계정에 있는 경우 S3 아티팩트 버킷을 안전하고 신뢰할 수 AWS 계정 있는에 서 소유해야 합니다.

Note

이 자습서는 GitHub 소스 리포지토리가 있는 CodePipeline 파이프라인에 대한 ECRBuildAndPublish 빌드 작업과 Amazon ECS 클러스터에 배포하기 위한 Amazon ECS 표준 작업을 위한 것입니다. ECR 이미지 리포지토리가 있는 파이프라인을 CodePipeline의 Amazon ECS to CodeDeploy 블루/그린 배포 작업의 소스로 사용하는 자습서는 섹션을 참조하 세요<u>자습서: Amazon ECR 소스 및 ECS-to-CodeDeploy 배포를 사용하여 파이프라인 생성</u>.

▲ Important

이 작업은 CodePipeline 관리형 CodeBuild 컴퓨팅을 사용하여 빌드 환경에서 명령을 실행합니

다. 명령 작업을 실행하면 AWS CodeBuild에서 별도의 요금이 발생합니다.

사전 조건

이 자습서를 이용하여 CD 파이프라인을 만들려면 먼저 몇 가지 리소스를 갖춰야 합니다. 다음은 시작 하기 위해 필요한 항목입니다.

Note

이러한 모든 리소스는 동일한 AWS 리전 내에 생성되어야 합니다.

- 이 자습서에 다음을 추가할 소스 제어 리포지토리(이 자습서에서는 GitHub 사용)입니다.
 - 1단계에서는 소스 리포지토리에 샘플 Dockerfile을 CodePipeline의 ECRBuildAndPublish 빌드 작 업에 대한 입력 아티팩트로 추가합니다.
 - 2단계에서는 CodePipeline의 Amazon ECS 표준 배포 작업에 대한 요구 사항으로 샘플 imagedefinitions.json 파일을 소스 리포지토리에 추가합니다.
- Dockerfile에서 빌드한 이미지가 포함된 Amazon ECR 이미지 리포지토리입니다. 자세한 내용은 Amazon Elastic Container Registry 사용 설명서의 리포지토리 생성 및 이미지 푸시를 참조하세요.
- 이미지 리포지토리와 동일한 리전에서 생성된 Amazon ECS 클러스터 및 서비스입니다. 자세한 내 용은 Amazon Elastic Container Service 개발자 안내서의 <u>클러스터 생성</u> 및 <u>서비스 생성</u>을 참조하세 요.

이러한 사전 조건을 모두 갖췄으면 이제 자습서를 이용하여 CD 파이프라인을 만들 수 있습니다.

1단계: 소스 리포지토리에 Dockerfile 추가

이 자습서에서는 ECRBuildAndPublish 작업을 사용하여 도커 이미지를 빌드하고 이미지를 Amazon ECR에 푸시합니다. CodePipeline의 관리형 컴퓨팅 작업은 CodeBuild를 사용하여 ECR 로그인 및 이미지 푸시에 대한 명령을 실행합니다. CodeBuild에 방법을 알려주기 위해 소스 코드 리포지토 리에 buildspec.yml 파일을 추가할 필요가 없습니다. 이 예제에서는 다음과 같이 리포지토리의 Dockerfile만 제공합니다.

이 샘플 텍스트를 붙여 넣어 Dockerfile 파일을 생성합니다. 이 샘플 Dockerfile은 사전 조건의 ECR 이미지 지침에 사용된 샘플과 동일합니다.

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

```
# Install dependencies
RUN yum update -y && \
yum install -y httpd
# Install apache and write hello world message
RUN echo 'Hello World!' > /var/www/html/index.html
# Configure apache
RUN echo 'mkdir -p /var/run/httpd' >> /root/run_apache.sh && \
echo 'mkdir -p /var/lock/httpd' >> /root/run_apache.sh && \
echo '/usr/sbin/httpd -D FOREGROUND' >> /root/run_apache.sh && \
chmod 755 /root/run_apache.sh
EXPOSE 80
```

CMD /root/run_apache.sh

Dockerfile 파일을 해당 소스 리포지토리에 추가하려면

- 1. 텍스트 편집기를 연 다음 위의 Dockerfile을 복사하여 새 파일에 붙여 넣습니다.
- 2. Dockerfile 파일을 커밋한 후 소스 리포지토리에 푸시합니다.
 - a. 파일을 추가합니다.

git add .

b. 변경 내용을 커밋합니다.

git commit -m "Adding Dockerfile."

c. 커밋을 푸시합니다.

git push

파일을 리포지토리의 루트 수준에 배치해야 합니다.

/ Dockerfile

2단계: 소스 리포지토리에 imagedefinitions.json 파일 추가

이 자습서에서는 CodePipelinetheAmazon ECS 표준 배포 작업을 사용하여 컨테이너를 Amazon ECS 클러스터에 배포합니다. Amazon ECS 표준 배포 작업에는 이미지 이름과 URI가 포함된 imagedefinitions.json 파일이 필요합니다. imagedefinitions.json 파일에 대한 자세한 내용은 섹션을 참 조하세요Amazon ECS 표준 배포 작업을 위한 imagedefinitions.json 파일.

이 샘플 텍스트를 붙여 넣어 imagedefinitions.json 파일을 생성합니다. hello-world와 같은 Dockerfile의 이름을 사용하고 이미지가 저장된 Amazon ECR 리포지토리의 URI를 사용합니다.



소스 리포지토리에 imagedefinitions.json 파일을 추가하려면

- 1. 텍스트 편집기를 연 다음 위 예제를 복사하여 새 파일에 붙여 넣습니다.
- 2. imagedefinitions.json 파일을 커밋한 후 소스 리포지토리에 푸시합니다.
 - a. 파일을 추가합니다.

git add .

b. 변경 내용을 커밋합니다.

git commit -m "Adding imagedefinitions.json."

c. 커밋을 푸시합니다.

git push

파일을 리포지토리의 루트 수준에 배치해야 합니다.

/ imagedefinitions.json
3단계: 파이프라인 생성

CodePipeline 마법사를 사용하여 파이프라인 단계를 생성하고 소스 리포지토리를 연결합니다.

파이프라인을 생성하려면

- 1. https://console.aws.amazon.com/codepipeline/에서 CodePipeline 콘솔을 엽니다.
- [Welcome] 페이지, [Getting started] 페이지 또는 [Pipelines] 페이지에서 Create pipeline(파이프라 인 생성)을 선택합니다.
- 1단계: 생성 옵션 선택 페이지의 생성 옵션에서 사용자 지정 파이프라인 빌드 옵션을 선택합니다.
 다음을 선택합니다.
- 4. 2단계: 파이프라인 설정 선택의 파이프라인 이름에 MyPipeline을 입력합니다.
- CodePipeline은 특성과 가격이 다른 V1 및 V2 유형 파이프라인을 제공합니다. V2 유형은 콘 솔에서 선택할 수 있는 유일한 유형입니다. 자세한 내용은 <u>파이프라인 유형</u>을 참조하세요. CodePipeline에 대한 요금 정보는 요금을 참조하세요.
- 서비스 역할에서 새 서비스 역할을 선택하여CodePipeline이 IAM에 서비스 역할을 생성하도록 허용합니다.
- 7. [Advanced settings]의 설정은 기본값 그대로 두고, [Next]를 선택합니다.
- 8. 3단계: 소스 스테이지 추가 페이지에서 소스 스테이지를 추가합니다.
 - a. 소스 공급자에서 GitHub(GitHub 앱을 통해)를 선택합니다.
 - b. 연결에서 기존 연결을 선택하거나 새로 생성합니다. GitHub 소스 작업에 대한 연결을 만들거 나 관리하려면 <u>GitHub 연결</u>을 참조하세요.
 - c. 리포지토리 이름에서 GitHub 리포지토리의 이름을 선택합니다.
 - d. 기본 브랜치에서 파이프라인을 수동으로 시작하거나 Git 태그가 아닌 소스 이벤트로 시작할
 때 지정할 브랜치를 선택합니다. 변경 소스가 트리거가 아니거나 파이프라인 실행이 수동으로 시작된 경우 기본 브랜치의 HEAD 커밋이 사용됩니다.

다음을 선택합니다.

9. 4단계: 빌드 단계 추가 페이지에서 다른 빌드 공급자를 선택하고 ECRBuildAndPublish를 선택합니다.

2p 1 Noose creation option	Add build stage Info Step 4 of 6
ep 2 noose pipeline settings	Build - optional
ep 3 dd source stage	Build provider Choose the tool you want to use to run build commands and specify artifacts for your build action.
tep 4 dd build stage	Commands Other build providers
tep 5	AWS ECRBuildAndPublish
ep 6	ECR Repository Name Enter a name for the ECR repository
eview	Q actions/image-repo X
	Image Tag - <i>optional</i> Enter an image tag
	Dockerfile Path - optional Enter the path to the Dockerfile
	Region
	US East (N. Virginia)
	Input artifacts Choose an input artifact for this action. Learn more

- a. ECR 리포지토리 이름에서 이미지 리포지토리를 선택합니다.
- b. 다음을 선택합니다.
- 10. 5단계: 테스트 단계 추가에서 테스트 단계 건너뛰기를 선택한 다음 건너뛰기를 다시 선택하여 경 고 메시지를 수락합니다.

다음을 선택합니다.

- 11. 6단계: 배포 단계 추가 페이지에서 배포 단계 건너뛰기를 선택합니다. 다음 단계에서 ECS 작업을 추가합니다.
- 12. 7단계: 검토 페이지에서 파이프라인 구성을 검토하고 파이프라인 생성을 선택하여 파이프라인을 생성합니다.
- 13. 파이프라인을 편집하여 파이프라인에 Amazon ECS 배포 작업을 추가합니다.
 - a. 오른쪽 위에서 편집을 선택합니다.

- b. 다이어그램의 하단에서 + 단계 추가를 선택합니다. Stage name(스테이지 이름)에서 이름(예: **Deploy**)을 입력합니다.
- c. + Add action group(작업 그룹 추가)을 선택합니다.
- d. 작업 이름에 이름을 입력하세요.
- e. 작업 공급자에서 Amazon ECS를 선택합니다. 리전이 파이프라인 리전으로 기본 설정되도록 합니다.
- f. 입력 아티팩트에서와 같은 소스 단계에서 입력 아티팩트를 선택합니다SourceArtifact.
- g. 클러스터 이름에서 해당 서비스가 실행 중인 Amazon ECS 클러스터를 선택합니다.
- h. 서비스 이름에서 업데이트할 서비스를 선택합니다.
- i. 저장을 선택합니다.
- j. 편집 중인 스테이지에서 완료를 선택합니다. AWS CodePipeline 창에서 저장을 선택한 다음 경고 메시지에서 저장을 선택합니다.
- k. 변경 사항을 제출하고 파이프라인 빌드를 시작하려면 변경 사항 배포를 선택한 다음 릴리
 스를 선택하세요.
- 14. 파이프라인이 실행된 후 파이프라인 구조 및 상태를 확인합니다.

Succeeded Succeeded	Start rollback	
Pipeline execution ID: <u>186696f4-0925-4</u>		
Build AWS ECRBuildAndPublish Succeeded - <u>5 minutes ago</u> View details)))
3524bc39 Source: Added imagedefinitions.json		
Disable transition		
⊘ Deploy Succeeded	Start rollback	
Pipeline execution ID: 186696f4-0925-4		
DeploytoECS		
View details		
3524bc39 Source: Added imagedefinitions.json		

15. 파이프라인이 성공적으로 실행되면 세부 정보 보기를 선택하여 작업에 대한 로그를 보고 관리형 컴퓨팅 작업 출력을 확인합니다.



16. 실패한 작업을 모두 해결합니다. 예를 들어, imagedefinitions.json 파일이 소스 리포지토리에 없는 경우 ECS 배포 작업이 실패할 수 있습니다. 다음은 imagedefinitions.json 파일이 누락된 경우 표시 되는 오류 메시지의 예입니다.

Developer Tools > CodePipeline > Pipelines > ecrbuild-to-ecs > Debug ecrbuild-to-ecs				
ecrbuild-to-ecs		Back to pipeline Release change		
√ ⊙ Source	 Pipeline execution details 			
Source GitHub (via GitHub App)	DeploytoECS Action execution ID: 68e3ec5a-5f06-4	-		
 ➢ Build AWS ECRBuildAndPublish ✓ ⊗ Deploy 	Summary Input			
BeploytoECS Amazon ECS	Status Failed	Last updated Just now		
	Action execution ID			
	Error code Invalid action configuration			
Error message Did not find the image definition file imagedefinitions.json in the input artifacts ZIP file. Ve file is stored in your pipeline's Amazon S3 artifact bucket: codepipeline-us-east-1- key: ecrbuild-to-ecs/SourceArti/				

4단계: 파이프라인 테스트

파이프라인에는 end-to-end네이티브 AWS 지속적 배포를 실행하기 위한 모든 것이 있어야 합니다. 이 제 소스 리포지토리에 코드 변경을 푸시하여 파이프라인 기능을 테스트해 보겠습니다.

파이프라인을 테스트하려면

- 1. 구성된 소스 리포지토리에 대한 코드를 변경하고 커밋한 후 변경 사항을 푸시합니다.
- 2. https://console.aws.amazon.com/codepipeline/에서 CodePipeline 콘솔을 엽니다.
- 3. 목록에서 파이프라인을 선택합니다.
- 단계를 수행하면서 파이프라인 진행 상황을 관찰합니다. 파이프라인이 완료되고 작업이 코드 변 경에서 생성된 ECR로 Docker 이미지를 푸시합니다.

자습서: CodePipeline을 사용하여 Amazon EKS에 배포

이 자습서는 CodePipeline에서 Amazon EKS에서 구성한 클러스터에 코드를 배포하는 배포 작업을 생 성하는 데 도움이 됩니다.

EKS 작업은 퍼블릭 및 프라이빗 EKS 클러스터를 모두 지원합니다. 프라이빗 클러스터는 EKS에서 권 장하는 유형이지만 두 유형 모두 지원됩니다.

Note

콘솔에서 파이프라인을 생성하는 과정에서 CodePipeline은 S3 아티팩트 버킷을 아티팩트에 사용합니다. (이는 S3 소스 작업에 사용되는 버킷과 다릅니다.) S3 아티팩트 버킷이 파이프라 인의 계정과 다른 계정에 있는 경우 S3 아티팩트 버킷을 안전하고 신뢰할 수 AWS 계정 있는에 서 소유해야 합니다.

Note

이 작업은 CodePipeline 관리형 CodeBuild 컴퓨팅을 사용하여 빌드 환경에서 명령을 실행합니 다. 명령 작업을 실행하면 AWS CodeBuild에서 별도의 요금이 발생합니다.

Note

EKS 배포 작업은 V2 유형 파이프라인에서만 사용할 수 있습니다.

사전 조건

이 자습서를 이용하여 CD 파이프라인을 만들려면 먼저 몇 가지 리소스를 갖춰야 합니다. 다음은 시작 하기 위해 필요한 항목입니다.

Note

이러한 모든 리소스는 동일한 AWS 리전 내에 생성되어야 합니다.

• 샘플 deployment.yaml 파일을 추가할 소스 제어 리포지토리(이 자습서에서는 GitHub 사용).

- <u>3단계: IAM에서 CodePipeline 서비스 역할 정책 업데이트</u> 아래를 사용하여이 작업에 대한 권한으로 업데이트할 기존 CodePipeline 서비스 역할을 사용해야 합니다. 필요한 권한은 생성하는 클러스터 유형에 따라 다릅니다. 자세한 내용은 서비스 역할 정책 권한 단원을 참조하십시오.
- ECR 또는 이미지 리포지토리로 푸시한 작업 이미지 및 리포지토리 태그입니다.

이러한 사전 조건을 모두 갖췄으면 이제 자습서를 이용하여 CD 파이프라인을 만들 수 있습니다.

1단계: (선택 사항) Amazon EKS에서 클러스터 생성

퍼블릭 또는 프라이빗 엔드포인트를 사용하여 EKS 클러스터를 생성하도록 선택할 수 있습니다.

다음 단계에서는 EKS에서 퍼블릭 또는 프라이빗 클러스터를 생성합니다. 클러스터를 이미 생성한 경 우이 단계는 선택 사항입니다.

Amazon EKS에서 퍼블릭 클러스터 생성

이 단계에서는 EKS에서 클러스터를 생성합니다.

퍼블릭 클러스터 생성

- 1. EKS 콘솔을 열고 클러스터 생성을 선택합니다.
- 2. 이름에서 클러스터의 이름을 지정합니다. [Next]를 선택합니다.
- 3. Create를 선택합니다.

Amazon EKS에서 프라이빗 클러스터 생성

프라이빗 엔드포인트가 있는 클러스터를 생성하기로 선택한 경우 프라이빗 서브넷만 연결하고 인터넷 에 연결되어 있는지 확인합니다.

프라이빗 엔드포인트가 있는 클러스터를 생성하려면 다음 5개의 하위 단계를 따릅니다.

콘솔에서 VPC 생성

- 1. VPC 콘솔을 열고 VPC 생성을 선택합니다.
- 2. VPC 설정에서 VPC 등을 선택합니다.
- 3. 퍼블릭 서브넷 1개와 프라이빗 서브넷 4개를 생성하도록 선택합니다. VPC 생성을 선택합니다.
- 4. 서브넷 페이지에서 프라이빗을 선택합니다.

VPC의 프라이빗 서브넷 확인

- 1. VPC로 이동하여 VPC ID를 선택하여 VPC 세부 정보 페이지를 엽니다.
- 2. VPC 세부 정보 페이지에서 리소스 맵 탭을 선택합니다.
- 다이어그램을 보고 프라이빗 서브넷을 기록해 둡니다. 서브넷은 퍼블릭 또는 프라이빗 상태를 나 타내는 레이블과 함께 표시되며 각 서브넷은 라우팅 테이블에 매핑됩니다.

VPC Show details	Subnets (6)	Route tables (6)
Your AWS virtual network	Subnets within this VPC	Route network traffic to resources
project-vpc	us-east-1a	project-rtb-private2-us-east-1b
	A project-subnet-public1-us-east-1a	project-rtb-private4-us-east-1b
	project-subnet-private3-us-east-1a	project-rtb-private1-us-east-1a
	project-subnet-private1-us-east-1a	rtb-0a3d97c04
	us-east-1b	project-rtb-private3-us-east-1a ::
	B project-subnet-public2-us-east-1b	project-rtb-public
	B project-subnet-private2-us-east-1b	
	B project-subnet-private4-us-east-1b	

프라이빗 클러스터에는 모든 프라이빗 서브넷이 있습니다.

 퍼블릭 서브넷을 생성하여 NAT 게이트웨이를 호스팅합니다. 한 번에 하나의 인터넷 게이트웨이 만 하나의 VPC에 연결할 수 있습니다.

퍼블릭 서브넷에서 NAT 게이트웨이 생성

- 퍼블릭 서브넷에서 NAT 게이트웨이를 생성합니다. VPC 콘솔로 이동한 다음 인터넷 게이트웨이 를 선택합니다. 인터넷 게이트웨이 생성을 선택합니다.
- 2. 이름에 인터넷 게이트웨이의 이름을 입력합니다. 인터넷 게이트웨이 생성을 선택합니다.

트래픽을 NAT 게이트웨이로 전달하도록 프라이빗 서브넷의 라우팅 테이블을 업데이트합니다.

프라이빗 서브넷의 라우팅 테이블에 NAT 게이트웨이 추가

- 1. VPC 콘솔로 이동한 다음 서브넷을 선택합니다.
- 각 프라이빗 서브넷에 대해 해당 서브넷을 선택한 다음 세부 정보 페이지에서 해당 서브넷의 라우 팅 테이블을 선택하고 라우팅 테이블 편집을 선택합니다.

- 인터넷 트래픽을 NAT 게이트웨이로 전달하도록 프라이빗 서브넷의 라우팅 테이블을 업데이트합 니다. 라우팅 추가를 선택합니다. 추가할 옵션에서 NAT 게이트웨이를 선택합니다. 생성한 인터넷 게이트웨이를 선택합니다.
- 4. 퍼블릭 서브넷의 경우 사용자 지정 라우팅 테이블을 생성합니다. 퍼블릭 서브넷의 네트워크 액세 스 제어 목록(ACL)이 프라이빗 서브넷의 인바운드 트래픽을 허용하는지 확인합니다.
- 5. Save changes(변경 사항 저장)를 선택합니다.

이 단계에서는 EKS에서 클러스터를 생성합니다.

프라이빗 클러스터 생성

- 1. EKS 콘솔을 열고 클러스터 생성을 선택합니다.
- 2. 이름에서 클러스터의 이름을 지정합니다. Next(다음)를 선택합니다.
- 3. VPC 및 기타 구성 정보를 지정합니다. 생성(Create)을 선택합니다.

EKS 클러스터는 퍼블릭 또는 프라이빗 클러스터일 수 있습니다. 이 단계는 프라이빗 엔드포인트만 있 는 클러스터를 위한 것입니다. 클러스터가 프라이빗인지 확인합니다.

2단계: Amazon EKS에서 프라이빗 클러스터 구성

이 단계는 프라이빗 클러스터를 생성한 경우에만 적용됩니다. 이 단계는 프라이빗 엔드포인트만 있는 클러스터를 위한 것입니다.

클러스터 구성

- 네트워킹 탭 아래의 EKS 클러스터에서만 프라이빗 서브넷을 연결합니다. 의 VPC에서 프라이빗 서브넷 확인 섹션에 캡처된 프라이빗 서브넷을 연결합니다<u>1단계: (선택 사항) Amazon EKS에서</u> 클러스터 생성.
- CodePipeline은 파이프라인의 S3 아티팩트 버킷에서 아티팩트를 저장하고 검색하므로 프라이빗 서브넷이 인터넷에 액세스할 수 있는지 확인합니다.

3단계: IAM에서 CodePipeline 서비스 역할 정책 업데이트

이 단계에서는 클러스터에 연결하는 데 CodePipeline에 필요한 권한**cp-service-role**으로와 같은 기존 CodePipeline 서비스 역할을 업데이트합니다. 기존 역할이 없는 경우 새 역할을 생성합니다.

다음 단계로 CodePipeline 서비스 역할을 업데이트합니다.

CodePipeline 서비스 역할 정책을 업데이트하려면

- 1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
- 2. 콘솔 대시보드에서 역할을 선택합니다.
- 3. 와 같은 CodePipeline 서비스 역할을 조회합니다cp-service-role.
- 4. 새 인라인 정책을 추가합니다.
- 5. 정책 편집기에서 다음을 입력합니다.
 - 퍼블릭 클러스터의 경우 다음 권한을 추가합니다.

```
{
    "Statement": [
        {
            "Sid": "EksClusterPolicy",
            "Effect": "Allow",
            "Action": "eks:DescribeCluster",
            "Resource": "arn:aws:eks:us-east-1:ACCOUNT-ID:cluster/my-cluster"
        },
        {
            "Sid": "EksVpcClusterPolicy",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeVpcs"
            ],
            "Resource": [
                "*"
            1
        }
    ],
    "Version": "2012-10-17"
}
```

• 프라이빗 클러스터의 경우 다음 권한을 추가합니다. 프라이빗 클러스터에는 해당하는 경우 VPC에 대한 추가 권한이 필요합니다.

{

```
"Statement": [
        {
            "Sid": "EksClusterPolicy",
            "Effect": "Allow",
            "Action": "eks:DescribeCluster",
            "Resource": "arn:aws:eks:us-east-1:ACCOUNT-ID:cluster/my-cluster"
        },
        {
            "Sid": "EksVpcClusterPolicy",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeVpcs"
            ],
            "Resource": [
                "*"
            1
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateNetworkInterface",
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                     "ec2:Subnet": [
                        "arn:aws:ec2:us-east-1:ACCOUNT-ID:subnet/
subnet-03ebd65daeEXAMPLE",
                        "arn:aws:ec2:us-east-1:ACCOUNT-ID:subnet/
subnet-0e377f6036EXAMPLE",
                         "arn:aws:ec2:us-east-1:ACCOUNT-ID:subnet/
subnet-0db658ba1cEXAMPLE",
                        "arn:aws:ec2:us-east-1:ACCOUNT-ID:subnet/
subnet-0db658ba1cEXAMPLE"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateNetworkInterfacePermission",
```

```
"Resource": "*",
            "Condition": {
                "ArnEquals": {
                    "ec2:Subnet": [
                        "arn:aws:ec2:us-east-1:ACCOUNT-ID:subnet/
subnet-03ebd65daeEXAMPLE",
                        "arn:aws:ec2:us-east-1:ACCOUNT-ID:subnet/
subnet-0e377f6036EXAMPLE",
                        "arn:aws:ec2:us-east-1:ACCOUNT-ID:subnet/
subnet-0db658ba1cEXAMPLE",
                        "arn:aws:ec2:us-east-1:ACCOUNT-ID:subnet/
subnet-0db658ba1cEXAMPLE"
                    1
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "ec2:DeleteNetworkInterface",
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                     "ec2:Subnet": [
                        "arn:aws:ec2:us-east-1:ACCOUNT-ID:subnet/
subnet-03ebd65daeEXAMPLE",
                        "arn:aws:ec2:us-east-1:ACCOUNT-ID:subnet/
subnet-0e377f6036EXAMPLE",
                        "arn:aws:ec2:us-east-1:ACCOUNT-ID:subnet/
subnet-0db658ba1cEXAMPLE",
                        "arn:aws:ec2:us-east-1:ACCOUNT-ID:subnet/
subnet-0db658ba1cEXAMPLE"
                    1
                }
            }
        }
    ],
    "Version": "2012-10-17"
}
```

```
6. 정책 업데이트를 선택합니다.
```

4단계: CodePipeline 서비스 역할에 대한 액세스 항목 생성

이 단계에서는 클러스터에 액세스 항목을 생성하여 관리형 액세스 정책과 함께 3단계에서 업데이트한 CodePipeline 서비스 역할을 추가합니다.

- 1. EKS 콘솔을 열고 클러스터로 이동합니다.
- 2. 액세스 탭을 선택합니다.
- 3. IAM 액세스 항목에서 액세스 항목 생성을 선택합니다.
- 4. IAM 보안 주체 ARN에와 같이 작업에 대해 방금 업데이트한 역할을 입력합니다**cp-servicerole**. Next(다음)를 선택합니다.
- 5. 2단계: 액세스 정책 추가 페이지의 정책 이름에서와 같은 액세스에 대한 관리형 정책을 선택합니 다AmazonEKSClusterAdminPolicy. 정책 추가를 선택합니다. Next(다음)를 선택합니다.

1 Note

이는 CodePipeline 작업이 Kubernetes와 대화하는 데 사용하는 정책입니다. 가장 좋은 방 법은 관리 정책이 아닌 최소 권한으로 정책의 권한을 축소하려면 대신 사용자 지정 정책을 연결하는 것입니다.

6. 검토 페이지에서 생성을 선택합니다.

5단계: 소스 helm chart 리포지토리 생성 및 구성 파일 추가

이 단계에서는 작업에 적합한 구성 파일(Kubernetes 매니페스트 파일 또는 차트 Helm)을 생성하고 소 스 리포지토리에 구성 파일을 저장합니다. 구성에 적합한 파일을 사용합니다. 자세한 내용은 <u>https://</u> <u>kubernetes.io/docs/reference/kubectl/quick-reference/</u> 또는 <u>https://helm.sh/docs/topics/charts/</u> 참조하 십시오.

- Kubernetes의 경우 매니페스트 파일을 사용합니다.
- Helm의 경우 Helm 차트를 사용합니다.
- 1. 기존 GitHub 리포지토리를 생성하거나 사용합니다.
- 2. 아래 예제와 같이 helm 차트 파일에 대한 새 구조를 리포지토리에 생성합니다.

mychart |-- Chart.yaml

	charts			
	templates			
I	NOTES.txt			
I	<pre> helpers.tpl</pre>			
I	<pre> deployment.yaml</pre>			
I	ingress.yaml			
I	` service.yaml			
`	values.yaml			

3. 파일을 리포지토리의 루트 수준에 추가합니다.

6단계: 파이프라인 생성

CodePipeline 마법사를 사용하여 파이프라인 단계를 생성하고 소스 리포지토리를 연결합니다.

파이프라인을 생성하려면

- 1. <u>https://console.aws.amazon.com/codepipeline/에서 CodePipeline 콘솔을 엽니다.</u>
- [Welcome] 페이지, [Getting started] 페이지 또는 [Pipelines] 페이지에서 Create pipeline(파이프라 인 생성)을 선택합니다.
- 3. 1단계: 생성 옵션 선택 페이지의 생성 옵션에서 사용자 지정 파이프라인 빌드 옵션을 선택합니다. Next(다음)를 선택합니다.
- 4. 2단계: 파이프라인 설정 선택의 파이프라인 이름에 MyEKSPipeline을 입력합니다.
- CodePipeline은 특성과 가격이 다른 V1 및 V2 유형 파이프라인을 제공합니다. V2 유형은 콘 솔에서 선택할 수 있는 유일한 유형입니다. 자세한 내용은 <u>파이프라인 유형</u>을 참조하세요. CodePipeline에 대한 요금 정보는 <u>요금</u>을 참조하세요.
- 6. 서비스 역할에서 3단계에서 업데이트한 서비스 역할을 선택합니다.
- 7. [Advanced settings]의 설정은 기본값 그대로 두고, [Next]를 선택합니다.
- 3단계: 소스 단계 추가 페이지의 소스 공급자에서 를 선택하여 GitHub 리포지토리에 대한 연결을 생성합니다.
- 9. 4단계: 빌드 단계 추가 페이지에서 건너뛰기를 선택합니다.
- 10. 5단계: 배포 단계 추가 페이지에서 Amazon EKS를 선택합니다.

Deploy configuration type

Please select deploy configuration type.

Helm
 Helm configuration type

O Kubectl

Kubectl configuration type

Helm release name

Enter the helm release name.

my-release

Helm chart location

Enter folder location of helm chart.

nginx-chart

Override for helm values files - optional

Enter comma-separated helm values files in helm chart location.

Kubernetes namespace - optional

You can provide a name for the Kubernetes namespace to override the default.

Subnet IDs

Specify the subnet IDs that your compute action will use.

▼

- a. 배포 구성 유형에서 Helm을 선택합니다.
- b. 차트 Helm 위치에와 같은 릴리스 이름을 입력합니다my-release. 차트 Helm 위치에와 같은 차트 Helm 파일의 경로를 입력합니다mychart.
- c. Next(다음)를 선택합니다.
- 11. 6단계: 검토 페이지에서 파이프라인 구성을 검토하고 파이프라인 생성을 선택하여 파이프라인을 생성합니다.

Pipeline execution ID: 0483d26d-5000-4	4
-	
Source	
Succeeded - <u>3 minutes ago</u>	
<u>11412f6c</u>	
View details	
11412f6c Source: Update helm chart values.yaml	
Disable transition	
() Doploy	
Deploy	
Deploy Amazon EKS 🖸	
Amazon EKS 2	

12. 파이프라인이 성공적으로 실행되면 세부 정보 보기를 선택하여 작업의 로그를 보고 작업 출력을 확인합니다.

자습서: 컴퓨팅을 사용하여 명령을 실행하는 파이프라인 생성(V2 유 형)

이 자습서에서는 빌드 단계에서 명령 작업을 사용하여 제공된 빌드 명령을 지속적으로 실행하는 파이 프라인을 구성합니다. 명령 작업에 대한 자세한 내용은 명령 작업 참조 섹션을 참조하세요.

A Important

파이프라인 생성의 일환으로 CodePipeline은 고객이 제공하는 S3 아티팩트 버킷을 아티팩트 에 사용합니다. (이는 S3 소스 작업에 사용되는 버킷과 다릅니다.) S3 아티팩트 버킷이 파이프 라인의 계정과 다른 계정에 있는 경우 S3 아티팩트 버킷을 안전하고 신뢰할 수 AWS 계정 있는 에서 소유해야 합니다.

사전 조건

다음 항목이 있어야 합니다.

• GitHub 리포지토리. <u>자습서: GitHub 파이프라인 소스와 함께 전체 복제 사용</u>에 생성한 GitHub 리포 지토리를 사용할 수 있습니다.

1단계: 소스 파일 생성 및 GitHub 리포지토리에 푸시

이 섹션에서는 파이프라인이 소스 스테이지에 대해 사용하는 리포지토리에 예제 소스 파일을 생성하 고 푸시합니다. 이 예제에서는 다음을 생성하고 푸시합니다.

• README.txt 파일.

소스 파일을 생성하려면

1. 다음 텍스트를 사용하여 파일을 생성합니다.

Sample readme file

2. 파일을 README.txt(으)로 저장합니다.

GitHub 리포지토리에 파일을 푸시하려면

 리포지토리에 파일을 푸시하거나 업로드합니다. 이러한 파일은 파이프라인 생성 마법사가 AWS CodePipeline에서 배포 작업을 위해 생성한 소스 아티팩트입니다. 파일은 로컬 디렉터리에 다음과 같이 나타납니다.

README.txt

- 2. 로컬 컴퓨터의 복제된 리포지토리에서 Git 명령줄을 사용하려면 다음과 같이 합니다.
 - a. 다음 명령을 실행하여 모든 파일을 한 번에 스테이징합니다.

git add -A

b. 다음 명령을 실행하여 커밋 메시지와 함께 파일을 커밋합니다.

git commit -m "Added source files"

c. 다음 명령을 실행하여 로컬 리포지토리의 파일을 리포지토리에 푸시합니다.

git push

2단계: 파이프라인 생성

이 단원에서는 다음 작업을 통해 파이프라인을 생성합니다.

- 소스 파일이 저장되는 리포지토리에 대한 GitHub(GitHub App을 통해) 작업이 있는 소스 단계입니다.
- 명령 작업이 있는 빌드 스테이지입니다.

마법사를 사용하여 파이프라인을 생성하려면

- 1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https//https//http
- [Welcome] 페이지, [Getting started] 페이지 또는 [Pipelines] 페이지에서 Create pipeline(파이프라 인 생성)을 선택합니다.
- 3. 1단계: 생성 옵션 선택 페이지의 생성 옵션에서 사용자 지정 파이프라인 빌드 옵션을 선택합니다. Next(다음)를 선택합니다.
- 4. 2단계: 파이프라인 설정 선택의 파이프라인 이름에 MyCommandsPipeline을 입력합니다.
- CodePipeline은 특성과 가격이 다른 V1 및 V2 유형 파이프라인을 제공합니다. V2 유형은 콘 솔에서 선택할 수 있는 유일한 유형입니다. 자세한 내용은 <u>파이프라인 유형</u>을 참조하세요. CodePipeline에 대한 요금 정보는 요금을 참조하세요.
- 서비스 역할에서 새 서비스 역할을 선택하여CodePipeline이 IAM에 서비스 역할을 생성하도록 허용합니다.

Note

기존 서비스 역할을 사용하는 경우 명령 작업을 사용하려면 서비스 역할에 대해 다음 권한 을 추가해야 합니다. 서비스 역할 정책 문의 리소스 기반 권한을 사용하여 파이프라인 리 소스 수준으로 권한 범위를 좁힙니다. 자세한 내용은 <u>서비스 역할 정책 권한</u>에서 정책 예 제를 참조하세요.

- logs:CreateLogGroup
- logs:CreateLogStream

- logs:PutLogEvents
- 7. [Advanced settings]의 설정은 기본값 그대로 두고, [Next]를 선택합니다.
- 8. 3단계: 소스 스테이지 추가 페이지에서 소스 스테이지를 추가합니다.
 - a. 소스 공급자에서 GitHub(GitHub 앱을 통해)를 선택합니다.
 - b. 연결에서 기존 연결을 선택하거나 새로 생성합니다. GitHub 소스 작업에 대한 연결을 만들거 나 관리하려면 GitHub 연결을 참조하세요.
 - c. 리포지토리 이름에서 GitHub.com 리포지토리의 이름을 선택합니다.
 - d. 기본 브랜치에서 파이프라인을 수동으로 시작하거나 Git 태그가 아닌 소스 이벤트로 시작할 때 지정할 브랜치를 선택합니다. 변경 소스가 트리거가 아니거나 파이프라인 실행이 수동으 로 시작된 경우 기본 브랜치의 HEAD 커밋이 사용됩니다. 선택적으로 필터링(트리거)을 사용 하여 웹후크를 지정할 수도 있습니다. 자세한 내용은 <u>트리거 및 필터링을 사용하여 시작 파이</u> 프라인 자동화 단원을 참조하십시오.

Next(다음)를 선택합니다.

9. 4단계: 빌드 스테이지 추가에서 명령을 선택합니다.

Note

명령 작업을 실행하면 AWS CodeBuild에서 별도의 요금이 발생합니다.

다음 명령을 입력합니다.

ls
echo hello world
cat README.txt
echo pipeline Execution Id is #{codepipeline.PipelineExecutionId}

```
Next(다음)를 선택합니다.
```

Add source stage	Build - optional				
Step 3					
Add build stage	Build provider Choose the tool you want to use to run build commands and specify artifacts for your build action.				
Step 4 Add deploy stage	Commands Other build providers				
Step 5 Review	Commands Specify the shell commands to run with your compute action in CodePipeline. You do not need to create any resources in CodeBuild. Note: Using compute time for this action will incur separate charges in AWS CodeBuild.				
	ls echo hello world echo pipeline Execution Id is #{codepipeline.PipelineExecutionId}				
	Input artifacts Choose an input artifact for this action. Learn more 🔀				
	SourceArtifact X Defined by: Source No more than 100 characters				

10. 5단계: 테스트 단계 추가에서 테스트 단계 건너뛰기를 선택한 다음 건너뛰기를 다시 선택하여 경 고 메시지를 수락합니다.

Next(다음)를 선택합니다.

11. 6단계: 배포 단계 추가에서 배포 단계 건너뛰기를 선택한 다음 건너뛰기를 다시 선택하여 경고 메 시지를 수락합니다.

Next(다음)를 선택합니다.

- 12. 7단계: 검토에서 정보를 검토한 다음 파이프라인 생성을 선택합니다.
- 13. 작업을 생성하기 위한 마지막 단계로, 작업에 환경 변수를 추가하여 작업에 대한 출력 변수를 생성 합니다. 명령 작업에서 편집을 선택합니다. 편집 화면에서 변수 네임스페이스 필드에 compute를 입력하여 작업에 대한 변수 네임스페이스를 지정합니다.

CodeBuild 출력 변수 AWS_Default_Region을 추가한 다음 변수 추가를 선택합니다.

Input artifacts

Choose an input artifact for this action. Learn more [

Choose an input artifact for this action. Learn more [2]	FAC Ove
SourceArtifact X Defined by: Source	
No more than 100 characters	
Commands Specify the shell commands to run with your compute action in CodePipeline. You do not ne CodeBuild.	ed to create any resources in CodeBuild. Note: Using compute time for this action will incur separ
ls echo hello echo pipeline Execution Id is #{codepipeline.PipelineExecutionId}	6
Variable namespace - optional	
Choose a namespace for the output variables from this action. You must choose a namespace	e if you want to use the variables this action produces in your configuration. Learn more [
compute	
Variables Specify the names of the variables in your environment that you want to export.	
AWS_DEFAULT_REGION	Add variable
Output artifacts Choose a name for the output of this action. CodePipeline will create the output artifact for	your pipeline artifact store.
Name	
Files	
	Add to file paths

3단계: 파이프라인 실행 및 빌드 명령 확인

변경 사항을 릴리스하여 파이프라인을 실행합니다. 실행 기록, 빌드 로그 및 출력 변수를 확인하여 빌 드 명령이 실행되었는지 확인합니다.

작업 로그 및 출력 변수를 보려면

- 파이프라인이 성공적으로 실행되면 작업에 대한 로그와 출력을 볼 수 있습니다. 1.
- 작업의 출력 변수를 보려면 기록을 선택한 다음 타임라인을 선택합니다. 2.

작업에 추가된 출력 변수를 봅니다. 명령 작업의 출력은 작업 리전으로 확인된 출력 변수를 보여줍 니다.

Artifacts		
Artifact name	Artifact type	Artifact provider
SourceArtifact [2]	Input	Amazon S3
Output variables		
- Kay		Value
AWS_DEFAULT_REGION		us-east-1

 작업에 대한 로그를 보려면 성공한 명령 작업의 세부 정보 보기를 선택합니다. 명령 작업에 대한 로그를 봅니다.

Action execution details

Action name: Commands_action Status: Succeeded

· · · · · · · · · · · · · · · · · · ·		LUL IT LUTUL		THADED TOURS IN THEFT
16	[Container]	2024/10/02	15:04:32.669748	BUILD: 3 commands
17	[Container]	2024/10/02	15:04:32.669974	Phase complete: DOWNLOAD_SOURCE State: SUCCEEDED
18	[Container]	2024/10/02	15:04:32.669989	Phase context status code: Message:
19	[Container]	2024/10/02	15:04:32.764013	Entering phase INSTALL
26) [Container]	2024/10/02	15:04:32.769349	Phase complete: INSTALL State: SUCCEEDED
21	[Container]	2024/10/02	15:04:32.769369	Phase context status code: Message:
22	[Container]	2024/10/02	15:04:32.815049	Entering phase PRE_BUILD
23	[Container]	2024/10/02	15:04:32.820275	Phase complete: PRE_BUILD State: SUCCEEDED
24	[Container]	2024/10/02	15:04:32.820297	Phase context status code: Message:
25	[Container]	2024/10/02	15:04:32.865495	Entering phase BUILD
26	[Container]	2024/10/02	15:04:32.915050	Running command ls
27	README.txt			
28	3			
29	[Container]	2024/10/02	15:04:32.923632	Running command echo hello
30) hello			
31	L			
32	[Container]	2024/10/02	15:04:32.929143	Running command echo pipeline Execution Id is a55e3d
33	3			
34	Ļ			
35	[Container]	2024/10/02	15:04:32.937518	Phase complete: BUILD State: SUCCEEDED
36	[Container]	2024/10/02	15:04:32.937536	Phase context status code: Message:
37	[Container]	2024/10/02	15:04:32.986928	Entering phase POST_BUILD
38	8 [Container]	2024/10/02	15:04:32.992223	Phase complete: POST_BUILD State: SUCCEEDED
39	[Container]	2024/10/02	15:04:32.992242	Phase context status code: Message:
48	3			

자습서: Git 태그를 사용하여 파이프라인 시작하기

이 자습서에서는 Git 태그 트리거 유형에 대해 소스 작업이 구성된 GitHub 리포지토리에 연결되는 파이 프라인을 생성합니다. 커밋에서 Git 태그가 생성되면 파이프라인이 시작됩니다. 이 예제에서는 태그 이 름의 구문을 기반으로 태그를 필터링할 수 있는 파이프라인을 만드는 방법을 보여줍니다. Glog 패턴을 사용한 필터링에 대한 자세한 내용은 <u>구문에서 glob 패턴 작업</u>을 참조하세요.

<u> Important</u>

파이프라인 생성의 일환으로 CodePipeline은 고객이 제공하는 S3 아티팩트 버킷을 아티팩트 에 사용합니다. (이는 S3 소스 작업에 사용되는 버킷과 다릅니다.) S3 아티팩트 버킷이 파이프 라인의 계정과 다른 계정에 있는 경우 S3 아티팩트 버킷을 안전하고 신뢰할 수 AWS 계정 있는 에서 소유해야 합니다.

이 자습서는 CodeStarSourceConnection 작업 유형을 통해 GitHub에 연결합니다.

Note

이 기능은 아시아 태평양(홍콩), 아프리카(케이프타운), 중동(바레인) 또는 유럽(취리히) 리전에서는 사용할 수 없습니다. 사용 가능한 다른 작업을 참조하려면 <u>CodePipeline과 제</u> <u>품 및 서비스 통합</u>을 참조하세요. 유럽(밀라노) 리전에서 이 조치를 고려할 경우 <u>Bitbucket</u> <u>Cloud, GitHub, GitHub Enterprise Server, GitLab.com 및 GitLab 자체 관리형 작업용</u> CodeStarSourceConnection의 참고 사항을 참조하세요.

주제

- <u>사전 조건</u>
- 1단계: CloudShell을 열고 리포지토리 복제
- 2단계: Git 태그에 트리거할 파이프라인 생성
- 3단계: 릴리스용 커밋에 태그 지정
- 4단계: 변경 사항 릴리스 및 로그 보기

사전 조건

시작하기 전에 다음을 수행해야 합니다.

- GitHub 계정을 사용하여 GitHub 리포지토리를 만듭니다.
- GitHub 자격 증명을 준비합니다. AWS Management Console 를 사용하여 연결을 설정하면 GitHub 자격 증명으로 로그인하라는 메시지가 표시됩니다.

1단계: CloudShell을 열고 리포지토리 복제

명령줄 인터페이스를 사용하여 리포지토리를 복제하고, 커밋하고, 태그를 추가할 수 있습니다. 이 자습 서에서는 명령줄 인터페이스용 CloudShell 인스턴스를 시작합니다.

- 1. AWS Management Console에 로그인합니다.
- 2. 상단 탐색 모음에서 AWS 아이콘을 선택합니다. AWS Management Console 디스플레이의 기본 페이지입니다.
- 3. 상단 탐색 모음에서 AWS CloudShell 아이콘을 선택합니다. CloudShell이 열립니다. CloudShell 환경이 생성되는 동안 기다립니다.

1 Note

CloudShell 아이콘이 보이지 않는 경우 <u>CloudShell에서 지원하는 리전</u>에 있는지 확인하세 요. 이 자습서에서는 사용자가 미국 서부(오레곤) 리전에 있다고 가정합니다.

- 4. GitHub에서 리포지토리로 이동합니다. 코드를 선택한 다음 HTTPS를 선택합니다. 경로를 복사합니다. Git 리포지토리를 복제할 주소는 클립보드에 복사됩니다.
- 5. 다음 명령을 실행하여 리포지토리를 복제합니다.

git clone https://github.com/<account>/MyGitHubRepo.git

6. 프롬프트가 표시되면 GitHub 계정 Username과 Password를 입력합니다. Password 항목의 경 우 계정 암호 대신 사용자가 만든 토큰을 사용해야 합니다.

2단계: Git 태그에 트리거할 파이프라인 생성

이 단원에서는 다음 작업을 통해 파이프라인을 생성합니다.

- GitHub 리포지토리 및 작업에 대한 연결이 있는 소스 단계입니다.
- 빌드 작업이 있는 AWS CodeBuild 빌드 단계입니다.

마법사를 사용하여 파이프라인을 생성하려면

- 1. <u>https://console.aws.amazon.com/codepipeline/</u>에서 CodePipeline 콘솔에 로그인합니다.
- 2. Welcome(시작) 페이지, 시작하기 페이지 또는 Pipelines(파이프라인) 페이지에서 파이프라인 생성을 선택합니다.

- 1단계: 생성 옵션 선택 페이지의 생성 옵션에서 사용자 지정 파이프라인 빌드 옵션을 선택합니다.
 다음을 선택합니다.
- 4. 2단계: 파이프라인 설정 선택의 파이프라인 이름에 MyGitHubTagsPipeline을 입력합니다.
- 5. 파이프라인 유형에서는 기본 선택을 V2로 유지합니다. 파이프라인 유형은 특성과 가격이 다릅니 다. 자세한 내용은 파이프라인 유형 단원을 참조하십시오.
- 6. Service role(서비스 역할)에서 New service role(새 서비스 역할)을 선택합니다.

Note

역할을 생성하지 않고 기존 CodePipeline 서비스 역할을 사용할 경우 서비스 역할 정책 에 대한 codestar-connections:UseConnection IAM 권한을 추가해야 합니다. CodePipeline 서비스 역할에 대한 지침은 <u>CodePipeline 서비스 역할에 권한 추가</u>를 참조 하세요.

7. 고급 설정에서 기본값을 그대로 둡니다. [아티팩트 스토어(Artifact store)]에서 [기본 위치(Default location)]를 선택하여 파이프라인에 대해 선택한 리전의 파이프라인에 대해 기본값으로 지정된 Amazon S3 아티팩트 버킷과 같은 기본 아티팩트 스토어를 사용합니다.

Note

이는 소스 코드에 대한 소스 버킷이 아닙니다. 이 파이프라인은 아티팩트 스토어입니다. S3 버킷과 같은 개별 아티팩트 스토어는 각 파이프라인에 필요합니다.

다음을 선택합니다.

- 8. 3단계: 소스 스테이지 추가 페이지에서 소스 스테이지를 추가합니다.
 - a. 소스 공급자에서 GitHub(GitHub 앱을 통해)를 선택합니다.
 - b. 연결에서 기존 연결을 선택하거나 새로 생성합니다. GitHub 소스 작업에 대한 연결을 만들거 나 관리하려면 GitHub 연결을 참조하세요.
 - c. 리포지토리 이름에서 GitHub 리포지토리의 이름을 선택합니다.
 - d. 기본 브랜치에서 파이프라인을 수동으로 시작하거나 Git 태그가 아닌 소스 이벤트로 시작할
 때 지정할 브랜치를 선택합니다. 변경 소스가 트리거가 아니거나 파이프라인 실행이 수동으로 시작된 경우 기본 브랜치의 HEAD 커밋이 사용됩니다.
 - e. Webhook 이벤트의 필터 유형에서 태그를 선택합니다.

태그 또는 패턴 필드에를 입력합니다release*.

\Lambda Important

트리거 유형의 Git 태그로 시작하는 파이프라인은 WebhookV2 이벤트용으로 구성되 며 파이프라인을 시작하는 데 Webhook 이벤트(모든 푸시 이벤트에 대한 변경 감지) 를 사용하지 않습니다.

다음을 선택합니다.

- 9. Add build stage(빌드 스테이지 추가)에서 빌드 스테이지를 추가합니다.
 - a. 빌드 공급자에서 AWS CodeBuild를 선택합니다. 리전이 파이프라인 리전으로 기본 설정되도 록 합니다.
 - b. 프로젝트 생성을 선택합니다.
 - c. 프로젝트 이름에 이 빌드 프로젝트의 이름을 입력합니다.
 - d. 환경 이미지에서 이미지 관리를 선택합니다. [Operating system]에서 [Ubuntu]를 선택합니다.
 - e. 실행 시간에서 표준을 선택합니다. 이미지에서 aws/codebuild/standard:5.0을 선택합니다.
 - f. 서비스 역할에서 New service role(새 서비스 역할)을 선택합니다.

Note

CodeBuild 서비스 역할의 이름을 기록해 둡니다. 이 자습서의 마지막 단계를 수행하 려면 역할 이름이 필요합니다.

g. [Buildspec]의 [빌드 사양(Build specifications)]에서 [빌드 명령 삽입(Insert build commands)] 을 선택합니다. 편집기로 전환을 선택하고 빌드 명령에 다음을 붙여 넣습니다.

```
version: 0.2
#env:
    #variables:
        # key: "value"
        # key: "value"
    #parameter-store:
        # key: "value"
        # key: "value"
        # key: "value"
    # git-credential-helper: yes
```

```
phases:
  install:
    #If you use the Ubuntu standard image 2.0 or later, you must specify
 runtime-versions.
    #If you specify runtime-versions and use an image other than Ubuntu
 standard image 2.0, the build fails.
    runtime-versions:
      nodejs: 12
    #commands:
      # - command
      # - command
  #pre_build:
    #commands:
      # - command
      # - command
  build:
    commands:
  #post_build:
    #commands:
      # - command
      # - command
artifacts:
  files:
     _ '*'
    # - location
  name: $(date +%Y-%m-%d)
  #discard-paths: yes
  #base-directory: location
#cache:
  #paths:
    # - paths
```

- h. Continue to CodePipeline(CodePipeline으로 계속)을 선택합니다. 그러면 CodePipeline 콘솔 로 돌아가고 구성을 위해 빌드 명령을 사용하는 CodeBuild 프로젝트가 생성됩니다. 빌드 프로 젝트는 서비스 역할을 사용하여 AWS 서비스 권한을 관리합니다. 이 단계는 몇 분이 걸릴 수 있습니다.
- i. 다음을 선택합니다.
- 5단계: 테스트 단계 추가에서 테스트 단계 건너뛰기를 선택한 다음 건너뛰기를 다시 선택하여 경고 메시지를 수락합니다.

다음을 선택합니다.

- 11. 6단계: 배포 단계 추가 페이지에서 배포 단계 건너뛰기를 선택한 다음 다시 건너뛰기를 선택하여 경고 메시지를 수락합니다. 다음을 선택합니다.
- 12. 7단계: 검토에서 파이프라인 생성을 선택합니다.

3단계: 릴리스용 커밋에 태그 지정

파이프라인을 생성하고 Git 태그를 지정한 후 GitHub 리포지토리에서 커밋에 태그를 지정할 수 있습니 다. 이 단계에서는 release-1 태그로 커밋에 태그를 지정합니다. Git 리포지토리의 각 커밋에는 고유 한 Git 태그가 있어야 합니다. 커밋을 선택하고 태그를 지정하면 여러 브랜치의 변경 사항을 파이프라 인 배포에 통합할 수 있습니다. 단, 태그 이름 릴리스는 GitHub의 릴리스 개념에는 적용되지 않습니다.

1. 태그를 지정할 복사한 커밋 ID를 참조하세요. 각 브랜치의 커밋을 보려면 CloudShell 터미널에서 다음 명령을 입력하여 태그하려는 커밋 ID를 캡처합니다.

git log

 CloudShell 터미널에서 커밋에 태그를 지정하고 오리진으로 푸시하는 명령을 입력합니다. 커밋 에 태그를 지정한 후 git 푸시 명령을 사용하여 태그를 오리진에 푸시합니다. 다음 예제에서는 ID 49366bd가 있는 두 번째 커밋에 release-1 태그를 사용하려면 다음 명령을 입력합니다. 이 태 그는 파이프라인 release* 태그 필터에 의해 필터링되고 파이프라인을 시작합니다.

git tag release-1 49366bd

git push origin release-1



4단계: 변경 사항 릴리스 및 로그 보기

1. 파이프라인이 성공적으로 실행되면 성공적으로 완료된 빌드 단계에서 로그 보기를 선택합니다.

로그에서 CodeBuild 빌드 출력을 확인합니다. 명령은 입력된 변수의 값을 출력합니다.

2. 기록 페이지에서 트리거 열을 확인합니다. 트리거 유형 GitTag : release-1을 확인합니다.

자습서: 파이프라인을 시작하기 위한 풀 요청의 브랜치 이름 필터링 (V2 유형)

이 자습서에서는 풀 요청을 필터링하는 트리거 구성으로 파이프라인을 시작하도록 소스 작업이 구성 된 GitHub.com 리포지토리에 연결되는 파이프라인을 생성합니다. 지정된 브랜치에 대해 지정된 풀 요 청 이벤트가 발생하면 파이프라인이 시작됩니다. 이 예제에서는 브랜치 이름에 대해 필터링할 수 있는 파이프라인을 만드는 방법을 보여줍니다. 트리거 작업에 대한 자세한 내용은 <u>푸시 및 풀 요청 이벤트</u> <u>유형에 대한 필터 추가(CLI)</u> 섹션을 참조하세요. glob 형식의 정규식 패턴을 사용한 필터링에 대한 자세 한 내용은 <u>구문에서 glob 패턴 작업</u> 섹션을 참조하세요.

A Important

파이프라인 생성의 일환으로 CodePipeline은 고객이 제공하는 S3 아티팩트 버킷을 아티팩트 에 사용합니다. (이는 S3 소스 작업에 사용되는 버킷과 다릅니다.) S3 아티팩트 버킷이 파이프 라인의 계정과 다른 계정에 있는 경우 S3 아티팩트 버킷을 안전하고 신뢰할 수 AWS 계정 있는 에서 소유해야 합니다.

이 자습서는 CodeStarSourceConnection 작업 유형을 통해 GitHub.com에 연결합니다.

주제

- 사전 조건
- 1단계: 지정된 브랜치에 대한 풀 요청을 시작할 파이프라인 생성
- 2단계: GitHub.com에서 풀 요청을 생성 및 병합하여 파이프라인 실행 시작

사전 조건

시작하기 전에 다음을 수행해야 합니다.

- GitHub.com 계정을 사용하여 GitHub.com 리포지토리를 만듭니다.
- GitHub 자격 증명을 준비합니다. AWS Management Console 를 사용하여 연결을 설정하면 GitHub 자격 증명으로 로그인하라는 메시지가 표시됩니다.

1단계: 지정된 브랜치에 대한 풀 요청을 시작할 파이프라인 생성

이 단원에서는 다음 작업을 통해 파이프라인을 생성합니다.

- GitHub.com 리포지토리 및 작업에 대한 연결이 있는 소스 스테이지입니다.
- 빌드 작업이 있는 AWS CodeBuild 빌드 단계입니다.

마법사를 사용하여 파이프라인을 생성하려면

1. https://console.aws.amazon.com/codepipeline/에서 CodePipeline 콘솔에 로그인합니다.

- 2. Welcome(시작) 페이지, 시작하기 페이지 또는 Pipelines(파이프라인) 페이지에서 파이프라인 생성을 선택합니다.
- 1단계: 생성 옵션 선택 페이지의 생성 옵션에서 사용자 지정 파이프라인 빌드 옵션을 선택합니다. Next(다음)를 선택합니다.
- 4. 2단계: 파이프라인 설정 선택의 파이프라인 이름에 MyFilterBranchesPipeline을 입력합니다.
- 파이프라인 유형에서는 기본 선택을 V2로 유지합니다. 파이프라인 유형은 특성과 가격이 다릅니 다. 자세한 내용은 파이프라인 유형 단원을 참조하십시오.
- 6. Service role(서비스 역할)에서 New service role(새 서비스 역할)을 선택합니다.

Note

역할을 생성하지 않고 기존 CodePipeline 서비스 역할을 사용할 경우 서비스 역할 정책에 대한 codeconnections:UseConnection IAM 권한을 추가해야 합니다. CodePipeline 서비스 역할에 대한 지침은 CodePipeline 서비스 역할에 권한 추가를 참조하세요.

7. 고급 설정에서 기본값을 그대로 둡니다. [아티팩트 스토어(Artifact store)]에서 [기본 위치(Default location)]를 선택하여 파이프라인에 대해 선택한 리전의 파이프라인에 대해 기본값으로 지정된 Amazon S3 아티팩트 버킷과 같은 기본 아티팩트 스토어를 사용합니다.

Note

이는 소스 코드에 대한 소스 버킷이 아닙니다. 이 파이프라인은 아티팩트 스토어입니다. S3 버킷과 같은 개별 아티팩트 스토어는 각 파이프라인에 필요합니다.

Next(다음)를 선택합니다.

- 8. 3단계: 소스 스테이지 추가 페이지에서 소스 스테이지를 추가합니다.
 - a. 소스 공급자에서 GitHub(GitHub 앱을 통해)를 선택합니다.
 - b. 연결에서 기존 연결을 선택하거나 새로 생성합니다. GitHub 소스 작업에 대한 연결을 만들거 나 관리하려면 GitHub 연결을 참조하세요.
 - c. 리포지토리 이름에서 GitHub.com 리포지토리의 이름을 선택합니다.
 - d. 트리거 유형에서 필터 지정을 선택합니다.

이벤트 유형에서 풀 요청을 선택합니다. 풀 요청의 모든 이벤트를 선택하여 생성, 업데이트 또 는 종료된 풀 요청에 대해 이벤트가 발생하도록 합니다.

브랜치의 포함 필드에 main*을 입력합니다.

For source action: Source			Remove
Filters			
Pull request	Ġ		
Events: Created Closed Re	vised		
Include branches: main*		+ Add filter	
	2 ×		
		+ Add trigger	

용하지 않습니다.

Next(다음)를 선택합니다.

- 4단계: 빌드 단계 추가의 빌드 공급자에서를 선택합니다AWS CodeBuild. 리전이 파이프라인 리전 으로 기본 설정되도록 합니다. <u>자습서: Git 태그를 사용하여 파이프라인 시작하기</u>의 지침에 따라 빌드 프로젝트를 선택하거나 생성합니다. 이 작업은 이 자습서에서 파이프라인을 생성하는 데 필 요한 두 번째 단계로만 사용됩니다.
- 5단계: 테스트 단계 추가에서 테스트 단계 건너뛰기를 선택한 다음 건너뛰기를 다시 선택하여 경고 메시지를 수락합니다.

Next(다음)를 선택합니다.

11. 6단계: 배포 단계 추가 페이지에서 배포 단계 건너뛰기를 선택한 다음 다시 건너뛰기를 선택하여 경고 메시지를 수락합니다. Next(다음)를 선택합니다. 12. 7단계: 검토에서 파이프라인 생성을 선택합니다.

2단계: GitHub.com에서 풀 요청을 생성 및 병합하여 파이프라인 실행 시작

이 섹션에서는 풀 요청을 생성하고 병합합니다. 이렇게 하면 파이프라인이 시작되며, 열린 풀 요청에 대해 한 번 실행되고 닫힌 풀 요청에 대해 한 번 실행됩니다.

풀 요청을 생성하고 파이프라인을 시작하려면

- 1. GitHub.com에서 기능 브랜치의 README.md를 변경하고 main 브랜치에 풀 요청을 제기하여 풀 요청을 생성합니다. Update README.md for PR과 같은 메시지로 변경 사항을 커밋합니다.
- 파이프라인은 풀 요청에 대한 소스 메시지가 PR용 README.md 업데이트로 표시되는 소스 개정 으로 시작됩니다.

Source Succeeded Pipeline execution ID: <u>7e1becf6-fc7d-44b5-9157</u>	<u>-56e9d7982a93</u>		
Source <u>GitHub (Version 2)</u> ^[2] Succeeded - <u>Just now</u> <u>030c6b39</u> ^[2] View details <u>030c6b39</u> ^[2] Source: Update README.md for PR			
Disable transition			
Build In progress Pipeline execution ID: <u>7e1becf6-fc7d-44b5-9157</u>	<u>-56e9d7982a93</u>		

 History(기록)를 선택합니다. 파이프라인 실행 기록에서 파이프라인 실행을 시작한 CREATED 및 MERGED 풀 요청 상태 이벤트를 확인합니다.

Developer Tools > CodePipeline > Pipelines > new-github > Execution history						
Execution history Info				Rerun Stop exect	ution View details	Release change
	Execution ID	Status	Trigger	Started	Duration	Completed
0	61986255	✓ Succeeded	PullRequest 5 MERGED From repository/branch: /MyGitHubRepo /feature-branch 2 To repository/branch: /MyGitHubRepo /main 2	Feb 7, 2024 6:26 PM (UTC-8:00)	5 minutes 31 seconds	Feb 7, 2024 6:32 PM (UTC-8:00)
0	b9614702	Succeeded	PullRequest 5 CREATED From repository/branch: //MyGitHubRepo /feature-branch [2] To repository/branch: //MyGitHubRepo /main [2]	Feb 7, 2024 6:26 PM (UTC-8:00)	4 minutes 7 seconds	Feb 7, 2024 6:30 PM (UTC-8:00)
0	09c14335	⊘ Succeeded	Webhook - connection/40d122c4-23fb- 48bf- a08f-1cd9	Feb 5, 2024 1:19 AM (UTC-8:00)	2 days 16 hours	Feb 7, 2024 5:38 PM (UTC-8:00)

자습서: 파이프라인 수준 변수 사용

이 자습서에서는 파이프라인 수준에서 변수를 추가하는 파이프라인을 만들고 변수 값을 출력하는 CodeBuild 빌드 작업을 실행합니다.

A Important

파이프라인 생성의 일환으로 CodePipeline은 고객이 제공하는 S3 아티팩트 버킷을 아티팩트 에 사용합니다. (이는 S3 소스 작업에 사용되는 버킷과 다릅니다.) S3 아티팩트 버킷이 파이프 라인의 계정과 다른 계정에 있는 경우 S3 아티팩트 버킷을 안전하고 신뢰할 수 AWS 계정 있는 에서 소유해야 합니다.

주제

- <u>사전 조건</u>
- 1단계: 파이프라인 및 빌드 프로젝트 생성
- 2단계: 변경 사항 릴리스 및 로그 보기

사전 조건

시작하기 전에 다음을 수행해야 합니다.

- CodeCommit 리포지토리를 생성합니다.
- 리포지토리에 .txt 파일을 추가합니다

1단계: 파이프라인 및 빌드 프로젝트 생성

이 단원에서는 다음 작업을 통해 파이프라인을 생성합니다.

- CodeCommit 리포지토리에 대한 연결이 있는 소스 스테이지입니다.
- 빌드 작업이 있는 AWS CodeBuild 빌드 단계입니다.

마법사를 사용하여 파이프라인을 생성하려면

- 1. <u>https://console.aws.amazon.com/codepipeline/에서 CodePipeline 콘솔에 로그인합니다.</u>
- 2. Welcome(시작) 페이지, 시작하기 페이지 또는 Pipelines(파이프라인) 페이지에서 파이프라인 생성을 선택합니다.
- 3. 1단계: 생성 옵션 선택 페이지의 생성 옵션에서 사용자 지정 파이프라인 빌드 옵션을 선택합니다. Next(다음)를 선택합니다.
- 4. 2단계: 파이프라인 설정 선택의 파이프라인 이름에 MyVariablesPipeline을 입력합니다.
- 파이프라인 유형에서는 기본 선택을 V2로 유지합니다. 파이프라인 유형은 특성과 가격이 다릅니 다. 자세한 내용은 <u>파이프라인 유형</u> 단원을 참조하십시오.
- 6. Service role(서비스 역할)에서 New service role(새 서비스 역할)을 선택합니다.

역할을 생성하지 않고 기존 CodePipeline 서비스 역할을 사용할 경우 서비스 역할 정책에 대한 codeconnections:UseConnection IAM 권한을 추가해야 합니다. CodePipeline 서비스 역할에 대한 지침은 CodePipeline 서비스 역할에 권한 추가를 참조하세요.

 변수에서 변수 추가를 선택합니다. 이름에 timeout를 입력합니다. 기본값에 1000을 입력합니다. 설명에 Timeout 설명을 입력합니다.

¹ Note
그러면 파이프라인 실행이 시작될 때 값을 선언할 수 있는 변수가 생성됩니다. 변수 이름은 [A-Za-z0-9e\-_]+와 일치해야 하며 빈 문자열 이외의 모두 가능합니다.

 고급 설정에서 기본값을 그대로 둡니다. [아티팩트 스토어(Artifact store)]에서 [기본 위치(Default location)]를 선택하여 파이프라인에 대해 선택한 리전의 파이프라인에 대해 기본값으로 지정된 Amazon S3 아티팩트 버킷과 같은 기본 아티팩트 스토어를 사용합니다.

In Note

이는 소스 코드에 대한 소스 버킷이 아닙니다. 이 파이프라인은 아티팩트 스토어입니다. S3 버킷과 같은 개별 아티팩트 스토어는 각 파이프라인에 필요합니다.

Next(다음)를 선택합니다.

- 9. 3단계: 소스 스테이지 추가 페이지에서 소스 스테이지를 추가합니다.
 - a. 소스 공급자에서 AWS CodeCommit을 선택합니다.
 - b. 리포지토리 이름 및 브랜치 이름에서 리포지토리와 브랜치를 선택합니다.

Next(다음)를 선택합니다.

- 10. 4단계: 빌드 단계 추가에서 빌드 단계를 추가합니다.
 - a. 빌드 공급자에서 AWS CodeBuild를 선택합니다. 리전이 파이프라인 리전으로 기본 설정되도 록 합니다.
 - b. 프로젝트 생성을 선택합니다.
 - c. 프로젝트 이름에 이 빌드 프로젝트의 이름을 입력합니다.
 - d. 환경 이미지에서 이미지 관리를 선택합니다. [Operating system]에서 [Ubuntu]를 선택합니다.
 - e. 실행 시간에서 표준을 선택합니다. 이미지에서 aws/codebuild/standard:5.0을 선택합니다.
 - f. 서비스 역할에서 New service role(새 서비스 역할)을 선택합니다.

Note

CodeBuild 서비스 역할의 이름을 기록해 둡니다. 이 자습서의 마지막 단계를 수행하 려면 역할 이름이 필요합니다. g. [Buildspec]의 [빌드 사양(Build specifications)]에서 [빌드 명령 삽입(Insert build commands)] 을 선택합니다. 편집기로 전환을 선택하고 빌드 명령에 다음을 붙여 넣습니다. buildspec에서 는 고객 변수 \$CUSTOM_VAR1을 사용하여 빌드 로그의 파이프라인 변수를 출력합니다. 다음 단계에서 \$CUSTOM_VAR1 출력 변수를 환경 변수로 생성합니다.

```
version: 0.2
#env:
  #variables:
     # key: "value"
     # key: "value"
  #parameter-store:
     # key: "value"
     # key: "value"
  #git-credential-helper: yes
phases:
  install:
    #If you use the Ubuntu standard image 2.0 or later, you must specify
 runtime-versions.
    #If you specify runtime-versions and use an image other than Ubuntu
 standard image 2.0, the build fails.
    runtime-versions:
      nodejs: 12
    #commands:
      # - command
      # - command
  #pre_build:
    #commands:
      # - command
      # - command
  build:
    commands:
      - echo $CUSTOM_VAR1
  #post_build:
    #commands:
      # - command
      # - command
artifacts:
  files:
     _ '*'
    # - location
  name: $(date +%Y-%m-%d)
  #discard-paths: yes
  #base-directory: location
```

#cache	9:	
#pat	h	5:
#	-	paths

- h. Continue to CodePipeline(CodePipeline으로 계속)을 선택합니다. 그러면 CodePipeline 콘솔 로 돌아가고 구성을 위해 빌드 명령을 사용하는 CodeBuild 프로젝트가 생성됩니다. 빌드 프로 젝트는 서비스 역할을 사용하여 AWS 서비스 권한을 관리합니다. 이 단계는 몇 분이 걸릴 수 있습니다.
- 환경 변수 선택 사항에서 파이프라인 수준 변수로 확인된 빌드 작업의 입력 변수로 환경 변수를 만들려면 환경 변수 추가를 선택합니다. 그러면 buildspec에 지정된 변 수가 \$CUSTOM_VAR1으로 생성됩니다. 이름에 CUSTOM_VAR1를 입력합니다. 값에는 #{variables.timeout}를 입력합니다. 유형에서 Plaintext를 선택합니다.

환경 변수의 #{variables.timeout} 값은 파이프라인 수준 변수 네임스페이 스variables와 7단계에서 파이프라인에 대해 timeout 생성된 파이프라인 수준 변수를 기 반으로 합니다.

- j. Next(다음)를 선택합니다.
- 11. 5단계: 테스트 단계 추가에서 테스트 단계 건너뛰기를 선택한 다음 건너뛰기를 다시 선택하여 경 고 메시지를 수락합니다.

Next(다음)를 선택합니다.

- 12. 6단계: 배포 단계 추가 페이지에서 배포 단계 건너뛰기를 선택한 다음 다시 건너뛰기를 선택하여 경고 메시지를 수락합니다. Next(다음)를 선택합니다.
- 13. 7단계: 검토에서 파이프라인 생성을 선택합니다.

2단계: 변경 사항 릴리스 및 로그 보기

파이프라인이 성공적으로 실행되면 성공적으로 완료된 빌드 단계에서 세부 정보 보기를 선택합니다.

세부 정보 페이지에서 로그 탭을 선택합니다. CodeBuild 빌드 출력을 확인합니다. 명령은 입력된 변수의 값을 출력합니다.

2. 왼쪽 탐색 창에서 기록을 선택합니다.

최근 실행을 선택한 다음, 변수 탭을 선택합니다. 파이프라인 변수의 확인된 값을 확인합니다.

자습서: 간단한 파이프라인 생성(S3 버킷)

파이프라인을 생성하는 가장 쉬운 방법은 AWS CodePipeline 콘솔에서 파이프라인 생성 마법사를 사 용하는 것입니다.

이 자습서에서는 버전이 지정된 S3 소스 버킷과 CodeDeploy를 사용하여 샘플 애플리케이션을 릴리스 하는 두 단계 파이프라인을 생성합니다.

Note

Amazon S3가 파이프라인의 소스 공급자인 경우, 소스 파일을 .zip 하나로 압축하고 그 .zip을 소스 버킷에 업로드할 수 있습니다. 압축이 풀린 단일 파일을 업로드할 수도 있지만 .zip 파일을 예상하는 다운스트림 작업은 실패합니다.

A Important

파이프라인 생성의 일환으로 CodePipeline은 고객이 제공하는 S3 아티팩트 버킷을 아티팩트 에 사용합니다. (이는 S3 소스 작업에 사용되는 버킷과 다릅니다.) S3 아티팩트 버킷이 파이프 라인의 계정과 다른 계정에 있는 경우 S3 아티팩트 버킷을 안전하고 신뢰할 수 AWS 계정 있는 에서 소유해야 합니다.

이 간단한 파이프라인을 생성한 후 다른 단계를 추가한 다음 단계 간 전환을 비활성화하고 활성화합니 다.

A Important

이 절차에서 파이프라인에 추가하는 많은 작업에는 파이프라인을 생성하기 전에 생성해야 하 는 AWS 리소스가 포함됩니다. 소스 작업에 대한 AWS 리소스는 항상 파이프라인을 생성하는 리전과 동일한 AWS 리전에서 생성해야 합니다. 예를 들어, 미국 동부(오하이오) 리전에서 파 이프라인을 생성하는 경우, CodeCommit 리포지토리도 미국 동부(오하이오) 리전에 있어야 합 니다.

파이프라인을 생성할 때 교차 리전 작업을 추가할 수 있습니다. 교차 리전 작업에 대한 AWS 리소스는 작업을 실행하려는 리전과 동일한 AWS 리전에 있어야 합니다. 자세한 내용은 CodePipeline에 교차 리전 작업 추가 단원을 참조하십시오. 시작하기 전에 CodePipeline 시작하기 단원의 사전 조건을 충족해야 합니다.

주제

- 1단계: 애플리케이션에 대한 S3 소스 버킷 생성
- 2단계: Amazon EC2 Windows 인스턴스 생성 및 CodeDeploy 에이전트 설치
- 3단계: CodeDeploy에서 애플리케이션 생성
- 4단계: CodePipeline에서 첫 번째 파이프라인 생성
- (선택 사항) 5단계: 파이프라인에 다른 단계 추가
- (선택 사항) 6단계: CodePipeline에서 단계 간 전환 비활성화 및 활성화
- 7단계: 리소스 정리

1단계: 애플리케이션에 대한 S3 소스 버킷 생성

버전이 지정된 위치에 소스 파일이나 애플리케이션을 저장할 수 있습니다. 이 자습서에서는 샘플 애플 리케이션 파일에 대한 S3 버킷을 생성하고 해당 버킷의 버전 관리를 활성화합니다. 버전 관리를 활성 화한 후 샘플 애플리케이션을 해당 버킷에 복사합니다.

S3 버킷을 생성하려면,

- 1. 에서 콘솔에 로그인합니다 AWS Management Console. S3 콘솔을 엽니다.
- 2. 버킷 생성을 선택합니다.
- 3. [Bucket Name]에서 버킷 이름을 입력합니다(예: awscodepipeline-demobucket-exampledate).

Note

Amazon S3에 있는 모든 버킷 이름은 고유해야 하므로, 예에 표시된 이름이 아닌, 사용자 고유의 이름을 사용하세요. 이름에 날짜만 추가하여 간단히 예제 이름을 변경해도 됩니다. 이 자습서의 나머지 부분에서 이 이름을 사용할 것이므로 이름을 메모해 둡니다.

리전에서 미국 서부(오레곤)과 같은 파이프라인을 생성할 리전을 선택한 다음, 버킷 생성을 선택합 니다.

4. 버킷이 생성되면 성공적으로 수행했다는 배너가 표시됩니다. [Go to bucket details]를 선택합니다.

5. [Properties] 탭에서 [Versioning]을 선택합니다. [Enable versioning]을 선택한 다음 [Save]를 선택 합니다.

버전 관리가 활성화되면 Amazon S3이 버킷의 모든 객체 버전을 저장합니다.

- [Permissions] 탭에서 기본값을 그대로 둡니다. S3의 버킷 및 객체 권한에 대한 자세한 내용은 <u>정</u> 책에서 권한 지정 단원을 참조하십시오.
- 7. 그런 다음 샘플을 다운로드하여 로컬 컴퓨터의 폴더나 디렉토리에 저장합니다.
 - a. 다음 중 하나를 선택합니다. Windows Server 인스턴스에 대해 이 자습서의 단계를 수행하려 는 경우 SampleApp_Windows.zip을 선택합니다.
 - CodeDeploy를 사용하여 Amazon Linux 인스턴스에 배포하려는 경우 여기에서 샘플 애플 리케이션을 다운로드하세요. SampleApp_Linux.zip
 - CodeDeploy를 사용하여 Windows Server 인스턴스에 배포하려는 경우 여기에서 샘플 애플 리케이션을 다운로드하세요. SampleApp_Windows.zip

샘플 애플리케이션에는 CodeDeploy로 배포하기 위한 다음 파일이 포함되어 있습니다.

- appspec.yml 애플리케이션 사양 파일(AppSpec file)은 CodeDeploy에서 배포를 관리 하는 데 사용하는 <u>YAML</u> 형식의 파일입니다. AppSpec 파일에 대한 자세한 내용은 AWS CodeDeploy 사용 설명서의 CodeDeploy AppSpec 파일 참조를 참조하세요.
- index.html 인덱스 파일에는 배포된 샘플 애플리케이션의 홈 페이지가 포함되어 있습니다.
- LICENSE.txt 라이선스 파일에는 샘플 애플리케이션에 대한 라이선스 정보가 포함되어 있습니다.
- 스크립트용 파일 샘플 애플리케이션은 스크립트를 사용하여 인스턴스의 위치에 텍스트 파일을 씁니다. 여러 CodeDeploy의 배포 수명 주기 이벤트 각각에 대해 하나의 파일이 작 성됩니다.
 - (Linux 샘플만 해당) scripts 폴더 이 폴더에는 종속성을 설치하고 자동 배포 를 위한 샘플 애플리케이션을 시작 및 중지하기 위한 install_dependencies, start_server, stop_server와 같은 쉘 스크립트가 포함되어 있습니다.
 - (Windows 샘플만 해당) before-install.bat 이것은 BeforeInstall 배포 수명 주기 이벤트의 배치 스크립트로, 이 샘플의 이전 배포 중 기록된 이전 파일을 제거하기 위 해 실행되고 인스턴스에 새 파일을 기록할 위치를 만듭니다.
- b. 압축된 파일을 다운로드합니다. 파일의 압축을 풀지 마십시오.

- 8. Amazon S3 콘솔에서 버킷용 파일 을 업로드합니다.
 - a. 업로드를 선택합니다.
 - b. 파일을 끌어서 놓거나 파일 추가를 선택하고 파일을 찾아봅니다.
 - c. 업로드를 선택합니다.

2단계: Amazon EC2 Windows 인스턴스 생성 및 CodeDeploy 에이전트 설치

Note

이 자습서는 Amazon EC2 Windows 인스턴스를 생성하기 위한 샘플 단계를 제공합니다. Amazon EC2 Linux 인스턴스를 생성하는 샘플 단계는 <u>3단계: Amazon EC2 Linux 인스턴스 생</u> <u>성 및 CodeDeploy 에이전트 설치</u> 단원을 참조하세요. 생성할 인스턴스 수를 묻는 메시지가 표 시되면 2개의 인스턴스를 지정하십시오.

이 단계에는 샘플 애플리케이션을 배포할 Windows Server Amazon EC2 인스턴스를 생성합니다. 이 프로세스의 일부로 인스턴스에 CodeDeploy 에이전트를 설치하고 관리할 수 있는 정책이 포함된 인스 턴스 역할을 생성합니다. CodeDeploy 에이전트는 인스턴스를 CodeDeploy 배포에서 사용할 수 있게 해주는 소프트웨어 패키지입니다. 또한 CodeDeploy 에이전트가 애플리케이션을 배포하는 데 사용하 는 파일을 인스턴스가 가져오도록 허용하고 SSM에서 인스턴스를 관리할 수 있도록 허용하는 정책을 첨부합니다.

인스턴스 역할을 생성하려면

- 1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
- 2. 콘솔 대시보드에서 역할을 선택합니다.
- 3. 역할 생성을 선택합니다.
- 4. 신뢰할 수 있는 유형의 엔터티 선택에서 AWS 서비스를 선택합니다. Choose a use case(사용 사 례 선택)에서 EC2를 선택한 후 Next: Permissions(다음: 권한)을 선택합니다.
- 5. 검색하여 AmazonEC2RoleforAWSCodeDeploy라는 정책을 선택합니다.
- 검색하여 AmazonSSMManagedInstanceCore라는 정책을 선택합니다. 다음: 태그를 선택합니다.
 다.
- 7. 다음: 검토를 선택합니다. 역할의 이름을 입력합니다(예: EC2InstanceRole).

Note

다음 단계를 위해 역할 이름을 적어 둡니다. 인스턴스를 생성할 때 이 역할을 선택합니다.

역할 생성을 선택합니다.

인스턴스를 시작하려면

- 1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
- 2. 측면 탐색에서 인스턴스를 선택하고 페이지 상단에서 인스턴스 시작을 선택합니다.
- 3. 이름 및 태그 아래의 이름에 MyCodePipelineDemo를 입력하세요. 그러면 인스턴스에 Name의 태그 키와 MyCodePipelineDemo의 태그 값이 할당됩니다. 나중에 샘플 애플리케이션을 인스턴 스에 배포하는 CodeDeploy 애플리케이션을 생성합니다. CodeDeploy는 태그를 기반으로 배포할 인스턴스를 선택합니다.
- 애플리케이션 및 OS 이미지(Amazon Machine Image)에서 Windows 옵션을 선택합니다. (이 AMI 는 Microsoft Windows Server 2019 Base로 설명되며 "프리 티어 사용 가능"으로 표시되고 빠른 시 작에서 확인할 수 있습니다.)
- 5. 인스턴스 유형에서 인스턴스의 하드웨어 구성으로 사용할 프리 티어 가능 t2.micro 유형을 선택 합니다.
- 6. 키 페어(로그인)에서 키 페어를 선택하거나 새로 생성합니다.

키 페어 없이 계속을 선택할 수도 있습니다.

Note

이 자습서의 목적상 키 페어 없이 진행할 수 있습니다. SSH를 사용하여 인스턴스에 연결 하려면 키 페어를 생성하거나 사용합니다.

7. 네트워크 설정에서 다음을 수행합니다.

퍼블릭 IP 자동 할당에서 상태가 활성화인지 확인합니다.

- [Assign a security group] 옆에 있는 [Create a new security group]을 선택합니다.
- SSH 행의 소스 유형에서 내 IP를 선택합니다.
- 보안 그룹 추가를 선택하고, HTTP를 선택한 다음, 소스 유형에서 내 IP를 선택합니다.

- 8. Advanced details(고급 세부 정보)를 확장합니다. IAM 인스턴스 프로파일에서 이전 절차에서 생성 한 IAM 역할을 선택합니다(예: EC2InstanceRole).
- 9. 요약에서 인스턴스 수에 2를 입력합니다.
- 10. 인스턴스 시작을 선택합니다.
- 11. 모든 인스턴스 보기(View all instances)를 선택하여 확인 페이지를 닫고 콘솔로 돌아갑니다.
- 12. [Instances] 페이지에서 시작 상태를 볼 수 있습니다. 인스턴스를 시작할 때 초기 상태는 pending입니다. 인스턴스가 시작된 후에는 상태가 running으로 바뀌고 퍼블릭 DNS 이름을 받 습니다. [Public DNS] 열이 표시되지 않으면 [Show/Hide] 아이콘을 선택하고 [Public DNS]를 선택 합니다.
- 13. 연결할 수 있도록 인스턴스가 준비될 때까지 몇 분 정도 걸릴 수 있습니다. 인스턴스가 상태 확인 을 통과했는지 확인합니다. [Status Checks] 열에서 이 정보를 볼 수 있습니다.

3단계: CodeDeploy에서 애플리케이션 생성

CodeDeploy에서 애플리케이션은 배포하려는 코드의 이름 형태의 식별자입니다. CodeDeploy는 이 이 름을 사용하여 배포 중에 개정, 배포 구성 및 배포 그룹의 올바른 조합을 참조하도록 합니다. 이 자습서 후반부에 파이프라인을 생성할 때, 이 단계에 생성되는 CodeDeploy 애플리케이션의 이름을 선택합니 다.

먼저 사용할 CodeDeploy에 대한 서비스 역할을 만듭니다. 서비스 역할을 이미 생성한 경우 다른 서비 스 역할을 생성할 필요가 없습니다.

CodeDeploy 서비스 역할을 만들려면

- 1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
- 2. 콘솔 대시보드에서 역할을 선택합니다.
- 3. 역할 생성을 선택합니다.
- 4. 신뢰할 수 있는 엔터티 선택에서 AWS 서비스를 선택합니다. 사용 사례에서 CodeDeploy를 선택합니다. 나열된 옵션에서 CodeDeploy를 선택합니다. Next(다음)를 선택합니다. AWSCodeDeployRole 관리형 정책이 역할에 연결됩니다.
- 5. Next(다음)를 선택합니다.
- 6. 역할 이름(예: CodeDeployRole)을 입력한 후 Create role(역할 생성)을 선택합니다.

CodeDeploy에서 애플리케이션을 만들려면

- 1. <u>https://console.aws.amazon.com/codedeploy</u>에서 CodeDeploy 콘솔을 엽니다.
- 2. 애플리케이션 페이지가 표시되지 않으면 AWS CodeDeploy 메뉴에서 애플리케이션을 선택합니다.
- 3. 애플리케이션 생성을 선택합니다.
- 4. 애플리케이션 이름에 MyDemoApplication을 입력합니다.
- 5. Compute Platform(컴퓨팅 플랫폼)에서 EC2/On-premises(EC2/온프레미스)를 선택합니다.
- 6. 애플리케이션 생성을 선택합니다.

CodeDeploy에서 배포 그룹을 생성하려면

- 1. 애플리케이션이 표시되는 페이지에서 Create deployment group(배포 그룹 생성)을 선택합니다.
- 2. Deployment group name(배포 그룹 이름)에 MyDemoDeploymentGroup을 입력합니다.
- 서비스 역할에서 앞서 생성한 서비스 역할을 선택합니다. AWS CodeDeploy에 대한 서비스 역할 <u>생성에 설명된 신뢰 및 권한과 함께 최소한 CodeDeploy를 신뢰하는 서비스 역할을</u> 사용해야 합니 다. 서비스 역할 ARN을 가져오려면 서비스 역할 ARN 가져오기(콘솔)를 참조하십시오.
- 4. 배포 유형 아래에서 인 플레이스를 선택합니다.
- 5. [Environment configuration]에서 [Amazon EC2 Instances] 탭을 선택합니다. 키 필드에서 이름을 선택하고 값 필드에 **MyCodePipelineDemo**를 입력합니다.

▲ Important

EC2 인스턴스를 생성할 때 할당한 동일한 값을 여기의 이름 키에 대해 선택해야 합니다. MyCodePipelineDemo가 아닌 이름으로 인스턴스에 태그를 지정한 경우, 여기에서 해당 이름을 사용해야 합니다.

- 6. AWS Systems Manager를 사용한 에이전트 구성에서 지금을 선택하고 업데이트를 예약합니다. 그러면 인스턴스에 에이전트가 설치됩니다. Windows 인스턴스는 이미 SSM 에이전트로 구성되어 있으며 이제 CodeDeploy 에이전트로 업데이트됩니다.
- 7. 배포 설정에서 CodeDeployDefault.OneAtaTime을 선택합니다.
- 로드 밸런서에서 로드 밸런싱 활성화 상자가 선택되지 않았는지 확인합니다. 이 예에서는 로드 밸 런서를 설정하거나 대상 그룹을 선택할 필요가 없습니다. 확인란의 선택을 취소하면 로드 밸런서 옵션이 표시되지 않습니다.
- 9. 고급 섹션에서 기본값을 그대로 둡니다.

10. [Create deployment group]을 선택합니다.

4단계: CodePipeline에서 첫 번째 파이프라인 생성

자습서의 이 부분에서는 파이프라인을 생성합니다. 샘플은 파이프라인을 통해 자동으로 실행됩니다.

CodePipeline 자동화 릴리스 프로세스 생성 방법

- 1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> <u>codepipeline/home</u>://https//https://https//h
- 2. [Welcome] 페이지, [Getting started] 페이지 또는 [Pipelines] 페이지에서 Create pipeline(파이프라 인생성)을 선택합니다.
- 3. 1단계: 생성 옵션 선택 페이지의 생성 옵션에서 사용자 지정 파이프라인 빌드 옵션을 선택합니다. Next(다음)를 선택합니다.
- 4. 2단계: 파이프라인 설정 선택의 파이프라인 이름에 MyFirstPipeline을 입력합니다.

Note

파이프라인에 다른 이름을 선택하는 경우, 이 자습서의 나머지 부분에서 MyFirstPipeline 대신 해당 이름을 사용해야 합니다. 파이프라인을 만든 후에는 해당 이름을 변경할 수 없습니다. 파이프라인 이름에는 일부 제한이 적용됩니다. 자세한 내용은 in AWS CodePipeline 할당량 단원을 참조하십시오.

- CodePipeline은 특성과 가격이 다른 V1 및 V2 유형 파이프라인을 제공합니다. V2 유형은 콘 솔에서 선택할 수 있는 유일한 유형입니다. 자세한 내용은 <u>파이프라인 유형</u>을 참조하세요. CodePipeline에 대한 요금 정보는 요금을 참조하세요.
- 6. 서비스 역할에서 다음 중 하나를 수행합니다.
 - 새 서비스 역할을 선택하여 CodePipeline이 IAM에 새 서비스 역할을 생성하도록 허용합니다.
 - Existing service role(기존 서비스 역할)을 선택하여 IAM에서 이미 생성된 서비스 역할을 사용합니다. 역할 이름의 목록에서 서비스 역할을 선택합니다.
- 7. [Advanced settings]의 설정은 기본값 그대로 두고, [Next]를 선택합니다.
- 3단계: 소스 스테이지 추가의 소스 공급자에서 Amazon S3를 선택합니다. [Bucket]에 <u>1단계:</u> <u>애플리케이션에 대한 S3 소스 버킷 생성</u>에서 생성한 S3 버킷의 이름을 입력합니다. S3 객체 키에 파일 경로가 있거나 없는 객체 키를 입력하고 파일 확장명을 포함해야 합니다. 예를 들어, SampleApp_Windows.zip의 경우 다음 예제에 표시된 대로 샘플 파일 이름을 입력합니다.

SampleApp_Windows.zip

다음 단계를 선택합니다.

[Change detection options] 아래에서 기본값을 그대로 둡니다. 그러면 CodePipeline이 Amazon CloudWatch Events를 사용하여 소스 버킷의 변경을 감지할 수 있습니다.

Next(다음)를 선택합니다.

- 9. 4단계: 빌드 스테이지 추가에서 빌드 스테이지 건너뛰기를 선택하고 건너뛰기를 다시 선택하여 경고 메시지를 수락합니다. Next(다음)를 선택합니다.
- 5단계: 테스트 단계 추가에서 테스트 단계 건너뛰기를 선택한 다음 건너뛰기를 다시 선택하여 경고 메시지를 수락합니다.

Next(다음)를 선택합니다.

11. 6단계: 배포 단계 추가의 배포 공급자에서 CodeDeploy를 선택합니다. 리전 필드는 기본적으로 파이프라인 AWS 리전 과 동일합니다. 애플리케이션 이름에 MyDemoApplication을 입력하 거나 새로 고침 버튼을 선택한 다음, 목록에서 애플리케이션 이름을 선택합니다. 배포 그룹에서 MyDemoDeploymentGroup을 입력하거나, 목록에서 이를 선택한 후 다음을 선택합니다.

Note

Deploy라는 이름은 파이프라인의 첫 단계에 Source라는 이름이 지정되는 것처럼 4단계: 배포 단계 추가 단계에서 생성한 단계에 기본적으로 지정되는 이름입니다.

- 12. 7단계: 검토에서 정보를 검토한 다음 파이프라인 생성을 선택합니다.
- 13. 파이프라인이 실행을 시작합니다. CodePipeline 샘플이 CodeDeploy 배포의 각 Amazon EC2 인 스턴스에 웹 페이지를 배포하므로 진행 사항과 성공 및 실패 메시지를 볼 수 있습니다.

축하합니다! 방금 CodePipeline에서 간단한 파이프라인을 생성했습니다. 파이프라인에는 두 단계가 있 습니다.

- Source라는 소스 단계는 S3 버킷에 저장된 버전이 지정된 샘플 애플리케이션의 변경 사항을 감지하고 이러한 변경 사항을 파이프라인으로 가져옵니다.
- 배포 단계는 CodeDeploy를 이용해 이러한 변경 사항을 EC2 인스턴스에 배포합니다.

이제 결과를 확인합니다.

파이프라인이 성공적으로 실행되었는지 확인하려면

- 파이프라인의 초기 진행 상황을 확인합니다. 각 단계의 상태는 [No executions yet]에서 [In Progress]로 바뀌며, 다시 [Succeeded]나 [Failed] 중 하나로 바뀝니다. 파이프라인은 몇 분 내로 첫 번째 실행을 완료해야 합니다.
- 2. 작업 상태가 성공으로 표시되면 배포 스테이지의 상태 영역에서 세부 정보를 선택합니다. 그러면 CodeDeploy 콘솔이 열립니다.
- 3. 배포 그룹 탭의 배포 수명 주기 이벤트에서 인스턴스 ID를 선택합니다. EC2 콘솔이 열립니다.
- 4. [Description] 탭의 [Public DNS]에서, 주소를 복사한 다음 이를 웹 브라우저의 주소 표시줄에 붙여 넣습니다. S3 버킷에 업로드한 샘플 애플리케이션에 대한 인덱스 페이지를 봅니다.

웹페이지에 S3 버킷에 업로드한 샘플 애플리케이션이 표시됩니다.

단계, 작업, 파이프라인의 작동 방식에 대한 자세한 내용은 CodePipeline 개념 단원을 참조하십시오.

(선택 사항) 5단계: 파이프라인에 다른 단계 추가

이제 파이프라인의 다른 단계를 추가하여 CodeDeploy를 통해 스테이징 서버에서 프로덕션 서버로 배포합니다. 먼저 CodeDeploy의 CodePipelineDemoApplication에서 또 다른 배포 그룹을 만듭니 다. 그런 다음 이 배포 그룹을 사용하는 작업이 포함된 단계를 추가합니다. 다른 단계를 추가하려면 CodePipeline 콘솔 또는 AWS CLI 를 사용하여 JSON 파일에서 파이프라인의 구조를 검색하고 수동으 로 편집한 다음 update-pipeline 명령을 실행하여 파이프라인을 변경 사항으로 업데이트합니다.

주제

- CodeDeploy에서 두 번째 배포 그룹 생성
- 파이프라인의 다른 단계로 배포 그룹 추가

CodeDeploy에서 두 번째 배포 그룹 생성

Note

자습서의 이 부분에서는 두 번째 배포 그룹을 생성하지만 이전과 동일한 Amazon EC2 인스턴 스에 배포합니다. 이는 데모용일 뿐입니다. 오류가 CodePipeline에 표시되는 방식을 보여 주기 위해 고의로 실패하도록 설계되었습니다.

CodeDeploy에서 두 번째 배포 그룹을 생성하려면

- 1. <u>https://console.aws.amazon.com/codedeploy</u>에서 CodeDeploy 콘솔을 엽니다.
- 2. 애플리케이션을 선택한 다음 애플리케이션 목록에서 MyDemoApplication을 선택합니다.
- 3. 배포 그룹 탭을 선택한 다음 Create deployment group(배포 그룹 생성)을 선택합니다.
- 4. Create deployment group(배포 그룹 생성) 페이지의 Deployment group name(배포 그룹 이름)에 두 번째 배포 그룹 이름(예: CodePipelineProductionFleet)을 입력합니다.
- 5. 서비스 역할에서 초기 배포에 사용되는 동일한 CodeDeploy 서비스 역할(CodePipeline 서비스 역 할이 아님)을 선택합니다.
- 6. 배포 유형 아래에서 인 플레이스를 선택합니다.
- 7. [Environment configuration]에서 [Amazon EC2 Instances] 탭을 선택합니다. 키 상자에서 이름을 선택하고, 값 상자에서 목록의 MyCodePipelineDemo를 선택합니다. 배포 설정의 기본 구성을 그대로 둡니다.
- 8. Deployment configuration(배포 구성)에서 CodeDeployDefault.OneAtaTime을 선택합니다.
- 9. 로드밸런서에서 Enable load balancing(로드 밸런싱 활성화)을 선택 해제합니다.
- 10. [Create deployment group]을 선택합니다.

파이프라인의 다른 단계로 배포 그룹 추가

이제 다른 배포 그룹을 생성했으므로 이 배포 그룹을 사용하여 이전에 사용한 동일한 EC2 인스턴스에 배포하는 단계를 추가할 수 있습니다. CodePipeline 콘솔 또는를 사용하여이 단계를 추가할 수 AWS CLI 있습니다.

주제

- 세 번째 단계 생성(콘솔)
- 세 번째 단계 생성(CLI)

세 번째 단계 생성(콘솔)

CodePipeline 콘솔을 사용하여 새 배포 그룹을 사용하는 새 단계를 추가할 수 있습니다. 이 배포 그룹 이 이미 사용한 EC2 인스턴스에 배포되고 있으므로 이 단계의 배포 작업은 실패합니다.

- 1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> <u>codepipeline/home</u>://https//https//h
- 2. 이름에서 앞에서 만든 파이프라인의 이름(MyFirstPipeline)을 선택합니다.

- 3. 파이프라인 세부 정보 페이지에서 [Edit]를 선택합니다.
- 4. 편집 페이지에서 + Add stage(단계 추가)를 선택하여 배포 단계 바로 다음에 단계를 추가합니다.

eveloper Tools > CodePipelir	e > Pipelines > M	MyFirstPipeline > Edit MyFirstPipeline	
Editing: MyFirstP	ipeline		Delete Cancel Save
Edit: Source			Edit stage
Source	٤		
53			
		+ Add stage	
Edit: Deploy			Edit stage
Deploy	(i)		
CodeDeploy			د ا
		+ Add stage	

- 5. Add stage(단계 추가)의 Stage name(단계 이름)에 **Production**을 입력합니다. Add stage(단계 추가)를 선택합니다.
- 6. 새 단계에서 + Add action group(작업 그룹 추가)을 선택합니다.
- 7. 작업 편집의 작업 이름에서 **Deploy-Second-Deployment**를 입력합니다. 작업 공급자에서, 배 포 아래에 CodeDeploy를 선택합니다.
- CodeDeploy 섹션에서 애플리케이션 이름에 파이프라인을 생성했을 때처럼 드롭다운 목록에서 MyDemoApplication을 선택합니다. 배포 그룹에서 방금 생성한 배포 그룹 (CodePipelineProductionFleet)을 선택합니다. 입력 아티팩트에서 소스 작업에서 입력 아 티팩트를 선택합니다. 저장(Save)을 선택합니다.
- 9. 편집 페이지에서 저장을 선택합니다. 파이프라인 변경 사항 저장에서 저장을 선택합니다.
- 10. 새로운 단계가 파이프라인에 추가되었더라도 변경 사항에 의해 파이프라인이 다시 실행되도록 트 리거되지 않았으므로 상태가 [No executions yet]으로 표시됩니다. 편집된 파이프라인이 어떻게 실 행되는지를 보려면 수동으로 마지막 개정을 다시 실행해야 합니다. 파이프라인 세부 정보 페이지 에서 변경 사항 릴리스를 선택한 다음, 메시지가 표시되면 릴리스를 선택합니다. 이렇게 하면 소스 작업에 지정된 각 소스 위치에서 사용 가능한 가장 최근의 개정이 파이프라인을 통해 실행됩니다.

또는 AWS CLI 를 사용하여 파이프라인을 다시 실행하려면 로컬 Linux, macOS 또는 Unix 시스 템의 터미널 또는 로컬 Windows 시스템의 명령 프롬프트에서 파이프라인 이름을 지정하여 startpipeline-execution 명령을 실행합니다. 이렇게 하면 파이프라인을 통해 소스 버킷의 애플리케이션 이 두 번째로 실행됩니다.

aws codepipeline start-pipeline-execution --name MyFirstPipeline

이 명령은 pipelineExecutionId 객체를 반환합니다.

11. CodePipeline 콘솔로 돌아가 파이프라인 목록에서 MyFirstPipeline을 선택하여 보기 페이지를 엽 니다.

파이프라인에 세 단계 및 이 세 단계를 통해 실행 중인 아티팩트의 상태가 표시됩니다. 파이프라인 이 모든 단계를 통해 실행되려면 최대 5분이 소요될 수 있습니다. 앞에서와 마찬가지로 첫 번째 두 단계에는 배포 성공이 표시되지만, Production 단계에는 Deploy-Second-Deployment 작업 실패가 표시됩니다.

12. [Deploy-Second-Deployment] 작업에서 [Details]를 선택합니다. CodeDeploy 배포의 페이지로 리 디렉션됩니다. 이 경우, 모든 EC2 인스턴스에 배포되는 첫 번째 인스턴스 그룹의 결과가 실패이므 로 두 번째 배포 그룹에는 인스턴스가 없게 됩니다.

Note

이 경우의 실패는 의도된 것으로 파이프라인 단계에 실패가 있는 경우 어떻게 되는지를 설 명하기 위한 것입니다.

세 번째 단계 생성(CLI)

AWS CLI 를 사용하여 파이프라인에 스테이지를 추가하는 것이 콘솔을 사용하는 것보다 더 복잡하지 만 파이프라인의 구조에 대한 가시성이 향상됩니다.

파이프라인에 대한 3번째 단계를 생성하려면

 로컬 Linux, macOS 또는 Unix 머신에서 터미널 세션을 열거나 로컬 Windows 머신에서 명령 프롬프트를 열고, get-pipeline 명령을 실행하여 방금 생성한 파이프라인의 구조를 표시합니다.
 MyFirstPipeline의 경우, 다음 명령을 입력합니다.

aws codepipeline get-pipeline --name "MyFirstPipeline"

이 명령은 MyFirstPipeline의 구조를 반환합니다. 출력의 첫 번째 부분에 다음과 비슷한 내용이 표시됩니다.

```
{
    "pipeline": {
        "roleArn": "arn:aws:iam::80398EXAMPLE:role/AWS-CodePipeline-Service",
        "stages": [
        ...
```

출력의 마지막 부분에 파이프라인 메타데이터가 포함되고 다음과 비슷한 내용이 표시됩니다.

```
. . .
        ],
        "artifactStore": {
            "type": "S3"
            "location": "amzn-s3-demo-bucket",
        },
        "name": "MyFirstPipeline",
        "version": 4
    },
    "metadata": {
        "pipelineArn": "arn:aws:codepipeline:us-
east-2:80398EXAMPLE:MyFirstPipeline",
        "updated": 1501626591.112,
        "created": 1501626591.112
    }
}
```

 이 구조를 복사하여 평문 편집기에 붙여 넣은 다음 파일을 pipeline.json로 저장합니다. 편의 를 위해 이 파일을 aws codepipeline 명령을 실행하는 동일한 디렉터리에 저장합니다.

```
    Note
    다음과 같이 get-pipeline을 사용하여 JSON을 파일에 직접 파이프할 수 있습니다.
    aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
```

 배포 스테이지 섹션을 복사하고 이를 처음 두 단계 이후에 붙여 넣습니다. 배포 스테이지와 같은 배포 단계이므로 이를 세 번째 단계의 템플릿으로 사용할 것입니다. 4. 단계 이름 및 배포 그룹 세부 정보를 변경합니다.

다음 예에서는 배포 단계 후 pipeline.json 파일에 추가하는 JSON을 보여 줍니다. 강조 표시된 요 소를 새 값으로 편집하십시오. 배포 스테이지와 프로덕션 단계를 구분하기 위해 쉼표를 포함해야 한다는 것을 기억하십시오.

```
{
    "name": "Production",
     "actions": [
        {
         "inputArtifacts": [
             {
              "name": "MyApp"
             }
           ],
          "name": "Deploy-Second-Deployment",
          "actionTypeId": {
              "category": "Deploy",
              "owner": "AWS",
              "version": "1",
              "provider": "CodeDeploy"
              },
         "outputArtifacts": [],
         "configuration": {
              "ApplicationName": "CodePipelineDemoApplication",
              "DeploymentGroupName": "CodePipelineProductionFleet"
               },
         "runOrder": 1
        }
    ]
}
```

5. get-pipeline 명령을 사용하여 검색한 파이프라인 구조로 작업을 수행할 경우, JSON 파일에서 metadata 행을 제거해야 합니다. 이렇게 하지 않으면 update-pipeline 명령에서 사용할 수 없습니다. "metadata": { } 행과, "created", "pipelineARN" 및 "updated" 필드를 제거합니다.

예를 들어, 구조에서 다음 행을 삭제합니다.

```
"metadata": {
    "pipelineArn": "arn:aws:codepipeline:region:account-ID:pipeline-name",
    "created": "date",
```

```
"updated": "date"
}
```

파일을 저장합니다.

6. 다음과 유사하게 파이프라인 JSON 파일을 지정하여 update-pipeline 명령을 실행합니다.

```
aws codepipeline update-pipeline --cli-input-json file://pipeline.json
```

이 명령은 업데이트한 파이프라인의 전체 구조를 반환합니다.

\Lambda Important

파일 이름 앞에 file://를 포함해야 합니다. 이 명령에 필수적입니다.

 start-pipeline-execution 명령을 실행하여 파이프라인의 이름을 지정합니다. 이렇게 하면 파이프라 인을 통해 소스 버킷의 애플리케이션이 두 번째로 실행됩니다.

aws codepipeline start-pipeline-execution --name MyFirstPipeline

이 명령은 pipelineExecutionId 객체를 반환합니다.

8. CodePipeline 콘솔을 열고 파이프라인 목록에서 MyFirstPipeline을 선택합니다.

파이프라인에 세 단계 및 이 세 단계를 통해 실행 중인 아티팩트의 상태가 표시됩니다. 파이프라인 이 모든 단계를 통해 실행되려면 최대 5분이 소요될 수 있습니다. 앞에서와 마찬가지로 첫 번째 두 단계에는 배포 성공이 표시되지만, [Production] 단계에는 [Deploy-Second-Deployment] 작업이 실 패했다고 표시됩니다.

[Deploy-Second-Deployment] 작업에서 [Details]를 선택하여 실패 세부 정보를 확인합니다.
 CodeDeploy 배포의 세부 정보 페이지로 리디렉션됩니다. 이 경우, 모든 EC2 인스턴스에 배포되는 첫 번째 인스턴스 그룹의 결과가 실패이므로 두 번째 배포 그룹에는 인스턴스가 없게 됩니다.

Note

이 경우의 실패는 의도된 것으로 파이프라인 단계에 실패가 있는 경우 어떻게 되는지를 설 명하기 위한 것입니다.

(선택 사항) 6단계: CodePipeline에서 단계 간 전환 비활성화 및 활성화

파이프라인의 단계 간 전환을 활성화하거나 비활성화할 수 있습니다. 단계 간 전환을 비활성화하면 단 계 간 전환을 수동으로 제어할 수 있습니다. 예를 들어 파이프라인의 처음 두 단계를 실행할 수 있지만 프로덕션에 배포할 준비가 될 때까지 또는 해당 단계의 문제나 오류를 해결하는 동안 세 번째 단계로의 전환을 비활성화할 수 있습니다.

CodePipeline 파이프라인에서 단계 간 전환 비활성화 및 활성화 방법

- 1. CodePipeline 콘솔을 열고 파이프라인 목록에서 MyFirstPipeline을 선택합니다.
- 파이프라인의 세부 정보 페이지에서 이전 섹션(프로덕션)에서 추가한 두 번째 단계(배포)와 세 번 째 단계 사이의 전환 비활성화 버튼을 선택합니다.
- 3. 전환 비활성화에서 전환을 비활성화하는 사유를 입력한 후 비활성화를 선택합니다.

단계 사이에 있는 화살표가 아이콘을 표시하고 색상이 변경되며 전환 활성화 버튼이 표시됩니다.

Disable transition	×
You are about to disable the transition between "Source" and "Producti	ion".
Disabling reason Explain why you are disabling the transition.	
Disabling transition while I troubleshoot the failure	
Cancel	Disable

- 샘플을 S3 버킷에 다시 업로드합니다. 버킷에 버전이 지정되어 있는 경우, 이 변경 사항으로 인해 파이프라인이 시작됩니다.
- 파이프라인의 세부 정보 페이지로 돌아간 다음 단계의 상태를 확인합니다. 파이프라인 보기가 변 경되어 처음 두 단계의 진행 상황 및 성공이 표시되지만, 세 번째 단계에는 변경된 내용이 없습니 다. 이 프로세스에는 몇 분이 걸릴 수 있습니다.
- 두 단계 사이에서 전환 활성화 버튼을 선택하여 전환을 활성화합니다. [Enable transition] 대화 상 자에서 [Enable]을 선택합니다. 몇 분 후에 단계가 실행되기 시작하고 파이프라인의 처음 두 단계 를 통해 이미 실행된 아티팩트를 처리하려고 시도합니다.

Note

이 세 번째 단계가 성공하도록 하려면 전환을 활성화하기 전에 CodePipelineProductionFleet 배포 그룹을 편집하고 애플리케이션을 배포할 다양한 EC2 인스턴스를 지정합니다. 이 작업을 수행하는 방법에 대한 자세한 내용은 배포 그룹 설정 변경을 참조하십시오. 더 많은 EC2 인스턴스를 생성하면 추가 비용이 발생할 수 있습니다.

7단계: 리소스 정리

이 자습서에서 생성한 리소스 중 일부를 <u>자습서: 4단계 파이프라인 생성</u>에 사용할 수 있습니다. 예를 들 어 CodeDeploy 애플리케이션 및 배포를 재사용할 수 있습니다. 클라우드 기반의 완전 관리형 빌드 서 비스인 CodeBuild 등의 공급자를 통해 빌드 작업을 구성할 수 있습니다. Jenkins 등의 빌드 서버나 시 스템과 함께 공급자를 사용하는 빌드 작업을 구성할 수도 있습니다.

하지만 이 자습서와 다른 자습서를 완료한 후에는 사용한 파이프라인과 리소스를 삭제해야 이 리소스 를 계속 사용할 경우 부과되는 요금을 피할 수 있습니다. 먼저 파이프라인을 삭제하고 CodeDeploy 애 플리케이션과 그에 연결된 Amazon EC2 인스턴스, 끝으로 S3 버킷을 삭제합니다.

이 자습서에서 사용한 리소스를 정리하려면

- 1. CodePipeline 리소스를 정리하려면 <u>AWS CodePipeline에서 파이프라인 삭제</u> 단원의 지침을 따릅 니다.
- 2. CodeDeploy 리소스를 정리하려면 리소스를 정리하려면(콘솔)의 지침을 따릅니다.
- S3 버킷을 삭제하려면 <u>버킷 삭제 또는 비우기</u>의 지침을 따릅니다. 더 많은 파이프라인을 생성하려 는 경우, 파이프라인의 아티팩트를 저장하기 위해 생성한 S3 버킷을 삭제합니다. 이 버킷에 대한 자세한 내용은 <u>CodePipeline 개념</u> 단원을 참조하십시오.

자습서: 간단한 파이프라인 생성(CodeCommit 리포지토리)

이 자습서에서는 CodePipeline을 이용해 CodeCommit 리포지토리에서 유지되는 코드를 단일 Amazon EC2 인스턴스에 배포합니다. CodeCommit 리포지토리에 변경 사항을 푸시하면 파이프라인이 트리거 됩니다. 파이프라인은 CodeDeploy를 배포 서비스로 사용해 Amazon EC2 인스턴스에 변경 사항을 배 포합니다.

A Important

파이프라인 생성의 일환으로 CodePipeline은 고객이 제공하는 S3 아티팩트 버킷을 아티팩트 에 사용합니다. (이는 S3 소스 작업에 사용되는 버킷과 다릅니다.) S3 아티팩트 버킷이 파이프 라인의 계정과 다른 계정에 있는 경우 S3 아티팩트 버킷을 안전하고 신뢰할 수 AWS 계정 있는 가 소유해야 합니다.

사용자 가이드

파이프라인에는 두 단계가 있습니다.

- CodeCommit 소스 작업을 위한 소스 단계(소스)입니다.
- CodeDeploy 배포 작업을 위한 배포 단계(배포)입니다.

를 시작하는 가장 쉬운 방법은 CodePipeline 콘솔에서 파이프라인 생성 마법사를 AWS CodePipeline 사용하는 것입니다.

1 Note

시작하기 전에 CodeCommit으로 작업할 Git 클라이언트를 설정해야 합니다. 관련 지침은 CodeCommit 설정을 참조하세요.

1단계: CodeCommit 리포지토리 생성

먼저 CodeCommit에서 리포지토리를 생성합니다. 파이프라인이 실행되면 이 리포지토리에서 소스 코 드를 가져옵니다. CodeCommit 리포지토리에 푸시하기 전에 코드를 유지하고 업데이트할 로컬 리포지 토리를 생성할 수도 있습니다.

CodeCommit 리포지토리를 생성하려면

- 1. https://console.aws.amazon.com/codecommit/에서 CodeCommit 콘솔을 엽니다.
- 2. 리전 선택기에서 리포지토리와 파이프라인을 생성할 AWS 리전 를 선택합니다. 자세한 내용은 AWS 리전 및 엔드포인트를 참조하세요.
- 3. 리포지토리 페이지에서 리포지토리 생성을 선택합니다.
- 4. 리포지토리 생성 페이지에서 리포지토리 이름에 해당 리포지토리의 이름(예: MyDemoRepo)을 입 력합니다.
- 5. 생성(Create)을 선택합니다.

Note

이 자습서의 나머지 단계에서는 CodeCommit 리포지토리의 이름으로 MyDemoRepo를 사용합 니다. 다른 이름을 선택하는 경우 이 자습서 전체에서 이를 사용해야 합니다.

CodeCommit 리포지토리 생성

로컬 리포지토리를 설정하려면

이 단계에서는 원격 CodeCommit 리포지토리에 연결할 로컬 리포지토리를 설정합니다.

Note 로컬 리포지토리를 설정할 필요는 없습니다. <u>2단계: CodeCommit 리포지토리에 샘플 코드 추</u> 가에 설명된 대로 콘솔을 사용하여 파일을 업로드할 수도 있습니다.

- 1. 콘솔에서 새 리포지토리를 연 상태에서 페이지 오른쪽 상단의 URL 복제를 선택한 후 Clone SSH(SSH 복제)를 선택합니다. Git 리포지토리를 복제할 주소는 클립보드에 복사됩니다.
- 터미널 또는 명령줄에서 로컬 리포지토리를 저장하고 싶은 로컬 디렉터리로 이동합니다. 이 자습 서에서는 /tmp를 사용합니다.
- 다음 명령을 실행하여 리포지토리를 복제하여 SSH 주소를 앞 단계에서 복사한 주소로 교체합니다.
 다. 이 명령을 통해 MyDemoRepo라는 디렉터리가 생성됩니다.
 이 디렉터리에 샘플 애플리케이션을 복사합니다.

git clone ssh://git-codecommit.us-west-2.amazonaws.com/v1/repos/MyDemoRepo

2단계: CodeCommit 리포지토리에 샘플 코드 추가

이 단계에는 CodeDeploy 예제 연습을 위해 생성했던 샘플 애플리케이션용 코드를 다운로드하고 CodeCommit 리포지토리에 추가합니다.

- 1. 그런 다음 샘플을 다운로드하여 로컬 컴퓨터의 폴더나 디렉토리에 저장합니다.
 - a. 다음 중 하나를 선택합니다. Linux 인스턴스에 대해 이 자습서의 단계를 수행하려는 경우 SampleApp_Linux.zip을 선택합니다.
 - CodeDeploy를 사용하여 Amazon Linux 인스턴스에 배포하려는 경우 여기에서 샘플 애플 리케이션을 다운로드하세요. SampleApp_Linux.zip
 - CodeDeploy를 사용하여 Windows Server 인스턴스에 배포하려는 경우 여기에서 샘플 애플 리케이션을 다운로드하세요. <u>SampleApp_Windows.zip</u>

샘플 애플리케이션에는 CodeDeploy로 배포하기 위한 다음 파일이 포함되어 있습니다.

- appspec.yml 애플리케이션 사양 파일(AppSpec file)은 CodeDeploy에서 배포를 관리 하는 데 사용하는 <u>YAML</u> 형식의 파일입니다. AppSpec 파일에 대한 자세한 내용은 AWS CodeDeploy 사용 설명서의 CodeDeploy AppSpec 파일 참조를 참조하세요.
- index.html 인덱스 파일에는 배포된 샘플 애플리케이션의 홈 페이지가 포함되어 있습니다.
- LICENSE.txt 라이선스 파일에는 샘플 애플리케이션에 대한 라이선스 정보가 포함되어 있습니다.
- 스크립트용 파일 샘플 애플리케이션은 스크립트를 사용하여 인스턴스의 위치에 텍스트 파일을 씁니다. 여러 CodeDeploy의 배포 수명 주기 이벤트 각각에 대해 하나의 파일이 작 성됩니다.
 - (Linux 샘플만 해당) scripts 폴더 이 폴더에는 종속성을 설치하고 자동 배포 를 위한 샘플 애플리케이션을 시작 및 중지하기 위한 install_dependencies, start_server, stop_server와 같은 쉘 스크립트가 포함되어 있습니다.
 - (Windows 샘플만 해당) before-install.bat 이것은 BeforeInstall 배포 수명 주기 이벤트의 배치 스크립트로, 이 샘플의 이전 배포 중 기록된 이전 파일을 제거하기 위 해 실행되고 인스턴스에 새 파일을 기록할 위치를 만듭니다.
- b. 압축된 파일을 다운로드합니다.
- 2. 앞서 생성한 로컬 디렉터리에 <u>SampleApp_Linux.zip</u>에 있는 파일의 압축을 풉니다(예: /tmp/ MyDemoRepo 또는 c:\temp\MyDemoRepo).

파일을 로컬 리포지토리에 바로 배치해야 합니다. SampleApp_Linux 폴더를 포함하면 안 됩니 다. 로컬 Linux, macOS 또는 Unix 머신에 다음과 같이 디렉터리 및 파일 계층 구조가 나타나야 합 니다.

/tr	np			
	#	MyDemoRepo		
		#	appspec.yml	
		#	index.html	
		#	LICENSE.txt	
		#	scripts	
			<pre># install_dependencies</pre>	
			<pre># start_server</pre>	
			<pre># stop_server</pre>	

- 3. 리포지토리에 파일을 업로드하려면 다음 방법 중 하나를 사용합니다.
 - a. CodeCommit 콘솔을 사용하여 파일을 업로드하려면 다음과 같이 합니다.

- i. CodeCommit 콘솔을 열고 리포지토리 목록에서 해당 리포지토리를 선택합니다.
- ii. 파일 추가를 선택한 후 파일 업로드를 선택합니다.
- iii. 파일 선택을 선택한 다음 파일을 찾습니다. 폴더 아래에 파일을 추가하려면 파일 생성을 선택한 다음 폴더 이름을 파일 이름과 함께 입력합니다(예: scripts/ install_dependencies). 파일 내용을 새 파일에 붙여 넣습니다.

사용자 이름과 이메일 주소를 입력하여 변경 사항을 커밋합니다.

변경 사항 커밋을 선택합니다.

iv. 각 파일에 대해 이 단계를 반복합니다.

리포지토리 콘텐츠는 다음과 같아야 합니다.

#-- appspec.yml
#-- index.html
#-- LICENSE.txt
#-- scripts
 #-- install_dependencies
 #-- start_server
 #-- stop_server

- b. git 명령을 사용하여 파일을 업로드하려면
 - i. 디렉터리를 로컬 리포지토리로 변경합니다.

(For Linux, macOS, or Unix) cd /tmp/MyDemoRepo
(For Windows) cd c:\temp\MyDemoRepo

ii. 다음 명령을 실행하여 모든 파일을 한 번에 스테이징합니다.

```
git add -A
```

iii. 다음 명령을 실행하여 커밋 메시지와 함께 파일을 커밋합니다.

git commit -m "Add sample application files"

iv. 다음 명령을 실행하여 로컬 리포지토리의 파일을 CodeCommit 리포지토리에 푸시합니 다. git push

4. 다운로드하여 로컬 리포지토리에 추가한 파일이 이제 CodeCommit MyDemoRepo 리포지토리의 main 브랜치에 추가되었으며 파이프라인에 이를 포함할 수 있습니다.

3단계: Amazon EC2 Linux 인스턴스 생성 및 CodeDeploy 에이전트 설치

이 단계에서는 샘플 애플리케이션이 배포되는 Amazon EC2 인스턴스를 생성합니다. 이 프로세스의 일부로 인스턴스에 CodeDeploy 에이전트를 설치하고 관리할 수 있는 인스턴스 역할을 생성합니다. CodeDeploy 에이전트는 인스턴스를 CodeDeploy 배포에서 사용할 수 있게 해주는 소프트웨어 패키지 입니다. 또한 CodeDeploy 에이전트가 애플리케이션을 배포하는 데 사용하는 파일을 인스턴스가 가져 오도록 허용하고 SSM에서 인스턴스를 관리할 수 있도록 허용하는 정책을 첨부합니다.

인스턴스 역할을 생성하려면

- 1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
- 2. 콘솔 대시보드에서 역할을 선택합니다.
- 3. 역할 생성을 선택합니다.
- 신뢰할 수 있는 유형의 엔터티 선택에서 AWS 서비스를 선택합니다. 사용 사례 선택에서 EC2를 선택합니다. Select your use case(사용 사례 선택) 아래에서 EC2를 선택합니다. 다음: 권한을 선 택합니다.
- 5. 검색하여 AmazonEC2RoleforAWSCodeDeploy라는 정책을 선택합니다.
- 검색하여 AmazonSSMManagedInstanceCore라는 정책을 선택합니다. 다음: 태그를 선택합니다.
 다.
- 7. 다음: 검토를 선택합니다. 역할의 이름을 입력합니다(예: EC2InstanceRole).

Note

다음 단계를 위해 역할 이름을 적어 둡니다. 인스턴스를 생성할 때 이 역할을 선택합니다.

역할 생성을 선택합니다.

인스턴스 시작

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.

- 2. 측면 탐색에서 인스턴스를 선택하고 페이지 상단에서 인스턴스 시작을 선택합니다.
- 이름에 MyCodePipelineDemo를 입력합니다. 그러면 인스턴스에 Name의 태그 키와 MyCodePipelineDemo의 태그 값이 할당됩니다. 나중에 샘플 애플리케이션을 이 인스턴스에 배 포하는 CodeDeploy 애플리케이션을 생성합니다. CodeDeploy는 태그를 기반으로 배포할 인스턴 스를 선택합니다.
- 4. 애플리케이션 및 OS 이미지(Amazon 머신 이미지)에서 AWS 로고가 있는 Amazon Linux AMI 옵 션을 찾아 선택했는지 확인합니다. (이 AMI는 Amazon Linux 2 AMI(HVM)로 기술되며 "프리 티어 가능"이라는 레이블이 지정되어 있습니다.)
- 5. 인스턴스 유형에서 인스턴스의 하드웨어 구성으로 사용할 프리 티어 가능 t2.micro 유형을 선택 합니다.
- 6. 키 페어(로그인)에서 키 페어를 선택하거나 새로 생성합니다.

키 페어 없이 계속을 선택할 수도 있습니다.

i Note

이 자습서의 목적상 키 페어 없이 진행할 수 있습니다. SSH를 사용하여 인스턴스에 연결 하려면 키 페어를 생성하거나 사용합니다.

7. 네트워크 설정에서 다음을 수행합니다.

퍼블릭 IP 자동 할당에서 상태가 활성화인지 확인합니다.

생성된 보안 그룹의 경우 HTTP를 선택한 다음 소스 유형에서 내 IP를 선택합니다.

- 8. Advanced details(고급 세부 정보)를 확장합니다. IAM 인스턴스 프로파일에서 이전 절차에서 생성 한 IAM 역할을 선택합니다(예: EC2InstanceRole).
- 9. 요약에서 인스턴스 수에 1를 입력합니다.
- 10. 인스턴스 시작을 선택합니다.
- 11. [Instances] 페이지에서 시작 상태를 볼 수 있습니다. 인스턴스를 시작할 때 초기 상태는 pending입니다. 인스턴스가 시작된 후에는 상태가 running으로 바뀌고 퍼블릭 DNS 이름을 받 습니다. [Public DNS] 열이 표시되지 않으면 [Show/Hide] 아이콘을 선택하고 [Public DNS]를 선택 합니다.

4단계: CodeDeploy에서 애플리케이션 생성

CodeDeploy에서 <u>애플리케이션</u>은 배포하고자 하는 소프트웨어 애플리케이션이 포함된 리소스입니다. 나중에 이 애플리케이션을 CodePipeline과 함께 사용해 Amazon EC2 인스턴스에 대한 샘플 애플리케 이션 배포를 자동화할 수 있습니다.

먼저 CodeDeploy가 배포를 수행하도록 허용하는 역할을 생성합니다. 그런 다음 CodeDeploy 애플리 케이션을 만듭니다.

CodeDeploy 서비스 역할을 만들려면

- 1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
- 2. 콘솔 대시보드에서 역할을 선택합니다.
- 3. 역할 생성을 선택합니다.
- 4. 신뢰할 수 있는 엔터티 선택에서 AWS 서비스를 선택합니다. 사용 사례에서 CodeDeploy를 선택합니다. 나열된 옵션에서 CodeDeploy를 선택합니다. Next(다음)를 선택합니다. AWSCodeDeployRole 관리형 정책이 역할에 연결됩니다.
- 5. Next(다음)를 선택합니다.
- 6. 역할 이름(예: CodeDeployRole)을 입력한 후 Create role(역할 생성)을 선택합니다.

CodeDeploy에서 애플리케이션을 만들려면

- 1. https://console.aws.amazon.com/codedeploy에서 CodeDeploy 콘솔을 엽니다.
- 2. 애플리케이션 페이지가 나타나지 않으면 메뉴에서 애플리케이션을 선택합니다.
- 3. 애플리케이션 생성을 선택합니다.
- 4. 애플리케이션 이름에 MyDemoApplication을 입력합니다.
- 5. Compute Platform(컴퓨팅 플랫폼)에서 EC2/On-premises(EC2/온프레미스)를 선택합니다.
- 6. 애플리케이션 생성을 선택합니다.

CodeDeploy에서 배포 그룹을 생성하려면

배포 그룹은 배포할 인스턴스, 배포 속도와 같은 배포 관련 설정을 정의하는 리소스입니다.

- 1. 애플리케이션이 표시되는 페이지에서 Create deployment group(배포 그룹 생성)을 선택합니다.
- 2. Deployment group name(배포 그룹 이름)에 MyDemoDeploymentGroup을 입력합니다.

- 서비스 역할에서 앞서 생성한 서비스 역할의 ARN을 선택합니다(예: arn:aws:iam::account_ID:role/CodeDeployRole).
- 4. 배포 유형 아래에서 인 플레이스를 선택합니다.
- 5. [Environment configuration]에서 [Amazon EC2 Instances] 탭을 선택합니다. 키 필드에 Name을 입력합니다. 값 필드에 인스턴스에 태그를 지정하는 데 사용한 이름을 입력합니다(예: MyCodePipelineDemo).
- AWS Systems Manager를 사용한 에이전트 구성에서 지금을 선택하고 업데이트를 예약합니다. 그러면 인스턴스에 에이전트가 설치됩니다. Linux 인스턴스는 이미 SSM 에이전트로 구성되어 있 으며 이제 CodeDeploy 에이전트로 업데이트됩니다.
- 7. Deployment configuration(배포 구성)에서 CodeDeployDefault.OneAtaTime을 선택합니다.
- 로드 밸런서에서 로드 밸런싱 활성화가 선택되지 않았는지 확인합니다. 이 예에서는 로드 밸런서 를 설정하거나 대상 그룹을 선택할 필요가 없습니다.
- 9. [Create deployment group]을 선택합니다.

5단계: CodePipeline에서 첫 번째 파이프라인 생성

이제 첫 번째 파이프라인을 생성하고 실행할 준비가 되었습니다. 이 단계에서는 코드가 CodeCommit 리포지토리로 푸시될 때 자동으로 실행되는 파이프라인을 생성합니다.

CodePipeline 파이프라인을 생성하려면

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https//https//https://https//ht

https://console.aws.amazon.com/codepipeline/에서 CodePipeline 콘솔을 엽니다.

- 2. [Welcome] 페이지, [Getting started] 페이지 또는 [Pipelines] 페이지에서 Create pipeline(파이프라 인생성)을 선택합니다.
- 3. 1단계: 생성 옵션 선택 페이지의 생성 옵션에서 사용자 지정 파이프라인 빌드 옵션을 선택합니다. Next(다음)를 선택합니다.
- 4. 2단계: 파이프라인 설정 선택의 파이프라인 이름에 MyFirstPipeline을 입력합니다.
- CodePipeline은 특성과 가격이 다른 V1 및 V2 유형 파이프라인을 제공합니다. V2 유형은 콘 솔에서 선택할 수 있는 유일한 유형입니다. 자세한 내용은 <u>파이프라인 유형</u>을 참조하세요. CodePipeline에 대한 요금 정보는 <u>요금</u>을 참조하세요.
- 6. 서비스 역할에서 새 서비스 역할을 선택하여CodePipeline이 IAM에 서비스 역할을 생성하도록 허 용합니다.

- 7. [Advanced settings]의 설정은 기본값 그대로 두고, [Next]를 선택합니다.
- 3단계: 소스 스테이지 추가의 소스 공급자에서 CodeCommit을 선택합니다. 리포지토리 이름에서 <u>1단계: CodeCommit 리포지토리 생성</u>에서 생성한 CodeCommit 리포지토리의 이름을 선택합니다. 브랜치 이름에서 main를 선택하고 다음 단계를 선택합니다.

리포지토리 이름과 브랜치를 선택하면 이 파이프라인에 대해 생성될 Amazon CloudWatch Events 규칙을 보여 주는 메시지가 표시됩니다.

[Change detection options] 아래에서 기본값을 그대로 둡니다. 그러면 CodePipeline이 Amazon CloudWatch Events를 사용하여 소스 리포지토리의 변경을 감지할 수 있습니다.

Next(다음)를 선택합니다.

9. 4단계: 빌드 스테이지 추가에서 빌드 스테이지 건너뛰기를 선택하고 건너뛰기를 다시 선택하여 경 고 메시지를 수락합니다. Next(다음)를 선택합니다.

Note

이 자습서에서는 빌드 서비스가 필요 없는 코드를 배포하므로 이 단계는 건너뛸 수 있습 니다. 그러나 소스 코드를 인스턴스에 배포하기 전에 빌드해야 하는 경우 이 단계에서 CodeBuild를 구성할 수 있습니다.

10. 5단계: 테스트 단계 추가에서 테스트 단계 건너뛰기를 선택한 다음 건너뛰기를 다시 선택하여 경 고 메시지를 수락합니다.

Next(다음)를 선택합니다.

- 11. 6단계: 배포 단계 추가의 배포 공급자에서 CodeDeploy를 선택합니다. 애플리케이션 이름에서 MyDemoApplication을 선택합니다. 배포 그룹에서 MyDemoDeploymentGroup을 선택하고 다 음 단계를 선택합니다.
- 12. 7단계: 검토에서 정보를 검토한 다음 파이프라인 생성을 선택합니다.
- 13. 파이프라인은 생성된 후 실행되기 시작하여 CodeCommit 리포지토리에서 코드를 다운로드하고 EC2 인스턴스에 CodeDeploy 배포를 생성합니다. CodePipeline 샘플이 CodeDeploy 배포의 Amazon EC2 인스턴스에 웹 페이지를 배포하므로 진행 사항과 성공 및 실패 메시지를 볼 수 있습니다.

reloper Tools 👌 CodePipel	ine > Pipelines > AnvCo	ompanyPipeline
nyCompanyPip	oeline	Edit View history Release change
¢ Source		View current revisions
Source CodeCommit C In progress - Just now	٥	
Disable transition		
⊖ Deploy		View current revisions
Deploy CodeDeploy Didn't Run	٥	

축하합니다! 방금 CodePipeline에서 간단한 파이프라인을 생성했습니다.

다음으로, 결과를 확인합니다.

파이프라인이 성공적으로 실행되었는지 확인하려면

- 파이프라인의 초기 진행 상황을 확인합니다. 각 단계의 상태는 [No executions yet]에서 [In Progress]로 바뀌며, 다시 [Succeeded]나 [Failed] 중 하나로 바뀝니다. 파이프라인은 몇 분 내로 첫 번째 실행을 완료해야 합니다.
- 파이프라인 상태가 성공함으로 표시되면 배포 단계의 상태 영역에서 CodeDeploy를 선택합니다. 그러면 CodeDeploy 콘솔이 열립니다. Succeeded가 표시되지 않는 경우 <u>CodePipeline 문제 해</u> 결를 참조하십시오.
- 배포 탭에서 배포 ID를 선택합니다. 배포 페이지의 배포 수명 주기 이벤트에서 인스턴스 ID를 선택 합니다. EC2 콘솔이 열립니다.
- 설명 탭의 퍼블릭 DNS에서 주소(예: ec2-192-0-2-1.uswest-2.compute.amazonaws.com)를 복사하여 웹 브라우저의 주소 표시줄에 붙여 넣습니다.

웹페이지에 CodeCommit 리포지토리에 다운로드하고 푸시한 샘플 애플리케이션이 표시됩니다.

단계, 작업, 파이프라인의 작동 방식에 대한 자세한 내용은 CodePipeline 개념 단원을 참조하십시오.

6단계: CodeCommit 리포지토리의 코드 수정

파이프라인은 CodeCommit 리포지토리에 코드 변경이 이루어질 때마다 실행되도록 구성됩니다. 이 단 계에서는 CodeCommit 리포지토리의 샘플 CodeDeploy 애플리케이션의 일부인 HTML 파일을 변경합 니다. 이러한 변경 사항을 푸시하면 파이프라인이 다시 실행되고 변경 사항이 앞서 액세스한 웹 주소에 표시됩니다.

1. 디렉터리를 로컬 리포지토리로 변경합니다.

```
(For Linux, macOS, or Unix) cd /tmp/MyDemoRepo
(For Windows) cd c:\temp\MyDemoRepo
```

2. 텍스트 편집기를 사용하여 index.html 파일을 수정합니다.

```
(For Linux or Unix)gedit index.html
(For OS X)open -e index.html
(For Windows)notepad index.html
```

 index.html 파일의 콘텐츠를 수정하여 웹 페이지의 배경 색과 텍스트 일부를 변경한 다음, 파일 을 저장합니다.

```
<!DOCTYPE html>
<html>
<head>
 <title>Updated Sample Deployment</title>
  <stvle>
    body {
      color: #000000;
      background-color: #CCFFCC;
      font-family: Arial, sans-serif;
      font-size:14px;
    }
    h1 {
      font-size: 250%;
      font-weight: normal;
      margin-bottom: 0;
    }
   h2 {
```

```
font-size: 175%;
     font-weight: normal;
     margin-bottom: 0;
   }
 </style>
</head>
<body>
 <div align="center"><h1>Updated Sample Deployment</h1></div>
 <div align="center"><h2>This application was updated using CodePipeline,
CodeCommit, and CodeDeploy.</h2></div>
 <div align="center">
   Learn more:
   <a href="https://docs.aws.amazon.com/codepipeline/latest/</p>
userguide/">CodePipeline User Guide</a>
   <a href="https://docs.aws.amazon.com/codecommit/latest/</p>
userguide/">CodeCommit User Guide</a>
   <a href="https://docs.aws.amazon.com/codedeploy/latest/
userguide/">CodeDeploy User Guide</a>
 </div>
</body>
</html>
```

 다음 명령을 실행하여 변경 사항을 커밋하고 CodeCommit 리포지토리에 푸시하는 작업을 한 번에 수행합니다.

git commit -am "Updated sample application files"

git push

파이프라인이 성공적으로 실행되었는지 확인하려면

- 파이프라인의 초기 진행 상황을 확인합니다. 각 단계의 상태는 [No executions yet]에서 [In Progress]로 바뀌며, 다시 [Succeeded]나 [Failed] 중 하나로 바뀝니다. 파이프라인 실행은 몇 분 내 로 완료되어야 합니다.
- 작업 상태에 Succeeded가 표시된 후에는 브라우저에서 앞서 액세스한 데모 페이지를 새로 고침 하십시오.

업데이트된 웹 페이지가 표시됩니다.

7단계: 리소스 정리

이 설명서에 있는 다른 자습서를 위해 이 자습서에서 생성한 리소스 중 일부를 사용할 수 있습니다. 예 를 들어 CodeDeploy 애플리케이션 및 배포를 재사용할 수 있습니다. 하지만 이 자습서와 다른 자습 서를 완료한 후에는 사용한 파이프라인과 리소스를 삭제해야 이 리소스를 계속 사용할 경우 부과되 는 요금을 피할 수 있습니다. 먼저 파이프라인을 삭제하고 CodeDeploy 애플리케이션과 그에 연결된 Amazon EC2 인스턴스, 끝으로 CodeCommit 리포지토리를 삭제합니다.

이 자습서에서 사용한 리소스를 정리하려면

- 1. CodePipeline 리소스를 정리하려면 <u>AWS CodePipeline에서 파이프라인 삭제</u> 단원의 지침을 따릅 니다.
- 2. CodeDeploy 리소스를 정리하려면 배포 연습 리소스 정리 단원의 지침을 따릅니다.
- 3. CodeCommit 리포지토리를 삭제하려면 CodeCommit 리포지토리 삭제의 지침을 따릅니다.

8단계: 참조 자료

CodePipeline이 어떻게 작동하는지 자세히 알아보세요.

- 단계, 작업, 파이프라인의 작동 방식에 대한 자세한 내용은 CodePipeline 개념 단원을 참조하십시오.
- CodePipeline을 사용해 수행할 수 있는 작업에 대한 자세한 내용은 <u>CodePipeline 작업 유형과의 통</u> 합을 참조하세요.
- 좀 더 수준이 높은 자습서인 <u>자습서: 4단계 파이프라인 생성</u>를 사용해 보십시오. 이 자습서에서는 코 드를 배포 전에 빌드하는 단계가 포함된 다단계 파이프라인을 생성합니다.

자습서: 4단계 파이프라인 생성

이제 <u>자습서: 간단한 파이프라인 생성(S3 버킷)</u> 또는 <u>자습서: 간단한 파이프라인 생성(CodeCommit 리</u> <u>포지토리)</u>에서 첫 번째 파이프라인을 생성했으므로 더 복잡한 파이프라인 생성을 시작할 수 있습니다. 이 자습서에서는 소스에 대한 GitHub 리포지토리, 프로젝트를 빌드하기 위한 Jenkins 빌드 서버, 빌드 된 코드를 스테이징 서버에 배포하기 위한 CodeDeploy 애플리케이션을 사용하는 4단계 파이프라인의 생성에 대해 안내합니다. 다음 다이어그램은 초기 3단계 파이프라인을 보여줍니다.



파이프라인이 생성된 후 Jenkins도 사용하여 코드를 테스트하는 테스트 작업으로 단계를 추가하도록 이 파이프라인을 편집합니다.

이 파이프라인을 생성하기 전에 필요한 리소스를 구성해야 합니다. 예를 들어 소스 코드에 대한 GitHub 리포지토리를 사용할 경우 리포지토리를 파이프라인에 추가하기 전에 생성해야 합니다. 설정의 일부 로 이 자습서에서는 데모용으로 EC2 인스턴스에서 Jenkins의 설정에 대해 안내합니다.

A Important

이 절차에서 파이프라인에 추가하는 많은 작업에는 파이프라인을 생성하기 전에 생성해야 하 는 AWS 리소스가 포함됩니다. 소스 작업에 대한 AWS 리소스는 항상 파이프라인을 생성하는 리전과 동일한 AWS 리전에서 생성해야 합니다. 예를 들어, 미국 동부(오하이오) 리전에서 파 이프라인을 생성하는 경우, CodeCommit 리포지토리도 미국 동부(오하이오) 리전에 있어야 합 니다.

파이프라인을 생성할 때 교차 리전 작업을 추가할 수 있습니다. 교차 리전 작업에 대한 AWS 리소스는 작업을 실행하려는 리전과 동일한 AWS 리전에 있어야 합니다. 자세한 내용은 <u>CodePipeline에 교차 리전 작업 추가</u> 단원을 참조하십시오.

▲ Important

파이프라인 생성의 일환으로 CodePipeline은 고객이 제공하는 S3 아티팩트 버킷을 아티팩트 에 사용합니다. (이는 S3 소스 작업에 사용되는 버킷과 다릅니다.) S3 아티팩트 버킷이 파이프 라인의 계정과 다른 계정에 있는 경우 S3 아티팩트 버킷을 안전하고 신뢰할 수 AWS 계정 있는 에서 소유해야 합니다.

이 자습서를 시작하기 전에 CodePipeline 시작하기의 일반적인 사전 조건을 이미 완료했어야 합니다.

주제

- 1단계: 사전 조건 완료
- 2단계: CodePipeline에서 파이프라인 생성
- 3단계: 파이프라인에 다른 단계 추가
- 4단계: 리소스 정리

1단계: 사전 조건 완료

를 Jenkins와 통합하려면 CodePipeline과 함께 사용하려는 모든 Jenkins 인스턴스에 Jenkins 용 CodePipeline 플러그인을 설치해야 AWS CodePipeline 합니다. 또한 Jenkins 프로젝트와 CodePipeline 간의 권한을 위해 사용하도록 전용 IAM 사용자 또는 역할을 구성해야 합니다. Jenkins와 CodePipeline을 통합하는 가장 간단한 방법은 Jenkins 통합을 위해 생성한 IAM 인스턴스 역할을 사용 하는 EC2 인스턴스에 Jenkins를 설치하는 것입니다. Jenkins 작업에 대한 파이프라인의 링크가 성공 적으로 연결되려면 Jenkins 프로젝트에서 사용된 포트에 대한 인바운드 연결을 허용하도록 서버 또는 EC2 인스턴스에 프록시 및 방화벽 설정을 구성해야 합니다. 해당 포트(예: HTTPS 연결만 사용하기 위 해 Jenkins를 보호한 경우 443 및 8443 또는 HTTP 연결을 허용한 경우 80 및 8080)에 대한 연결을 허 용하기 전에 액세스 제어를 적용하고 사용자를 인증하도록 Jenkins를 구성했는지 확인합니다. 자세한 내용은 Jenkins 보호를 참조하십시오.

Note

이 자습서에서는 코드 샘플을 사용하고 샘플을 Haml에서 HTML로 변환하는 빌드 단계를 구성 합니다. <u>GitHub 리포지토리에 샘플 복사 또는 복제</u>의 단계에 따라 GitHub 리포지토리에서 오 픈 소스 샘플 코드를 다운로드할 수 있습니다. .zip 파일뿐만 아니라 GitHub 리포지토리의 전체 샘플도 필요합니다.

이 자습서는 다음 사항도 가정합니다.
- Jenkins를 설치 및 관리하고 Jenkins 프로젝트를 생성하는 데 익숙합니다.
- Jenkins 프로젝트를 호스팅하는 동일한 컴퓨터나 인스턴스에 Rake 및 Haml gem for Ruby 를 설치했습니다.
- Rake 명령을 터미널이나 명령줄에서 실행할 수 있도록 필요한 시스템 환경 변수를 설정했습 니다(예: Windows 시스템의 경우 Rake를 설치한 디렉터리를 포함하도록 PATH 변수 수정).

주제

- GitHub 리포지토리에 샘플 복사 또는 복제
- Jenkins 통합에 사용할 IAM 역할 생성
- Jenkins 및 Jenkins용 CodePipeline 플러그인 설치 및 구성

GitHub 리포지토리에 샘플 복사 또는 복제

샘플을 복제하고 GitHub 리포지토리에 푸시하려면

- GitHub 리포지토리로부터 샘플 코드를 다운로드하거나 로컬 컴퓨터에 리포지토리를 복제합니다.
 다음과 같은 두 가지 샘플 패키지가 있습니다.
 - 샘플을 Amazon Linux, RHEL 또는 Ubuntu Server 인스턴스에 배포하려는 경우 <u>codepipeline-jenkins-aws-codedeploy_linux.zip</u>를 선택하세요.
 - 샘플을 Windows Server 인스턴스에 배포하려는 경우 <u>CodePipeline-Jenkins-</u> AWSCodeDeploy_Windows.zip를 선택하세요.
- 2. 리포지토리에서, [Fork]를 선택하여 샘플 리포지토리를 Github 계정의 리포지토리에 복제합니다. 자세한 내용은 GitHub 문서를 참조하십시오.

Jenkins 통합에 사용할 IAM 역할 생성

EC2 인스턴스를 시작하여 Jenkins 서버를 호스팅하고 IAM 역할을 사용하여 CodePipeline과 상호 작 용하는 데 필요한 권한을 인스턴스에 부여하는 것이 좋습니다.

- 1. 에 로그인 AWS Management Console 하고 <u>https://console.aws.amazon.com/iam/</u>://https://https:// https://https://://https://://https://://https://://https://://
- 2. IAM 콘솔의 탐색 창에서 역할을 선택한 후 역할 생성을 선택합니다.

- 3. 신뢰할 수 있는 유형의 엔터티 선택에서 AWS 서비스(을)를 선택합니다. Choose the service that will use this role(이 역할을 사용할 서비스 선택) 아래에서 EC2를 선택합니다. Select your use case(사용 사례 선택) 아래에서 EC2를 선택합니다.
- 4. 다음: 권한을 선택합니다. Attach permissions policies(권한 정책 연결) 페이지에서 AWSCodePipelineCustomActionAccess 관리형 정책을 선택한 다음, Next: Tags(다음: 태 그)를 선택합니다. 다음: 검토를 선택합니다.
- 검토 페이지의 역할 이름에 Jenkins 통합을 위해 특별히 생성할 역할 이름(예: JenkinsAccess) 을 입력한 다음, 역할 생성을 선택합니다.

Jenkins를 설치하는 EC2 인스턴스를 생성하면 3단계: 인스턴스 세부 정보 구성에서 인스턴스 역할(예: *JenkinsAccess*)을 선택해야 합니다.

인스턴스 역할 및 Amazon EC2에 대한 자세한 내용은 <u>Amazon EC2의 IAM 역할, IAM 역할을 사용하여</u> <u>Amazon EC2 인스턴스에서 실행 중인 애플리케이션에 대한 권한 부여,</u> <u>AWS 서비스에 대한 권한을 위</u> 임할 역할 생성을 참조하세요.

Jenkins 및 Jenkins용 CodePipeline 플러그인 설치 및 구성

Jenkins 및 Jenkins용 CodePipeline 플러그인을 설치하려면

 Jenkins를 설치할 EC2 인스턴스를 생성하고 3단계: 인스턴스 세부 정보 구성에서 방금 생성한 인 스턴스 역할(예: JenkinsAccess)을 선택해야 합니다. EC2 인스턴스 생성에 대한 자세한 내용은 Amazon EC2 사용 설명서의 Amazon EC2 인스턴스 시작하기를 참조하세요.

Note

사용할 Jenkins 리소스가 이미 있는 경우, 이 리소스를 사용할 수 있지만, 특정 IAM 사용자 를 생성하고, AWSCodePipelineCustomActionAccess 관리형 정책을 이 사용자에게 적용한 다음, Jenkins 리소스의 해당 사용자에 대한 액세스 자격 증명을 구성하고 사용해 야 합니다. Jenkins UI를 사용하여 자격 증명을 공급하려면 HTTPS만 허용하도록 Jenkins 를 구성합니다. 자세한 내용은 <u>CodePipeline 문제 해결</u> 단원을 참조하십시오.

- 2. EC2 인스턴스에 Jenkins를 설치합니다. 자세한 내용은 <u>Jenkins 설치</u> 및 <u>Jenkins 시작 및 액세스</u>와 <u>CodePipeline과 제품 및 서비스 통합</u>의 <u>details of integration with Jenkins</u>에 대한 Jenkins 설명서 를 참조하십시오.
- 3. Jenkins를 시작하고 홈 페이지에서 [Manage Jenkins]를 선택합니다.
- 4. [Manage Jenkins] 페이지에서 [Manage Plugins]를 선택합니다.

- 5. [Available] 탭을 선택하고, [Filter] 검색 상자에서, AWS CodePipeline을 입력합니다. 목록에서 Jenkins용 CodePipeline 플러그인을 선택하고 지금 다운로드 및 다시 시작 후 설치를 선택합니다.
- 6. [Installing Plugins/Upgrades] 페이지에서 [Restart Jenkins when installation is complete and no jobs are running]을 선택합니다.
- 7. [Back to Dashboard]를 선택합니다.
- 8. 기본 페이지에서 [New Item]을 선택합니다.
- 9. [Item Name]에 Jenkins 프로젝트의 이름을 입력합니다(예: *MyDemoProject*). [Freestyle project] 를 선택한 다음, [OK]를 선택합니다.

Note

프로젝트 이름이 CodePipeline의 요구 사항을 충족해야 합니다. 자세한 내용은 <u>in AWS</u> <u>CodePipeline 할당량</u> 단원을 참조하십시오.

- 10. 프로젝트의 구성 페이지에서 [Execute concurrent builds if necessary] 확인란을 선택합니다. [Source Code Management]에서, [AWS CodePipeline]를 선택합니다. EC2 인스턴스에 Jenkins를 설치하고 CodePipeline과 Jenkins 간의 통합을 위해 생성한 IAM 사용자의 프로필 AWS CLI 로를 구성한 경우 다른 모든 필드는 비워 둡니다.
- 11. 고급을 선택하고, 공급자에서 CodePipeline에 표시될 작업의 공급자 이름(예: MyJenkinsProviderName)을 입력합니다. 이 이름은 고유해야 하며 기억하기 쉬워야 합니다. 이 자습서 후반부에 파이프라인에 빌드 작업을 추가할 때 및 테스트 작업을 추가할 때 이 이름을 사용 할 것입니다.

Note

이 작업 이름은 CodePipeline의 작업에 대한 명명 요구사항을 만족해야 합니다. 자세한 내 용은 <u>in AWS CodePipeline 할당량</u> 단원을 참조하십시오.

12. [Build Triggers]에서 모든 상자의 선택을 취소한 다음 [Poll SCM]을 선택합니다. [Schedule]에서 다 음과 같이 5개의 별표를 공백으로 구분하여 입력합니다.

* * * * *

이렇게 하면 1분마다 CodePipeline이 폴링됩니다.

13. [Build]에서 [Add build step]을 선택합니다. 쉘 실행(Amazon Linux, RHEL 또는 Ubuntu Server) 배 치 명령 실행(Windows Server)을 선택하고 다음을 입력합니다.

rake

Note

rake를 실행하는 데 필요한 변수 및 설정을 사용하여 환경을 구성해야 합니다. 그렇지 않으면 빌드가 실패합니다.

- 14. 빌드 후 작업 추가를 선택한 다음, AWS CodePipeline 게시자를 선택합니다. [Add]를 선택한 다음, [Build Output Locations]에서 위치를 비워 둡니다. 이 구성이 기본 구성입니다. 이렇게 하면 빌드 프로세스 마지막에 압축된 파일이 생성됩니다.
- 15. [Save]를 선택하여 Jenkins 프로젝트를 저장합니다.

2단계: CodePipeline에서 파이프라인 생성

자습서의 이 부분에서는 Create Pipeline 마법사를 사용하여 파이프라인을 생성합니다.

CodePipeline 자동화 릴리스 프로세스 생성 방법

- 1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> <u>codepipeline/home://https://https://https://https://i/httpsCodePipeline.com.com.com.com.</u>
- 필요하면, 리전 선택기를 사용하여 파이프라인 리소스가 위치하는 리전으로 리전을 변경합니다.
 예를 들어 이전 자습서에서 us-east-2에 리소스를 만든 경우 리전 선택기가 미국 동부(오하이 오)로 설정되어야 합니다.

CodePipeline에 사용할 수 있는 리전과 엔드포인트에 대한 자세한 내용은 <u>AWS CodePipeline 엔</u> 드포인트 및 할당량을 참조하세요.

- 3. [Welcome] 페이지, [Getting started] 페이지 또는 [Pipelines] 페이지에서 Create pipeline(파이프라 인생성)을 선택합니다.
- 4. 1단계: 생성 옵션 선택 페이지의 생성 옵션에서 사용자 지정 파이프라인 빌드 옵션을 선택합니다. Next(다음)를 선택합니다.
- 5. 2단계: 파이프라인 설정 선택 페이지의 파이프라인 이름에 파이프라인 이름을 입력합니다.
- CodePipeline은 특성과 가격이 다른 V1 및 V2 유형 파이프라인을 제공합니다. V2 유형은 콘 솔에서 선택할 수 있는 유일한 유형입니다. 자세한 내용은 <u>파이프라인 유형</u>을 참조하세요. CodePipeline에 대한 요금 정보는 <u>요금</u>을 참조하세요.

- 7. 서비스 역할에서 새 서비스 역할을 선택하여CodePipeline이 IAM에 서비스 역할을 생성하도록 허 용합니다.
- 8. [Advanced settings]의 설정은 기본값 그대로 두고, [Next]를 선택합니다.
- 9. 3단계: 소스 스테이지 추가 페이지의 소스 공급자에서 GitHub를 선택합니다.
- 10. 연결에서 기존 연결을 선택하거나 새로 생성합니다. GitHub 소스 작업에 대한 연결을 만들거나 관 리하려면 <u>GitHub 연결</u>을 참조하세요.
- 11. 4단계: 빌드 스테이지 추가에서 Jenkins 추가를 선택합니다. 공급자 이름에서 Jenkins용 CodePipeline 플러그인에 제공한 작업 이름(예: *MyJenkinsProviderName*)을 입력합니다. 이 이름은 Jenkins용 CodePipeline 플러그인의 이름과 정확하게 일치해야 합니다. [Server URL]에서 Jenkins가 설치된 EC2 인스턴스의 URL을 입력합니다. 프로젝트 이름에서, Jenkins에서 생성한 프로젝트 이름(예: *MyDemoProject*)을 입력한 다음, 다음을 선택합니다.
- 12. 5단계: 테스트 단계 추가에서 테스트 단계 건너뛰기를 선택한 다음 건너뛰기를 다시 선택하여 경 고 메시지를 수락합니다.

Next(다음)를 선택합니다.

13. 6단계: 배포 단계 추가에서에서 생성한 CodeDeploy 애플리케이션 및 배포 그룹을 재사용합니 다<u>자습서: 간단한 파이프라인 생성(S3 버킷)</u>. 배포 공급자에서 CodeDeploy를 선택합니다. 애플리 케이션 이름에 CodePipelineDemoApplication을 입력하거나 새로 고침 버튼을 선택한 다음, 목록에서 애플리케이션 이름을 선택합니다. 배포 그룹에서 CodePipelineDemoFleet을 입력하 거나, 목록에서 이를 선택한 후 다음을 선택합니다.

Note

사용자 고유의 CodeDeploy 리소스를 사용하거나 새로운 리소스를 만들 수 있지만, 추가 비용이 발생할 수 있습니다.

- 14. 7단계: 검토에서 정보를 검토한 다음 파이프라인 생성을 선택합니다.
- 15. 파이프라인이 자동으로 시작되고 파이프라인을 통해 샘플이 실행됩니다. 파이프라인이 Haml 샘 플을 HTML에 빌드하고 이를 CodeDeploy 배포의 각 Amazon EC2 인스턴스에 웹 페이지로 배포 하므로 진행 사항과 성공 및 실패 메시지를 볼 수 있습니다.

3단계: 파이프라인에 다른 단계 추가

이제 테스트 단계를 추가한 다음 샘플에 포함된 Jenkins 테스트를 사용하는 해당 단계에 테스트 작업을 추가하여 웹 페이지에 콘텐츠가 있는지 여부를 확인합니다. 이 테스트는 데모용일 뿐입니다.

Note

파이프라인에 다른 단계를 추가하지 않은 경우 배포 작업의 이전이나 이후에 파이프라인의 스 테이징 단계에 테스트 작업을 추가할 수 있습니다.

파이프라인에 테스트 단계 추가

주제

- 인스턴스의 IP 주소 조회
- 배포 테스트를 위해 Jenkins 프로젝트 생성
- 네 번째 단계 생성

인스턴스의 IP 주소 조회

코드를 배포한 인스턴스의 IP 주소를 확인하려면

- 1. 파이프라인 상태가 [Succeeded]로 표시되면 스테이징 단계의 상태 영역에서 [Details]를 선택합니 다.
- 2. [Deployment Details] 섹션의 [Instance ID]에서 성공적으로 배포한 인스턴스 중 하나의 인스턴스 ID를 선택합니다.
- 3. 인스턴스의 IP 주소를 복사합니다(예: 192.168.0.4). Jenkins 테스트에서 이 IP 주소를 사용할 것입니다.

배포 테스트를 위해 Jenkins 프로젝트 생성

Jenkins 프로젝트를 만들려면

- 1. Jenkins를 설치한 인스턴스에서, Jenkins를 열고 기본 페이지에서 [New Item]을 선택합니다.
- [Item Name]에 Jenkins 프로젝트의 이름을 입력합니다(예: *MyTestProject*). [Freestyle project] 를 선택한 다음, [OK]를 선택합니다.

1 Note

프로젝트 이름이 CodePipeline 요구 사항을 충족해야 합니다. 자세한 내용은 <u>in AWS</u> <u>CodePipeline 할당량</u> 단원을 참조하십시오. 3. 프로젝트의 구성 페이지에서 [Execute concurrent builds if necessary] 확인란을 선택합니다. [Source Code Management]에서, [AWS CodePipeline]를 선택합니다. EC2 인스턴스에 Jenkins를 설치하고 CodePipeline과 Jenkins 간의 통합을 위해 생성한 IAM 사용자의 프로필 AWS CLI 로를 구성한 경우 다른 모든 필드는 비워 둡니다.

A Important

Jenkins 프로젝트를 구성하는 중이고 Amazon EC2 인스턴스에 설치되어 있지 않거나, Windows 운영 체제를 실행 중인 EC2 인스턴스에 설치되어 있는 경우, 프록시 호스트 및 포트 설정에서 요구하는 대로 필드를 채우고, CodePipeline과 Jenkins 간 통합용으로 구성 한 IAM 사용자 또는 역할의 자격 증명을 제공합니다.

- 4. [Advanced]를 선택하고, [Category]에서 [Test]를 선택합니다.
- 5. [Provider]에서, 빌드 프로젝트에 사용한 동일한 이름(예: *MyJenkinsProviderName*)을 입력합 니다. 이 이름은 이 자습서의 후반부에서 파이프라인에 테스트 작업을 추가할 때 사용됩니다.

1 Note

이 이름은 작업에 대한 CodePipeline 명명 요구사항을 만족해야 합니다. 자세한 내용은 in <u>AWS CodePipeline 할당량</u> 단원을 참조하십시오.

6. [Build Triggers]에서 모든 상자의 선택을 취소한 다음 [Poll SCM]을 선택합니다. [Schedule]에서 다 음과 같이 5개의 별표를 공백으로 구분하여 입력합니다.

* * * * *

이렇게 하면 1분마다 CodePipeline이 폴링됩니다.

7. [Build]에서 [Add build step]을 선택합니다. Amazon Linux, RHEL 또는 Ubuntu Server 인스턴스에 배포하는 경우 쉘 실행을 선택합니다. 그런 다음 다음을 입력합니다. 여기서 IP 주소는 이전에 복 사한 EC2 인스턴스의 주소입니다.

TEST_IP_ADDRESS=192.168.0.4 rake test

Windows Server 인스턴스를 배포하는 경우 배치 명령 실행을 선택한 후 다음을 입력합니다. 여기 서, IP 주소는 앞에서 복사한 EC2 인스턴스의 주소입니다.

set TEST_IP_ADDRESS=192.168.0.4 rake test

Note

테스트에서는 포트 번호를 기본값인 80으로 간주합니다. 다른 포트를 지정하려면 다음과 같이 테스트 포트 설명문을 추가합니다.

TEST_IP_ADDRESS=192.168.0.4 TEST_PORT=8000 rake test

- 빌드 후 작업 추가를 선택한 다음, AWS CodePipeline 게시자를 선택합니다. [Add]는 선택하지 마 십시오.
- 9. [Save]를 선택하여 Jenkins 프로젝트를 저장합니다.

네 번째 단계 생성

Jenkins 테스트 작업을 포함하는 파이프라인에 단계를 추가하려면

- 1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> <u>codepipeline/home</u>://https//https//
- 2. 이름에서 앞에서 만든 파이프라인의 이름(MySecondPipeline)을 선택합니다.
- 3. 파이프라인 세부 정보 페이지에서 [Edit]를 선택합니다.
- 4. [Edit] 페이지에서 [+ Stage]를 선택하여 빌드 단계 바로 다음에 단계를 추가합니다.
- 5. 새 단계의 이름 필드에 이름(예: **Testing**)을 입력한 다음, + Add action group(+ 작업 그룹 추가)을 선택합니다.
- [Action name]에 MyJenkinsTest-Action을 입력합니다. [Test provider]에서, Jenkins에 지 정한 공급자 이름(예: MyJenkinsProviderName)을 선택합니다. [Project name]에 Jenkins에 서 생성한 프로젝트 이름(예: MyTestProject)을 입력합니다. 입력 아티팩트에서 기본 이름이 BuildArtifact인 Jenkins 빌드로부터 아티팩트를 선택한 후 완료를 선택합니다.

1 Note

Jenkins 테스트 작업은 Jenkins 빌드 단계에서 빌드된 애플리케이션에서 실행되므로 이 테스트 작업에 대한 입력 아티팩트에 빌드 아티팩트를 사용하십시오.

입력 및 출력 아티팩트 및 파이프라인의 구조에 대한 자세한 내용은 <u>CodePipeline 파이프라인 구</u> 조 참조 단원을 참조하십시오.

- 7. [Edit] 페이지에서 [Save pipeline changes]를 선택합니다. [Save pipeline changes] 대화 상자에서 [Save and continue]를 선택합니다.
- 새로운 단계가 파이프라인에 추가되었더라도 변경 사항에 의해 파이프라인이 다시 실행되도록 트 리거되지 않았으므로 해당 단계의 상태가 [No executions yet]으로 표시됩니다. 개정된 파이프라인 을 통해 샘플을 실행하려면 파이프라인 세부 정보 페이지에서 변경 사항 릴리스를 선택합니다.

파이프라인 보기에 파이프라인의 상태 및 작업과 해당 네 단계를 통해 실행되는 개정 상태가 표시 됩니다. 파이프라인이 모든 단계를 통해 실행되는 데 걸리는 시간은 아티팩트의 크기, 빌드 및 테 스트 작업의 복잡성, 그리고 기타 요인에 따라 달라집니다.

4단계: 리소스 정리

이 자습서를 완료한 후에는 사용한 파이프라인과 리소스를 삭제해야 이 리소스를 계속 사용할 경우 부과되는 요금을 피할 수 있습니다. CodePipeline을 계속 사용하지 않을 경우 파이프라인을 삭제하고 CodeDeploy 애플리케이션과 그에 연결된 Amazon EC2 인스턴스, 끝으로 아티팩트를 저장하는 데 사 용된 Amazon S3 버킷을 삭제합니다. 다른 리소스(예: GitHub 리포지토리)를 계속 사용하지 않을 경우 이 리소스를 삭제할지 여부도 고려해야 합니다.

이 자습서에서 사용한 리소스를 정리하려면

 로컬 Linux, macOS 또는 Unix 머신에서 터미널 세션을 열거나 로컬 Windows 머신에서 명 령 프롬프트를 열고, delete-pipeline 명령을 실행하여 생성한 파이프라인을 삭제합니다. MySecondPipeline의 경우, 다음 명령을 입력합니다.

aws codepipeline delete-pipeline --name "MySecondPipeline"

이 명령은 아무 것도 반환하지 않습니다.

- 2. CodeDeploy 리소스를 정리하려면 정리 단원의 지침을 따릅니다.
- 인스턴스 리소스를 정리하려면 Jenkins를 설치한 EC2 인스턴스를 삭제합니다. 자세한 내용은 <u>인</u>
 <u>스턴스 정리</u>를 참조하십시오.
- 더 많은 파이프라인을 생성하거나 CodePipeline을 다시 사용하려는 경우, 파이프라인의 아티팩트 를 저장하는 데 사용한 Amazon S3 버킷을 삭제합니다. 버킷을 삭제하려면 <u>버킷 삭제</u>의 지침을 따 릅니다.
- 이 파이프라인에 다른 리소스를 다시 사용하지 않으려는 경우, 해당 특정 리소스에 대한 지침을 따라 이를 삭제하는 것을 고려하십시오. 예를 들어, GitHub 리포지토리를 삭제하려는 경우 GitHub 웹 사이트에서 리포지토리 삭제의 지침을 따릅니다.

자습서: 파이프라인 상태 변경에 대한 이메일 알림을 수신하도록 CloudWatch Events 규칙 설정

에서 파이프라인을 설정한 후 파이프라인의 실행 상태가 변경될 때마다 또는 파이프라인의 단계 또 는 작업이 변경될 때마다 알림을 보내도록 CloudWatch Events 규칙을 설정할 AWS CodePipeline 수 있습니다. CloudWatch Events를 사용하여 파이프라인 상태 변경에 대한 알림을 설정하는 방법은 CodePipeline 이벤트 모니터링 단원을 참조하세요.

이 자습서에서는 파이프라인의 상태가 FAILED로 변경되면 이메일을 보내도록 알림을 구성합니다. 이 자습서에서는 CloudWatch Events 규칙을 만들 때 입력 변환기 방법을 사용합니다. 이는 사람이 읽을 수 있는 텍스트로 메시지를 전달하도록 메시지 스키마 세부 정보를 변환합니다.

Note

Amazon SNS 알림 및 CloudWatch Events 규칙과 같은이 자습서의 리소스를 생성할 때 리소 스가 파이프라인과 동일한 AWS 리전에 생성되었는지 확인합니다.

주제

- 1단계: Amazon SNS를 사용하여 이메일 알림 설정
- 2단계: 규칙을 만들고 SNS 주제를 대상으로 추가
- <u>3단계: 리소스 정리</u>

1단계: Amazon SNS를 사용하여 이메일 알림 설정

Amazon SNS는 주제 사용을 조정하여 구독 엔드포인트 또는 클라이언트에 메시지를 전달합니다. Amazon SNS를 사용하여 알림 주제를 만든 후 이메일 주소를 사용하여 해당 주제를 구독합니다. Amazon SNS 주제가 CloudWatch Events 규칙에 대상으로 추가됩니다. 자세한 내용은 <u>Amazon</u> Simple Notification Service Developer Guide를 참조하십시오.

Amazon SNS 주제를 생성하거나 식별합니다. CodePipeline은 CloudWatch Events를 사용하여 Amazon SNS를 통해 이 주제에 대한 알림을 보냅니다. 주제를 생성하려면 다음과 같이 합니다.

- 1. Amazon SNS 콘솔을 <u>https://console.aws.amazon.com/sns</u>://https://https://https://https://https://https://https://
- 2. 주제 생성을 선택합니다.

 새로운 주제 생성 대화 상자의 주제 이름에 주제 이름(예: PipelineNotificationTopic)을 입 력합니다.

opic name will be used to	o create a permanent unique identifier called an Amazon Resource Name (ARN).	
Topic name	PipelineNotificationTopic	0
Display name	Enter topic display name. Required for topics with SMS subscriptions.	0

4. 주제 생성을 선택합니다.

자세한 내용은 Amazon SNS 개발자 안내서의 주제 생성을 참조하세요.

한 명 이상의 수신자가 주제를 구독하여 이메일 알림을 수신하게 합니다. 수신자가 주제를 구독하게 하 려면 다음과 같이 합니다.

- 1. Amazon SNS 콘솔의 주제 목록에서 새 주제 옆의 확인란을 선택합니다. [Actions, Subscribe to topic]을 선택합니다.
- 2. [Create subscription] 대화 상자에서 ARN이 [Topic ARN]에 표시되는지 확인합니다.
- 3. 프로토콜에서 이메일을 선택합니다.
- 4. 엔드포인트에 수신자의 전체 이메일 주소를 입력합니다.
- 5. 구독 생성을 선택합니다.
- Amazon SNS가 수신자에게 구독 확인 이메일을 보냅니다. 이메일 알림을 수신하려면 수신자는 이 이메일에서 [Confirm subscription] 링크를 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.

2단계: 규칙을 만들고 SNS 주제를 대상으로 추가

CodePipeline이 이벤트 소스인 CloudWatch Events 알림 규칙을 만듭니다.

- 1. <u>https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.</u>
- 2. 탐색 창에서 이벤트를 선택합니다.

- 규칙 생성을 선택합니다. 이벤트 소스 아래에서 AWS CodePipeline을 선택합니다. [Event Type]에 서 [Pipeline Execution State Change]를 선택합니다.
- 4. 특정 상태를 선택한 후 FAILED를 선택합니다.
- 5. [Edit]를 선택하여 [Event Pattern Preview] 창의 JSON 편집기를 엽니다. 파이프라인 이름이 "myPipeline"인 다음 예와 같이 파이프라인 이름을 지정하여 **pipeline** 파라미터를 추가합니다.

여기에서 이벤트 패턴을 복사하여 콘솔에 붙여 넣을 수 있습니다.

```
{
  "source": [
    "aws.codepipeline"
  ],
  "detail-type": [
    "CodePipeline Pipeline Execution State Change"
  ],
  "detail": {
    "state": [
      "FAILED"
    ],
    "pipeline": [
      "myPipeline"
    ]
  }
}
```

- 6. 대상(Targets)에서 대상 추가(Add target)를 선택합니다.
- 7. 대상 목록에서 SNS 주제를 선택합니다. [Topic]에 생성한 주제를 입력합니다.
- 8. 입력 구성을 확장한 후 입력 변환기를 선택합니다.
- 9. [Input Path] 상자에 다음 키-값 페어를 입력합니다.

{ "pipeline" : "\$.detail.pipeline" }

[Input Template] 상자에 다음을 입력합니다.

"The Pipeline <pipeline> has failed."

- 10. 세부 정보 구성을 선택합니다.
- 11. [Configure rule details] 페이지에 이름과 설명(선택 사항)을 입력합니다. [State]에서 [Enabled] 상 자를 선택된 상태로 둡니다.

- 12. 규칙 생성을 선택합니다.
- 13. CodePipeline이 현재 빌드 알림을 보내고 있는지 확인합니다. 예를 들어 빌드 알림 이메일이 현재 받은 편지함에 있는지 확인합니다.
- 14. 규칙의 동작을 변경하려면 CloudWatch 콘솔에서 규칙을 선택한 후 작업, 편집을 선택합니다. 규 칙을 편집하고 [Configure details]를 선택한 후 [Update rule]을 선택합니다.

규칙을 사용하여 빌드 알림을 보내는 것을 중지하려면 CloudWatch 콘솔에서 해당 규칙을 선택한 후 작업, 비활성화를 선택합니다.

규칙을 삭제하려면 CloudWatch 콘솔에서 해당 규칙을 선택한 후 작업, 삭제를 선택합니다.

3단계: 리소스 정리

이 자습서를 완료한 후에는 사용한 파이프라인과 리소스를 삭제해야 이 리소스를 계속 사용할 경우 부 과되는 요금을 피할 수 있습니다.

SNS 알림을 정리하고 Amazon CloudWatch Events 규칙을 삭제하는 방법에 대한 자세한 내용은 <u>정리</u> (Amazon SNS 주제에서 구독 취소) 및 <u>Amazon CloudWatch Events API 참조</u>의 DeleteRule을 참조 하세요.

자습서:를 사용하여 Android 앱을 빌드하고 테스트하는 파이프라인 생성 AWS Device Farm

AWS CodePipeline 를 사용하여 커밋이 푸시될 때마다 앱을 빌드하고 테스트하는 지속적인 통합 흐름 을 구성할 수 있습니다. 이 자습서는 GitHub 리포지토리에서 소스 코드로 Android 앱을 빌드하고 테스 트하는 파이프라인을 생성하고 구성하는 방법을 보여줍니다. 파이프라인은 새로운 GitHub 커밋의 도 착을 감지한 다음 CodeBuild를 사용하여 앱을 빌드하고 Device Farm을 사용하여 앱을 테스트합니다.

A Important

콘솔에서 파이프라인을 생성하는 과정에서 CodePipeline은 S3 아티팩트 버킷을 아티팩트에 사용합니다. (이는 S3 소스 작업에 사용되는 버킷과 다릅니다.) S3 아티팩트 버킷이 파이프라 인의 계정과 다른 계정에 있는 경우 S3 아티팩트 버킷을 안전하고 신뢰할 수 AWS 계정 있는가 소유해야 합니다.

A Important

이 절차에서 파이프라인에 추가하는 많은 작업에는 파이프라인을 생성하기 전에 생성해야 하 는 AWS 리소스가 포함됩니다. 소스 작업에 대한 AWS 리소스는 항상 파이프라인을 생성하는 리전과 동일한 AWS 리전에서 생성해야 합니다. 예를 들어, 미국 동부(오하이오) 리전에서 파 이프라인을 생성하는 경우, CodeCommit 리포지토리도 미국 동부(오하이오) 리전에 있어야 합 니다.

파이프라인을 생성할 때 교차 리전 작업을 추가할 수 있습니다. 교차 리전 작업에 대한 AWS 리소스는 작업을 실행하려는 리전과 동일한 AWS 리전에 있어야 합니다. 자세한 내용은 CodePipeline에 교차 리전 작업 추가 단원을 참조하십시오.

기존 Android 앱 및 테스트 정의를 사용하여 시도하거나 <u>Device Farm에서 제공하는 샘플 앱 및 테스트</u> 정의를 사용할 수 있습니다.

Note

시작하기 전에

- 1. AWS Device Farm 콘솔에 로그인하고 새 프로젝트 생성을 선택합니다.
- 2. 프로젝트를 선택합니다. 브라우저에서 새 프로젝트의 URL을 복사합니다. URL에 프로젝트 ID가 포 함되어 있습니다.
- 3. 이 프로젝트 ID를 복사하여 보관합니다. CodePipeline에서 파이프라인을 생성할 때 사용합니다.

다음은 프로젝트 URL의 예입니다. 프로젝트 ID를 추출하려면 projects/ 뒤의 값을 복사합니다. 이 예시에서 제품 ID는 eec4905f-98f8-40aa-9afc-4c1cfexample입니다.

```
https://<region-URL>/devicefarm/home?region=us-west-2#/projects/
eec4905f-98f8-40aa-9afc-4c1cfexample/runs
```

Device Farm 테스트를 사용하도록 CodePipeline 구성

1.

앱 코드의 루트에서 <u>buildspec.yml</u>이라는 파일을 추가하고 커밋한 다음 리포지토리로 푸시합 니다. CodeBuild는 이 파일을 사용하여 앱을 빌드하는 데 필요한 명령을 수행하고 아티팩트에 액 세스합니다.

```
version: 0.2
phases:
    build:
        commands:
            - chmod +x ./gradlew
            - ./gradlew assembleDebug
artifacts:
    files:
            - './android/app/build/outputs/**/*.apk'
    discard-paths: yes
```

 (선택 사항) Calabash 또는 Appium을 사용하여 앱을 테스트한 경우 테스트 정의 파일을 리포지토 리에 추가합니다. 이후 단계에서 테스트 제품군을 수행하기 위한 정의를 사용하도록 Device Farm 을 구성할 수 있습니다.

Device Farm 내장형 테스트를 사용하는 경우 이 단계를 건너뛸 수 있습니다.

- 3. 파이프라인을 생성하고 소스 단계를 추가하려면 다음을 수행합니다.
 - a. 에 로그인 AWS Management Console 하고 <u>https://console.aws.amazon.com/codepipeline/</u>:// https://https://https://i/https://:/https://i/https//i/https/
 - b. [Welcome] 페이지, [Getting started] 페이지 또는 [Pipelines] 페이지에서 Create pipeline(파이 프라인 생성)을 선택합니다.
 - c. 1단계: 생성 옵션 선택 페이지의 생성 옵션에서 사용자 지정 파이프라인 빌드 옵션을 선택합니다. Next(다음)를 선택합니다.
 - d. 2단계: 파이프라인 설정 선택 페이지의 파이프라인 이름에 파이프라인 이름을 입력합니다.
 - e. CodePipeline은 특성과 가격이 다른 V1 및 V2 유형 파이프라인을 제공합니다. V2 유형은 콘 솔에서 선택할 수 있는 유일한 유형입니다. 자세한 내용은 <u>파이프라인 유형</u>을 참조하세요. CodePipeline에 대한 요금 정보는 <u>요금</u>을 참조하세요.
 - f. 서비스 역할에서 새 서비스 역할을 선택된 상태로 두고, 역할 이름도 변경하지 않고 그대로 둡니다. 이미 있는 경우 기존 서비스 역할을 사용하도록 선택할 수도 있습니다.

Note 2018년 7월 이전에 생성된 CodePip

2018년 7월 이전에 생성된 CodePipeline 서비스 역할을 사용할 경우 Device Farm에 대한 권한을 추가해야 합니다. 이렇게 하려면 IAM 콘솔을 열고 역할을 찾은 다음 역할 의 정책에 다음 권한을 추가합니다. 자세한 내용은 <u>CodePipeline 서비스 역할에 권한</u> 추가 단원을 참조하십시오.

{	
	"Effect": "Allow",
	"Action": [
	"devicefarm:ListProjects",
	"devicefarm:ListDevicePools",
	"devicefarm:GetRun",
	"devicefarm:GetUpload",
	"devicefarm:CreateUpload",
	"devicefarm:ScheduleRun"
],
	"Resource": "*"
}	

- g. [Advanced settings]의 설정은 기본값 그대로 두고, [Next]를 선택합니다.
- h. 3단계: 소스 단계 추가 페이지의 소스 공급자에서 GitHub(GitHub 앱을 통해)를 선택합니다.
- i. 연결에서 기존 연결을 선택하거나 새로 생성합니다. GitHub 소스 작업에 대한 연결을 만들거 나 관리하려면 GitHub 연결을 참조하세요.
- j. 리포지토리에서 소스 리포지토리를 선택합니다.
- k. 브랜치에서 사용할 브랜치를 선택합니다.
- I. 소스 작업의 나머지 기본값은 그대로 둡니다. Next(다음)를 선택합니다.
- 4. 4단계: 빌드 단계 추가에서 빌드 단계를 추가합니다.
 - a. 빌드 공급자에서 기타 빌드 공급자를 선택한 다음를 선택합니다AWS CodeBuild. 리전이 파이 프라인 리전으로 기본 설정되도록 합니다.
 - b. 프로젝트 생성을 선택합니다.
 - c. 프로젝트 이름에 이 빌드 프로젝트의 이름을 입력합니다.
 - d. 환경 이미지에서 이미지 관리를 선택합니다. [Operating system]에서 [Ubuntu]를 선택합니다.
 - e. 실행 시간에서 표준을 선택합니다. 이미지에서 aws/codebuild/standard:5.0을 선택합니다.

CodeBuild는 앱을 빌드하기 위해 Android Studio가 설치된 이 OS 이미지를 사용합니다.

- f. 서비스 역할에 기존 CodeBuild 서비스 역할을 선택하거나 새 역할을 생성합니다.
- g. Build specifications(빌드 사양)에서 Use a buildspec file(빌드 사양 파일 사용)을 선택합니다.
- h. Continue to CodePipeline(CodePipeline으로 계속)을 선택합니다. 이렇게 하면 CodePipeline 콘솔로 반환되고 구성을 위해 리포지토리에서 buildspec.yml을 사용하는 CodeBuild 프로 젝트를 생성합니다. 빌드 프로젝트가 서비스 역할을 사용하여 AWS 서비스 권한을 관리합니 다. 이 단계는 몇 분이 걸릴 수 있습니다.
- i. Next(다음)를 선택합니다.
- 5. 5단계: 테스트 단계 추가에서 테스트 단계 건너뛰기를 선택한 다음 건너뛰기를 다시 선택하여 경고 메시지를 수락합니다.

Next(다음)를 선택합니다.

- 6단계: 배포 단계 추가 페이지에서 배포 단계 건너뛰기를 선택한 다음 다시 건너뛰기를 선택하여 경고 메시지를 수락합니다. Next(다음)를 선택합니다.
- 7. 7단계: 검토에서 파이프라인 생성을 선택합니다. 소스 및 빌드 단계를 보여주는 다이어그램을 확 인해야 합니다.
- 8. 파이프라인에 Device Farm 테스트 작업을 추가하세요.
 - a. 오른쪽 위에서 편집을 선택합니다.
 - b. 다이어그램의 하단에서 + 단계 추가를 선택합니다. Stage name(스테이지 이름)에서 이름(예: **Test**)을 입력합니다.
 - c. + Add action group(작업 그룹 추가)을 선택합니다.
 - d. 작업 이름에 이름을 입력하세요.
 - e. 작업 공급자에서 AWS Device Farm을 선택합니다. 리전이 파이프라인 리전으로 기본 설정되 도록 합니다.
 - f. 입력 아티팩트에서 BuildArtifact과 같이 파이프라인의 테스트 스테이지 전에 오는 단계
 의 출력 아티팩트와 일치하는 입력 아티팩트를 선택합니다.

AWS CodePipeline 콘솔에서 파이프라인 다이어그램의 정보 아이콘 위로 마우스를 가져 가면 각 단계의 출력 아티팩트 이름을 찾을 수 있습니다. 소스 스테이지에서 파이프라인이 바로 앱을 테스트하면 SourceArtifact를 선택합니다. 파이프라인이 빌드 단계를 포함하면 BuildArtifact를 선택합니다.

g. ProjectId에 Device Farm 프로젝트 ID를 입력합니다. 이 자습서의 시작 부분에 있는 단계를 따라 프로젝트 ID를 검색합니다.

h. DevicePoolArn에서 디바이스 풀의 ARN을 입력합니다. 상위 디바이스에 대한 ARNs을 포함 하여 프로젝트에 사용 가능한 디바이스 풀 ARN을 가져오려면 AWS CLI를 사용하여 다음 명 령을 입력합니다.

aws devicefarm list-device-pools --arn arn:aws:devicefarm:uswest-2:account_ID:project:project_ID

i. AppType에서 Android를 입력합니다.

AppType의 유효한 값 목록은 다음과 같습니다.

- iOS
- Android
- 웹
- j. 앱에서 컴파일된 앱 패키지의 경로를 입력합니다. 경로는 테스트 단계용 입력 아티팩트의 루 트와 상대적입니다. 일반적으로 이 경로는 app-release.apk와 유사합니다.
- k. TestType에 테스트 유형을 입력한 다음, 테스트에 테스트 정의 파일의 경로를 입력합니다. 경 로는 테스트용 입력 아티팩트의 루트와 상대적입니다.

TestType의 유효한 값 목록은 다음과 같습니다.

- APPIUM_JAVA_JUNIT
- APPIUM_JAVA_TESTNG
- APPIUM_NODE
- APPIUM_RUBY
- APPIUM_PYTHON
- APPIUM_WEB_JAVA_JUNIT
- APPIUM_WEB_JAVA_TESTNG
- APPIUM_WEB_NODE
- APPIUM_WEB_RUBY
- APPIUM_WEB_PYTHON
- BUILTIN_FUZZ
- INSTRUMENTATION
- XCTEST

Device Farm 테스토를 산용하도록 CodePipeline 구성 • XCTEST_UI Note
 사용자 지정 환경 노드는 지원되지 않습니다.

- I. 나머지 필드에서 테스트 및 애플리케이션 유형에 적절한 구성을 제공하세요.
- m. (선택 사항) 고급에서 테스트 실행 시 구성 정보를 제공합니다.
- n. 저장(Save)을 선택합니다.
- . 편집 중인 스테이지에서 완료를 선택합니다. AWS CodePipeline 창에서 저장을 선택한 다음
 경고 메시지에서 저장을 선택합니다.
- p. 변경 사항을 제출하고 파이프라인 빌드를 시작하려면 변경 사항 배포를 선택한 다음 릴리 스를 선택하세요.

자습서:를 사용하여 iOS 앱을 테스트하는 파이프라인 생성 AWS Device Farm

AWS CodePipeline 를 사용하여 소스 버킷이 변경될 때마다 앱을 테스트하는 지속적인 통합 흐름을 쉽 게 구성할 수 있습니다. 이 자습서는 S3 버킷에서 빌드된 iOS 앱을 테스트하는 파이프라인을 생성하고 구성하는 방법을 보여줍니다. 파이프라인은 Amazon CloudWatch Events을 통해 저장된 변경 사항의 도착을 감지한 다음 Device Farm을 사용하여 빌드된 애플리케이션을 테스트합니다.

Important

파이프라인 생성의 일환으로 CodePipeline은 고객이 제공하는 S3 아티팩트 버킷을 아티팩트 에 사용합니다. (이는 S3 소스 작업에 사용되는 버킷과 다릅니다.) S3 아티팩트 버킷이 파이프 라인의 계정과 다른 계정에 있는 경우 S3 아티팩트 버킷을 안전하고 신뢰할 수 AWS 계정 있는 가 소유해야 합니다.

A Important

이 절차에서 파이프라인에 추가하는 많은 작업에는 파이프라인을 생성하기 전에 생성해야 하는 AWS 리소스가 포함됩니다. 소스 작업에 대한 AWS 리소스는 항상 파이프라인을 생성하는 리전과 동일한 AWS 리전에서 생성해야 합니다. 예를 들어, 미국 동부(오하이오) 리전에서 파

사용자 가이드

이프라인을 생성하는 경우, CodeCommit 리포지토리도 미국 동부(오하이오) 리전에 있어야 합니다.

파이프라인을 생성할 때 교차 리전 작업을 추가할 수 있습니다. 교차 리전 작업에 대한 AWS 리소스는 작업을 실행하려는 리전과 동일한 AWS 리전에 있어야 합니다. 자세한 내용은 CodePipeline에 교차 리전 작업 추가 단원을 참조하십시오.

기존 iOS 앱을 사용하여 시도하거나 샘플 iOS 앱을 사용할 수 있습니다.



시작하기 전에

- 1. AWS Device Farm 콘솔에 로그인하고 새 프로젝트 생성을 선택합니다.
- 2. 프로젝트를 선택합니다. 브라우저에서 새 프로젝트의 URL을 복사합니다. URL에 프로젝트 ID가 포 함되어 있습니다.
- 3. 이 프로젝트 ID를 복사하여 보관합니다. CodePipeline에서 파이프라인을 생성할 때 사용합니다.

다음은 프로젝트 URL의 예입니다. 프로젝트 ID를 추출하려면 projects/ 뒤의 값을 복사합니다. 이 예시에서 제품 ID는 eec4905f-98f8-40aa-9afc-4c1cfexample입니다.

https://<region-URL>/devicefarm/home?region=us-west-2#/projects/ eec4905f-98f8-40aa-9afc-4c1cfexample/runs

Device Farm 테스트를 사용하도록 CodePipeline 구성(Amazon S3 예)

- 버전 관리가 활성화된 S3 버킷을 생성하거나 사용합니다. <u>1단계: 애플리케이션에 대한 S3 소스 버</u> 킷 생성의 지침을 따라 S3 버킷을 생성합니다.
- 2. 버킷의 Amazon S3 콘솔에서 업로드를 선택하고 지침을 따라 .zip 파일을 업로드합니다.

샘플 애플리케이션은 .zip 파일로 압축해야 합니다.

- 3. 파이프라인을 생성하고 소스 단계를 추가하려면 다음을 수행합니다.
 - a. 에 로그인 AWS Management Console 하고 <u>https://console.aws.amazon.com/codepipeline/</u>:// https://https://https://i/https://:/https://i/https//i/https/

- b. [Welcome] 페이지, [Getting started] 페이지 또는 [Pipelines] 페이지에서 Create pipeline(파이 프라인 생성)을 선택합니다.
- c. 1단계: 생성 옵션 선택 페이지의 생성 옵션에서 사용자 지정 파이프라인 빌드 옵션을 선택합니다. Next(다음)를 선택합니다.
- d. 2단계: 파이프라인 설정 선택 페이지의 파이프라인 이름에 파이프라인 이름을 입력합니다.
- e. CodePipeline은 특성과 가격이 다른 V1 및 V2 유형 파이프라인을 제공합니다. V2 유형은 콘 솔에서 선택할 수 있는 유일한 유형입니다. 자세한 내용은 <u>파이프라인 유형</u>을 참조하세요. CodePipeline에 대한 요금 정보는 <u>요금</u>을 참조하세요.
- f. 서비스 역할에서 새 서비스 역할을 선택된 상태로 두고, 역할 이름도 변경하지 않고 그대로 둡니다. 이미 있는 경우 기존 서비스 역할을 사용하도록 선택할 수도 있습니다.

Note

2018년 7월 이전에 생성된 CodePipeline 서비스 역할을 사용할 경우 Device Farm에 대한 권한을 추가해야 합니다. 이렇게 하려면 IAM 콘솔을 열고 역할을 찾은 다음 역할 의 정책에 다음 권한을 추가합니다. 자세한 내용은 <u>CodePipeline 서비스 역할에 권한</u> 추가 단원을 참조하십시오.

```
{
    "Effect": "Allow",
    "Action": [
        "devicefarm:ListProjects",
        "devicefarm:ListDevicePools",
        "devicefarm:GetRun",
        "devicefarm:GetUpload",
        "devicefarm:CreateUpload",
        "devicefarm:ScheduleRun"
    ],
    "Resource": "*"
}
```

- g. [Advanced settings]의 설정은 기본값 그대로 두고, [Next]를 선택합니다.
- h. 3단계: 소스 스테이지 추가 페이지의 소스 공급자에서 Amazon S3를 선택합니다.
- i. Amazon S3 위치에 버킷(예: my-storage-bucket) 및 객체 키(예: .zip 파일의 s3-iostest-1.zip)를 입력합니다.
- j. Next(다음)를 선택합니다.

- 4단계: 빌드 단계 추가에서 파이프라인에 대한 자리 표시자 빌드 단계를 생성합니다. 이렇게 하면 마법사에서 파이프라인을 생성할 수 있습니다. 마법사를 사용하여 두 단계 파이프라인을 만든 후 에는 자리 표시자 빌드 단계가 더 이상 필요 없습니다. 파이프라인이 완료된 후에는 이 두 번째 단 계가 삭제되고 새로운 테스트 단계가 5단계에 추가됩니다.
 - a. 빌드 공급자에서 Jenkins 추가를 선택합니다. 이 빌드 선택은 자리 표시자입니다. 사용되지 않 습니다.
 - b. 공급자 이름에 이름을 입력합니다. 이름은 자리 표시자입니다. 사용되지 않습니다.
 - c. 서버 URL에 텍스트를 입력합니다. 텍스트는 자리 표시자입니다. 사용되지 않습니다.
 - d. 프로젝트 이름에 이름을 입력합니다. 이름은 자리 표시자입니다. 사용되지 않습니다.
 - e. Next(다음)를 선택합니다.
 - f. 5단계: 테스트 단계 추가에서 테스트 단계 건너뛰기를 선택한 다음 건너뛰기를 다시 선택하여 경고 메시지를 수락합니다.

Next(다음)를 선택합니다.

- g. 6단계: 배포 단계 추가 페이지에서 배포 단계 건너뛰기를 선택한 다음 다시 건너뛰기를 선택 하여 경고 메시지를 수락합니다.
- h. 7단계: 검토에서 파이프라인 생성을 선택합니다. 소스 및 빌드 단계를 보여주는 다이어그램을 확인해야 합니다.

Source Amazon S3 🖸 ? In progress - Just now	Ġ
Disable transition	
Build Custom Jenkins1 (Version: 1) 🖸 O Didn't Run	١

- 5. 다음과 같이 파이프라인에 Device Farm 테스트 작업을 추가합니다.
 - a. 오른쪽 위에서 편집을 선택합니다.
 - b. Edit stage(단계 편집)을 선택합니다. Delete(삭제)를 선택합니다. 그러면 자리 표시자 단계가 삭제됩니다(이제 더 이상 파이프라인 생성에 필요 없으므로).
 - c. 다이어그램의 하단에서 + 단계 추가를 선택합니다.
 - d. 스테이지 이름에서 스테이지 이름(예: 테스트)을 입력한 다음 Add stage(스테이지 추가)를 선 택합니다.
 - e. + Add action group(작업 그룹 추가)을 선택합니다.
 - f. 작업 이름에서 이름(예: DeviceFarmTest)을 입력합니다.
 - g. 작업 공급자에서 AWS Device Farm을 선택합니다. 리전이 파이프라인 리전으로 기본 설정되 도록 합니다.
 - h. 입력 아티팩트에서 SourceArtifact과 같이 파이프라인의 테스트 스테이지 전에 오는 단계 의 출력 아티팩트와 일치하는 입력 아티팩트를 선택합니다.

AWS CodePipeline 콘솔에서 파이프라인 다이어그램의 정보 아이콘 위로 마우스를 가져 가면 각 단계의 출력 아티팩트 이름을 찾을 수 있습니다. 소스 스테이지에서 파이프라인이 바로 앱을 테스트하면 SourceArtifact를 선택합니다. 파이프라인이 빌드 단계를 포함하면 BuildArtifact를 선택합니다.

- i. ProjectId에서 Device Farm 프로젝트 ID를 선택합니다. 이 자습서의 시작 부분에 있는 단계를 따라 프로젝트 ID를 검색합니다.
- j. DevicePoolArn에서 디바이스 풀의 ARN을 입력합니다. 상위 디바이스에 대한 ARNs을 포함 하여 프로젝트에 사용 가능한 디바이스 풀 ARN을 가져오려면 AWS CLI를 사용하여 다음 명 령을 입력합니다.

aws devicefarm list-device-pools --arn arn:aws:devicefarm:uswest-2:account_ID:project:project_ID

k. AppType에서 iOS를 입력합니다.

AppType의 유효한 값 목록은 다음과 같습니다.

- iOS
- Android
- 웹

- 앱에서 컴파일된 앱 패키지의 경로를 입력합니다. 경로는 테스트 단계용 입력 아티팩트의 루 트와 상대적입니다. 일반적으로 이 경로는 ios-test.ipa와 유사합니다.
- m. TestType에 테스트 유형을 입력한 다음, 테스트에 테스트 정의 파일의 경로를 입력합니다. 경
 로는 테스트용 입력 아티팩트의 루트와 상대적입니다.

내장형 Device Farm 테스트 중 하나를 사용하는 경우 Device Farm 프로젝트에 구성된 테스 트 유형(예: BUILTIN_FUZZ)을 선택합니다. FuzzEventCount에서 밀리초 단위로 시간을 입력 합니다(예: 6000). FuzzEventThrottle에서 밀리초 단위로 시간을 입력합니다(예: 50).

내장형 Device Farm 테스트 중 하나를 사용하지 않는 경우, 테스트 유형을 선택한 다음 테스 트에서 테스트 정의 파일의 경로를 입력합니다. 경로는 테스트용 입력 아티팩트의 루트와 상 대적입니다.

TestType의 유효한 값 목록은 다음과 같습니다.

- APPIUM_JAVA_JUNIT
- APPIUM_JAVA_TESTNG
- APPIUM_NODE
- APPIUM_RUBY
- APPIUM_PYTHON
- APPIUM_WEB_JAVA_JUNIT
- APPIUM_WEB_JAVA_TESTNG
- APPIUM_WEB_NODE
- APPIUM_WEB_RUBY
- APPIUM_WEB_PYTHON
- BUILTIN_FUZZ
- INSTRUMENTATION
- XCTEST
- XCTEST_UI

Note

사용자 지정 환경 노드는 지원되지 않습니다.

Device Farm 텐片 려륏사 헐느 에 왜 예 베 쁘 뜨 몇 (해 쫄 면 케이션 유형에 적절한 구성을 제공하세요. API 버전 2015-07-09 179

- o. (선택 사항) 고급에서 테스트 실행 시 구성 정보를 제공합니다.
- p. 저장(Save)을 선택합니다.
- q. 편집 중인 스테이지에서 완료를 선택합니다. AWS CodePipeline 창에서 저장을 선택한 다음 경고 메시지에서 저장을 선택합니다.
- r. 변경 사항을 제출하고 파이프라인 실행을 시작하려면 변경 사항 배포를 선택한 다음 릴리
 스를 선택합니다.

자습서: Service Catalog에 배포하는 파이프라인 생성

Service Catalog를 사용하면 AWS CloudFormation 템플릿을 기반으로 제품을 생성하고 프로비저닝할 수 있습니다.

A Important

파이프라인 생성의 일환으로 CodePipeline은 고객이 제공하는 S3 아티팩트 버킷을 아티팩트 에 사용합니다. (이는 S3 소스 작업에 사용되는 버킷과 다릅니다.) S3 아티팩트 버킷이 파이프 라인의 계정과 다른 계정에 있는 경우 S3 아티팩트 버킷을 안전하고 신뢰할 수 AWS 계정 있는 가 소유해야 합니다.

이 자습서에서는 Service Catalog에 제품 템플릿을 배포하고 소스 리포지토리(이미 GitHub, CodeCommit 또는 Amazon S3에 생성됨)에서 변경한 사항을 제공하기 위해 파이프라인을 생성하고 구성하는 방법을 보여 줍니다.

Note

Amazon S3가 파이프라인의 소스 공급자인 경우, 단일 .zip 파일로 패키지된 모든 소스 파일을 버킷에 업로드해야 합니다. 그렇지 않으면 소스 작업이 실패합니다.

먼저 Service Catalog에서 제품을 생성한 다음 AWS CodePipeline에서 파이프 라인을 생성합니다. 이 자습서에서는 배포 구성을 설정하기 위한 두 가지 옵션을 제공합니다.

 Service Catalog에서 제품을 생성하고 소스 리포지토리에 템플릿 파일을 업로드합니다. 별도의 구 성 파일 없이 CodePipeline 콘솔에서 제품 버전 및 배포 구성을 제공합니다. <u>옵션 1: 구성 파일 없이</u> Service Catalog에 배포을 참조하세요.

자습서: Service Catalog에 배포하는 파이프라인 생성

Note

템플릿 파일은 YAML 또는 JSON 형식으로 생성할 수 있습니다.

 Service Catalog에서 제품을 생성하고 소스 리포지토리에 템플릿 파일을 업로드합니다. 별도의 구성 파일에서 제품 버전 및 배포 구성을 제공합니다. <u>옵션 2: 구성 파일을 사용하여 Service Catalog에 배</u> 포을 참조하세요.

옵션 1: 구성 파일 없이 Service Catalog에 배포

이 예제에서는 S3 버킷에 대한 샘플 AWS CloudFormation 템플릿 파일을 업로드한 다음 Service Catalog에서 제품을 생성합니다. 그런 다음 파이프라인을 생성하고 CodePipeline 콘솔에서 배포 구성 을 지정합니다.

1단계: 샘플 템플릿 파일을 소스 리포지토리에 업로드

 텍스트 편집기를 엽니다. 다음을 파일에 붙여 넣고 샘플 템플릿을 생성합니다. 파일을 S3_template.json(으)로 저장합니다.

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "CloudFormation Sample Template S3_Bucket: Sample template showing
how to create a privately accessible S3 bucket. **WARNING** This template creates
 an S3 bucket. You will be billed for the resources used if you create a stack from
this template.",
  "Resources": {
    "S3Bucket": {
      "Type": "AWS::S3::Bucket",
     "Properties": {}
   }
  },
  "Outputs": {
    "BucketName": {
      "Value": {
        "Ref": "S3Bucket"
      },
      "Description": "Name of Amazon S3 bucket to hold website content"
    }
  }
```

}

이 템플릿을 사용하면가 Service Catalog에서 사용할 수 있는 S3 버킷을 AWS CloudFormation 생성할 수 있습니다.

2. AWS CodeCommit 리포지토리에 S3_template.json 파일을 업로드합니다.

2단계: Service Catalog에서 제품 생성

- 1. IT 관리자로 Service Catalog 콘솔에 로그인하고 제품 페이지로 이동한 다음 제품 신규 업로드를 선택합니다.
- 2. 제품 신규 업로드 페이지에서 다음 작업을 완료합니다.
 - a. 제품 이름에 새 제품에 사용할 이름을 입력합니다.
 - b. 설명에 제품 카탈로그 설명을 입력합니다. 이 설명은 사용자가 올바른 제품을 선택할 수 있도
 록 제품 목록에 표시됩니다.
 - c. 공급자 IT 부서 또는 관리자의 이름을 입력합니다.
 - d. Next(다음)를 선택합니다.
- (선택 사항) 지원 세부 정보 입력 페이지에서 제품 지원에 대한 연락처 정보를 입력하고 다음을 선 택합니다.
- 4. 버전 세부 정보에서 다음을 완료합니다.
 - a. 템플릿 파일 업로드를 선택합니다. S3_template.json 파일을 찾고 업로드합니다.
 - b. 버전 제목 제품 버전의 이름을 입력합니다(예: devops S3 v2).
 - c. 설명에서 이 버전을 다른 버전과 구별하는 세부 정보를 입력합니다.
 - d. Next(다음)를 선택합니다.
- 5. 검토 페이지에서 정보가 올바른지 확인한 다음 생성을 선택합니다.
- 제품 페이지의 브라우저에서 새 제품의 URL을 복사합니다. 이는 제품 ID를 포함합니다. 이 제품 ID를 복사하여 보관합니다. CodePipeline에서 파이프라인을 생성할 때 사용합니다.

다음은 my-product 제품의 URL입니다. 제품 ID를 추출하려면 등호 기호(=) 및 앰퍼샌드(&) 사이의 값을 복사합니다. 이 예시에서의 제품 ID는 prod-example123456입니다.

https://<region-URL>/servicecatalog/home?region=<region>#/admin-products?
productCreated=prod-example123456&createdProductTitle=my-product

Note

페이지에서 벗어나기 전에 제품의 URL을 복사합니다. 이 페이지를 벗어난 후 CLI를 사용 하여 제품 ID를 얻어야 합니다.

몇 초 후 제품 페이지에 제품이 표시됩니다. 목록의 제품을 보기 위해 브라우저를 새로 고쳐야 할 수도 있습니다.

3단계: 파이프라인 생성

- 1. 파이프라인에 이름을 지정하고 파이프라인에 대한 파라미터를 선택하려면 다음을 수행합니다.
 - a. 에 로그인 AWS Management Console 하고 <u>https://console.aws.amazon.com/codepipeline/</u>:// https://https://https://https://:/httpsCodePipeline.com.com.com.
 - b. [Welcome] 페이지, [Getting started] 페이지 또는 [Pipelines] 페이지에서 Create pipeline(파이 프라인 생성)을 선택합니다.
 - c. 1단계: 생성 옵션 선택 페이지의 생성 옵션에서 사용자 지정 파이프라인 빌드 옵션을 선택합니다. Next(다음)를 선택합니다.
 - d. 2단계: 파이프라인 설정 선택의 파이프라인 이름에 파이프라인 이름을 입력합니다.
 - e. CodePipeline은 특성과 가격이 다른 V1 및 V2 유형 파이프라인을 제공합니다. V2 유형은 콘 솔에서 선택할 수 있는 유일한 유형입니다. 자세한 내용은 <u>파이프라인 유형</u>을 참조하세요. CodePipeline에 대한 요금 정보는 요금을 참조하세요.
 - f. 서비스 역할에서 새 서비스 역할을 선택하여CodePipeline이 IAM에 서비스 역할을 생성하도 록 허용합니다.
 - g. [Advanced settings]의 설정은 기본값 그대로 두고, [Next]를 선택합니다.
- 2. 3단계: 소스 단계 추가 페이지에서 소스 단계를 추가하려면 다음을 수행합니다.
 - a. 소스 공급자에서 AWS CodeCommit을 선택합니다.
 - b. 리포지토리 이름 및 브랜치 이름에서 소스 작업에 사용할 리포지토리와 브랜치를 입력합니다.
 - c. Next(다음)를 선택합니다.
- 4단계: 빌드 스테이지 추가에서 빌드 스테이지 건너뛰기를 선택하고 건너뛰기를 다시 선택하여 경고 메시지를 수락합니다.

5단계: 테스트 단계 추가에서 테스트 단계 건너뛰기를 선택한 다음 건너뛰기를 다시 선택하여 경고 메시지를 수락합니다.

Next(다음)를 선택합니다.

- 5. 6단계: 배포 단계 추가에서 다음을 완료합니다.
 - a. Deploy provider(배포 공급자)에서 AWS Service Catalog를 선택합니다.
 - b. 배포 구성에서 Enter deployment configuration(배포 구성 입력)을 선택합니다.
 - c. 제품 ID에 Service Catalog 콘솔에서 복사한 제품 ID를 붙여 넣습니다.
 - d. Template file path(템플릿 파일 경로)에 템플릿 파일이 저장된 상대 경로를 입력합니다.
 - e. 제품 유형에서 AWS CloudFormation 템플릿을 선택합니다.
 - f. 제품 버전 이름에 Service Catalog에서 지정한 제품 버전의 이름을 입력합니다. 템플릿 변경 사항을 새 제품 버전에 배포하려면 동일한 제품의 이전 제품 버전에 사용되지 않은 제품 버전 이름을 입력합니다.
 - g. 입력 아티팩트에 소스 입력 아티팩트를 선택합니다.
 - h. Next(다음)를 선택합니다.
- 6. 7단계: 검토에서 파이프라인 설정을 검토한 다음 생성을 선택합니다.
- 7. 파이프라인이 성공적으로 실행된 후 배포 단계에서 세부 정보를 선택합니다. 그러면 제품이 Service Catalog에서 열립니다.

⊙ Source		
Source CodeCommit	٩	
Succeeded - 1 minute ago 99e3b988 Source: Added S3 template ison 99	e3h988	
Disable transition		
⊙ Deploy		
Deploy	١	
ServiceCatalog		
Succeeded - 1 minute ago		
	o7b099	

8. 제품 정보에서 버전 이름을 선택하여 제품 템플릿을 엽니다. 템플릿 배포를 봅니다.

4단계: Service Catalog에서 변경 사항 푸시 및 제품 확인

 CodePipeline 콘솔의 파이프라인을 보고 소스 페이지에서 세부 정보를 선택합니다. 소스 AWS CodeCommit 리포지토리가 콘솔에서 열립니다. 편집을 선택하고 파일에 대해 변경합니다(예: 설 명 변경).

"Description": "Name of Amazon S3 bucket to hold and version website content"

- 변경 사항을 커밋하고 푸시합니다. 변경 사항을 푸시한 후 파이프라인이 시작됩니다. 파이프라인 실행이 완료되면 배포 단계에서 세부 정보를 선택하여 Service Catalog에서 제품을 엽니다.
- 제품 정보에서 새 버전 이름을 선택하여 제품 템플릿을 엽니다. 배포된 템플릿 변경 사항을 봅니다.

옵션 2: 구성 파일을 사용하여 Service Catalog에 배포

이 예제에서는 S3 버킷에 대한 샘플 AWS CloudFormation 템플릿 파일을 업로드한 다음 Service Catalog에서 제품을 생성합니다. 또한 배포 구성을 지정하는 별도의 구성 파일을 업로드합니다. 그런 다음 파이프 라인을 생성하고 구성 파일의 위치를 지정합니다.

1단계: 샘플 템플릿 파일을 소스 리포지토리에 업로드

 텍스트 편집기를 엽니다. 다음을 파일에 붙여 넣고 샘플 템플릿을 생성합니다. 파일을 S3_template.json(으)로 저장합니다.

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "CloudFormation Sample Template S3_Bucket: Sample template showing
 how to create a privately accessible S3 bucket. **WARNING** This template creates
 an S3 bucket. You will be billed for the resources used if you create a stack from
 this template.",
  "Resources": {
    "S3Bucket": {
      "Type": "AWS::S3::Bucket",
      "Properties": {}
   }
  },
  "Outputs": {
    "BucketName": {
      "Value": {
        "Ref": "S3Bucket"
      },
      "Description": "Name of Amazon S3 bucket to hold website content"
    }
  }
}
```

이 템플릿을 사용하면가 Service Catalog에서 사용할 수 있는 S3 버킷을 AWS CloudFormation 생성할 수 있습니다.

2. AWS CodeCommit 리포지토리에 S3_template.json 파일을 업로드합니다.

2단계: 제품 배포 구성 파일 생성

1. 텍스트 편집기를 엽니다. 제품의 구성 파일을 생성합니다. 구성 파일은 Service Catalog 배포 파라 미터/기본 설정을 정의하는 데 사용됩니다. 이 파일은 파이프라인을 생성할 때 사용합니다.

이 샘플은 "devops S3 v2"의 ProductVersionName 및 MyProductVersionDescription의 ProductVersionDescription을 제공합니다. 템플릿 변경 사항을 새 제품 버전에 배포하려면 동일한 제품의 이전 제품 버전에 사용되지 않은 제품 버전 이름을 입력합니다.

파일을 sample_config.json(으)로 저장합니다.

```
{
    "SchemaVersion": "1.0",
    "ProductVersionName": "devops S3 v2",
    "ProductVersionDescription": "MyProductVersionDescription",
    "ProductType": "CLOUD_FORMATION_TEMPLATE",
    "Properties": {
        "TemplateFilePath": "/S3_template.json"
    }
}
```

이 파일은 파이프라인이 실행될 때마다 제품 버전 정보를 생성합니다.

2. AWS CodeCommit 리포지토리에 sample_config.json 파일을 업로드합니다. 이 파일을 소스 리포지토리에 업로드했는지 확인합니다.

3단계: Service Catalog에서 제품 생성

- IT 관리자로 Service Catalog 콘솔에 로그인하고 제품 페이지로 이동한 다음 제품 신규 업로드를 선택합니다.
- 2. 제품 신규 업로드 페이지에서 다음 작업을 완료합니다.
 - a. 제품 이름에 새 제품에 사용할 이름을 입력합니다.
 - b. 설명에 제품 카탈로그 설명을 입력합니다. 이 설명은 사용자가 올바른 제품을 선택할 수 있도
 록 제품 목록에 표시됩니다.
 - c. 공급자 IT 부서 또는 관리자의 이름을 입력합니다.
 - d. Next(다음)를 선택합니다.
- (선택 사항) 지원 세부 정보 입력 페이지에서 제품 지원 연락처 정보를 입력하고 다음을 선택합니다.

- 4. 버전 세부 정보에서 다음을 완료합니다.
 - a. 템플릿 파일 업로드를 선택합니다. S3_template.json 파일을 찾고 업로드합니다.
 - b. 버전 제목 제품 버전의 이름을 입력합니다(예: "devops S3 v2").
 - c. 설명에서 이 버전을 다른 버전과 구별하는 세부 정보를 입력합니다.
 - d. Next(다음)를 선택합니다.
- 5. 검토 페이지에서 정보가 올바른지 확인한 후 Confirm and upload(확인 및 업로드)를 선택합니다.
- 제품 페이지의 브라우저에서 새 제품의 URL을 복사합니다. 이는 제품 ID를 포함합니다. 이 제품 ID를 복사하여 보관합니다. CodePipeline에서 파이프라인을 생성할 때 사용합니다.

다음은 my-product 제품의 URL입니다. 제품 ID를 추출하려면 등호 기호(=) 및 앰퍼샌드(&) 사이의 값을 복사합니다. 이 예시에서의 제품 ID는 prod-example123456입니다.

https://<region-URL>/servicecatalog/home?region=<region>#/admin-products?
productCreated=prod-example123456&createdProductTitle=my-product

Note

페이지에서 벗어나기 전에 제품의 URL을 복사합니다. 이 페이지를 벗어난 후 CLI를 사용 하여 제품 ID를 얻어야 합니다.

몇 초 후 제품 페이지에 제품이 표시됩니다. 목록의 제품을 보기 위해 브라우저를 새로 고쳐야 할 수도 있습니다.

4단계: 파이프라인 생성

- 1. 파이프라인에 이름을 지정하고 파이프라인에 대한 파라미터를 선택하려면 다음을 수행합니다.
 - a. 에 로그인 AWS Management Console 하고 <u>https://console.aws.amazon.com/codepipeline/</u>:// https://https://https://i/https://:/https://i/https//i/https://i/https/
 - b. 시작하기를 선택합니다. 파이프라인 생성을 선택한 다음 파이프라인의 이름을 입력합니다.
 - c. 서비스 역할에서 새 서비스 역할을 선택하여CodePipeline이 IAM에 서비스 역할을 생성하도 록 허용합니다.
 - d. [Advanced settings]의 설정은 기본값 그대로 두고, [Next]를 선택합니다.
- 2. 소스 단계를 추가하려면 다음을 수행합니다.

- a. 소스 공급자에서 AWS CodeCommit을 선택합니다.
- b. 리포지토리 이름 및 브랜치 이름에서 소스 작업에 사용할 리포지토리와 브랜치를 입력합니 다.
- c. Next(다음)를 선택합니다.
- 3. Add build stage(빌드 단계 추가)에서 Skip build stage(빌드 단계 건너뛰기)를 선택하고 Skip(건너 뛰기)을 다시 선택하여 경고 메시지를 수락합니다.
- 4. Add deploy stage(배포 단계 추가)에서 다음을 완료합니다.
 - a. Deploy provider(배포 공급자)에서 AWS Service Catalog를 선택합니다.
 - b. Use configuration file(구성 파일 사용)을 선택합니다.
 - c. 제품 ID에 Service Catalog 콘솔에서 복사한 제품 ID를 붙여 넣습니다.
 - d. Configuration file path(구성 파일 경로)에서 리포지토리의 구성 파일 경로를 입력합니다.
 - e. Next(다음)를 선택합니다.
- 5. Review(검토)에서 파이프라인 설정을 검토한 다음 Create(생성)를 선택합니다.
- 파이프라인이 성공적으로 실행된 후 배포 단계에서 세부 정보를 선택하여 Service Catalog에서 제 품을 엽니다.

⊙ Source		
Source CodeCommit Succeeded - 1 minute ago 99e3b988	١	
iource: Added S3_template.json 99	e3b988	
🔊 Deploy		
Deploy ServiceCatalog 🖸	١	
Succeeded - 1 minute ago		

7. 제품 정보에서 버전 이름을 선택하여 제품 템플릿을 엽니다. 템플릿 배포를 봅니다.

5단계: Service Catalog에서 변경 사항 푸시 및 제품 확인

 CodePipeline 콘솔의 파이프라인을 보고 소스 페이지에서 세부 정보를 선택합니다. 소스 AWS CodeCommit 리포지토리가 콘솔에서 열립니다. 편집을 선택한 다음 파일에 대해 변경합니다(예: 설명 변경).

"Description": "Name of Amazon S3 bucket to hold and version website content"

- 변경 사항을 커밋하고 푸시합니다. 변경 사항을 푸시한 후 파이프라인이 시작됩니다. 파이프라인 실행이 완료되면 배포 단계에서 세부 정보를 선택하여 Service Catalog에서 제품을 엽니다.
- 제품 정보에서 새 버전 이름을 선택하여 제품 템플릿을 엽니다. 배포된 템플릿 변경 사항을 봅니다.

자습서:를 사용하여 파이프라인 생성 AWS CloudFormation

이 예제에서는 소스 코드가 변경될 때마다를 사용하여 인스턴스 AWS CloudFormation 에 애플리 케이션을 배포하는 파이프라인을 생성할 수 있는 샘플 템플릿을 제공합니다. 샘플 템플릿은 AWS CodePipeline에서 볼 수 있는 파이프라인을 생성합니다. 파이프라인은 Amazon CloudWatch Events를 통해 저장된 변경 사항의 도착을 감지합니다.

Important

파이프라인 생성의 일환으로 CodePipeline은 고객이 제공하는 S3 아티팩트 버킷을 아티팩트 에 사용합니다. (이는 S3 소스 작업에 사용되는 버킷과 다릅니다.) S3 아티팩트 버킷이 파이프 라인의 계정과 다른 계정에 있는 경우 S3 아티팩트 버킷을 안전하고 신뢰할 수 AWS 계정 있는 에서 소유해야 합니다.

주제

- <u>예제 1:를 사용하여 AWS CodeCommit 파이프라인 생성 AWS CloudFormation</u>
- <u>예 2: AWS CloudFormation을 사용하여 Amazon S3 파이프라인 생성</u>

예제 1:를 사용하여 AWS CodeCommit 파이프라인 생성 AWS

CloudFormation

이 연습에서는 AWS CloudFormation 콘솔을 사용하여 CodeCommit 소스 리포지토리에 연결된 파이 프라인을 포함하는 인프라를 생성하는 방법을 보여줍니다. 이 자습서에서는 제공된 샘플 템플릿 파 일을 사용하여 아티팩트 스토어, 파이프라인 및 Amazon CloudWatch Events 규칙과 같은 변경 감지 리소스가 포함된 리소스 스택을 생성합니다. 에서 리소스 스택을 생성한 후 AWS CodePipeline 콘솔 에서 파이프라인을 볼 AWS CloudFormation수 있습니다. 파이프라인은 CodeCommit 소스 단계와 CodeDeploy 배포 단계가 있는 2단계 파이프라인입니다.

사전 조건:

AWS CloudFormation 샘플 템플릿과 함께 사용하려면 다음 리소스를 생성해야 합니다.

- 소스 리포지토리가 생성되어 있어야 합니다. 에서 생성한 AWS CodeCommit 리포지토리를 사용할 수 있습니다<u>자습서: 간단한 파이프라인 생성(CodeCommit 리포지토리)</u>.
- CodeDeploy 애플리케이션 및 배포 그룹을 생성해야 합니다. <u>자습서: 간단한 파이프라인 생성</u> (CodeCommit 리포지토리)에 생성한 CodeDeploy 리소스를 사용할 수 있습니다.
• 다음 링크 중 하나를 선택하여 파이프라인 생성을 위한 샘플 AWS CloudFormation 템플릿 파일을 다 운로드합니다. YAML | JSON

파일의 압축을 풀고 로컬 컴퓨터에 저장합니다.

• SampleApp_Linux.zip 샘플 애플리케이션 파일을 다운로드합니다.

에서 파이프라인 생성 AWS CloudFormation

- <u>SampleApp_Linux.zip</u>에서 파일의 압축을 풀고 AWS CodeCommit 리포지토리에 파일을 업로드합 니다. 압축을 푼 파일을 리포지토리의 루트 디렉터리에 업로드해야 합니다. <u>2단계: CodeCommit</u> 리포지토리에 샘플 코드 추가의 지침에 따라 파일을 리포지토리로 푸시할 수 있습니다.
- 2. AWS CloudFormation 콘솔을 열고 스택 생성을 선택합니다. 새 리소스 사용(표준)(With new resources (standard))을 선택합니다.
- 템플릿 지정에서 템플릿 업로드를 선택합니다. 파일 선택을 선택한 후 로컬 컴퓨터에서 템플릿 파 일을 선택합니다. Next(다음)를 선택합니다.
- 스택 이름에 파이프라인의 이름을 입력합니다. 샘플 템플릿에 지정된 파라미터가 표시됩니다. 다 음 파라미터를 입력합니다.
 - a. ApplicationName에 CodeDeploy 애플리케이션의 이름을 입력합니다.
 - b. BetaFleet에 CodeDeploy 배포 그룹 이름을 입력합니다.
 - c. BranchName에서 사용할 리포지토리 브랜치를 입력합니다.
 - d. RepositoryName에서 CodeCommit 소스 리포지토리의 이름을 입력합니다.
- 5. Next(다음)를 선택합니다. 다음 페이지에서 기본값을 적용한 후 다음을 선택합니다.
- 기능에서 이 IAM 리소스를 생성할 AWS CloudFormation 수 있음을 승인합니다를 선택한 다음 스 택 생성을 선택합니다.
- 7. 스택 생성을 완료한 후 이벤트 목록에서 오류를 확인합니다.

문제 해결

에서 파이프라인을 생성하는 IAM 사용자는 파이프라인에 대한 리소스를 생성할 AWS CloudFormation 수 있는 추가 권한이 필요할 수 있습니다. 가 CodeCommit 파이프라인에 필요한 Amazon CloudWatch Events 리소스를 AWS CloudFormation 생성하도록 허용하려면 정책에 다 음 권한이 필요합니다. }

"Effect": "Allow",
"Action": [
"events:PutRule",
"events:PutEvents",
"events:PutTargets",
"events:DeleteRule",
"events:RemoveTargets",
"events:DescribeRule"
],
"Resource": " <i>resource_ARN</i> "

8. 에 로그인 AWS Management Console 하고 <u>https://console.aws.amazon.com/codepipeline/</u>:// https://https://https://https://i/httpsCodePipeline.com.com.com.com.

파이프라인에서 파이프라인을 선택하고 보기를 선택합니다. 다이어그램은 파이프라인 소스 및 배 포 단계를 보여줍니다.

Note

생성된 파이프라인을 보려면 AWS CloudFormation의 스택에 대한 리소스 탭에서 논 리적 ID 열을 찾으세요. 파이프라인의 물리적 ID 열에 있는 이름을 기록해 둡니다. CodePipeline에서는 스택을 생성한 리전의 동일한 물리적 ID(파이프라인 이름)로 파이프 라인을 볼 수 있습니다.

 소스 리포지토리에서 변경 사항을 커밋하고 푸시합니다. 변경 감지 리소스가 변경 사항을 선택하 면 파이프라인이 시작됩니다.

예 2: AWS CloudFormation을 사용하여 Amazon S3 파이프라인 생성

이 연습에서는 AWS CloudFormation 콘솔을 사용하여 Amazon S3 소스 버킷에 연결된 파이프라인을 포함하는 인프라를 생성하는 방법을 보여줍니다. 샘플 템플릿 파일을 사용하여 소스 버킷, 아티팩트 스 토어, 파이프라인 및 Amazon CloudWatch Events 규칙 및 CloudTrail 추적과 같은 변경 감지 리소스가 포함된 리소스 스택을 만듭니다. 에서 리소스 스택을 생성한 후 AWS CodePipeline 콘솔에서 파이프라 인을 볼 AWS CloudFormation수 있습니다. 파이프라인은 Amazon S3 소스 단계와 CodeDeploy 배포 단계가 있는 2단계 파이프라인입니다.

사전 조건:

AWS CloudFormation 샘플 템플릿과 함께 사용하려면 다음 리소스가 있어야 합니다.

- 인스턴스에 CodeDeploy 에이전트를 설치한 Amazon EC2 인스턴스를 생성해야 합니다. CodeDeploy 애플리케이션 및 배포 그룹을 생성해야 합니다. <u>자습서: 간단한 파이프라인 생성</u> (CodeCommit 리포지토리)에 생성한 Amazon EC2 및 CodeDeploy 리소스를 사용합니다.
- 다음 링크를 선택하여 Amazon S3 소스로 파이프라인을 생성하기 위한 샘플 AWS CloudFormation 템플릿 파일을 다운로드합니다.
 - 파이프라인에 대한 샘플 템플릿을 YAML 또는 JSON 형식으로 다운로드합니다.
 - CloudTrail 버킷 및 추적을 위한 샘플 템플릿(YAML | JSON 형식)을 다운로드합니다.
 - 파일의 압축을 풀고 로컬 컴퓨터에 저장합니다.
- SampleApp_Linux.zip 샘플 애플리케이션 파일을 다운로드합니다.

zip 파일을 로컬 컴퓨터에 저장합니다. 스택을 만든 후에 zip 파일을 업로드합니다.

에서 파이프라인 생성 AWS CloudFormation

- 1. AWS CloudFormation 콘솔을 열고 스택 생성을 선택합니다. 새 리소스 사용(표준)(With new resources (standard))을 선택합니다.
- 템플릿 선택 페이지에서 템플릿 업로드를 선택합니다. 파일 선택을 선택한 후 로컬 컴퓨터에서 템 플릿 파일을 선택합니다. Next(다음)를 선택합니다.
- 스택 이름에 파이프라인의 이름을 입력합니다. 샘플 템플릿에 지정된 파라미터가 표시됩니다. 다 음 파라미터를 입력합니다.
 - a. ApplicationName에 CodeDeploy 애플리케이션의 이름을 입력합니다. DemoApplication 기 본 이름을 바꿀 수 있습니다.
 - b. BetaFleet에 CodeDeploy 배포 그룹 이름을 입력합니다. DemoFleet 기본 이름을 바꿀 수 있 습니다.
 - c. SourceObjectKey에서 SampleApp_Linux.zip을 입력합니다. 이 파일은 템플릿에서 버킷 과 파이프라인을 생성하는 즉시 버킷에 업로드됩니다.
- 4. Next(다음)를 선택합니다. 다음 페이지에서 기본값을 적용한 후 다음을 선택합니다.
- 5. 기능에서 이 IAM 리소스를 생성할 AWS CloudFormation 수 있음을 승인합니다를 선택한 다음 스 택 생성을 선택합니다.
- 6. 스택 생성을 완료한 후 이벤트 목록에서 오류를 확인합니다.

문제 해결

에서 파이프라인을 생성하는 IAM 사용자는 파이프라인에 대한 리소스를 생성하는 데 추가 권 한이 필요할 AWS CloudFormation 수 있습니다. 가 Amazon S3 파이프라인에 필요한 Amazon CloudWatch Events 리소스를 AWS CloudFormation 생성하도록 허용하려면 정책에 다음 권한이 필요합니다. Amazon S3

```
{
    "Effect": "Allow",
    "Action": [
        "events:PutRule",
        "events:PutEvents",
        "events:PutTargets",
        "events:DeleteRule",
        "events:RemoveTargets",
        "events:DescribeRule"
    ],
    "Resource": "resource_ARN"
}
```

7. 의 스택에 대한 리소스 AWS CloudFormation탭에서 스택에 대해 생성된 리소스를 확인합니다.

Note

생성된 파이프라인을 보려면 AWS CloudFormation의 스택에 대한 리소스 탭에서 논 리적 ID 열을 찾으세요. 파이프라인의 물리적 ID 열에 있는 이름을 기록해 둡니다. CodePipeline에서는 스택을 생성한 리전의 동일한 물리적 ID(파이프라인 이름)로 파이프 라인을 볼 수 있습니다.

이름에 sourcebucket 레이블이 있는 S3 버킷을 선택합니다(예: s3-cfn-codepipelinesourcebucket-y04EXAMPLE.). 파이프라인의 아티팩트 버킷을 선택하지 않습니다.

리소스가 AWS CloudFormation에서 새로 생성되었으므로 원본 버킷이 비어 있습니다. Amazon S3 콘솔을 열고 sourcebucket 버킷을 선택합니다. 업로드를 선택하고 지침에 따라 SampleApp_Linux.zip .zip 파일을 업로드합니다.

Note

Amazon S3가 파이프라인의 소스 공급자인 경우, 단일 .zip 파일로 패키지된 모든 소스 파 일을 버킷에 업로드해야 합니다. 그렇지 않으면 소스 작업이 실패합니다.

8. 에 로그인 AWS Management Console 하고 <u>https://console.aws.amazon.com/codepipeline/</u>:// https://https://https://i/httpsCodePipeline.com.com.com.com.

파이프라인에서 파이프라인을 선택한 후 보기를 선택합니다. 다이어그램은 파이프라인 소스 및 배포 단계를 보여줍니다.

9. 다음 절차의 단계를 완료하여 AWS CloudTrail 리소스를 생성합니다.

에서 AWS CloudTrail 리소스 생성 AWS CloudFormation

- 1. AWS CloudFormation 콘솔을 열고 스택 생성을 선택합니다.
- 템플릿 선택 페이지에서 Amazon S3에 템플릿 업로드를 선택합니다. 찾아보기를 선택한 다음 로 컬 컴퓨터의 AWS CloudTrail 리소스에 대한 템플릿 파일을 선택합니다. Next(다음)를 선택합니다.
- 스택 이름에 리소스 스택의 이름을 입력합니다. 샘플 템플릿에 지정된 파라미터가 표시됩니다. 다 음 파라미터를 입력합니다.
 - SourceObjectKey에서 샘플 애플리케이션의 zip 파일에 대한 기본값을 적용합니다.
- 4. Next(다음)를 선택합니다. 다음 페이지에서 기본값을 적용한 후 다음을 선택합니다.
- 5. 기능에서 이 IAM 리소스를 생성할 AWS CloudFormation 수 있음을 승인합니다를 선택한 다음 생성을 선택합니다.
- 6. 스택 생성을 완료한 후 이벤트 목록에서 오류를 확인합니다.

가 Amazon S3 파이프라인에 필요한 CloudTrail 리소스를 AWS CloudFormation 생성하도록 허용 하려면 정책에 다음 권한이 필요합니다.

```
{
    "Effect": "Allow",
    "Action": [
        "cloudtrail:CreateTrail",
        "cloudtrail:DeleteTrail",
        "cloudtrail:StartLogging",
        "cloudtrail:StopLogging",
        "cloudtrail:PutEventSelectors"
}
```

```
],
"Resource": "resource_ARN"
}
```

7. 에 로그인 AWS Management Console 하고 <u>https://console.aws.amazon.com/codepipeline/</u>:// https://https://https://i/httpsCodePipeline.com.com.com.com.

파이프라인에서 파이프라인을 선택한 후 보기를 선택합니다. 다이어그램은 파이프라인 소스 및 배포 단계를 보여줍니다.

 소스 버킷에서 변경 사항을 커밋하고 푸시합니다. 변경 감지 리소스가 변경 사항을 선택하면 파이 프라인이 시작됩니다.

자습서: AWS CloudFormation 배포 작업의 변수를 사용하는 파이프 라인 생성

이 자습서에서는 AWS CodePipeline 콘솔을 사용하여 배포 작업이 포함된 파이프라인을 생성합니다. 파이프라인이 실행되면 템플릿은 스택을 생성하고 outputs 파일도 생성합니다. 스택 템플릿에서 생 성된 출력은 CodePipeline에서 AWS CloudFormation 작업에 의해 생성된 변수입니다.

템플릿에서 스택을 생성하는 작업에서 변수 네임스페이스를 지정합니다. 그런 다음 outputs 파일에 의해 생성된 변수는 후속 작업에서 사용할 수 있습니다. 이 예제에서는 AWS CloudFormation 작업에서 생성된 StackName 변수를 기반으로 변경 세트를 생성합니다. 수동 승인 후 변경 세트를 실행한 다음 StackName 변수를 기반으로 스택을 삭제하는 스택 삭제 작업을 생성합니다.

A Important

파이프라인 생성의 일환으로 CodePipeline은 고객이 제공하는 S3 아티팩트 버킷을 아티팩트 에 사용합니다. (이는 S3 소스 작업에 사용되는 버킷과 다릅니다.) S3 아티팩트 버킷이 파이프 라인의 계정과 다른 계정에 있는 경우 S3 아티팩트 버킷을 안전하고 신뢰할 수 AWS 계정 있는 에서 소유해야 합니다.

주제

- <u>사전 조건: AWS CloudFormation 서비스 역할 및 CodeCommit 리포지토리 생성</u>
- 1단계: 샘플 AWS CloudFormation 템플릿 다운로드, 편집 및 업로드
- 2단계: 파이프라인 생성
- <u>3단계: AWS CloudFormation 배포 작업을 추가하여 변경 세트 생성</u>

- 4단계: 수동 승인 작업 추가
- <u>5단계: CloudFormation 배포 작업을 추가하여 변경 세트 실행</u>
- 6단계: CloudFormation 배포 작업을 추가하여 스택 삭제

사전 조건: AWS CloudFormation 서비스 역할 및 CodeCommit 리포지토리 생성

다음 항목이 있어야 합니다.

- CodeCommit 리포지토리. 에서 생성한 AWS CodeCommit 리포지토리를 사용할 수 있습니다<u>자습서:</u> 간단한 파이프라인 생성(CodeCommit 리포지토리).
- 이 예제에서는 템플릿에서 Amazon DocumentDB 스택을 생성합니다. Amazon DocumentDB에 대 해 다음 권한이 있는 AWS CloudFormation 서비스 역할을 생성하려면 AWS Identity and Access Management (IAM)을 사용해야 합니다.

```
"rds:DescribeDBClusters",
"rds:CreateDBCluster",
"rds:DeleteDBCluster",
"rds:CreateDBInstance"
```

1단계: 샘플 AWS CloudFormation 템플릿 다운로드, 편집 및 업로드

샘플 AWS CloudFormation 템플릿 파일을 다운로드하여 CodeCommit 리포지토리에 업로드합니다.

- 1. 해당 리전의 샘플 템플릿으로 이동합니다. 예를 들어의 테이블<u>https://docs.aws.amazon.com/</u>

 <u>documentdb/latest/developerguide/quick_start_cfn.html#quick_start_cfn-launch_stack</u>을 사용하여

 리전을 선택하고 템플릿을 다운로드합니다. Amazon DocumentDB 클러스터용 템플릿을 다운로

 드합니다. 파일 이름은 documentdb_full_stack.yaml입니다.
- 2. documentdb_full_stack.yaml 파일의 압축을 풀고 텍스트 편집기에서 엽니다. 다음과 같이 변경합니다.
 - a. 이 예제에서는 템플릿의 Parameters 섹션에 다음 Purpose: 파라미터를 추가합니다.

Purpose: Type: String Default: testing AllowedValues: - testing

```
- production
```

Description: The purpose of this instance.

b. 이 예제에서는 템플릿의 Outputs: 섹션에 다음 StackName 출력을 추가합니다.

StackName: Value: !Ref AWS::StackName

 템플릿 파일을 AWS CodeCommit 리포지토리에 업로드합니다. 압축을 풀고 편집한 템플릿 파일 을 리포지토리의 루트 디렉터리에 업로드해야 합니다.

CodeCommit 콘솔을 사용하여 파일을 업로드하려면 다음과 같이 합니다.

- a. CodeCommit 콘솔을 열고 리포지토리 목록에서 해당 리포지토리를 선택합니다.
- b. 파일 추가를 선택한 후 파일 업로드를 선택합니다.
- c. 파일 선택을 선택한 다음 파일을 찾습니다. 사용자 이름과 이메일 주소를 입력하여 변경 사항
 을 커밋합니다. 변경 사항 커밋을 선택합니다.

파일은 리포지토리의 루트 수준에서 다음과 같아야 합니다.

documentdb_full_stack.yaml

2단계: 파이프라인 생성

이 단원에서는 다음 작업을 통해 파이프라인을 생성합니다.

- 소스 아티팩트가 템플릿 파일인 CodeCommit 작업이 있는 소스 단계입니다.
- 배포 작업이 있는 AWS CloudFormation 배포 단계입니다.

마법사가 생성한 소스 및 배포 단계의 각 작업에는 각각 변수 네임스페이스 SourceVariables 및 DeployVariables가 할당됩니다. 작업에 네임스페이스가 할당되어 있으므로 이 예제에 구성된 변수 를 다운스트림 작업에 사용할 수 있습니다. 자세한 내용은 <u>변수 참조</u> 단원을 참조하십시오.

마법사를 사용하여 파이프라인을 생성하려면

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https//https//https://https//ht

- 2. [Welcome] 페이지, [Getting started] 페이지 또는 [Pipelines] 페이지에서 Create pipeline(파이프라 인생성)을 선택합니다.
- 3. 1단계: 생성 옵션 선택 페이지의 생성 옵션에서 사용자 지정 파이프라인 빌드 옵션을 선택합니다. Next(다음)를 선택합니다.
- 4. 2단계: 파이프라인 설정 선택의 파이프라인 이름에 MyCFNDeployPipeline을 입력합니다.
- CodePipeline은 특성과 가격이 다른 V1 및 V2 유형 파이프라인을 제공합니다. V2 유형은 콘 솔에서 선택할 수 있는 유일한 유형입니다. 자세한 내용은 <u>파이프라인 유형</u>을 참조하세요. CodePipeline에 대한 요금 정보는 <u>요금</u>을 참조하세요.
- 6. 서비스 역할에서 다음 중 하나를 수행합니다.
 - 새 서비스 역할을 선택하여 CodePipeline이 IAM에 서비스 역할을 생성하도록 허용합니다.
 - Existing service role(기존 서비스 역할)을 선택합니다. 역할 이름의 목록에서 서비스 역할을 선택합니다.
- 7. 아티팩트 스토어에서 다음과 같이 합니다.
 - a. 파이프라인에 대해 선택한 리전의 파이프라인에 기본값으로 지정된 Amazon S3 아티팩트 버 킷과 같은 기본 아티팩트 스토어를 사용하려면 기본 위치를 선택합니다.
 - b. 파이프라인과 동일한 리전에 Amazon S3 아티팩트 버킷과 같이 이미 아티팩트 스토어가 있는 경우 사용자 지정 위치을 선택합니다.

Note

이는 소스 코드에 대한 소스 버킷이 아닙니다. 이 파이프라인은 아티팩트 스토어입니다. S3 버킷과 같은 개별 아티팩트 스토어는 각 파이프라인에 필요합니다. 파이프라인을 생성 하거나 편집할 때 파이프라인 리전에 아티팩트 버킷이 하나 있어야 하며 작업을 실행하는 AWS 리전당 하나의 아티팩트 버킷이 있어야 합니다. 자세한 내용은 <u>입력 및 출력 아티팩트</u> 및 <u>CodePipeline 파이프라인 구조 참조</u> 단원을 참조 하세요.

Next(다음)를 선택합니다.

- 8. 3단계: 소스 스테이지 추가에서 다음과 같이 합니다.
 - a. 소스 공급자에서 AWS CodeCommit을 선택합니다.

- b. 리포지토리 이름에서 <u>1단계: CodeCommit 리포지토리 생성</u>에서 생성한 CodeCommit 리포지 토리의 이름을 선택합니다.
- c. [Branch name]에서 가장 마지막 코드 업데이트가 포함된 브랜치의 이름을 선택합니다.

리포지토리 이름과 브랜치를 선택하면 이 파이프라인에 대해 생성될 Amazon CloudWatch Events 규칙이 표시됩니다.

Next(다음)를 선택합니다.

4단계: 빌드 스테이지 추가에서 빌드 스테이지 건너뛰기를 선택하고 건너뛰기를 다시 선택하여 경고 메시지를 수락합니다.

Next(다음)를 선택합니다.

 5단계: 테스트 단계 추가에서 테스트 단계 건너뛰기를 선택한 다음 건너뛰기를 다시 선택하여 경 고 메시지를 수락합니다.

Next(다음)를 선택합니다.

- 11. 6단계: 배포 단계 추가:
 - a. 작업 이름에서 배포를 선택합니다. 배포 공급자에서 CloudFormation을 선택합니다.
 - b. 작업 모드에서 스택 생성 또는 업데이트를 선택합니다.
 - c. 스택 이름에 스택의 이름을 입력합니다. 템플릿이 생성하는 스택의 이름입니다.
 - d. 출력 파일 이름에 출력 파일의 이름(예: outputs)을 입력합니다. 스택이 생성된 후 작업에 의 해 생성되는 파일의 이름입니다.
 - e. 고급을 확장합니다. 매개변수 재정의에서 템플릿 재정의를 키-값 쌍으로 입력합니다. 예를 들 어 이 템플릿에는 다음과 같은 재정의가 필요합니다.

```
{
  "DBClusterName": "MyDBCluster",
  "DBInstanceName": "MyDBInstance",
  "MasterUser": "UserName",
  "MasterPassword": "Password",
  "DBInstanceClass": "db.r4.large",
  "Purpose": "testing"}
```

재정의를 입력하지 않으면 템플릿에서 기본값으로 스택을 생성합니다.

f. Next(다음)를 선택합니다.

g. 7단계: 검토에서 파이프라인 생성을 선택합니다. 파이프라인 단계를 보여주는 다이어그램이 표시되어야 합니다. 파이프라인이 실행되도록 허용합니다. 2단계 파이프라인이 완료되어 단 계를 추가할 준비가 되었습니다.

3단계: AWS CloudFormation 배포 작업을 추가하여 변경 세트 생성

파이프라인에서 수동 승인 작업 전에가 변경 세트를 AWS CloudFormation 생성할 수 있는 다음 작업을 생성합니다.

1. https://console.aws.amazon.com/codepipeline/에서 CodePipeline 콘솔을 엽니다.

파이프라인에서 파이프라인을 선택하고 보기를 선택합니다. 다이어그램은 파이프라인 소스 및 배 포 단계를 보여줍니다.

- 2. 파이프라인을 편집하거나 편집 모드에서 파이프라인을 계속 표시합니다.
- 3. 배포 단계를 편집하려면 선택합니다.
- 이전 작업에서 생성된 스택에 대한 변경 세트를 생성하는 배포 작업을 추가합니다. 단계의 기존 작 업 뒤에 이 작업을 추가합니다.
 - a. 작업 이름에 Change_Set를 입력합니다. 작업 공급자에서 AWS CloudFormation 을 선택합니다.
 - b. 입력 아티팩트에서 SourceArtifact를 선택합니다.
 - c. 작업 모드에서 변경 사항 세트 생성 또는 교체를 선택합니다.
 - d. 스택 이름에 다음과 같이 변수 구문을 입력합니다. 변경 세트가 생성되는 스택의 이름입니다.
 여기서 기본 네임스페이스 DeployVariables는 작업에 할당됩니다.

#{DeployVariables.StackName}

e. 변경 세트 이름에서 변경 세트의 이름을 입력합니다.

my-changeset

f. 파라미터 재정의에서 Purpose 파라미터를 testing에서 production으로 변경합니다.

```
{
  "DBClusterName": "MyDBCluster",
  "DBInstanceName": "MyDBInstance",
  "MasterUser": "UserName",
```

```
"MasterPassword": "Password",
"DBInstanceClass": "db.r4.large",
"Purpose": "production"}
```

g. 완료를 선택하여 작업을 저장합니다.

4단계: 수동 승인 작업 추가

파이프라인에서 수동 승인 작업을 생성합니다.

- 1. 파이프라인을 편집하거나 편집 모드에서 파이프라인을 계속 표시합니다.
- 2. 배포 단계를 편집하려면 선택합니다.
- 변경 세트를 생성하는 배포 작업 다음에 수동 승인 작업을 추가합니다. 이 작업을 통해 파이프라인 이 변경 세트를 실행 AWS CloudFormation 하기 전에에서 생성된 리소스 변경 세트를 확인할 수 있습니다.

5단계: CloudFormation 배포 작업을 추가하여 변경 세트 실행

파이프라인에서 수동 승인 작업 후가 변경 세트를 AWS CloudFormation 실행할 수 있는 다음 작업을 생성합니다.

1. <u>https://console.aws.amazon.com/codepipeline/</u>에서 CodePipeline 콘솔을 엽니다.

파이프라인에서 파이프라인을 선택하고 보기를 선택합니다. 다이어그램은 파이프라인 소스 및 배 포 단계를 보여줍니다.

- 2. 파이프라인을 편집하거나 편집 모드에서 파이프라인을 계속 표시합니다.
- 3. 배포 단계를 편집하려면 선택합니다.
- 4. 이전 수동 작업에서 승인된 변경 세트를 실행할 배포 작업을 추가합니다.
 - a. 작업 이름에 Execute_Change_Set를 입력합니다. 작업 공급자에서 AWS CloudFormation을 선택합니다.
 - b. 입력 아티팩트에서 SourceArtifact를 선택합니다.
 - c. 작업 모드에서 변경 세트 실행을 선택합니다.
 - d. 스택 이름에 다음과 같이 변수 구문을 입력합니다. 변경 세트가 생성되는 스택의 이름입니다.

#{DeployVariables.StackName}

e. 변경 세트 이름에 이전 작업에서 생성한 변경 세트의 이름을 입력합니다.

my-changeset

- f. 완료를 선택하여 작업을 저장합니다.
- g. 파이프라인 실행을 계속합니다.

6단계: CloudFormation 배포 작업을 추가하여 스택 삭제

파이프라인에서가 출력 파일의 변수에서 스택 이름을 AWS CloudFormation 가져오고 스택을 삭제할 수 있는 최종 작업을 생성합니다.

1. https://console.aws.amazon.com/codepipeline/에서 CodePipeline 콘솔을 엽니다.

파이프라인에서 파이프라인을 선택하고 보기를 선택합니다. 다이어그램은 파이프라인 소스 및 배 포 단계를 보여줍니다.

- 2. 파이프라인을 편집하도록 선택합니다.
- 3. 배포 단계를 편집하려면 선택합니다.
- 4. 스택을 삭제할 배포 작업을 추가합니다.
 - a. 작업 이름에서 DeleteStack을 선택합니다. 배포 공급자에서 CloudFormation을 선택합니다.
 - b. 작업 모드에서 스택 삭제를 선택합니다.
 - c. 스택 이름에 다음과 같이 변수 구문을 입력합니다. 작업이 삭제하는 스택의 이름입니다.
 - d. 완료를 선택하여 작업을 저장합니다.
 - e. 저장을 선택하여 파이프라인을 저장합니다.

파이프라인은 저장될 때 실행됩니다.

자습서: CodePipeline을 사용한 Amazon ECS 표준 배포

이 자습서에서는 CodePipeline을 통해 Amazon ECS를 사용하여 종단 간에 CD(Continuous Deployment) 파이프라인을 생성하는 방법을 설명합니다.

A Important

콘솔에서 파이프라인을 생성하는 과정에서 CodePipeline은 S3 아티팩트 버킷을 아티팩트에 사용합니다. (이는 S3 소스 작업에 사용되는 버킷과 다릅니다.) S3 아티팩트 버킷이 파이프라 인의 계정과 다른 계정에 있는 경우 S3 아티팩트 버킷을 안전하고 신뢰할 수 AWS 계정 있는에 서 소유해야 합니다.

Note

이 자습서는 CodePipeline의 Amazon ECS 표준 배포 작업에 대한 것입니다. CodePipeline 의 Amazon ECS에서 CodeDeploy로의 블루/그린 배포 작업을 사용하는 자습서는 <u>자습서:</u> <u>Amazon ECR 소스 및 ECS-to-CodeDeploy 배포를 사용하여 파이프라인 생성</u> 단원을 참조하 세요.

Note

이 자습서는 소스 작업이 있는 CodePipeline에 대한 Amazon ECS 표준 배포 작업을 위한 것입 니다. CodePipeline의 ECRBuildAndPublish 빌드 작업과 함께 Amazon ECSstandard 배포 작 업을 사용하여 이미지를 푸시하는 자습서는 섹션을 참조하세요<u>자습서: CodePipeline을 사용하</u> 여 Docker 이미지를 빌드하고 Amazon ECR에 푸시(V2 유형).

사전 조건

이 자습서를 이용하여 CD 파이프라인을 만들려면 먼저 몇 가지 리소스를 갖춰야 합니다. 다음은 시작 하기 위해 필요한 항목입니다.

1 Note

이러한 모든 리소스는 동일한 AWS 리전 내에 생성되어야 합니다.

• Dockerfile 및 애플리케이션 소스에 대한 소스 컨트롤 리포지토리(이 자습서에서는 CodeCommit 사용). 자세한 내용은 AWS CodeCommit 사용 설명서의 CodeCommit 리포지토리 생성을 참조하세요.

- Dockerfile 및 애플리케이션 소스에서 만든 이미지를 포함하는 도커 이미지 리포지토리(이 자습서에 서는 Amazon ECR 사용). 자세한 내용은 Amazon Elastic Container Registry 사용 설명서의 <u>리포지</u> 토리 생성 및 이미지 푸시를 참조하세요.
- 해당 이미지 리포지토리에서 호스팅되는 도커 이미지를 참조하는 Amazon ECS 작업 정의. 자세한 내용을 알아보려면 Amazon Elastic Container Service 개발자 안내서의 <u>작업 정의 생성</u>을 참조하세 요.

▲ Important

CodePipeline의 Amazon ECS 표준 배포 작업은 Amazon ECS 서비스에서 사용하는 수정 버 전을 기반으로 작업 정의의 자체 수정 버전을 생성합니다. Amazon ECS 서비스를 업데이트 하지 않고 작업 정의에 대한 새 수정 사항을 생성하면 배포 작업에서 해당 수정 사항을 무시 합니다.

다음은 이 자습서에서 사용된 샘플 태스크 정의입니다. name 및 family에 사용하는 값은 빌드 사양 파일의 다음 단계에서 사용됩니다.

```
{
  "ipcMode": null,
  "executionRoleArn": "role_ARN",
  "containerDefinitions": [
    {
      "dnsSearchDomains": null,
      "environmentFiles": null,
      "logConfiguration": {
        "logDriver": "awslogs",
        "secretOptions": null,
        "options": {
          "awslogs-group": "/ecs/hello-world",
          "awslogs-region": "us-west-2",
          "awslogs-stream-prefix": "ecs"
        }
      },
      "entryPoint": null,
      "portMappings": [
        {
          "hostPort": 80,
          "protocol": "tcp",
          "containerPort": 80
```

```
}
    ],
    "command": null,
    "linuxParameters": null,
    "cpu": 0,
    "environment": [],
    "resourceRequirements": null,
    "ulimits": null,
    "dnsServers": null,
    "mountPoints": [],
    "workingDirectory": null,
    "secrets": null,
    "dockerSecurityOptions": null,
    "memory": null,
    "memoryReservation": 128,
    "volumesFrom": [],
    "stopTimeout": null,
    "image": "image_name",
    "startTimeout": null,
    "firelensConfiguration": null,
    "dependsOn": null,
    "disableNetworking": null,
    "interactive": null,
    "healthCheck": null,
    "essential": true,
    "links": null,
    "hostname": null,
    "extraHosts": null,
    "pseudoTerminal": null,
    "user": null,
    "readonlyRootFilesystem": null,
    "dockerLabels": null,
    "systemControls": null,
    "privileged": null,
    "name": "hello-world"
  }
],
"placementConstraints": [],
"memory": "2048",
"taskRoleArn": null,
"compatibilities": [
  "EC2",
  "FARGATE"
],
```

```
"taskDefinitionArn": "ARN",
"family": "hello-world",
"requiresAttributes": [],
"pidMode": null,
"requiresCompatibilities": [
"FARGATE"
],
"networkMode": "awsvpc",
"cpu": "1024",
"revision": 1,
"status": "ACTIVE",
"inferenceAccelerators": null,
"proxyConfiguration": null,
"volumes": []
}
```

• 앞에서 언급한 해당 작업 정의를 사용하는 서비스를 실행 중인 Amazon ECS 클러스터. 자세한 내용 은 Amazon Elastic Container Service 개발자 안내서의 클러스터 생성 및 서비스 생성을 참조하세요.

이러한 사전 조건을 모두 갖췄으면 이제 자습서를 이용하여 CD 파이프라인을 만들 수 있습니다.

1단계: 소스 리포지토리에 빌드 사양 파일 추가

이 자습서에서는 CodeBuild를 사용하여 도커 이미지를 생성하고 Amazon ECR에 이미지를 푸시합니다. buildspec.yml 파일을 소스 코드 리포지토리에 추가하여 CodeBuild가 작업을 수행하는 방식을 지정합니다. 아래의 빌드 사양 예제는 다음을 수행합니다.

- 빌드 전 단계:
 - Amazon ECR에 로그인합니다.
 - 리포지토리 URI를 해당 ECR 이미지로 설정하고 이미지 태그와 소스의 Git 커밋 ID의 첫 7자를 추 가합니다.
- 빌드 단계:
 - 도커 이미지를 빌드하고 latest 및 Git 커밋 ID로 이미지에 태그를 지정합니다.
- 빌드 후 단계:
 - ECR 리포지토리에 두 태그와 함께 이미지를 푸시합니다.
 - Amazon ECS 서비스의 컨테이너 이름과 이미지 및 태그를 포함하는 imagedefinitions.json 파일을 빌드 루트에 씁니다. CD 파이프라인의 배포 단계에서는 이 정보를 사용하여 개정된

서비스 작업 정의를 생성한 후 새로운 작업 정의를 사용하도록 서비스를 업데이트합니다. imagedefinitions.json 파일은 ECS 작업자에게 필요합니다.

이 샘플 텍스트를 붙여 넣어 buildspec.yml 파일을 생성하고 이미지 및 작업 정의의 값을 바꿉니다. 이 텍스트는 예제 계정 ID인 111122223333을 사용합니다.

```
version: 0.2
phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws --version
      - aws ecr get-login-password --region $AWS_DEFAULT_REGION | docker login --
username AWS --password-stdin 111122223333.dkr.ecr.us-west-2.amazonaws.com
      - REPOSITORY_URI=012345678910.dkr.ecr.us-west-2.amazonaws.com/hello-world
      - COMMIT_HASH=$(echo $CODEBUILD_RESOLVED_SOURCE_VERSION | cut -c 1-7)
      - IMAGE_TAG=${COMMIT_HASH:=latest}
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t $REPOSITORY_URI:latest .
      - docker tag $REPOSITORY_URI:latest $REPOSITORY_URI:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker images...
      - docker push $REPOSITORY_URI:latest
      docker push $REPOSITORY_URI:$IMAGE_TAG
      - echo Writing image definitions file...
      - printf '[{"name":"hello-world","imageUri":"%s"}]' $REPOSITORY_URI:$IMAGE_TAG >
 imagedefinitions.json
artifacts:
    files: imagedefinitions.json
```

이 자습서의 경우 Amazon ECS 서비스에서 사용하는 <u>사전 조건</u>에서 제공된 샘플 태스크 정의에 대해 빌드 사양을 썼습니다. REPOSITORY_URI 값은 image 리포지토리(이미지 태그 제외)에 해당하고, 파 일 끝부분의 *hello-world* 값은 서비스 작업 정의의 컨테이너 이름에 해당합니다. buildspec.yml 파일을 해당 소스 리포지토리에 추가하려면

- 1. 텍스트 편집기를 연 후 위의 빌드 사양을 복사하여 새 파일에 붙여 넣습니다.
- REPOSITORY_URI 값(012345678910.dkr.ecr.us-west-2.amazonaws.com/helloworld)을 해당 도커 이미지의 Amazon ECR 리포지토리 URI(이미지 태그 제외)로 바꿉니다. hello-world를 해당 서비스 작업 정의에서 도커 이미지를 참조하는 컨테이너 이름으로 바꿉니 다.
- 3. buildspec.yml 파일을 커밋한 후 소스 리포지토리에 푸시합니다.
 - a. 파일을 추가합니다.

git add .

b. 변경 내용을 커밋합니다.

git commit -m "Adding build specification."

c. 커밋을 푸시합니다.

git push

2단계: CD(Continuous Deployment) 파이프라인 만들기

CodePipeline 마법사를 사용하여 파이프라인 단계를 생성하고, 소스 리포지토리를 ECS 서비스에 연 결합니다.

파이프라인을 생성하려면

- 1. https://console.aws.amazon.com/codepipeline/에서 CodePipeline 콘솔을 엽니다.
- 2. [Welcome] 페이지에서 [Create pipeline]을 선택합니다.

CodePipeline을 처음으로 사용하는 경우 Welcome 페이지 대신 소개 페이지가 나타납니다. 지금 시작을 선택합니다.

- 1단계: 생성 옵션 선택 페이지의 생성 옵션에서 사용자 지정 파이프라인 빌드 옵션을 선택합니다. Next(다음)를 선택합니다.
- 2단계: 파이프라인 설정 선택의 파이프라인 이름에 파이프라인의 이름을 입력합니다. 이 자습서에 서 파이프라인 이름은 hello-world입니다.

- 5. 파이프라인 유형에서는 기본 선택을 V2로 유지합니다. 파이프라인 유형은 특성과 가격이 다릅니 다. 자세한 내용은 <u>파이프라인 유형</u> 단원을 참조하십시오. Next(다음)를 선택합니다.
- 6. 3단계: 소스 단계 추가 페이지의 소스 공급자에서 AWS CodeCommit를 선택합니다.
 - a. 리포지토리 이름에서 파이프라인의 소스 위치로 사용할 CodeCommit 리포지토리의 이름을 선택합니다.
 - b. 브랜치 이름에서 사용할 브랜치를 선택한 후 다음을 선택합니다.
- 7. 4단계: 빌드 단계 추가 페이지에서 빌드 공급자에서 AWS CodeBuild를 선택한 다음 프로젝트 생성을 선택합니다.
 - a. 프로젝트 이름에서 고유한 빌드 프로젝트 이름을 선택합니다. 이 자습서에서 프로젝트 이름 은 hello-world입니다.
 - b. 환경 이미지에서 이미지 관리를 선택합니다.
 - c. 운영 체제에서 Amazon Linux 2를 선택합니다.
 - d. 런타임에서 표준을 선택합니다.
 - e. 이미지에서 aws/codebuild/amazonlinux2-x86_64-standard:3.0를 선택합니다.
 - f. 이미지 버전 및 환경 유형의 경우 기본값을 사용합니다.
 - g. Docker 이미지를 빌드하거나 빌드에서 승격된 권한을 얻으려는 경우 이 플래그 활성화를 선 택합니다.
 - h. CloudWatch 로그를 선택 취소합니다. 고급을 확장해야 할 수도 있습니다.
 - i. Continue to CodePipeline(CodePipeline으로 계속)을 선택합니다.
 - j. Next(다음)를 선택합니다.

Note

마법사가 빌드 프로젝트를 위한 CodeBuild 서비스 역할(codebuild-*build-project-name*-service-role)을 생성합니다. 이 역할 이름은 나중에 Amazon ECR 권 한을 역할에 추가할 때 필요하므로 메모해 둡니다.

- 8. 5단계: 배포 단계 추가 페이지의 배포 공급자에서 Amazon ECS를 선택합니다.
 - a. 클러스터 이름에서 해당 서비스가 실행 중인 Amazon ECS 클러스터를 선택합니다. 이 자습 서에서는 클러스터가 default입니다.
 - b. 서비스 이름에서 업데이트할 서비스를 선택한 후 다음을 선택합니다. 이 자습서에서 서비스 이름은 hello-world입니다.

 6단계: 검토 페이지에서 파이프라인 구성을 검토하고 파이프라인 생성을 선택하여 파이프라인을 생성합니다.

Note

이제 파이프라인이 생성되었으며 다른 파이프라인 단계에서 이 파이프라인이 실행하려고 시도합니다. 하지만 마법사가 만든 기본 CodeBuild 역할이 buildspec.yml 파일에 포함 된 모든 명령을 실행할 수 있는 권한을 갖고 있지 않으므로 빌드 단계가 실패합니다. 다음 단계에서는 빌드 단계를 위한 권한을 추가합니다.

3단계: CodeBuild 역할에 Amazon ECR 권한 추가

CodePipeline 마법사가 CodeBuild 빌드 프로젝트를 위한 IAM 역할(codebuild-*build-project-name*-service-role)을 생성했습니다. 이 자습서에서 이름은 codebuild-hello-world-service-role입니다. buildspec.yml 파일은 Amazon ECR API 작업을 호출하기 때문에 이 역할은 이러한 Amazon ECR 호출을 할 수 있는 권한을 허용하는 정책이 있어야 합니다. 다음 절차에서는 적절한 권한을 역할에 연 결합니다.

CodeBuild 역할에 Amazon ECR 권한을 추가하려면

- 1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
- 2. 왼쪽 탐색 창에서 역할을 선택합니다.
- 3. 검색란에 codebuild-를 입력하고 CodePipeline 마법사가 생성한 역할을 선택합니다. 이 자습서에 서 역할 이름은 codebuild-hello-world-service-role입니다.
- 4. 요약 페이지에서 정책 연결을 선택합니다.
- 5. AmazonEC2ContainerRegistryPowerUser 정책 왼쪽의 상자를 선택하고 정책 연결을 선택합니다.

4단계: 파이프라인 테스트

파이프라인에는 end-to-end네이티브 AWS 지속적 배포를 실행하기 위한 모든 것이 있어야 합니다. 이 제 소스 리포지토리에 코드 변경을 푸시하여 파이프라인 기능을 테스트해 보겠습니다.

파이프라인을 테스트하려면

- 1. 구성된 소스 리포지토리에 대한 코드를 변경하고 커밋한 후 변경 사항을 푸시합니다.
- 2. https://console.aws.amazon.com/codepipeline/에서 CodePipeline 콘솔을 엽니다.

- 3. 목록에서 파이프라인을 선택합니다.
- 단계를 수행하면서 파이프라인 진행 상황을 관찰합니다. 파이프라인이 완료되고 코드 변경을 통 해 생성된 도커 이미지를 Amazon ECS 서비스가 실행해야 합니다.

자습서: Amazon ECR 소스 및 ECS-to-CodeDeploy 배포를 사용하 여 파이프라인 생성

이 자습서에서는 Docker 이미지를 지원하는 블루/그린 AWS CodePipeline 배포를 사용하여 컨테이너 애플리케이션을 배포하는 파이프라인을에 구성합니다. 블루/그린 배포에서는 트래픽을 다시 라우팅하 기 전에 이전 버전과 함께 새 버전의 애플리케이션을 시작하고 새 버전을 테스트할 수 있습니다. 또한 배포 프로세스를 모니터링하고 문제가 발생할 경우 신속하게 롤백할 수 있습니다.

A Important

파이프라인 생성의 일환으로 CodePipeline은 고객이 제공하는 S3 아티팩트 버킷을 아티팩트 에 사용합니다. (이는 S3 소스 작업에 사용되는 버킷과 다릅니다.) S3 아티팩트 버킷이 파이프 라인의 계정과 다른 계정에 있는 경우 S3 아티팩트 버킷을 안전하고 신뢰할 수 AWS 계정 있는 에서 소유해야 합니다.

Note

이 자습서는 CodePipeline인의 Amazon ECS에서 CodeDeploy로의 블루/그린 배포 작업을 위한 것입니다. CodePipeline의 Amazon ECS 표준 배포 작업을 사용하는 자습서는 <u>자습서:</u> CodePipeline을 사용한 Amazon ECS 표준 배포 단원을 참조하세요.

완료된 파이프라인은 이미지 변경 사항을 감지하고 Amazon ECR과 같은 이미지 리포지토리에 저장하 고 CodeDeploy를 사용하여 트래픽을 Amazon ECS 클러스터 및 로드 밸런서로 라우팅하고 배포합니 다. CodeDeploy는 리스너를 사용하여 AppSpec 파일에 지정된 업데이트된 컨테이너의 포트로 트래픽 을 다시 라우팅합니다. 블루/그린 배포에서 로드 밸런서, 프로덕션 리스너, 대상 그룹 및 Amazon ECS 애플리케이션을 사용하는 방법에 대한 자세한 내용은 <u>자습서: Amazon ECS 서비스 배포</u>를 참조하세 요.

또한 파이프라인도 Amazon ECS 작업 정의가 저장된 CodeCommit과 같은 소스 위치를 사용하도록 구 성됩니다. 이 자습서에서는 이러한 각 AWS 리소스를 구성한 다음 각 리소스에 대한 작업이 포함된 단 계로 파이프라인을 생성합니다. 지속적 전달 파이프라인은 소스 코드가 변경되거나 새 기본 이미지가 Amazon ECR에 업로드될 때마 다 컨테이너 이미지를 자동으로 빌드하고 배포합니다.

이 플로우는 다음 아티팩트를 사용합니다.

- Amazon ECR 이미지 리포지토리의 URI와 컨테이너 이름을 지정하는 도커 이미지 파일.
- 도커 이미지 이름, 컨테이너 이름, Amazon ECS 서비스 이름, 로드 밸런서 구성의 목록을 표시하는 Amazon ECS 작업 정의입니다.
- Amazon ECS 작업 정의 파일의 이름, 업데이트된 애플리케이션 컨테이너의 이름, CodeDeploy가 프 로덕션 트래픽을 다시 라우팅하는 컨테이너 포트를 지정하는 CodeDeploy AppSpec 파일입니다. 배 포 수명 주기 이벤트 후크 중 실행할 수 있는 Lambda 함수와 네트워크 구성을 선택적으로 지정할 수 도 있습니다.

Note

Amazon 이미지 리포지토리에 변경을 커밋하면 파이프라인 ECR 소스 작업이 해당 커밋에 대 한 imageDetail.json 파일을 생성합니다. imageDetail.json 파일에 대한 자세한 내용 은 Amazon ECS 블루/그린 배포 작업을 위한 imageDetail.json 파일 단원을 참조하십시오.

파이프라인을 생성 또는 편집하고 배포 단계를 위한 소스 아티팩트를 업데이트 또는 지정할 경우, 사용 하려는 최신 이름과 버전의 소스 아티팩트를 지정해야 합니다. 파이프라인을 설정한 후 이미지나 작업 정의를 변경하면 리포지토리에서 소스 결과물을 업데이트한 후 파이프라인에서 배포 단계를 편집해야 할 수 있습니다.

주제

- <u>사전 조건</u>
- 1단계: 이미지 생성 및 Amazon ECR 리포지토리로 푸시
- 2단계: 작업 정의 및 AppSpec 소스 파일 생성과 CodeCommit 리포지토리에 푸시
- 3단계: Application Load Balancer 및 대상 그룹 만들기
- 4단계: Amazon ECS 클러스터 및 서비스 생성
- 5단계: CodeDeploy 애플리케이션 및 배포 그룹 만들기(ECS 컴퓨팅 플랫폼)
- 6단계: 파이프라인 생성
- 7단계: 파이프라인 변경 및 배포 확인

사전 조건

다음 리소스를 이미 생성했어야 합니다.

- CodeCommit 리포지토리. 에서 생성한 AWS CodeCommit 리포지토리를 사용할 수 있습니다<u>자습서:</u> 간단한 파이프라인 생성(CodeCommit 리포지토리).
- 이 자습서에 나와 있는 대로 Amazon EC2 Linux 인스턴스를 시작하고 Docker를 설치하여 이미지를 생성합니다. 사용할 이미지가 이미 있으면 이 조건을 무시해도 됩니다.

1단계: 이미지 생성 및 Amazon ECR 리포지토리로 푸시

이 섹션에서는 Docker를 사용하여 이미지를 생성한 다음 AWS CLI 를 사용하여 Amazon ECR 리포지 토리를 생성하고 이미지를 리포지토리로 푸시합니다.

Note

사용할 이미지가 이미 있으면 단계를 건너뛰어도 됩니다.

이미지 생성

1. Docker를 설치한 Linux 인스턴스에 로그인합니다.

nginx용 이미지를 풀다운합니다. 이 명령은 nginx:latest 이미지를 제공합니다.

docker pull nginx

2. docker images을(를) 실행합니다. 목록에 이미지가 나타나야 합니다.

docker images

Amazon ECR 리포지토리를 생성하고 이미지를 푸시하려면

1. 이미지를 저장할 Amazon ECR 리포지토리를 생성합니다. 출력에서 repositoryUri를 기록합 니다.

aws ecr create-repository --repository-name nginx

```
{
    "repository": {
        "registryId": "aws_account_id",
        "repositoryName": "nginx",
        "repositoryArn": "arn:aws:ecr:us-east-1:aws_account_id:repository/nginx",
        "createdAt": 1505337806.0,
        "repositoryUri": "aws_account_id.dkr.ecr.us-east-1.amazonaws.com/nginx"
    }
}
```

2. 이전 단계의 repositoryUri 값을 사용하여 이미지에 태그를 지정합니다.

docker tag nginx:latest aws_account_id.dkr.ecr.us-east-1.amazonaws.com/nginx:latest

3. us-west-2 리전 및 111122223333 계정 ID의 예와 같이 aws ecr get-login-password 명령을 실 행합니다.

```
aws ecr get-login-password --region us-west-2 | docker login --username AWS --
password-stdin 111122223333.dkr.ecr.us-west-2.amazonaws.com/nginx
```

4. 이전 단계의 repositoryUri를 사용하여 Amazon ECR로 이미지를 푸시합니다.

docker push 111122223333.dkr.ecr.us-east-1.amazonaws.com/nginx:latest

2단계: 작업 정의 및 AppSpec 소스 파일 생성과 CodeCommit 리포지토리에 푸시

이 단원에서는 작업 정의 JSON 파일을 생성하고 Amazon ECS에 등록합니다. 그런 다음 CodeDeploy 에 대한 AppSpec 파일을 생성하고, Git 클라이언트를 사용하여 CodeCommit 리포지토리로 파일을 푸 시합니다.

이미지에 대한 작업 정의를 생성하려면

 다음 콘텐츠를 가진 taskdef.json이라는 파일을 생성합니다: image에 이미지 이름(예: nginx) 을 입력합니다. 파이프라인이 실행되면 이 값이 업데이트됩니다.

1 Note

작업 정의에 지정된 실행 역할에 AmazonECSTaskExecutionRolePolicy가 있는지 확 인합니다. 자세한 내용을 알아보려면 Amazon ECS 개발자 안내서의 <u>Amazon ECS 태스크</u> 실행 IAM 역할을 참조하세요.

```
{
    "executionRoleArn": "arn:aws:iam::account_ID:role/ecsTaskExecutionRole",
    "containerDefinitions": [
        ſ
            "name": "sample-website",
            "image": "nginx",
            "essential": true,
            "portMappings": [
                {
                     "hostPort": 80,
                     "protocol": "tcp",
                     "containerPort": 80
                }
            ]
        }
    ],
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "networkMode": "awsvpc",
    "cpu": "256",
    "memory": "512",
    "family": "ecs-demo"
}
```

2. taskdef.json 파일을 사용하여 작업 정의를 등록합니다.

aws ecs register-task-definition --cli-input-json file://taskdef.json

작업 정의가 등록된 후, 파일을 편집하여 이미지 이름을 제거하고 이미지 필드에
 <IMAGE1_NAME> 자리 표시자 텍스트를 포함시킵니다.

```
"executionRoleArn": "arn:aws:iam::account_ID:role/ecsTaskExecutionRole",
    "containerDefinitions": [
        {
            "name": "sample-website",
            "image": "<IMAGE1_NAME>",
            "essential": true,
            "portMappings": [
                {
                     "hostPort": 80,
                     "protocol": "tcp",
                     "containerPort": 80
                }
            ]
        }
    ],
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "networkMode": "awsvpc",
    "cpu": "256",
    "memory": "512",
    "family": "ecs-demo"
}
```

AppSpec 파일을 만들려면

• AppSpec 파일은 CodeDeploy 배포에 사용됩니다. 선택 필드를 포함하는 파일은 다음 형식을 사용 합니다.

```
version: 0.0
Resources:
    TargetService:
    Type: AWS::ECS::Service
    Properties:
        TaskDefinition: "task-definition-ARN"
        LoadBalancerInfo:
        ContainerName: "container-name"
        ContainerPort: container-port-number
# Optional properties
        PlatformVersion: "LATEST"
        NetworkConfiguration:
        AwsvpcConfiguration:
        AwsvpcConfiguration:
        AwsvpcConfiguration:
        AwsvpcConfiguration:
        AwsvpcConfiguration:
        AwsvpcConfiguration:
        AwsvpcConfiguration:
        AwsvpcConfiguration:
        ContainerPort: Container
        ContainerPort: ContainerPort
```

<pre>Subnets: ["subnet-name-1", "subnet-name-2"]</pre>	
SecurityGroups: [" <i>security-group</i> "]	
AssignPublicIp: " <u>ENABLED</u> "	
Hooks:	
 BeforeInstall: "BeforeInstallHookFunctionName" 	
- AfterInstall: "AfterInstallHookFunctionName"	
 AfterAllowTestTraffic: "AfterAllowTestTrafficHookFunctionName" 	
 BeforeAllowTraffic: "BeforeAllowTrafficHookFunctionName" 	
- AfterAllowTraffic: "AfterAllowTrafficHookEunctionName"	

AppSpec 파일에 대한 자세한 내용과 예제는 CodeDeploy AppSpec 파일 참조를 참조하세요.

다음 콘텐츠를 가진 appspec.yaml이라는 파일을 생성합니다: TaskDefinition에서 <TASK_DEFINITION> 자리 표시자 텍스트를 변경하지 마십시오. 파이프라인이 실행되면 이 값이 업데이트됩니다.

```
version: 0.0
Resources:
    - TargetService:
    Type: AWS::ECS::Service
    Properties:
    TaskDefinition: <TASK_DEFINITION>
    LoadBalancerInfo:
        ContainerName: "sample-website"
        ContainerPort: 80
```

CodeCommit 리포지토리에 파일을 푸시하려면

 CodeCommit 리포지토리에 파일을 푸시하거나 업로드합니다. 이러한 파일은 파이프라인 생성 마 법사가 CodePipeline에서 배포 작업을 위해 생성한 소스 아티팩트입니다. 파일은 로컬 디렉터리에 다음과 같이 나타납니다.

```
/tmp
|my-demo-repo
|-- appspec.yaml
|-- taskdef.json
```

- 2. 파일을 업로드하는 데 사용할 방법을 선택합니다.
 - a. 로컬 컴퓨터의 복제된 리포지토리에서 git 명령줄을 사용하려면 다음과 같이 합니다.

i. 디렉터리를 해당 로컬 리포지토리로 변경합니다.

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

ii. 다음 명령을 실행하여 모든 파일을 한 번에 스테이징합니다.

```
git add -A
```

iii. 다음 명령을 실행하여 커밋 메시지와 함께 파일을 커밋합니다.

git commit -m "Added task definition files"

iv. 다음 명령을 실행하여 로컬 리포지토리의 파일을 CodeCommit 리포지토리에 푸시합니 다.

git push

- b. CodeCommit 콘솔을 사용하여 파일을 업로드하려면 다음과 같이 합니다.
 - i. CodeCommit 콘솔을 열고 리포지토리 목록에서 해당 리포지토리를 선택합니다.
 - ii. 파일 추가를 선택한 후 파일 업로드를 선택합니다.
 - iii. 파일 선택을 선택한 후 파일을 찾습니다. 사용자 이름과 이메일 주소를 입력하여 변경 사 항을 커밋합니다. 변경 사항 커밋을 선택합니다.
 - iv. 업로드하려는 각 파일에 대해 이 단계를 반복합니다.

3단계: Application Load Balancer 및 대상 그룹 만들기

이 단원에서는 Amazon EC2 Application Load Balancer를 만듭니다. 로드밸런서를 사용하여 생성하는 대상 그룹 값과 서브넷 이름은 나중에 Amazon ECS 서비스를 생성할 때 사용합니다. Application Load Balancer 또는 Network Load Balancer를 생성할 수 있습니다. 로드밸런서는 다른 가용 영역에 두 개의 퍼블릭 서브넷이 있는 VPC를 사용해야 합니다. 이 단계에서는 기본 VPC를 확인하고 로드밸런서를 생 성한 후 로드밸런서를 위한 대상 그룹을 두 개 생성합니다. 자세한 내용은 <u>네트워크 로드밸런서의 대상</u> 그룹 지정을 참조하십시오.

기본 VPC와 퍼블릭 서브넷을 확인하려면

- 1. 에 로그인 AWS Management Console 하고 <u>https://console.aws.amazon.com/vpc/</u>://https//https//h
- 사용할 기본 VPC를 확인합니다. 탐색 창에서 Your VPCs를 선택합니다. 기본 VPC 열에서 예로 표 시된 VPC를 확인합니다. 이것이 기본 VPC입니다. 이 VPC에는 선택할 기본 서브넷이 포함됩니 다.
- 3. 서브넷을 선택합니다. 기본 서브넷 열에 예로 표시되는 서브넷을 두 개 선택합니다.

Note

서브넷 ID를 기록해 둡니다. 이 자습서 뒷부분에서 이 정보가 필요합니다.

- 4. 서브넷을 선택한 후 설명 탭을 선택합니다. 사용할 서브넷이 다른 가용 영역에 있는지 확인합니다.
- 서브넷을 선택한 후 라우팅 테이블 탭을 선택합니다. 사용할 각 서브넷이 퍼블릭 서브넷인지 확인 하려면, 라우팅 테이블에 게이트웨이 행이 포함되어 있는지 확인합니다.

Amazon EC2 Application Load Balancer를 생성하려면

- 1. 에 로그인 AWS Management Console 하고 <u>https://console.aws.amazon.com/ec2/</u>://https:// https://://https://://https://://https://://https://://https://://
- 2. 탐색 창에서 로드 밸런서를 선택합니다.
- 3. 로드 밸런서 생성을 선택합니다.
- 4. Application Load Balancer를 선택하고 생성을 선택합니다.
- 5. 이름에 로드밸런서의 이름을 입력합니다.
- 6. 체계에서 인터넷 연결을 선택합니다.
- 7. IP 주소 유형에서 ipv4를 선택합니다.
- 8. 로드밸런서를 위한 리스너 두 개를 구성하려면 다음과 같이 합니다.
 - a. 로드밸런서 프로토콜에서 HTTP를 선택합니다. 로드 밸런서 포트에 80을 입력합니다.
 - b. 리스너 추가를 선택합니다.
 - c. 두 번째 리스너의 로드밸런서 프로토콜에서 HTTP를 선택합니다. 로드 밸런서 포트에 8080을 입력합니다.
- 9. 가용 영역의 VPC에서 기본 VPC를 선택합니다. 그런 다음 사용할 기본 서브넷 두 개를 선택합니다.

- 11. 다음: 보안 그룹 구성(Next: Configure Security Groups)을 선택합니다.
- 12. 기존 보안 그룹 선택을 선택하고 보안 그룹 ID를 기록해 둡니다.
- 13. 다음: 라우팅 구성(Next: Configure Routing)을 선택합니다.
- 14. 대상 그룹에서 새 대상 그룹을 선택하고 첫 번째 대상 그룹을 구성합니다.
 - a. 이름에 대상 그룹 이름(예: target-group-1)을 입력합니다.
 - b. 대상 형식에서 IP을 선택합니다.
 - c. 프로토콜에서 HTTP를 선택합니다. 포트에 80을 입력합니다.
 - d. 다음: 대상 등록(Next: Register Targets)을 선택합니다.
- 15. 다음: 검토를 선택한 후 역할 생성을 선택합니다.

로드밸런서에 대한 두 번째 대상 그룹을 생성하려면

- 1. 로드밸런서가 프로비저닝된 후 Amazon EC2 콘솔을 엽니다. 탐색 창에서 대상 그룹을 선택합니다.
- 2. 대상 그룹 생성을 선택합니다.
- 3. 이름에 대상 그룹 이름(예: target-group-2)을 입력합니다.
- 4. 대상 형식에서 IP을 선택합니다.
- 5. 프로토콜에서 HTTP를 선택합니다. 포트에 8080을 입력합니다.
- 6. VPC에서 기본 VPC를 선택합니다.
- 7. 생성(Create)을 선택합니다.

1 Note

배포를 실행하려면 로드밸런서에 대해 두 개의 대상 그룹을 생성해야 합니다. 첫 번째 대상 그룹의 ARN만 기록해 두면 됩니다. 이 ARN은 다음 단계에서 create-service JSON 파일에 사용됩니다.

두 번째 대상 그룹을 포함하도록 로드밸런서를 업데이트하려면

1. Amazon EC2 콘솔을 엽니다. 탐색 창에서 로드 밸런서를 선택합니다.

- 로드 밸런서를 선택한 후 리스너 탭을 선택합니다. 포트 번호가 8080인 리스너를 선택한 후 편 집을 선택합니다.
- 전달 대상 옆의 연필 아이콘을 선택합니다. 두 번째 대상 그룹을 선택한 후 확인 표시를 선택합니다.
 다. 업데이트를 선택하여 업데이트를 저장합니다.

4단계: Amazon ECS 클러스터 및 서비스 생성

이 단원에서는 배포 중 CodeDeploy가 EC2 인스턴스가 아닌 Amazon ECS 클러스터로 트래픽을 라우 팅할 Amazon ECS 클러스터와 서비스를 생성합니다. Amazon ECS 서비스를 생성하려면 로드 밸런서 를 사용하여 생성한 대상 그룹 값과 서브넷 이름 및 보안 그룹을 사용하여 서비스를 생성해야 합니다.

1 Note

이 단계를 사용하여 Amazon ECS 클러스터를 생성할 때 Fargate 컨테이너를 프로비저닝 AWS 하는 네트워킹 전용 클러스터 템플릿을 사용합니다. AWS Fargate는 컨테이너 인스턴스 인프 라를 관리하는 기술입니다. Amazon ECS 클러스터에 대한 Amazon EC2 인스턴스를 선택하거 나 수동으로 생성할 필요가 없습니다.

Amazon ECS 클러스터를 생성하려면

- 1. https://console.aws.amazon.com/ecs/에서 Amazon ECS 클래식 콘솔을 엽니다.
- 2. 탐색 창에서 클러스터를 선택합니다.
- 3. 클러스터 생성을 선택합니다.
- 4. AWS Fargate 기반의 네트워킹 전용 클러스터 템플릿을 선택한 후 다음 단계를 선택합니다.
- 5. Configure cluster(클러스터 구성) 페이지에 클러스터 이름을 입력합니다. 리소스에 대한 선택적 태 그를 추가할 수 있습니다. 생성(Create)을 선택합니다.

Amazon ECS 서비스를 생성하려면

AWS CLI 를 사용하여 Amazon ECS에서 서비스를 생성합니다.

1. JSON 파일을 생성하고 이름을 create-service.json으로 지정합니다. JSON 파일에 다음을 붙여 넣습니다.

taskDefinition 필드의 경우 Amazon ECS에 작업 정의를 등록할 때 패밀리를 정의합니다. 패 밀리는 개정 번호를 사용하여 지정된 작업 정의의 여러 버전에 대한 이름과 비슷합니다. 이 예에 서는 파일에 있는 패밀리 및 개정 번호에 "ecs-demo:1"을 사용합니다. <u>3단계: Application Load</u> Balancer 및 대상 그룹 만들기 단원에서 로드 밸런서를 사용하여 생성한 대상 그룹 값과 서브넷 이름 및 보안 그룹을 사용합니다.

Note

이 파일에 대상 그룹 ARN을 포함시켜야 합니다. Amazon EC2 콘솔을 열고 탐색 창의 로 드 밸런싱에서 대상 그룹을 선택합니다. 첫 번째 대상 그룹을 선택합니다. 설명 탭에서 ARN을 복사합니다.

```
{
    "taskDefinition": "family:revision-number",
    "cluster": "my-cluster",
    "loadBalancers": [
        {
            "targetGroupArn": "target-group-arn",
            "containerName": "sample-website",
            "containerPort": 80
        }
    ],
    "desiredCount": 1,
    "launchTvpe": "FARGATE",
    "schedulingStrategy": "REPLICA",
    "deploymentController": {
        "type": "CODE_DEPLOY"
    },
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": [
                "subnet-1",
                "subnet-2"
            ],
            "securityGroups": [
                "security-group"
            ],
            "assignPublicIp": "ENABLED"
        }
    }
}
```

2. JSON 파일을 지정하여 create-service 명령을 실행합니다.

A Important

파일 이름 앞에 file://를 포함해야 합니다. 이 명령에 필수적입니다.

이 예제에서는 my-service라는 서비스를 생성합니다.

Note

이 예제 명령은 my-service라는 서비스를 생성합니다. 이 이름의 서비스가 이미 있는 경우, 명령은 오류를 반환합니다.

```
aws ecs create-service --service-name my-service --cli-input-json file://create-
service.json
```

출력에 서비스의 설명 필드가 반환됩니다.

3. describe-services 명령을 실행하여 서비스가 생성되었는지 확인합니다.

aws ecs describe-services --cluster *cluster-name* --services *service-name*

5단계: CodeDeploy 애플리케이션 및 배포 그룹 만들기(ECS 컴퓨팅 플랫폼)

Amazon ECS 컴퓨팅 플랫폼에 대한 CodeDeploy 애플리케이션 및 배포 그룹을 만드는 경우 애플리케 이션은 배포 중에 올바른 배포 그룹, 대상 그룹, 리스너 및 트래픽 라우팅 동작을 참조하는 데 사용됩니 다.

CodeDeploy 애플리케이션을 만들려면

- 1. CodeDeploy 콘솔을 열고 애플리케이션 생성을 선택합니다.
- 2. 애플리케이션 이름에 사용하려는 이름을 입력합니다.
- 3. 컴퓨팅 플랫폼에서 Amazon ECS를 선택합니다.
- 4. 애플리케이션 생성을 선택합니다.

CodeDeploy 배포 그룹을 생성하려면

- 1. 애플리케이션 페이지의 배포 그룹 탭에서 배포 그룹 생성을 선택합니다.
- 2. Deployment group name(배포 그룹 이름)에 배포 그룹을 설명하는 이름을 입력합니다.
- 서비스 역할에서 Amazon ECS에 액세스할 수 있는 권한을 CodeDeploy에 부여하는 서비스 역할 을 선택합니다. 새 서비스 역할을 생성하려면 다음 단계를 수행합니다.
 - a. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
 - b. 콘솔 대시보드에서 역할을 선택합니다.
 - c. 역할 생성을 선택합니다.
 - d. 신뢰할 수 있는 유형의 엔터티 선택에서 AWS 서비스를 선택합니다. Choose a use case(사용 사례 선택)에서 CodeDeploy를 선택합니다. Select your use case(사용 사례 선택)에서 CodeDeploy ECS를 선택합니다. 다음: 권한을 선택합니다. AWSCodeDeployRoleForECS 관리형 정책이 역할에 연결됩니다.
 - e. Next: Tags(다음: 태그)를 선택한 후 Next: Review(다음: 검토)를 선택합니다.
 - f. 역할 이름(예: CodeDeployECSRole)을 입력한 후 Create role(역할 생성)을 선택합니다.
- 4. 환경 구성에서 Amazon ECS 클러스터 이름과 서비스 이름을 선택합니다.
- 5. 로드 밸런서에서 Amazon ECS 서비스에 트래픽을 공급하는 로드밸런서의 이름을 선택합니다.
- 프로덕션 리스너 포트에서 해당 Amazon ECS 서비스에 서비스 프로덕션 트래픽을 공급하는 리스 너의 프로토콜과 포트를 선택합니다. Test listener port(테스트 리스너 포트)에서 테스트 리스너를 위한 포트와 프로토콜을 선택합니다.
- 7. 대상 그룹 1 이름과 대상 그룹 2 이름에서 배포 중 트래픽을 라우팅하는 데 사용되는 대상 그룹을 선택합니다. 해당 로드밸런서용으로 생성한 대상 그룹인지 확인합니다.
- 즉시 트래픽 다시 라우팅을 선택하여 배포 성공 후 얼마 후에 업데이트된 Amazon ECS 작업으로 트래픽을 다시 라우팅할지를 지정합니다.
- 9. [Create deployment group]을 선택합니다.

6단계: 파이프라인 생성

이 단원에서는 다음 작업을 통해 파이프라인을 생성합니다.

- 소스 아티팩트가 작업 정의 및 AppSpec 파일인 경우 CodeCommit 작업.
- 소스 아티팩트가 이미지 파일인 경우 Amazon ECR 소스 작업을 포함하는 소스 단계.

• CodeDeploy 애플리케이션 및 배포 그룹을 통한 배포가 실행될 경우 Amazon ECS 배포 작업을 포함 하는 배포 단계.

마법사를 사용하여 2단계 파이프라인을 생성하려면

- 1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https://https://https://https://i/httpsCodePipeline.com.com.com.com.
- 2. [Welcome] 페이지, [Getting started] 페이지 또는 [Pipelines] 페이지에서 Create pipeline(파이프라 인생성)을 선택합니다.
- 3. 1단계: 생성 옵션 선택 페이지의 생성 옵션에서 사용자 지정 파이프라인 빌드 옵션을 선택합니다. Next(다음)를 선택합니다.
- 4. 2단계: 파이프라인 설정 선택의 파이프라인 이름에 MyImagePipeline을 입력합니다.
- CodePipeline은 특성과 가격이 다른 V1 및 V2 유형 파이프라인을 제공합니다. V2 유형은 콘 솔에서 선택할 수 있는 유일한 유형입니다. 자세한 내용은 <u>파이프라인 유형</u>을 참조하세요. CodePipeline에 대한 요금 정보는 <u>요금</u>을 참조하세요.
- 서비스 역할에서 새 서비스 역할을 선택하여CodePipeline이 IAM에 서비스 역할을 생성하도록 허용합니다.
- 7. [Advanced settings]의 설정은 기본값 그대로 두고, [Next]를 선택합니다.
- 3단계: 소스 스테이지 추가의 소스 공급자에서 AWS CodeCommit을 선택합니다. 리포지토리 이 름에서 <u>1단계: CodeCommit 리포지토리 생성</u>에서 생성한 CodeCommit 리포지토리의 이름을 선택 합니다. [Branch name]에서 가장 마지막 코드 업데이트가 포함된 브랜치의 이름을 선택합니다.

Next(다음)를 선택합니다.

- 4단계: 빌드 스테이지 추가에서 빌드 스테이지 건너뛰기를 선택하고 건너뛰기를 다시 선택하여 경고 메시지를 수락합니다. Next(다음)를 선택합니다.
- 5단계: 테스트 단계 추가에서 테스트 단계 건너뛰기를 선택한 다음 건너뛰기를 다시 선택하여 경고 메시지를 수락합니다.

Next(다음)를 선택합니다.

- 11. 6단계: 배포 단계 추가:
 - a. 배포 공급자에서 Amazon ECS(블루/그린)를 선택합니다. 애플리케이션 이름에 애플리케이션 이름을 입력하거나 목록에서 선택합니다(codedeployapp). 배포 그룹에 배포 그룹 이름을 입력하거나 목록에서 선택합니다(예: codedeploydeplgroup).
In Note

"Deploy"라는 이름은 파이프라인의 첫 단계에 "Source"라는 이름이 지정되는 것처럼 4단계: 배포 단계에서 생성한 단계에 기본적으로 지정되는 이름입니다.

- b. Amazon ECS 작업 정의에서 SourceArtifact를 선택합니다. 필드에 **taskdef.json**을 입력합 니다.
- c. AWS CodeDeploy AppSpec 파일에서 SourceArtifact를 선택합니다. 필드에 appspec.yaml을 입력합니다.

Note

이때 작업 정의 이미지를 동적으로 업데이트에는 아무 정보도 입력하지 마십시오.

d. Next(다음)를 선택합니다.

12. 7단계: 검토에서 정보를 검토한 다음 파이프라인 생성을 선택합니다.

파이프라인에 Amazon ECR 소스 작업을 추가하려면

파이프라인을 보고 파이프라인에 Amazon ECR 소스 작업을 추가합니다.

- 1. 파이프라인을 선택합니다. 왼쪽 위에서 편집을 선택합니다.
- 2. 소스 스테이지에서 단계 편집을 선택합니다.
- 3. CodeCommit 소스 작업 옆의 + 작업 추가를 선택하여 병렬 작업을 추가합니다.
- 4. 작업 이름에 이름을 입력합니다(예: Image).
- 5. 작업 공급자에서 Amazon ECR을 선택합니다.

Action name Choose a name for your action	
Image	
No more than 100 characters	
Action provider	
Amazon ECR	▼
Amazon ECR Repository name Choose an Amazon ECR repository as the s	source location.
nginx 🔻	Create repository
nginx Image tag - optional Choose the image tag that triggers your p	Create repository
nginx Image tag - optional Choose the image tag that triggers your p Q If on image tag is not explored of the line to	Create repository
nginx ▼ Image tag - optional Choose the image tag that triggers your p Q If an image tag is not selected, defaults to Output artifacts Choose a name for the output of this action	Create repository Z ipeline when a change occurs in the image repository o latest on.
nginx Image tag - optional Choose the image tag that triggers your p Q If an image tag is not selected, defaults to Output artifacts Choose a name for the output of this action MyImage	Create repository Z ipeline when a change occurs in the image repository b latest on. Remove

- 6. 리포지토리 이름에서 Amazon ECR 리포지토리의 이름을 선택합니다.
- 7. 이미지 이름과 버전이 최신과 다른 경우 이미지 태그에서 지정합니다.
- 8. 출력 아티팩트에서 출력 아티팩트 기본값(예: MyImage)을 선택합니다. 이 아티팩트는 다음 단계 에서 사용하려는 이미지 이름과 리포지토리 URI를 포함합니다.
- 9. 작업 화면에서 저장을 선택합니다. 단계 화면에서 완료를 선택합니다. 파이프라인에서 저장을 선택합니다. Amazon ECR 소스 작업에 대해 생성될 Amazon CloudWatch Events 규칙이 메시지에 표시됩니다.

소스 아티팩트를 배포 작업에 연결하려면

- 1. 배포 단계에서 편집을 선택하고 아이콘을 선택하여 Amazon ECS(블루/그린) 작업을 편집합니다.
- 2. 창 하단으로 스크롤합니다. 입력 아티팩트에서 추가를 선택합니다. 새로운 Amazon ECR 리포지 토리에서 아티팩트 결과물을 추가합니다(예: MyImage).

- 3. 작업 정의에서 SourceArtifact를 선택한 후 taskdef.json이 입력되어 있는지 확인합니다.
- 4. AWS CodeDeploy AppSpec 파일에서 SourceArtifact를 선택한 후 **appspec.yaml**이 입력되어 있 는지 확인합니다.
- 5. Dynamically update task definition image(작업 정의 이미지를 동적으로 업데이트)의 Input Artifact with Image URI(이미지 URI가 있는 입력 아티팩트)에서 MyImage를 선택한 후 taskdef.json 파일에 사용되는 자리 표시자 텍스트인 IMAGE1_NAME을 입력합니다. 저장(Save)을 선택합니다.
- AWS CodePipeline 창에서 파이프라인 변경 사항 저장을 선택한 다음 변경 사항 저장을 선택합니다.

이 예제 파이프라인이 생성되면 콘솔 항목에 대한 작업 구성이 파이프라인 구조에 다음과 같이 표 시됩니다.

"configuration": { "AppSpecTemplateArtifact": "SourceArtifact", "AppSpecTemplatePath": "appspec.yaml", "TaskDefinitionTemplateArtifact": "SourceArtifact", "TaskDefinitionTemplatePath": "taskdef.json", "ApplicationName": "codedeployapp", "DeploymentGroupName": "codedeploydeplgroup", "Image1ArtifactName": "MyImage", "Image1ContainerName": "IMAGE1_NAME" },

- 변경 사항을 제출하고 파이프라인 빌드를 시작하려면 변경 사항 배포를 선택한 다음 릴리스를 선 택하세요.
- 8. CodeDeploy에서 볼 배포 작업을 선택하고 트래픽 이동의 진행 상황을 확인하세요.

Note

선택적인 대기 시간을 보여주는 배포 단계를 볼 수 있습니다. 기본적으로 CodeDeploy는 원래 작업 세트를 종료하기 전에 성공적인 배포 후 1시간을 대기합니다. 이 시간을 사용하 여 작업을 롤백하거나 종료할 수 있지만 작업 세트가 종료되면 배포가 완료됩니다.

7단계: 파이프라인 변경 및 배포 확인

이미지를 변경한 후 변경 사항을 Amazon ECR 리포지토리에 푸시합니다. 이렇게 하면 파이프라인이 실행됩니다. 이미지 소스 변경이 배포되었는지 확인합니다.

자습서: Amazon Alexa Skill을 배포하는 파이프라인 생성

이 자습서에서는 배포 단계에서 배포 공급자로 Alexa Skills Kit를 사용하여 Alexa 스킬을 지속적으로 제공하는 파이프라인을 구성합니다. 완성된 파이프 라인은 소스 리포지토리의 소스 파일을 변경할 때 스킬에 대한 변경 사항을 감지합니다. 파이프라인은 Alexa Skills Kit를 사용하여 Alexa 스킬 개발 단계 에 배포합니다.

▲ Important

파이프라인 생성의 일환으로 CodePipeline은 고객이 제공하는 S3 아티팩트 버킷을 아티팩트 에 사용합니다. (이는 S3 소스 작업에 사용되는 버킷과 다릅니다.) S3 아티팩트 버킷이 파이프 라인의 계정과 다른 계정에 있는 경우 S3 아티팩트 버킷을 안전하고 신뢰할 수 AWS 계정 있는 에서 소유해야 합니다.

1 Note

이 기능은 아시아 태평양(홍콩) 또는 유럽(밀라노) 리전에서 사용할 수 없습니다. 해당 리전에 서 사용 가능한 다른 배포 작업을 사용하려면 배포 작업 통합을 참조하세요.

사용자 지정 스킬을 Lambda 함수로 생성하려면 사용자 <u>지정 스킬을 AWS Lambda 함수로 호스팅을</u> 참조하세요. Lambda 소스 파일과 CodeBuild 프로젝트를 사용하여 스킬 변경 사항을 Lambda에 배포 하는 파이프라인을 생성할 수도 있습니다.

사전 조건

다음 항목이 있어야 합니다.

- CodeCommit 리포지토리. 에서 생성한 AWS CodeCommit 리포지토리를 사용할 수 있습니다<u>자습서:</u> 간단한 파이프라인 생성(CodeCommit 리포지토리).
- Amazon 개발자 계정. 이 계정은 Alexa Skill을 소유합니다. <u>Alexa Skills Kit</u>에서 무료로 계정을 만들 수 있습니다.
- Alexa Skill. Get Custom Skill Sample Code 자습서를 사용하여 샘플 스킬을 만들 수 있습니다.
- ASK CLI를 설치하고 AWS 자격 증명으로 ask init를 사용하여 구성합니다. 자세한 내용은 <u>Install</u> and initialize ASK CLI 단원을 참조하십시오.

자습서: Amazon Alexa Skill을 배포하는 파이프라인 생성

1단계: Alexa 개발자 서비스 LWA 보안 프로필 생성

이 단원에서는 Login With Amazon(LWA)에 사용할 보안 프로필을 만듭니다. 이미 프로필이 있다면 이 단계를 생략할 수 있습니다.

- 보안 프로필을 만들려면 generate-Iwa-tokens의 단계를 사용합니다.
- 프로필을 만든 후 Client ID(클라이언트 ID) 및 Client Secret(클라이언트 암호)을 메모합니다.
- 지침에 제공된 대로 Allowed Return URLs(허용된 반환 URL)를 입력했는지 확인합니다. URL을 통해 ASK CLI 명령은 새로 고침 토큰 요청을 리디렉션할 수 있습니다.

2단계: Alexa Skill 소스 파일 생성 및 CodeCommit 리포지토리에 푸시

이 단원에서는 파이프라인이 소스 단계에 대해 사용하는 리포지토리에 Alexa Skill 소스 파일을 생성하고 푸시합니다. Amazon 개발자 콘솔에서 생성한 스킬의 경우 다음 항목을 생성하고 푸시합니다.

- skill.json 파일.
- interactionModel/custom 폴더.

Note

이 디렉터리 구조는 <u>기술 패키지 형식</u>에 개략적으로 설명된 Alexa Skills Kit 기술 패키지 형 식 요구 사항을 준수합니다. 해당 디렉터리 구조에서 올바른 기술 패키지 형식을 사용하지 않는 경우 변경 사항은 Alexa Skills Kit 콘솔에 성공적으로 배포되지 않습니다.

스킬에 대한 소스 파일을 생성하려면

1. Alexa Skills Kit 개발자 콘솔에서 스킬 ID를 검색합니다. 다음 명령을 사용합니다.

ask api list-skills

스킬별로 이름을 지정한 다음 skillId 필드에 연결된 ID를 복사합니다.

2. 스킬 세부 정보가 포함되어 있는 skill.json 파일을 생성합니다. 다음 명령을 사용합니다.

ask api get-skill -s skill-ID > skill.json

3. (선택 사항) interactionModel/custom 폴더를 만듭니다.

이 명령을 사용하여 폴더 내에 상호 작용 모델 파일을 생성합니다. 언어의 경우 이 자습서에서는 파일 이름의 언어로 en-US를 사용합니다.

ask api get-model --skill-id skill-ID --locale locale >
 ./interactionModel/custom/locale.json

CodeCommit 리포지토리에 파일을 푸시하려면

 CodeCommit 리포지토리에 파일을 푸시하거나 업로드합니다. 이러한 파일은 파이프라인 생성 마 법사가 AWS CodePipeline에서 배포 작업을 위해 생성한 소스 아티팩트입니다. 파일은 로컬 디렉 터리에 다음과 같이 나타납니다.

skill.json
/interactionModel
 /custom
 len-US.json

- 2. 파일을 업로드하는 데 사용할 방법을 선택합니다.
 - a. 로컬 컴퓨터의 복제된 리포지토리에서 Git 명령줄을 사용하려면 다음과 같이 합니다.
 - i. 다음 명령을 실행하여 모든 파일을 한 번에 스테이징합니다.

```
git add -A
```

ii. 다음 명령을 실행하여 커밋 메시지와 함께 파일을 커밋합니다.

git commit -m "Added Alexa skill files"

iii. 다음 명령을 실행하여 로컬 리포지토리의 파일을 CodeCommit 리포지토리에 푸시합니
 다.

git push

- b. CodeCommit 콘솔을 사용하여 파일을 업로드하려면 다음과 같이 합니다.
 - i. CodeCommit 콘솔을 열고 리포지토리 목록에서 해당 리포지토리를 선택합니다.
 - ii. 파일 추가를 선택한 후 파일 업로드를 선택합니다.

- iii. 파일 선택을 선택한 후 파일을 찾습니다. 사용자 이름과 이메일 주소를 입력하여 변경 사 항을 커밋합니다. 변경 사항 커밋을 선택합니다.
- iv. 업로드하려는 각 파일에 대해 이 단계를 반복합니다.

3단계: ASK CLI 명령을 사용하여 새로 고침 토큰 생성

CodePipeline은 Amazon 개발자 계정의 클라이언트 ID와 암호를 기반으로 새로 고침 토큰을 사용하여 사용자를 대신하여 수행하는 작업을 인증합니다. 이 단원에서는 ASK CLI를 사용하여 토큰을 만듭니 다. 파이프라인 생성 마법사를 사용할 때 이 자격 증명을 사용합니다.

Amazon 개발자 계정 자격 증명으로 새로 고침 토큰을 생성하려면

1. 다음 명령을 사용합니다.

ask util generate-lwa-tokens

2. 메시지가 표시되면 다음 예와 같이 클라이언트 ID와 암호를 입력합니다.

```
? Please type in the client ID:
amzn1.application-client.example112233445566
? Please type in the client secret:
example112233445566
```

- 3. 로그인 브라우저 페이지가 표시됩니다. Amazon 개발자 계정 자격 증명으로 로그인합니다.
- 명령줄 화면으로 돌아갑니다. 액세스 토큰 및 새로 고침 토큰이 출력에 생성됩니다. 출력에 반환된 새로 고침 토큰을 복사합니다.

4단계: 파이프라인 생성

이 단원에서는 다음 작업을 통해 파이프라인을 생성합니다.

- 소스 아티팩트가 사용자의 스킬을 지원하는 Alexa Skill 파일인 CodeCommit 작업이 있는 소스 단계.
- Alexa Skill Kit 배포 작업이 적용된 배포 단계.

마법사를 사용하여 파이프라인을 생성하려면

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https//https/

- 2. 프로젝트와 리소스를 생성할 AWS 리전을 선택합니다. Alexa skill 실행 시간은 다음 리전에서만 사용 가능합니다.
 - 아시아 태평양(도쿄)
 - 유럽(아일랜드)
 - 미국 동부(버지니아 북부)
 - 미국 서부(오레곤)
- [Welcome] 페이지, [Getting started] 페이지 또는 [Pipelines] 페이지에서 Create pipeline(파이프라 인 생성)을 선택합니다.
- 4. 1단계: 생성 옵션 선택 페이지의 생성 옵션에서 사용자 지정 파이프라인 빌드 옵션을 선택합니다. Next(다음)를 선택합니다.
- 5. 2단계: 파이프라인 설정 선택의 파이프라인 이름에 MyAlexaPipeline을 입력합니다.
- CodePipeline은 특성과 가격이 다른 V1 및 V2 유형 파이프라인을 제공합니다. V2 유형은 콘 솔에서 선택할 수 있는 유일한 유형입니다. 자세한 내용은 <u>파이프라인 유형</u>을 참조하세요. CodePipeline에 대한 요금 정보는 <u>요금</u>을 참조하세요.
- 7. 서비스 역할에서 새 서비스 역할을 선택하여CodePipeline이 IAM에 서비스 역할을 생성하도록 허 용합니다.
- 8. [Advanced settings]의 설정은 기본값 그대로 두고, [Next]를 선택합니다.
- 3단계: 소스 스테이지 추가의 소스 공급자에서 AWS CodeCommit을 선택합니다. 리포지토리 이 름에서 <u>1단계: CodeCommit 리포지토리 생성</u>에서 생성한 CodeCommit 리포지토리의 이름을 선택 합니다. [Branch name]에서 가장 마지막 코드 업데이트가 포함된 브랜치의 이름을 선택합니다.

리포지토리 이름과 브랜치를 선택하면 이 파이프라인에 대해 생성될 Amazon CloudWatch Events 규칙을 보여 주는 메시지가 표시됩니다.

Next(다음)를 선택합니다.

4단계: 빌드 스테이지 추가에서 빌드 스테이지 건너뛰기를 선택하고 건너뛰기를 다시 선택하여 경고 메시지를 수락합니다.

Next(다음)를 선택합니다.

11. 5단계: 테스트 단계 추가에서 테스트 단계 건너뛰기를 선택한 다음 건너뛰기를 다시 선택하여 경 고 메시지를 수락합니다.

Next(다음)를 선택합니다.

12. 6단계: 배포 단계 추가:

- a. 배포 공급자에서 Alexa Skills Kit를 선택합니다.
- b. Alexa skill ID에 Alexa Skills Kit 개발자 콘솔의 스킬에 할당된 스킬 ID를 입력합니다.
- c. Client ID(클라이언트 ID)에 등록한 애플리케이션의 ID를 입력합니다.
- d. Client secret(클라이언트 암호)에 등록할 때 선택한 암호를 입력합니다.
- e. Refresh token(새로 고침 토큰)에 3단계에서 생성한 토큰을 입력합니다.

You cannot skip this stage Pipelines must have at least two stages. Your second stage must be either a build or deployment stage. Choose a provider for either the build stage or deployment stage.			
Deploy	,		
Deploy p	rovider by you deploy to instances. Choose the provider, and then provide the configuration details for that provider.		
Alexa S	jkills Kit 💌		
Alexa ski The uniqu amzn1 Client ID The client page.	ILI D e identifier of the skill. .ask.skill.da55cd70-9eaf-4cf1-b326-: ID of the application registered for LWA (Login With Amazon). Look for this on the Alexa Skills Kit developer portal LWA		
amzn1	.application-oa2-client.e		
Client se The client .WA page	cret secret of the application registered for LWA (Login With Amazon). Look for this on the Alexa Skills Kit developer portal		
lefresh t he refres	Loken Loken used to request new access tokens.		

f. Next(다음)를 선택합니다.

13. 7단계: 검토에서 정보를 검토한 다음 파이프라인 생성을 선택합니다.

5단계: 모든 소스 파일을 변경하고 배포 확인

스킬을 변경한 후 변경 사항을 리포지토리에 푸시합니다. 이렇게 하면 파이프라인이 실행됩니다. Alexa Skills Kit 개발자 콘솔에서 스킬이 업데이트되었는지 확인합니다.

자습서: Amazon S3를 배포 공급자로 사용하는 파이프라인 생성

이 자습서에서는 배포 단계에서 배포 작업 공급자로 Amazon S3를 사용하여 파일을 지속적으로 제공 하는 파이프라인을 구성합니다. 완성된 파이프 라인은 소스 리포지토리의 소스 파일을 변경할 때 변경 사항을 감지합니다. 그런 다음 파이프라인은 Amazon S3를 사용하여 파일을 버킷에 배포합니다. 소스 위치에서 웹 사이트 파일을 수정 또는 추가할 때마다 배포 작업에서 최신 파일을 사용하여 웹 사이트가 작성됩니다.

A Important

파이프라인 생성의 일환으로 CodePipeline은 고객이 제공하는 S3 아티팩트 버킷을 아티팩트 에 사용합니다. (이는 S3 소스 작업에 사용되는 버킷과 다릅니다.) S3 아티팩트 버킷이 파이프 라인의 계정과 다른 계정에 있는 경우 S3 아티팩트 버킷을 안전하고 신뢰할 수 AWS 계정 있는 가 소유해야 합니다.

Note

소스 리포지토리에서 파일을 삭제하더라도 S3 배포 작업은 삭제된 파일에 해당하는 S3 객체를 삭제하지 않습니다.

이 자습서에서는 두 가지 옵션을 제공합니다.

- 정적 웹 사이트를 S3 퍼블릭 버킷에 배포하는 파이프라인을 생성합니다. 이 예제에서는 AWS CodeCommit 소스 작업과 Amazon S3 배포 작업이 있는 파이프라인을 생성합니다. <u>옵션 1: Amazon</u> S3에 정적 웹 사이트 배포을 참조하세요.
- 샘플 TypeScript 코드를 JavaScript로 컴파일하는 파이프라인을 생성하고 CodeBuild 출력 아티팩트 를 아카이브용 S3 버킷에 배포합니다. 이 예제에서는 Amazon S3 소스 작업, CodeBuild 빌드 작업 및 Amazon S3 배포 작업을 통해 파이프라인을 생성합니다. <u>옵션 2: S3 소스 버킷에서 Amazon S3에</u> 빌드된 아카이브 파일 배포을 참조하세요.

A Important

이 절차에서 파이프라인에 추가하는 많은 작업에는 파이프라인을 생성하기 전에 생성해야 하 는 AWS 리소스가 포함됩니다. 소스 작업에 대한 AWS 리소스는 항상 파이프라인을 생성하는 리전과 동일한 AWS 리전에서 생성해야 합니다. 예를 들어, 미국 동부(오하이오) 리전에서 파 이프라인을 생성하는 경우, CodeCommit 리포지토리도 미국 동부(오하이오) 리전에 있어야 합니다.

파이프라인을 생성할 때 교차 리전 작업을 추가할 수 있습니다. 교차 리전 작업에 대한 AWS 리소스는 작업을 실행하려는 리전과 동일한 AWS 리전에 있어야 합니다. 자세한 내용은 CodePipeline에 교차 리전 작업 추가 단원을 참조하십시오.

옵션 1: Amazon S3에 정적 웹 사이트 배포

이 예제에서는 샘플 정적 웹 사이트 템플릿 파일을 다운로드하고, AWS CodeCommit 리포지토리에 파 일을 업로드하고, 버킷을 생성하고, 호스팅을 위해 구성합니다. 다음으로 AWS CodePipeline 콘솔을 사용하여 파이프라인을 생성하고 Amazon S3 배포 구성을 지정합니다.

사전 조건

다음 항목이 있어야 합니다.

- CodeCommit 리포지토리. 에서 생성한 AWS CodeCommit 리포지토리를 사용할 수 있습니다<u>자습서:</u> 간단한 파이프라인 생성(CodeCommit 리포지토리).
- 정적 웹 사이트의 소스 파일. 이 링크를 사용하여 <u>샘플 정적 웹 사이트</u>를 다운로드합니다. samplewebsite.zip 다운로드는 다음 파일을 생성합니다.
 - index.html 파일
 - main.css 파일
 - graphic.jpg 파일
- 웹 사이트 호스팅용으로 구성된 S3 버킷. <u>Amazon S3에서 정적 웹 사이트 호스팅</u>을 참조하십시오. 버킷을 파이프라인과 같은 리전에 생성해야 합니다.

Note

웹 사이트를 호스팅하려면 버킷에 퍼블릭 읽기 액세스 권한이 있어야 합니다. 이를 통해 모 든 사람에게 읽기 권한이 부여됩니다. 웹 사이트 호스팅을 제외하고 S3 버킷에 대한 퍼블릭 액세스를 차단하는 기본 액세스 설정을 유지해야 합니다.

1단계: 소스 파일을 CodeCommit 리포지토리에 푸시

이 단원에서는 파이프라인이 소스 단계에 대해 사용하는 리포지토리에 소스 파일을 푸시합니다.

CodeCommit 리포지토리에 파일을 푸시하려면

- 1. 다운로드한 샘플 파일을 추출합니다. ZIP 파일을 리포지토리에 업로드하지 마십시오.
- CodeCommit 리포지토리에 파일을 푸시하거나 업로드합니다. 이러한 파일은 파이프라인 생성 마 법사가 CodePipeline에서 배포 작업을 위해 생성한 소스 아티팩트입니다. 파일은 로컬 디렉터리에 다음과 같이 나타납니다.

```
index.html
main.css
graphic.jpg
```

- 3. Git 또는 CodeCommit 콘솔을 사용하여 파일을 업로드할 수 있습니다.
 - a. 로컬 컴퓨터의 복제된 리포지토리에서 Git 명령줄을 사용하려면 다음과 같이 합니다.
 - i. 다음 명령을 실행하여 모든 파일을 한 번에 스테이징합니다.

```
git add -A
```

ii. 다음 명령을 실행하여 커밋 메시지와 함께 파일을 커밋합니다.

```
git commit -m "Added static website files"
```

iii. 다음 명령을 실행하여 로컬 리포지토리의 파일을 CodeCommit 리포지토리에 푸시합니다.

git push

- b. CodeCommit 콘솔을 사용하여 파일을 업로드하려면 다음과 같이 합니다.
 - i. CodeCommit 콘솔을 열고 리포지토리 목록에서 해당 리포지토리를 선택합니다.
 - ii. 파일 추가를 선택한 후 파일 업로드를 선택합니다.
 - iii. 파일 선택을 선택한 다음 파일을 찾습니다. 사용자 이름과 이메일 주소를 입력하여 변경
 사항을 커밋합니다. 변경 사항 커밋을 선택합니다.
 - iv. 업로드하려는 각 파일에 대해 이 단계를 반복합니다.

2단계: 파이프라인 생성

이 단원에서는 다음 작업을 통해 파이프라인을 생성합니다.

- 소스 아티팩트가 웹 사이트용 파일인 CodeCommit 작업이 있는 소스 단계.
- Amazon S3 배포 작업이 적용된 배포 단계.

마법사를 사용하여 파이프라인을 생성하려면

- 1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https://https://https://https://i/httpsCodePipeline.com.com.com.com.
- 2. Welcome(시작) 페이지, 시작하기 페이지 또는 Pipelines(파이프라인) 페이지에서 파이프라인 생성을 선택합니다.
- 1단계: 생성 옵션 선택 페이지의 생성 옵션에서 사용자 지정 파이프라인 빌드 옵션을 선택합니다. Next(다음)를 선택합니다.
- 4. 2단계: 파이프라인 설정 선택의 파이프라인 이름에 MyS3DeployPipeline을 입력합니다.
- 5. 파이프라인 유형에서 V2를 선택합니다. 자세한 내용은 <u>파이프라인 유형</u> 단원을 참조하십시오. Next(다음)를 선택합니다.
- 서비스 역할에서 새 서비스 역할을 선택하여CodePipeline이 IAM에 서비스 역할을 생성하도록 허 용합니다.
- 7. [Advanced settings]의 설정은 기본값 그대로 두고, [Next]를 선택합니다.
- 3단계: 소스 스테이지 추가의 소스 공급자에서 AWS CodeCommit을 선택합니다. 리포지토리 이 름에서 <u>1단계: CodeCommit 리포지토리 생성</u>에서 생성한 CodeCommit 리포지토리의 이름을 선택 합니다. [Branch name]에서 가장 마지막 코드 업데이트가 포함된 브랜치의 이름을 선택합니다. 사 용자가 자체적으로 다른 브랜치를 생성하지 않았다면 main만 선택할 수 있습니다.

리포지토리 이름과 브랜치를 선택하면 이 파이프라인에 대해 생성될 Amazon CloudWatch Events 규칙이 표시됩니다.

Next(다음)를 선택합니다.

4단계: 빌드 스테이지 추가에서 빌드 스테이지 건너뛰기를 선택하고 건너뛰기를 다시 선택하여 경고 메시지를 수락합니다.

Next(다음)를 선택합니다.

10. 5단계: 테스트 단계 추가에서 테스트 단계 건너뛰기를 선택한 다음 건너뛰기를 다시 선택하여 경 고 메시지를 수락합니다.

Next(다음)를 선택합니다.

11. 6단계: 배포 단계 추가:

- a. Deploy provider(배포 공급자)에서 Amazon S3를 선택합니다.
- b. 버킷에 퍼블릭 버킷 이름을 입력합니다.
- c. Extract file before deploy(배포 전 파일 추출)를 선택합니다.

Note

Extract file before deploy(배포 전 파일 추출)를 선택하지 않으면 배포가 실패합니다. 이는 파이프라인의 AWS CodeCommit 작업이 소스 아티팩트를 압축하고 파일이 ZIP 파일이기 때문입니다.

Extract file before deploy(배포 전 파일 추출)를 선택하면 Deployment path(배포 경로)가 표시 됩니다. 사용할 경로의 이름을 입력합니다. 그러면 파일이 추출되는 Amazon S3에 폴더 구조 가 생성됩니다. 이 자습서에서는 이 필드를 비워 둡니다.

Deploy	- optional
Deploy pro	ovider
Amazon	S3
Amazon	53
Specify you	r Amazon S3 location.
Bucket	
my-code	pipeline-website-bucket
Deployme	nt path - optional
Extrac The dep	t file before deploy ployed artifact will be unzipped before deployment.
▶ Additi	onal configuration
	Cancel Previous Skip deploy stage

- d. (선택 사항) 표준 ACL에서 <u>표준 ACL</u>이라고 하는 미리 정의된 일련의 권한 부여를 업로드된 아티팩트에 적용할 수 있습니다.
- e. (선택 사항) 캐시 제어에 캐싱 파라미터를 입력합니다. 요청/응답에 대한 캐싱 동작을 제어하 도록 이를 설정할 수 있습니다. 유효한 값의 경우 HTTP 작업에 대한 <u>Cache-Control</u> 헤더 필드를 확인합니다.

- f. Next(다음)를 선택합니다.
- 12. 7단계: 검토에서 정보를 검토한 다음 파이프라인 생성을 선택합니다.
- 13. 파이프라인이 성공적으로 실행된 후 Amazon S3 콘솔을 열고 파일이 다음과 같이 퍼블릭 버킷에 나타나는지 확인합니다.

index.html
main.css
graphic.jpg

14. 엔드포인트에 액세스하여 웹 사이트를 테스트합니다. 엔드포인트의 형식은 다음과 같습니다. http://bucket-name.s3-website-region.amazonaws.com/

엔드포인트 예제: http://my-bucket.s3-website-us-west-2.amazonaws.com/.

샘플 웹페이지가 표시됩니다.

3단계: 모든 소스 파일을 변경하고 배포 확인

소스 파일을 변경한 후 변경 사항을 리포지토리에 푸시합니다. 이렇게 하면 파이프라인이 실행됩니다. 웹 사이트가 업데이트되었는지 확인합니다.

옵션 2: S3 소스 버킷에서 Amazon S3에 빌드된 아카이브 파일 배포

이 옵션에서 빌드 단계의 빌드 명령은 TypeScript 코드를 JavaScript 코드로 컴파일하고 별도의 타임스 탬프 폴더 아래에 있는 S3 대상 버킷에 출력을 배포합니다. 먼저 TypeScript 코드와 buildspec.yml 파 일을 생성합니다. ZIP 파일의 소스 파일을 결합한 후에 소스 ZIP 파일을 S3 소스 버킷에 업로드하고, CodeBuild 단계를 사용하여 빌드된 애플리케이션 ZIP 파일을 S3 대상 버킷에 배포합니다. 컴파일된 코 드는 대상 버킷에 아카이브로 보관됩니다.

사전 조건

다음 항목이 있어야 합니다.

- S3 소스 버킷. 자습서: 간단한 파이프라인 생성(S3 버킷)에서 생성한 버킷을 사용할 수 있습니다.
- S3 대상 버킷. <u>Amazon S3에서 정적 웹 사이트 호스팅</u>을 참조하십시오. 생성하려는 파이프라인 AWS 리전 과 동일한에서 버킷을 생성해야 합니다.

Note

이 예제는 파일을 프라이빗 버킷에 배포하는 방법을 보여 줍니다. 웹 사이트 호스팅용 대상 버킷을 활성화하거나 버킷을 공개하는 정책을 연결하지 마십시오.

1단계: 소스 파일을 생성하고 S3 소스 버킷에 업로드

이 단원에서는 파이프라인이 소스 단계에 대해 사용하는 버킷에 소스 파일을 생성하고 업로드합니다. 이 단원에서는 다음 소스 파일을 생성하는 방법에 대해 설명합니다.

- CodeBuild 빌드 프로젝트에 사용되는 buildspec.yml 파일.
- index.ts 파일.

buildspec.yml 파일을 만들려면

• 다음 콘텐츠를 가진 buildspec.yml이라는 파일을 생성합니다: 이러한 빌드 명령은 TypeScript 를 설치하고 TypeScript 컴파일러를 사용하여 index.ts의 코드를 JavaScript 코드로 다시 작성 합니다.

```
version: 0.2
phases:
    install:
        commands:
            - npm install -g typescript
    build:
        commands:
            - tsc index.ts
artifacts:
    files:
        - index.js
```

index.ts 파일을 만들려면

• 다음 콘텐츠를 가진 index.ts이라는 파일을 생성합니다:

interface Greeting {

```
message: string;
}
class HelloGreeting implements Greeting {
    message = "Hello!";
}
function greet(greeting: Greeting) {
    console.log(greeting.message);
}
let greeting = new HelloGreeting();
greet(greeting);
```

S3 소스 버킷에 파일을 업로드하려면

1. 파일은 로컬 디렉터리에 다음과 같이 나타납니다.

```
buildspec.yml
index.ts
```

파일을 압축하고 해당 파일의 이름을 source.zip으로 지정합니다.

- Amazon S3 콘솔에서 소스 버킷에 대해 업로드를 선택합니다. 파일 추가를 선택한 다음 생성한 ZIP 파일을 찾습니다.
- 업로드를 선택합니다. 이러한 파일은 파이프라인 생성 마법사가 CodePipeline에서 배포 작업을 위 해 생성한 소스 아티팩트입니다. 파일은 버킷에 다음과 같이 나타납니다.

```
source.zip
```

2단계: 파이프라인 생성

이 단원에서는 다음 작업을 통해 파이프라인을 생성합니다.

- 소스 아티팩트가 다운로드 가능한 애플리케이션용 파일인 Amazon S3 작업이 있는 소스 단계.
- Amazon S3 배포 작업이 적용된 배포 단계.

마법사를 사용하여 파이프라인을 생성하려면

- 1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https//https
- 2. Welcome(시작) 페이지, 시작하기 페이지 또는 Pipelines(파이프라인) 페이지에서 파이프라인 생성을 선택합니다.
- 3. 1단계: 생성 옵션 선택 페이지의 생성 옵션에서 사용자 지정 파이프라인 빌드 옵션을 선택합니다. Next(다음)를 선택합니다.
- 4. 2단계: 파이프라인 설정 선택의 파이프라인 이름에 MyS3DeployPipeline을 입력합니다.
- 5. 서비스 역할에서 새 서비스 역할을 선택하여CodePipeline이 IAM에 서비스 역할을 생성하도록 허 용합니다.
- 6. [Advanced settings]의 설정은 기본값 그대로 두고, [Next]를 선택합니다.
- 7. 3단계: 소스 스테이지 추가의 소스 공급자에서 Amazon S3를 선택합니다. Bucket(버킷)에서 소스 버킷의 이름을 선택합니다. S3 object key(S3 객체 키)에 소스 ZIP 파일의 이름을 입력합니다. .zip 파일 확장명을 포함해야 합니다.

Next(다음)를 선택합니다.

- 8. 4단계: 빌드 스테이지 추가에서 다음과 같이 합니다.
 - a. 빌드 공급자에서 CodeBuild를 선택합니다.
 - b. 빌드 프로젝트 생성을 선택합니다. Create project(프로젝트 생성) 페이지:
 - c. 프로젝트 이름에 이 빌드 프로젝트의 이름을 입력합니다.
 - d. 환경에서 Managed image(관리형 이미지)를 선택합니다. [Operating system]에서 [Ubuntu]를 선택합니다.
 - e. 실행 시간에서 표준을 선택합니다. 실행 시간 버전에서 aws/codebuild/standard:1.0을 선택합니다.
 - f. Image version(이미지 버전)에서 Always use the latest image for this runtime version(이 실행 시간 버전에 항상 최신 이미지 사용)을 선택합니다.
 - g. 서비스 역할에 CodeBuild 서비스 역할을 선택하거나 하나 생성합니다.
 - h. Build specifications(빌드 사양)에서 Use a buildspec file(빌드 사양 파일 사용)을 선택합니다.
 - i. Continue to CodePipeline(CodePipeline으로 계속)을 선택합니다. 프로젝트가 성공적으로 생성되면 메시지가 표시됩니다.
 - j. Next(다음)를 선택합니다.
- 9. 5단계: 배포 스테이지 추가에서 다음과 같이 합니다.

- a. Deploy provider(배포 공급자)에서 Amazon S3를 선택합니다.
- b. Bucket(버킷)에 S3 대상 버킷 이름을 입력합니다.
- c. Extract file before deploy(배포 전 파일 추출)를 선택 해제해야 합니다.

Extract file before deploy(배포 전 파일 추출)를 선택 해제하면 S3 object key(S3 객체 키)가 표시됩니다. 사용할 경로의 이름 js-application/{datetime}.zip을 입력합니다.

그러면 파일이 추출되는 Amazon S3에 js-application 폴더가 생성됩니다. 이 폴더에서 파이프라인이 실행될 때 {datetime} 변수는 각 출력 파일에 타임스탬프를 생성합니다.

Add deploy stage	3
Deploy - optional	A we
Deploy provider Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.	
Amazon S3	
Amazon S3	
Specify your Amazon S3 location.	
Bucket	
my-codepipeline-website-bucket	3
S3 object key	
js-application/{datetime}.zip	-
Extract file before deploy The deployed artifact will be unzipped before deployment.	
Additional configuration	

- d. (선택 사항) 표준 ACL에서 <u>표준 ACL</u>이라고 하는 미리 정의된 일련의 권한 부여를 업로드된 아티팩트에 적용할 수 있습니다.
- e. (선택 사항) 캐시 제어에 캐싱 파라미터를 입력합니다. 요청/응답에 대한 캐싱 동작을 제어하 도록 이를 설정할 수 있습니다. 유효한 값의 경우 HTTP 작업에 대한 <u>Cache-Control</u> 헤더 필드를 확인합니다.
- f. Next(다음)를 선택합니다.
- 10. 6단계: 검토에서 정보를 검토한 다음, 파이프라인 생성을 선택합니다.

11. 파이프라인이 성공적으로 실행되면 Amazon S3 콘솔에서 버킷을 봅니다. 배포된 ZIP 파일이 jsapplication 폴더 아래의 대상 버킷에 표시되는지 확인합니다. ZIP 파일에 포함된 JavaScript 파일은 index.js여야 합니다. index.js 파일에는 다음 출력이 포함되어 있습니다.

```
var HelloGreeting = /** @class */ (function () {
    function HelloGreeting() {
        this.message = "Hello!";
    }
    return HelloGreeting;
}());
function greet(greeting) {
    console.log(greeting.message);
}
var greeting = new HelloGreeting();
greet(greeting);
```

3단계: 모든 소스 파일을 변경하고 배포 확인

소스 파일을 변경한 후 소스 버킷에 업로드합니다. 이렇게 하면 파이프라인이 실행됩니다. 대상 버킷을 보고 다음과 같이 배포된 출력 파일을 js-application 폴더에서 사용할 수 있는지 확인합니다.

Amazon S3 > my-codepipeline-website-bucket > js-applica	ition
Overview	-
	<
Q Type a prefix and press Enter to search. Press ESC to clear.	
▲ Upload	}
	5
□ Name ▼	Last mod
2019-01-10_07-28-07.zip	Jan 9, 25, GMT-08
2019-01-10_07-39-57.zip	Jan 9, 20

자습서: 서버리스 애플리케이션을에 게시하는 파이프라인 생성 AWS Serverless Application Repository

AWS CodePipeline 를 사용하여 AWS SAM 서버리스 애플리케이션을에 지속적으로 전달할 수 있습니 다 AWS Serverless Application Repository.

▲ Important

파이프라인 생성의 일환으로 CodePipeline은 고객이 제공하는 S3 아티팩트 버킷을 아티팩트 에 사용합니다. (이는 S3 소스 작업에 사용되는 버킷과 다릅니다.) S3 아티팩트 버킷이 파이프 라인의 계정과 다른 계정에 있는 경우 S3 아티팩트 버킷을 안전하고 신뢰할 수 AWS 계정 있는 에서 소유해야 합니다.

이 자습서에서는 GitHub에서 호스팅되는 서버리스 애플리케이션을 빌드하고에 AWS Serverless Application Repository 자동으로 게시하도록 파이프라인을 생성하고 구성하는 방법을 보여줍니다. 파이프라인은 GitHub를 소스 공급자로 사용하고 CodeBuild를 빌드 공급자로 사용합니다. 서버리스 애플리케이션을에 게시하려면 <u>애플리케이션</u>(에서 AWS Serverless Application Repository)을 AWS Serverless Application Repository배포하고 해당 애플리케이션에서 생성한 Lambda 함수를 파이 프라인의 호출 작업 공급자로 연결합니다. 그런 다음 코드를 작성하지 AWS Serverless Application Repository않고도에 애플리케이션 업데이트를 지속적으로 제공할 수 있습니다.

\Lambda Important

이 절차에서 파이프라인에 추가하는 많은 작업에는 파이프라인을 생성하기 전에 생성해야 하 는 AWS 리소스가 포함됩니다. 소스 작업에 대한 AWS 리소스는 항상 파이프라인을 생성하는 리전과 동일한 AWS 리전에서 생성해야 합니다. 예를 들어, 미국 동부(오하이오) 리전에서 파 이프라인을 생성하는 경우, CodeCommit 리포지토리도 미국 동부(오하이오) 리전에 있어야 합 니다.

파이프라인을 생성할 때 교차 리전 작업을 추가할 수 있습니다. 교차 리전 작업에 대한 AWS 리소스는 작업을 실행하려는 리전과 동일한 AWS 리전에 있어야 합니다. 자세한 내용은 CodePipeline에 교차 리전 작업 추가 단원을 참조하십시오.

시작하기 전 준비 사항

이 자습서에서는 다음을 가정합니다.

- 사용자는 <u>AWS Serverless Application Model (AWS SAM)</u> 및 <u>AWS Serverless Application</u> Repository에 익숙합니다.
- AWS SAM CLI를 AWS Serverless Application Repository 사용하여에 게시한 서버리스 애플리 케이션이 GitHub에 호스팅되어 있습니다. 예제 애플리케이션을에 게시하려면 AWS Serverless Application Repository 개발자 안내서의 <u>빠른 시작: 애플리케이션 게시</u>를 AWS Serverless Application Repository참조하세요. 자체 애플리케이션을에 게시하려면 AWS Serverless Application Model 개발자 안내서<u>의 AWS SAM CLI를 사용하여 애플리케이션 게시</u>를 AWS Serverless Application Repository참조하세요.

1단계: buildspec.yml 파일 생성

다음 내용으로 buildspec.yml 파일을 생성하고 서버리스 애플리케이션의 GitHub 리포지토리에 추 가합니다. template.yml을 애플리케이션의 AWS SAM 템플릿으로 바꾸고 bucketname을 패키지 애플리케이션이 저장된 S3 버킷으로 바꿉니다.

```
version: 0.2
phases:
    install:
    runtime-versions:
        python: 3.8
    build:
        commands:
            - sam package --template-file template.yml --s3-bucket bucketname --output-
template-file packaged-template.yml
artifacts:
    files:
        - packaged-template.yml
```

2단계: 파이프라인 생성 및 구성

다음 단계에 따라 서버리스 애플리케이션을 게시 AWS 리전 하려는에서 파이프라인을 생성합니다.

- 1. 에 로그인 AWS Management Console 하고 <u>https://console.aws.amazon.com/codepipeline/</u>:// https://https://https://i/httpsCodePipeline.com.com.com.com.
- 2. 필요한 경우 서버리스 애플리케이션을 게시 AWS 리전 할 로 전환합니다.
- [Welcome] 페이지, [Getting started] 페이지 또는 [Pipelines] 페이지에서 Create pipeline(파이프라 인 생성)을 선택합니다.

- 1단계: 생성 옵션 선택 페이지의 생성 옵션에서 사용자 지정 파이프라인 빌드 옵션을 선택합니다. Next(다음)를 선택합니다.
- 5. [파이프라인 생성]을 선택합니다. 2단계: 파이프라인 설정 선택 페이지의 파이프라인 이름에 파이 프라인 이름을 입력합니다.
- 파이프라인 유형에서 V2를 선택합니다. 자세한 내용은 <u>파이프라인 유형</u> 단원을 참조하십시오. Next(다음)를 선택합니다.
- 서비스 역할에서 새 서비스 역할을 선택하여CodePipeline이 IAM에 서비스 역할을 생성하도록 허용합니다.
- 8. [Advanced settings]의 설정은 기본값 그대로 두고, [Next]를 선택합니다.
- 9. 3단계: 소스 스테이지 추가 페이지의 소스 공급자에서 GitHub를 선택합니다.
- 10. 연결에서 기존 연결을 선택하거나 새로 생성합니다. GitHub 소스 작업에 대한 연결을 만들거나 관 리하려면 GitHub 연결을 참조하세요.
- 11. 리포지토리에서 GitHub 소스 리포지토리를 선택합니다.
- 12. 브랜치에서 GitHub 브랜치를 선택합니다.
- 13. 소스 작업의 나머지 기본값은 그대로 둡니다. Next(다음)를 선택합니다.
- 14. 4단계: 빌드 단계 추가 페이지에서 빌드 단계를 추가합니다.
 - a. 빌드 공급자에서 AWS CodeBuild를 선택합니다. 리전에서 파이프라인 리전을 사용합니다.
 - b. 프로젝트 생성을 선택합니다.
 - c. 프로젝트 이름에 이 빌드 프로젝트의 이름을 입력합니다.
 - d. 환경 이미지에서 이미지 관리를 선택합니다. [Operating system]에서 [Ubuntu]를 선택합니다.
 - e. 실행 시간 및 실행 시간 버전에서 서버리스 애플리케이션에 필요한 런타임 및 버전을 선택합 니다.
 - f. 서비스 역할에서 New service role(새 서비스 역할)을 선택합니다.
 - g. Build specifications(빌드 사양)에서 Use a buildspec file(빌드 사양 파일 사용)을 선택합니다.
 - h. Continue to CodePipeline(CodePipeline으로 계속)을 선택합니다. 이렇게 하면 CodePipeline 콘솔이 열리고 구성을 위해 리포지토리에서 buildspec.yml을 사용하는 CodeBuild 프로젝 트를 생성합니다. 빌드 프로젝트가 서비스 역할을 사용하여 AWS 서비스 권한을 관리합니다. 이 단계는 몇 분이 걸릴 수 있습니다.
 - i. Next(다음)를 선택합니다.
- 15. 5단계: 테스트 단계 추가에서 테스트 단계 건너뛰기를 선택한 다음 건너뛰기를 다시 선택하여 경 고 메시지를 수락합니다.

Next(다음)를 선택합니다.

- 16. 6단계: 배포 단계 추가 페이지에서 배포 단계 건너뛰기를 선택한 다음 다시 건너뛰기를 선택하여 경고 메시지를 수락합니다. Next(다음)를 선택합니다.
- 17. 7단계: 검토에서 파이프라인 생성을 선택합니다. 단계를 보여주는 다이어그램이 표시되어야 합니다.
 다.
- 18. 패키지 애플리케이션이 저장된 S3 버킷에 액세스하려면 CodeBuild 서비스 역할 권한을 부여합니다.
 - a. 새 파이프라인의 빌드 단계에서 CodeBuild를 선택합니다.
 - b. Build details(빌드 세부 정보) 탭을 선택합니다.
 - c. 환경에서 CodeBuild 서비스 역할을 선택하여 IAM 콘솔을 엽니다.
 - d. CodeBuildBasePolicy에 대한 선택을 확장하고 Edit policy(정책 편집)을 선택합니다.
 - e. JSON을 선택합니다.
 - f. 다음 내용으로 새 정책문을 추가합니다. 이 정책문을 사용하여 CodeBuild가 패키징된 애플리 케이션이 저장된 S3 버킷에 객체를 넣을 수 있습니다. bucketname을 S3 버킷 이름으로 바 꿉니다.

{ "Effect": "Allow", "Resource": ["arn:aws:s3:::bucketname/*"], "Action": ["s3:PutObject"] }

- g. 정책 검토를 선택합니다.
- h. Save changes(변경 사항 저장)를 선택합니다.

3단계: 게시 애플리케이션 배포

AWS Serverless Application Repository에 게시를 수행하는 Lambda 함수가 포함된 애플리케이션을 배포하려면 다음 단계를 수행하세요. 이 애플리케이션은 aws-serverless-codepipeline-serverlessrepopublish입니다.

Note

파이프라인 AWS 리전 과 동일한에 애플리케이션을 배포해야 합니다.

- 1. 애플리케이션 페이지로 이동하고 배포를 선택합니다.
- I acknowledge that this app creates custom IAM roles(이 앱에서 사용자 지정 IAM 역할을 생성하는 것을 확인합니다)를 선택합니다.
- 3. 배포(Deploy)를 선택합니다.
- 4. AWS CloudFormation 스택 보기를 선택하여 AWS CloudFormation 콘솔을 엽니다.
- 5. 리소스 섹션을 확장합니다. AWS::Lambda::Function 유형의 ServerlessRepoPublish가 표시됩니다. 다음 단계에서 이 리소스의 물리적 ID를 적어 둡니다. CodePipeline에서 새 게시 작업을 생성할 때 이 물리적 ID를 사용합니다.

4단계: 게시 작업 생성

다음 단계에 따라 파이프라인에 게시 작업을 생성합니다.

- 1. https://console.aws.amazon.com/codepipeline/에서 CodePipeline 콘솔을 엽니다.
- 2. 왼쪽 탐색 섹션에서 편집하려는 파이프라인을 선택합니다.
- 3. 편집을 선택합니다.
- 현재 파이프라인의 마지막 스테이지 후 + Add stage(+ 스테이지 추가)를 선택합니다. Stage name(스테이지 이름)에서 이름(예: Publish)을 입력하고 Add stage(스테이지 추가)를 선택합니 다.
- 5. 새 단계에서 + Add action group(작업 그룹 추가)을 선택합니다.
- 6. 작업 이름을 입력합니다. Action provider(작업 공급자)의 호출에서 AWS Lambda을 선택합니다.
- 7. 입력 아티팩트에서 BuildArtifact를 선택합니다.
- 8. 함수 이름에서 이전 단계에서 기록한 Lambda 함수의 물리적 ID를 선택합니다.
- 9. 작업에 대해 저장을 선택합니다.
- 10. 스테이지에 대해 완료를 선택합니다.
- 11. 오른쪽 상단에서 저장을 선택합니다.
- 12. 파이프라인을 확인하려면 GitHub에서 애플리케이션을 변경합니다. 예를 들어 AWS SAM 템플릿 파일의 Metadata 섹션에서 애플리케이션 설명을 변경합니다. 변경 사항을 커밋하고 GitHub 브랜

치로 푸시합니다. 이렇게 하면 파이프라인이 실행됩니다. 파이프라인이 완료되면 애플리케이션이 AWS Serverless Application Repository에서 변경 사항으로 업데이트 되었는지 확인합니다.

자습서: Lambda 호출 작업과 함께 변수 사용

Lambda 호출 작업은 입력의 일부로 다른 작업의 변수를 사용하고 출력과 함께 새로운 변수를 반환할 수 있습니다. CodePipeline의 작업 변수에 대한 자세한 내용은 변수 참조 단원을 참조하십시오.

A Important

파이프라인 생성의 일환으로 CodePipeline은 고객이 제공하는 S3 아티팩트 버킷을 아티팩트 에 사용합니다. (이는 S3 소스 작업에 사용되는 버킷과 다릅니다.) S3 아티팩트 버킷이 파이프 라인의 계정과 다른 계정에 있는 경우 S3 아티팩트 버킷을 안전하고 신뢰할 수 AWS 계정 있는 에서 소유해야 합니다.

이 자습서의 마지막에서는 다음 항목을 갖게 됩니다.

- 다음과 같은 Lambda 호출 작업:
 - CodeCommit 소스 동작에서 CommitId 변수 사용
 - dateTime, testRunId 및 region의 세 가지 새 변수 출력
- Lambda 호출 작업의 새 변수를 사용하여 테스트 URL 및 테스트 실행 ID를 제공하는 수동 승인 작업
- 새 작업으로 업데이트된 파이프라인

주제

- <u>사전 조건</u>
- 1단계: Lambda 함수 생성
- 2단계: 파이프라인에 Lambda 호출 작업 및 수동 승인 작업 추가

사전 조건

시작하기 전에 다음을 완비해야 합니다.

• <u>자습서: 간단한 파이프라인 생성(CodeCommit 리포지토리)</u>에서 CodeCommit 소스와 함께 파이프라 인을 생성하거나 사용할 수 있습니다. CodeCommit 소스 작업에 네임스페이스가 있도록 기존 파이프라인을 편집합니다. 작업에 SourceVariables 네임스페이스를 할당합니다.

1단계: Lambda 함수 생성

Lambda 함수 및 Lambda 실행 역할을 만들려면 다음 단계를 따르세요. Lambda 함수를 만든 후 파이프 라인에 Lambda 작업을 추가합니다.

Lambda 함수 및 실행 역할을 생성하려면

- 1. 에 로그인 AWS Management Console 하고 <u>https://console.aws.amazon.com/lambda/</u>://https:// https://https://https://https://https://https://https://https://
- 2. 함수 생성(Create function)을 선택합니다. Author from scratch(새로 작성)를 선택합니다.
- 3. Function name(함수 이름)에 myInvokeFunction과 같은 함수 이름을 입력합니다. Runtime(실 행 시간)에서 기본 옵션을 선택된 상태로 둡니다.
- 4. 실행 역할 선택 또는 생성을 확장합니다. 기본 Lambda 권한을 가진 새 역할 생성을 선택합니다.
- 5. 함수 생성(Create function)을 선택합니다.
- 다른 작업의 변수를 사용하려면 이 변수를 Lambda 호출 작업 구성에서 UserParameters에 전 달해야 합니다. 자습서의 뒷부분에서 파이프라인에 작업을 구성하지만 변수가 전달될 것으로 가 정하여 코드를 추가합니다.

새 변수를 생성하려면 입력에서 outputVariables라는 속성을 putJobSuccessResult로 설 정합니다. putJobFailureResult의 일부로 변수를 생성할 수 없습니다.

```
const putJobSuccess = async (message) => {
    const params = {
        jobId: jobId,
        outputVariables: {
            testRunId: Math.floor(Math.random() * 1000).toString(),
            dateTime: Date(Date.now()).toString(),
            region: lambdaRegion
        }
    };
```

새 함수의 코드 탭에서 index.mjs 아래에 다음 예제 코드를 붙여 넣습니다.

import { CodePipeline } from '@aws-sdk/client-codepipeline';

```
export const handler = async (event, context) => {
    const codepipeline = new CodePipeline({});
   // Retrieve the Job ID from the Lambda action
   const jobId = event["CodePipeline.job"].id;
   // Retrieve UserParameters
   const params =
 event["CodePipeline.job"].data.actionConfiguration.configuration.UserParameters;
   // The region from where the lambda function is being executed
   const lambdaRegion = process.env.AWS_REGION;
   // Notify CodePipeline of a successful job
    const putJobSuccess = async (message) => {
        const params = {
            jobId: jobId,
            outputVariables: {
                testRunId: Math.floor(Math.random() * 1000).toString(),
                dateTime: Date(Date.now()).toString(),
                region: lambdaRegion
            }
       };
       try {
            await codepipeline.putJobSuccessResult(params);
            return message;
        } catch (err) {
            throw err;
        }
   };
   // Notify CodePipeline of a failed job
    const putJobFailure = async (message) => {
        const params = {
            jobId: jobId,
            failureDetails: {
                message: JSON.stringify(message),
                type: 'JobFailed',
                externalExecutionId: context.invokeid
            }
        };
        try {
```

```
await codepipeline.putJobFailureResult(params);
            throw message;
        } catch (err) {
            throw err;
        }
    };
    try {
        console.log("Testing commit - " + params);
        // Your tests here
        // Succeed the job
        return await putJobSuccess("Tests passed.");
    } catch (ex) {
        // If any of the assertions failed then fail the job
        return await putJobFailure(ex);
    }
};
```

7. 함수가 자동으로 저장되도록 허용합니다.

- 8. 화면 상단에서 함수 ARN 필드에 포함된 Amazon 리소스 이름(ARN)을 복사합니다.
- 9. 마지막 단계로 <u>https://console.aws.amazon.com/iam/</u>://https://https://https://https://https:// https AWS Identity and Access Management ://://https://https://https://https://https:// https://https://ittps://https Lambda 실행 역할을 수정하여 다음 정책을 추가합니다 (<u>AWSCodePipelineCustomActionAccess</u>). Lambda 실행 역할을 생성하거나 역할 정책을 수정하 는 단계는 <u>2단계: Lambda 함수 생성</u> 단원을 참조하십시오.

2단계: 파이프라인에 Lambda 호출 작업 및 수동 승인 작업 추가

이 단계에서는 파이프라인에 Lambda 호출 작업을 추가합니다. [Test(테스트)]라는 단계의 일부로 작업 을 추가합니다. 작업 유형은 호출 작업입니다. 그런 다음 호출 작업 후 수동 승인 작업을 추가합니다.

파이프라인에 Lambda 작업 및 수동 승인 작업을 추가하는 방법

1. <u>https://console.aws.amazon.com/codepipeline/</u>에서 CodePipeline 콘솔을 엽니다.

AWS 계정과 연결된 모든 파이프라인의 이름이 표시됩니다. 작업을 추가할 파이프라인을 선택합 니다.

2. 파이프라인에 Lambda 테스트 작업을 추가합니다.

- a. 파이프라인을 편집하기 위해 편집을 선택합니다. 기존 파이프라인에서 소스 작업 이후에 단 계를 추가합니다. 단계 이름을 입력합니다(예:**Test**).
- b. 새 스테이지에서 작업 그룹 추가를 선택하여 작업을 추가합니다. Action name(작업 이름)에 호출 작업의 이름을 입력합니다(예: **Test_Commit**).
- c. 작업 공급자에서 AWS Lambda을 선택합니다.
- d. Input artifacts(입력 아티팩트)에서 소스 작업의 출력 아티팩트 이름을 선택합니다(예: SourceArtifact).
- e. FunctionName에 생성한 Lambda 함수의 ARN을 추가합니다.
- f. Variable namespace(변수 네임스페이스)에 **TestVariables**와 같은 네임스페이스 이름을 추가합니다.
- g. 출력 아티팩트에서 LambdaArtifact와 같은 출력 아티팩트 이름을 추가합니다.
- h. 완료를 선택합니다.
- 3. 파이프라인에 수동 승인 작업을 추가합니다.
 - a. 파이프라인이 편집 모드 상태인 경우, 호출 작업 후에 단계를 추가합니다. 단계 이름을 입력합 니다(예:**Approval**).
 - b. 새 단계에서 아이콘을 선택하여 작업을 추가합니다. Action name(작업 이름)에 승인 작업의 이름을 입력합니다(예:**Change_Approval**).
 - c. Action provider(작업 공급자)에서 수동 승인을 선택합니다.
 - d. URL for review(검토할 URL)에서 region 변수와 CommitId 변수에 대한 변수 구문을 추가 하여 URL을 구성합니다. 출력 변수를 제공하는 작업에 할당되는 네임스페이스를 사용해야 합니다.

이 예제의 경우 CodeCommit 작업에 대한 변수 구문을 사용하는 URL의 기본 네임스페이스는 SourceVariables입니다. Lambda 리전 출력 변수의 네임스페이스는 TestVariables입 니다. URL은 다음과 같습니다.

https://#{TestVariables.region}.console.aws.amazon.com/codesuite/codecommit/ repositories/MyDemoRepo/commit/#{SourceVariables.CommitId}

Comments(설명)에서 testRunId 변수에 대한 변수 구문을 추가하여 승인 메시지 텍스트 를 구성합니다. 이 예제의 경우 Lambda testRunId 출력 변수에 대한 변수 구문을 사용하는 URL의 네임스페이스는 TestVariables입니다. 다음 메시지를 입력합니다. Make sure to review the code before approving this action. Test Run ID: #{TestVariables.testRunId}

 Done(완료)을 선택하여 작업에 대한 편집 화면을 닫은 다음 Done(완료)을 선택하여 해당 단계의 편집 화면을 닫습니다. 파이프라인을 저장하려면 Done(완료)을 선택합니다. 이제 완료된 파이프 라인에는 소스, 테스트, 승인 및 배포 단계로 구성된 구조가 포함됩니다.

Release change(변경 사항 배포)를 선택하여 파이프라인 구조를 통해 최신 변경 사항을 실행합니다.

- 5. 파이프라인이 수동 승인 단계에 도달하면 Review(검토)를 선택합니다. 해결된 변수는 커밋 ID의 URL로 나타납니다. 승인자는 커밋을 볼 URL을 선택할 수 있습니다.
- 6. 파이프라인이 성공적으로 실행되면 작업 실행 기록 페이지에서 변수 값을 볼 수도 있습니다.

자습서: 파이프라인에서 AWS Step Functions 호출 작업 사용

AWS Step Functions 를 사용하여 상태 시스템을 생성하고 구성할 수 있습니다. 이 자습서에서는 파이 프라인에서 상태 시스템 실행을 활성화하는 파이프라인에 호출 작업을 추가하는 방법을 보여 줍니다.

A Important

파이프라인 생성의 일환으로 CodePipeline은 고객이 제공하는 S3 아티팩트 버킷을 아티팩트 에 사용합니다. (이는 S3 소스 작업에 사용되는 버킷과 다릅니다.) S3 아티팩트 버킷이 파이프 라인의 계정과 다른 계정에 있는 경우 S3 아티팩트 버킷을 안전하고 신뢰할 수 AWS 계정 있는 에서 소유해야 합니다.

이 자습서에서는 다음 태스크를 수행합니다.

- 에서 표준 상태 시스템을 생성합니다 AWS Step Functions.
- 상태 시스템 입력 JSON을 직접 입력합니다. 상태 시스템 입력 파일을 Amazon Simple Storage Service(S3) 버킷에 업로드할 수도 있습니다.
- 상태 시스템 작업을 추가하여 파이프라인을 업데이트합니다.

주제

• 사전 조건: 단순 파이프라인 생성 또는 선택

- 1단계: 샘플 상태 시스템 생성
- 2단계: 파이프라인에 Step Functions 호출 작업 추가

사전 조건: 단순 파이프라인 생성 또는 선택

이 자습서에서는 기존 파이프라인에 호출 작업을 추가합니다. <u>자습서: 간단한 파이프라인 생성(S3 버</u><u>킷)</u> 또는 <u>자습서: 간단한 파이프라인 생성(CodeCommit 리포지토리)</u>에서 생성한 파이프라인을 사용할 수 있습니다.

소스 작업과 최소한 2-스테이지 구조의 기존 파이프라인을 사용하지만 이 예제에서는 소스 아티팩트를 사용하지 않습니다.

1 Note

파이프라인에서 사용하는 서비스 역할을 이 작업을 실행하는 데 필요한 추가 권한으로 업데이 트해야 할 수 있습니다. 이렇게 하려면 AWS Identity and Access Management (IAM) 콘솔을 열고 역할을 찾은 다음 역할 정책에 권한을 추가합니다. 자세한 내용은 <u>CodePipeline 서비스</u> 역할에 권한 추가 단원을 참조하십시오.

1단계: 샘플 상태 시스템 생성

Step Functions 콘솔에서 HelloWorld 샘플 템플릿을 사용하여 상태 시스템을 생성합니다. 지침은 AWS Step Functions 개발자 가이드의 상태 머신 생성을 참조하세요.

2단계: 파이프라인에 Step Functions 호출 작업 추가

다음과 같이 파이프라인에 Step Functions 호출 작업을 추가합니다.

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https//https//https://https//ht

AWS 계정과 연결된 모든 파이프라인의 이름이 표시됩니다.

- [Name]에서 편집할 파이프라인의 이름을 선택합니다. 이렇게 하면 파이프라인 각 단계의 각 작업 상태를 포함하여 파이프라인의 세부 정보 보기가 열립니다.
- 3. 파이프라인 세부 정보 페이지에서 [Edit]를 선택합니다.
- 단순 파이프라인의 두 번째 스테이지에서 스테이지 편집을 선택합니다. Delete(삭제)를 선택합니다.
 다. 그러면 더 이상 필요하지 않으므로 두 번째 스테이지가 삭제됩니다.

- 5. 다이어그램의 하단에서 + 단계 추가를 선택합니다.
- 6. 스테이지 이름에서 스테이지 이름(예: Invoke)을 입력한 다음 스테이지 추가를 선택합니다.
- 7. + Add action group(작업 그룹 추가)을 선택합니다.
- 8. 작업 이름에 이름(예: Invoke)을 입력합니다.
- 9. 작업 공급자에서 AWS Step Functions를 선택합니다. 리전이 파이프라인 리전으로 기본 설정되도 록 합니다.
- 10. 입력 아티팩트에서 SourceArtifact를 선택합니다.
- 11. 상태 시스템 ARN에서 이전에 생성한 상태 시스템의 Amazon 리소스 이름(ARN)을 선택합니다.
- 12. (선택 사항) 실행 이름 접두사에 상태 시스템 실행 ID에 추가할 접두사를 입력합니다.
- 13. 입력 유형에서 리터럴을 선택합니다.
- 14. 입력에 HelloWorld 샘플 상태 시스템이 예상하는 입력 JSON을 입력합니다.

Note

상태 시스템 실행에 대한 입력은 CodePipeline에서 작업에 대한 입력 아티팩트를 설명하 는 데 사용되는 용어와 다릅니다.

이 예제에서는 다음 JSON을 입력합니다.

{"IsHelloWorldExample": true}

- 15. 완료를 선택합니다.
- 16. 편집 중인 스테이지에서 완료를 선택합니다. AWS CodePipeline 창에서 저장을 선택한 다음 경고 메시지에서 저장을 선택합니다.
- 17. 변경 사항을 제출하고 파이프라인 실행을 시작하려면 변경 사항 배포를 선택한 다음 릴리스를 선 택합니다.
- 18. 완료된 파이프라인에서 호출 작업의 AWS Step Functions를 선택합니다. AWS Step Functions 콘 솔에서 상태 시스템 실행 ID를 확인합니다. ID에는 상태 시스템 이름인 Helloworld와 상태 시스 템 실행 ID가 접두사 my-prefix와 함께 표시됩니다.

```
arn:aws:states:us-west-2:account-ID:execution:HelloWorld:my-
prefix-0d9a0900-3609-4ebc-925e-83d9618fcca1
```

자습서: AWS AppConfig를 배포 공급자로 사용하는 파이프라인 생 성

이 자습서에서는 배포 단계의 배포 작업 공급자로 AWS AppConfig를 사용하여 구성 파일을 지속적으 로 전달하는 파이프라인을 구성합니다.

A Important

파이프라인 생성의 일환으로 CodePipeline은 고객이 제공하는 S3 아티팩트 버킷을 아티팩트 에 사용합니다. (이는 S3 소스 작업에 사용되는 버킷과 다릅니다.) S3 아티팩트 버킷이 파이프 라인의 계정과 다른 계정에 있는 경우 S3 아티팩트 버킷을 안전하고 신뢰할 수 AWS 계정 있는 에서 소유해야 합니다.

주제

- <u>사전 조건</u>
- <u>1단계: your AWS AppConfig 리소스 생성</u>
- 2단계: S3 소스 버킷에 파일 업로드
- <u>3단계: 파이프라인 생성</u>
- 4단계: 모든 소스 파일을 변경하고 배포 확인

사전 조건

시작하기 전에 다음을 완료해야 합니다.

• 이 예제에서는 파이프라인에 S3 소스를 사용합니다. 버전 관리가 활성화된 Amazon S3 버킷을 생성 하거나 사용합니다. <u>1단계: 애플리케이션에 대한 S3 소스 버킷 생성</u>의 지침을 따라 S3 버킷을 생성 합니다.

1단계: your AWS AppConfig 리소스 생성

이 섹션에서는 다음 리소스를 생성합니다.

- 애플리케이션 in AWS AppConfig는 고객에게 기능을 제공하는 논리적 코드 단위입니다.
- 환경 in AWS AppConfig는 베타 또는 프로덕션 환경의 애플리케이션과 같은 AppConfig 대상의 논리 적 배포 그룹입니다.

- 구성 프로필은 애플리케이션의 동작에 영향을 미치는 설정의 모음입니다. 구성 프로파일을 사용하 면 AWS AppConfig가 저장된 위치에 있는 구성에 액세스할 수 있습니다.
- (선택 사항) AWS AppConfig의 배포 전략은 배포 중 특정 시점에 새로 배포된 구성을 수신해야 하는 클라이언트의 비율과 같은 구성 배포의 동작을 정의합니다.

애플리케이션, 환경, 구성 프로필 및 배포 전략을 생성하려면

- 1. AWS Management Console에 로그인합니다.
- 2. AWS AppConfig에서 리소스를 생성하려면 다음 주제의 단계를 사용합니다.
 - 애플리케이션 생성.
 - <u>환경을 생성</u>합니다.
 - AWS CodePipeline 구성 프로파일을 생성합니다.
 - (선택 사항) 사전 정의된 배포 전략을 선택 또는 직접 생성.

2단계: S3 소스 버킷에 파일 업로드

이 섹션에서 구성 파일을 생성합니다. 그런 다음 소스 파일을 압축하여 파이프라인이 소스 단계에 대해 사용하는 버킷으로 푸시합니다.

구성 파일을 생성하려면

각 리전의 각 구성에 대한 configuration.json 파일을 생성합니다. 다음 콘텐츠를 포함합니다.
 다.

Hello World!

2. 다음 단계를 사용하여 구성 파일을 압축하고 업로드하세요.

소스 파일을 압축하고 업로드하려면

1. 파일이 포함된 .zip 파일을 생성하고 .zip 파일의 이름을 configuration-files.zip으로 지정 합니다. 예를 들어 .zip 파일은 다음과 같은 구조를 사용할 수 있습니다.

appconfig-configurations
 ### MyConfigurations

```
사용자 가이드
```

```
### us-east-1
# ### configuration.json
### us-west-2
    ### configuration.json
```

2. 버킷의 Amazon S3 콘솔에서 업로드를 선택하고 지침을 따라 .zip 파일을 업로드합니다.

3단계: 파이프라인 생성

이 단원에서는 다음 작업을 통해 파이프라인을 생성합니다.

- 소스 아티팩트가 구성용 파일인 Amazon S3 작업이 있는 소스 단계.
- AppConfig 배포 작업이 적용된 배포 단계.

마법사를 사용하여 파이프라인을 생성하려면

- 1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https//https//https://https//ht
- 2. [Welcome] 페이지, [Getting started] 페이지 또는 [Pipelines] 페이지에서 Create pipeline(파이프라 인생성)을 선택합니다.
- 1단계: 생성 옵션 선택 페이지의 생성 옵션에서 사용자 지정 파이프라인 빌드 옵션을 선택합니다. Next(다음)를 선택합니다.
- 4. 2단계: 파이프라인 설정 선택의 파이프라인 이름에 MyAppConfigPipeline을 입력합니다.
- CodePipeline은 특성과 가격이 다른 V1 및 V2 유형 파이프라인을 제공합니다. V2 유형은 콘 솔에서 선택할 수 있는 유일한 유형입니다. 자세한 내용은 <u>파이프라인 유형</u>을 참조하세요. CodePipeline에 대한 요금 정보는 요금을 참조하세요.
- 서비스 역할에서 새 서비스 역할을 선택하여CodePipeline이 IAM에 서비스 역할을 생성하도록 허용합니다.
- 7. [Advanced settings]의 설정은 기본값 그대로 두고, [Next]를 선택합니다.
- 8. 3단계: 소스 스테이지 추가의 소스 공급자에서 Amazon S3를 선택합니다. 버킷에서 S3 소스 버킷 의 이름을 선택합니다.

S3 객체 키에 .zip 파일: configuration-files.zip의 이름을 입력합니다.

Next(다음)를 선택합니다.
4단계: 빌드 스테이지 추가에서 빌드 스테이지 건너뛰기를 선택하고 건너뛰기를 다시 선택하여 경고 메시지를 수락합니다.

Next(다음)를 선택합니다.

5단계: 테스트 단계 추가에서 테스트 단계 건너뛰기를 선택한 다음 건너뛰기를 다시 선택하여 경고 메시지를 수락합니다.

Next(다음)를 선택합니다.

- 11. 6단계: 배포 단계 추가:
 - a. 배포 공급자에서 AWS AppConfig를 선택합니다.
 - b. 애플리케이션에서 AWS AppConfig에서 생성한 애플리케이션의 이름을 선택합니다. 필드에 애플리케이션의 ID가 표시됩니다.
 - c. 환경에서 AWS AppConfig에서 생성한 환경의 이름을 선택합니다. 필드에 환경의 ID가 표시됩 니다.
 - d. 구성 프로필에서 AWS AppConfig에서 생성한 구성 프로필의 이름을 선택합니다. 필드에 구성 프로필의 ID가 표시됩니다.
 - e. 배포 전략에서 배포 전략의 이름을 선택합니다. 이것은 AppConfig에서 만든 배포 전략이거나 AppConfig에서 미리 정의된 배포 전략 중에서 선택한 전략일 수 있습니다. 필드에는 배포 전 략의 ID가 표시됩니다.
 - f. 입력 아티팩트 구성 경로에 파일 경로를 입력합니다. 입력 아티팩트 구성 경로가 S3 버킷 .zip 파일의 디렉터리 구조와 일치하는지 확인하세요. 이 예제에서는 파일 경로(appconfigconfigurations/MyConfigurations/us-west-2/configuration.json)를 입력합 니다.
 - g. Next(다음)를 선택합니다.
- 12. 7단계: 검토에서 정보를 검토한 다음 파이프라인 생성을 선택합니다.

4단계: 모든 소스 파일을 변경하고 배포 확인

소스 파일을 변경하고 변경 사항을 버킷에 업로드합니다. 이렇게 하면 파이프라인이 실행됩니다. 버전 을 확인하여 구성을 사용할 수 있는지 확인하세요.

자습서: GitHub 파이프라인 소스와 함께 전체 복제 사용

CodePipeline에서 GitHub 소스 작업에 대한 전체 복제 옵션을 선택할 수 있습니다. 이 옵션을 사용하면 파이프라인 빌드 작업에서 Git 메타데이터에 대한 CodeBuild 명령을 실행할 수 있습니다.

Note

여기에 설명된 전체 복제 옵션은 CodePipeline이 CodeBuild 명령에서만 사용할 수 있는 리포 지토리 메타데이터를 복제할지 여부를 지정하는 것을 나타냅니다. CodeBuild 프로젝트에서 사 용할 GitHub <u>사용자 액세스 토큰</u>을 사용하려면 다음 단계에 따라 GitHub용 AWS 커넥터를 설 치한 다음 앱 설치 필드를 비워 둡니다. CodeConnections는 연결에 사용자 액세스 토큰을 사 용합니다.

A Important

파이프라인 생성의 일환으로 CodePipeline은 고객이 제공하는 S3 아티팩트 버킷을 아티팩트 에 사용합니다. (이는 S3 소스 작업에 사용되는 버킷과 다릅니다.) S3 아티팩트 버킷이 파이프 라인의 계정과 다른 계정에 있는 경우 S3 아티팩트 버킷을 안전하고 신뢰할 수 AWS 계정 있는 가 소유해야 합니다.

이 자습서에서는 GitHub 리포지토리에 연결하는 파이프라인을 만들고, 소스 데이터에 대한 전체 복제 옵션을 사용하고, 리포지토리를 복제하고 리포지토리에 대한 Git 명령을 수행하는 CodeBuild 빌드를 실행합니다.

Note

이 기능은 아시아 태평양(홍콩), 아프리카(케이프타운), 중동(바레인), 유럽(취리히) 또는 AWS GovCloud(미국 서부) 리전에서는 사용할 수 없습니다. 사용 가능한 다른 작업을 참조하려면 <u>CodePipeline과 제품 및 서비스 통합</u>을 참조하세요. 유럽(밀라노) 리전에서 이 조치를 고려할 경우 <u>Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com 및 GitLab 자체 관리형</u> <u>작업용 CodeStarSourceConnection</u>의 참고 사항을 참조하세요.

주제

- <u>사전 조건</u>
- <u>1단계: README 파일 생성</u>
- 2단계: 파이프라인 및 빌드 프로젝트 생성
- 3단계: 연결을 사용하도록 CodeBuild 서비스 역할 정책 업데이트
- 4단계: 빌드 출력에서 리포지토리 명령 보기

사전 조건

시작하기 전에 다음을 수행해야 합니다.

- GitHub 계정을 사용하여 GitHub 리포지토리를 만듭니다.
- GitHub 자격 증명을 준비합니다. AWS Management Console 를 사용하여 연결을 설정하면 GitHub 자격 증명으로 로그인하라는 메시지가 표시됩니다.

1단계: README 파일 생성

GitHub 리포지토리를 만든 후 다음 단계를 사용하여 README 파일을 추가하세요.

- 1. GitHub 리포지토리에 로그인하고 리포지토리를 선택합니다.
- 새 파일을 생성하려면 파일 추가 > 새 파일 생성을 선택합니다. 파일 이름을 README.md. 파일로 지정하고 다음 텍스트를 추가합니다.

This is a GitHub repository!

3. 변경 사항 커밋을 선택합니다.

README.md 파일이 리포지토리의 루트 수준에 있는지 확인합니다.

2단계: 파이프라인 및 빌드 프로젝트 생성

이 단원에서는 다음 작업을 통해 파이프라인을 생성합니다.

- GitHub 리포지토리 및 작업에 대한 연결이 있는 소스 단계입니다.
- 빌드 작업이 있는 AWS CodeBuild 빌드 단계입니다.

마법사를 사용하여 파이프라인을 생성하려면

- 1. https://console.aws.amazon.com/codepipeline/에서 CodePipeline 콘솔에 로그인합니다.
- 2. Welcome(시작) 페이지, 시작하기 페이지 또는 Pipelines(파이프라인) 페이지에서 파이프라인 생성을 선택합니다.
- 3. 1단계: 생성 옵션 선택 페이지의 생성 옵션에서 사용자 지정 파이프라인 빌드 옵션을 선택합니다. Next(다음)를 선택합니다.
- 4. 2단계: 파이프라인 설정 선택의 파이프라인 이름에 MyGitHubPipeline을 입력합니다.

- 파이프라인 유형에서 이 자습서의 목적에 맞는 V1을 선택합니다. V2를 선택할 수도 있지만 파이 프라인 유형별 특성과 가격이 다르다는 점에 유의하십시오. 자세한 내용은 <u>파이프라인 유형</u> 단원 을 참조하십시오.
- 6. Service role(서비스 역할)에서 New service role(새 서비스 역할)을 선택합니다.

Note

역할을 생성하지 않고 기존 CodePipeline 서비스 역할을 사용할 경우 서비스 역할 정책 에 대한 codestar-connections:UseConnection IAM 권한을 추가해야 합니다. CodePipeline 서비스 역할에 대한 지침은 <u>CodePipeline 서비스 역할에 권한 추가</u>를 참조 하세요.

7. 고급 설정에서 기본값을 그대로 둡니다. [아티팩트 스토어(Artifact store)]에서 [기본 위치(Default location)]를 선택하여 파이프라인에 대해 선택한 리전의 파이프라인에 대해 기본값으로 지정된 Amazon S3 아티팩트 버킷과 같은 기본 아티팩트 스토어를 사용합니다.

Note

이는 소스 코드에 대한 소스 버킷이 아닙니다. 이 파이프라인은 아티팩트 스토어입니다. S3 버킷과 같은 개별 아티팩트 스토어는 각 파이프라인에 필요합니다.

Next(다음)를 선택합니다.

- 8. 3단계: 소스 스테이지 추가 페이지에서 소스 스테이지를 추가합니다.
 - a. 소스 공급자에서 GitHub(GitHub 앱을 통해)를 선택합니다.
 - b. 연결에서 기존 연결을 선택하거나 새로 생성합니다. GitHub 소스 작업에 대한 연결을 만들거 나 관리하려면 GitHub 연결을 참조하세요.

특정 공급자에 대한 모든 연결에 대해 하나의 앱을 설치합니다. AWS Connector for GitHub 앱을 이미 설치한 경우 해당 앱을 선택하고이 단계를 건너뜁니다.

In Note

<u>사용자 액세스 토큰</u>을 생성하려면 AWS Connector for GitHub 앱을 이미 설치했는지 확인한 다음 앱 설치 필드를 비워 둡니다. CodeConnections는 연결에 사용자 액세스 토큰을 사용합니다. 자세한 내용은 CodeBuild의 소스 공급자 액세스를 참조하세요.

- c. 리포지토리 이름에서 GitHub 리포지토리의 이름을 선택합니다.
- d. 브랜치 이름에서 사용할 리포지토리 브랜치를 선택합니다.
- e. 소스 코드 변경 시 파이프라인 시작 옵션을 선택합니다.
- f. 출력 아티팩트 형식에서 전체 복제를 선택하여 소스 리포지토리의 Git 복제 옵션을 활성화합니다. CodeBuild에서 제공하는 작업만 Git 복제 옵션을 사용할 수 있습니다. 이 자습서에서는 CodeBuild 프로젝트 서비스 역할이 이 옵션을 사용하도록 권한을 업데이트하는 데 <u>3단계: 연</u> 결을 사용하도록 CodeBuild 서비스 역할 정책 업데이트을 사용합니다.

Next(다음)를 선택합니다.

- 9. 4단계: 빌드 단계 추가에서 빌드 단계를 추가합니다.
 - a. 빌드 공급자에서 AWS CodeBuild를 선택합니다. 리전이 파이프라인 리전으로 기본 설정되도 록 합니다.
 - b. 프로젝트 생성을 선택합니다.
 - c. 프로젝트 이름에 이 빌드 프로젝트의 이름을 입력합니다.
 - d. 환경 이미지에서 이미지 관리를 선택합니다. [Operating system]에서 [Ubuntu]를 선택합니다.
 - e. 실행 시간에서 표준을 선택합니다. 이미지에서 aws/codebuild/standard:5.0을 선택합니다.
 - f. 서비스 역할에서 New service role(새 서비스 역할)을 선택합니다.

Note

CodeBuild 서비스 역할의 이름을 기록해 둡니다. 이 자습서의 마지막 단계를 수행하 려면 역할 이름이 필요합니다.

g. [Buildspec]의 [빌드 사양(Build specifications)]에서 [빌드 명령 삽입(Insert build commands)] 을 선택합니다. 편집기로 전환을 선택하고 빌드 명령에 다음을 붙여 넣습니다.

Note

빌드 사양의 env 섹션에서 이 예제와 같이 git 명령의 보안 인증 도우미가 활성화되어 있는지 확인하세요.

version: 0.2

2단계: 파이프라인 및 빌드 프로젝트 생성

```
env:
  git-credential-helper: yes
phases:
  install:
    #If you use the Ubuntu standard image 2.0 or later, you must specify
 runtime-versions.
    #If you specify runtime-versions and use an image other than Ubuntu
 standard image 2.0, the build fails.
    runtime-versions:
      nodejs: 12
      # name: version
    #commands:
      # - command
      # - command
  pre_build:
    commands:
      - ls -lt
      - cat README.md
  build:
    commands:
      - git log | head -100
      - git status
      - ls
      - git archive --format=zip HEAD > application.zip
  #post build:
    #commands:
      # - command
      # - command
artifacts:
  files:
    - application.zip
    # - location
  #name: $(date +%Y-%m-%d)
  #discard-paths: yes
  #base-directory: location
#cache:
  #paths:
    # - paths
```

h. Continue to CodePipeline(CodePipeline으로 계속)을 선택합니다. 그러면 CodePipeline 콘솔 로 돌아가고 구성을 위해 빌드 명령을 사용하는 CodeBuild 프로젝트가 생성됩니다. 빌드 프로 젝트는 서비스 역할을 사용하여 AWS 서비스 권한을 관리합니다. 이 단계는 몇 분이 걸릴 수 있습니다.

- i. Next(다음)를 선택합니다.
- 5단계: 테스트 단계 추가에서 테스트 단계 건너뛰기를 선택한 다음 건너뛰기를 다시 선택하여 경고 메시지를 수락합니다.

Next(다음)를 선택합니다.

- 11. 6단계: 배포 단계 추가 페이지에서 배포 단계 건너뛰기를 선택한 다음 다시 건너뛰기를 선택하여 경고 메시지를 수락합니다. Next(다음)를 선택합니다.
- 12. 7단계: 검토에서 파이프라인 생성을 선택합니다.

3단계: 연결을 사용하도록 CodeBuild 서비스 역할 정책 업데이트

CodeBuild 서비스 역할을 연결 사용 권한으로 업데이트해야 하므로 초기 파이프라인 실행이 실패합니 다. 서비스 역할 정책에 codestar-connections:UseConnection IAM 권한을 추가합니다. IAM 콘솔에서 정책을 업데이트하는 방법에 대한 지침은 <u>Bitbucket, GitHub, GitHub Enterprise Server 또는</u> <u>GitLab.com 연결을 위한 CodeBuild GitClone 권한 추가</u>을 참조하세요.

4단계: 빌드 출력에서 리포지토리 명령 보기

- 1. 서비스 역할이 성공적으로 업데이트되면 실패한 CodeBuild 단계에서 재시도를 선택합니다.
- 파이프라인이 성공적으로 실행되면 성공적으로 완료된 빌드 단계에서 세부 정보 보기를 선택합니다.

세부 정보 페이지에서 로그 탭을 선택합니다. CodeBuild 빌드 출력을 확인합니다. 명령은 입력된 변수의 값을 출력합니다.

명령은 README.md 파일 콘텐츠를 출력하고, 디렉터리의 파일을 나열하고, 리포지토리를 복제하고, 로그를 보고, 리포지토리를 ZIP 파일로 보관합니다.

자습서: CodeCommit 파이프라인 소스와 함께 전체 복제 사용

CodePipeline에서 CodeCommit 소스 작업에 대한 전체 복제 옵션을 선택할 수 있습니다. 이 옵션을 사 용하면 CodeBuild가 파이프라인 빌드 작업에서 Git 메타데이터에 액세스할 수 있습니다.

이 자습서에서는 CodeCommit 리포지토리에 액세스하는 파이프라인을 만들고, 소스 데이터에 대한 전 체 복제 옵션을 사용하고, 리포지토리를 복제하고 리포지토리에 대한 Git 명령을 수행하는 CodeBuild 빌드를 실행합니다.

Note

CodeBuild 작업은 Git 복제 옵션과 함께 사용할 수 있는 Git 메타데이터 사용을 지원하는 유일 한 다운스트림 작업입니다. 또한 파이프라인에 계정 간 작업이 포함될 수 있지만 전체 복제 옵 션이 성공하려면 CodeCommit 작업과 CodeBuild 작업이 동일한 계정에 있어야 합니다.

A Important

파이프라인 생성의 일환으로 CodePipeline은 고객이 제공하는 S3 아티팩트 버킷을 아티팩트 에 사용합니다. (이는 S3 소스 작업에 사용되는 버킷과 다릅니다.) S3 아티팩트 버킷이 파이프 라인의 계정과 다른 계정에 있는 경우 S3 아티팩트 버킷을 안전하고 신뢰할 수 AWS 계정 있는 가 소유해야 합니다.

주제

• <u>사전 조건</u>

- <u>1단계: README 파일 생성</u>
- 2단계: 파이프라인 및 빌드 프로젝트 생성
- 3단계: 리포지토리를 복제하도록 CodeBuild 서비스 역할 정책 업데이트
- 4단계: 빌드 출력에서 리포지토리 명령 보기

사전 조건

시작하기 전에 파이프라인과 동일한 AWS 계정 및 리전에 CodeCommit 리포지토리를 생성해야 합니 다.

1단계: README 파일 생성

다음 단계를 사용하여 소스 리포지토리에 README 파일을 추가합니다. README 파일은 CodeBuild 다운스트림 작업에서 읽을 수 있는 예제 소스 파일을 제공합니다.

README 파일을 추가하려면

- 1. 리포지토리에 로그인하고 리포지토리를 선택합니다.
- 새 파일을 생성하려면 파일 추가 > 파일 생성을 선택합니다. 파일 이름을 README.md. 파일로 지 정하고 다음 텍스트를 추가합니다.

This is a CodeCommit repository!

3. 변경 사항 커밋을 선택합니다.

README.md 파일이 리포지토리의 루트 수준에 있는지 확인합니다.

2단계: 파이프라인 및 빌드 프로젝트 생성

이 단원에서는 다음 작업을 통해 파이프라인을 생성합니다.

- CodeCommit 소스 작업이 있는 소스 단계입니다.
- 빌드 작업이 있는 AWS CodeBuild 빌드 단계입니다.

마법사를 사용하여 파이프라인을 생성하려면

- 1. https://console.aws.amazon.com/codepipeline/에서 CodePipeline 콘솔에 로그인합니다.
- 2. Welcome(시작) 페이지, 시작하기 페이지 또는 Pipelines(파이프라인) 페이지에서 파이프라인 생성을 선택합니다.
- 1단계: 생성 옵션 선택 페이지의 생성 옵션에서 사용자 지정 파이프라인 빌드 옵션을 선택합니다. Next(다음)를 선택합니다.
- 4. 2단계: 파이프라인 설정 선택의 파이프라인 이름에 MyCodeCommitPipeline을 입력합니다.
- CodePipeline은 특성과 가격이 다른 V1 및 V2 유형 파이프라인을 제공합니다. V2 유형은 콘 솔에서 선택할 수 있는 유일한 유형입니다. 자세한 내용은 <u>파이프라인 유형</u>을 참조하세요. CodePipeline에 대한 요금 정보는 요금을 참조하세요.
- 6. 서비스 역할에서 다음 중 하나를 수행합니다.
 - Existing service role(기존 서비스 역할)을 선택합니다.
 - 기존 CodePipeline 서비스 역할을 선택합니다. 이 역할에는 서비스 역할 정책에 대한 codecommit:GetRepository IAM 권한이 있어야 합니다. <u>CodePipeline 서비스 역할에 권</u> 한 추가를 참조하세요.
- 7. 고급 설정에서 기본값을 그대로 둡니다. Next(다음)를 선택합니다.
- 8. 3단계: 소스 스테이지 추가 페이지에서 다음을 수행합니다.
 - a. 소스 공급자에서 CodeCommit을 선택합니다.
 - b. 리포지토리 이름에서 리포지토리의 이름을 선택합니다.

- c. 브랜치 이름에서 브랜치 이름을 선택합니다.
- d. 소스 코드 변경 시 파이프라인 시작 옵션을 선택합니다.
- e. 출력 아티팩트 형식에서 전체 복제를 선택하여 소스 리포지토리의 Git 복제 옵션을 활성화합니다. CodeBuild에서 제공하는 작업만 Git 복제 옵션을 사용할 수 있습니다.

Next(다음)를 선택합니다.

- 9. 4단계: 빌드 단계 추가에서 다음을 수행합니다.
 - a. 빌드 공급자에서 AWS CodeBuild를 선택합니다. 리전이 파이프라인 리전으로 기본 설정되도 록 합니다.
 - b. 프로젝트 생성을 선택합니다.
 - c. 프로젝트 이름에 이 빌드 프로젝트의 이름을 입력합니다.
 - d. 환경 이미지에서 이미지 관리를 선택합니다. [Operating system]에서 [Ubuntu]를 선택합니다.
 - e. 실행 시간에서 표준을 선택합니다. 이미지에서 aws/codebuild/standard:5.0을 선택합니다.
 - f. 서비스 역할에서 New service role(새 서비스 역할)을 선택합니다.

Note

CodeBuild 서비스 역할의 이름을 기록해 둡니다. 이 자습서의 마지막 단계를 수행하 려면 역할 이름이 필요합니다.

g. [Buildspec]의 [빌드 사양(Build specifications)]에서 [빌드 명령 삽입(Insert build commands)] 을 선택합니다. 편집기로 전환을 선택한 다음 빌드 명령에 다음 코드를 붙여 넣습니다.

```
version: 0.2
env:
  git-credential-helper: yes
phases:
  install:
    #If you use the Ubuntu standard image 2.0 or later, you must specify
runtime-versions.
    #If you specify runtime-versions and use an image other than Ubuntu
standard image 2.0, the build fails.
    runtime-versions:
        nodejs: 12
        # name: version
```

```
#commands:
      # - command
      # - command
  pre_build:
    commands:
      - ls -lt
      - cat README.md
  build:
    commands:
      - git log | head -100
      - git status
      - ls
      - git describe --all
  #post_build:
    #commands:
      # - command
      # - command
#artifacts:
  #files:
    # - location
  #name: $(date +%Y-%m-%d)
 #discard-paths: yes
  #base-directory: location
#cache:
  #paths:
    # - paths
```

- h. Continue to CodePipeline(CodePipeline으로 계속)을 선택합니다. 그러면 CodePipeline 콘솔 로 돌아가고 구성을 위해 빌드 명령을 사용하는 CodeBuild 프로젝트가 생성됩니다. 빌드 프로 젝트가 서비스 역할을 사용하여 AWS 서비스 권한을 관리합니다. 이 단계는 몇 분이 걸릴 수 있습니다.
- i. Next(다음)를 선택합니다.
- 5단계: 테스트 단계 추가에서 테스트 단계 건너뛰기를 선택한 다음 건너뛰기를 다시 선택하여 경고 메시지를 수락합니다.

Next(다음)를 선택합니다.

- 11. 6단계: 배포 단계 추가 페이지에서 배포 단계 건너뛰기를 선택한 다음 다시 건너뛰기를 선택하여 경고 메시지를 수락합니다. Next(다음)를 선택합니다.
- 12. 7단계: 검토에서 파이프라인 생성을 선택합니다.

3단계: 리포지토리를 복제하도록 CodeBuild 서비스 역할 정책 업데이트

리포지토리에서 가져올 수 있는 권한으로 CodeBuild 서비스 역할을 업데이트해야 하므로 초기 파이프 라인 실행이 실패합니다.

서비스 역할 정책에 codecommit:GitPull IAM 권한을 추가합니다. IAM 콘솔에서 정책을 업데이트 하는 방법에 대한 지침은 CodeCommit 소스 작업에 대한 CodeBuild GitClone 권한 추가을 참조하세요.

4단계: 빌드 출력에서 리포지토리 명령 보기

빌드 출력을 보려면

- 1. 서비스 역할이 성공적으로 업데이트되면 실패한 CodeBuild 단계에서 재시도를 선택합니다.
- 파이프라인이 성공적으로 실행되면 성공적으로 완료된 빌드 단계에서 세부 정보 보기를 선택합니다.

세부 정보 페이지에서 로그 탭을 선택합니다. CodeBuild 빌드 출력을 확인합니다. 명령은 입력된 변수의 값을 출력합니다.

명령은 README.md 파일 콘텐츠를 출력하고, 디렉터리의 파일을 나열하고, 리포지토리를 복제하고, 로그를 보고, git describe --all을 실행합니다.

자습서: AWS CloudFormation StackSets 배포 작업을 사용하여 파 이프라인 생성

이 자습서에서는 AWS CodePipeline 콘솔을 사용하여 스택 세트를 생성하고 스택 인스턴스를 생성하 기 위한 배포 작업이 포함된 파이프라인을 생성합니다. 파이프라인이 실행되면 템플릿은 스택 세트를 생성하고 스택 세트가 배포된 인스턴스를 생성 및 업데이트합니다.

A Important

파이프라인 생성의 일환으로 CodePipeline은 고객이 제공하는 S3 아티팩트 버킷을 아티팩트 에 사용합니다. (이는 S3 소스 작업에 사용되는 버킷과 다릅니다.) S3 아티팩트 버킷이 파이프 라인의 계정과 다른 계정에 있는 경우 S3 아티팩트 버킷을 안전하고 신뢰할 수 AWS 계정 있는 에서 소유해야 합니다. 스택 세트에 대한 권한을 관리하는 방법에는 자체 관리형 및 AWS관리형 IAM 역할이라는 두 가지가 있 습니다. 이 자습서에서는 자체 관리형 권한의 예를 제공합니다.

CodePipeline에서 Stacksets를 가장 효과적으로 사용하려면 AWS CloudFormation StackSets의 개념 과 작동 방식을 명확하게 이해해야 합니다.AWS CloudFormation 사용 설명서의 <u>StackSets 개념</u>을 참 조하세요.

주제

- <u>사전 조건</u>
- 1단계: 샘플 AWS CloudFormation 템플릿 및 파라미터 파일 업로드
- 2단계: 파이프라인 생성
- 3단계: 초기 배포 보기
- 4단계: CloudFormationStackInstances 작업 추가
- 5단계: 배포를 위한 스택 세트 리소스 보기
- 6단계: 스택 세트 업데이트

사전 조건

스택 세트 작업의 경우 관리자 계정과 대상 계정이라는 두 개의 다른 계정을 사용합니다. 관리자 계정 에서 스택 세트를 생성합니다. 대상 계정에서 스택 세트에 속하는 개별 스택을 생성합니다.

관리자 계정으로 관리자 역할을 만들려면

• <u>스택 세트 작업에 대한 기본 권한 설정</u>의 지침을 따릅니다. 역할 이름은 AWSCloudFormationStackSetAdministrationRole이어야 합니다.

대상 계정에서 서비스 역할을 생성하려면

• 관리자 계정을 신뢰하는 대상 계정에서 서비스 역할을 생성합니다. <u>스택 세트 작업에 대한 기본 권</u> 한 설정의 지침을 따릅니다. 역할 이름은 AWSCloudFormationStackSetExecutionRole이어 야 합니다.

1단계: 샘플 AWS CloudFormation 템플릿 및 파라미터 파일 업로드

스택 세트 템플릿과 파라미터 파일에 대한 소스 버킷을 생성합니다. 샘플 AWS CloudFormation 템플릿 파일을 다운로드하고 파라미터 파일을 설정한 다음 S3 소스 버킷에 업로드하기 전에 파일을 압축합니 다.

Note

소스 파일이 템플릿뿐이더라도 S3 소스 버킷에 업로드하기 전에 소스 파일을 압축해야 합니 다.

S3 소스 버킷을 생성하려면

- 1. 에 로그인 AWS Management Console 하고 <u>https://console.aws.amazon.com/s3/</u>://https://https:// https://https://://https://://https://i/https://https://https://https://https://
- 2. 버킷 생성을 선택합니다.
- 3. 버킷 이름에 버킷 이름을 입력합니다.

리전에서 파이프라인을 생성하려는 리전을 선택합니다. 버킷 생성을 선택합니다.

- 4. 버킷이 생성되면 성공적으로 수행했다는 배너가 표시됩니다. [Go to bucket details]를 선택합니다.
- 5. [Properties] 탭에서 [Versioning]을 선택합니다. [Enable versioning]을 선택한 다음 [Save]를 선택 합니다.

AWS CloudFormation 템플릿 파일을 생성하려면

- 1. 스택 세트에 대한 CloudTrail 구성을 생성하기 위한 샘플 템플릿 파일(<u>https://s3.amazonaws.com/</u> <u>cloudformation-stackset-sample-templates-us-east-1/EnableAWSCloudtrail.yml</u>)을 다운로드하세 요.
- 2. 파일을 template.yml(으)로 저장합니다.

parameters.txt 파일을 생성하려면

 배포용 파라미터가 포함된 파일을 생성합니다. 파라미터는 런타임 시 스택에서 업데이트하려는 값입니다. 다음 샘플 파일은 스택 세트의 템플릿 파라미터를 업데이트하여 로깅 검증 및 글로벌 이 벤트를 활성화합니다.

```
사용자 가이드
```

2. 파일을 parameters.txt(으)로 저장합니다.

accounts.txt 파일을 생성하려면

1. 다음 샘플 파일과 같이 인스턴스를 만들려는 계정이 포함된 파일을 생성합니다.

```
[
"11111222222","333333444444"
]
```

2. 파일을 accounts.txt(으)로 저장합니다.

소스 파일을 압축하고 업로드하려면

1. 파일을 하나의 ZIP 파일로 결합합니다. 파일은 ZIP 파일에 다음과 같이 나타납니다.

```
template.yml
parameters.txt
accounts.txt
```

 ZIP 파일을 S3 버킷에 업로드합니다. 이 파일은 파이프라인 생성 마법사가 CodePipeline에서 배 포 작업을 위해 생성한 소스 아티팩트입니다.

2단계: 파이프라인 생성

이 단원에서는 다음 작업을 통해 파이프라인을 생성합니다.

• 소스 아티팩트가 템플릿 파일 및 모든 지원 소스 파일인 S3 작업이 있는 소스 단계입니다.

- AWS CloudFormation 스택 세트를 생성하는 스택 세트 배포 작업이 있는 배포 단계입니다.
- 대상 계정 내에 AWS CloudFormation 스택과 인스턴스를 생성하는 스택 인스턴스 배포 작업이 있는 배포 단계입니다.

CloudFormationStackSet 작업을 사용하여 파이프라인을 만들려면

- 1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https//https//
- 2. Welcome(시작) 페이지, 시작하기 페이지 또는 Pipelines(파이프라인) 페이지에서 파이프라인 생성을 선택합니다.
- 1단계: 생성 옵션 선택 페이지의 생성 옵션에서 사용자 지정 파이프라인 빌드 옵션을 선택합니다. Next(다음)를 선택합니다.
- 4. 2단계: 파이프라인 설정 선택의 파이프라인 이름에 MyStackSetsPipeline을 입력합니다.
- 파이프라인 유형에서 이 자습서의 목적에 맞는 V1을 선택합니다. V2를 선택할 수도 있지만 파이 프라인 유형별 특성과 가격이 다르다는 점에 유의하십시오. 자세한 내용은 <u>파이프라인 유형</u> 단원 을 참조하십시오.
- 서비스 역할에서 새 서비스 역할을 선택하여CodePipeline이 IAM에 서비스 역할을 생성하도록 허 용합니다.
- 7. 아티팩트 스토어에서는 기본값을 그대로 유지합니다.

Note

이는 소스 코드에 대한 소스 버킷이 아닙니다. 이 파이프라인은 아티팩트 스토어입니다. S3 버킷과 같은 개별 아티팩트 스토어는 각 파이프라인에 필요합니다. 파이프라인을 생성 하거나 편집할 때 파이프라인 리전에 아티팩트 버킷이 하나 있어야 하며 작업을 실행하는 AWS 리전당 하나의 아티팩트 버킷이 있어야 합니다. 자세한 내용은 <u>입력 및 출력 아티팩트</u> 및 <u>CodePipeline 파이프라인 구조 참조</u> 단원을 참조 하세요.

Next(다음)를 선택합니다.

- 8. 3단계: 소스 스테이지 추가 페이지의 소스 공급자에서 Amazon S3를 선택합니다.
- 9. 버킷에 이 자습서를 위해 생성한 S3 소스 버킷(예: BucketName)을 입력합니다. S3 객체 키에 ZIP 파일의 파일 경로와 파일 이름(예: MyFiles.zip)을 입력합니다.

10. Next(다음)를 선택합니다.

11. 4단계: 빌드 스테이지 추가에서 빌드 스테이지 건너뛰기를 선택하고 건너뛰기를 다시 선택하여 경 고 메시지를 수락합니다.

Next(다음)를 선택합니다.

12. 5단계: 테스트 단계 추가에서 테스트 단계 건너뛰기를 선택한 다음 건너뛰기를 다시 선택하여 경 고 메시지를 수락합니다.

Next(다음)를 선택합니다.

- 13. 6단계: 배포 단계 추가:
 - a. 배포 공급자에서 AWS CloudFormation 스택 세트를 선택합니다.
 - b. 스택 세트 이름에 스택 세트의 이름을 입력합니다. 템플릿이 생성하는 스택 세트의 이름입니 다.

Note

스택 세트 이름을 적어 둡니다. 파이프라인에 두 번째 StackSets 배포 작업을 추가할 때 사용합니다.

c. 템플릿 경로에 템플릿 파일을 업로드한 아티팩트 이름과 파일 경로를 입력합니다. 예를 들어, 기본 소스 아티팩트 이름 SourceArtifact를 사용하여 다음을 입력합니다.

SourceArtifact::template.yml

d. 배포 대상에서 계정 파일을 업로드한 아티팩트 이름과 파일 경로를 입력합니다. 예를 들어, 기
 본 소스 아티팩트 이름 SourceArtifact를 사용하여 다음을 입력합니다.

SourceArtifact::accounts.txt

- e. 배포 대상 AWS 리전에와 같이 초기 스택 인스턴스를 배포할 리전 하나를 입력합니다useast-1.
- f. 배포 옵션을 확장합니다. 파라미터에 파라미터 파일을 업로드한 아티팩트 이름과 파일 경로 를 입력합니다. 예를 들어, 기본 소스 아티팩트 이름 SourceArtifact를 사용하여 다음을 입력합니다.

SourceArtifact::parameters.txt

파라미터를 파일 경로가 아닌 리터럴 입력으로 입력하려면 다음을 입력합니다.

ParameterKey=EnableLogFileValidation,ParameterValue=true ParameterKey=IncludeGlobalEvents,ParameterValue=true

- g. 기능에서 CAPABILITY_IAM 및 CAPABILITY_NAMED_IAM을 선택합니다.
- h. 권한 모델에서 SELF_MANAGED를 선택합니다.
- i. 내결함성 비율에 20을 입력합니다.
- j. 최대 동시 비율에 25를 입력합니다.
- k. Next(다음)를 선택합니다.
- I. 7단계: 검토에서 파이프라인 생성을 선택합니다. 파이프라인이 표시됩니다.
- m. 파이프라인이 실행되도록 허용합니다.

3단계: 초기 배포 보기

초기 배포에 대한 리소스와 상태를 확인합니다. 배포에서 스택 세트가 성공적으로 생성되었는지 확인 한 후 배포 단계에 두 번째 작업을 추가할 수 있습니다.

리소스를 보려면

- 1. https://console.aws.amazon.com/codepipeline/에서 CodePipeline 콘솔을 엽니다.
- 파이프라인에서 파이프라인을 선택하고 보기를 선택합니다. 다이어그램은 파이프라인 소스 및 배 포 단계를 보여줍니다.
- 3. 파이프라인의 CloudFormationStackSet AWS CloudFormation 작업에서 작업을 선택합니다. 스택 세트의 템플릿, 리소스 및 이벤트는 AWS CloudFormation 콘솔에 표시됩니다.
- 4. 왼쪽 탐색 창에서 StackSets를 선택합니다. 목록에서 새 스택 세트를 선택합니다.
- 5. 스택 인스턴스 탭을 선택합니다. 제공한 각 계정의 스택 인스턴스 1개가 us-east-1 리전에서 생성 되었는지 확인합니다. 각 스택 인스턴스의 상태가 CURRENT인지 확인하세요.

4단계: CloudFormationStackInstances 작업 추가

파이프라인에서 AWS CloudFormation StackSets가 나머지 스택 인스턴스를 생성하도록 허용하는 다 음 작업을 생성합니다.

파이프라인에서 다음 작업을 만들려면

1. https://console.aws.amazon.com/codepipeline/에서 CodePipeline 콘솔을 엽니다.

파이프라인에서 파이프라인을 선택하고 보기를 선택합니다. 다이어그램은 파이프라인 소스 및 배 포 단계를 보여줍니다.

- 2. 파이프라인을 편집하도록 선택합니다. 파이프라인이 편집 모드에 표시됩니다.
- 3. 배포 단계에서 편집을 선택합니다.
- 4. AWS CloudFormation 스택 세트 배포 작업에서 작업 그룹 추가를 선택합니다.
- 5. 작업 편집 페이지에서 작업 세부 정보를 추가합니다.
 - a. 작업 이름에 작업 이름을 입력합니다.
 - b. 작업 공급자에서 AWS CloudFormation 스택 인스턴스를 선택합니다.
 - c. 입력 아티팩트에서 SourceArtifact를 선택합니다.
 - d. 스택 세트 이름에 스택 세트의 이름을 입력합니다. 첫 번째 작업에서 제공한 스택 세트의 이름 입니다.
 - e. 배포 대상에서 계정 파일을 업로드한 아티팩트 이름과 파일 경로를 입력합니다. 예를 들어, 기 본 소스 아티팩트 이름 SourceArtifact를 사용하여 다음을 입력합니다.

SourceArtifact::accounts.txt

f. 배포 대상 AWS 리전에 다음과 eu-central-1 같이 us-east-2 및와 같이 나머지 스택 인 스턴스를 배포할 리전을 입력합니다.

us-east2, eu-central-1

- g. 내결함성 비율에 20을 입력합니다.
- h. 최대 동시 비율에 25를 입력합니다.
- i. 저장(Save)을 선택합니다.
- j. .변경 사항을 수동으로 릴리스합니다. 업데이트된 파이프라인은 배포 단계에서 두 가지 작업 과 함께 표시됩니다.

5단계: 배포를 위한 스택 세트 리소스 보기

스택 세트 배포에 대한 리소스와 상태를 확인할 수 있습니다.

리소스를 보려면

1. https://console.aws.amazon.com/codepipeline/에서 CodePipeline 콘솔을 엽니다.

- 파이프라인에서 파이프라인을 선택한 후 보기를 선택합니다. 다이어그램은 파이프라인 소스 및 배포 단계를 보여줍니다.
- 3. 파이프라인의 AWS CloudFormation 작업에 대한 AWS CloudFormation Stack Instances 작업을 선택합니다. 스택 세트의 템플릿, 리소스 및 이벤트는 AWS CloudFormation 콘솔에 표시됩 니다.
- 4. 왼쪽 탐색 창에서 StackSets를 선택합니다. 목록에서 스택 세트를 선택합니다.
- 5. 스택 인스턴스 탭을 선택합니다. 제공한 각 계정의 나머지 모든 스택 인스턴스가 예상 리전에서 생 성되거나 업데이트되었는지 확인하세요. 각 스택 인스턴스의 상태가 CURRENT인지 확인하세요.

6단계: 스택 세트 업데이트

스택 세트를 업데이트하고 인스턴스에 업데이트를 배포합니다. 이 예시에서는 업데이트를 위해 지정 하려는 배포 대상도 변경합니다. 업데이트에 포함되지 않은 인스턴스는 오래됨 상태로 전환됩니다.

- 1. https://console.aws.amazon.com/codepipeline/에서 CodePipeline 콘솔을 엽니다.
- 2. 파이프라인에서 파이프라인을 선택한 후 편집을 선택합니다. 배포 단계에서 편집을 선택합니다.
- 3. 파이프라인에서 AWS CloudFormation 스택 세트 작업을 편집하도록 선택합니다. 설명에서 스택 세트에 대한 새 설명으로 기존 설명을 덮어씁니다.
- 4. 파이프라인에서 AWS CloudFormation 스택 인스턴스 작업을 편집하도록 선택합니다. 배포 대상 AWS 리전에서 작업이 생성될 때 입력한 us-east-2 값을 삭제합니다.
- 5. 변경 사항을 저장합니다. 변경 사항 릴리스를 선택하여 파이프라인을 실행합니다.
- 6. AWS CloudFormation에서 작업을 엽니다. StackSet 정보 탭을 선택합니다. StackSet 설명에서 새 설명이 표시되는지 확인합니다.
- 7. 스택 인스턴스 탭을 선택합니다. 상태에서 us-east-2의 스택 인스턴스 상태가 OUTDATED인지 확 인합니다.

자습서: 입력 조건으로 파이프라인에 대한 변수 확인 규칙 생성

이 자습서에서는 소스 스테이지에서 소스 작업 공급자로 GitHub를 사용하여 파일을 지속적으로 제공 하는 파이프라인을 구성합니다. 완성된 파이프 라인은 소스 리포지토리의 소스 파일을 변경할 때 변경 사항을 감지합니다. 파이프라인은 실행된 후, 빌드 스테이지에 진입하기 위해 조건에 제공된 소스 리포 지토리 이름 및 브랜치 이름과 비교하여 출력 변수를 확인합니다.

A Important

파이프라인 생성의 일환으로 CodePipeline은 고객이 제공하는 S3 아티팩트 버킷을 아티팩트 에 사용합니다. (이는 S3 소스 작업에 사용되는 버킷과 다릅니다.) S3 아티팩트 버킷이 파이프 라인의 계정과 다른 계정에 있는 경우 S3 아티팩트 버킷을 안전하고 신뢰할 수 AWS 계정 있는 에서 소유해야 합니다.

▲ Important

이 절차에서 파이프라인에 추가하는 많은 작업에는 파이프라인을 생성하기 전에 생성해야 하 는 AWS 리소스가 포함됩니다. 소스 작업에 대한 AWS 리소스는 항상 파이프라인을 생성하는 리전과 동일한 AWS 리전에서 생성해야 합니다. 예를 들어, 미국 동부(오하이오) 리전에서 파 이프라인을 생성하는 경우, CodeCommit 리포지토리도 미국 동부(오하이오) 리전에 있어야 합 니다.

파이프라인을 생성할 때 교차 리전 작업을 추가할 수 있습니다. 교차 리전 작업에 대한 AWS 리소스는 작업을 실행하려는 리전과 동일한 AWS 리전에 있어야 합니다. 자세한 내용은 CodePipeline에 교차 리전 작업 추가 단원을 참조하십시오.

이 예제에서는 GitHub(버전 2) 소스 작업과 CodeBuild 빌드 작업이 있는 예제 파이프라인을 사용합니다. 여기서 빌드 스테이지의 입력 조건에 변수가 있는지 확인합니다.

사전 조건

시작하기 전에 다음을 수행해야 합니다.

- GitHub 계정을 사용하여 GitHub 리포지토리를 만듭니다.
- GitHub 자격 증명을 준비합니다. AWS Management Console 를 사용하여 연결을 설정하면 GitHub 자격 증명으로 로그인하라는 메시지가 표시됩니다.
- GitHub(GitHub 앱을 통해)를 파이프라인의 소스 작업으로 설정하기 위한 리포지토리에 대한 연결입니다. GitHub 리포지토리에 대한 연결을 생성하려면 GitHub 연결 섹션을 참조하세요.

1단계: 샘플 소스 파일을 생성하고 GitHub 리포지토리에 추가

이 섹션에서는 파이프라인이 소스 스테이지에 대해 사용하는 리포지토리에 예제 소스 파일을 생성하 고 추가합니다. 이 예제에서는 다음을 생성하고 추가합니다. • README.md 파일.

GitHub 리포지토리를 만든 후 다음 단계를 사용하여 README 파일을 추가합니다.

- 1. GitHub 리포지토리에 로그인하고 리포지토리를 선택합니다.
- 새 파일을 생성하려면 파일 추가를 선택한 다음 새 파일 생성을 선택합니다. 파일 이름을 README.md로 지정하고 다음 텍스트를 추가합니다.

This is a GitHub repository!

 변경 사항 커밋을 선택합니다. 이 자습서의 목적상 다음 예제와 같이 대문자로 표시된 'Update'가 포함된 커밋 메시지를 추가합니다.

Update to source files

Note

문자열에 대한 규칙 확인은 대소문자를 구분합니다.

README.md 파일이 리포지토리의 루트 수준에 있는지 확인합니다.

2단계: 파이프라인 생성

이 단원에서는 다음 작업을 통해 파이프라인을 생성합니다.

- GitHub 리포지토리 및 작업에 대한 연결이 있는 소스 단계입니다.
- 변수 확인 규칙에 대해 입력 시 조건이 구성된 CodeBuild 빌드 스테이지입니다.

마법사를 사용하여 파이프라인을 생성하려면

- 1. https://console.aws.amazon.com/codepipeline/에서 CodePipeline 콘솔에 로그인합니다.
- 2. Welcome(시작) 페이지, 시작하기 페이지 또는 Pipelines(파이프라인) 페이지에서 파이프라인 생성을 선택합니다.
- 1단계: 생성 옵션 선택 페이지의 생성 옵션에서 사용자 지정 파이프라인 빌드 옵션을 선택합니다. Next(다음)를 선택합니다.

- 4. 2단계: 파이프라인 설정 선택의 파이프라인 이름에 MyVarCheckPipeline을 입력합니다.
- CodePipeline은 특성과 가격이 다른 V1 및 V2 유형 파이프라인을 제공합니다. V2 유형은 콘 솔에서 선택할 수 있는 유일한 유형입니다. 자세한 내용은 <u>파이프라인 유형</u>을 참조하세요. CodePipeline에 대한 요금 정보는 <u>요금</u>을 참조하세요.
- 6. Service role(서비스 역할)에서 New service role(새 서비스 역할)을 선택합니다.

Note

역할을 생성하지 않고 기존 CodePipeline 서비스 역할을 사용할 경우 서비스 역할 정책에 대한 codeconnections:UseConnection IAM 권한을 추가해야 합니다. CodePipeline 서비스 역할에 대한 지침은 CodePipeline 서비스 역할에 권한 추가를 참조하세요.

7. 고급 설정에서 기본값을 그대로 둡니다.

Next(다음)를 선택합니다.

- 8. 3단계: 소스 스테이지 추가 페이지에서 소스 스테이지를 추가합니다.
 - a. 소스 공급자에서 GitHub(GitHub 앱을 통해)를 선택합니다.
 - b. 연결에서 기존 연결을 선택하거나 새로 생성합니다. GitHub 소스 작업에 대한 연결을 만들거 나 관리하려면 GitHub 연결을 참조하세요.
 - c. 리포지토리 이름에서 GitHub 리포지토리의 이름을 선택합니다.
 - d. 브랜치 이름에서 사용할 리포지토리 브랜치를 선택합니다.
 - e. 트리거 없음 옵션이 선택되어 있는지 확인합니다.

Next(다음)를 선택합니다.

- 9. 4단계: 빌드 단계 추가에서 빌드 단계를 추가합니다.
 - a. 빌드 공급자에서 AWS CodeBuild를 선택합니다. 리전이 파이프라인 리전으로 기본 설정되도 록 합니다.
 - b. 프로젝트 생성을 선택합니다.
 - c. 프로젝트 이름에 이 빌드 프로젝트의 이름을 입력합니다.
 - d. 환경 이미지에서 이미지 관리를 선택합니다. [Operating system]에서 [Ubuntu]를 선택합니다.
 - e. 실행 시간에서 표준을 선택합니다. 이미지에서 aws/codebuild/standard:5.0을 선택합니다.
 - f. 서비스 역할에서 New service role(새 서비스 역할)을 선택합니다.

1 Note

CodeBuild 서비스 역할의 이름을 기록해 둡니다. 이 자습서의 마지막 단계를 수행하 려면 역할 이름이 필요합니다.

g. [Buildspec]의 [빌드 사양(Build specifications)]에서 [빌드 명령 삽입(Insert build commands)] 을 선택합니다. 편집기로 전환을 선택하고 빌드 명령에 다음을 붙여 넣습니다.

```
version: 0.2
#env:
 #variables:
     # kev: "value"
    # key: "value"
 #parameter-store:
    # key: "value"
    # key: "value"
 #git-credential-helper: yes
phases:
 install:
    #If you use the Ubuntu standard image 2.0 or later, you must specify
runtime-versions.
    #If you specify runtime-versions and use an image other than Ubuntu
standard image 2.0, the build fails.
    runtime-versions:
     nodejs: 12
   #commands:
      # - command
     # - command
 #pre_build:
    #commands:
      # - command
     # - command
 build:
    commands:
 #post_build:
    #commands:
      # - command
     # - command
artifacts:
 files:
     _ '*'
```

```
# - location
name: $(date +%Y-%m-%d)
#discard-paths: yes
#base-directory: location
#cache:
#paths:
# - paths
```

- h. Continue to CodePipeline(CodePipeline으로 계속)을 선택합니다. 그러면 CodePipeline 콘솔 로 돌아가고 구성을 위해 빌드 명령을 사용하는 CodeBuild 프로젝트가 생성됩니다. 빌드 프로 젝트는 서비스 역할을 사용하여 AWS 서비스 권한을 관리합니다. 이 단계는 몇 분이 걸릴 수 있습니다.
- i. Next(다음)를 선택합니다.
- 5단계: 테스트 단계 추가에서 테스트 단계 건너뛰기를 선택한 다음 건너뛰기를 다시 선택하여 경고 메시지를 수락합니다.

Next(다음)를 선택합니다.

- 11. 6단계: 배포 단계 추가 페이지에서 배포 단계 건너뛰기를 선택한 다음 다시 건너뛰기를 선택하여 경고 메시지를 수락합니다. Next(다음)를 선택합니다.
- 12. 7단계: 검토에서 파이프라인 생성을 선택합니다.

2단계: 빌드 스테이지를 편집하여 조건 및 규칙 추가

이 단계에서는 스테이지를 편집하여 변수 확인 규칙에 대한 입력 시 조건을 추가합니다.

 파이프라인을 선택한 다음 편집을 선택합니다. 빌드 스테이지에 입력 규칙을 추가하려면 선택합 니다.

규칙 공급자에서 VariableCheck를 선택합니다.

변수에 확인할 변수를 입력합니다. 값에 문자열 값을 입력하여 확인된 변수를 확인합니다. 다음 예
 제 화면에서는 '같음' 검사에 대해 규칙이 생성되고 '포함' 검사에 대해 다른 규칙이 생성됩니다.

Edit rule

Rule name

Choose a name for your rule

varrule

No more than 100 characters

Rule provider

AWS VariableCheck

Variable

The variable with the resolved value that will be checked against the provided string value.

#{SourceVariables.FullRepositoryName}

Variables must use the following syntax: #{namespace.variable_key}.

Value

The string value to check against the resolved variable value.

/MyGitHubRepo

Operator

Operator for the comparison.

 Equals Checks whether the variable is equal to the string value. Not equals
 Checks whether the variable is not equal to the string value. Contains

Checks whether the variable contains the string value as a substring.

Matches

Checks whether the variable matches a given regex expression as the string value.

v

Choose a name for your rule		
varcheckmsg		
No more than 100 characters		
Rule provider		
AWS VariableCheck		
#{SourceVariables.CommitN	lessage}	acti string value.
Variables must use the following s	yntax: #{namespace.variable_key}.	
Value		
Value The string value to check against t	the resolved variable value.	
Value The string value to check against t Update	he resolved variable value.	
Value The string value to check against t Update Operator	he resolved variable value.	
Value The string value to check against t Update Operator Operator for the comparison.	the resolved variable value.	
Value The string value to check against t Update Operator Operator for the comparison.	the resolved variable value.	• Contains
Value The string value to check against t Update Operator Operator for the comparison. C Equals Checks whether the	the resolved variable value.	• Contains Checks whether the
Value The string value to check against t Update Operator Operator for the comparison. Equals Checks whether the variable is equal to the string value	he resolved variable value.	• Contains Checks whether the variable contains the string value as a
Value The string value to check against t Update Operator Operator for the comparison. Equals Checks whether the variable is equal to the string value.	he resolved variable value.	• Contains Checks whether the variable contains the string value as a substring.
Value The string value to check against t Update Operator Operator for the comparison. Equals Checks whether the variable is equal to the string value.	he resolved variable value.	• Contains Checks whether the variable contains the string value as a substring.
Value The string value to check against t Update Operator Operator for the comparison. Equals Checks whether the variable is equal to the string value. Matches	the resolved variable value. Not equals Checks whether the variable is not equal to the string value.	• Contains Checks whether the variable contains the string value as a substring.
Value The string value to check against t Update Operator Operator for the comparison. Equals Checks whether the variable is equal to the string value. Matches Checks whether the	he resolved variable value. Not equals Checks whether the variable is not equal to the string value.	• Contains Checks whether the variable contains the string value as a substring.
Value The string value to check against t Update Operator Operator for the comparison. Equals Checks whether the variable is equal to the string value. Matches Checks whether the variable matches a given	the resolved variable value. Not equals Checks whether the variable is not equal to the string value.	• Contains Checks whether the variable contains the string value as a substring.

3. 저장(Save)을 선택합니다.

완료를 선택합니다.

3단계: 파이프라인 실행 및 확인된 변수 보기

이 단계에서는 변수 확인 규칙의 확인된 값과 결과를 확인합니다.

1. 다음 예제와 같이 규칙 확인에 성공한 후 확인된 실행을 확인합니다.

Source Succeeded
Pipeline execution ID: <u>1438349d-9809-4249-9621-a</u>
Source
Surreaded - 21 minutes and
Successory 2 minutes ago
View details
77cc2e44 🗹 Source: Merge pull request #5 from //feature-branch 🚥
Disable transition
\downarrow
Entry condition: O Succeeded Execution ID: 1438349d
Succeeded
Pipeline execution ID: <u>1438349d-9809-4249-9621-</u>
Build
AWS CodeBuild
Succeeded - <u>18 minutes ago</u>
view details

2. 타임라인 탭에서 변수 정보를 봅니다.

Visua	lization	Timeline	Variables Revisio	ns				
Actions							View execution details	
	Action name	Stage name	Status	Action provider	Started	Completed	Duration	
0	Source	Source	⊘ Succeeded	GitHub (Version 2) 🖪	4 minutes ago	4 minutes ago	4 seconds	
0	Build	Build	⊘ Succeeded	AWS CodeBuild	4 minutes ago	Just now	3 minutes 31 seconds	

Rules					
Name	Stage Condition	Status	Started	Duration	Reason
varcheckmsg AWS VariableCheck	Build Entry	Succeeded	4 minutes ago	less than one second	-
varrule AWS VariableCheck	Build Entry	Succeeded	4 minutes ago	less than one second	-

CodePipeline 사용 사례

다음 섹션에서는 CodePipeline의 사용 사례를 설명합니다.

주제

• CodePipeline의 사용 사례

CodePipeline의 사용 사례

다른와 통합되는 파이프라인을 생성할 수 있습니다 AWS 서비스. 이러한 서비스는 Amazon S3와 같 은 AWS 서비스또는 GitHub와 같은 타사 제품이 될 수 있습니다. 이 단원에서는 CodePipeline을 통 해 여러 제품 통합을 사용하는 코드 릴리스를 자동화하는 예제를 설명합니다. 작업 유형별로 분류된 CodePipeline과의 통합 전체 목록을 보려면 <u>CodePipeline 파이프라인 구조 참조</u> 단원을 참조하세요.

주제

- <u>Amazon S3 AWS CodeCommit, 및에서 CodePipeline 사용 AWS CodeDeploy</u>
- <u>CodePipeline과 타사 작업 공급자 사용(GitHub 및 Jenkins)</u>
- CodeBuild에서 CodePipeline을 사용하여 코드 컴파일, 빌드, 테스트
- <u>CodePipeline과 Amazon ECS를 함께 사용하여 컨테이너 기반 애플리케이션을 클라우드에 지속적</u>으로 제공
- CodePipeline과 Elastic Beanstalk를 함께 사용하여 웹 애플리케이션을 클라우드에 지속적으로 제공
- <u>Lambda 기반 및 서버리스 애플리케이션을 지속적으로 제공 AWS Lambda 하려면와 함께</u> CodePipeline 사용
- CodePipeline을 AWS CloudFormation 템플릿과 함께 사용하여 클라우드에 지속적으로 제공

Amazon S3 AWS CodeCommit, 및에서 CodePipeline 사용 AWS CodeDeploy

파이프라인을 생성하면 CodePipeline은 파이프라인의 각 단계에서 작업 공급자 역할을 하는 AWS 제 품 및 서비스와 통합됩니다. 마법사에서 단계를 선택할 때 소스 단계와 빌드 또는 배포 단계(최소 한 개)를 선택해야 합니다. 마법사가 기본 이름으로 단계를 생성합니다. 이 이름은 변경할 수 없습니다. 이 이름은 마법사에서 3단계 파이프라인 전체를 설정할 때 생성되는 단계 이름입니다.

• 기본 이름인 "Source"로 이름이 지정되는 소스 작업 단계

- 기본 이름인 "Build"로 이름이 지정되는 빌드 작업 단계
- 기본 이름인 "Staging"으로 이름이 지정되는 배포 작업 단계

이 설명서의 자습서를 사용하여 파이프라인을 생성하고 단계를 지정할 수 있습니다.

- <u>자습서: 간단한 파이프라인 생성(S3 버킷)</u>의 단계는 마법사를 사용하여 두 개의 기본 단계인 "Source"와 "Staging"을 포함하는 파이프라인을 생성합니다. 소스 공급자는 Amazon S3 리포지토리 입니다. 이 자습서에서는를 AWS CodeDeploy 사용하여 Amazon S3 버킷에서 Amazon Linux를 실 행하는 Amazon EC2 인스턴스로 샘플 애플리케이션을 배포하는 파이프라인을 생성합니다.
- 의 단계는 마법사를 사용하여 AWS CodeCommit 리포지토리를 소스 공급자로 사용하는 "소스" 단 계가 있는 파이프라인을 생성하는 데 <u>자습서: 간단한 파이프라인 생성(CodeCommit 리포지토리)</u> 도 움이 됩니다. 이 자습서에서는를 AWS CodeDeploy 사용하여 리포지 AWS CodeCommit 토리에서 Amazon Linux를 실행하는 Amazon EC2 인스턴스로 샘플 애플리케이션을 배포하는 파이프라인을 생성합니다.

CodePipeline과 타사 작업 공급자 사용(GitHub 및 Jenkins)

GitHub 및 Jenkins 등과 같은 타사 제품과 통합되는 파이프라인을 생성할 수 있습니다. <u>자습서: 4단계</u> 파이프라인 생성의 단계는 다음과 같은 파이프라인을 생성하는 방법을 보여 줍니다.

- GitHub 리포지토리에서 소스를 받음
- Jenkins를 사용하여 소스 코드를 빌드하고 테스트함
- AWS CodeDeploy 를 사용하여 빌드되고 테스트된 소스 코드를 Amazon Linux 또는 Microsoft Windows Server를 실행하는 Amazon EC2 인스턴스에 배포합니다.

CodeBuild에서 CodePipeline을 사용하여 코드 컴파일, 빌드, 테스트

CodeBuild는 서버나 시스템 없이 코드를 빌드하고 테스트할 수 있는 클라우드의 관리형 빌드 서비 스입니다. CodePipeline과 CodeBuild를 함께 사용하면 파이프라인을 통해 개정 실행을 자동화함으 로써 소스 코드가 변경될 때마다 소프트웨어 빌드를 지속적으로 제공할 수 있습니다. 자세한 내용은 CodePipeline과 CodeBuild를 함께 사용하여 코드 테스트 및 빌드 실행을 참조하세요.

CodePipeline과 Amazon ECS를 함께 사용하여 컨테이너 기반 애플리케이션 을 클라우드에 지속적으로 제공

Amazon ECS는 컨테이너 기반 애플리케이션을 클라우드의 Amazon ECS 인스턴스에 배포할 수 있는 컨테이너 관리 서비스입니다. Amazon ECS와 CodePipeline을 함께 사용하면 파이프라인을 통해 개정 실행을 자동화함으로써 소스 이미지 리포지토리가 변경될 때마다 컨테이너 기반 애플리케이션을 계속 배포할 수 있습니다. 자세한 내용은 <u>자습서: CodePipeline을 사용하여 지속적으로 배포</u>를 참조하세요.

CodePipeline과 Elastic Beanstalk를 함께 사용하여 웹 애플리케이션을 클라 우드에 지속적으로 제공

Elastic Beanstalk는 웹 애플리케이션 및 서비스를 웹 서버에 배포할 수 있는 컴퓨팅 서비스입니다. CodePipeline과 Elastic Beanstalk를 함께 사용하여 웹 애플리케이션을 애플리케이션 환경에 지속적으 로 배포하세요. AWS CodeStar 를 사용하여 Elastic Beanstalk 배포 작업이 있는 파이프라인을 생성할 수도 있습니다.

Lambda 기반 및 서버리스 애플리케이션을 지속적으로 제공 AWS Lambda 하려면와 함께 CodePipeline 사용

<u>서버리스 애플리케이션 배포</u>에 설명된 대로 CodePipeline과 AWS Lambda 함께를 사용하여 AWS Lambda 함수를 호출할 수 있습니다. AWS Lambda 및 AWS CodeStar 를 사용하여 서버리스 애플리케 이션을 배포하기 위한 파이프라인을 생성할 수도 있습니다.

CodePipeline을 AWS CloudFormation 템플릿과 함께 사용하여 클라우드에 지속적으로 제공

지속적인 전송 및 자동화를 위해 CodePipeline과 AWS CloudFormation 함께를 사용할 수 있습니다. 자 세한 내용은 <u>CodePipeline을 사용한 지속적 전달</u>을 참조하세요. AWS CloudFormation 또한에서 생성 된 파이프라인에 대한 템플릿을 생성하는 데 사용됩니다 AWS CodeStar.

Amazon Virtual Private Cloud와 함께 CodePipeline 사용

AWS CodePipeline 는 이제 로 구동되는 <u>Amazon Virtual Private Cloud(VPC)</u> 엔드포인트를 지원합니 다<u>AWS PrivateLink</u>. 즉, VPC의 프라이빗 엔드포인트를 통해 CodePipeline에 직접 연결하여 VPC 및 AWS 네트워크 내의 모든 트래픽을 유지할 수 있습니다.

Amazon VPC는 정의 AWS 서비스 한 가상 네트워크에서 AWS 리소스를 시작하는 데 사용할 수 있는 입니다. VPC를 사용하면 다음과 같은 네트워크 설정을 제어할 수 있습니다.

- IP 주소 범위
- 서브넷
- 라우팅 테이블
- 네트워크 게이트웨이

인터페이스 VPC 엔드포인트는 프라이빗 IP 주소와 함께 탄력적 네트워크 인터페이스를 AWS 서비 스 사용하는 간의 프라이빗 통신을 용이하게 하는 AWS PrivateLink AWS 기술로 구동됩니다. VPC를 CodePipeline에 연결하려면 CodePipeline에 대해 인터페이스 VPC 엔드포인트를 정의하세요. 이 유형 의 엔드포인트를 사용하여 VPC를 AWS 서비스에 연결할 수 있습니다. 이 엔드포인트를 이용하면 인터 넷 게이트웨이나 NAT(네트워크 주소 변환) 인스턴스 또는 VPN 연결 없이도 CodePipeline에 안정적이 고 확장 가능하게 연결됩니다. VPC 설정에 대한 자세한 내용은 <u>VPC 사용 설명서</u>를 참조하십시오.

가용성

CodePipeline은 현재 AWS 리전다음과 같은 VPC 엔드포인트를 지원합니다.

- 미국 동부(오하이오)
- 미국 동부(버지니아 북부)
- 미국 서부(캘리포니아 북부)
- 미국 서부(오레곤)
- 캐나다(중부)
- 유럽(프랑크푸르트)
- 유럽(아일랜드)
- 유럽(런던)
- 유럽(밀라노)*

- 유럽(파리)
- 유럽(스톡홀름)
- 아시아 태평양(홍콩)*
- 아시아 태평양(뭄바이)
- 아시아 태평양(도쿄)
- 아시아 태평양(서울)
- 아시아 태평양(싱가포르)
- 아시아 태평양(시드니)
- 남아메리카(상파울루)
- AWS GovCloud(미국 서부)

* 이 리전을 사용하려면 먼저 활성화해야 합니다.

CodePipeline의 VPC 엔드포인트 생성

Amazon VPC 콘솔을 사용하여 com.amazonaws.*region*.codepipeline VPC 엔드포인트를 생성할 수 있습니다. 콘솔에서 *region*은 미국 동부(오하이오) 리전과 같이 CodePipeline에서 AWS 리전 지원하 는 us-east-2의 리전 식별자입니다. 자세한 내용은 Amazon VPC 사용 설명서의 <u>인터페이스 엔드포</u> 인트 생성을 참조하세요.

엔드포인트는 AWS에 로그인할 때 지정한 리전으로 사전 입력됩니다. 다른 리전에 로그인하면 VPC 엔 드포인트가 새 리전으로 업데이트됩니다.

Note

CodeCommit과 같이 VPC 지원을 제공하고 CodePipeline과 통합 AWS 서비스 하는 다른 는 해당 통합에 Amazon VPC 엔드포인트 사용을 지원하지 않을 수 있습니다. 예를 들어 CodePipeline과 CodeCommit 간의 트래픽은 VPC 서브넷 범위로 제한할 수 없습니다.

VPC 설정 문제 해결

VPC 문제를 해결할 때 인터넷 연결 오류 메시지에 표시되는 정보를 사용하면 해당 문제를 식별, 진단 및 해결하는 데 도움이 됩니다.

- 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. 프라이빗 서브넷의 라우팅 테이블이 가상 프라이빗 게이트웨이를 가리키는지 확인합니다.
- 6. CodePipeline에서 사용하는 서비스 역할에 적절한 권한이 있는지 확인합니다. 예를 들어 CodePipeline에 Amazon VPC에서 작업을 수행하는 데 필요한 Amazon EC2 권한이 없으면 "예상 치 못한 EC2 오류: UnauthorizedOperation"이라는 오류 메시지가 나타날 수 있습니다.

스테이지 및 작업을 사용하여 CI/CD 파이프라인 정의

에서 자동 릴리스 프로세스를 정의하려면 파이프라인을 AWS CodePipeline생성합니다. 파이프라인은 소프트웨어 변경 사항이 릴리스 프로세스를 거치는 방식을 설명하는 워크플로 구조입니다. 파이프라 인은 구성한 단계와 작업으로 구성됩니다.

Note

CodePipeline과 함께 제공된 기본 옵션 외에도 빌드, 배포, 테스트 또는 호출 단계를 추가하면 파이프라인과 함께 사용하기 위해 이미 생성한 사용자 지정 작업을 선택할 수 있습니다. 사용 자 지정 작업은 내부적으로 개발된 빌드 프로세스 또는 테스트 제품군을 실행하는 등의 작업에 사용할 수 있습니다. 공급자 목록에 있는 여러 버전의 사용자 지정 작업 중에서 구별하는 데 도 움이 되도록 버전 식별자가 포함되어 있습니다. 자세한 내용은 <u>CodePipeline에서 사용자 지정</u> 작업 생성 및 추가 단원을 참조하십시오.

파이프라인을 생성하기 전에 먼저 CodePipeline 시작하기의 단계를 완료해야 합니다.

파이프라인에 대한 자세한 내용은 <u>CodePipeline 개념, CodePipeline 자습서</u>및를 사용하여 파이프라인 을 생성하려는 경우 섹션을 참조 AWS CLI 하세요<u>파이프라인 스테이지 및 작업 생성</u>. 파이프라인의 목 록을 보려면 <u>CodePipeline에서 파이프라인 및 세부 정보 보기</u> 단원을 참조하십시오.

주제

- 파이프라인 스테이지 및 작업 생성
- CodePipeline에서 파이프라인 편집
- CodePipeline에서 파이프라인 및 세부 정보 보기
- CodePipeline에서 파이프라인 삭제
- CodePipeline에서 다른 AWS 계정의 리소스를 사용하는 파이프라인 생성
- 이벤트 기반 변경 감지를 사용하도록 폴링 파이프라인 마이그레이션
- <u>CodePipeline 서비스 역할 생성</u>
- 리소스에 태그 지정
- CodePipeline에서 파이프라인 태그 지정
- <u>알림 규칙 생</u>성
파이프라인 스테이지 및 작업 생성

AWS CodePipeline 콘솔 또는를 사용하여 파이프라인 AWS CLI 을 생성할 수 있습니다. 파이프라인에 는 두 개 이상의 단계가 포함되어야 합니다. 파이프라인의 첫 번째 단계는 소스 단계여야 합니다. 파이 프라인에는 빌드 또는 배포 단계인 하나 이상의 다른 단계가 있어야 합니다.

🛕 Important

파이프라인 생성의 일환으로 CodePipeline은 고객이 제공하는 S3 아티팩트 버킷을 아티팩트 에 사용합니다. (이는 S3 소스 작업에 사용되는 버킷과 다릅니다.) S3 아티팩트 버킷이 파이프 라인의 계정과 다른 계정에 있는 경우 S3 아티팩트 버킷을 안전하고 신뢰할 수 AWS 계정 있는 가 소유해야 합니다.

파이프라인과 AWS 리전 다른에 있는 작업을 파이프라인에 추가할 수 있습니다. 교차 리전 작업은가 작업의 공급자 AWS 서비스 이고 작업 유형 또는 공급자 유형이 파이프라인과 다른 AWS 리전에 있는 작업입니다. 자세한 내용은 <u>CodePipeline에 교차 리전 작업 추가</u> 단원을 참조하십시오.

Amazon ECS를 배포 공급자로 사용하여 컨테이너 기반 애플리케이션을 빌드하고 배포하는 파이프라 인을 만들 수도 있습니다. Amazon ECS를 사용하여 컨테이너 기반 애플리케이션을 배포하는 파이프 라인을 만들기 전에 <u>이미지 정의 파일 참조</u> 단원의 설명에 따라 이미지 정의 파일을 생성해야 합니다.

CodePipeline은 소스 코드 변경이 푸시될 때 변경 감지 방법을 사용하여 파이프라인을 시작합니다. 이 러한 감지 방법은 소스 유형을 기반으로 합니다.

• CodePipeline은 Amazon CloudWatch Events를 사용하여 CodeCommit 소스 리포지토리와 브랜치 또는 S3 소스 버킷의 변경을 감지합니다.

Note

콘솔을 사용하여 파이프라인을 생성하거나 편집하면 변경 감지 리소스가 자동으로 생성됩니 다. AWS CLI를 사용하여 파이프라인을 생성하는 경우 추가 리소스를 직접 생성해야 합니다. 자세한 내용은 <u>CodeCommit 소스 작업 및 EventBridge</u> 단원을 참조하십시오.

주제

- 사용자 지정 파이프라인 생성(콘솔)
- <u>파이프라인 생성(CLI)</u>

• 정적 템플릿에서 파이프라인 생성

사용자 지정 파이프라인 생성(콘솔)

콘솔에서 사용자 지정 파이프라인을 생성하려면 작업에 사용하는 공급자에 대한 정보와 소스 파일 위 치를 제공해야 합니다.

콘솔을 사용하여 파이프라인을 생성하면 소스 단계와 다음 중 하나 또는 둘 다를 포함해야 합니다.

- 빌드 단계.
- 배포 단계.

파이프라인 마법사를 사용할 때 CodePipeline은 단계의 이름을 생성합니다(소스, 빌드, 스테이 징). 이러한 이름은 변경할 수 없습니다. 나중에 추가한 단계에 특정 이름(예: BuildToGamma 또는 DeployToProd)을 더 사용할 수 있습니다.

1단계: 파이프라인 생성 및 이름 지정

- 1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https//https//https://https//ht
- 2. [Welcome] 페이지에서 [Create pipeline]을 선택합니다.

CodePipeline을 처음 사용하는 경우 시작하기를 선택합니다.

- 1단계: 생성 옵션 선택 페이지의 생성 옵션에서 사용자 지정 파이프라인 빌드 옵션을 선택합니다. Next(다음)를 선택합니다.
- 4. 2단계: 파이프라인 설정 선택 페이지의 파이프라인 이름에 파이프라인 이름을 입력합니다.

단일 AWS 계정에서 AWS 리전에서 생성하는 각 파이프라인에는 고유한 이름이 있어야 합니다. 다른 리전의 파이프라인에 이름을 재사용해도 됩니다.

Note

파이프라인을 만든 후에는 해당 이름을 변경할 수 없습니다. 다른 제한에 대한 자세한 내 용은 <u>in AWS CodePipeline 할당량</u> 섹션을 참조하십시오.

 파이프라인 유형에서 다음 옵션 중 하나를 선택합니다. 파이프라인 유형은 특성과 가격이 다릅니 다. 자세한 내용은 파이프라인 유형 단원을 참조하십시오.

- V1 유형 파이프라인은 표준 파이프라인, 단계, 작업 수준 파라미터를 포함하는 JSON 구조를 가 지고 있습니다.
- V2 유형 파이프라인은 Git 태그의 트리거 및 파이프라인 수준 변수와 같은 추가 파라미터 지원 과 함께 V1 유형과 구조가 동일합니다.
- 6. 서비스 역할에서 다음 중 하나를 수행합니다.
 - 새 서비스 역할을 선택하여 CodePipeline이 IAM에 새 서비스 역할을 생성하도록 허용합니다.
 - Existing service role(기존 서비스 역할)을 선택하여 IAM에서 이미 생성된 서비스 역할을 사용합니다. 역할 ARN의 목록에서 서비스 역할 ARN을 선택합니다.

Note

서비스 역할이 생성된 시기에 따라 추가 지원을 위해 권한을 업데이트해야 할 수 있습니다 AWS 서비스. 자세한 내용은 CodePipeline 서비스 역할에 권한 추가을 참조하세요.

서비스 역할 및 해당 정책 설명에 대한 자세한 내용은 <u>CodePipeline 서비스 역할 관리</u> 섹션을 참조 하십시오.

7. (선택 사항) 변수에서 변수 추가를 선택하여 파이프라인 수준에서 변수를 추가합니다.

파이프라인 수준에서 변수에 대한 자세한 정보는 <u>변수 참조</u>을 참조하세요. 파이프라인 실행 시 전 달되는 파이프라인 수준 변수에 대한 자습서는 자습서: 파이프라인 수준 변수 사용을 참조하세요.

Note

파이프라인 수준에서 변수를 추가하는 것은 선택 사항이지만, 값이 제공되지 않은 파이프 라인 수준의 변수로 지정된 파이프라인의 경우 파이프라인 실행이 실패합니다.

- 8. (선택 사항) 고급 설정을 확장합니다.
- 9. 아티팩트 스토어에서 다음 중 하나를 수행합니다.
 - a. 파이프라인에 대해 AWS 리전 선택한의 파이프라인에 대해 기본값으로 지정된 S3 아티팩트 버킷과 같은 기본 아티팩트 스토어를 사용하려면 기본 위치를 선택합니다.
 - b. 파이프라인과 동일한 리전에 S3 아티팩트 버킷과 같이 이미 아티팩트 스토어가 있는 경우 Custom location(사용자 지정 위치)을 선택합니다. [Bucket]에서 버킷 이름을 선택합니다.

Note

이는 소스 코드에 대한 소스 버킷이 아닙니다. 이 파이프라인은 아티팩트 스토어입니다. S3 버킷과 같은 개별 아티팩트 스토어는 각 파이프라인에 필요합니다. 파이프라인을 생성 하거나 편집할 때 파이프라인 리전에 아티팩트 버킷이 하나 있고 작업을 실행 중인 AWS 리전당 아티팩트 버킷이 하나 있어야 합니다. 자세한 내용은 <u>입력 및 출력 아티팩트</u> 및 <u>CodePipeline 파이프라인 구조 참조</u> 단원을 참조 하세요.

- 10. [Encryption key]에서 다음 중 하나를 수행합니다.
 - a. CodePipeline 기본값을 사용하여 파이프라인 아티팩트 스토어(S3 버킷)의 데이터를 AWS KMS key 암호화하려면 기본 AWS 관리형 키를 선택합니다.
 - b. 고객 관리형 키를 사용하여 파이프라인 아티팩트 스토어(S3 버킷)의 데이터를 암호화하려면 고객 관리형 키를 선택합니다. 키 ID, 키 ARN 또는 별칭 ARN을 선택합니다.
- 11. Next(다음)를 선택합니다.

2단계: 소스 단계 생성

- 단계 3: 소스 스테이지 추가 페이지의 소스 공급자에서 소스 코드가 저장된 리포지토리의 유형을 선택하고 필수 옵션을 지정합니다. 다음과 같이 선택한 소스 공급자에 따라 추가 필드가 표시됩니 다.
 - Bitbucket Cloud, GitHub(GitHub App을 통해), GitHub Enterprise Server, GitLab.com, 또는 GitLab 자체 관리형의 경우:
 - 1. 연결에서 기존 연결을 선택하거나 새로 생성합니다. GitHub 소스 작업에 대한 연결을 만들거 나 관리하려면 GitHub 연결을 참조하세요.
 - 2. 파이프라인의 소스 위치로 사용할 리포지토리를 선택합니다.

트리거를 추가하거나 트리거 유형을 필터링하도록 선택하여 파이프라인을 시작합니다. 트리 거 작업에 대한 자세한 내용은 <u>코드 푸시 또는 풀 요청 이벤트 유형을 사용하여 트리거 추가</u> 섹션을 참조하세요. Glog 패턴을 사용한 필터링에 대한 자세한 내용은 <u>구문에서 glob 패턴 작</u> 업을 참조하세요.

3. 출력 아티팩트 형식에서 아티팩트의 형식을 선택합니다.

- 기본 방법을 사용하여 GitHub 작업의 출력 아티팩트를 저장하려면 CodePipeline 기본 방법을 선택합니다. 그러면 GitHub 리포지토리의 파일에 액세스하여 파이프라인 아티팩트 스토어에 ZIP 파일로 아티팩트가 저장됩니다.
- 다운스트림 작업이 Git 명령을 직접 수행할 수 있도록 리포지토리에 대한 URL 참조가 포함 된 JSON 파일을 저장하려면 Full clone(전체 복제)을 선택합니다. 이 옵션은 CodeBuild 다 운스트림 작업에서만 사용할 수 있습니다.

이 옵션을 선택하면 <u>CodePipeline 문제 해결</u>과 같이 CodeBuild 프로젝트 서비스 역할에 대 한 권한을 업데이트해야 합니다. 전체 복제 옵션을 사용하는 방법을 보여주는 자습서는 <u>자</u> 습서: GitHub 파이프라인 소스와 함께 전체 복제 사용을 참조하세요.

- Amazon S3의 경우:
 - 1. Amazon S3 위치에서 버전 관리 기능이 활성화된 버킷에 S3 버킷 이름 및 객체 경로를 제공 합니다. 버킷 이름과 경로의 형식은 다음과 같습니다.

s3://bucketName/folderName/objectName

Note

Amazon S3가 파이프라인의 소스 공급자인 경우, 소스 파일을 .zip 하나로 압축하고 그 .zip을 소스 버킷에 업로드할 수 있습니다. 압축이 풀린 단일 파일을 업로드할 수도 있지만 .zip 파일을 예상하는 다운스트림 작업은 실패합니다.

- 2. S3 소스 버킷을 선택하면 CodePipeline은이 파이프라인에 대해 생성할 Amazon CloudWatch Events 규칙과 AWS CloudTrail 추적을 생성합니다. [Change detection options] 아래에서 기 본값을 적용합니다. 그러면 CodePipeline이 Amazon CloudWatch Events 및 AWS CloudTrail 를 사용하여 새 파이프라인의 변경을 감지할 수 있습니다. Next(다음)를 선택합니다.
- AWS CodeCommit의 경우:
 - 리포지토리 이름에서 파이프라인의 소스 위치로 사용할 CodeCommit 리포지토리의 이름을 선택합니다. [Branch name]의 드롭다운 목록에서 사용하려는 브랜치를 선택합니다.
 - 출력 아티팩트 형식에서 아티팩트의 형식을 선택합니다.
 - 기본 방법을 사용하여 CodeCommit 작업의 출력 아티팩트를 저장하려면 CodePipeline 기 본 방법을 선택합니다. 그러면 CodeCommit 리포지토리의 파일에 액세스하여 파이프라인 아티팩트 스토어에 ZIP 파일로 아티팩트가 저장됩니다.

• 다운스트림 작업이 Git 명령을 직접 수행할 수 있도록 리포지토리에 대한 URL 참조가 포함 된 JSON 파일을 저장하려면 Full clone(전체 복제)을 선택합니다. 이 옵션은 CodeBuild 다 운스트림 작업에서만 사용할 수 있습니다.

이 옵션을 선택하면 <u>CodeCommit 소스 작업에 대한 CodeBuild GitClone 권한 추가</u>과 같 이 CodeBuild 서비스 역할에 대한 codecommit:GitPull 권한을 추가해야 합니다. 또 한 <u>CodePipeline 서비스 역할에 권한 추가</u>에 표시된 대로 CodePipeline 서비스 역할에 codecommit:GetRepository 권한을 추가해야 합니다. 전체 복제 옵션을 사용하는 방법 을 보여주는 자습서는 <u>자습서: GitHub 파이프라인 소스와 함께 전체 복제 사용</u>을 참조하세 요.

- CodeCommit 리포지토리 이름과 브랜치를 선택하면 변경 감지 옵션에 이 파이프라인에 생성 될 Amazon CloudWatch Events 규칙을 보여주는 메시지가 표시됩니다. [Change detection options] 아래에서 기본값을 적용합니다. 그러면 CodePipeline이 Amazon CloudWatch Events를 사용하여 새 파이프라인의 변경을 감지할 수 있습니다.
- Amazon ECR의 경우:
 - 리포지토리 이름에서 Amazon ECR 리포지토리의 이름을 선택합니다.
 - 이미지 이름과 버전이 LATEST와 다른 경우 이미지 태그에서 지정합니다.
 - 출력 아티팩트에서 출력 아티팩트 기본값(예: MyApp)을 선택합니다. 이 아티팩트는 다음 단 계에서 사용하려는 이미지 이름과 리포지토리 URI를 포함합니다.

Amazon ECR 소스 단계가 포함된 CodeDeploy 블루-그린 배포를 사용하여 Amazon ECS 용 파이프라인을 생성하는 방법에 대한 자습서는 <u>자습서: Amazon ECR 소스 및 ECS-to-</u> CodeDeploy 배포를 사용하여 파이프라인 생성을 참조하세요.

파이프라인에 Amazon ECR 소스 단계를 포함하면 소스 작업이 변경 사항을 커밋할 때 imageDetail.json 파일을 출력 아티팩트로 생성합니다.imageDetail.json 파일에 대한 자세한 내용은 <u>Amazon ECS 블루/그린 배포 작업을 위한 imageDetail.json 파일</u> 단원을 참조하 십시오.

Note

객체 및 파일 형식이 사용하려는 배포 시스템과 호환되어야 합니다(예: Elastic Beanstalk 또는 CodeDeploy). 지원되는 파일 형식에는 .zip, .tar, .tgz 파일 등이 있습니다. Elastic Beanstalk에 지원되는 컨테이너 형식에 대한 자세한 내용은 <u>Elastic Beanstalk 환경 사용</u> <u>자 지정 및 구성</u> 및 <u>지원 플랫폼</u> 단원을 참조하세요. CodeDeploy를 사용하여 개정을 배포 하는 방법에 대한 자세한 내용은 <u>애플리케이션 개정 업로드</u> 및 <u>개정 준비</u> 단원을 참조하세 요.

- 자동 재시도를 위해 스테이지를 구성하려면 스테이지 장애 시 자동 재시도 활성화를 선택합니다.
 자동 재시도에 대한 자세한 내용은 실패 시 자동 재시도를 위한 스테이지 구성 섹션을 참조하세요.
- 3. Next(다음)를 선택합니다.

4단계: 빌드 스테이지 생성

배포 단계를 생성하려는 경우 이 단계는 선택 사항입니다.

- 1. 4단계: 빌드 단계 추가 페이지에서 다음 중 하나를 수행한 후 다음을 선택합니다.
 - 테스트 또는 배포 단계를 생성하려는 경우 빌드 단계 건너뛰기를 선택합니다.
 - 빌드 스테이지에 대한 명령 작업을 선택하려면 명령을 선택합니다.

Note

명령 작업을 실행하면 AWS CodeBuild에서 별도의 요금이 발생합니다. CodeBuild 작업 의 일부로 빌드 명령을 삽입하려는 경우 계속 진행하여 기타 빌드 공급자를 선택한 다음 CodeBuild를 선택합니다.

명령에 작업에 대한 쉘 명령을 입력합니다. 명령 작업에 대한 자세한 내용은 <u>명령 작업 참조</u> 섹 션을 참조하세요.

- CodeBuild와 같은 다른 빌드 공급자를 선택하려면 다른 공급자를 선택합니다. 빌드 공급자에서 빌드 서비스의 사용자 지정 작업 공급자를 선택하고 그 공급자에 대한 구성 세부 정보를 입력합 니다. Jenkins를 빌드 공급자로 추가하는 방법의 예는 <u>자습서: 4단계 파이프라인 생성</u> 섹션을 참 조하십시오.
- 빌드 공급자에서 AWS CodeBuild를 선택합니다.

리전에서 리소스가 있는 AWS 리전을 선택합니다. 리전 필드는이 작업 유형 및 공급자 유형에 대해 AWS 리소스가 생성되는 위치를 지정합니다. 이 필드는 작업 공급자가 AWS 서비스인 작 업에 대해서만 표시합니다. 리전 필드는 기본적으로 파이프라인과 동일한 AWS 리전으로 설정 됩니다.

프로젝트 이름에 빌드 프로젝트를 선택합니다. CodeBuild에서 이미 빌드 프로젝트를 만들었다 면 이 프로젝트를 선택합니다. 또는 CodeBuild에서 빌드 프로젝트를 생성한 다음, 이 작업으로 돌아오면 됩니다. CodeBuild 사용 설명서의 <u>CodeBuild를 사용하는 파이프라인 생성</u> 지침을 따 르세요.

빌드 사양에서 CodeBuild buildspec 파일은 선택 사항이며 대신 명령을 입력할 수 있습니다. 빌 드 명령 삽입에 작업에 대한 셸 명령을 입력합니다. 빌드 명령 사용 시 고려 사항에 대한 자세한 내용은 섹션을 참조하세요<u>명령 작업 참조</u>. 다른 단계에서 명령을 실행하거나 명령 목록이 긴 경 우 buildspec 파일 사용을 선택합니다.

환경 변수에서 빌드 작업에 CodeBuild 환경 변수를 추가하려면 환경 변수 추가를 선택합니다. 각 변수는 세 개의 항목으로 구성됩니다.

- 이름에 환경 변수의 이름 또는 키를 입력합니다.
- 값에 환경 변수의 값을 입력합니다. 변수 유형으로 파라미터를 선택하는 경우이 값이 AWS Systems Manager 파라미터 스토어에 이미 저장한 파라미터의 이름인지 확인합니다.

Note

환경 변수를 사용하여 민감한 값, 특히 AWS 자격 증명을 저장하지 않는 것이 좋습니 다. CodeBuild 콘솔 또는 AWS CLI를 사용하면 환경 변수가 일반 텍스트로 표시됩니 다. 민감한 값의 경우 대신 파라미터 유형을 사용하는 것이 좋습니다.

• (선택 사항) 유형에 환경 변수의 유형을 입력합니다. 유효한 값은 일반 텍스트 또는 파라미 터입니다. 기본값은 일반 텍스트입니다.

(선택 사항) 빌드 유형에서 다음 중 하나를 선택합니다.

- 단일 빌드 작업 실행으로 각 빌드를 실행하려면 단일 빌드를 선택합니다.
- 동일한 빌드 작업 실행에서 여러 빌드를 실행하려면 배치 빌드를 선택합니다.

(선택 사항) 배치 빌드를 실행하도록 선택한 경우 배치의 모든 아티팩트를 단일 위치로 결합을 선택하여 모든 빌드 아티팩트를 단일 출력 아티팩트에 배치할 수 있습니다.

- 자동 재시도를 위해 스테이지를 구성하려면 스테이지 장애 시 자동 재시도 활성화를 선택합니다.
 자동 재시도에 대한 자세한 내용은 실패 시 자동 재시도를 위한 스테이지 구성 섹션을 참조하세요.
- 3. Next(다음)를 선택합니다.

5단계: 테스트 단계 생성

빌드 또는 배포 단계를 생성하려는 경우이 단계는 선택 사항입니다.

1. 5단계: 테스트 단계 추가 페이지에서 다음 중 하나를 수행한 후 다음을 선택합니다.

- 빌드 또는 배포 단계를 생성하려는 경우 테스트 단계 건너뛰기를 선택합니다.
- 테스트 공급자에서 테스트 작업 공급자를 선택하고 적절한 필드를 작성합니다.
- 2. Next(다음)를 선택합니다.

6단계: 배포 단계 생성

빌드 단계를 이미 만든 경우 이 단계는 선택 사항입니다.

- 1. 6단계: 배포 단계 추가 페이지에서 다음 중 하나를 수행한 후 다음을 선택합니다.
 - 이전 단계에서 빌드 또는 테스트 단계를 생성한 경우 배포 단계 건너뛰기를 선택합니다.

Note

빌드 또는 테스트 단계를 이미 건너뛰었다면이 옵션이 표시되지 않습니다.

• 배포 공급자에서 배포 공급자에 대해 생성한 사용자 지정 작업을 선택합니다.

리전에서 교차 리전 작업의 경우에만 리소스가 생성되는 AWS 리전을 선택합니다. 리전 필드는 이 작업 유형 및 공급자 유형에 대해 AWS 리소스가 생성되는 위치를 지정합니다. 이 필드는 작 업 공급자가 AWS 서비스인 작업에 대해서만 표시합니다. 리전 필드는 기본적으로 파이프라인 과 동일한 AWS 리전으로 설정됩니다.

- 배포 공급자에서 다음과 같이 기본 공급자 필드를 사용 가능합니다.
 - CodeDeploy

애플리케이션 이름에서 기존 CodeDeploy 애플리케이션 이름을 입력하거나 선택합니다. Deployment group(배포 그룹)에 애플리케이션의 배포 그룹 이름을 입력합니다. Next(다음)를 선택합니다. CodeDeploy 콘솔에서 애플리케이션이나 배포 그룹을 생성하거나 둘 다 생성할 수 있습니다.

AWS Elastic Beanstalk

애플리케이션 이름에서 기존 Elastic Beanstalk 애플리케이션 이름을 입력하거나 선택합니다. Environment name(환경 이름)에 애플리케이션의 환경을 입력합니다. Next(다음)를 선택합니 다. Elastic Beanstalk 콘솔에서 애플리케이션이나 환경을 생성하거나 둘 다 생성할 수 있습니 다.

AWS OpsWorks Stacks

사용하려는 스택 이름을 Stack(스택)에 입력하거나 선택합니다. 계층에서 대상 인스턴스가 속 하는 레이어를 선택합니다. [App]에서 업데이트 및 배포하려는 애플리케이션을 선택합니다. 앱을 생성해야 하는 경우 AWS OpsWorks에서 새로 생성을 선택합니다.

의 스택 및 계층에 애플리케이션을 추가하는 방법에 대한 자세한 내용은 AWS OpsWorks 사용 설명서의 앱 추가를 AWS OpsWorks참조하세요.

CodePipeline에서 간단한 파이프라인을 AWS OpsWorks 계층에서 실행하는 코드의 소스로 사용하는 방법에 대한 end-to-end 예제는 <u>에서 CodePipeline 사용을 AWS OpsWorks Stacks</u> 참조하세요.

AWS CloudFormation

다음 중 하나를 수행합니다.

- 작업 모드에서 스택 생성 또는 업데이트를 선택하고 스택 이름과 템플릿 파일 이름을 입력 한 다음 수임 AWS CloudFormation 할 역할의 이름을 선택합니다. 선택 사항으로 구성 파일 의 이름을 입력한 다음 IAM 기능 옵션을 선택할 수도 있습니다.
- 작업 모드에서 변경 세트 생성 또는 교체를 선택하고 스택 이름을 입력한 다음 변경 세트 이 름을 변경한 다음 수임 AWS CloudFormation 할 역할의 이름을 선택합니다. 선택 사항으로 구성 파일의 이름을 입력한 다음 IAM 기능 옵션을 선택할 수도 있습니다.

CodePipeline의 파이프라인에 AWS CloudFormation 기능을 통합하는 방법에 대한 자세한 내 용은 AWS CloudFormation 사용 설명서의 <u>CodePipeline을 사용한 지속적 전달</u>을 참조하세 요.

Amazon ECS

클러스터 이름에 기존 Amazon ECS 클러스터의 이름을 입력하거나 선택합니다. Service name(서비스 이름)에서 클러스터에서 실행 중인 서비스의 이름을 입력하거나 선택합니다. 클 러스터와 서비스를 생성할 수도 있습니다. 이미지 파일 이름에 서비스의 컨테이너와 이미지 를 설명하는 이미지 정의 파일의 이름을 입력합니다.

Note

Amazon ECS 배포 작업에서는 imagedefinitions.json 파일을 배포 작업에 대 한 입력으로 사용해야 합니다. 파일의 기본 파일 이름은 imagedefinitions.json입니다. 다른 파일 이름을 사용하려면 파이프라인 배포 단계를 만들 때 이름을 제공해야 합니 다. 자세한 내용은 <u>Amazon ECS 표준 배포 작업을 위한 imagedefinitions.json 파일</u> 단 원을 참조하십시오.

Next(다음)를 선택합니다.

Note

Amazon ECS 클러스터가 두 개 이상의 인스턴스로 구성되어 있는지 확인하세요. Amazon ECS 클러스터에는 적어도 두 개의 인스턴스가 있어야 하며, 하나는 기본 인 스턴스로 유지되고 다른 하나는 새 배포를 수용하는 데 사용됩니다.

파이프라인을 통한 컨테이너 기반 애플리케이션 배포에 대한 자습서는 <u>자습서: CodePipeline</u> 을 사용하여 연속 배포를 참조하세요.

• Amazon ECS(블루/그린)

CodeDeploy 애플리케이션 및 배포 그룹, Amazon ECS 작업 정의 및 AppSpec 파일 정보를 입력하고 다음을 선택합니다.

Note

Amazon ECS(블루/그린) 작업을 수행하려면 배포 작업의 입력 아티팩트로 imageDetail.json 파일이 필요합니다. Amazon ECR 소스 작업이 이 파일을 작성하 므로 Amazon ECR 소스 작업이 있는 파이프라인은 imageDetail.json 파일을 제공할 필요가 없습니다. 자세한 내용은 <u>Amazon ECS 블루/그린 배포 작업을 위한</u> imageDetail.json 파일 단원을 참조하십시오.

CodeDeploy를 통한 Amazon ECS 클러스터 블루-그린 배포의 파이프라인 생성에 대한 자습 서는 <u>자습서: Amazon ECR 소스 및 ECS-to-CodeDeploy 배포를 사용하여 파이프라인 생성</u> 단원을 참조하세요.

AWS Service Catalog

콘솔에서 필드를 사용하여 구성을 지정하려면 Enter deployment configuration(배포 구성 입 력)을 선택하고 별도의 구성 파일이 있는 경우 구성 파일을 선택합니다. 제품 및 구성 정보를 입력하고 Next(다음)를 선택합니다. 파이프라인을 통해 Service Catalog에 제품 변경 사항을 배포하는 방법에 대한 자습서는 <u>자습</u> 서: Service Catalog에 배포하는 파이프라인 생성 단원을 참조하세요.

Alexa Skills Kit

Alexa Skill ID에 Alexa Skill의 스킬 ID를 입력합니다. Client ID(클라이언트 ID) 및 Client secret(클라이언트 암호)에서 Login with Amazon(LWA) 보안 프로필을 사용하여 생성한 자격 증명을 입력합니다. Refresh token(새로 고침 토큰)에서 새로 고침 토큰을 검색하기 위해 ASK CLI 명령을 사용하여 생성한 새로 고침 토큰을 입력합니다. Next(다음)를 선택합니다.

파이프라인에 Alexa Skill을 배포하고 LWA 자격 증명을 생성하는 방법에 대한 자습서는 <u>자습</u> 서: Amazon Alexa Skill을 배포하는 파이프라인 생성 단원을 참조하십시오.

Amazon S3

[Bucket]에 사용하려는 S3 버킷 이름을 입력합니다. 배포 단계의 입력 아티팩트가 ZIP 파일인 경우 Extract file before deploy(배포 전 파일 추출)를 선택합니다. Extract file before deploy(배 포 전 파일 추출)가 선택된 경우, 선택적으로 ZIP 파일의 압축을 해제할 Deployment path(배 포 경로) 값을 입력할 수 있습니다. 선택되지 않은 경우, S3 object key(S3 객체 키)에 값을 입 력해야 합니다.

Note

대부분의 소스 및 빌드 단계 출력 아티팩트는 압축됩니다. Amazon S3를 제외한 모든 파이프라인 소스 공급자는 소스 파일을 다음 작업의 입력 아티팩트로 제공하기 전에 압축합니다.

(선택 사항) 미리 준비된 ACL에 Amazon S3에 배포된 객체에 적용할 <u>미리 준비된 ACL</u>을 입 력합니다.

Note

표준 ACL을 적용하면 객체에 적용된 기존 ACL을 덮어씁니다.

(선택 사항) 캐시 제어에는 버킷에서 객체를 다운로드하는 요청에 대한 캐시 제어 파라미터를 지정합니다. 유효한 값 목록의 경우 HTTP 작업에 대한 <u>Cache-Control</u> 헤더 필드를 확인합

니다. 캐시 제어에 여러 값을 입력하려면 각 값 사이에 쉼표를 사용합니다. 이 예제와 같이 각 쉼표 뒤에 공백을 추가할 수 있습니다(선택 사항).

> Cache control - optional Set cache control for objects requested from your Amazon S3 bucket. public, max-age=0, no-transform

앞의 예제 항목은 CLI에 다음과 같이 표시됩니다.

"CacheControl": "public, max-age=0, no-transform"

Next(다음)를 선택합니다.

Amazon S3 배포 작업 공급자를 통한 파이프라인 생성에 대한 자습서는 <u>자습서: Amazon S3</u> 를 배포 공급자로 사용하는 파이프라인 생성 단원을 참조하세요.

- 자동 재시도를 위해 스테이지를 구성하려면 스테이지 장애 시 자동 재시도 활성화를 선택합니다.
 자동 재시도에 대한 자세한 내용은 실패 시 자동 재시도를 위한 스테이지 구성 섹션을 참조하세요.
- 자동 롤백을 위해 스테이지를 구성하려면 스테이지 장애 시 자동 롤백 구성을 선택합니다. 자동 롤 백에 대한 자세한 내용은 자동 롤백을 위한 스테이지 구성 섹션을 참조하세요.
- 4. 다음 단계를 선택합니다.

7단계: 파이프라인 검토

7단계: 검토 페이지에서 파이프라인 구성을 검토한 다음 파이프라인 생성을 선택하여 파이프라인
 을 생성하거나 이전을 선택하여 돌아가서 선택 사항을 편집합니다. 파이프라인을 생성하지 않고
 마법사를 종료하려면 [Cancel]을 선택합니다.

이제 파이프라인을 생성했으므로 콘솔에서 볼 수 있습니다. 파이프라인이 생성된 후 실행을 시작합니 다. 자세한 내용은 <u>CodePipeline에서 파이프라인 및 세부 정보 보기</u> 단원을 참조하십시오. 파이프라인 변경에 대한 자세한 내용은 CodePipeline에서 파이프라인 편집 단원을 참조하십시오.

파이프라인 생성(CLI)

AWS CLI 를 사용하여 파이프라인을 생성하려면 JSON 파일을 생성하여 파이프라인 구조를 정의한 다 음 --cli-input-json 파라미터를 사용하여 create-pipeline 명령을 실행합니다.

▲ Important

AWS CLI 를 사용하여 파트너 작업이 포함된 파이프라인을 생성할 수 없습니다. 대신 CodePipeline 콘솔을 사용해야 합니다.

파이프라인 구조에 대한 자세한 내용은 CodePipeline <u>API 참조</u>의 <u>CodePipeline 파이프라인 구조 참조</u> 및 파이프라인 생성을 참조하세요.

JSON 파일을 생성하려면 샘플 파이프라인 JSON 파일을 사용하고, 이를 편집한 다음 create-pipeline 명령을 실행할 때 해당 파일을 호출합니다.

사전 조건:

CodePipeline 시작하기에서 CodePipeline에 대해 생성한 서비스 역할의 ARN이 필요합니다. createpipeline 명령을 실행할 때 파이프라인 JSON 파일에서 CodePipeline 서비스 역할 ARN을 사용합니다. 서비스 역할 생성에 대한 자세한 내용은 <u>CodePipeline 서비스 역할 생성</u>을 참조하십시오. 콘솔과 달리 에서 create-pipeline 명령을 실행하면 CodePipeline 서비스 역할을 생성할 수 AWS CLI 있는 옵션이 없 습니다. 서비스 역할이 이미 있어야 합니다.

파이프라인의 아티팩트가 저장되는 S3 버킷의 이름이 필요합니다. 이 버킷은 파이프라인과 동일한 리 전에 있어야 합니다. create-pipeline 명령을 실행할 때 파이프라인 JSON 파일에서 버킷 이름을 사용합 니다. 콘솔과 달리에서 create-pipeline 명령을 실행해도 아티팩트를 저장하기 위한 S3 버킷이 생성되 지 AWS CLI 않습니다. 버킷은 이미 존재해야 합니다.

Note

get-pipeline 명령을 사용하여 해당 파이프라인의 JSON 구조에 대한 복사본을 가져온 다음 일 반 텍스트 편집기에서 해당 구조를 수정할 수 있습니다.

JSON 파일을 생성하려면

- 1. 터미널((Linux, macOS 또는 Unix) 또는 명령 프롬프트(Windows)에서 로컬 디렉터리의 새 텍스트 파일을 생성합니다.
- (선택 사항) 파이프라인 수준에서 변수를 하나 이상 추가할 수 있습니다. CodePipeline 작업 구성 에서 이 값을 참조할 수 있습니다. 파이프라인을 생성할 때 변수 이름과 값을 추가할 수 있으며 콘 솔에서 파이프라인을 시작할 때 값을 할당하도록 선택할 수도 있습니다.

Note

파이프라인 수준에서 변수를 추가하는 것은 선택 사항이지만, 값이 제공되지 않은 파이프 라인 수준의 변수로 지정된 파이프라인의 경우 파이프라인 실행이 실패합니다.

파이프라인 수준의 변수는 파이프라인 런타임에 확인됩니다. 모든 변수는 변경할 수 없습니다. 즉, 값을 할당한 후에는 업데이트할 수 없습니다. 확인된 값이 있는 파이프라인 수준의 변수는 각 실행 의 기록에 표시됩니다.

파이프라인 구조의 변수 속성을 사용하여 파이프라인 수준에서 변수를 제공합니다. 다음 예제에 서 변수 Variable1 값은 Value1입니다.

```
"variables": [
{
"name": "Timeout",
"defaultValue": "1000",
"description": "description"
}
]
```

이 구조를 파이프라인 JSON이나 다음 단계의 예제 JSON에 추가합니다. 네임스페이스 정보를 비 롯한 변수에 대한 자세한 내용은 변수 참조 단원을 참조하세요.

- 일반 텍스트 편집기에서 파일을 열고 값을 편집하여 생성하려는 구조를 반영합니다. 적어도 파이 프라인의 이름은 변경해야 합니다. 또한 변경을 하고자 하는지 고려해야 합니다.
 - 이 파이프라인에 대한 아티팩트가 저장되는 S3 버킷입니다.
 - 코드에 있는 소스 위치입니다.
 - 배포 공급자입니다.
 - 원하는 코드 배포 방법입니다.
 - 파이프라인의 태그입니다.

다음의 두 단계로 이루어진 샘플 파이프라인 구조에는 파이프라인 변경을 고려해야 하는 값이 강 조되어 있습니다. 파이프라인에는 3개 이상의 단계가 포함되어 있을 가능성이 큽니다.

{

```
"pipeline": {
    "roleArn": "arn:aws:iam::80398EXAMPLE::role/AWS-CodePipeline-Service",
    "stages": [
        {
            "name": "Source",
            "actions": [
                {
                     "inputArtifacts": [],
                    "name": "Source",
                    "actionTypeId": {
                         "category": "Source",
                        "owner": "AWS",
                        "version": "1",
                        "provider": "S3"
                    },
                    "outputArtifacts": [
                        {
                             "name": "MyApp"
                        }
                    ],
                    "configuration": {
                        "S3Bucket": "amzn-s3-demo-source-bucket",
                        "S3ObjectKey": "ExampleCodePipelineSampleBundle.zip",
                        "PollForSourceChanges": "false"
                    },
                    "runOrder": 1
                }
            ]
        },
        {
            "name": "Staging",
            "actions": [
                {
                    "inputArtifacts": [
                        {
                             "name": "MyApp"
                        }
                    ],
                    "name": "Deploy-CodeDeploy-Application",
                     "actionTypeId": {
                        "category": "Deploy",
                        "owner": "AWS",
                        "version": "1",
                         "provider": "CodeDeploy"
```

```
},
                 "outputArtifacts": [],
                 "configuration": {
                     "ApplicationName": "CodePipelineDemoApplication",
                     "DeploymentGroupName": "CodePipelineDemoFleet"
                },
                 "runOrder": 1
            }
        ]
    }
],
"artifactStore": {
    "type": "S3",
    "location": "codepipeline-us-east-2-250656481468"
},
"name": "MyFirstPipeline",
"version": 1,
"variables": [
    {
        "name": "Timeout",
        "defaultValue": "1000",
        "description": "description"
    }
  ]
},
"triggers": [
    {
        "providerType": "CodeStarSourceConnection",
        "gitConfiguration": {
            "sourceActionName": "Source",
            "push": [
                {
                     "tags": {
                         "includes": [
                             "v1"
                         ],
                         "excludes": [
                             "v2"
                         ]
                     }
                }
            ]
        }
    }
```

```
]
"metadata": {
    "pipelineArn": "arn:aws:codepipeline:us-
east-2:80398EXAMPLE:MyFirstPipeline",
    "updated": 1501626591.112,
    "created": 1501626591.112
    },
    "tags": [{
        "key": "Project",
        "value": "ProjectA"
    }]
}
```

이 예제는 파이프라인에 Project 태그 키와 ProjectA 값을 포함하여 파이프라인에 태그 지정 을 추가합니다. CodePipeline의 리소스 태깅에 대한 자세한 내용은 <u>리소스에 태그 지정</u> 섹션을 참 조하세요.

JSON 파일의 PollForSourceChanges 파라미터는 다음과 같이 설정되어야 합니다.

"PollForSourceChanges": "false",

CodePipeline은 Amazon CloudWatch Events를 사용하여 CodeCommit 소스 리포지토리와 브랜 치 또는 S3 소스 버킷의 변경을 감지합니다. 다음 단계에는 파이프라인에 대해 이러한 리소스를 수동으로 생성하기 위한 설명이 포함되어 있습니다. 플래그를 false로 설정하면 정기적 확인이 비활성화되어 권장되는 변경 감지 방법을 사용할 때 필요하지 않습니다.

- 4. 파이프라인과 다른 리전에서 빌드, 테스트, 배포 작업을 생성하려면 파이프라인 구조에 다음을 추 가해야 합니다. 지침은 CodePipeline에 교차 리전 작업 추가 단원을 참조하십시오.
 - 작업 파이프라인 구조에 Region 파라미터를 추가합니다.
 - artifactStores 파라미터를 사용하여 작업이 있는 각 AWS 리전에 대한 아티팩트 버킷을 지 정합니다.
- 5. 이 구조에 만족하는 경우 pipeline.json과 같은 이름으로 파일을 저장합니다.

파이프라인을 만들려면

1. create-pipeline 명령을 실행하고 --cli-input-json 파라미터를 사용하여 앞에서 생성한 JSON 파일을 지정합니다.

JSON의 name 값으로 "*MySecondPipeline*"이라는 이름을 포함하는 pipeline.json이라는 JSON 파일을 사용하여 *MySecondPipeline*이라는 파이프라인을 생성하려면 다음과 같은 명령을 사용 해야 합니다.

aws codepipeline create-pipeline --cli-input-json file://pipeline.json

🛕 Important

파일 이름 앞에 file://를 포함해야 합니다. 이 명령에 필수적입니다.

이 명령은 사용자가 생성한 전체 파이프라인의 구조를 반환합니다.

- 파이프라인을 보려면 CodePipeline 콘솔을 열고 파이프라인 목록에서 이를 선택하거나 getpipeline-state 명령을 사용합니다. 자세한 내용은 <u>CodePipeline에서 파이프라인 및 세부 정보 보기</u> 단원을 참조하십시오.
- CLI를 사용하여 파이프라인을 생성하는 경우 파이프라인에 대해 권장되는 변경 감지 리소스를 수 동으로 생성해야 합니다.
 - CodeCommit 리포지토리가 포함된 파이프라인의 경우 <u>CodeCommit 소스에 대한 EventBridge</u> 규칙 생성(CLI) 단원에 설명된 대로 CloudWatch Events 규칙을 수동으로 생성해야 합니다.
 - Amazon S3 소스가 있는 파이프라인의 경우에 설명된 대로 CloudWatch Events 규칙 및 AWS CloudTrail 추적을 수동으로 생성해야 합니다<u>EventBridge 및를 사용하는 Amazon S3 소스 작업</u> <u>에 연결 AWS CloudTrail</u>.

정적 템플릿에서 파이프라인 생성

콘솔에서 템플릿을 사용하여 지정한 소스 코드 및 속성으로 파이프라인을 구성하는 파이프라인을 생성할 수 있습니다. 작업에 사용할 소스 공급자에 대한 정보와 소스 파일 위치를 제공해야 합니다. Amazon ECR 또는 GitHub와 같이 CodeConnections에서 지원하는 타사 리포지토리에 대한 소스 작업 을 지정할 수 있습니다.

템플릿은 파이프라인에 AWS CloudFormation 대해 다음 리소스를 포함하는 스택을에 생성합니다.

 파이프라인은 V2 파이프라인 유형으로 생성됩니다. 파이프라인 유형에서 다음 옵션 중 하나를 선택 합니다. 파이프라인 유형은 특성과 가격이 다릅니다. 자세한 내용은 <u>파이프라인 유형</u> 단원을 참조하 십시오.

- 파이프라인에 대한 서비스 역할이 생성되고 템플릿에서 참조됩니다.
- 아티팩트 스토어는 파이프라인에 대해 선택한 AWS 리전 의 파이프라인에 대해 기본값으로 지정된 S3 아티팩트 버킷과 같은 기본 아티팩트 스토어를 사용하여 만들어집니다.

정적 템플릿 생성 마법사에 사용되는 오픈 소스 스타터 템플릿 모음을 보려면 <u>https://github.com/aws/</u> <u>codepipeline-starter-templates</u>://https//https//h

정적 템플릿을 사용하여 파이프라인을 생성할 때 파이프라인 구조는 사용 사례의 필요에 따라 각 템플 릿에 구성됩니다. 예를 들어에 배포하기 위한 템플릿 AWS CloudFormation 은이 절차의 예로 사용됩니 다. 템플릿은 다음 구조를 사용하여 DeployToCloudFormationService라는 파이프라인을 생성합니다.

- 마법사에서 지정한 구성의 소스 작업이 포함된 빌드 스테이지입니다.
- AWS CloudFormation에서 배포 작업 및 관련 리소스 스택이 있는 배포 스테이지입니다.

정적 템플릿을 사용하여 파이프라인을 생성할 때 CodePipeline은 스테이지 이름(소스, 빌드, 스 테이징)을 생성합니다. 이러한 이름은 변경할 수 없습니다. 나중에 추가한 단계에 특정 이름(예: BuildToGamma 또는 DeployToProd)을 더 사용할 수 있습니다.

1단계: 생성 옵션 선택

- 1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> <u>codepipeline/home</u> CodePipeline 콘솔을 엽니다.
- 2. [Welcome] 페이지에서 [Create pipeline]을 선택합니다.

CodePipeline을 처음 사용하는 경우 시작하기를 선택합니다.

 1단계: 생성 옵션 선택 페이지의 생성 옵션에서 템플릿에서 파이프라인 생성 옵션을 선택합니다. Next(다음)를 선택합니다.

2단계: 템플릿 선택

배포 스테이지, 자동화 또는 CI 파이프라인을 사용하여 파이프라인을 생성할 템플릿을 선택합니다.

- 1. 2단계: 템플릿 선택 페이지에서 다음 중 하나를 수행한 후 다음을 선택합니다.
 - 배포 스테이지를 생성하려는 경우 배포를 선택합니다. ECR 또는 CloudFormation에 배포하는 템플릿에 대한 옵션을 봅니다. 이 예에서는 배포를 선택한 다음, CloudFormation에 배포하도록 선택합니다.

- CI 파이프라인을 생성하려는 경우 지속적 통합을 선택합니다. Gradle에 빌드와 같은 CI 파이프 라인의 옵션을 봅니다.
- 자동화된 파이프라인을 생성하려는 경우 자동화를 선택합니다. Python 빌드 예약과 같은 자동 화 옵션을 확인합니다.

Choose creation option	Step 2 of 4		
Choose template	Category		
Step 3 Choose source	Deployment	 Continuous Integration 	 Automation
Step 4 Configure template			
<u> </u>			
	Template		
	Push to ECR Build and push a new container image to Amazon ECR	O Deploy to ECS Fargate Deploy a ECR image to Amazon ECS Fargate cluster	 Deploy to CloudFormatic Deploy your cloud formation template
	Template Push to ECR Build and push a new container image to Amazon ECR	O Deploy to ECS Fargate Deploy a ECR image to Amazon ECS Fargate cluster	Deploy to CloudForm Deploy your cloud form template

Choose template	Category				
Step 3 Choose source Step 4	Deployment	• Continuous Integration	Automation		
Configure template	Template				
	CI Build Gradle Build a gradle project	CI Build Node JS Build a node is project	CI Build Maven Build a maven project		
	Cl Build Python Build a python project				
			Cancel Previous Next		
Choose template	Category				
Step 3 Choose source Step 4 Configure template	O Deployment	O Continuous Integration	• Automation		
	Template				
	 Schedule a python build pipeline Build a python project periodically 	Schedule a gradle build pipeline Build a gradle project periodically	 Schedule a nodejs build pipeline Build a nodejs project periodically 		

3단계: 소스 선택

- 3단계: 소스 선택 페이지의 소스 공급자에서 소스 코드가 저장된 리포지토리의 공급자를 선택하고 필수 옵션을 지정한 다음, 다음 단계를 선택합니다.
 - Bitbucket Cloud, GitHub(GitHub App을 통해), GitHub Enterprise Server, GitLab.com, 또는 GitLab 자체 관리형의 경우:
 - 1. 연결에서 기존 연결을 선택하거나 새로 생성합니다. GitHub 소스 작업에 대한 연결을 만들거 나 관리하려면 GitHub 연결을 참조하세요.
 - 2. 파이프라인의 소스 위치로 사용할 리포지토리를 선택합니다.

트리거를 추가하거나 트리거 유형을 필터링하도록 선택하여 파이프라인을 시작합니다. 트리 거 작업에 대한 자세한 내용은 <u>코드 푸시 또는 풀 요청 이벤트 유형을 사용하여 트리거 추가</u> 섹션을 참조하세요. Glog 패턴을 사용한 필터링에 대한 자세한 내용은 <u>구문에서 glob 패턴 작</u> 업을 참조하세요.

- 3. 출력 아티팩트 형식에서 아티팩트의 형식을 선택합니다.
 - 기본 방법을 사용하여 GitHub 작업의 출력 아티팩트를 저장하려면 CodePipeline 기본 방법을 선택합니다. 그러면 GitHub 리포지토리의 파일에 액세스하여 파이프라인 아티팩트 스토어에 ZIP 파일로 아티팩트가 저장됩니다.
 - 다운스트림 작업이 Git 명령을 직접 수행할 수 있도록 리포지토리에 대한 URL 참조가 포함 된 JSON 파일을 저장하려면 Full clone(전체 복제)을 선택합니다. 이 옵션은 CodeBuild 다 운스트림 작업에서만 사용할 수 있습니다.

이 옵션을 선택하면 <u>CodePipeline 문제 해결</u>과 같이 CodeBuild 프로젝트 서비스 역할에 대 한 권한을 업데이트해야 합니다. 전체 복제 옵션을 사용하는 방법을 보여주는 자습서는 <u>자</u> 습서: GitHub 파이프라인 소스와 함께 전체 복제 사용을 참조하세요.

- Amazon ECR의 경우:
 - 리포지토리 이름에서 Amazon ECR 리포지토리의 이름을 선택합니다.
 - 이미지 이름과 버전이 LATEST와 다른 경우 이미지 태그에서 지정합니다.
 - 출력 아티팩트에서 출력 아티팩트 기본값(예: MyApp)을 선택합니다. 이 아티팩트는 다음 단 계에서 사용하려는 이미지 이름과 리포지토리 URI를 포함합니다.

파이프라인에 Amazon ECR 소스 단계를 포함하면 소스 작업이 변경 사항을 커밋할 때 imageDetail.json 파일을 출력 아티팩트로 생성합니다.imageDetail.json 파일에 대한 자세한 내용은 <u>Amazon ECS 블루/그린 배포 작업을 위한 imageDetail.json 파일</u> 단원을 참조하 십시오.

1 Note

객체 및 파일 형식이 사용하려는 배포 시스템과 호환되어야 합니다(예: Elastic Beanstalk 또는 CodeDeploy). 지원되는 파일 형식에는 .zip, .tar, .tgz 파일 등이 있습니다. Elastic Beanstalk에 지원되는 컨테이너 형식에 대한 자세한 내용은 <u>Elastic Beanstalk 환경 사용</u> <u>자 지정 및 구성 및 지원 플랫폼</u> 단원을 참조하세요. CodeDeploy를 사용하여 개정을 배포 하는 방법에 대한 자세한 내용은 <u>애플리케이션 개정 업로드</u> 및 <u>개정 준비</u> 단원을 참조하세 요.

4단계: 템플릿 구성

이 예에서는 CloudFormation에 대한 배포가 선택되었습니다. 이 단계에서는 템플릿에 대한 구성을 추 가합니다.

Choose template	Template Details		
Step 3			
Choose source	ConnectionArn		
Step 4 Configure template	arn:aws:codeconnections:us-east-1:		
	FullRepositoryId The full repository ID to use with your CodeConnections connection. :/MyGitHubRepo		
	BranchName The branch name to use with your CodeConnections connection.		
	main		
	CodePipelineName The CodePipeline pipeline name that will deploy your cfn template.		
	DeployToCloudFormationService2		
	StackName The CloudFormation Stack Name that you want to create and/or update. DeployToCloudFormationService2		
	TemplatePath The path in your source repository to the CloudFormation template to create and/or update your Stack. template.yaml		
	OutputFileName The path the output from the CloudFormation stack update will be written to. output.json		

- 1. 4단계: 템플릿 구성의 스택 이름에 파이프라인의 이름을 입력합니다.
- 2. 템플릿에 적용되는 권한에 대한 자리 표시자 IAM 정책을 편집합니다.
- 3. 템플릿에서 파이프라인 생성을 선택합니다.
- 4. 파이프라인 리소스가 생성 중이라는 메시지가 표시됩니다.

5단계: 파이프라인 검토

 이제 파이프라인을 생성했으므로 CodePipeline 콘솔에서 파이프라인을 볼 수 있으며, AWS CloudFormation에서 스택을 볼 수 있습니다. 파이프라인이 생성된 후 실행을 시작합니다. 자세한 내용은 <u>CodePipeline에서 파이프라인 및 세부 정보 보기</u> 단원을 참조하십시오. 파이프라인 변경에 대한 자세한 내용은 CodePipeline에서 파이프라인 편집 단원을 참조하십시오.

CodePipeline에서 파이프라인 편집

파이프라인은 완료해야 하는 단계 및 작업을 포함하여 따르 AWS CodePipeline 려는 릴리스 프로세스 를 설명합니다. 파이프라인을 편집하여 이러한 요소를 추가하거나 제거할 수 있습니다. 그러나 파이프 라인을 편집할 때 파이프라인 이름이나 파이프라인 메타데이터 등의 값은 변경할 수 없습니다.

파이프라인 편집 페이지를 사용하여 파이프라인 유형, 변수, 트리거를 편집할 수 있습니다. 파이프라인 에서 단계 및 작업을 추가하거나 변경할 수도 있습니다.

파이프라인 생성과 달리 파이프라인 편집은 파이프라인을 통해 최신 개정을 다시 실행하지 않습니다. 방금 편집한 파이프라인을 통해 최신 개정을 실행하려면 수동으로 다시 실행해야 합니다. 그렇지 않으 면 다음에 사용자가 소스 단계에 구성된 소스 위치를 변경할 때 편집된 파이프라인이 실행됩니다. 자세 한 내용은 수동으로 파이프라인 시작을 참조하세요.

파이프라인과 다른 AWS 리전에 있는 파이프라인에 작업을 추가할 수 있습니다. AWS 서비스 가 작업 의 공급자이고이 작업 유형/공급자 유형이 파이프라인과 다른 AWS 리전에 있는 경우 이는 리전 간 작 업입니다. 교차 리전 작업에 대한 자세한 내용은 <u>CodePipeline에 교차 리전 작업 추가</u> 항목을 참조하십 시오.

CodePipeline은 소스 코드 변경이 푸시될 때 변경 감지 방법을 사용하여 파이프라인을 시작합니다. 이 러한 감지 방법은 소스 유형을 기반으로 합니다.

• CodePipeline은 Amazon CloudWatch Events를 사용하여 CodeCommit 소스 리포지토리 또는 Amazon S3 소스 버킷의 변경을 감지합니다.

Note

콘솔을 사용하면 변경 감지 리소스가 자동으로 생성됩니다. 콘솔을 사용하여 파이프라인을 생 성하거나 편집하면 추가 리소스가 자동으로 생성됩니다. AWS CLI 를 사용하여 파이프라인을 생성하는 경우 추가 리소스를 직접 생성해야 합니다. CodeCommit 파이프라인 생성 또는 업데 이트 방법에 대한 자세한 내용은 <u>CodeCommit 소스에 대한 EventBridge 규칙 생성(CLI)</u> 단원 을 참조하세요. CLI를 사용하여 Amazon S3 파이프라인을 생성 또는 업데이트하는 방법에 대 한 자세한 내용은 Amazon S3 소스에 대한 EventBridge 규칙 생성(CLI) 단원을 참조하세요.

주제

- 파이프라인 편집(콘솔)
- <u>파이프라인 편집(AWS CLI)</u>

파이프라인 편집(콘솔)

CodePipeline 콘솔을 사용하여 파이프라인의 단계를 추가, 편집 또는 제거하거나, 단계 내의 작업을 추가, 편집, 제거할 수 있습니다.

파이프라인을 업데이트하면 CodePipeline이 실행 중인 모든 작업을 정상적으로 완료한 다음 실행 중인 작업이 완료된 단계 및 파이프라인 실행에 실패합니다. 파이프라인이 업데이트되면 파이프라인을 다 시 실행해야 합니다. 파이프라인 실행에 대한 자세한 내용은 <u>수동으로 파이프라인 시작</u> 단원을 참조하 세요.

파이프라인을 편집하려면

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https//https//https://https//ht

AWS 계정과 연결된 모든 파이프라인의 이름이 표시됩니다.

- [Name]에서 편집할 파이프라인의 이름을 선택합니다. 이렇게 하면 파이프라인 각 단계의 각 작업 상태를 포함하여 파이프라인의 세부 정보 보기가 열립니다.
- 3. 파이프라인 세부 정보 페이지에서 [Edit]를 선택합니다.
- 파이프라인 유형을 편집하려면 편집: 파이프라인 속성 카드에서 편집을 선택합니다. 다음 옵션 중 하나를 선택한 다음 완료를 선택합니다.

- V1 유형 파이프라인은 표준 파이프라인, 단계, 작업 수준 파라미터를 포함하는 JSON 구조를 가 지고 있습니다.
- V2 유형 파이프라인은 트리거 및 파이프라인 수준 변수와 같은 추가 매개변수 지원과 함께 V1
 유형과 구조가 동일합니다.

파이프라인 유형은 특성과 가격이 다릅니다. 자세한 내용은 <u>파이프라인 유형</u> 단원을 참조하십시 오.

 파이프라인 변수를 편집하려면 편집: 변수 카드에서 변수 편집을 선택합니다. 파이프라인 수준의 변수를 추가하거나 변경한 다음 완료를 선택합니다.

파이프라인 수준에서 변수에 대한 자세한 정보는 <u>변수 참조</u>을 참조하세요. 파이프라인 실행 시 전 달되는 파이프라인 수준 변수에 대한 자습서는 자습서: 파이프라인 수준 변수 사용을 참조하세요.

Note

파이프라인 수준에서 변수를 추가하는 것은 선택 사항이지만, 값이 제공되지 않은 파이프 라인 수준의 변수로 지정된 파이프라인의 경우 파이프라인 실행이 실패합니다.

 파이프라인 트리거를 편집하려면 편집: 트리거 카드에서 트리거 편집을 선택합니다. 트리거를 추 가 또는 변경한 다음 완료를 선택합니다.

트리거 추가에 대한 자세한 내용은 Bitbucket Cloud, GitHub(GitHub App을 통해), GitHub Enterprise Server, GitLab.com, 또는와 같은 GitLab 자체 관리형에 대한 연결을 생성하는 단계를 참조하세요GitHub 연결.

- 7. 편집 페이지에서 단계 및 작업을 편집하려면 다음 중 하나를 수행합니다.
 - 단계를 편집하려면 Edit stage(단계 편집)를 선택합니다. 기존 작업과 연속되게/동시에 실행되는 작업을 추가할 수 있습니다.

해당 작업의 편집 아이콘을 선택하면 이 보기에서 작업을 편집할 수도 있습니다. 작업을 삭제하 려면 해당 작업의 삭제 아이콘을 선택합니다.

- 작업을 편집하려면 해당 작업의 편집 아이콘을 선택한 후 [Edit action]에서 값을 변경합니다. 별 표(*)로 표시된 항목은 필수 항목입니다.
 - CodeCommit 리포지토리 이름과 브랜치의 경우, 이 파이프라인에 대해 생성될 Amazon CloudWatch Events 규칙을 보여주는 메시지가 표시됩니다. CodeCommit 소스를 제거하는 경우 제거될 Amazon CloudWatch Events 규칙을 보여주는 메시지가 표시됩니다.

- Amazon S3 소스 버킷의 경우, 이 파이프라인에 대해 생성될 Amazon CloudWatch Events 규 칙 및 AWS CloudTrail 추적을 보여주는 메시지가 표시됩니다. Amazon S3 소스를 제거하면 삭제할 Amazon CloudWatch Events 규칙과 AWS CloudTrail 추적을 보여주는 메시지가 나타 납니다. AWS CloudTrail 추적이 다른 파이프라인에서 사용 중인 경우 추적이 제거되지 않고 데이터 이벤트가 삭제됩니다.
- 단계를 추가하려면 단계를 추가하려는 파이프라인의 지점에서 + Add stage(단계 추가)를 선택 합니다. 단계의 이름을 입력한 다음 하나 이상의 작업을 추가합니다. 별표(*)로 표시된 항목은 필 수 항목입니다.
- 단계를 삭제하려면 해당 단계의 삭제 아이콘을 선택합니다. 단계 및 해당 작업 모두가 삭제됩니다.
- 실패 시 자동으로 롤백하도록 스테이지를 구성하려면 스테이지 편집을 선택한 다음 스테이지 실패 시 자동 롤백 구성 확인란을 선택합니다.

예를 들어, 파이프라인의 단계에 연쇄 작업을 추가하려면

- 1. 작업을 추가하려는 단계에서 Edit stage(단계 편집)를 선택한 다음, + Add action group(작업 그 룹 추가)을 선택합니다.
- 2.

Edit action(작업 편집)의 Action name(작업 이름)에 작업 이름을 입력합니다. Action provider(작 업 공급자) 목록은 공급자 옵션을 범주별로 표시합니다. 해당 범주를 찾습니다(예: Deploy). 해 당 범주에서 공급자를 선택합니다(예: AWS CodeDeploy). 리전에서 리소스가 생성된 또는 리 소스를 생성하려는 AWS 리전을 선택합니다. 리전 필드는이 작업 유형 및 공급자 유형에 대해 AWS 리소스가 생성되는 위치를 지정합니다. 이 필드는 작업 공급자가 AWS 서비스인 작업에 대해서만 표시합니다. 리전 필드는 기본적으로 파이프라인과 동일한 AWS 리전으로 설정됩니 다.

작업 공급자를 추가하고 각 공급자에 대한 기본 필드를 사용하는 예제는 <u>사용자 지정 파이프라</u> 인 생성(콘솔)을 참조하십시오.

CodeBuild를 빌드 작업 또는 스테이지에 테스트 액션으로 추가하려면 CodeBuild 사용 설명 서의 CodeBuild와 함께 CodePipeline을 사용하여 코드 테스트 및 빌드 실행을 참조하세요.

Note

GitHub와 같은 일부 작업 공급자는 작업 구성을 완료하기 전에 공급자의 웹 사이트에 연결하도록 요청하기도 합니다. 공급자의 웹 사이트에 연결할 때는 해당 웹 사이트의 자격 증명을 사용해야 합니다. 자격 AWS 증명을 사용하지 마세요.

3. 작업 구성을 마쳤으면 저장을 선택합니다.

Note

콘솔 보기에서는 스테이지 이름을 바꿀 수 없습니다. 변경하려는 이름으로 스테이지를 추 가한 다음 이전 스테이지를 삭제할 수 있습니다. 기존 단계나 작업을 삭제하기 전에 해당 단계에 있어야 하는 모든 작업이 추가되었는지 확인합니다.

8. 파이프라인 편집을 마쳤으면 저장을 선택하여 요약 페이지로 돌아갑니다.

▲ Important

변경 내용을 저장한 후에는 취소할 수 없습니다. 파이프라인을 다시 편집해야 합니다. 변 경 사항을 저장할 때 파이프라인을 통해 개정이 실행되고 있으면 실행이 완료되지 않습니 다. 편집한 파이프라인을 통해 특정 커밋이나 변경 사항이 실행되도록 하려면 파이프라인 을 통해 해당 커밋이나 변경 사항을 수동으로 실행해야 합니다. 그렇지 않으면, 다음 커밋 이나 변경 사항이 파이프라인을 통해 자동으로 실행됩니다.

9. 작업을 테스트하려면 변경 사항 릴리스를 선택하여 파이프라인을 통해 해당 커밋을 처리하거나, 파이프라인의 소스 단계에 지정되어 있는 소스에 대한 변경 사항을 커밋합니다. 또는의 단계에 따 라 수동으로 파이프라인 시작를 사용하여 변경 사항을 AWS CLI 수동으로 릴리스합니다.

파이프라인 편집(AWS CLI)

update-pipeline 명령을 사용하여 파이프라인을 편집할 수 있습니다.

파이프라인을 업데이트하면 CodePipeline이 실행 중인 모든 작업을 정상적으로 완료한 다음 실행 중인 작업이 완료된 단계 및 파이프라인 실행에 실패합니다. 파이프라인이 업데이트되면 파이프라인을 다 시 실행해야 합니다. 파이프라인 실행에 대한 자세한 내용은 <u>수동으로 파이프라인 시작</u> 단원을 참조하 세요.

A Important

AWS CLI 를 사용하여 파트너 작업이 포함된 파이프라인을 편집할 수 있지만 파트너 작업의 JSON을 수동으로 편집해서는 안 됩니다. 이렇게 하면 파이프라인 업데이트 후 파트너 작업이 실패합니다.

파이프라인을 편집하려면

 터미널 세션(Linux, macOS 또는 Unix) 또는 명령 프롬프트(Windows)를 열고 get-pipeline 명령을 실행하여 파이프라인 구조를 JSON 파일에 복사합니다. 예를 들어, MyFirstPipeline라는 파이 프라인에서는 다음 명령을 입력합니다.

aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json

이 명령은 아무 것도 반환하지 않지만 생성한 파일이 명령을 실행한 디렉터리에 표시되어야 합니 다.

 일반 텍스트 편집기에서 JSON 파일을 열고 파일의 구조를 수정하여 파이프라인에 대한 변경 사항 을 반영합니다. 예를 들어 단계를 추가 또는 제거하거나 기존 단계에 다른 작업을 추가할 수 있습 니다.

다음 예에서는 pipeline.json 파일에 다른 배포 단계를 추가하는 방법을 보여 줍니다. 이 단계는 *Staging*이라는 첫 번째 배포 단계 이후에 실행됩니다.

Note

다음은 파일의 전체 구조가 아닌 일부입니다. 자세한 내용은 <u>CodePipeline 파이프라인 구</u> <u>조 참조</u> 단원을 참조하십시오.

```
}
                     ],
                     "name": "Deploy-CodeDeploy-Application",
                     "actionTypeId": {
                         "category": "Deploy",
                         "owner": "AWS",
                         "version": "1",
                         "provider": "CodeDeploy"
                    },
                     "outputArtifacts": [],
                     "configuration": {
                         "ApplicationName": "CodePipelineDemoApplication",
                         "DeploymentGroupName": "CodePipelineDemoFleet"
                    },
                     "runOrder": 1
                }
            ]
         },
    {
        "name": "Production",
        "actions": [
                {
                     "inputArtifacts": [
                         {
                             "name": "MvApp"
                         }
                    ],
                     "name": "Deploy-Second-Deployment",
                     "actionTypeId": {
                         "category": "Deploy",
                         "owner": "AWS",
                         "version": "1",
                         "provider": "CodeDeploy"
                    },
                     "outputArtifacts": [],
                     "configuration": {
                     "ApplicationName": "CodePipelineDemoApplication",
                     "DeploymentGroupName": "CodePipelineProductionFleet"
                    },
                    "runOrder": 1
                }
            ]
    }
]
```

}

CLI를 사용하여 파이프라인에 승인 작업을 추가하는 방법에 대한 자세한 내용은 <u>CodePipeline의</u> 파이프라인에 수동 승인 작업 추가 단원을 참조하십시오.

JSON 파일의 PollForSourceChanges 파라미터는 다음과 같이 설정되어야 합니다.

"PollForSourceChanges": "false",

CodePipeline은 Amazon CloudWatch Events를 사용하여 CodeCommit 소스 리포지토리와 브랜 치 또는 Amazon S3 소스 버킷의 변경을 감지합니다. 다음 단계에는 이러한 리소스를 수동으로 생 성하기 위한 설명이 포함되어 있습니다. 플래그를 false로 설정하면 정기적 확인이 비활성화되 어 권장되는 변경 감지 방법을 사용할 때 필요하지 않습니다.

- 3. 파이프라인과 다른 리전에서 빌드, 테스트, 배포 작업을 추가하려면 파이프라인 구조에 다음을 추 가해야 합니다. 자세한 지침은 CodePipeline에 교차 리전 작업 추가 섹션을 참조하세요.
 - 작업 파이프라인 구조에 Region 파라미터를 추가합니다.
 - artifactStores 파라미터를 사용하여 작업이 있는 각 리전에 대해 아티팩트 버킷을 지정합 니다.
- 4. get-pipeline 명령을 사용하여 검색한 파이프라인 구조로 작업 중인 경우, JSON 파일의 구조를 수정해야 합니다. update-pipeline 명령이 JSON 파일을 사용할 수 있도록 하려면 이 파일에서 metadata 라인을 삭제해야 합니다. JSON 파일의 파이프라인 구조에서 단원("metadata":
 { } 행과 "created", "pipelineARN" 및 "updated" 필드)을 삭제합니다.

예를 들어, 구조에서 다음 행을 삭제합니다.

```
"metadata": {
    "pipelineArn": "arn:aws:codepipeline:region:account-ID:pipeline-name",
    "created": "date",
    "updated": "date"
}
```

파일을 저장합니다.

- 5. CLI를 사용하여 파이프라인을 편집하는 경우 파이프라인에 대해 권장되는 변경 감지 리소스를 수 동으로 관리해야 합니다.
 - CodeCommit 리포지토리의 경우 <u>CodeCommit 소스에 대한 EventBridge 규칙 생성(CLI)</u> 단원에 설명된 대로 CloudWatch Events 규칙을 생성해야 합니다.

- Amazon S3 소스의 경우에 설명된 대로 CloudWatch Events 규칙 및 AWS CloudTrail 추적을 생 성해야 합니다EventBridge 및를 사용하는 Amazon S3 소스 작업에 연결 AWS CloudTrail.
- 6. 변경 사항을 적용하려면 파이프라인 JSON 파일을 지정하여 update-pipeline 명령을 실행합니다.

▲ Important

파일 이름 앞에 file://를 포함해야 합니다. 이 명령에 필수적입니다.

aws codepipeline update-pipeline --cli-input-json file://pipeline.json

이 명령은 편집한 파이프라인의 전체 구조를 반환합니다.

Note

update-pipeline 명령을 실행하면 파이프라인이 중지됩니다. update-pipeline 명령을 실행 할 때 파이프라인을 통해 개정을 실행하는 중이라면 해당 실행이 중지됩니다. 업데이트된 파이프라인을 통해 해당 개정을 실행하려면 파이프라인을 수동으로 시작해야 합니다.

7. CodePipeline 콘솔을 열고 방금 편집한 파이프라인을 선택합니다.

파이프라인에 변경 사항이 표시됩니다. 다음에 사용자가 소스 위치를 변경할 경우, 파이프라인의 개정된 구조를 통해 해당 개정이 실행됩니다.

 파이프라인의 개정된 구조를 통해 마지막 개정을 수동으로 실행하려면 start-pipeline-execution 명 령을 실행합니다. 자세한 내용은 수동으로 파이프라인 시작 단원을 참조하십시오.

파이프라인의 구조와 예상 값에 대한 자세한 내용은 <u>CodePipeline 파이프라인 구조 참조</u> 및 <u>AWS</u> CodePipeline API 참조를 참조하세요.

CodePipeline에서 파이프라인 및 세부 정보 보기

AWS CodePipeline 콘솔 또는 AWS CLI 를 사용하여 AWS 계정과 연결된 파이프라인에 대한 세부 정 보를 볼 수 있습니다.

주제

• <u>파이프라인 보기(콘솔)</u>

- 파이프라인에서 작업 세부 정보 보기(콘솔)
- 파이프라인 ARN 및 서비스 역할 ARN 보기(콘솔)
- 파이프라인 세부 정보 및 이력 보기(CLI)
- 실행 기록에서 스테이지 조건에 대한 규칙 결과 보기

파이프라인 보기(콘솔)

파이프라인의 상태, 이전 및 아티팩트 업데이트를 볼 수 있습니다.

Note

한 시간 후에는 파이프라인의 상세 보기가 사용자의 브라우저에서 자동으로 새로고침을 중단 합니다. 현재 정보를 보려면 페이지를 새로 고치십시오.

파이프라인을 보려면

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home CodePipeline 콘솔을 엽니다.

파이프라인 페이지가 표시됩니다. 해당 리전에 대한 모든 파이프라인의 목록이 표시됩니다.

AWS 계정과 연결된 모든 파이프라인의 이름, 유형, 상태, 버전, 생성 날짜 및 마지막 수정 날짜가 가장 최근에 시작된 실행 시간과 함께 표시됩니다.

2. 가장 최근 5개 실행의 상태가 표시됩니다.

Pipe	clines Info	🗘 Notify 🔻 Vie	ew history Release c	hange Delete pipeline	Create pipeline
Q					< 1 > 🕲
	Name	Latest execution status	Latest execution started	Most recent executions	
0	Pipeline-trigger (Type: V2 Execution mode: SUPERSEDED)	Succeeded	2 days ago	$\oslash \odot \odot \odot \ominus \ominus$ View details	5
0	check1 (Type: V2 Execution mode: SUPERSEDED)	⊗ Failed	2 days ago	🛞 🛞 🛞 🔗 🕑 View details	5
0	tr-pi2 (Type: V2 Execution mode: QUEUED)	⊖ Stopped	19 days ago	\ominus 😣 View details	
0	Pipeline-Stack (Type: V1 Execution mode: SUPERSEDED)	⊗ Failed	2 months ago	🛞 🛞 🛞 View details	
0	Pipeline-ChangeSet (Type: V2 Execution mode: QUEUED)	⊗ Failed	2 months ago	🛞 🛞 🛞 View details	

특정 행 옆의 세부 정보 보기를 선택하면 가장 최근 실행이 나열된 세부 정보 대화 상자가 표시됩 니다.

Most recent executions ×				
Trigger StartPipelineExecution	- assumed-role,		2	Â
Pipeline execution ID	Status	Last updated		
7cb97af6	In progress	51 minutes ago		
Trigger				
StartPipelineExecution	- assumed-role/			
Pipeline execution ID	Status	Last updated		
b289be6e	⊘ Succeeded	1 hour ago		
Trigger				
Webhook - connection/	40d122c4-23fb-48	bf-a08f-	2	
Pipeline execution ID	Status	Last updated		
049c2110	Succeeded Succeeded	3 months ago		
Trigger				
Webhook - connection/	40d122c4-23fb-48	bf-a08f	2	
Pipeline execution ID	Status	Last updated		
3dcaf66a	Succeeded Succeeded	4 months ago		
Trigger				
Webhook - connection/	40d122c4-23fb-48	8bf-a08f-	Z	
				· · · · · · · · · · · · · · · · · · ·
				Done

파이프라인의 가장 최근의 실행 내역에 대한 세부 정보를 보려면 View history(기록 보기)를 선택 합니다. 과거 실행 내역의 경우, 실행 ID, 상태, 시작 및 종료 시간, 실행 기간, 커밋 ID 및 메시지 등 의 소스 아티팩트와 연결된 개정 세부 정보를 볼 수 있습니다.

Note

PARALLEL 실행 모드의 파이프라인의 경우 기본 파이프라인 보기에 파이프라인 구조 또 는 진행 중인 실행이 표시되지 않습니다. PARALLEL 실행 모드의 파이프라인의 경우 실행 기록 페이지에서 보려는 실행의 ID를 선택하여 파이프라인 구조에 액세스합니다. 왼쪽 탐
색에서 기록을 선택하고 병렬 실행의 실행 ID를 선택한 다음 시각화 탭에서 파이프라인을 봅니다.

 단일 파이프라인에 대한 세부 정보를 보려면 이름에서 파이프라인을 선택합니다. 각 단계의 각 작 업 상태 및 전환 상태를 포함하여 파이프라인의 세부 정보 보기가 표시됩니다.

Source Succeeded Pipeline execution ID: <u>b289be6e-e25d-4f61-a5c3-821028cfe4d3</u>	
Source <u>GitHub (Version 2)</u> [2] Sourceeded - <u>1 minute 200</u> 2de85792 [2] View details	
2de8579a 🖾 Source: Add files via upload	0
Disable transition	
Succeeded Pipeline execution ID: <u>b289be6e-e25d-4f61-a5c3-821028cfe4d3</u>	
Build AWS CodeBuild Succeeded - Just now View details	
2de8579a 🖾 Source: Add files via upload	

그래프 형태 보기에는 각 단계에 대해 다음과 같은 정보가 표시됩니다.

- 단계 이름.
- 상태에 대해 구성된 모든 작업.
- 단계 사이에 있는 화살표의 상태로 알 수 있는 단계 간 전환 상태(활성화됨 또는 비활성화됨). 활 성화된 전환은 그 옆에 위치한 Disable transition(전환 비활성화) 버튼을 이용하면 화살표로 표 시됩니다. 비활성화된 전환은 그 아래에 위치한 그 옆에 위치한 취소선과 그 옆에 위치한 Enable transition(전환 활성화) 버튼을 이용하면 화살표로 표시됩니다.
- 단계의 상태를 나타내는 색상 표시줄.
 - 회색: 아직 실행이 없음

- 파란색: 진행 중
- 녹색: 성공
- 빨간색: 실패

그래프 형태 보기에는 각 단계의 작업에 대해 다음과 같은 정보도 표시됩니다.

- 작업의 이름.
- CodeDeploy와 같은 작업의 공급자.
- 작업이 마지막으로 실행된 시기.
- 작업의 성공 또는 실패 여부.
- 작업의 마지막 실행에 대한 기타 세부 정보에 대한 링크(있는 경우).
- 단계의 가장 마지막 파이프라인 실행을 통해 실행 중인 소스 개정에 대한 세부 정보 또는 대상 인스턴스에 배포된 가장 마지막 소스 개정(CodeDeploy 배포의 경우).
- 세부 정보 보기 버튼을 클릭하면 작업 실행, 로그 및 작업 구성에 대한 세부 정보가 포함된 대화 상자가 열립니다.

Note

로그 탭은 CodeBuild 및 파이프라인 계정에서 실행된 AWS CloudFormation 작업에 사용할 수 있습니다.

- 작업 공급자의 세부 정보를 보려면 공급자를 선택합니다. 예를 들어, 앞의 파이프라인 예에서, 스 테이징이나 프로덕션 단계 중 하나에서 CodeDeploy를 선택했다면 해당 단계에 대해 구성된 배포 그룹의 CodeDeploy 콘솔 페이지가 표시됩니다.
- 작업에 대한 진행 세부 정보를 보려면 진행 중인 작업(진행 중 메시지로 표시됨) 옆에 표시되는 세 부 정보를 확인합니다. 작업이 진행 중인 경우 증분적 진행 상황과 함께 작업의 진행 상황에 따른 단계나 작업이 표시됩니다.
- 6. 수동 승인을 받도록 구성된 작업을 승인하거나 거부하려면 검토를 선택합니다.
- 7. 성공적으로 완료되지 않은 단계의 작업을 다시 시도하려면 Retry(재시도)를 선택합니다.
- 8. 성공 또는 실패 등의 작업 결과를 포함하여 가장 최근 실행 작업의 상태가 표시됩니다.

파이프라인에서 작업 세부 정보 보기(콘솔)

각 단계의 작업 세부 정보를 포함하여 파이프라인의 세부 정보를 볼 수 있습니다.

Note

한 시간 후에는 파이프라인의 상세 보기가 사용자의 브라우저에서 자동으로 새로고침을 중단 합니다. 현재 정보를 보려면 페이지를 새로 고치십시오.

파이프라인에서 작업 세부 정보를 보려면

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home CodePipeline 콘솔을 엽니다.

파이프라인 페이지가 표시됩니다.

 어떤 작업에서든 세부 정보 보기를 선택하면 작업 실행, 로그 및 작업 구성에 대한 세부 정보가 포 함된 대화 상자가 열립니다.

Note

로그 탭은 CodeBuild와 AWS CloudFormation 작업에 사용할 수 있습니다.

 파이프라인 단계의 작업에 대한 작업 요약을 보려면 작업에 대한 세부 정보 보기를 선택한 다음 요 약 탭을 선택합니다.

Action name: Build Status: Succeeded				
Summary Logs Configuration				
Status	Last updated			
⊘ Succeeded	1 minute ago			
Action execution ID				
850739e4-13ef-4de8-a721-32c87727a1c7				
Message -				
Execution details				
View in CodeBuild 🛛				

Done

 로그가 있는 작업의 작업 로그를 보려면 작업에 대한 세부 정보 보기를 선택한 다음 로그 탭을 선 택합니다.

Summary	Logs	Configuration		
Succeeded			Start time: 3 minutes ago	Current phase: COMPLETED
Showing the	e last 51 line	s of the build log. <u>Vie</u>	ew entire log	
			∧ Show previous l	ogs
1 [Contai 2 [Contai 3 [Contai 5 [Contai 6 [Contai 7 [Contai 8 [Contai 9 [Contai 10 [Contai 11 Install 12 13 [Contai 14 co 15 inst	ner] 2024/0 ner] 2024/0 ner] 2024/0 ner] 2024/0 ner] 2024/0 ner] 2024/0 ner] 2024/0 ner] 2024/0 ner] 2024/0 ing Node.js ner] 2024/0 pying : nod alled : v12	1/10 19:23:33.8421 1/10 19:23:34.0434 1/10 19:23:35.2327 1/10 19:23:35.2339 1/10 19:23:35.2345 1/10 19:23:35.2345 1/10 19:23:35.2346 1/10 19:23:35.2364 1/10 19:23:35.4352 1/10 19:23:35.4352 1/10 19:23:36.8936 2 version 12 1/10 19:23:36.8980 e/12.22.12 (with npm 6	20 Waiting for agent ping 95 Waiting for DOWNLOAD_SOURC 26 Phase is DOWNLOAD_SOURCE 79 CODEBUILD_SRC_DIR=/codebuil 39 YAML location is /codebuil 56 No commands found for phas 08 Setting HTTP client timeou 91 Processing environment var 10 Selecting 'nodejs' runtime 84 Running command echo "Inst 49 Running command n \$NODE_12 .14.16)	E ld/output/src180370599/src d/readonly/buildspec.yml e name: install t to higher timeout for S3 source iables e version '12' based on manual selections alling Node.js version 12"
17 [Contai 18 [Contai 19 [Contai 22850a8 20 [Contai 21 [Contai 22 [Contai 23 [Contai	ner] 2024/0 ner] 2024/0 ner] 2024/0 7b0f4 ner] 2024/0 ner] 2024/0 ner] 2024/0 ner] 2024/0	1/10 19:24:09.7533 1/10 19:24:09.7548 1/10 19:24:09.7916 1/10 19:24:09.8222 1/10 19:24:09.8226 1/10 19:24:09.8227 1/10 19:24:09.8227	46 Moving to directory /codeb 65 Unable to initialize cache 97 Configuring ssm agent with 49 Successfully updated ssm a 69 Registering with agent 16 Phases found in YAML: 2 23 INSTALL: 0 commands	uild/output/src180370599/src download: no paths specified to be cached target id: codebuild:f79dc603-3eb0-48ff-970e- gent configuration
24 [Contai	ner] 2024/0	1/10 19:24:09.8227	27 PRE_BUILD: 2 commands	

Done

5. 작업에 대한 구성 세부 정보를 보려면 구성 탭을 선택합니다.

Action execution o	letails		×
Action name: Build Sta	atus: Succeeded		
Summary Logs	s Configuration		
Variable namespace	BuildVariables		
Input artifact	SourceArtifact		
Output artifact	BuildArtifact		
ProjectName	cb-porject		
			Done

파이프라인 ARN 및 서비스 역할 ARN 보기(콘솔)

콘솔을 사용하여 파이프라인 ARN, 서비스 역할 ARN, 파이프라인 아티팩트 스토어와 같은 파이프라인 설정을 볼 수 있습니다.

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home CodePipeline 콘솔을 엽니다.

AWS 계정과 연결된 모든 파이프라인의 이름이 표시됩니다.

- 파이프라인 이름을 선택한 다음 왼쪽 탐색 창에서 설정을 선택합니다. 페이지에서는 다음을 확인 할 수 있습니다.
 - 파이프라인 이름
 - 파이프라인 Amazon 리소스 이름(ARN)

파이프라인 ARN은 다음 형식으로 구성됩니다.

arn:aws:codepipeline:region:account:pipeline-name

파이프라인 ARN 샘플:

arn:aws:codepipeline:us-east-2:80398EXAMPLE:MyFirstPipeline

- 파이프라인의 CodePipeline 서비스 역할 ARN
- 파이프라인 버전
- 파이프라인의 아티팩트 스토어 이름 및 위치

파이프라인 세부 정보 및 이력 보기(CLI)

다음 명령을 실행하면 파이프라인 및 파이프라인 실행에 대한 세부 정보를 볼 수 있습니다.

- list-pipelines AWS 계정과 연결된 모든 파이프라인의 요약을 보는 명령입니다.
- get-pipeline 명령을 실행하면 단일 파이프라인의 세부 정보를 검토할 수 있습니다.
- list-pipeline-executions를 실행하면 파이프라인의 가장 최근 실행 내역에 대한 요약 정보가 표시됩니다.
- get-pipeline-execution을 실행하면 아티팩트의 세부 정보, 파이프라인 실행 ID, 파이프라인의 이름과 버전 및 상태 등 파이프라인 실행에 대한 정보가 표시됩니다.
- get-pipeline-state 명령으로 파이프라인, 스테이지 및 작업 상태를 봅니다.
- list-action-executions로 파이프라인 실행 세부 정보를 봅니다.
- 1. 터미널(Linux, macOS 또는 Unix) 또는 명령 프롬프트(Windows)를 열고 AWS CLI 를 사용하여 list-pipelines 명령을 실행합니다.

aws codepipeline list-pipelines

이 명령은 사용자의 AWS 계정에 연결된 모든 파이프라인의 목록을 반환합니다.

 파이프라인에 대한 세부 정보를 보려면 파이프라인의 고유 이름을 지정한 채 <u>get-pipeline</u> 명령을 실행합니다. 예를 들어, *MyFirstPipeline*이라는 파이프라인의 세부 정보를 보려면 다음을 입 력합니다.

aws codepipeline get-pipeline --name MyFirstPipeline

이 명령은 파이프라인의 구조를 반환합니다.

실행 기록에서 스테이지 조건에 대한 규칙 결과 보기

스테이지 조건이 규칙을 실행하고 롤백 또는 실패와 같은 스테이지 결과를 포함한 실행의 규칙 결과를 볼 수 있습니다.

조건 및 규칙의 유효한 상태 값은 다음과 같습니다. InProgress | Failed | Errored | Succeeded | Cancelled | Abandoned | Overridden

실행 기록에서 스테이지 조건에 대한 규칙 결과 보기(콘솔)

스테이지 조건이 규칙을 실행하고 스테이지 결과를 포함한 실행의 규칙 결과를 콘솔을 사용하여 볼 수 있습니다.

스테이지 조건에 대한 규칙 결과 보기(콘솔)

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home CodePipeline 콘솔을 엽니다.

사용자의 AWS 계정 에 연결된 모든 파이프라인의 이름과 상태가 표시됩니다.

- 2. 이름에서 확인하려는 파이프라인의 이름을 선택합니다.
- 기록을 선택한 다음 실행을 선택합니다. 기록 페이지에서 타임라인 탭을 선택합니다. 규칙에서 실 행에 대한 규칙 결과를 확인합니다.

🗗 e1a7e739-t211-420e-aet9-ta7837666968



list-rule-executions(CLI)로 스테이지 조건에 대한 규칙 결과 보기

스테이지 조건이 규칙을 실행하고 스테이지 결과를 포함한 실행의 규칙 결과를 CLI를 사용하여 볼 수 있습니다.

• 터미널(Linux, macOS, Unix) 또는 명령 프롬프트(Windows)를 열고 AWS CLI 를 사용하여 *MyPipeline*이라는 파이프라인에 대해 list-rule-executions 명령을 실행합니다.

```
aws codepipeline list-rule-executions --pipeline-name MyFirstPipeline
```

이 명령은 파이프라인과 연결되어 있는 완료된 모든 규칙 실행의 목록을 반환합니다.

다음 예제는 규칙의 이름이 *MyMonitorRule*인 스테이지 조건이 있는 파이프라인에 대해 반환된 데이터를 보여줍니다.

```
{
    "ruleExecutionDetails": [
        {
            "pipelineExecutionId": "e1a7e739-f211-420e-aef9-fa7837666968",
            "ruleExecutionId": "3aafc0c7-0e1c-44f1-b357-d1b16a28e483",
            "pipelineVersion": 9,
            "stageName": "Deploy",
            "ruleName": "MyMonitorRule",
            "startTime": "2024-07-29T15:55:01.271000+00:00",
            "lastUpdateTime": "2024-07-29T15:56:08.682000+00:00",
            "status": "Succeeded",
            "input": {
                "ruleTypeId": {
                    "category": "Rule",
                    "owner": "AWS",
                    "provider": "CloudWatchAlarm",
                    "version": "1"
                },
                "configuration": {
                    "AlarmName": "CWAlarm",
                    "WaitTime": "1"
                },
                "resolvedConfiguration": {
                    "AlarmName": "CWAlarm",
                    "WaitTime": "1"
                },
                "region": "us-east-1",
                "inputArtifacts": []
            },
            "output": {
                "executionResult": {
                    "externalExecutionSummary": "Succeeded with alarm 'CWAlarm'
being i
n an 'OK' state."
                }
```

}

}

CodePipeline에서 파이프라인 삭제

언제든지 파이프라인을 편집하여 기능을 변경할 수 있지만 대신에 파이프라인을 삭제할 것인지 결정 할 수 있습니다. 에서 AWS CodePipeline 콘솔 또는 delete-pipeline 명령을 사용하여 파이프라인 AWS CLI 을 삭제할 수 있습니다.

주제

- 파이프라인 삭제(콘솔)
- 파이프라인 삭제(CLI)

파이프라인 삭제(콘솔)

파이프라인 삭제

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https//https//https://https//ht

AWS 계정과 연결된 모든 파이프라인의 이름과 상태가 표시됩니다.

- 2. [Name]에서 삭제할 파이프라인의 이름을 선택합니다.
- 3. 파이프라인 세부 정보 페이지에서 [Edit]를 선택합니다.
- 4. [편집] 페이지에서 [삭제]를 선택합니다.
- 5. 필드에 delete를 입력하여 확인한 후, 삭제를 선택합니다.

▲ Important 이 작업은 실행을 취소할 수 없습니다.

파이프라인 삭제(CLI)

AWS CLI 를 사용하여 파이프라인을 수동으로 삭제하려면 delete-pipeline 명령을 사용합니다.

▲ Important

파이프라인 삭제 작업은 되돌릴 수 없습니다. 확인 대화 상자가 없습니다. 명령이 실행된 후 파 이프라인이 삭제되지만 파이프라인에서 사용된 리소스는 삭제되지 않습니다. 이를 통해 해당 리소스를 사용하여 소프트웨어의 릴리스를 자동화하는 새 파이프라인을 보다 쉽게 생성할 수 있습니다.

파이프라인 삭제

 터미널(Linux, macOS 또는 Unix) 또는 명령 프롬프트(Windows)를 열고 AWS CLI 를 사용 하여 명령을 실행하고 삭제할 파이프라인의 이름을 delete-pipeline 지정합니다. 예를 들어, *MyFirstPipeline*이라는 파이프라인을 삭제하려면 다음을 입력합니다.

aws codepipeline delete-pipeline --name MyFirstPipeline

이 명령은 아무 것도 반환하지 않습니다.

2. 더 이상 필요하지 않은 리소스는 모두 삭제합니다.

Note

파이프라인을 삭제한다고 해서 코드를 배포하는 데 사용한 CodeDeploy 또는 Elastic Beanstalk 애플리케이션과 같은 파이프라인에 사용된 리소스(CodePipeline 콘솔로부터 파이프라인을 생성했다면 파이프라인의 아티팩트를 저장하기 위해 생성한 Amazon S3 버 킷 CodePipeline)가 삭제되지는 않습니다. 더 이상 필요하지 않은 리소스는 삭제해야만 차 후에 이에 대한 요금이 발생하지 않습니다. 예를 들어, 콘솔을 이용해 처음으로 파이프라 인을 만드는 경우, CodePipeline이 하나의 Amazon S3 버킷을 만들어 모든 파이프라인의 모든 아티팩트를 저장합니다. 파이프라인을 모두 삭제했다면 <u>버킷 삭제</u>에 있는 단계를 따 르십시오.

CodePipeline에서 다른 AWS 계정의 리소스를 사용하는 파이프라인 생성

다른 AWS 계정에서 생성되거나 관리된 리소스를 사용하는 파이프라인을 생성할 수 있습니다. 예를 들 어 파이프라인에 한 계정을 사용하고 CodeDeploy 리소스에 다른 계정을 사용할 수 있습니다.

Note

여러 계정의 작업으로 파이프라인을 생성할 때는 교차 계정 파이프라인의 제한 내에서 여전히 아티팩트에 액세스할 수 있도록 작업을 구성해야 합니다. 교차 계정 작업에는 다음 제한 사항 이 적용됩니다.

- 일반적으로 작업은 다음과 같은 경우에만 아티팩트를 소비할 수 있습니다.
 - 작업이 파이프라인 계정과 동일한 계정에 있거나
 - 아티팩트가 다른 계정의 작업에 대한 파이프라인 계정에서 생성되었거나
 - 아티팩트가 작업과 동일한 계정에서 이전 작업에 의해 생성된 경우

즉, 두 계정 모두 파이프라인 계정이 아닌 경우 한 계정에서 다른 계정으로 아티팩트를 전달 할 수 없습니다.

- 다음 작업 유형에 대해서는 교차 계정 작업이 지원되지 않습니다.
 - Jenkins 빌드 작업

이 예제에서는 사용할 AWS Key Management Service (AWS KMS) 키를 생성하고, 파이프라인에 키를 추가하고, 계정 간 액세스를 활성화하도록 계정 정책 및 역할을 설정해야 합니다. AWS KMS 키의 경우 키 ID, 키 ARN 또는 별칭 ARN을 사용할 수 있습니다.

Note

별칭은 KMS 키를 생성한 계정에서만 인식됩니다. 교차 계정 작업의 경우 키 ID 또는 키 ARN만 사용하여 키를 식별할 수 있습니다. 계정 간 작업에는 다른 계정(AccountB)의 역할을 사용하는 것이 포함되므로 키 ID를 지정하면 다른 계정(AccountB)의 키가 사용됩니다.

이 연습과 해당 예에서 *AccountA*는 파이프라인을 생성하는 데 원래 사용된 계정입니다. 파이프라인 아티팩트를 저장하는 데 사용되는 Amazon S3 버킷과에서 사용하는 서비스 역할에 액세스할 수 있습 니다 AWS CodePipeline. *AccountB*는 CodeDeploy에서 사용된 CodeDeploy 애플리케이션, 배포 그 룹, 서비스 역할을 생성하는 데 원래 사용된 계정입니다.

AccountA가 AccountB에서 만든 CodeDeploy 애플리케이션을 사용하도록 파이프라인을 편집하려 면 AccountA는 다음을 수행해야 합니다.

• AccountB의 ARN 또는 계정 ID를 요청합니다(이 연습에서 AccountB ID는 012ID_ACCOUNT_B임).

- 파이프라인의 리전에서 AWS KMS 고객 관리형 키를 생성하거나 사용하고 서비스 역할 (CodePipeline_Service_Role) 및 AccountB에 해당 키를 사용할 수 있는 권한을 부여합니다.
- *AccountB*에 Amazon S3 버킷에 대한 액세스 권한을 부여하는 Amazon S3 버킷 정책을 생성합니다 (예: *codepipeline-us-east-2-1234567890*).
- AccountA가 AccountB에서 구성된 역할을 맡을 수 있도록 하는 정책을 생성하고 해당 정책을 서비 스 역할(CodePipeline_Service_Role)에 연결합니다.
- 파이프라인을 편집하여 기본 AWS KMS 키 대신 고객 관리형 키를 사용합니다.

AccountB가 AccountA에서 생성된 파이프라인의 리소스에 액세스할 수 있도록 하려면 AccountB는 다음 작업을 수행해야 합니다.

- AccountA의 ARN 또는 계정 ID를 요청합니다(이 연습에서 AccountA ID는 012ID_ACCOUNT_A임).
- Amazon S3 버킷(*codepipeline-us-east-2-1234567890*)에 대한 액세스를 허용하는 CodeDeploy용으로 구성된 Amazon EC2 인스턴스 역할에 적용되는 정책을 생성합니다.
- AccountA에서 파이프라인 아티팩트를 암호화하는 데 사용되는 AWS KMS 고객 관리형 키에 대한 액세스를 허용하는 CodeDeploy용으로 구성된 <u>Amazon EC2 인스턴스 역할에</u> 적용되는 정책을 생성 합니다.
- AccountA의 CodePipeline 서비스 역할이 역할을 맡도록 허용하는 신뢰 관계 정책을 사용하여 IAM 역할(CrossAccount_Role)을 구성하고 연결합니다.
- 파이프라인에 필요한 배포 리소스에 액세스할 수 있는 정책을 생성하고 *CrossAccount_Role*에 연 결합니다.
- Amazon S3 버킷(*codepipeline-us-east-2-1234567890*)에 액세스할 수 있는 정책을 생성하 고 *CrossAccount_Role*에 연결합니다.

주제

- <u>필수 조건: AWS KMS 암호화 키 생성</u>
- 1단계:계정 정책 및 역할 설정
- 2단계: 파이프라인 편집

필수 조건: AWS KMS 암호화 키 생성

고객 관리형 키는 모든 AWS KMS 키와 마찬가지로 리전에 고유합니다. 파이프라인이 생성된 리전과 동일한 리전에서 고객 관리형 AWS KMS 키를 생성해야 합니다(예: us-east-2). 에서 고객 관리형 키를 생성하려면 AWS KMS

- 1. AccountA를 AWS Management Console 사용하여에 로그인하고 콘솔을 AWS KMS 엽니다.
- 2. 왼쪽에서 고객 관리형 키를 선택합니다.
- 3. 키 생성을 선택합니다. 구성 키에서 대칭 기본값을 선택한 상태로 두고 다음을 선택합니다.
- 별칭에 이 키에 사용할 별칭을 입력합니다(예: PipelineName-Key). 필요한 경우 이 키에 대한 설명과 태그를 입력하고 다음을 선택합니다.
- 5. 키 관리 권한 정의에서 이 키의 관리자 역할을 수행할 역할을 선택하고 다음을 선택합니다.
- 6. 키 사용 권한 정의의 이 계정에서 파이프라인의 서비스 역할 이름을 선택합니다(예: CodePipeline_Service_Role). 기타 AWS 계정에서 다른 AWS 계정 추가를 선택합니다. AccountB에 대한 계정 ID를 입력하여 ARN을 완료하고 다음을 선택합니다.
- 7. Review and edit key policy(키 정책 검토 및 편집)에서 정책을 검토한 다음 완료를 선택합니다.
- 8. 키 목록에서 키의 별칭을 선택하고 ARN을 복사합니다(예: arn:aws:kms:useast-2:012ID_ACCOUNT_A:key/222222-3333333-4444-556677EXAMPLE). 파이프라인 을 편집하고 정책을 구성할 때 이 정보가 필요합니다.

1단계: 계정 정책 및 역할 설정

AWS KMS 키를 생성한 후에는 교차 계정 액세스를 활성화하는 정책을 생성하고 연결해야 합니다. 이 경우 AccountA 및 AccountB 둘 다의 작업이 필요합니다.

주제

- 파이프라인을 생성하는 계정에 정책 및 역할 구성(AccountA)
- AWS 리소스를 소유한 계정에서 정책 및 역할 구성(AccountB)

파이프라인을 생성하는 계정에 정책 및 역할 구성(AccountA)

다른 AWS 계정과 연결된 CodeDeploy 리소스를 사용하는 파이프라인을 생성하려면 AccountA는 아 티팩트를 저장하는 데 사용되는 Amazon S3 버킷과 CodePipeline의 서비스 역할 모두에 대한 정책을 구성해야 합니다.

AccountB에 대한 액세스 권한을 부여하는 정책을 Amazon S3 버킷에 대해 만들려면(콘솔)

1. AccountA를 AWS Management Console 사용하여에 로그인하고 Amazon S3 콘솔을 <u>https://</u> <u>console.aws.amazon.com/s3/</u>://https://https://https://https://i/ttps://i/ttps://i/ttps://i// https://://://i/ttps://://

- Amazon S3 버킷 목록에서 파이프라인의 아티팩트가 저장될 Amazon S3 버킷을 선택합니다. 이 버킷의 이름은 이며codepipeline-*region-1234567EXAMPLE*, 여기서 *region*은 파이프라인 을 생성한 AWS 리전이고 1234567EXAMPLE은 버킷 이름이 고유하도록 보장하는 10자리 난수입 니다(예: codepipeline-us-east-2-1234567890).
- 3. Amazon S3 버킷의 세부 정보 페이지에서 속성을 선택합니다.
- 4. 속성 창에서 [Permissions]를 확장한 다음, [Add bucket policy]를 선택합니다.

Note

정책이 Amazon S3 버킷에 이미 연결되어 있는 경우 버킷 정책 편집을 선택합니다. 그러면 기존 정책에 다음 예의 설명문을 추가할 수 있습니다. 새 정책을 추가하려면 링크를 선택 하고 AWS 정책 생성기의 지침을 따릅니다. 자세한 정보는 IAM 정책 개요를 참조하세요.

5. [Bucket Policy Editor] 창에서 다음 정책을 입력합니다. 이렇게 하면 AccountB가 파이프라인 아 티팩트에 액세스할 수 있으며, 사용자 지정 소스나 빌드 작업과 같은 작업에서 출력 아티팩트를 생 성하는 경우 AccountB가 출력 아티팩트를 추가할 수 있습니다.

다음 예에서 *AccountB*에 대한 ARN은 *012ID_ACCOUNT_B*입니다. Amazon S3 버킷의 ARN은 *codepipeline-us-east-2-1234567890*입니다. 이들 ARN을 액세스를 허용하려는 계정에 대한 ARN과 해당 Amazon S3 버킷에 대한 ARN으로 바꿉니다.

```
{
 "Version": "2012-10-17",
 "Id": "SSEAndSSLPolicy",
"Statement": [
{
 "Sid": "DenyUnEncryptedObjectUploads",
  "Effect": "Deny",
  "Principal": "*",
  "Action": "s3:PutObject",
  "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*",
 "Condition": {
 "StringNotEquals": {
 "s3:x-amz-server-side-encryption": "aws:kms"
}
  }
},
ł
  "Sid": "DenyInsecureConnections",
  "Effect": "Deny",
```

```
"Principal": "*",
   "Action": "s3:*",
   "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*",
   "Condition": {
  "Bool": {
     "aws:SecureTransport": false
  }
  }
   },
 {
   "Sid": "",
   "Effect": "Allow",
   "Principal": {
  "AWS": "arn:aws:iam::012ID_ACCOUNT_B:root"
  },
   "Action": [
            "s3:Get*",
            "s3:Put*"
        ],
   "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*"
 },
 {
   "Sid": "",
   "Effect": "Allow",
   "Principal": {
       "AWS": "arn:aws:iam::012ID_ACCOUNT_B:root"
   },
   "Action": "s3:ListBucket",
   "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890"
 }
   ]
}
```

- 6. [Save]를 선택한 후 정책 편집기를 닫습니다.
- 7. 저장을 선택하여 Amazon S3 버킷에 대한 권한을 저장합니다.

CodePipeline용 서비스 역할에 대한 정책을 생성하려면(콘솔)

- 2. 탐색 창에서 Roles를 선택합니다.

- 3. 역할 목록의 역할 이름에서 CodePipeline에 대한 서비스 역할 이름을 선택합니다.
- 4. 권한 탭에서 인라인 정책 추가(Add inline policy)를 선택합니다.
- 5. JSON 탭을 선택하고 다음 정책을 입력하여 *AccountB*가 역할을 맡을 수 있도록 허용합니다. 다 음 예에서 *012ID_ACCOUNT_B*는 *AccountB*에 대한 ARN입니다.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "sts:AssumeRole",
        "Resource": [
            "arn:aws:iam::012ID_ACCOUNT_B:role/*"
        ]
    }
}
```

- 6. 정책 검토를 선택합니다.
- 7. 이름에 이 정책의 이름을 입력합니다. 정책 생성을 선택합니다.

AWS 리소스를 소유한 계정에서 정책 및 역할 구성(AccountB)

CodeDeploy에서 애플리케이션, 배포 및 배포 그룹을 생성할 때 <u>Amazon EC2 인스턴스 역할</u>도 생성합 니다. (배포 연습 실행 마법사를 사용하는 경우 이 역할이 생성되지만 수동으로 생성할 수도 있습니다.) *AccountA*에서 생성한 파이프라인이 *AccountB*에서 생성된 CodeDeploy 리소스를 사용하려면 다음 을 수행해야 합니다.

- 파이프라인 아티팩트가 저장된 Amazon S3 버킷에 액세스할 수 있는 인스턴스 역할에 대한 정책을 구성합니다.
- 교차 계정 액세스에 대해 구성된 AccountB에서 두 번째 역할을 생성합니다.

이 두 번째 역할은 *AccountA*의 Amazon S3 버킷에 액세스할 수 있을 뿐만 아니라 CodeDeploy 리 소스에 대한 액세스를 허용하는 정책과 *AccountA*의 CodePipeline 서비스 역할이 역할을 맡도록 허 용하는 신뢰 관계 정책도 포함해야 합니다. Note

이러한 정책은 다른 AWS 계정을 사용하여 생성된 파이프라인에서 사용할 CodeDeploy 리 소스를 설정하는 데에만 적용됩니다. 다른 AWS 리소스에는 리소스 요구 사항과 관련된 정 책이 필요합니다.

CodeDeploy용으로 구성된 Amazon EC2 인스턴스 역할에 대한 정책을 만들려면(콘솔)

- 1. AccountB를 AWS Management Console 사용하여에 로그인하고 <u>https://</u> <u>console.aws.amazon.com/iam/</u>://https://https://i//https://:///https://:////https://:////https://:////
- 2. 탐색 창에서 Roles를 선택합니다.
- 역할 목록의 역할 이름에서 CodeDeploy 애플리케이션에 대한 Amazon EC2 인스턴스 역할로 사용된 서비스 역할 이름을 선택합니다. 이 역할 이름은 다를 수 있으며, 하나의 배포 그룹에서 둘 이상의 인스턴스 역할을 사용할 수 있습니다. 자세한 내용은 <u>Amazon EC2 인스턴스용 IAM 인스턴스</u> 프로파일 생성을 참조하세요.
- 4. 권한 탭에서 인라인 정책 추가(Add inline policy)를 선택합니다.
- JSON 탭을 선택하고 다음 정책을 입력하여 AccountA가 파이프라인용 아티팩트를 저장하는 데 사용하는 Amazon S3 버킷에 대한 액세스 권한을 부여합니다(이 예에서는 codepipeline-useast-2-1234567890).

```
"Version": "2012-10-17",
"Statement": [
{
    "Effect": "Allow",
    "Action": [
        "s3:Get*"
    ],
    "Resource": [
        "arn:aws:s3:::codepipeline-us-east-2-1234567890/*"
    ]
    },
    {
    "Effect": "Allow",
    "Action": [
        "s3:ListBucket"
```

{

```
],
    "Resource": [
    "arn:aws:s3:::codepipeline-us-east-2-1234567890"
    ]
    }
]
```

- 6. 정책 검토를 선택합니다.
- 7. 이름에 이 정책의 이름을 입력합니다. 정책 생성을 선택합니다.
- 8. 에 대한 두 번째 정책을 생성합니다. AWS KMS 여기서 arn: aws: kms: us-

east-1:012ID_ACCOUNT_A:key/222222-3333333-4444-556677EXAMPLE는 AccountA에서 생성되고 AccountB가 이를 사용할 수 있도록 구성된 고객 관리형 키의 ARN입니 다.

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": [
           "kms:DescribeKey",
           "kms:GenerateDataKey*",
           "kms:Encrypt",
           "kms:ReEncrypt*",
           "kms:Decrypt"
          ],
        "Resource": [
           "arn:aws:kms:us-
east-1:012ID_ACCOUNT_A:key/222222-3333333-4444-556677EXAMPLE"
          ٦
      }
   ]
}
```

A Important

여기에 표시된 대로이 정책에서 *AccountA*의 계정 ID를 AWS KMS 키에 대한 리소스 ARN의 일부로 사용해야 합니다. 그렇지 않으면 정책이 작동하지 않습니다.

9. 정책 검토를 선택합니다.

10. 이름에 이 정책의 이름을 입력합니다. 정책 생성을 선택합니다.

이제 교차 계정 액세스에 사용할 IAM 역할을 생성하고 구성하여 *AccountA*의 CodePipeline 서비스 역할이 역할을 맡을 수 있도록 합니다. 이 역할에는 *AccountA*에 아티팩트를 저장하는 데 사용되는 CodeDeploy 리소스 및 Amazon S3 버킷에 대한 액세스를 허용하는 정책이 포함되어야 합니다.

IAM에서 교차 계정 역할을 구성하려면

- 2. 탐색 창에서 Roles를 선택합니다. Create role(역할 생성)을 선택합니다.
- 3. Select type of trusted entity(신뢰할 수 있는 엔터티 유형 선택) 아래에서 다른 Another AWS account (AWS 계정)를 선택합니다. 이 역할을 사용할 수 있는 계정 지정의 계정 ID에 CodePipeline(*AccountA*)에서 파이프라인을 생성할 계정의 계정 AWS ID를 입력한 후 다음: 권한 을 선택합니다.

A Important

이 단계는 *AccountB*와 *AccountA* 사이에 신뢰 관계 정책을 만듭니다. 그러나 이렇 게 하면 계정에 루트 수준의 액세스 권한이 부여되므로, CodePipeline은 *AccountA*의 CodePipeline 서비스 역할까지 범위를 축소할 것을 권장합니다. 16단계에 따라 권한을 제 한합니다.

4. 권한 정책 연결에서 AmazonS3ReadOnlyAccess를 선택한 후 다음: 태그를 선택합니다.

이 정책은 사용할 정책이 아닙니다. 마법사를 완료하려면 정책을 선택해야 합니다.

- 다음: 검토를 선택합니다. 역할 이름에 이 역할의 이름을 입력합니다(예: CrossAccount_Role).
 역할의 이름은 IAM의 명명 규칙을 따르는 것이라면 어떤 것이든 가능합니다. 역할의 목적을 명확 하게 나타내는 이름으로 지정하는 것을 고려하십시오. 역할 생성을 선택합니다.
- 6. 역할 목록에서, 방금 만든 역할(예: *CrossAccount_Role*)을 선택하여 해당 역할에 대한 요약 페 이지를 엽니다.
- 7. 권한 탭에서 인라인 정책 추가(Add inline policy)를 선택합니다.
- 8. JSON 탭을 선택하고 다음 정책을 입력하여 CodeDeploy 리소스에 대한 액세스를 허용합니다.

Note



- 9. 정책 검토를 선택합니다.
- 10. 이름에 이 정책의 이름을 입력합니다. 정책 생성을 선택합니다.
- 11. 권한 탭에서 인라인 정책 추가(Add inline policy)를 선택합니다.
- 12. JSON 탭을 선택하고 다음 정책을 입력하여 이 역할이 *AccountA*의 Amazon S3 버킷에서 입력 아 티팩트를 검색하고 출력 아티팩트를 넣을 수 있도록 허용합니다.

```
{
   "Version": "2012-10-17",
   "Statement": [
     {
       "Effect": "Allow",
       "Action": [
         "s3:GetObject*",
         "s3:PutObject",
         "s3:PutObjectAcl"
       ],
       "Resource": [
         "arn:aws:s3:::codepipeline-us-east-2-1234567890/*"
       ]
     }
   ]
}
```

13. 정책 검토를 선택합니다.

- 14. 이름에 이 정책의 이름을 입력합니다. 정책 생성을 선택합니다.
- 15. 권한 탭의 정책 이름 아래에 있는 정책 목록에서 AmazonS3ReadOnlyAccess를 찾아 정책 옆에 있는 삭제 아이콘(X)을 선택합니다. 확인 메시지가 나타나면 [Detach]를 선택합니다.
- 16. 신뢰 관계 탭을 선택한 후 신뢰 정책 편집을 선택합니다. 왼쪽 열에서 보안 주체 추가 옵션을 선택 합니다. 보안 주체 유형에서 IAM 역할을 선택한 다음 AccountA의 CodePipeline 서비스 역할에 대해 ARN을 제공합니다. AWS 보안 주체 목록에서 arn:aws:iam::Account_A:root를 제거 한 다음, 정책 업데이트를 선택합니다.

2단계: 파이프라인 편집

CodePipeline 콘솔을 사용하여 다른 AWS 계정과 연결된 리소스를 사용하는 파이프라인을 생성하거나 편집할 수 없습니다. 그러나 콘솔을 사용하여 파이프라인의 일반 구조를 생성한 다음를 사용하여 파이 프라인 AWS CLI 을 편집하고 해당 리소스를 추가할 수 있습니다. 또는 기존 파이프라인의 구조를 사용 하고 리소스를 수동으로 추가할 수 있습니다.

다른 AWS 계정과 연결된 리소스를 추가하려면(AWS CLI)

 터미널(Linux, macOS 또는 Unix) 또는 명령 프롬프트(Windows)에서, 리소스를 추가하려는 파 이프라인에 get-pipeline 명령을 실행합니다. JSON 파일에 명령 출력을 복사합니다. 예를 들어 MyFirstPipeline이라는 파이프라인의 경우 다음과 유사한 명령을 입력합니다.

aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json

출력은 pipeline.json 파일로 전송됩니다.

 일반 텍스트 편집기에서 JSON 파일을 엽니다. 아티팩트 스토어의 "type": "S3" 이후에, KMS encryptionKey, ID 및 유형 정보를 추가합니다. 여기서, codepipeline-us-east-2-1234567890은 파이프라인에 대한 아티팩 트를 저장하는 데 사용된 Amazon S3 버킷의 이름이며, arn:aws:kms:useast-1:012ID_ACCOUNT_A:key/222222-3333333-4444-556677EXAMPLE은 방금 생성 한 고객 관리형 키의 ARN입니다.

```
{
    "artifactStore": {
        "location": "codepipeline-us-east-2-1234567890",
        "type": "S3",
        "encryptionKey": {
        "
```

```
"id": "arn:aws:kms:us-
east-1:012ID_ACCOUNT_A:key/222222-3333333-4444-556677EXAMPLE",
    "type": "KMS"
    }
},
```

 생성한 교차 계정 역할(CrossAccount_Role)의 roleArn 값을 포함하여 AccountB와 연결된 CodeDeploy 리소스를 사용하기 위한 배포 작업을 단계에 추가합니다.

다음은 *ExternalDeploy*라는 배포 작업을 추가하는 JSON을 보여 주는 예입니다. *Staging*이 라는 단계에서 *AccountB*에서 만든 CodeDeploy 리소스를 사용합니다. 다음 예에서 *AccountB*에 대한 ARN은 *012ID_ACCOUNT_B*입니다.

```
{
               "name": "Staging",
               "actions": [
                   {
                        "inputArtifacts": [
                            {
                                "name": "MyAppBuild"
                            }
                        1,
                        "name": "ExternalDeploy",
                        "actionTypeId": {
                            "category": "Deploy",
                            "owner": "AWS",
                            "version": "1",
                            "provider": "CodeDeploy"
                       },
                        "outputArtifacts": [],
                        "configuration": {
                            "ApplicationName": "AccountBApplicationName",
                            "DeploymentGroupName": "AccountBApplicationGroupName"
                       },
                        "runOrder": 1,
                        "roleArn":
"arn:aws:iam::012ID_ACCOUNT_B:role/CrossAccount_Role"
                   }
               ]
           }
```

Note

이것은 전체 파이프라인의 JSON이 아닌, 한 단계의 작업 구조입니다.

4. update-pipeline 명령이 JSON 파일을 사용할 수 있도록 하려면 이 파일에서 metadata 라인을 삭 제해야 합니다. JSON 파일의 파이프라인 구조에서 단원("metadata": { } 행과 "created", "pipelineARN" 및 "updated" 필드)을 삭제합니다.

예를 들어, 구조에서 다음 행을 삭제합니다.

```
"metadata": {
    "pipelineArn": "arn:aws:codepipeline:region:account-ID:pipeline-name",
    "created": "date",
    "updated": "date"
}
```

파일을 저장합니다.

5. 변경 사항을 적용하려면 다음과 유사하게 파이프라인 JSON 파일을 지정하여 update-pipeline 명 령을 실행합니다.

A Important

파일 이름 앞에 file://를 포함해야 합니다. 이 명령에 필수적입니다.

aws codepipeline update-pipeline --cli-input-json file://pipeline.json

이 명령은 편집한 파이프라인의 전체 구조를 반환합니다.

다른 AWS 계정과 연결된 리소스를 사용하는 파이프라인을 테스트하려면

1. 터미널(Linux, macOS 또는 Unix) 또는 명령 프롬프트(Windows)에서, 다음과 유사하게, 파이프라 인의 이름을 지정한 채 start-pipeline-execution 명령을 실행합니다.

aws codepipeline start-pipeline-execution --name MyFirstPipeline

자세한 내용은 <u>수동으로 파이프라인 시작</u> 단원을 참조하십시오.

2. AccountA를 AWS Management Console 사용하여에 로그인하고 <u>http://</u> <u>console.aws.amazon.com/codesuite/codepipeline/home</u>://https://https://https://:// https://://www.com.com.com.com.com에서 CodePipeline 콘솔을 엽니다.

AWS 계정과 연결된 모든 파이프라인의 이름이 표시됩니다.

- [Name]에서 방금 편집한 파이프라인의 이름을 선택합니다. 이렇게 하면 파이프라인의 각 단계의 각 작업 상태를 포함하여 파이프라인의 세부 정보 보기가 열립니다.
- 파이프라인을 통해 진행 상황을 관찰합니다. 다른 AWS 계정과 연결된 리소스를 사용하는 작업에 대한 성공 메시지를 기다립니다.

Note

AccountA로 로그인한 상태에서 작업의 세부 정보를 보려고 시도하는 경우 오류가 발생 합니다. 로그아웃한 다음 AccountB로 로그인하여 CodeDeploy의 배포 세부 정보를 봅니 다.

이벤트 기반 변경 감지를 사용하도록 폴링 파이프라인 마이그레이션

AWS CodePipeline 는 코드 변경이 있을 때마다 파이프라인을 시작하는 것을 포함하여 전체 end-toend 지속적 전달을 지원합니다. 코드 변경 시 파이프라인을 시작하는 방법은 이벤트 기반 변경 감지 및 폴링 두 가지가 있습니다. 파이프라인에는 이벤트 기반 변경 감지를 사용하는 것이 좋습니다.

여기에 포함된 절차를 사용하여 폴링 파이프라인을 파이프라인의 이벤트 기반 변경 감지 방법으로 마이그레이션(업데이트)하세요.

파이프라인에 권장되는 이벤트 기반 변경 감지 방법은 CodeCommit과 같은 파이프라인 소스에 의해 결정됩니다. 예를 들어, 이 경우 폴링 파이프라인은 EventBridge를 통한 이벤트 기반 변경 감지로 마이 그레이션해야 합니다.

폴링 파이프라인을 마이그레이션하는 방법

폴링 파이프라인을 마이그레이션하려면 폴링 파이프라인을 결정한 다음 권장되는 이벤트 기반 변경 감지 방법을 결정하세요.

• 계정의 폴링 파이프라인 보기의 단계를 사용하여 폴링 파이프라인을 확인하세요.

• 표에서 파이프라인 소스 유형을 찾은 다음 폴링 파이프라인을 마이그레이션하는 데 사용할 구현이 포함된 절차를 선택합니다. 각 섹션에는 CLI 또는 AWS CloudFormation사용과 같은 여러 마이그레 이션 방법이 포함되어 있습니다.

파이프라인을 권장되는 변경 감지 방법으로 마이그레이션하는 방법			
파이프라인 소스	권장되는 이벤트 기반 탐지 방법	마이그레이션 절차	
AWS CodeCommit	EventBridge(권장).	<u>CodeCommit 소스를 사용하여 폴링 파</u> <u>이프라인 마이그레이션</u> 을 참조하세요.	
Amazon S3	EventBridge 및 버킷이 이벤트 알 림을 사용하도록 설정되었습니다 (권장).	<u>이벤트용으로 활성화된 S3 소스를 사</u> <u>용하여 폴링 파이프라인 마이그레이</u> <u>션</u> 을 참조하세요.	
Amazon S3	EventBridge 및 AWS CloudTrail 추적.	<u>S3 소스 및 CloudTrail 트레일을 사용하</u> <u>여 폴링 파이프라인 마이그레이션</u> 을 참 조하세요.	
GitHub(GitHub 앱을 통해)	연결(권장)	<u>GitHub(OAuth 앱을 통해) 소스 작업에</u> 대한 폴링 파이프라인을 연결로 마이그 레이션을 참조하세요.	
GitHub(OAuth 앱 사 용)	Webhook	GitHub(OAuth 앱 사용) 소스 작업에 대 한 폴링 파이프라인을 웹후크로 마이그 레이션을 참조하세요.	

▲ Important

GitHub(viaOAuth) 작업이 있는 파이프라인과 같은 해당 파이프라인 작업 구성 업데이트의 경우, 파이프라인의 폴링을 중지하려면 원본 작업 구성 내에서 PollForSourceChanges 파라미터를 false로 명시적으로 설정해야 합니다. 따라서 EventBridge 규칙을 구성하고 PollForSourceChanges 파라미터를 생략하는 등의 방법으로 이벤트 기반 변경 감지 및 폴 링을 모두 포함하는 파이프라인을 잘못 구성할 수 있습니다. 이 결과 파이프라인 실행이 중복 되고 파이프라인이 기본적으로 이벤트 기반 파이프라인보다 훨씬 낮은 총 폴링 파이프라인 수 의 제한에 개수가 기록됩니다. 자세한 내용은 <u>in AWS CodePipeline 할당량</u> 단원을 참조하십시 오.

계정의 폴링 파이프라인 보기

첫 번째 단계로 다음 스크립트 중 하나를 사용하여 계정의 어떤 파이프라인을 폴링용으로 구성했는지 확인하세요. 다음은 이벤트 기반 변경 감지로 마이그레이션하기 위한 파이프라인입니다.

계정의 폴링 파이프라인 보기(스크립트)

다음 단계에 따라 스크립트를 사용하여 계정에서 폴링을 사용하는 파이프라인을 확인하세요.

- 1. 터미널 창을 열고 다음 중 하나를 수행합니다.
 - 다음 명령을 실행하여 PollingPipelinesExtractor.sh라는 새 스크립트를 생성합니다.

vi PollingPipelinesExtractor.sh

• Python 스크립트를 사용하려면 다음 명령을 실행하여 PollingPipelinesExtractor.py라는 새 Python 스크립트를 생성합니다.

vi PollingPipelinesExtractor.py

- 2. 다음 코드를 복사하여 PollingPipelinesExtractor 스크립트에 붙여 넣습니다. 다음 중 하나를 수행 합니다.
 - 다음 코드를 복사하여 PollingPipelinesExtractor.sh 스크립트에 붙여 넣습니다.

```
#!/bin/bash
set +x
POLLING_PIPELINES=()
LAST_EXECUTED_DATES=()
NEXT_TOKEN=null
HAS_NEXT_TOKEN=true
if [[ $# -eq 0 ]] ; then
        echo 'Please provide region name'
        exit 0
fi
```

```
REGION=$1
while [ "$HAS_NEXT_TOKEN" != "false" ]; do
    if [ "$NEXT_TOKEN" != "null" ];
        then
            LIST_PIPELINES_RESPONSE=$(aws codepipeline list-pipelines --region
 $REGION --next-token $NEXT_TOKEN)
        else
            LIST_PIPELINES_RESPONSE=$(aws codepipeline list-pipelines --region
 $REGION)
    fi
    LIST_PIPELINES=$(jq -r '.pipelines[].name' <<< "$LIST_PIPELINES_RESPONSE")</pre>
    NEXT_TOKEN=$(jg -r '.nextToken' <<< "$LIST_PIPELINES_RESPONSE")</pre>
    if [ "$NEXT_TOKEN" == "null" ];
        then
            HAS_NEXT_TOKEN=false
    fi
    for pipline_name in $LIST_PIPELINES
    do
        PIPELINE=$(aws codepipeline get-pipeline --name $pipline_name --region
 $REGION)
        HAS_POLLABLE_ACTIONS=$(jq '.pipeline.stages[].actions[] |
 select(.actionTypeId.category == "Source") | select(.actionTypeId.owner
 == ("ThirdParty","AWS")) | select(.actionTypeId.provider ==
 ("GitHub","S3","CodeCommit")) | select(.configuration.PollForSourceChanges ==
 ("true",null))' <<< "$PIPELINE")</pre>
        if [ ! -z "$HAS_POLLABLE_ACTIONS" ];
        then
            POLLING_PIPELINES+=("$pipline_name")
            PIPELINE_EXECUTIONS=$(aws codepipeline list-pipeline-executions --
pipeline-name $pipline_name --region $REGION)
            LAST_EXECUTION=$(jq -r '.pipelineExecutionSummaries[0]' <<<</pre>
 "$PIPELINE_EXECUTIONS")
            if [ "$LAST_EXECUTION" != "null" ];
                then
                    LAST_EXECUTED_TIMESTAMP=$(jq -r '.startTime' <<<</pre>
 "$LAST EXECUTION")
                    LAST_EXECUTED_DATE="$(date -r ${LAST_EXECUTED_TIMESTAMP%.*})"
                else
                    LAST_EXECUTED_DATE="Not executed in last year"
            fi
            LAST_EXECUTED_DATES+=("$LAST_EXECUTED_DATE")
```

```
fi
done

done

fileName=$REGION-$(date +%s)
printf "| %-30s | %-30s |\n" "Polling Pipeline Name" "Last Executed Time"
printf "| %-30s | %-30s |\n" "______" "_____""
for i in "${!POLLING_PIPELINES[@]}"; do
    printf "| %-30s | %-30s |\n" "${POLLING_PIPELINES[i]}"
    "${LAST_EXECUTED_DATES[i]}"
    printf "${POLLING_PIPELINES[i]}," >> $fileName.csv
done
printf "\nSaving Polling Pipeline Names to file $fileName.csv."
```

• 다음 코드를 복사하여 PollingPipelinesExtractor.py 스크립트에 붙여 넣습니다.

```
import boto3
import sys
import time
import math
hasNextToken = True
nextToken = ""
pollablePipelines = []
lastExecutedTimes = []
if len(sys.argv) == 1:
    raise Exception("Please provide region name.")
session = boto3.Session(profile_name='default', region_name=sys.argv[1])
codepipeline = session.client('codepipeline')
def is_pollable_action(action):
    actionTypeId = action['actionTypeId']
    configuration = action['configuration']
    return actionTypeId['owner'] in {"AWS", "ThirdParty"}
 and actionTypeId['provider'] in {"GitHub", "CodeCommit",
 "S3"} and ('PollForSourceChanges' not in configuration or
 configuration['PollForSourceChanges'] == 'true')
def has_pollable_actions(pipeline):
    hasPollableAction = False
    pipelineDefinition = codepipeline.get_pipeline(name=pipeline['name'])
['pipeline']
```

```
for action in pipelineDefinition['stages'][0]['actions']:
       hasPollableAction = is_pollable_action(action)
       if hasPollableAction:
           break
   return hasPollableAction
def get_last_executed_time(pipelineName):
pipelineExecutions=codepipeline.list_pipeline_executions(pipelineName=pipelineName)
['pipelineExecutionSummaries']
    if pipelineExecutions:
       return pipelineExecutions[0]['startTime'].strftime("%A %m/%d/%Y, %H:%M:
%S")
   else:
       return "Not executed in last year"
while hasNextToken:
   if nextToken=="":
       list_pipelines_response = codepipeline.list_pipelines()
   else:
       list_pipelines_response =
 codepipeline.list_pipelines(nextToken=nextToken)
    if 'nextToken' in list_pipelines_response:
       nextToken = list_pipelines_response['nextToken']
    else:
       hasNextToken= False
   for pipeline in list_pipelines_response['pipelines']:
       if has_pollable_actions(pipeline):
           pollablePipelines.append(pipeline['name'])
           lastExecutedTimes.append(get_last_executed_time(pipeline['name']))
fileName="{region}-
{timeNow}.csv".format(region=sys.argv[1],timeNow=math.trunc(time.time()))
file = open(fileName, 'w')
print ("{:<30} {:<30}".format('Polling Pipeline Name', '|','Last Executed</pre>
Time'))
print ("{:<30} {:<30} ".format('_____',</pre>
'|','____'))
for i in range(len(pollablePipelines)):
   print("{:<30} {:<30}".format(pollablePipelines[i], '|',</pre>
lastExecutedTimes[i]))
   file.write("{pipeline},".format(pipeline=pollablePipelines[i]))
file.close()
```

print("\nSaving Polling Pipeline Names to file
 {fileName}".format(fileName=fileName))

- 파이프라인이 있는 각 리전에 대해 해당 리전에 대한 스크립트를 실행해야 합니다. 스크립트를 실 행하려면 다음 중 하나를 수행합니다.
 - 다음 명령을 실행하여 PollingPipelinesExtractor.sh라는 스크립트를 실행합니다. 이 예제에서 리 전은 us-west-2입니다.

```
./PollingPipelinesExtractor.sh us-west-2
```

• Python 스크립트의 경우 다음 명령을 실행하여 PollingPipelinesExtractor.py라는 Python 스크립 트를 실행합니다. 이 예제에서 리전은 us-west-2입니다.

python3 PollingPipelinesExtractor.py us-west-2

스크립트의 다음 샘플 출력에서 us-west-2 리전은 폴링 파이프라인 목록을 반환하고 각 파이프라 인의 마지막 실행 시간을 보여줍니다.

```
% ./pollingPipelineExtractor.sh us-west-2

| Polling Pipeline Name | Last Executed Time |
| _______ | ______ |
| myCodeBuildPipeline | Wed Mar 8 09:35:49 PST 2023 |
| myCodeCommitPipeline | Mon Apr 24 22:32:32 PDT 2023 |
| TestPipeline | Not executed in last year |
Saving list of polling pipeline names to us-west-2-1682496174.csv...%
```

스크립트 출력을 분석하고 목록의 각 파이프라인에 대해 폴링 소스를 권장되는 이벤트 기반 변경 감지 방법으로 업데이트합니다.

Note

폴링 파이프라인은 PollForSourceChanges 파라미터에 대한 파이프라인의 작업 구성 에 따라 결정됩니다. 파이프라인 소스 구성에서 PollForSourceChanges 파라미터가 생 략된 경우 CodePipeline은 기본적으로 리포지토리를 폴링하여 소스 변경 사항을 확인합니 다. 이러한 동작은 PollForSourceChanges이 포함되었고 true로 설정된 경우와 똑같습 니다. 자세한 내용은 <u>Amazon S3 소스 작업 참조</u>에서 Amazon S3 소스 작업 구성 파라미 터 등 파이프라인 소스 작업에 대한 구성 파라미터를 참조하세요.

참고로 이 스크립트는 계정의 폴링 파이프라인 목록이 포함된.csv 파일을 생성하고.csv 파일을 현 재 작업 폴더에 저장합니다.

CodeCommit 소스를 사용하여 폴링 파이프라인 마이그레이션

EventBridge를 사용하여 CodeCommit 소스 리포지토리 또는 Amazon S3 소스 버킷의 변경 사항을 감 지하도록 폴링 파이프라인을 마이그레이션할 수 있습니다.

CodeCommit -- CodeCommit 소스가 있는 파이프라인의 경우 EventBridge를 통해 변경 감지가 자동화 되도록 파이프라인을 수정합니다. 다음 방법 중에서 선택하여 마이그레이션을 구현하세요.

- 콘솔: 폴링 파이프라인 마이그레이션(CodeCommit 또는 Amazon S3 소스)(콘솔)
- CLI: 폴링 파이프라인 마이그레이션(CodeCommit 소스)(CLI)
- AWS CloudFormation: <u>폴링 파이프라인 마이그레이션(CodeCommit 소스)(AWS CloudFormation</u> <u>템플릿)</u>

폴링 파이프라인 마이그레이션(CodeCommit 또는 Amazon S3 소스)(콘솔)

CodePipeline 콘솔을 사용하여 파이프라인을 업데이트하고 EventBridge를 사용하여 CodeCommit 소 스 리포지토리 또는 Amazon S3 소스 버킷의 변경 사항을 감지할 수 있습니다.

Note

콘솔을 사용하여 CodeCommit 소스 리포지토리 또는 Amazon S3 소스 버킷이 있는 파이프라 인을 편집하면 규칙과 IAM 역할이 생성됩니다. 를 사용하여 파이프라인을 편집 AWS CLI 하는 경우 EventBridge 규칙과 IAM 역할을 직접 생성해야 합니다. 자세한 내용은 <u>CodeCommit 소스</u> <u>작업 및 EventBridge</u> 단원을 참조하십시오.

정기적 확인을 사용하는 파이프라인을 편집하려면 다음 단계를 사용합니다. 파이프라인을 생성하려면 파이프라인 스테이지 및 작업 생성 단원을 참조하십시오.

파이프라인 소스 단계를 편집하려면

AWS 계정과 연결된 모든 파이프라인의 이름이 표시됩니다.

- [Name]에서 편집할 파이프라인의 이름을 선택합니다. 이렇게 하면 파이프라인 각 단계의 각 작업 상태를 포함하여 파이프라인의 세부 정보 보기가 열립니다.
- 3. 파이프라인 세부 정보 페이지에서 [Edit]를 선택합니다.
- 4. 편집 단계에서 소스 작업의 편집 아이콘을 선택합니다.
- 5. 변경 탐지 옵션을 확장하고 변경 사항이 생길 때 CloudWatch Events를 사용하여 내 파이프라인을 자동으로 시작합니다(권장)를 선택합니다.

이 파이프라인에 대해 생성될 EventBridge 규칙을 보여주는 메시지가 표시됩니다. 업데이트를 선 택합니다.

Amazon S3 소스가 있는 파이프라인을 업데이트하는 경우, 다음 메시지가 표시됩니다. 업데이 트를 선택합니다.

6. 파이프라인 편집을 마쳤으면 [Save pipeline changes]를 선택하여 요약 페이지로 돌아갑니다.

해당 파이프라인에 대해 생성될 EventBridge 규칙의 이름을 표시하는 메시지가 나타납니다. [Save and continue]를 선택합니다.

 작업을 테스트하려면를 사용하여 파이프라인의 소스 단계에 지정된 소스에 변경 사항을 AWS CLI 커밋하여 변경 사항을 릴리스합니다.

폴링 파이프라인 마이그레이션(CodeCommit 소스)(CLI)

EventBridge 규칙을 사용하여 파이프라인을 시작하도록 폴링(정기적 확인)을 사용하는 파이프라인을 편집하려면 다음 단계를 수행합니다. 파이프라인을 생성하려면 <u>파이프라인 스테이지 및 작업 생성</u> 단 원을 참조하십시오.

CodeCommit을 사용하여 이벤트 기반 파이프라인을 빌드하려면 파이프라인의 PollForSourceChanges 파라미터를 편집한 후 다음 리소스를 생성합니다.

- EventBridge 이벤트
- 이 이벤트가 파이프라인을 시작하도록 허용하는 IAM 역할

파이프라인의 PollForSourceChanges 파라미터를 편집하려면

A Important

이 방법으로 파이프라인을 생성할 때 명시적으로 false로 설정되지 않은 경우 PollForSourceChanges 파라미터 기본값은 true입니다. 이벤트 기반 변경 감지를 추가할 때 는 출력에 파라미터를 추가하고 false로 설정하여 폴링을 비활성화해야 합니다. 그렇지 않으면 파이프라인이 단일 소스 변경 시 두 번 시작됩니다. 세부 정보는 <u>PollForSourceChanges 파</u> 라미터의 유효한 설정을 참조하세요.

1. get-pipeline 명령을 실행하여 파이프라인 구조를 JSON 파일로 복사합니다. 예를 들어, MyFirstPipeline라는 파이프라인의 경우 다음 명령을 입력합니다.

aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json

이 명령은 아무 것도 반환하지 않지만 생성한 파일이 명령을 실행한 디렉터리에 표시되어야 합니 다.

2. 일반 텍스트 편집기에서 JSON 파일을 열고 이 예제에 나와 있는 것처럼 PollForSourceChanges 파라미터를 false로 변경하여 소스 단계를 편집합니다.

이렇게 변경하는 이유는 무엇입니까? 이 파라미터를 false로 변경하면 정기적 확인이 비활성화 되어 이벤트 기반 변경 탐지만 사용할 수 있습니다.

```
"configuration": {
    "PollForSourceChanges": "false",
    "BranchName": "main",
    "RepositoryName": "MyTestRepo"
},
```

 get-pipeline 명령을 사용하여 검색한 파이프라인 구조로 작업을 수행할 경우, JSON 파일에서 metadata 행을 제거하십시오. 이렇게 하지 않으면 update-pipeline 명령에서 사용할 수 없습니다. "metadata": { } 행과, "created", "pipelineARN" 및 "updated" 필드를 제거합니다.

예를 들어, 구조에서 다음 행을 삭제합니다.

```
"metadata": {
    "pipelineArn": "arn:aws:codepipeline:region:account-ID:pipeline-name",
    "created": "date",
```

},

```
"updated": "date"
```

파일을 저장합니다.

4. 변경 사항을 적용하려면 파이프라인 JSON 파일을 지정하여 update-pipeline 명령을 실행합니다.



aws codepipeline update-pipeline --cli-input-json file://pipeline.json

이 명령은 편집한 파이프라인의 전체 구조를 반환합니다.

Note

update-pipeline 명령을 실행하면 파이프라인이 중지됩니다. update-pipeline 명령을 실행 할 때 파이프라인을 통해 개정을 실행하는 중이라면 해당 실행이 중지됩니다. 업데이트 된 파이프라인을 통해 해당 개정을 실행하려면 파이프라인을 수동으로 시작해야 합니다. start-pipeline-execution 명령을 사용하여 수동으로 파이프라인을 시작합니다.

CodeCommit이 이벤트 소스이고 CodePipeline이 대상인 EventBridge 규칙을 만들려면

- 1. CodePipeline을 사용하여 규칙을 호출하도록 EventBridge에 권한을 추가합니다. 자세한 내용은 Amazon EventBridge에 대한 리소스 기반 정책 사용을 참조하세요.
 - a. 다음 예제를 사용하여 EventBridge가 서비스 역할을 담당하도록 허용하는 신뢰 정책을 생성 합니다. 신뢰 정책 이름을 trustpolicyforEB.json으로 지정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
               "Service": "events.amazonaws.com"
        },
    }
}
```
```
"Action": "sts:AssumeRole"
}
]
}
```

b. 다음 명령을 사용하여 Role-for-MyRule 역할을 생성한 후 신뢰 정책에 연결합니다.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
file://trustpolicyforEB.json
```

c. 이 샘플에서 보이는 것처럼 MyFirstPipeline이라는 파이프라인에 대한 권한 정책 JSON 을 생성합니다. 권한 정책 이름을 permissionspolicyforEB.json으로 지정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "codepipeline:StartPipelineExecution"
        ],
        "Resource": [
              "arn:aws:codepipeline:us-west-2:80398EXAMPLE:MyFirstPipeline"
        ]
        }
    ]
}
```

 d. 다음 명령을 사용하여 CodePipeline-Permissions-Policy-for-EB 권한 정책을 Role-for-MyRule 역할에 연결합니다.

이렇게 변경하는 이유는 무엇입니까? 이 정책을 역할에 추가하면 EventBridge에 대한 권한이 생성됩니다.

aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json

2. put-rule 명령을 호출하고 --name, --event-pattern 및 --role-arn 파라미터를 포함시킵니 다.

이렇게 변경하는 이유는 무엇입니까? 이 명령은 AWS CloudFormation 에서 이벤트를 생성할 수 있게 합니다.

다음 샘플 명령은 MyCodeCommitRepoRule이라는 역할 별칭을 생성합니다.

aws events put-rule --name "MyCodeCommitRepoRule" --event-pattern "{\"source\":
[\"aws.codecommit\"],\"detail-type\":[\"CodeCommit Repository State Change\"],
\"resources\":[\"repository-ARN\"],\"detail\":{\"referenceType\":[\"branch\"],
\"referenceName\":[\"main\"]}}" --role-arn "arn:aws:iam::ACCOUNT_ID:role/Role-forMyRule"

- 3. CodePipeline을 대상으로 추가하려면 put-targets 명령을 호출하고 다음 파라미터를 포함합니다.
 - --rule 파라미터는 put-rule을 사용하여 생성한 rule_name에 사용됩니다.
 - --targets 파라미터는 대상 목록에 있는 대상의 목록 Id 및 대상 파이프라인의 ARN에 사용됩니다.

다음 예제 명령은 MyCodeCommitRepoRule이라는 규칙에 대해 대상 Id가 숫자 1로 구성됨을 지 정하며, 규칙의 대상 목록에서 1로 대상 1로 표시됩니다. 이 예제 명령은 또한 파이프라인에 대한 예제 ARN를 지정합니다. 파이프라인은 리포지토리에서 변경이 발생하면 시작됩니다.

```
aws events put-targets --rule MyCodeCommitRepoRule --targets
Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
```

- 4. (선택 사항) 특정 이미지 ID에 대한 소스 재정의로 입력 변환기를 구성하려면 CLI 명령에서 다음 JSON을 사용합니다. 다음 예제에서는 다음과 같은 재정의를 구성합니다.
 - 이 예제actionNameSource에서는 소스 이벤트에서 파생되지 않은 파이프라인 생성 시 정의 된 동적 값입니다.
 - 이 예제revisionTypeCOMMIT_ID에서는 소스 이벤트에서 파생되지 않은 파이프라인 생성 시 정의된 동적 값입니다.
 - 이 예제의, revisionValue<revisionValue>는 소스 이벤트 변수에서 파생됩니다.



폴링 파이프라인 마이그레이션(CodeCommit 소스)(AWS CloudFormation 템플릿)

를 사용하여 이벤트 기반 파이프라인을 빌드하려면 파이프라인의 PollForSourceChanges 파라미 터를 AWS CodeCommit편집한 다음 템플릿에 다음 리소스를 추가합니다.

- EventBridge 규칙
- EventBridge 규칙의 IAM 역할

AWS CloudFormation 를 사용하여 파이프라인을 생성하고 관리하는 경우 템플릿에는 다음과 같은 콘 텐츠가 포함됩니다.

1 Note

PollForSourceChanges라고 하는 소스 단계의 Configuration 속성. 템플릿에 해당 속성 이 포함되어 있지 않으면 기본적으로 PollForSourceChanges가 true로 설정됩니다.

```
Resources:
AppPipeline:
Type: AWS::CodePipeline::Pipeline
Properties:
Name: codecommit-polling-pipeline
RoleArn:
```

```
!GetAtt CodePipelineServiceRole.Arn
Stages:
    Name: Source
    Actions:
      _
        Name: SourceAction
        ActionTypeId:
          Category: Source
          Owner: AWS
          Version: 1
          Provider: CodeCommit
        OutputArtifacts:
          - Name: SourceOutput
        Configuration:
          BranchName: !Ref BranchName
          RepositoryName: !Ref RepositoryName
          PollForSourceChanges: true
        RunOrder: 1
```

```
"Stages": [
   {
        "Name": "Source",
 "Actions": [{
      "Name": "SourceAction",
      "ActionTypeId": {
    "Category": "Source",
    "Owner": "AWS",
    "Version": 1,
    "Provider": "CodeCommit"
      },
      "OutputArtifacts": [{
          "Name": "SourceOutput"
     }],
      "Configuration": {
          "BranchName": {
        "Ref": "BranchName"
   },
    "RepositoryName": {
        "Ref": "RepositoryName"
   },
```

```
"PollForSourceChanges": true
      },
      "RunOrder": 1
    }]
},
```

파이프라인 AWS CloudFormation 템플릿을 업데이트하고 EventBridge 규칙을 생성하려면

- 템플릿의에서 AWS::IAM::Role AWS CloudFormation 리소스를 Resources사용하여 이벤트가 파이프라인을 시작하도록 허용하는 IAM 역할을 구성합니다. 이 항목은 두 가지 정책을 사용하는 역할을 만듭니다.
 - 첫 번째 정책은 가 역할을 수임하도록 허용합니다.
 - 두 번째 정책은 파이프라인을 시작할 권한을 부여합니다.

이렇게 변경하는 이유는 무엇입니까? AWS : : IAM : : Role 리소스를 추가 AWS CloudFormation 하 면가 EventBridge에 대한 권한을 생성할 수 있습니다. 이 리소스는 AWS CloudFormation 스택에 추가됩니다.

```
EventRole:
 Type: AWS::IAM::Role
 Properties:
   AssumeRolePolicyDocument:
      Version: 2012-10-17
      Statement:
          Effect: Allow
          Principal:
            Service:
              - events.amazonaws.com
          Action: sts:AssumeRole
    Path: /
    Policies:
        PolicyName: eb-pipeline-execution
        PolicyDocument:
          Version: 2012-10-17
          Statement:
```

```
Effect: Allow
Action: codepipeline:StartPipelineExecution
Resource: !Join [ '', [ 'arn:aws:codepipeline:', !Ref
'AWS::Region', ':', !Ref 'AWS::AccountId', ':', !Ref AppPipeline ] ]
```

```
"EventRole": {
  "Type": "AWS::IAM::Role",
  "Properties": {
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": [
              "events.amazonaws.com"
            ]
          },
          "Action": "sts:AssumeRole"
        }
      ]
    },
    "Path": "/",
    "Policies": [
      {
        "PolicyName": "eb-pipeline-execution",
        "PolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Effect": "Allow",
              "Action": "codepipeline:StartPipelineExecution",
              "Resource": {
                "Fn::Join": [
                  "",
                  Ε
                    "arn:aws:codepipeline:",
                    {
                      "Ref": "AWS::Region"
                    },
```

```
":",
{
"Ref": "AWS::AccountId"
},
":",
{
"Ref": "AppPipeline"
}
]
```

2. 템플릿의에서 AWS::Events::Rule AWS CloudFormation 리소스를 Resources사용하여 EventBridge 규칙을 추가합니다. 이 이벤트 패턴은 리포지토리에 대한 푸시 변경 사항을 모니터링 하는 이벤트를 생성합니다. EventBridge가 리포지토리 상태 변경을 감지하면 해당 규칙이 대상 파 이프라인에서 StartPipelineExecution을 호출합니다.

이렇게 변경하는 이유는 무엇입니까? AWS::Events::Rule 리소스를 추가 AWS CloudFormation 하면가 이벤트를 생성할 수 있습니다. 이 리소스는 AWS CloudFormation 스택에 추가됩니다.

```
EventRule:
  Type: AWS::Events::Rule
  Properties:
    EventPattern:
      source:
         - aws.codecommit
      detail-type:
         - 'CodeCommit Repository State Change'
      resources:
         - !Join [ '', [ 'arn:aws:codecommit:', !Ref 'AWS::Region', ':', !Ref
'AWS::AccountId', ':', !Ref RepositoryName ] ]
      detail:
         event:
           - referenceCreated
           - referenceUpdated
         referenceType:
           - branch
         referenceName:
           - main
    Targets:
```

```
"EventRule": {
  "Type": "AWS::Events::Rule",
  "Properties": {
    "EventPattern": {
      "source": [
        "aws.codecommit"
      ],
      "detail-type": [
        "CodeCommit Repository State Change"
      ],
      "resources": [
        {
          "Fn::Join": [
            "",
            Γ
              "arn:aws:codecommit:",
              {
                "Ref": "AWS::Region"
              },
              ":",
              {
                "Ref": "AWS::AccountId"
              },
              ":",
              {
                "Ref": "RepositoryName"
              }
            ]
          ]
        }
      ],
      "detail": {
        "event": [
```

```
"referenceCreated",
          "referenceUpdated"
        ],
        "referenceType": [
          "branch"
        ],
        "referenceName": [
          "main"
        ]
      }
    },
    "Targets": [
      {
        "Arn": {
          "Fn::Join": [
            "",
            Γ
              "arn:aws:codepipeline:",
              {
                "Ref": "AWS::Region"
              },
              ":",
              {
                "Ref": "AWS::AccountId"
              },
              ":",
              {
                "Ref": "AppPipeline"
              }
            ]
          ]
        },
        "RoleArn": {
          "Fn::GetAtt": [
            "EventRole",
            "Arn"
          ]
        },
        "Id": "codepipeline-AppPipeline"
      }
    ]
 }
},
```

- (선택 사항) 특정 이미지 ID에 대한 소스 재정의로 입력 변환기를 구성하려면 다음 YAML 코드 조 각을 사용합니다. 다음 예제에서는 다음과 같은 재정의를 구성합니다.
 - 이 예제actionNameSource에서는 소스 이벤트에서 파생되지 않은 파이프라인 생성 시 정의 된 동적 값입니다.
 - 이 예제revisionTypeCOMMIT_ID에서는 소스 이벤트에서 파생되지 않은 파이프라인 생성 시 정의된 동적 값입니다.
 - 이 예제의, revisionValue<*revisionValue*>는 소스 이벤트 변수에서 파생됩니다.
 - BranchName 및의 출력 변수가 지정Value됩니다.

```
Rule: my-rule
Targets:
- Id: MyTargetId
Arn: pipeline-ARN
InputTransformer:
    sourceRevisions:
        actionName: Source
        revisionType: COMMIT_ID
        revisionValue: <revisionValue>
        variables:
            name: BranchName
            value: value
```

- 4. 업데이트된 템플릿을 로컬 컴퓨터에 저장한 다음 AWS CloudFormation 콘솔을 엽니다.
- 5. 스택을 선택한 후 현재 스택에 대한 변경 세트 만들기를 선택합니다.
- 템플릿을 업로드한 후 AWS CloudFormation에 나열된 변경 사항을 확인합니다. 이는 스택에 적용 될 변경 사항입니다. 목록에 새로운 리소스가 표시됩니다.
- 7. 실행을 선택합니다.

파이프라인의 PollForSourceChanges 파라미터를 편집하려면

A Important

많은 경우 파이프라인을 생성할 때 PollForSourceChanges 파라미터 기본값은 true입니다. 이벤트 기반 변경 감지를 추가할 때는 출력에 파라미터를 추가하고 false로 설정하여 폴링을 비 활성화해야 합니다. 그렇지 않으면 파이프라인이 단일 소스 변경 시 두 번 시작됩니다. 세부 정 보는 PollForSourceChanges 파라미터의 유효한 설정을 참조하세요.

템플릿에서 PollForSourceChanges를 false로 변경합니다. PollForSourceChanges를 파 이프라인 정의에 포함하지 않은 경우 추가하고 false로 설정하세요.

이렇게 변경하는 이유는 무엇입니까? 이 파라미터를 false로 변경하면 정기적 확인이 비활성화 되어 이벤트 기반 변경 탐지만 사용할 수 있습니다.

YAML

Name: Source	
Actions:	
-	
Name: SourceAction	
ActionTypeId:	
Category: Source	
Owner: AWS	
Version: 1	
Provider: CodeCommit	
OutputArtifacts:	
- Name: SourceOutput	
Configuration:	
BranchName: !Ref BranchName	
RepositoryName: !Ref RepositoryName	
PollForSourceChanges: false	
RunOrder: 1	

```
{
   "Name": "Source",
   "Actions": [
    {
        "Name": "SourceAction",
        "ActionTypeId": {
            "Category": "Source",
            "Owner": "AWS",
            "Version": 1,
            "Provider": "CodeCommit"
        },
```

```
"OutputArtifacts": [
        {
          "Name": "SourceOutput"
        }
      ],
      "Configuration": {
        "BranchName": {
          "Ref": "BranchName"
        },
        "RepositoryName": {
          "Ref": "RepositoryName"
        },
        "PollForSourceChanges": false
      },
      "RunOrder": 1
    }
  ]
},
```

Example

를 사용하여 이러한 리소스를 생성하면 AWS CloudFormation리포지토리의 파일이 생성되거나 업데이 트될 때 파이프라인이 트리거됩니다. 다음은 최종 템플릿 스니펫입니다.

```
Resources:

EventRole:

Type: AWS::IAM::Role

Properties:

AssumeRolePolicyDocument:

Version: 2012-10-17

Statement:

-

Effect: Allow

Principal:

Service:

- events.amazonaws.com

Action: sts:AssumeRole

Path: /

Policies:
```

```
PolicyName: eb-pipeline-execution
        PolicyDocument:
           Version: 2012-10-17
           Statement:
               Effect: Allow
               Action: codepipeline:StartPipelineExecution
               Resource: !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region',
':', !Ref 'AWS::AccountId', ':', !Ref AppPipeline ] ]
EventRule:
  Type: AWS::Events::Rule
  Properties:
     EventPattern:
       source:
         - aws.codecommit
       detail-type:
         - 'CodeCommit Repository State Change'
       resources:
         - !Join [ '', [ 'arn:aws:codecommit:', !Ref 'AWS::Region', ':', !Ref
'AWS::AccountId', ':', !Ref RepositoryName ] ]
       detail:
         event:
           - referenceCreated
           - referenceUpdated
        referenceType:
           - branch
         referenceName:
           - main
    Targets:
        Arn:
           !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region', ':', !Ref
'AWS::AccountId', ':', !Ref AppPipeline ] ]
        RoleArn: !GetAtt EventRole.Arn
        Id: codepipeline-AppPipeline
AppPipeline:
  Type: AWS::CodePipeline::Pipeline
  Properties:
     Name: codecommit-events-pipeline
     RoleArn:
       !GetAtt CodePipelineServiceRole.Arn
     Stages:
```

```
Name: Source
Actions:
-
Name: SourceAction
ActionTypeId:
Category: Source
Owner: AWS
Version: 1
Provider: CodeCommit
OutputArtifacts:
- Name: SourceOutput
Configuration:
BranchName: !Ref BranchName
RepositoryName: !Ref RepositoryName
PollForSourceChanges: false
RunOrder: 1
```

. . .

```
"Resources": {
. . .
        "EventRole": {
            "Type": "AWS:::IAM::Role",
            "Properties": {
                "AssumeRolePolicyDocument": {
                     "Version": "2012-10-17",
                     "Statement": [
                         {
                             "Effect": "Allow",
                             "Principal": {
                                 "Service": [
                                      "events.amazonaws.com"
                                 ]
                             },
                             "Action": "sts:AssumeRole"
                         }
                    ]
                },
```

```
"Path": "/",
        "Policies": [
            {
                "PolicyName": "eb-pipeline-execution",
                "PolicyDocument": {
                     "Version": "2012-10-17",
                     "Statement": [
                         {
                             "Effect": "Allow",
                             "Action": "codepipeline:StartPipelineExecution",
                             "Resource": {
                                 "Fn::Join": [
                                     "",
                                     Ε
                                          "arn:aws:codepipeline:",
                                          {
                                              "Ref": "AWS::Region"
                                          },
                                          ":",
                                          {
                                              "Ref": "AWS::AccountId"
                                          },
                                          ":",
                                          {
                                              "Ref": "AppPipeline"
                                          }
                                     ]
                                 ]
                             }
                         }
                    ]
                }
            }
        ]
    }
},
"EventRule": {
    "Type": "AWS::Events::Rule",
    "Properties": {
        "EventPattern": {
            "source": [
                "aws.codecommit"
            ],
            "detail-type": [
```

```
"CodeCommit Repository State Change"
    ],
    "resources": [
        {
             "Fn::Join": [
                 "",
                 Ε
                     "arn:aws:codecommit:",
                     {
                         "Ref": "AWS::Region"
                     },
                     ":",
                     {
                         "Ref": "AWS::AccountId"
                     },
                     ":",
                     {
                         "Ref": "RepositoryName"
                     }
                 ]
            ]
        }
    ],
    "detail": {
        "event": [
            "referenceCreated",
            "referenceUpdated"
        ],
        "referenceType": [
            "branch"
        ],
        "referenceName": [
            "main"
        ]
    }
},
"Targets": [
    {
        "Arn": {
             "Fn::Join": [
                 "",
                 Ε
                     "arn:aws:codepipeline:",
                     {
```

```
"Ref": "AWS::Region"
                             },
                             ":",
                             {
                                 "Ref": "AWS::AccountId"
                             },
                             ":",
                             {
                                 "Ref": "AppPipeline"
                             }
                         ]
                     ]
                },
                 "RoleArn": {
                     "Fn::GetAtt": [
                         "EventRole",
                         "Arn"
                     ]
                },
                "Id": "codepipeline-AppPipeline"
            }
        ]
    }
},
"AppPipeline": {
    "Type": "AWS::CodePipeline::Pipeline",
    "Properties": {
        "Name": "codecommit-events-pipeline",
        "RoleArn": {
            "Fn::GetAtt": [
                "CodePipelineServiceRole",
                "Arn"
            ]
        },
        "Stages": [
            {
                 "Name": "Source",
                "Actions": [
                     {
                         "Name": "SourceAction",
                         "ActionTypeId": {
                             "Category": "Source",
                             "Owner": "AWS",
                             "Version": 1,
```

```
"Provider": "CodeCommit"
                                 },
                                  "OutputArtifacts": [
                                      {
                                          "Name": "SourceOutput"
                                      }
                                 ],
                                  "Configuration": {
                                      "BranchName": {
                                          "Ref": "BranchName"
                                      },
                                      "RepositoryName": {
                                          "Ref": "RepositoryName"
                                      },
                                      "PollForSourceChanges": false
                                 },
                                 "RunOrder": 1
                             }
                         ]
                     },
. . .
```

이벤트용으로 활성화된 S3 소스를 사용하여 폴링 파이프라인 마이그레이션

Amazon S3 소스가 있는 파이프라인의 경우, EventBridge와 이벤트 알림이 활성화된 소스 버킷을 통해 변경 감지가 자동화되도록 파이프라인을 수정하세요. CLI를 사용하거나 파이프라인을 마이그레이션 AWS CloudFormation 하는 경우 권장되는 방법입니다.

Note

여기에는 별도의 CloudTrail 트레일을 생성할 필요가 없는 이벤트 알림이 활성화된 버킷을 사용하는 것도 포함됩니다. 콘솔을 사용하는 경우 이벤트 규칙과 CloudTrail 트레일이 자동으로 설정됩니다. 이러한 단계는 <u>S3 소스 및 CloudTrail 트레일을 사용하여 폴링 파이프라인 마이그</u> 레이션을 참조하세요.

• CLI: S3 소스 및 CloudTrail 트레일을 사용하여 폴링 파이프라인 마이그레이션(CLI)

• AWS CloudFormation: <u>S3 소스 및 CloudTrail 추적을 사용하여 폴링 파이프라인 마이그레이션(AWS</u> CloudFormation 템플릿)

이벤트용으로 활성화된 S3 소스를 사용하여 폴링 파이프라인 마이그레이션(CLI)

대신 EventBridge의 이벤트를 사용하도록 폴링(정기적 확인)을 사용하는 파이프라인을 편집하려면 다 음 단계를 수행합니다. 파이프라인을 생성하려면 <u>파이프라인 스테이지 및 작업 생성</u> 단원을 참조하십 시오.

Amazon S3를 사용하여 이벤트 기반 파이프라인을 빌드하려면 파이프라인의 PollForSourceChanges 파라미터를 편집한 후 다음 리소스를 생성합니다.

- EventBridge 이벤트 규칙
- EventBridge 이벤트가 파이프라인을 시작하도록 허용하는 IAM 역할

Amazon S3가 이벤트 소스이고 CodePipeline이 대상인 EventBridge 규칙을 생성하고 권한 정책을 적 용하려면

- 1. CodePipeline을 사용하여 규칙을 호출하도록 EventBridge에 권한을 부여합니다. 자세한 내용은 Amazon EventBridge에 대한 리소스 기반 정책 사용을 참조하세요.
 - a. 다음 예제를 사용하여 EventBridge가 서비스 역할을 담당하도록 허용하는 신뢰 정책을 생성 합니다. 이름을 trustpolicyforEB.json로 지정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
               "Principal": {
                 "Service": "events.amazonaws.com"
               },
               "Action": "sts:AssumeRole"
               }
    ]
}
```

b. 다음 명령을 사용하여 Role-for-MyRule 역할을 생성한 후 신뢰 정책에 연결합니다.

이렇게 변경하는 이유는 무엇입니까? 이 신뢰 정책을 역할에 추가하면 EventBridge에 대한 권한이 생성됩니다.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
file://trustpolicyforEB.json
```

c. 이 샘플에서 보이는 것처럼 MyFirstPipeline이라는 파이프라인에 대한 권한 정책 JSON 을 생성합니다. 권한 정책 이름을 permissionspolicyforEB.json으로 지정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "codepipeline:StartPipelineExecution"
        ],
        "Resource": [
              "arn:aws:codepipeline:us-west-2:80398EXAMPLE:MyFirstPipeline"
        ]
        }
    ]
}
```

d. 다음 명령을 사용하여 앞에서 생성한 Role-for-MyRule 역할에 새로운 CodePipeline-Permissions-Policy-for-EB 권한 정책을 연결합니다.

aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json

2. put-rule 명령을 호출하고 --name, --event-pattern 및 --role-arn 파라미터를 포함시킵니 다.

다음 샘플 명령은 EnabledS3SourceRule이라는 역할 별칭을 생성합니다.

```
aws events put-rule --name "EnabledS3SourceRule" --event-pattern "{\"source\":
[\"aws.s3\"],\"detail-type\":[\"Object Created\"],\"detail\":{\"bucket\":{\"name\":
[\"amzn-s3-demo-source-bucket\"]}}" --role-arn "arn:aws:iam::ACCOUNT_ID:role/Role-
for-MyRule"
```

3. CodePipeline을 대상으로 추가하려면 put-targets 명령을 호출하고 --rule 및 --targets 파라 미터를 포함합니다.

다음 명령은 EnabledS3SourceRule이라는 규칙에 대해 대상 Id가 숫자 1로 구성됨을 지정 하며, 규칙에 대한 대상 목록에서 대상 1로 표시됩니다. 이 명령은 또한 파이프라인에 대한 예제 ARN를 지정합니다. 파이프라인은 리포지토리에서 변경이 발생하면 시작됩니다.

aws events put-targets --rule EnabledS3SourceRule --targets Id=codepipeline-AppPipeline,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline

파이프라인의 PollForSourceChanges 파라미터를 편집하려면

A Important

이 방법으로 파이프라인을 생성할 때 명시적으로 false로 설정되지 않은 경우 PollForSourceChanges 파라미터 기본값은 true입니다. 이벤트 기반 변경 감지를 추가할 때 는 출력에 파라미터를 추가하고 false로 설정하여 폴링을 비활성화해야 합니다. 그렇지 않으면 파이프라인이 단일 소스 변경 시 두 번 시작됩니다. 세부 정보는 <u>PollForSourceChanges 파</u> 라미터의 유효한 설정을 참조하세요.

1. get-pipeline 명령을 실행하여 파이프라인 구조를 JSON 파일로 복사합니다. 예를 들어, MyFirstPipeline라는 파이프라인의 경우 다음 명령을 입력합니다.

aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json

이 명령은 아무 것도 반환하지 않지만 생성한 파일이 명령을 실행한 디렉터리에 표시되어야 합니 다.

2. 일반 텍스트 편집기에서 JSON 파일을 열고 다음 예에 나와 있는 것처럼 amzn-s3-demosource-bucket 버킷의 PollForSourceChanges 파라미터를 false로 변경하여 소스 단계를 편집합니다.

이렇게 변경하는 이유는 무엇입니까? 이 파라미터를 false로 설정하면 정기적 확인이 비활성화 되어 이벤트 기반 변경 탐지만 사용할 수 있습니다.

```
"configuration": {
    "S3Bucket": "amzn-s3-demo-source-bucket",
    "PollForSourceChanges": "false",
```

```
"S3ObjectKey": "index.zip"
},
```

 get-pipeline 명령을 사용하여 검색한 파이프라인 구조로 작업을 수행할 경우, JSON 파일에서 metadata 행을 제거해야 합니다. 이렇게 하지 않으면 update-pipeline 명령에서 사용할 수 없습니 다. "metadata": { } 행과, "created", "pipelineARN" 및 "updated" 필드를 제거합니다.

예를 들어, 구조에서 다음 행을 삭제합니다.

```
"metadata": {
    "pipelineArn": "arn:aws:codepipeline:region:account-ID:pipeline-name",
    "created": "date",
    "updated": "date"
},
```

파일을 저장합니다.

4. 변경 사항을 적용하려면 파이프라인 JSON 파일을 지정하여 update-pipeline 명령을 실행합니다.

```
\Lambda Important
```

파일 이름 앞에 file://를 포함해야 합니다. 이 명령에 필수적입니다.

```
aws codepipeline update-pipeline --cli-input-json file://pipeline.json
```

이 명령은 편집한 파이프라인의 전체 구조를 반환합니다.

1 Note

update-pipeline 명령을 실행하면 파이프라인이 중지됩니다. update-pipeline 명령을 실행 할 때 파이프라인을 통해 개정을 실행하는 중이라면 해당 실행이 중지됩니다. 업데이트 된 파이프라인을 통해 해당 개정을 실행하려면 파이프라인을 수동으로 시작해야 합니다. start-pipeline-execution 명령을 사용하여 수동으로 파이프라인을 시작합니다. 이벤트에 대해 활성화된 S3 소스를 사용하여 폴링 파이프라인 마이그레이션(AWS CloudFormation 템플릿)

이 절차는 소스 버킷에 이벤트가 활성화된 파이프라인을 위한 것입니다.

폴링에서 이벤트 기반 변경 감지로 Amazon S3 소스가 있는 파이프라인을 편집하려면 다음 단계를 수 행합니다.

Amazon S3를 사용하여 이벤트 기반 파이프라인을 빌드하려면 해당 파이프라인의 PollForSourceChanges 파라미터를 편집한 후 다음 리소스를 템플릿에 추가합니다.

• 이 이벤트가 파이프라인을 시작할 수 있게 하는 EventBridge 규칙 및 IAM 역할.

AWS CloudFormation 를 사용하여 파이프라인을 생성하고 관리하는 경우 템플릿에는 다음과 같은 콘 텐츠가 포함됩니다.

1 Note

PollForSourceChanges라고 하는 소스 단계의 Configuration 속성. 템플릿에 해당 속성 이 포함되어 있지 않으면 기본적으로 PollForSourceChanges가 true로 설정됩니다.

```
AppPipeline:

Type: AWS::CodePipeline::Pipeline

Properties:

RoleArn: !GetAtt CodePipelineServiceRole.Arn

Stages:

-

Name: Source

Actions:

-

Name: SourceAction

ActionTypeId:

Category: Source

Owner: AWS

Version: 1

Provider: S3

OutputArtifacts:
```

```
Name: SourceOutput
Configuration:
S3Bucket: !Ref SourceBucket
S3ObjectKey: !Ref S3SourceObjectKey
PollForSourceChanges: true
RunOrder: 1
```

• • •

```
"AppPipeline": {
    "Type": "AWS::CodePipeline::Pipeline",
    "Properties": {
        "RoleArn": {
            "Fn::GetAtt": ["CodePipelineServiceRole", "Arn"]
        },
        "Stages": [
            {
                "Name": "Source",
                "Actions": [
                    {
                        "Name": "SourceAction",
                        "ActionTypeId": {
                             "Category": "Source",
                             "Owner": "AWS",
                            "Version": 1,
                            "Provider": "S3"
                        },
                        "OutputArtifacts": [
                            {
                                 "Name": "SourceOutput"
                            }
                        ],
                        "Configuration": {
                             "S3Bucket": {
                                 "Ref": "SourceBucket"
                            },
                             "S3ObjectKey": {
                                 "Ref": "SourceObjectKey"
                            },
                             "PollForSourceChanges": true
```



Amazon S3가 이벤트 소스이고 CodePipeline이 대상인 EventBridge 규칙을 생성하고 권한 정책을 적용하려면

- 템플릿의에서 AWS::IAM::Role AWS CloudFormation 리소스를 Resources사용하여 이벤트가 파이프라인을 시작하도록 허용하는 IAM 역할을 구성합니다. 이 항목은 두 가지 정책을 사용하는 역할을 만듭니다.
 - 첫 번째 정책은 가 역할을 수임하도록 허용합니다.
 - 두 번째 정책은 파이프라인을 시작할 권한을 부여합니다.

이렇게 변경하는 이유는 무엇입니까? AWS::IAM::Role 리소스를 추가하면 AWS CloudFormation 가 EventBridge에 대한 권한을 생성할 수 있습니다. 이 리소스는 AWS CloudFormation 스택에 추가됩니다.

```
EventRole:

Type: AWS::IAM::Role

Properties:

AssumeRolePolicyDocument:

Version: 2012-10-17

Statement:

-

Effect: Allow

Principal:

Service:

- events.amazonaws.com

Action: sts:AssumeRole

Path: /

Policies:
```

```
PolicyName: eb-pipeline-execution
PolicyDocument:
    Version: 2012-10-17
    Statement:
        -
        Effect: Allow
        Action: codepipeline:StartPipelineExecution
        Resource: !Join [ '', [ 'arn:aws:codepipeline:', !Ref
'AWS::Region', ':', !Ref 'AWS::AccountId', ':', !Ref AppPipeline ] ]
...
```

```
"EventRole": {
  "Type": "AWS::IAM::Role",
  "Properties": {
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": [
              "events.amazonaws.com"
            ]
          },
          "Action": "sts:AssumeRole"
        }
      ]
    },
    "Path": "/",
    "Policies": [
      {
        "PolicyName": "eb-pipeline-execution",
        "PolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Effect": "Allow",
              "Action": "codepipeline:StartPipelineExecution",
              "Resource": {
```

```
"Fn::Join": [
                      "",
                      Г
                        "arn:aws:codepipeline:",
                        {
                          "Ref": "AWS::Region"
                        },
                        ":",
                        {
                          "Ref": "AWS::AccountId"
                        },
                        ":",
                        {
                          "Ref": "AppPipeline"
                        }
                      ]
                    ]
. . .
```

2. AWS::Events::Rule AWS CloudFormation 리소스를 사용하여 EventBridge 규칙을 추가합니다. 이 이벤트 패턴은 Amazon S3 원본 버킷의 객체 생성 또는 삭제를 모니터링하는 이벤트를 생성합니다. 또한 파이프라인 대상도 포함하세요. 객체가 생성되면 이 규칙이 대상 파이프라인의 StartPipelineExecution을 호출합니다.

이렇게 변경하는 이유는 무엇입니까? AWS::Events::Rule 리소스를 추가 AWS CloudFormation 하면가 이벤트를 생성할 수 있습니다. 이 리소스는 AWS CloudFormation 스택에 추가됩니다.

```
EventRule:
Type: AWS::Events::Rule
Properties:
EventBusName: default
EventPattern:
source:
- aws.s3
detail-type:
- Object Created
detail:
bucket:
name:
```

```
- !Ref SourceBucket
Name: EnabledS3SourceRule
State: ENABLED
Targets:
    -
    Arn:
    !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region', ':', !Ref
'AWS::AccountId', ':', !Ref AppPipeline ] ]
    RoleArn: !GetAtt EventRole.Arn
    Id: codepipeline-AppPipeline
...
```

```
"EventRule": {
  "Type": "AWS::Events::Rule",
  "Properties": {
"EventBusName": "default",
"EventPattern": {
    "source": [
 "aws.s3"
    ],
    "detail-type": [
  "Object Created"
    ],
    "detail": {
  "bucket": {
      "name": [
    "s3-pipeline-source-fra-bucket"
      ]
      }
          }
},
"Name": "EnabledS3SourceRule",
      "State": "ENABLED",
      "Targets": [
      {
        "Arn": {
          "Fn::Join": [
            "",
```



- 3. 업데이트된 템플릿을 로컬 컴퓨터에 저장하고 AWS CloudFormation 콘솔을 엽니다.
- 4. 스택을 선택한 후 현재 스택에 대한 변경 세트 만들기를 선택합니다.
- 5. 업데이트된 템플릿을 업로드한 후 AWS CloudFormation에 나열된 변경 사항을 확인합니다. 이는 스택에 적용될 변경 사항입니다. 목록에 새로운 리소스가 표시됩니다.
- 6. 실행을 선택합니다.

파이프라인의 PollForSourceChanges 파라미터를 편집하려면

A Important

이 방법으로 파이프라인을 생성할 때 명시적으로 false로 설정되지 않은 경우 PollForSourceChanges 파라미터 기본값은 true입니다. 이벤트 기반 변경 감지를 추가할 때 는 출력에 파라미터를 추가하고 false로 설정하여 폴링을 비활성화해야 합니다. 그렇지 않으면 파이프라인이 단일 소스 변경 시 두 번 시작됩니다. 세부 정보는 <u>PollForSourceChanges 파</u> 라미터의 유효한 설정을 참조하세요.

템플릿에서 PollForSourceChanges를 false로 변경합니다. PollForSourceChanges를 파 이프라인 정의에 포함하지 않은 경우 추가하고 false로 설정하세요.

이렇게 변경하는 이유는 무엇입니까? PollForSourceChanges를 false로 변경하면 정기적 확 인이 비활성화되어 이벤트 기반 변경 탐지만 사용할 수 있습니다.

YAML

```
Name: Source
Actions:
-
Name: SourceAction
ActionTypeId:
Category: Source
Owner: AWS
Version: 1
Provider: S3
OutputArtifacts:
- Name: SourceOutput
Configuration:
S3Bucket: !Ref SourceBucket
S3ObjectKey: !Ref SourceObjectKey
PollForSourceChanges: false
RunOrder: 1
```

JSON

{

```
"Name": "SourceAction",
"ActionTypeId": {
```

```
"Category": "Source",
    "Owner": "AWS",
    "Version": 1,
    "Provider": "S3"
 },
  "OutputArtifacts": [
   {
      "Name": "SourceOutput"
   }
  ],
  "Configuration": {
    "S3Bucket": {
      "Ref": "SourceBucket"
   },
    "S3ObjectKey": {
      "Ref": "SourceObjectKey"
   },
    "PollForSourceChanges": false
 },
 "RunOrder": 1
}
```

Example

AWS CloudFormation 를 사용하여 이러한 리소스를 생성하면 리포지토리의 파일이 생성되거나 업데 이트될 때 파이프라인이 트리거됩니다.

Note

여기에서 중단하지 마십시오. 파이프라인이 생성되더라도 Amazon S3 파이프라인에 대한 두 번째 AWS CloudFormation 템플릿을 생성해야 합니다. 두 번째 템플릿을 생성하지 않으면 파 이프라인에 변경 감지 기능이 없습니다.

```
Parameters:
SourceObjectKey:
Description: 'S3 source artifact'
Type: String
```

```
Default: SampleApp_Linux.zip
 ApplicationName:
    Description: 'CodeDeploy application name'
    Type: String
    Default: DemoApplication
  BetaFleet:
    Description: 'Fleet configured in CodeDeploy'
    Type: String
    Default: DemoFleet
Resources:
  SourceBucket:
    Type: AWS::S3::Bucket
    Properties:
      NotificationConfiguration:
        EventBridgeConfiguration:
          EventBridgeEnabled: true
      VersioningConfiguration:
        Status: Enabled
  CodePipelineArtifactStoreBucket:
    Type: AWS::S3::Bucket
  CodePipelineArtifactStoreBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket: !Ref CodePipelineArtifactStoreBucket
      PolicyDocument:
        Version: 2012-10-17
        Statement:
            Sid: DenyUnEncryptedObjectUploads
            Effect: Deny
            Principal: '*'
            Action: s3:PutObject
            Resource: !Join [ '', [ !GetAtt CodePipelineArtifactStoreBucket.Arn, '/
*']]
            Condition:
              StringNotEquals:
                s3:x-amz-server-side-encryption: aws:kms
            Sid: DenyInsecureConnections
            Effect: Deny
            Principal: '*'
            Action: s3:*
            Resource: !Sub ${CodePipelineArtifactStoreBucket.Arn}/*
```

```
Condition:
            Bool:
              aws:SecureTransport: false
CodePipelineServiceRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: 2012-10-17
      Statement:
          Effect: Allow
          Principal:
            Service:
              - codepipeline.amazonaws.com
          Action: sts:AssumeRole
    Path: /
    Policies:
        PolicyName: AWS-CodePipeline-Service-3
        PolicyDocument:
          Version: 2012-10-17
          Statement:
              Effect: Allow
              Action:
                - codecommit:CancelUploadArchive
                - codecommit:GetBranch
                - codecommit:GetCommit
                - codecommit:GetUploadArchiveStatus
                - codecommit:UploadArchive
              Resource: 'resource_ARN'
              Effect: Allow
              Action:
                - codedeploy:CreateDeployment
                - codedeploy:GetApplicationRevision
                - codedeploy:GetDeployment
                - codedeploy:GetDeploymentConfig
                - codedeploy:RegisterApplicationRevision
              Resource: 'resource_ARN'
              Effect: Allow
              Action:
                - codebuild:BatchGetBuilds
```

```
- codebuild:StartBuild
              Resource: 'resource_ARN'
              Effect: Allow
              Action:
                - devicefarm:ListProjects
                - devicefarm:ListDevicePools
                - devicefarm:GetRun
                - devicefarm:GetUpload
                - devicefarm:CreateUpload
                - devicefarm:ScheduleRun
              Resource: 'resource_ARN'
              Effect: Allow
              Action:
                - lambda:InvokeFunction
                - lambda:ListFunctions
              Resource: 'resource_ARN'
              Effect: Allow
              Action:
                - iam:PassRole
              Resource: 'resource_ARN'
              Effect: Allow
              Action:
                - elasticbeanstalk:*
                - ec2:*
                - elasticloadbalancing:*
                - autoscaling:*
                - cloudwatch:*
                - s3:*
                - sns:*
                - cloudformation:*
                - rds:*
                - sqs:*
                - ecs:*
              Resource: 'resource_ARN'
AppPipeline:
  Type: AWS::CodePipeline::Pipeline
  Properties:
    Name: s3-events-pipeline
    RoleArn:
      !GetAtt CodePipelineServiceRole.Arn
```

```
Stages:
      _
        Name: Source
        Actions:
          _
            Name: SourceAction
            ActionTypeId:
              Category: Source
              Owner: AWS
              Version: 1
              Provider: S3
            OutputArtifacts:
              - Name: SourceOutput
            Configuration:
              S3Bucket: !Ref SourceBucket
              S3ObjectKey: !Ref SourceObjectKey
              PollForSourceChanges: false
            RunOrder: 1
        Name: Beta
        Actions:
            Name: BetaAction
            InputArtifacts:
              - Name: SourceOutput
            ActionTypeId:
              Category: Deploy
              Owner: AWS
              Version: 1
              Provider: CodeDeploy
            Configuration:
              ApplicationName: !Ref ApplicationName
              DeploymentGroupName: !Ref BetaFleet
            RunOrder: 1
    ArtifactStore:
      Type: S3
      Location: !Ref CodePipelineArtifactStoreBucket
EventRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: 2012-10-17
      Statement:
```

```
Effect: Allow
           Principal:
             Service:
               - events.amazonaws.com
           Action: sts:AssumeRole
     Path: /
     Policies:
         PolicyName: eb-pipeline-execution
         PolicyDocument:
           Version: 2012-10-17
           Statement:
               Effect: Allow
               Action: codepipeline:StartPipelineExecution
               Resource: !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region',
':', !Ref 'AWS::AccountId', ':', !Ref AppPipeline ] ]
EventRule:
  Type: AWS::Events::Rule
  Properties:
     EventBusName: default
     EventPattern:
       source:
         - aws.s3
       detail-type:
         - Object Created
       detail:
         bucket:
           name:
             - !Ref SourceBucket
     Name: EnabledS3SourceRule
    State: ENABLED
    Targets:
         Arn:
           !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region', ':', !Ref
'AWS::AccountId', ':', !Ref AppPipeline ] ]
         RoleArn: !GetAtt EventRole.Arn
         Id: codepipeline-AppPipeline
```
{

```
"Parameters": {
    "SourceObjectKey": {
        "Description": "S3 source artifact",
        "Type": "String",
        "Default": "SampleApp_Linux.zip"
    },
    "ApplicationName": {
        "Description": "CodeDeploy application name",
        "Type": "String",
        "Default": "DemoApplication"
    },
    "BetaFleet": {
        "Description": "Fleet configured in CodeDeploy",
        "Type": "String",
        "Default": "DemoFleet"
    }
},
"Resources": {
    "SourceBucket": {
        "Type": "AWS::S3::Bucket",
          "Properties": {
            "NotificationConfiguration": {
                "EventBridgeConfiguration": {
                     "EventBridgeEnabled": true
                }
            },
            "VersioningConfiguration": {
                "Status": "Enabled"
            }
        }
    },
    "CodePipelineArtifactStoreBucket": {
        "Type": "AWS::S3::Bucket"
    },
    "CodePipelineArtifactStoreBucketPolicy": {
        "Type": "AWS::S3::BucketPolicy",
        "Properties": {
            "Bucket": {
                "Ref": "CodePipelineArtifactStoreBucket"
            },
            "PolicyDocument": {
```

```
"Version": "2012-10-17",
"Statement": [
    {
        "Sid": "DenyUnEncryptedObjectUploads",
        "Effect": "Deny",
        "Principal": "*",
        "Action": "s3:PutObject",
        "Resource": {
            "Fn::Join": [
                "",
                Ε
                     {
                         "Fn::GetAtt": [
                             "CodePipelineArtifactStoreBucket",
                             "Arn"
                         ]
                     },
                     "/*"
                ]
            ]
        },
        "Condition": {
            "StringNotEquals": {
                "s3:x-amz-server-side-encryption": "aws:kms"
            }
        }
   },
    {
        "Sid": "DenyInsecureConnections",
        "Effect": "Deny",
        "Principal": "*",
        "Action": "s3:*",
        "Resource": {
            "Fn::Join": [
                "",
                Ε
                     {
                         "Fn::GetAtt": [
                             "CodePipelineArtifactStoreBucket",
                             "Arn"
                         ]
                     },
                     "/*"
                ]
```

```
]
                    },
                     "Condition": {
                         "Bool": {
                             "aws:SecureTransport": false
                         }
                    }
                }
            ]
        }
    }
},
"CodePipelineServiceRole": {
    "Type": "AWS:::IAM::Role",
    "Properties": {
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                     "Principal": {
                         "Service": [
                             "codepipeline.amazonaws.com"
                         ]
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        },
        "Path": "/",
        "Policies": [
            {
                 "PolicyName": "AWS-CodePipeline-Service-3",
                "PolicyDocument": {
                     "Version": "2012-10-17",
                     "Statement": [
                         {
                             "Effect": "Allow",
                             "Action": [
                                 "codecommit:CancelUploadArchive",
                                 "codecommit:GetBranch",
                                 "codecommit:GetCommit",
                                 "codecommit:GetUploadArchiveStatus",
                                 "codecommit:UploadArchive"
```

사용자 가이드

```
],
    "Resource": "resource_ARN"
},
{
    "Effect": "Allow",
    "Action": [
        "codedeploy:CreateDeployment",
        "codedeploy:GetApplicationRevision",
        "codedeploy:GetDeployment",
        "codedeploy:GetDeploymentConfig",
        "codedeploy:RegisterApplicationRevision"
    ],
    "Resource": "resource_ARN"
},
{
    "Effect": "Allow",
    "Action": [
        "codebuild:BatchGetBuilds",
        "codebuild:StartBuild"
    ],
    "Resource": "resource_ARN"
},
{
    "Effect": "Allow",
    "Action": [
        "devicefarm:ListProjects",
        "devicefarm:ListDevicePools",
        "devicefarm:GetRun",
        "devicefarm:GetUpload",
        "devicefarm:CreateUpload",
        "devicefarm:ScheduleRun"
    ],
    "Resource": "resource_ARN"
},
{
    "Effect": "Allow",
    "Action": [
        "lambda:InvokeFunction",
        "lambda:ListFunctions"
    ],
    "Resource": "resource_ARN"
},
{
    "Effect": "Allow",
```

```
"Action": [
                                  "iam:PassRole"
                             ],
                             "Resource": "resource_ARN"
                         },
                         {
                             "Effect": "Allow",
                             "Action": [
                                  "elasticbeanstalk:*",
                                  "ec2:*",
                                  "elasticloadbalancing:*",
                                  "autoscaling:*",
                                 "cloudwatch:*",
                                  "s3:*",
                                  "sns:*",
                                  "cloudformation:*",
                                  "rds:*",
                                 "sqs:*",
                                  "ecs:*"
                             ],
                             "Resource": "resource_ARN"
                         }
                     ]
                }
            }
        ]
    }
},
"AppPipeline": {
    "Type": "AWS::CodePipeline::Pipeline",
    "Properties": {
        "Name": "s3-events-pipeline",
        "RoleArn": {
            "Fn::GetAtt": [
                 "CodePipelineServiceRole",
                 "Arn"
            ]
        },
        "Stages": [
            {
                 "Name": "Source",
                 "Actions": [
                     {
                         "Name": "SourceAction",
```

```
"ActionTypeId": {
                "Category": "Source",
                "Owner": "AWS",
                "Version": 1,
                "Provider": "S3"
            },
            "OutputArtifacts": [
                {
                     "Name": "SourceOutput"
                }
            ],
            "Configuration": {
                "S3Bucket": {
                     "Ref": "SourceBucket"
                },
                "S3ObjectKey": {
                    "Ref": "SourceObjectKey"
                },
                "PollForSourceChanges": false
            },
            "RunOrder": 1
        }
    ]
},
{
    "Name": "Beta",
    "Actions": [
        {
            "Name": "BetaAction",
            "InputArtifacts": [
                {
                     "Name": "SourceOutput"
                }
            ],
            "ActionTypeId": {
                "Category": "Deploy",
                "Owner": "AWS",
                "Version": 1,
                "Provider": "CodeDeploy"
            },
            "Configuration": {
                "ApplicationName": {
                     "Ref": "ApplicationName"
                },
```

```
"DeploymentGroupName": {
                                 "Ref": "BetaFleet"
                             }
                         },
                         "RunOrder": 1
                    }
                ]
            }
        ],
        "ArtifactStore": {
            "Type": "S3",
            "Location": {
                "Ref": "CodePipelineArtifactStoreBucket"
            }
        }
    }
},
"EventRole": {
    "Type": "AWS::IAM::Role",
    "Properties": {
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                         "Service": [
                             "events.amazonaws.com"
                         ]
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        },
        "Path": "/",
        "Policies": [
            {
                "PolicyName": "eb-pipeline-execution",
                "PolicyDocument": {
                     "Version": "2012-10-17",
                    "Statement": [
                         {
                             "Effect": "Allow",
                             "Action": "codepipeline:StartPipelineExecution",
```

```
"Resource": {
                                 "Fn::Join": [
                                     "",
                                     Ε
                                          "arn:aws:codepipeline:",
                                          {
                                              "Ref": "AWS::Region"
                                          },
                                         ":",
                                          {
                                             "Ref": "AWS::AccountId"
                                          },
                                          ":",
                                          {
                                              "Ref": "AppPipeline"
                                          }
                                     ]
                                 ]
                             }
                         }
                    ]
                }
            }
        ]
    }
},
"EventRule": {
    "Type": "AWS::Events::Rule",
    "Properties": {
        "EventBusName": "default",
  "EventPattern": {
      "source": [
    "aws.s3"
  ],
        "detail-type": [
"Object Created"
 ],
  "detail": {
"bucket": {
                 "name": [
                      {
                      "Ref": "SourceBucket"
                      }
```



S3 소스 및 CloudTrail 트레일을 사용하여 폴링 파이프라인 마이그레이션

Amazon S3 소스가 있는 파이프라인의 경우 EventBridge를 통해 변경 감지가 자동화되도록 파이프라 인을 수정합니다. 다음 방법 중에서 선택하여 마이그레이션을 구현하세요.

- 콘솔: 폴링 파이프라인 마이그레이션(CodeCommit 또는 Amazon S3 소스)(콘솔)
- CLI: S3 소스 및 CloudTrail 트레일을 사용하여 폴링 파이프라인 마이그레이션(CLI)
- AWS CloudFormation: <u>S3 소스 및 CloudTrail 추적을 사용하여 폴링 파이프라인 마이그레이션(AWS</u> CloudFormation 템플릿)

S3 소스 및 CloudTrail 트레일을 사용하여 폴링 파이프라인 마이그레이션(CLI)

대신 EventBridge의 이벤트를 사용하도록 폴링(정기적 확인)을 사용하는 파이프라인을 편집하려면 다 음 단계를 수행합니다. 파이프라인을 생성하려면 <u>파이프라인 스테이지 및 작업 생성</u> 단원을 참조하십 시오.

Amazon S3를 사용하여 이벤트 기반 파이프라인을 빌드하려면 파이프라인의 PollForSourceChanges 파라미터를 편집한 후 다음 리소스를 생성합니다.

- AWS CloudTrail Amazon S3가 이벤트를 로깅하는 데 사용할 수 있는 추적, 버킷 및 버킷 정책입니다.
- EventBridge 이벤트
- EventBridge 이벤트가 파이프라인을 시작하도록 허용하는 IAM 역할

AWS CloudTrail 추적을 생성하고 로깅을 활성화하려면

AWS CLI 를 사용하여 추적을 생성하려면 다음을 지정하여 create-trail 명령을 호출합니다.

- 추적 이름입니다.
- AWS CloudTrail에 대한 버킷 정책을 이미 적용한 버킷입니다.

자세한 내용은 AWS 명령줄 인터페이스를 사용하여 추적 생성을 참조하세요.

1. create-trail 명령을 호출하고 --name 및 --s3-bucket-name 파라미터를 포함시킵니다.

이렇게 변경하는 이유는 무엇입니까? 이렇게 하면 S3 소스 버킷에 필요한 CloudTrail 추적이 생성 됩니다. 다음 명령은 --name 및 --s3-bucket-name을 사용하여 my-trail이라는 추적과 amzn-s3demo-source-bucket이라는 버킷을 생성합니다.

aws cloudtrail create-trail --name my-trail --s3-bucket-name amzn-s3-demo-sourcebucket

2. start-logging 명령을 호출하고 - - name 파라미터를 포함시킵니다.

이렇게 변경하는 이유는 무엇입니까? 이 명령은 소스 버킷에 대한 CloudTrail 로깅을 시작하고, EventBridge에 이벤트를 전송합니다.

예제:

다음 명령은 --name을 사용하여 my-trail이라는 추적에서 로깅을 시작합니다.

aws cloudtrail start-logging --name my-trail

 put-event-selectors 명령을 호출하고 --trail-name 및 --event-selectors 파라미터를 포함 시킵니다. 이벤트 선택기를 사용하여 추적에서 소스 버킷에 대한 데이터 이벤트를 기록하고 이벤 트를 EventBridge 규칙으로 전송하도록 지정합니다.

이렇게 변경하는 이유는 무엇입니까? 이 명령은 이벤트를 필터링합니다.

예제:

다음 명령은 --trail-name 및 --event-selectors를 사용하여 소스 버킷 및 amzn-s3demo-source-bucket/myFolder라는 접두사에 대한 데이터 이벤트 관리를 지정합니다.

```
aws cloudtrail put-event-selectors --trail-name my-trail --event-selectors
'[{ "ReadWriteType": "WriteOnly", "IncludeManagementEvents":false,
"DataResources": [{ "Type": "AWS::S3::Object", "Values": ["arn:aws:s3:::amzn-s3-
demo-source-bucket/myFolder/file.zip"] }] }]'
```

Amazon S3가 이벤트 소스이고 CodePipeline이 대상인 EventBridge 규칙을 생성하고 권한 정책을 적 용하려면

 CodePipeline을 사용하여 규칙을 호출하도록 EventBridge에 권한을 부여합니다. 자세한 내용은 <u>Amazon EventBridge에 대한 리소스 기반 정책 사용</u>을 참조하세요. a. 다음 예제를 사용하여 EventBridge가 서비스 역할을 담당하도록 허용하는 신뢰 정책을 생성 합니다. 이름을 trustpolicyforEB.json로 지정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
               "Service": "events.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
        }
   ]
}
```

b. 다음 명령을 사용하여 Role-for-MyRule 역할을 생성한 후 신뢰 정책에 연결합니다.

이렇게 변경하는 이유는 무엇입니까? 이 신뢰 정책을 역할에 추가하면 EventBridge에 대한 권한이 생성됩니다.

aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document file://trustpolicyforEB.json

c. 이 샘플에서 보이는 것처럼 MyFirstPipeline이라는 파이프라인에 대한 권한 정책 JSON 을 생성합니다. 권한 정책 이름을 permissionspolicyforEB.json으로 지정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "codepipeline:StartPipelineExecution"
        ],
            "Resource": [
               "arn:aws:codepipeline:us-west-2:80398EXAMPLE:MyFirstPipeline"
        ]
        }
    ]
}
```

d. 다음 명령을 사용하여 앞에서 생성한 Role-for-MyRule 역할에 새로운 CodePipeline-Permissions-Policy-for-EB 권한 정책을 연결합니다.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. put-rule 명령을 호출하고 --name, --event-pattern 및 --role-arn 파라미터를 포함시킵니 다.

다음 샘플 명령은 MyS3SourceRule이라는 역할 별칭을 생성합니다.

```
aws events put-rule --name "MyS3SourceRule" --event-pattern "{\"source\":
[\"aws.s3\"],\"detail-type\":[\"AWS API Call via CloudTrail\"],\"detail\":
{\"eventSource\":[\"s3.amazonaws.com\"],\"eventName\":[\"CopyObject\",\"PutObject
\",\"CompleteMultipartUpload\"],\"requestParameters\":{\"bucketName\":[\"amzn-s3-
demo-source-bucket\"],\"key\":[\"my-key\"]}}
    --role-arn "arn:aws:iam::ACCOUNT_ID:role/Role-for-MyRule"
```

3. CodePipeline을 대상으로 추가하려면 put-targets 명령을 호출하고 --rule 및 --targets 파라 미터를 포함합니다.

다음 명령은 MyS3SourceRule이라는 규칙에 대해 대상 Id가 숫자 1로 구성됨을 지정하며, 규칙 에 대한 대상 목록에서 대상 1로 표시됩니다. 이 명령은 또한 파이프라인에 대한 예제 ARN를 지정 합니다. 파이프라인은 리포지토리에서 변경이 발생하면 시작됩니다.

aws events put-targets --rule MyS3SourceRule --targets Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline

- (선택 사항) 특정 이미지 ID에 대한 소스 재정의로 입력 변환기를 구성하려면 CLI 명령에서 다음 JSON을 사용합니다. 다음 예제에서는 다음과 같은 재정의를 구성합니다.
 - 이 예제actionNameSource에서는 소스 이벤트에서 파생되지 않은 파이프라인 생성 시 정의 된 동적 값입니다.
 - 이 예제revisionTypeS3_OBJECT_VERSION_ID에서는 소스 이벤트에서 파생되지 않은 파이 프라인 생성 시 정의된 동적 값입니다.
 - 이 예제의, revisionValue<*revisionValue*>는 소스 이벤트 변수에서 파생됩니다.

{

[&]quot;Rule": "my-rule",

```
"Targets": [
        {
            "Id": "MyTargetId",
            "Arn": "ARN",
            "InputTransformer": {
                "InputPathsMap": {
                     "revisionValue": "$.detail.object.version-id"
                },
                "InputTemplate": {
                     "sourceRevisions": {
                         "actionName": "Source",
                         "revisionType": "S3_OBJECT_VERSION_ID",
                         "revisionValue": "<revisionValue>"
                    }
                }
            }
        }
    ]
}
```

파이프라인의 PollForSourceChanges 파라미터를 편집하려면

A Important

이 방법으로 파이프라인을 생성할 때 명시적으로 false로 설정되지 않은 경우 PollForSourceChanges 파라미터 기본값은 true입니다. 이벤트 기반 변경 감지를 추가할 때 는 출력에 파라미터를 추가하고 false로 설정하여 폴링을 비활성화해야 합니다. 그렇지 않으면 파이프라인이 단일 소스 변경 시 두 번 시작됩니다. 세부 정보는 <u>PollForSourceChanges 파</u> 라미터의 유효한 설정을 참조하세요.

1. get-pipeline 명령을 실행하여 파이프라인 구조를 JSON 파일로 복사합니다. 예를 들어, MyFirstPipeline라는 파이프라인의 경우 다음 명령을 입력합니다.

aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json

이 명령은 아무 것도 반환하지 않지만 생성한 파일이 명령을 실행한 디렉터리에 표시되어야 합니 다. 2. 일반 텍스트 편집기에서 JSON 파일을 열고 다음 예에 나와 있는 것처럼 amzn-s3-demosource-bucket 버킷의 PollForSourceChanges 파라미터를 false로 변경하여 소스 단계를 편집합니다.

이렇게 변경하는 이유는 무엇입니까? 이 파라미터를 false로 설정하면 정기적 확인이 비활성화 되어 이벤트 기반 변경 탐지만 사용할 수 있습니다.

```
"configuration": {
    "S3Bucket": "amzn-s3-demo-source-bucket",
    "PollForSourceChanges": "false",
    "S3ObjectKey": "index.zip"
},
```

 get-pipeline 명령을 사용하여 검색한 파이프라인 구조로 작업을 수행할 경우, JSON 파일에서 metadata 행을 제거해야 합니다. 이렇게 하지 않으면 update-pipeline 명령에서 사용할 수 없습니 다. "metadata": { } 행과, "created", "pipelineARN" 및 "updated" 필드를 제거합니다.

예를 들어, 구조에서 다음 행을 삭제합니다.

```
"metadata": {
    "pipelineArn": "arn:aws:codepipeline:region:account-ID:pipeline-name",
    "created": "date",
    "updated": "date"
},
```

파일을 저장합니다.

4. 변경 사항을 적용하려면 파이프라인 JSON 파일을 지정하여 update-pipeline 명령을 실행합니다.

Important

파일 이름 앞에 file://를 포함해야 합니다. 이 명령에 필수적입니다.

aws codepipeline update-pipeline --cli-input-json file://pipeline.json

이 명령은 편집한 파이프라인의 전체 구조를 반환합니다.

Note

update-pipeline 명령을 실행하면 파이프라인이 중지됩니다. update-pipeline 명령을 실행 할 때 파이프라인을 통해 개정을 실행하는 중이라면 해당 실행이 중지됩니다. 업데이트 된 파이프라인을 통해 해당 개정을 실행하려면 파이프라인을 수동으로 시작해야 합니다. start-pipeline-execution 명령을 사용하여 수동으로 파이프라인을 시작합니다.

S3 소스 및 CloudTrail 추적을 사용하여 폴링 파이프라인 마이그레이션(AWS CloudFormation 템플릿)

폴링에서 이벤트 기반 변경 감지로 Amazon S3 소스가 있는 파이프라인을 편집하려면 다음 단계를 수 행합니다.

Amazon S3를 사용하여 이벤트 기반 파이프라인을 빌드하려면 해당 파이프라인의 PollForSourceChanges 파라미터를 편집한 후 다음 리소스를 템플릿에 추가합니다.

- EventBridge는 모든 Amazon S3 이벤트를 로깅해야 합니다. Amazon S3가 발생 이벤트를 로깅하기 위해 사용할 수 있는 AWS CloudTrail 추적, 버킷 및 버킷 정책을 생성해야 합니다. 자세한 내용은 <u>추</u> 적에 대한 데이터 이벤트 로깅 및 추적에 대한 관리 이벤트 로깅을 참조하세요.
- 이 이벤트가 파이프라인을 시작할 수 있게 하는 EventBridge 규칙 및 IAM 역할.

AWS CloudFormation 를 사용하여 파이프라인을 생성하고 관리하는 경우 템플릿에는 다음과 같은 콘 텐츠가 포함됩니다.

Note

PollForSourceChanges라고 하는 소스 단계의 Configuration 속성. 템플릿에 해당 속성 이 포함되어 있지 않으면 기본적으로 PollForSourceChanges가 true로 설정됩니다.

YAML

```
AppPipeline:
Type: AWS::CodePipeline::Pipeline
Properties:
RoleArn: !GetAtt CodePipelineServiceRole.Arn
```

```
Stages:
        _
          Name: Source
          Actions:
            _
              Name: SourceAction
              ActionTypeId:
                Category: Source
                Owner: AWS
                Version: 1
                Provider: S3
              OutputArtifacts:
                  Name: SourceOutput
              Configuration:
                S3Bucket: !Ref SourceBucket
                S3ObjectKey: !Ref S3SourceObjectKey
                PollForSourceChanges: true
              RunOrder: 1
. . .
```

```
"AppPipeline": {
    "Type": "AWS::CodePipeline::Pipeline",
   "Properties": {
        "RoleArn": {
            "Fn::GetAtt": ["CodePipelineServiceRole", "Arn"]
        },
       "Stages": [
            {
                "Name": "Source",
                "Actions": [
                    {
                        "Name": "SourceAction",
                        "ActionTypeId": {
                             "Category": "Source",
                             "Owner": "AWS",
                             "Version": 1,
                            "Provider": "S3"
                        },
```

```
"OutputArtifacts": [
                                      {
                                          "Name": "SourceOutput"
                                      }
                                 ],
                                  "Configuration": {
                                      "S3Bucket": {
                                          "Ref": "SourceBucket"
                                      },
                                      "S3ObjectKey": {
                                          "Ref": "SourceObjectKey"
                                      },
                                      "PollForSourceChanges": true
                                 },
                                  "RunOrder": 1
                             }
                         ]
                     },
. . .
```

Amazon S3가 이벤트 소스이고 CodePipeline이 대상인 EventBridge 규칙을 생성하고 권한 정책을 적 용하려면

- 템플릿의에서 AWS::IAM::Role AWS CloudFormation 리소스를 Resources사용하여 이벤트가 파이프라인을 시작하도록 허용하는 IAM 역할을 구성합니다. 이 항목은 두 가지 정책을 사용하는 역할을 만듭니다.
 - 첫 번째 정책은 가 역할을 수임하도록 허용합니다.
 - 두 번째 정책은 파이프라인을 시작할 권한을 부여합니다.

이렇게 변경하는 이유는 무엇입니까? AWS::IAM::Role 리소스를 추가하면 AWS CloudFormation 가 EventBridge에 대한 권한을 생성할 수 있습니다. 이 리소스는 AWS CloudFormation 스택에 추가됩니다.

YAML

EventRole: Type: AWS::IAM::Role

```
Properties:
     AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          _
            Effect: Allow
            Principal:
              Service:
                - events.amazonaws.com
            Action: sts:AssumeRole
     Path: /
     Policies:
          PolicyName: eb-pipeline-execution
          PolicyDocument:
            Version: 2012-10-17
            Statement:
                Effect: Allow
                Action: codepipeline:StartPipelineExecution
                Resource: !Join [ '', [ 'arn:aws:codepipeline:', !Ref
'AWS::Region', ':', !Ref 'AWS::AccountId', ':', !Ref AppPipeline ] ]
. . .
```

```
]
      },
      "Path": "/",
      "Policies": [
        {
          "PolicyName": "eb-pipeline-execution",
          "PolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
              {
                 "Effect": "Allow",
                 "Action": "codepipeline:StartPipelineExecution",
                 "Resource": {
                   "Fn::Join": [
                     "",
                     Ε
                       "arn:aws:codepipeline:",
                       {
                         "Ref": "AWS::Region"
                       },
                       ":",
                       {
                         "Ref": "AWS::AccountId"
                       },
                       ":",
                       {
                         "Ref": "AppPipeline"
                       }
                     ]
                   ]
. . .
```

2. AWS::Events::Rule AWS CloudFormation 리소스를 사용하여 EventBridge 규칙을 추가합니다. 이 이벤트 패턴은 Amazon S3 소스 버킷의 CopyObject, PutObject 및 CompleteMultipartUpload를 모니터링하는 이벤트를 생성합니다. 또한 파이프라인 대상도 포함하세요. 이 규칙은 CopyObject, PutObject 또는 CompleteMultipartUpload 발생 시 대상 파이프라인에서 StartPipelineExecution을 호출합니다.

이렇게 변경하는 이유는 무엇입니까? AWS::Events::Rule 리소스를 추가 AWS CloudFormation 하면가 이벤트를 생성할 수 있습니다. 이 리소스는 AWS CloudFormation 스택에 추가됩니다.

YAML

```
EventRule:
   Type: AWS::Events::Rule
   Properties:
     EventPattern:
        source:
          - aws.s3
        detail-type:
          - 'AWS API Call via CloudTrail'
        detail:
          eventSource:
            - s3.amazonaws.com
          eventName:
            - CopyObject
            - PutObject
            - CompleteMultipartUpload
          requestParameters:
            bucketName:
              - !Ref SourceBucket
            key:
              - !Ref SourceObjectKey
     Targets:
        -
          Arn:
            !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region', ':', !Ref
'AWS::AccountId', ':', !Ref AppPipeline ] ]
          RoleArn: !GetAtt EventRole.Arn
          Id: codepipeline-AppPipeline
. . .
```

```
"EventRule": {
   "Type": "AWS::Events::Rule",
   "Properties": {
    "EventPattern": {
        "source": [
            "aws.s3"
```

```
],
  "detail-type": [
    "AWS API Call via CloudTrail"
  ],
  "detail": {
    "eventSource": [
      "s3.amazonaws.com"
    ],
    "eventName": [
      "CopyObject",
      "PutObject",
      "CompleteMultipartUpload"
    ],
    "requestParameters": {
      "bucketName": [
        {
          "Ref": "SourceBucket"
        }
      ],
      "key": [
        {
          "Ref": "SourceObjectKey"
        }
      ]
    }
  }
},
"Targets": [
  {
    "Arn": {
      "Fn::Join": [
        "",
        Ε
          "arn:aws:codepipeline:",
          {
            "Ref": "AWS::Region"
          },
          ":",
          {
            "Ref": "AWS::AccountId"
          },
          ":",
          {
            "Ref": "AppPipeline"
```

```
}
               ]
             ]
           },
           "RoleArn": {
             "Fn::GetAtt": [
               "EventRole",
               "Arn"
             ]
           },
           "Id": "codepipeline-AppPipeline"
        }
      ]
    }
 }
},
. . .
```

3. 이 코드 조각을 첫 번째 템플릿에 추가하여 교차 스택 기능을 허용하세요.

YAML

```
Outputs:
SourceBucketARN:
Description: "S3 bucket ARN that Cloudtrail will use"
Value: !GetAtt SourceBucket.Arn
Export:
Name: SourceBucketARN
```

```
"Outputs" : {
    "SourceBucketARN" : {
        "Description" : "S3 bucket ARN that Cloudtrail will use",
        "Value" : { "Fn::GetAtt": ["SourceBucket", "Arn"] },
        "Export" : {
            "Name" : "SourceBucketARN"
        }
    }
....
```

- (선택 사항) 특정 이미지 ID에 대한 소스 재정의로 입력 변환기를 구성하려면 다음 YAML 코드 조 각을 사용합니다. 다음 예제에서는 다음과 같은 재정의를 구성합니다.
 - 이 예제actionNameSource에서는 소스 이벤트에서 파생되지 않은 파이프라인 생성 시 정의 된 동적 값입니다.
 - 이 예제revisionTypeS3_OBJECT_VERSION_ID에서는 소스 이벤트에서 파생되지 않은 파이 프라인 생성 시 정의된 동적 값입니다.
 - 이 예제의, revisionValue<revisionValue>는 소스 이벤트 변수에서 파생됩니다.

```
---
Rule: my-rule
Targets:
- Id: MyTargetId
Arn: pipeline-ARN
InputTransformer:
InputPathsMap:
revisionValue: "$.detail.object.version-id"
InputTemplate:
sourceRevisions:
actionName: Source
revisionType: S3_OBJECT_VERSION_ID
revisionValue: '<revisionValue>'
```

- 5. 업데이트된 템플릿을 로컬 컴퓨터에 저장하고 AWS CloudFormation 콘솔을 엽니다.
- 6. 스택을 선택한 후 현재 스택에 대한 변경 세트 만들기를 선택합니다.
- 업데이트된 템플릿을 업로드한 후 AWS CloudFormation에 나열된 변경 사항을 확인합니다. 이는 스택에 적용될 변경 사항입니다. 목록에 새로운 리소스가 표시됩니다.
- 8. 실행을 선택합니다.

파이프라인의 PollForSourceChanges 파라미터를 편집하려면

A Important

이 방법으로 파이프라인을 생성할 때 명시적으로 false로 설정되지 않은 경우 PollForSourceChanges 파라미터 기본값은 true입니다. 이벤트 기반 변경 감지를 추가할 때 는 출력에 파라미터를 추가하고 false로 설정하여 폴링을 비활성화해야 합니다. 그렇지 않으면 파이프라인이 단일 소스 변경 시 두 번 시작됩니다. 세부 정보는 <u>PollForSourceChanges 파</u> 라미터의 유효한 설정을 참조하세요.

템플릿에서 PollForSourceChanges를 false로 변경합니다. PollForSourceChanges를 파 이프라인 정의에 포함하지 않은 경우 추가하고 false로 설정하세요.

이렇게 변경하는 이유는 무엇입니까? PollForSourceChanges를 false로 변경하면 정기적 확 인이 비활성화되어 이벤트 기반 변경 탐지만 사용할 수 있습니다.

YAML

```
Name: Source
Actions:
-
Name: SourceAction
ActionTypeId:
Category: Source
Owner: AWS
Version: 1
Provider: S3
OutputArtifacts:
- Name: SourceOutput
Configuration:
S3Bucket: !Ref SourceBucket
S3ObjectKey: !Ref SourceObjectKey
PollForSourceChanges: false
RunOrder: 1
```

```
}
],
"Configuration": {
    "S3Bucket": {
        "Ref": "SourceBucket"
    },
    "S3ObjectKey": {
        "Ref": "SourceObjectKey"
    },
    "PollForSourceChanges": false
},
"RunOrder": 1
}
```

Amazon S3 파이프라인의 CloudTrail 리소스를 위한 두 번째 템플릿을 만들려면

 별도의 템플릿에서 Resources, AWS::S3::Bucket AWS::S3::BucketPolicy및 AWS::CloudTrail::Trail AWS CloudFormation 리소스를 사용하여 CloudTrail에 대한 간단 한 버킷 정의 및 추적을 제공합니다.

이렇게 변경하는 이유는 무엇입니까? 현재 계정당 5개의 추적으로 제한되어 있으므로 CloudTrail 추적을 별도로 생성하고 관리해야 합니다. (<u>의 제한 AWS CloudTrail</u> 참조) 하지만 단일 추적에 많 은 Amazon S3 버킷을 포함할 수 있으므로 추적을 한 번 생성한 다음 필요에 따라 다른 파이프라 인용 Amazon S3 버킷을 추가할 수 있습니다. 다음 내용을 두 번째 샘플 템플릿 파일에 붙여넣습 니다.

YAML

```
AWSCloudTrailBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket: !Ref AWSCloudTrailBucket
      PolicyDocument:
        Version: 2012-10-17
        Statement:
            Sid: AWSCloudTrailAclCheck
            Effect: Allow
            Principal:
              Service:
                - cloudtrail.amazonaws.com
            Action: s3:GetBucketAcl
            Resource: !GetAtt AWSCloudTrailBucket.Arn
            Sid: AWSCloudTrailWrite
            Effect: Allow
            Principal:
              Service:
                - cloudtrail.amazonaws.com
            Action: s3:PutObject
            Resource: !Join [ '', [ !GetAtt AWSCloudTrailBucket.Arn, '/
AWSLogs/', !Ref 'AWS::AccountId', '/*' ] ]
            Condition:
              StringEquals:
                s3:x-amz-acl: bucket-owner-full-control
 AWSCloudTrailBucket:
    Type: AWS::S3::Bucket
    DeletionPolicy: Retain
 AwsCloudTrail:
    DependsOn:

    AWSCloudTrailBucketPolicy

    Type: AWS::CloudTrail::Trail
    Properties:
      S3BucketName: !Ref AWSCloudTrailBucket
      EventSelectors:
          DataResources:
              Type: AWS::S3::Object
              Values:
                - !Join [ '', [ !ImportValue SourceBucketARN, '/', !Ref
 SourceObjectKey ] ]
```

```
ReadWriteType: WriteOnly
IncludeManagementEvents: false
IncludeGlobalServiceEvents: true
IsLogging: true
IsMultiRegionTrail: true
```

JSON

• • •

```
{
  "Parameters": {
    "SourceObjectKey": {
      "Description": "S3 source artifact",
      "Type": "String",
      "Default": "SampleApp_Linux.zip"
    }
  },
  "Resources": {
    "AWSCloudTrailBucket": {
      "Type": "AWS::S3::Bucket",
        "DeletionPolicy": "Retain"
    },
    "AWSCloudTrailBucketPolicy": {
      "Type": "AWS::S3::BucketPolicy",
      "Properties": {
        "Bucket": {
          "Ref": "AWSCloudTrailBucket"
        },
        "PolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Sid": "AWSCloudTrailAclCheck",
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  "cloudtrail.amazonaws.com"
                ]
              },
              "Action": "s3:GetBucketAcl",
              "Resource": {
```

```
"Fn::GetAtt": [
              "AWSCloudTrailBucket",
              "Arn"
            ]
          }
        },
        {
          "Sid": "AWSCloudTrailWrite",
          "Effect": "Allow",
          "Principal": {
            "Service": [
              "cloudtrail.amazonaws.com"
            ]
          },
          "Action": "s3:PutObject",
          "Resource": {
            "Fn::Join": [
              "",
              Γ
                {
                   "Fn::GetAtt": [
                    "AWSCloudTrailBucket",
                     "Arn"
                  ]
                },
                 "/AWSLogs/",
                 {
                  "Ref": "AWS::AccountId"
                },
                "/*"
              ]
            ]
          },
          "Condition": {
            "StringEquals": {
              "s3:x-amz-acl": "bucket-owner-full-control"
            }
          }
        }
      ]
    }
 }
},
"AwsCloudTrail": {
```

```
"DependsOn": [
        "AWSCloudTrailBucketPolicy"
      ],
      "Type": "AWS::CloudTrail::Trail",
      "Properties": {
        "S3BucketName": {
          "Ref": "AWSCloudTrailBucket"
        },
        "EventSelectors": [
          {
            "DataResources": [
              {
                "Type": "AWS::S3::Object",
                "Values": [
                  {
                     "Fn::Join": [
                       "",
                       Ε
                         {
                           "Fn::ImportValue": "SourceBucketARN"
                         },
                         "/",
                         {
                           "Ref": "SourceObjectKey"
                         }
                       ]
                     ]
                   }
                ]
              }
            ],
            "ReadWriteType": "WriteOnly",
            "IncludeManagementEvents": false
          }
        ],
        "IncludeGlobalServiceEvents": true,
        "IsLogging": true,
        "IsMultiRegionTrail": true
      }
    }
 }
}
```

•••

Example

AWS CloudFormation 를 사용하여 이러한 리소스를 생성하면 리포지토리의 파일이 생성되거나 업데 이트될 때 파이프라인이 트리거됩니다.

1 Note

여기에서 중단하지 마십시오. 파이프라인이 생성되더라도 Amazon S3 파이프라인에 대한 두 번째 AWS CloudFormation 템플릿을 생성해야 합니다. 두 번째 템플릿을 생성하지 않으면 파 이프라인에 변경 감지 기능이 없습니다.

YAML

```
Resources:
  SourceBucket:
    Type: AWS::S3::Bucket
    Properties:
      VersioningConfiguration:
        Status: Enabled
  CodePipelineArtifactStoreBucket:
    Type: AWS::S3::Bucket
  CodePipelineArtifactStoreBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket: !Ref CodePipelineArtifactStoreBucket
      PolicyDocument:
        Version: 2012-10-17
        Statement:
            Sid: DenyUnEncryptedObjectUploads
            Effect: Deny
            Principal: '*'
            Action: s3:PutObject
            Resource: !Join [ '', [ !GetAtt CodePipelineArtifactStoreBucket.Arn, '/
*']]
            Condition:
              StringNotEquals:
                s3:x-amz-server-side-encryption: aws:kms
```

```
Sid: DenyInsecureConnections
            Effect: Deny
            Principal: '*'
            Action: s3:*
            Resource: !Join [ '', [ !GetAtt CodePipelineArtifactStoreBucket.Arn, '/
*']]
            Condition:
              Bool:
                aws:SecureTransport: false
  CodePipelineServiceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
            Effect: Allow
            Principal:
              Service:
                - codepipeline.amazonaws.com
            Action: sts:AssumeRole
      Path: /
      Policies:
          PolicyName: AWS-CodePipeline-Service-3
          PolicyDocument:
            Version: 2012-10-17
            Statement:
                Effect: Allow
                Action:
                  - codecommit:CancelUploadArchive
                  - codecommit:GetBranch
                  - codecommit:GetCommit
                  - codecommit:GetUploadArchiveStatus
                  - codecommit:UploadArchive
                Resource: 'resource_ARN'
                Effect: Allow
                Action:
                  - codedeploy:CreateDeployment
                  - codedeploy:GetApplicationRevision
                  - codedeploy:GetDeployment
```

```
- codedeploy:GetDeploymentConfig
  - codedeploy:RegisterApplicationRevision
Resource: 'resource_ARN'
Effect: Allow
Action:
  - codebuild:BatchGetBuilds
  - codebuild:StartBuild
Resource: 'resource_ARN'
Effect: Allow
Action:
  - devicefarm:ListProjects
  - devicefarm:ListDevicePools
  - devicefarm:GetRun
  - devicefarm:GetUpload
  - devicefarm:CreateUpload
  - devicefarm:ScheduleRun
Resource: 'resource_ARN'
Effect: Allow
Action:
  - lambda:InvokeFunction
  - lambda:ListFunctions
Resource: 'resource_ARN'
Effect: Allow
Action:
  - iam:PassRole
Resource: 'resource_ARN'
Effect: Allow
Action:
  - elasticbeanstalk:*
  - ec2:*
  - elasticloadbalancing:*
  - autoscaling:*
  - cloudwatch:*
  - s3:*
  - sns:*
  - cloudformation:*
  - rds:*
  - sqs:*
  - ecs:*
```

```
사용자 가이드
```

```
Resource: 'resource_ARN'
AppPipeline:
  Type: AWS::CodePipeline::Pipeline
  Properties:
    Name: s3-events-pipeline
    RoleArn:
      !GetAtt CodePipelineServiceRole.Arn
    Stages:
      _
        Name: Source
        Actions:
          _
            Name: SourceAction
            ActionTypeId:
              Category: Source
              Owner: AWS
              Version: 1
              Provider: S3
            OutputArtifacts:
              - Name: SourceOutput
            Configuration:
              S3Bucket: !Ref SourceBucket
              S3ObjectKey: !Ref SourceObjectKey
              PollForSourceChanges: false
            RunOrder: 1
        Name: Beta
        Actions:
            Name: BetaAction
            InputArtifacts:
              - Name: SourceOutput
            ActionTypeId:
              Category: Deploy
              Owner: AWS
              Version: 1
              Provider: CodeDeploy
            Configuration:
              ApplicationName: !Ref ApplicationName
              DeploymentGroupName: !Ref BetaFleet
            RunOrder: 1
    ArtifactStore:
      Type: S3
      Location: !Ref CodePipelineArtifactStoreBucket
```

```
EventRole:
  Type: AWS::IAM::Role
   Properties:
    AssumeRolePolicyDocument:
      Version: 2012-10-17
       Statement:
           Effect: Allow
           Principal:
             Service:
               - events.amazonaws.com
           Action: sts:AssumeRole
     Path: /
     Policies:
         PolicyName: eb-pipeline-execution
         PolicyDocument:
           Version: 2012-10-17
           Statement:
               Effect: Allow
               Action: codepipeline:StartPipelineExecution
               Resource: !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region',
':', !Ref 'AWS::AccountId', ':', !Ref AppPipeline ] ]
EventRule:
  Type: AWS::Events::Rule
  Properties:
     EventPattern:
       source:
         - aws.s3
      detail-type:
         - 'AWS API Call via CloudTrail'
       detail:
         eventSource:
           - s3.amazonaws.com
         eventName:
           - PutObject
           - CompleteMultipartUpload
         resources:
           ARN:
             - !Join [ '', [ !GetAtt SourceBucket.Arn, '/', !Ref
SourceObjectKey ] ]
     Targets:
```

```
Arn:

|Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region', ':', !Ref

'AWS::AccountId', ':', !Ref AppPipeline ] ]

RoleArn: !GetAtt EventRole.Arn

Id: codepipeline-AppPipeline

Outputs:

SourceBucketARN:

Description: "S3 bucket ARN that Cloudtrail will use"

Value: !GetAtt SourceBucket.Arn

Export:

Name: SourceBucketARN
```

```
"Resources": {
    "SourceBucket": {
        "Type": "AWS::S3::Bucket",
        "Properties": {
            "VersioningConfiguration": {
                "Status": "Enabled"
            }
        }
    },
    "CodePipelineArtifactStoreBucket": {
        "Type": "AWS::S3::Bucket"
    },
    "CodePipelineArtifactStoreBucketPolicy": {
        "Type": "AWS::S3::BucketPolicy",
        "Properties": {
            "Bucket": {
                "Ref": "CodePipelineArtifactStoreBucket"
            },
            "PolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Sid": "DenyUnEncryptedObjectUploads",
                        "Effect": "Deny",
                        "Principal": "*",
                        "Action": "s3:PutObject",
                        "Resource": {
                             "Fn::Join": [
```
```
"",
                Ε
                     {
                         "Fn::GetAtt": [
                             "CodePipelineArtifactStoreBucket",
                             "Arn"
                         ]
                     },
                     "/*"
                ]
            ]
        },
        "Condition": {
            "StringNotEquals": {
                 "s3:x-amz-server-side-encryption": "aws:kms"
            }
        }
    },
    {
        "Sid": "DenyInsecureConnections",
        "Effect": "Deny",
        "Principal": "*",
        "Action": "s3:*",
        "Resource": {
            "Fn::Join": [
                "",
                 Γ
                     {
                         "Fn::GetAtt": [
                             "CodePipelineArtifactStoreBucket",
                             "Arn"
                         ]
                     },
                     "/*"
                ]
            ]
        },
        "Condition": {
            "Bool": {
                 "aws:SecureTransport": false
            }
        }
    }
]
```

}

},

```
}
"CodePipelineServiceRole": {
    "Type": "AWS::IAM::Role",
    "Properties": {
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                         "Service": [
                             "codepipeline.amazonaws.com"
                        ]
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        },
        "Path": "/",
        "Policies": [
            {
                "PolicyName": "AWS-CodePipeline-Service-3",
                "PolicyDocument": {
                     "Version": "2012-10-17",
                    "Statement": [
                        {
                             "Effect": "Allow",
                             "Action": [
                                 "codecommit:CancelUploadArchive",
                                 "codecommit:GetBranch",
                                 "codecommit:GetCommit",
                                 "codecommit:GetUploadArchiveStatus",
                                 "codecommit:UploadArchive"
                             ],
                             "Resource": "resource_ARN"
                        },
                        {
                             "Effect": "Allow",
                             "Action": [
                                 "codedeploy:CreateDeployment",
                                 "codedeploy:GetApplicationRevision",
                                 "codedeploy:GetDeployment",
```

```
"codedeploy:GetDeploymentConfig",
        "codedeploy:RegisterApplicationRevision"
    ],
    "Resource": "resource_ARN"
},
{
    "Effect": "Allow",
    "Action": [
        "codebuild:BatchGetBuilds",
        "codebuild:StartBuild"
    ],
    "Resource": "resource_ARN"
},
{
    "Effect": "Allow",
    "Action": [
        "devicefarm:ListProjects",
        "devicefarm:ListDevicePools",
        "devicefarm:GetRun",
        "devicefarm:GetUpload",
        "devicefarm:CreateUpload",
        "devicefarm:ScheduleRun"
    ],
    "Resource": "resource_ARN"
},
{
    "Effect": "Allow",
    "Action": [
        "lambda:InvokeFunction",
        "lambda:ListFunctions"
    ],
    "Resource": "resource_ARN"
},
{
    "Effect": "Allow",
    "Action": [
        "iam:PassRole"
    ],
    "Resource": "resource_ARN"
},
{
    "Effect": "Allow",
    "Action": [
        "elasticbeanstalk:*",
```

```
"ec2:*",
                                 "elasticloadbalancing:*",
                                 "autoscaling:*",
                                 "cloudwatch:*",
                                 "s3:*",
                                 "sns:*",
                                 "cloudformation:*",
                                 "rds:*",
                                 "sqs:*",
                                 "ecs:*"
                             ],
                             "Resource": "resource_ARN"
                         }
                    ]
                }
            }
        ]
    }
},
"AppPipeline": {
    "Type": "AWS::CodePipeline::Pipeline",
    "Properties": {
        "Name": "s3-events-pipeline",
        "RoleArn": {
            "Fn::GetAtt": [
                 "CodePipelineServiceRole",
                "Arn"
            ]
        },
        "Stages": [
            {
                 "Name": "Source",
                 "Actions": [
                    {
                         "Name": "SourceAction",
                         "ActionTypeId": {
                             "Category": "Source",
                             "Owner": "AWS",
                             "Version": 1,
                             "Provider": "S3"
                         },
                         "OutputArtifacts": [
                             {
                                 "Name": "SourceOutput"
```

```
}
                ],
                "Configuration": {
                     "S3Bucket": {
                         "Ref": "SourceBucket"
                     },
                     "S3ObjectKey": {
                         "Ref": "SourceObjectKey"
                     },
                     "PollForSourceChanges": false
                },
                "RunOrder": 1
            }
        ]
    },
    {
        "Name": "Beta",
        "Actions": [
            {
                "Name": "BetaAction",
                "InputArtifacts": [
                     {
                         "Name": "SourceOutput"
                     }
                ],
                "ActionTypeId": {
                     "Category": "Deploy",
                     "Owner": "AWS",
                     "Version": 1,
                     "Provider": "CodeDeploy"
                },
                "Configuration": {
                     "ApplicationName": {
                         "Ref": "ApplicationName"
                     },
                     "DeploymentGroupName": {
                         "Ref": "BetaFleet"
                     }
                },
                "RunOrder": 1
            }
        ]
    }
],
```

```
"ArtifactStore": {
            "Type": "S3",
            "Location": {
                "Ref": "CodePipelineArtifactStoreBucket"
            }
        }
    }
},
"EventRole": {
    "Type": "AWS:::IAM::Role",
    "Properties": {
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                     "Effect": "Allow",
                     "Principal": {
                         "Service": [
                             "events.amazonaws.com"
                         ]
                    },
                     "Action": "sts:AssumeRole"
                }
            ]
        },
        "Path": "/",
        "Policies": [
            {
                 "PolicyName": "eb-pipeline-execution",
                "PolicyDocument": {
                     "Version": "2012-10-17",
                     "Statement": [
                         {
                             "Effect": "Allow",
                             "Action": "codepipeline:StartPipelineExecution",
                             "Resource": {
                                 "Fn::Join": [
                                     "",
                                     Ε
                                          "arn:aws:codepipeline:",
                                          {
                                              "Ref": "AWS::Region"
                                          },
                                          ":"
```

```
{
                                              "Ref": "AWS::AccountId"
                                          },
                                          ":",
                                          {
                                              "Ref": "AppPipeline"
                                          }
                                      ]
                                 ]
                             }
                         }
                     ]
                }
            }
        ]
    }
},
"EventRule": {
    "Type": "AWS::Events::Rule",
    "Properties": {
        "EventPattern": {
            "source": [
                 "aws.s3"
            ],
            "detail-type": [
                 "AWS API Call via CloudTrail"
            ],
            "detail": {
                 "eventSource": [
                     "s3.amazonaws.com"
                ],
                 "eventName": [
                     "PutObject",
                     "CompleteMultipartUpload"
                ],
                 "resources": {
                     "ARN": [
                         {
                             "Fn::Join": [
                                 "",
                                  Ε
                                      {
                                          "Fn::GetAtt": [
                                              "SourceBucket",
```



```
}
}
},
"Outputs" : {
"SourceBucketARN" : {
"Description" : "S3 bucket ARN that Cloudtrail will use",
"Value" : { "Fn::GetAtt": ["SourceBucket", "Arn"] },
"Export" : {
"Name" : "SourceBucketARN"
}
}
....
```

GitHub(OAuth 앱을 통해) 소스 작업에 대한 폴링 파이프라인을 연결로 마이 그레이션

GitHub(OAuth 앱 사용) 소스 작업을 마이그레이션하여 외부 리포지토리에 대한 연결을 사용할 수 있습니다. 이는 GitHub(OAuth 앱 사용) 소스 작업이 있는 파이프라인에 권장되는 변경 감지 방법입니다.

GitHub(OAuth 앱 사용) 소스 작업이 있는 파이프라인의 경우 변경 감지가 자동화되도록 GitHub(GitHub 앱 사용) 작업을 사용하도록 파이프라인을 수정하는 것이 좋습니다 AWS CodeConnections. 연결 관련 작업에 대한 자세한 내용은 <u>GitHub 연결</u> 섹션을 참조하세요.

GitHub에 대한 연결 생성(콘솔)

콘솔을 사용하여 GitHub에 대한 연결을 생성할 수 있습니다.

1단계: GitHub 교체(OAuth 앱 사용) 작업

파이프라인 편집 페이지를 사용하여 GitHub(OAuth 앱 사용) 작업을 GitHub(GitHub 앱 사용) 작업으로 바꿉니다.

GitHub(OAuth 앱 사용) 작업을 교체하려면

1. CodePipeline 콘솔에 로그인합니다.

- 파이프라인을 선택한 다음 편집을 선택합니다. 소스 단계에서 단계 편집을 선택합니다. 작업 업데 이트를 권장하는 메시지가 표시됩니다.
- 3. 작업 공급자에서 GitHub(GitHub 앱을 통해)를 선택합니다.
- 4. 다음 중 하나를 수행합니다.
 - 연결에서 공급자와의 연결을 아직 생성하지 않은 경우 GitHub에 연결을 선택합니다. 2단계: GitHub에 대한 연결 생성으로 이동합니다.
 - 연결에서 공급자와의 연결을 이미 생성한 경우 연결을 선택합니다. 3단계: 연결에 대한 소스 작 업 저장으로 이동합니다.

2단계: GitHub에 대한 연결 생성

연결을 생성하도록 선택하면 GitHub에 연결 페이지가 표시됩니다.

GitHub에 대한 연결을 생성하려면

1. GitHub 연결 설정 아래의 연결 이름에 연결 이름이 표시됩니다.

GitHub 앱에서 앱 설치를 선택하거나 새 앱 설치를 선택하여 앱을 새로 만듭니다.

Note

특정 공급자에 대한 모든 연결에 대해 하나의 앱을 설치합니다. GitHub 앱을 이미 설치한 경우 선택한 다음 이 단계를 건너뜁니다.

- 2. GitHub의 승인 페이지가 표시되면 자격 증명으로 로그인한 다음 계속하도록 선택합니다.
- 3. 앱 설치 페이지에서 AWS CodeStar 앱이 GitHub 계정에 연결을 시도하고 있음을 보여주는 메시지 가 표시됩니다.

Note

각 GitHub 계정마다 앱을 한 번만 설치합니다. 이전에 앱을 설치한 경우 [구성(Configure)] 을 선택하여 앱 설치의 수정 페이지로 이동하거나 뒤로 버튼을 사용하여 콘솔로 돌아갈 수 있습니다.

- 4. AWS CodeStar설치 페이지에서 설치를 선택합니다.
- 5. GitHub에 연결 페이지에서 새 설치의 연결 ID가 표시됩니다. 연결을 선택합니다.

3단계: GitHub 소스 작업 저장

작업 편집 페이지에서 업데이트를 완료하여 새 소스 작업을 저장합니다.

GitHub 소스 작업을 저장하려면

 리포지토리에서 타사 리포지토리의 이름을 입력합니다. 브랜치에서, 파이프라인에서 소스 변경 사항을 감지할 브랜치를 입력합니다.

Note
 리포지토리에 다음 예와 같이 owner-name/repository-name을 입력합니다.
 my-account/my-repository

2. 출력 아티팩트 형식에서 아티팩트의 형식을 선택합니다.

- 기본 방법을 사용하여 GitHub 작업의 출력 아티팩트를 저장하려면 CodePipeline 기본 방법을 선택합니다. 그러면 GitHub 리포지토리의 파일에 액세스하여 파이프라인 아티팩트 스토어에 ZIP 파일로 아티팩트가 저장됩니다.
- 다운스트림 작업이 Git 명령을 직접 수행할 수 있도록 리포지토리에 대한 URL 참조가 포함된 JSON 파일을 저장하려면 Full clone(전체 복제)을 선택합니다. 이 옵션은 CodeBuild 다운스트림 작업에서만 사용할 수 있습니다.

이 옵션을 선택하면 <u>Bitbucket, GitHub, GitHub Enterprise Server 또는 GitLab.com 연결을 위한</u> <u>CodeBuild GitClone 권한 추가</u>과 같이 CodeBuild 프로젝트 서비스 역할에 대한 권한을 업데이 트해야 합니다. 전체 복제 옵션을 사용하는 방법을 보여주는 자습서는 <u>자습서: GitHub 파이프라</u> 인 소스와 함께 전체 복제 사용을 참조하세요.

- 3. 출력 아티팩트에서 이 작업에 대한 출력 아티팩트의 이름을 유지할 수 있습니다(예: SourceArtifact). 완료를 선택하여 작업 편집 페이지를 닫습니다.
- 완료를 선택하여 단계 편집 페이지를 닫습니다. 저장을 선택하여 파이프라인 편집 페이지를 닫습 니다.

GitHub에 대한 연결 생성(CLI)

AWS Command Line Interface (AWS CLI)를 사용하여 GitHub에 대한 연결을 생성할 수 있습니다.

이렇게 하려면 create-connection 명령을 사용합니다.

▲ Important

AWS CLI 또는를 통해 생성된 연결 AWS CloudFormation 은 기본적으로 PENDING 상태입니 다. CLI 또는 와의 연결을 생성한 후 콘솔을 AWS CloudFormation사용하여 연결을 편집하여 상태를 로 설정합니다AVAILABLE.

GitHub에 대한 연결을 생성하려면

 터미널(Linux, macOS, Unix) 또는 명령 프롬프트(Windows)를 엽니다. AWS CLI 를 사용하여 연결 을 --connection-name 위해 --provider-type 및를 지정하여 create-connection 명령을 실 행합니다. 이 예제에서 타사 공급자 이름은 GitHub이고 지정된 연결 이름은 MyConnection입니 다.

```
aws codeconnections create-connection --provider-type GitHub --connection-name
MyConnection
```

이 명령이 제대로 실행되면 다음과 비슷한 연결 ARN 정보가 반환됩니다.

```
{
    "ConnectionArn": "arn:aws:codeconnections:us-west-2:account_id:connection/
aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f"
}
```

2. 콘솔을 사용하여 연결을 완료합니다.

GitHub(OAuth 앱 사용) 소스 작업에 대한 폴링 파이프라인을 웹후크로 마이 그레이션

Webhook를 사용하고 GitHub 소스 리포지토리의 변경 사항을 탐지하도록 파이프라인을 마이그레이션 할 수 있습니다. 웹후크로의 마이그레이션은 GitHub(OAuth 앱 사용) 작업에만 해당됩니다.

- 콘솔: 폴링 파이프라인을 Webhook로 마이그레이션(GitHub(OAuth 앱 사용) 소스 작업)(콘솔)
- CLI: <u>폴링 파이프라인을 웹후크로 마이그레이션(GitHub(OAuth 앱 사용) 소스 작업)(CLI)</u>
- AWS CloudFormation: <u>푸시 이벤트에 대한 파이프라인 업데이트(GitHub(OAuth 앱 사용) 소스 작업)</u> (AWS CloudFormation 템플릿)

▲ Important

CodePipeline 웹후크를 생성할 때 자체 자격 증명을 사용하거나 여러 웹후크에서 동일한 보 안 암호 토큰을 재사용하지 마세요. 최적의 보안을 위해 생성하는 각 웹후크에 대해 고유한 보 안 암호 토큰을 생성합니다. 보안 암호 토큰은 사용자가 제공하는 임의의 문자열로, GitHub는 CodePipeline으로 전송된 웹후크 페이로드를 계산하고 서명하는 데 이를 사용하여 웹후크 페 이로드의 무결성과 신뢰성을 보호합니다. 자체 자격 증명을 사용하거나 여러 웹후크에서 동일 한 토큰을 재사용하면 보안 취약성이 발생할 수 있습니다.

폴링 파이프라인을 Webhook로 마이그레이션(GitHub(OAuth 앱 사용) 소스 작업)(콘솔)

GitHub(OAuth 앱 사용) 소스 작업의 경우 CodePipeline 콘솔을 사용하여 Webhook를 사용하여 GitHub 소스 리포지토리의 변경 사항을 감지하도록 파이프라인을 업데이트할 수 있습니다.

대신 EventBridge를 사용하도록 폴링(정기적 확인)을 사용하는 파이프라인을 편집하려면 다음 단계를 수행합니다. 파이프라인을 생성하려면 <u>파이프라인 스테이지 및 작업 생성</u> 단원을 참조하십시오.

콘솔을 사용할 경우 해당 파이프라인에 대한 PollForSourceChanges 파라미터가 변경됩니다. GitHub Webhook가 생성되어 등록됩니다.

파이프라인 소스 단계를 편집하려면

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> <u>codepipeline/home://https://https://www.com에서 CodePipeline 콘솔을 엽니다.</u>

AWS 계정과 연결된 모든 파이프라인의 이름이 표시됩니다.

- [Name]에서 편집할 파이프라인의 이름을 선택합니다. 이렇게 하면 파이프라인 각 단계의 각 작업 상태를 포함하여 파이프라인의 세부 정보 보기가 열립니다.
- 3. 파이프라인 세부 정보 페이지에서 [Edit]를 선택합니다.
- 4. 편집 단계에서 소스 작업의 편집 아이콘을 선택합니다.
- 5. 변경 탐지 옵션을 확장하고 변경 사항이 생길 때 Amazon CloudWatch Events를 사용하여 내 파이 프라인을 자동으로 시작합니다(권장)를 선택합니다.

CodePipeline이 GitHub에 웹후크를 생성하여 소스 변경 사항을 감지함을 알리는 메시지가 표시됩 니다. AWS CodePipeline은 사용자를 대신하여 웹후크를 생성합니다. 아래 옵션에서 옵트아웃할 수 있습니다. 업데이트를 선택합니다. CodePipeline은 Webhook와 함께 다음을 생성합니다.

• 보안 암호는 무작위로 생성되며 GitHub에 대한 연결을 승인하는 데 사용됩니다.

• Webhook URL은 리전의 퍼블릭 엔드포인트를 사용하여 생성됩니다.

CodePipeline은 Webhook를 GitHub에 등록합니다. 이를 통해 URL을 구독하여 리포지토리 이벤 트를 수신할 수 있습니다.

6. 파이프라인 편집을 마쳤으면 [Save pipeline changes]를 선택하여 요약 페이지로 돌아갑니다.

파이프라인에 대해 생성될 Webhook의 이름을 표시하는 메시지가 나타납니다. [Save and continue]를 선택합니다.

 작업을 테스트하려면를 사용하여 파이프라인의 소스 단계에 지정된 소스에 변경 사항을 AWS CLI 커밋하여 변경 사항을 릴리스합니다.

폴링 파이프라인을 웹후크로 마이그레이션(GitHub(OAuth 앱 사용) 소스 작업)(CLI)

대시 Webhook를 사용하도록 정기적 확인을 사용하는 파이프라인을 편집하려면 다음 단계를 수행합니 다. 파이프라인을 생성하려면 파이프라인 스테이지 및 작업 생성 단원을 참조하십시오.

이벤트 기반 파이프라인을 빌드하려면 파이프라인의 PollForSourceChanges 파라미터를 편집한 후 다음 리소스를 수동으로 생성합니다.

• GitHub Webhook 및 권한 부여 파라미터

Webhook를 생성하고 등록하려면

CLI 또는를 사용하여 파이프라인 AWS CloudFormation 을 생성하고 웹후크를 추가할 때는 정기 검사를 비활성화해야 합니다. 주기적 점검을 비활성화하지 않으려면 아래의 마지막 절차에서 설명한 대로 PollForSourceChanges 파라미터를 명시적으로 추가하고 false 로 설정해야 합니다. 그렇지 않으면 CLI 또는 AWS CloudFormation 파이프라인의 기본값은 PollForSourceChanges 기본값이 true이고 파이프라인 구조 출력에 표시되지 않습니다. PollForSourceChanges 기본값에 대한 자세한 내용은 <u>PollForSourceChanges 파라미터의</u> <u>유효한 설정</u> 항목을 참조하십시오.

1. 텍스트 편집기에서 생성할 Webhook에 대한 JSON 파일을 생성 및 저장합니다. my-webhook라는 Webhook에 이 샘플 파일을 사용합니다.

Note

```
{
    "webhook": {
        "name": "my-webhook",
    "targetPipeline": "pipeline_name",
    "targetAction": "source_action_name",
    "filters": [{
        "jsonPath": "$.ref",
        "matchEquals": "refs/heads/{Branch}"
    }],
    "authentication": "GITHUB_HMAC",
    "authenticationConfiguration": {
        "SecretToken": "secret"
    }
    }
}
```

2. put-webhook 명령을 호출하고 --cli-input 및 --region 파라미터를 포함시킵니다.

다음 샘플 명령은 webhook_json JSON 파일을 사용하여 Webhook을 생성합니다.

```
aws codepipeline put-webhook --cli-input-json file://webhook_json.json --region
  "eu-central-1"
```

3. 이 예제의 출력에는 my-webhook라는 Webhook의 URL과 ARN이 반환됩니다.

```
{
    "webhook": {
        "url": "https://webhooks.domain.com/
trigger111111111EXAMPLE111111111111111111,
        "definition": {
            "authenticationConfiguration": {
                "SecretToken": "secret"
            },
            "name": "my-webhook",
            "authentication": "GITHUB_HMAC",
            "targetPipeline": "pipeline_name",
            "targetAction": "Source",
            "filters": [
                {
                    "jsonPath": "$.ref",
                    "matchEquals": "refs/heads/{Branch}"
                }
```

```
]
},
"arn": "arn:aws:codepipeline:eu-central-1:ACCOUNT_ID:webhook:my-webhook"
},
"tags": [{
    "key": "Project",
    "value": "ProjectA"
}]
}
```

이 예제는 Webhook에 Project 태그 키와 ProjectA 값을 포함하여 Webhook에 태그 지정을 추 가합니다. CodePipeline의 리소스 태깅에 대한 자세한 내용은 <u>리소스에 태그 지정</u> 섹션을 참조하 세요.

4. register-webhook-with-third-party 명령을 호출하고 --webhook-name 파라미터를 포함시킵니다.

다음 샘플 명령은 my-webhook이라는 Webhook을 등록합니다.

aws codepipeline register-webhook-with-third-party --webhook-name my-webhook

파이프라인의 PollForSourceChanges 파라미터를 편집하려면

A Important

이 방법으로 파이프라인을 생성할 때 명시적으로 false로 설정되지 않은 경우 PollForSourceChanges 파라미터 기본값은 true입니다. 이벤트 기반 변경 감지를 추가할 때 는 출력에 파라미터를 추가하고 false로 설정하여 폴링을 비활성화해야 합니다. 그렇지 않으면 파이프라인이 단일 소스 변경 시 두 번 시작됩니다. 세부 정보는 <u>PollForSourceChanges 파</u> <u>라미터의 유효한 설정</u>을 참조하세요.

1. get-pipeline 명령을 실행하여 파이프라인 구조를 JSON 파일로 복사합니다. 예를 들어 MyFirstPipeline이라는 파이프라인에 대해 다음 명령을 입력합니다.

aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json

이 명령은 아무 것도 반환하지 않지만 생성한 파일이 명령을 실행한 디렉터리에 표시되어야 합니 다. 2. 일반 텍스트 편집기에서 JSON 파일을 열고 PollForSourceChanges 파라미터를 변경하거나 추가하여 소스 단계를 편집합니다. 이 예시에서는 UserGitHubRepo라는 리포지토리의 경우 파 라미터가 false로 설정됩니다.

이렇게 변경하는 이유는 무엇입니까? 이 파라미터를 변경하면 정기적 확인이 비활성화되어 이벤 트 기반 변경 탐지만 사용할 수 있습니다.

```
"configuration": {
    "Owner": "name",
    "Repo": "UserGitHubRepo",
    "PollForSourceChanges": "false",
    "Branch": "main",
    "OAuthToken": "****"
},
```

get-pipeline 명령을 사용하여 검색한 파이프라인 구조로 작업 중인 경우, 파일에서 metadata 행을 삭제하여 JSON 파일의 구조를 편집해야 합니다. 이렇게 하지 않으면 update-pipeline 명령에서 사용할 수 없습니다. JSON 파일의 파이프라인 구조에서 "metadata" 섹션({ }, "created", "pipelineARN" 및 "updated" 필드 포함)을 삭제합니다.

예를 들어, 구조에서 다음 행을 삭제합니다.

```
"metadata": {
    "pipelineArn": "arn:aws:codepipeline:region:account-ID:pipeline-name",
    "created": "date",
    "updated": "date"
},
```

파일을 저장합니다.

4. 변경 사항을 적용하려면 다음과 유사하게 파이프라인 JSON 파일을 지정하여 update-pipeline 명 령을 실행합니다.

```
▲ Important
```

파일 이름 앞에 file://를 포함해야 합니다. 이 명령에 필수적입니다.

aws codepipeline update-pipeline --cli-input-json file://pipeline.json

이 명령은 편집한 파이프라인의 전체 구조를 반환합니다.

Note

update-pipeline 명령을 실행하면 파이프라인이 중지됩니다. update-pipeline 명령을 실행 할 때 파이프라인을 통해 개정을 실행하는 중이라면 해당 실행이 중지됩니다. 업데이트 된 파이프라인을 통해 해당 개정을 실행하려면 파이프라인을 수동으로 시작해야 합니다. start-pipeline-execution 명령을 사용하여 수동으로 파이프라인을 시작합니다.

푸시 이벤트에 대한 파이프라인 업데이트(GitHub(OAuth 앱 사용) 소스 작업)(AWS CloudFormation 템플릿)

정기적 확인(폴링)에서 webhook를 사용한 이벤트 기반 변경 감지로 GitHub 소스가 있는 파이프라인을 업데이트하려면 다음 단계를 수행합니다.

를 사용하여 이벤트 기반 파이프라인을 빌드하려면 파이프라인의 PollForSourceChanges 파라미 터를 AWS CodeCommit편집한 다음 GitHub 웹후크 리소스를 템플릿에 추가합니다.

AWS CloudFormation 를 사용하여 파이프라인을 생성하고 관리하는 경우 템플릿에는 다음과 같은 콘 텐츠가 있습니다.

Note

소스 단계의 PollForSourceChanges 구성 속성을 메모합니다. 템플릿에 해당 속성이 포함 되어 있지 않으면 기본적으로 PollForSourceChanges가 true로 설정됩니다.

YAML

```
Resources:

AppPipeline:

Type: AWS::CodePipeline::Pipeline

Properties:

Name: github-polling-pipeline

RoleArn:

!GetAtt CodePipelineServiceRole.Arn

Stages:
```



JSON

```
"AppPipeline": {
    "Type": "AWS::CodePipeline::Pipeline",
   "Properties": {
        "Name": "github-polling-pipeline",
        "RoleArn": {
            "Fn::GetAtt": [
                "CodePipelineServiceRole",
                "Arn"
            1
        },
        "Stages": [
            {
                "Name": "Source",
                "Actions": [
                    {
                        "Name": "SourceAction",
                        "ActionTypeId": {
                             "Category": "Source",
```

```
"Owner": "ThirdParty",
                                     "Version": 1,
                                     "Provider": "GitHub"
                                 },
                                 "OutputArtifacts": [
                                     {
                                          "Name": "SourceOutput"
                                     }
                                 ],
                                 "Configuration": {
                                     "Owner": {
                                          "Ref": "GitHubOwner"
                                     },
                                     "Repo": {
                                         "Ref": "RepositoryName"
                                     },
                                     "Branch": {
                                          "Ref": "BranchName"
                                     },
                                     "OAuthToken":
"{{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}",
                                     "PollForSourceChanges": true
                                 },
                                 "RunOrder": 1
                             }
                         ]
                    },
. . .
```

템플릿에서 파라미터를 추가하고 Webhook를 만들려면

를 사용하여 자격 증명을 저장하는 AWS Secrets Manager 것이 좋습니다. Secrets Manager를 사용 하는 경우 Secrets Manager에서 보안 파라미터를 이미 구성하고 저장했어야 합니다. 이 예에서는 Webhook의 GitHub 자격 증명에 대한 Secrets Manager에 대한 동적 참조를 사용합니다. 자세한 내용 은 동적 참조를 사용하여 템플릿 값 지정을 참조하십시오.

▲ Important

보안 파라미터를 전달할 때는 값을 템플릿에 직접 입력하지 마십시오. 값은 일반 텍스트로 렌 더링되므로 읽을 수 있습니다. 보안상의 이유로 AWS CloudFormation 템플릿에서 일반 텍스트 를 사용하여 자격 증명을 저장하지 마십시오.

CLI 또는를 사용하여 파이프라인 AWS CloudFormation 을 생성하고 웹후크를 추가할 때는 정기 검사 를 비활성화해야 합니다.

```
    Note
```

주기적 점검을 비활성화하지 않으려면 아래의 마지막 절차에서 설명한 대로 PollForSourceChanges 파라미터를 명시적으로 추가하고 false로 설정해야 합니다. 그렇지 않으면 CLI 또는 AWS CloudFormation 파이프라인의 기본값은 PollForSourceChanges 기 본값이 true이고 파이프라인 구조 출력에 표시되지 않습니다. PollForSourceChanges 기본값 에 대한 자세한 내용은 <u>PollForSourceChanges 파라미터의 유효한 설정</u> 항목을 참조하십시 오.

1. 템플릿에서 Resources에 파라미터를 추가합니다.

YAML

Parameters: GitHubOwner: Type: String

JSON

```
{
    "Parameters": {
        "BranchName": {
            "Description": "GitHub branch name",
            "Type": "String",
            "Default": "main"
        },
        "GitHubOwner": {
    }
}
```

```
"Type": "String"
},
...
```

AWS::CodePipeline::Webhook AWS CloudFormation 리소스를 사용하여 웹후크를 추가합니다.

Note

지정한 TargetAction은 파이프라인에 정의된 소스 작업의 Name 속성과 일치해야 합니다.

RegisterWithThirdParty가 true로 설정된 경우 OAuthToken에 연결된 사용자가 GitHub에 서 필요한 범위를 설정할 수 있는지 확인합니다. 토큰 및 Webhook에 필요한 GitHub 범위는 다음 과 같습니다.

- repo 퍼블릭 및 프라이빗 리포지토리에서 파이프라인으로 아티팩트를 읽고 가져올 수 있도록 완전히 제어하는 데 사용됩니다.
- admin:repo_hook 리포지토리 후크를 완전히 제어하는 데 사용됩니다.

그렇지 않은 경우 GitHub는 404를 반환합니다. 반환된 404에 대한 자세한 내용은 <u>https://</u> help.github.com/articles/about-webhooks 단원을 참조하십시오.

YAML

```
AppPipelineWebhook:
Type: AWS::CodePipeline::Webhook
Properties:
Authentication: GITHUB_HMAC
AuthenticationConfiguration:
SecretToken:
{{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}
Filters:
JsonPath: "$.ref"
MatchEquals: refs/heads/{Branch}
TargetPipeline: !Ref AppPipeline
TargetAction: SourceAction
```

```
Name: AppPipelineWebhook
TargetPipelineVersion: !GetAtt AppPipeline.Version
RegisterWithThirdParty: true
```

JSON

. . .

```
"AppPipelineWebhook": {
    "Type": "AWS::CodePipeline::Webhook",
    "Properties": {
        "Authentication": "GITHUB_HMAC",
        "AuthenticationConfiguration": {
            "SecretToken":
 "{{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}"
        },
        "Filters": [{
            "JsonPath": "$.ref",
            "MatchEquals": "refs/heads/{Branch}"
        }],
        "TargetPipeline": {
            "Ref": "AppPipeline"
        },
        "TargetAction": "SourceAction",
        "Name": "AppPipelineWebhook",
        "TargetPipelineVersion": {
            "Fn::GetAtt": [
              "AppPipeline",
              "Version"
            1
        },
        "RegisterWithThirdParty": true
    }
},
. . .
```

- 3. 업데이트된 템플릿을 로컬 컴퓨터에 저장한 다음 AWS CloudFormation 콘솔을 엽니다.
- 4. 스택을 선택한 후 현재 스택에 대한 변경 세트 만들기를 선택합니다.
- 5. 템플릿을 업로드한 후 AWS CloudFormation에 나열된 변경 사항을 확인합니다. 이는 스택에 적용 될 변경 사항입니다. 목록에 새로운 리소스가 표시됩니다.

6. 실행을 선택합니다.

파이프라인의 PollForSourceChanges 파라미터를 편집하려면

🛕 Important

이 방법으로 파이프라인을 생성할 때 명시적으로 false로 설정되지 않은 경우 PollForSourceChanges 파라미터 기본값은 true입니다. 이벤트 기반 변경 감지를 추가할 때 는 출력에 파라미터를 추가하고 false로 설정하여 폴링을 비활성화해야 합니다. 그렇지 않으면 파이프라인이 단일 소스 변경 시 두 번 시작됩니다. 세부 정보는 <u>PollForSourceChanges 파</u> 라미터의 유효한 설정을 참조하세요.

• 템플릿에서 PollForSourceChanges를 false로 변경합니다. PollForSourceChanges를 파 이프라인 정의에 포함하지 않은 경우 추가하고 false로 설정하세요.

이렇게 변경하는 이유는 무엇입니까? 이 파라미터를 false로 변경하면 정기적 확인이 비활성화 되어 이벤트 기반 변경 탐지만 사용할 수 있습니다.

YAML

Name: Source
Actions:
-
Name: SourceAction
ActionTypeId:
Category: Source
Owner: ThirdParty
Version: 1
Provider: GitHub
OutputArtifacts:
- Name: SourceOutput
Configuration:
Owner: !Ref GitHubOwner
Repo: !Ref RepositoryName
Branch: !Ref BranchName
OAuthToken:
{{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}
PollForSourceChanges: false
RunOrder: 1

JSON

```
{
   "Name": "Source",
   "Actions": [{
 "Name": "SourceAction",
 "ActionTypeId": {
     "Category": "Source",
     "Owner": "ThirdParty",
     "Version": 1,
     "Provider": "GitHub"
},
"OutputArtifacts": [{
    "Name": "SourceOutput"
}],
"Configuration": {
    "Owner": {
  "Ref": "GitHubOwner"
    },
    "Repo": {
  "Ref": "RepositoryName"
    },
    "Branch": {
        "Ref": "BranchName"
    },
    "OAuthToken":
"{{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}",
    PollForSourceChanges: false
},
"RunOrder": 1
   }]
```

Example

를 사용하여 이러한 리소스를 생성하면 정의된 AWS CloudFormation웹후크가 지정된 GitHub 리포지 토리에 생성됩니다. 커밋 시 파이프라인이 트리거됩니다.

YAML

Parameters: GitHubOwner:

```
Type: String
Resources:
  AppPipelineWebhook:
    Type: AWS::CodePipeline::Webhook
    Properties:
      Authentication: GITHUB_HMAC
      AuthenticationConfiguration:
        SecretToken: {{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}
      Filters:
          JsonPath: "$.ref"
          MatchEquals: refs/heads/{Branch}
      TargetPipeline: !Ref AppPipeline
      TargetAction: SourceAction
      Name: AppPipelineWebhook
      TargetPipelineVersion: !GetAtt AppPipeline.Version
      RegisterWithThirdParty: true
  AppPipeline:
    Type: AWS::CodePipeline::Pipeline
    Properties:
      Name: github-events-pipeline
      RoleArn:
        !GetAtt CodePipelineServiceRole.Arn
      Stages:
        _
          Name: Source
          Actions:
              Name: SourceAction
              ActionTypeId:
                Category: Source
                Owner: ThirdParty
                Version: 1
                Provider: GitHub
              OutputArtifacts:
                - Name: SourceOutput
              Configuration:
                Owner: !Ref GitHubOwner
                Repo: !Ref RepositoryName
                Branch: !Ref BranchName
                OAuthToken:
 {{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}
                PollForSourceChanges: false
```

```
RunOrder: 1
```

• • •

JSON

```
{
    "Parameters": {
        "BranchName": {
            "Description": "GitHub branch name",
            "Type": "String",
            "Default": "main"
        },
        "RepositoryName": {
            "Description": "GitHub repository name",
            "Type": "String",
            "Default": "test"
        },
        "GitHubOwner": {
            "Type": "String"
        },
        "ApplicationName": {
            "Description": "CodeDeploy application name",
            "Type": "String",
            "Default": "DemoApplication"
        },
        "BetaFleet": {
            "Description": "Fleet configured in CodeDeploy",
            "Type": "String",
            "Default": "DemoFleet"
        }
    },
    "Resources": {
        },
        "AppPipelineWebhook": {
            "Type": "AWS::CodePipeline::Webhook",
            "Properties": {
                "Authentication": "GITHUB_HMAC",
                "AuthenticationConfiguration": {
                    "SecretToken": {
```

```
"{{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}"
                   }
               },
               "Filters": [
                   {
                        "JsonPath": "$.ref",
                        "MatchEquals": "refs/heads/{Branch}"
                   }
               ],
               "TargetPipeline": {
                   "Ref": "AppPipeline"
               },
               "TargetAction": "SourceAction",
               "Name": "AppPipelineWebhook",
               "TargetPipelineVersion": {
                   "Fn::GetAtt": [
                        "AppPipeline",
                       "Version"
                   ]
               },
               "RegisterWithThirdParty": true
           }
       },
       "AppPipeline": {
           "Type": "AWS::CodePipeline::Pipeline",
           "Properties": {
               "Name": "github-events-pipeline",
               "RoleArn": {
                   "Fn::GetAtt": [
                        "CodePipelineServiceRole",
                        "Arn"
                   ]
               },
               "Stages": [
                   {
                        "Name": "Source",
                        "Actions": [
                            {
                                "Name": "SourceAction",
                                "ActionTypeId": {
                                    "Category": "Source",
                                    "Owner": "ThirdParty",
                                    "Version": 1,
```

```
"Provider": "GitHub"
                                 },
                                 "OutputArtifacts": [
                                     {
                                          "Name": "SourceOutput"
                                     }
                                 ],
                                 "Configuration": {
                                     "Owner": {
                                          "Ref": "GitHubOwner"
                                     },
                                     "Repo": {
                                          "Ref": "RepositoryName"
                                     },
                                     "Branch": {
                                          "Ref": "BranchName"
                                     },
                                     "OAuthToken":
"{{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}",
                                     "PollForSourceChanges": false
                                 },
                                 "RunOrder": 1
. . .
```

CodePipeline 서비스 역할 생성

파이프라인 생성 시 서비스 역할을 생성하거나 기존 서비스 역할을 사용합니다.

CodePipeline 콘솔 또는를 사용하여 CodePipeline 서비스 역할을 AWS CLI 생성할 수 있습니다. 서비 스 역할은 파이프라인 생성에 필요하며 파이프라인은 항상 해당 서비스 역할과 연결됩니다.

AWS CLI를 사용하여 파이프라인을 생성하기 전에 파이프라인에 대한 CodePipeline 서비스 역할을 생 성해야 합니다. 서비스 역할 및 정책이 지정된 예제 AWS CloudFormation 템플릿은의 자습서를 참조하 세요자습서: AWS CloudFormation 배포 작업의 변수를 사용하는 파이프라인 생성.

서비스 역할은 AWS 관리형 역할이 아니지만 파이프라인 생성을 위해 처음에 생성된 다음 서비스 역할 정책에 새 권한이 추가되면 파이프라인의 서비스 역할을 업데이트해야 할 수 있습니다. 파이프라인이 서비스 역할로 생성되면 해당 파이프라인에 다른 서비스 역할을 적용할 수 없습니다. 권장 정책을 서비 스 역할에 연결합니다.

서비스 역할에 대한 자세한 내용은 CodePipeline 서비스 역할 관리 단원을 참조하십시오.

CodePipeline 서비스 역할 생성(콘솔)

콘솔을 사용하여 파이프라인을 생성할 때는 파이프라인 생성 마법사를 사용하여 CodePipeline 서비스 역할을 생성합니다.

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/</u> <u>codesuite/codepipeline/home</u>://https://https://https://i/https://://https://://https://://https:// www.com.com.com에서 CodePipeline 콘솔을 엽니다

파이프라인 마법사에서 파이프라인 생성을 선택하고 Step 1: Choose pipeline settings(1단계: 파이프라인 설정 선택) 페이지를 작성합니다.

Note

파이프라인을 만든 후에는 해당 이름을 변경할 수 없습니다. 다른 제한에 대한 자세한 내 용은 in AWS CodePipeline 할당량 섹션을 참조하십시오.

- 2. 서비스 역할에서 새 서비스 역할을 선택하여CodePipeline이 IAM에 새 서비스 역할을 생성하도록 허용합니다.
- 파이프라인 생성을 완료합니다. 파이프라인 서비스 역할은 IAM 역할 목록에서 볼 수 있으며 AWS CLI로 get-pipeline 명령을 실행하여 파이프라인과 관련된 서비스 역할 ARN을 볼 수 있습니 다.

CodePipeline 서비스 역할 생성(CLI)

AWS CLI 또는를 사용하여 파이프라인을 생성하기 전에 파이프라인에 대한 CodePipeline 서비스 역할 을 생성하고 서비스 역할 정책과 신뢰 정책을 연결 AWS CloudFormation해야 합니다. CLI를 사용하여 서비스 역할을 생성하려면 먼저 아래 단계를 사용하여 CLI 명령을 실행할 디렉터리에 신뢰 정책 JSON 과 역할 정책 JSON을 별도의 파일로 생성합니다.

Note

관리 사용자만 서비스 역할을 생성할 수 있도록 허용하는 것이 좋습니다. 역할을 생성하고 정 책을 연결할 수 있는 권한이 있는 사람은 자신의 권한을 에스컬레이션할 수 있습니다. 대신 필 요한 역할만 생성하거나 관리자가 대신 서비스 역할을 생성하도록 허용하는 정책을 만듭니다. 1. 터미널 창에서 다음 명령을 입력하여 역할 정책 JSON을 붙여 넣을 TrustPolicy.json이라는 이름의 파일을 생성합니다. 이 예제에서는 VIM을 사용합니다.

vim TrustPolicy.json

2. 파일에 다음 JSON을 붙여 넣습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
               "Service": "codepipeline.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
        }
   ]
}
```

파일을 저장하고 종료하려면 다음 VIM 명령을 입력합니다.

:wq

 터미널 창에서 다음 명령을 입력하여 역할 정책 JSON을 붙여 넣을 RolePolicy.json이라는 이 름의 파일을 생성합니다. 이 예제에서는 VIM을 사용합니다.

vim RolePolicy.json

 JSON 정책을 파일에 붙여 넣습니다. 에 제공된 최소 서비스 역할 정책을 사용합니다<u>CodePipeline</u> <u>서비스 역할 정책</u>. 또한 사용하려는 작업에 따라 서비스 역할에 적절한 권한을 추가합니다. 작업 목록과 각 작업의 필수 서비스 역할 권한에 대한 링크는 섹션을 참조하세요<u>CodePipeline 서비스</u> 역할에 권한 추가.

Resource 필드의 리소스 수준까지 범위를 축소하여 권한의 범위를 최대한 줄여야 합니다.

파일을 저장하고 종료하려면 다음 VIM 명령을 입력합니다.

:wq

 다음 명령을 입력하여 역할을 생성한 후 신뢰 역할 정책에 연결합니다. 정책 이름 형식은 일반적으 로 역할 이름 형식과 동일합니다. 이 예제에서는 별도의 파일로 생성된 역할 이름 MyRole과 정책 TrustPolicy를 사용합니다.

```
aws iam create-role --role-name MyRole --assume-role-policy-document file://
TrustPolicy.json
```

 다음 명령을 입력하여 역할 정책을 생성하고 역할에 연결합니다. 정책 이름 형식은 일반적으로 역할 이름 형식과 동일합니다. 이 예제에서는 별도의 파일로 생성된 역할 이름 MyRole과 정책 MyRole를 사용합니다.

```
aws iam put-role-policy --role-name MyRole --policy-name RolePolicy --policy-
document file://RolePolicy.json
```

생성된 역할 이름 및 신뢰 정책을 확인하려면 MyRole이라는 역할에 대해 다음 명령을 입력합니다.

```
aws iam get-role --role-name MyRole
```

8. AWS CLI 또는를 사용하여 파이프라인을 생성할 때 서비스 역할 ARN을 사용합니다 AWS CloudFormation.

리소스에 태그 지정

태그는 사용자 또는가 AWS 리소스에 AWS 할당하는 사용자 지정 속성 레이블입니다. 각 AWS 태그는 두 부분으로 구성됩니다.

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

태그 키와 태그 값을 합해서 키 값 페어라고 합니다.

태그는 AWS 리소스를 식별하고 구성하는 데 도움이 됩니다. 많은가 태그 지정을 AWS 서비스 지원하 므로 서로 다른 서비스의 리소스에 동일한 태그를 할당하여 리소스가 관련이 있음을 나타낼 수 있습니 다. 예를 들어 Amazon S3 소스 버킷에 할당한 것과 동일한 태그를 파이프라인에 할당할 수 있습니다.

태그 사용에 대한 팁은 AWS Answers 블로그의 게시글 <u>AWS Tagging Strategies</u>를 참조하세요.

CodePipeline에서 다음 리소스 유형에 태그를 지정할 수 있습니다.

- CodePipeline에서 파이프라인 태그 지정
- CodePipeline에서 사용자 지정 작업 태그 지정

AWS CLI CodePipeline APIs 또는 AWS SDKs

- 파이프라인, 사용자 지정 작업 또는 Webhook을 만들 때 태그를 추가합니다.
- 파이프라인, 사용자 지정 작업 또는 Webhook에 대한 태그를 추가, 관리 및 제거합니다.

또한 콘솔을 사용하여 파이프라인에 대한 태그를 추가, 관리 및 제거할 수 있습니다.

태그로 리소스를 식별, 구성 및 추적하는 것 외에도 IAM 정책의 태그를 사용하여 리소스를 보고 상 호 작용할 수 있는 사용자를 제어할 수 있습니다. 태그 기반 액세스 정책의 예는 <u>태그를 사용하여</u> CodePipeline 리소스에 대한 액세스 통제 단원을 참조하세요.

CodePipeline에서 파이프라인 태그 지정

태그는 AWS 리소스와 연결된 키-값 페어입니다. CodePipeline에서 파이프라인에 태그를 적용할 수 있 습니다. CodePipeline 리소스 태그 지정, 사용 사례, 태그 키 및 값 제약, 지원되는 리소스 유형에 대한 자세한 내용은 리소스에 태그 지정 단원을 참조하세요.

CLI를 사용하여 파이프라인을 만들 때 태그를 지정할 수 있습니다. 파이프라인에서 콘솔 또는 CLI를 사용하여 태그를 추가 또는 제거하고 태그 값을 업데이트할 수 있습니다. 각 파이프라인에 최대 50개의 태그를 추가할 수 있습니다.

주제

- 파이프라인 태그 지정(콘솔)
- <u>파이프라인 태그 지정(CLI)</u>

파이프라인 태그 지정(콘솔)

콘솔 또는 CLI를 사용하여 리소스에 태그를 지정할 수 있습니다. 파이프라인은 콘솔이나 CLI로 관리할 수 있는 유일한 CodePipeline 리소스입니다.

주제

• 파이프라인에 태그 추가(콘솔)

파이프라인 태그 지정

- 파이프라인에 대한 태그 보기(콘솔)
- 파이프라인에 대한 태그 편집(콘솔)
- 파이프라인에서 태그 제거(콘솔)

파이프라인에 태그 추가(콘솔)

콘솔을 사용하여 기존 파이프라인에 태그를 추가할 수 있습니다.

- 1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> <u>codepipeline/home</u>://https//https//ht
- 2. 파이프라인 페이지에서 태그를 추가할 파이프라인을 선택합니다.
- 3. 탐색 창에서 설정을 선택합니다.
- 4. 파이프라인 태그에서 편집을 선택합니다.
- 키 및 값 필드에 추가할 각 태그 세트에 대한 키 페어를 입력합니다. (값 필드는 선택 사항입니다.)
 예를 들어 키에 Project을 입력합니다. 값에는 ProjectA를 입력합니다.
- 6. (선택 사항) 행을 추가하고 태그를 더 입력하려면 태그 추가를 선택합니다.
- 7. 제출을 선택합니다. 태그는 파이프라인 설정 아래에 나열됩니다.

파이프라인에 대한 태그 보기(콘솔)

콘솔을 사용하여 기존 파이프라인에 대한 태그를 나열할 수 있습니다.

- 1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https//https//http
- 2. 파이프라인 페이지에서 태그를 볼 파이프라인을 선택합니다.
- 3. 탐색 창에서 설정을 선택합니다.
- 4. 파이프라인 태그에서 키 및 값 열 아래에 파이프라인에 대한 태그가 표시됩니다.

파이프라인에 대한 태그 편집(콘솔)

콘솔을 사용하여 파이프라인에 추가된 태그를 편집할 수 있습니다.

- 1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> <u>codepipeline/home</u>://https//https://https
- 2. 파이프라인 페이지에서 태그를 업데이트할 파이프라인을 선택합니다.

- 3. 탐색 창에서 설정을 선택합니다.
- 4. 파이프라인 태그에서 편집을 선택합니다.
- 5. 키 및 값 필드에서 필요에 따라 각 필드의 값을 업데이트합니다. 예를 들어 Project 키의 값에서 ProjectA를 ProjectB로 변경합니다.
- 6. 제출을 선택합니다.

파이프라인에서 태그 제거(콘솔)

콘솔을 사용하여 파이프라인에서 태그를 삭제할 수 있습니다. 연결된 리소스에서 태그를 제거하면 태 그가 삭제됩니다.

- 1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> <u>codepipeline/home</u>://https//https//h
- 2. 파이프라인 페이지에서 태그를 제거할 파이프라인을 선택합니다.
- 3. 탐색 창에서 설정을 선택합니다.
- 4. 파이프라인 태그에서 편집을 선택합니다.
- 5. 삭제할 각 태그의 키와 값 옆에 있는 태그 제거를 선택합니다.
- 6. 제출을 선택합니다.

파이프라인 태그 지정(CLI)

CLI를 사용하여 리소스에 태그를 지정할 수 있습니다. 파이프라인의 태그를 관리하려면 콘솔을 사용해 야 합니다.

주제

- <u>파이프라인에 태그 추가(CLI)</u>
- 파이프라인에 대한 태그 보기(CLI)
- <u>파이프라인에 대한 태그 편집(CLI)</u>
- 파이프라인에서 태그 제거(CLI)

파이프라인에 태그 추가(CLI)

콘솔 또는를 사용하여 파이프라인 AWS CLI 에 태그를 지정할 수 있습니다.

파이프라인을 생성할 때 태그를 추가하려면 파이프라인 스테이지 및 작업 생성 단원을 참조하십시오.

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

터미널이나 명령줄에서 tag-resource 명령을 실행하여, 태그를 추가할 파이프라인의 Amazon 리소스 이름(ARN)과 추가할 태그의 키와 값을 지정합니다. 파이프라인에 두 개 이상의 태그를 추가할 수 있습니다. 예를 들어, *MyPipeline*이라는 파이프라인에 두 개의 태그를 지정하고, *DeploymentEnvironment*라는 태그 키에는 *Test* 태그 값을, 그리고 *IscontainerBased*라는 태 그 키에는 태그 값이 *true*인 태그를 지정하려면 다음과 같이 하세요.

aws codepipeline tag-resource --resource-arn arn:aws:codepipeline:us-west-2:accountid:MyPipeline --tags key=Project,value=ProjectA key=IscontainerBased,value=true

성공한 경우 이 명령은 아무 것도 반환하지 않습니다.

파이프라인에 대한 태그 보기(CLI)

다음 단계에 따라를 사용하여 파이프라인의 AWS 태그를 AWS CLI 확인합니다. 태그가 추가되지 않은 경우 반환되는 목록은 비어 있습니다.

터미널 또는 명령줄에서 list-tags-for-resource 명령을 실행합니다. 예를 들어 arn:aws:codepipeline:*us-west-2:account-id:MyPipeline* ARN 값이 있는 *MyPipeline*이라는 파이프라인의 태그 키 및 태그 값 목록을 보려면

```
aws codepipeline list-tags-for-resource --resource-arn arn:aws:codepipeline:us-
west-2:account-id:MyPipeline
```

이 명령이 제대로 실행되면 다음과 비슷한 정보를 반환합니다.

```
{
    "tags": {
        "Project": "ProjectA",
        "IscontainerBased": "true"
    }
}
```

파이프라인에 대한 태그 편집(CLI)

다음 단계에 따라 AWS CLI 를 사용하여 파이프라인의 태그를 편집합니다. 기존 키의 값을 변경하거나 다른 키를 추가할 수 있습니다. 다음 단원에서 설명하는 것처럼 파이프라인에서 태그를 제거할 수도 있 습니다.

파이프라인 태그 지정(CLI)
터미널이나 명령줄에서 tag-resource 명령을 실행하여, 태그를 업데이트하고 태그 키 및 태그 값을 지 정할 파이프라인의 ARN을 지정합니다.

aws codepipeline tag-resource --resource-arn arn:aws:codepipeline:us-west-2:accountid:MyPipeline --tags key=Project,value=ProjectA

성공한 경우 이 명령은 아무 것도 반환하지 않습니다.

파이프라인에서 태그 제거(CLI)

다음 단계에 따라 AWS CLI 를 사용하여 파이프라인에서 태그를 제거합니다. 연결된 리소스에서 태그 를 제거하면 태그가 삭제됩니다.

Note

파이프라인을 삭제하면 삭제된 파이프라인에서 모든 태그 연결이 제거됩니다. 파이프라인을 삭제하기 전에 태그를 제거할 필요가 없습니다.

터미널이나 명령줄에서 untag-resource 명령을 실행하여, 태그를 제거할 파이프라인의 ARN과 제거 할 태그의 태그 키를 지정합니다. 예를 들어 *MyPipeline* 파이프라인에서 태그 키가 *Project* 및 *IscontainerBased*인 여러 태그를 제거하려면 다음을 수행하십시오.

aws codepipeline untag-resource --resource-arn arn:aws:codepipeline:us-west-2:accountid:MyPipeline --tag-keys Project IscontainerBased

성공한 경우 이 명령은 아무 것도 반환하지 않습니다. 파이프라인과 연결된 태그를 확인하려면 listtags-for-resource 명령을 실행하십시오.

알림 규칙 생성

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

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

알림 규칙을 생성하려면 (콘솔)

- 1. 에 로그인 AWS Management Console 하고 <u>https://console.aws.amazon.com/codepipeline/</u>:// https://https://https://i/https//i/https://i/https://i/https://i/https://i/https://i/https://i/https://i/https://i/https://i/https://i/https://i/https://i/https://i/https://i/https
- 2. 파이프라인을 선택한 다음 알림을 추가할 파이프라인을 선택합니다.
- 3. 파이프라인 페이지에서 알림을 선택한 다음 Create notification rule(알림 규칙 생성)을 선택합니다. 파이프라인의 설정 페이지로 이동하여 Create notification rule(알림 규칙 생성)을 선택할 수도 있습니다.
- 4. 알림 이름에 규칙에 대한 이름을 입력합니다.
- 5.

알림에 포함된 Amazon EventBridge에 제공된 정보만 원하는 경우 세부 정보 유형에서 기본을 선 택합니다. Amazon EventBridge에 제공된 정보와 CodePipeline 또는 알림 관리자에서 제공할 수 있는 정보를 포함하려는 경우 전체를 선택합니다.

자세한 내용은 <u>알림 내용 및 보안 이해</u>를 참조하세요.

- 알림을 트리거하는 이벤트에서 알림을 보내고자 하는 이벤트를 선택합니다. 자세한 내용은 <u>파이</u> 프라인의 고지 규정 이벤트를 참조하십시오.
- 7. 대상에서 다음 중 하나를 수행합니다.
 - 알림에 사용할 리소스를 이미 구성한 경우 대상 유형 선택에서 채팅 애플리케이션에서 Amazon Q Developer(Slack) 또는 SNS 주제를 선택합니다. 대상 선택에서 클라이언트 이름(채팅 애플리 케이션의 Amazon Q Developer에 구성된 Slack 클라이언트의 경우) 또는 Amazon SNS 주제의 Amazon 리소스 이름(ARN)(알림에 필요한 정책으로 이미 구성된 Amazon SNS 주제의 경우)을 선택합니다.
 - 알림과 함께 사용할 리소스를 구성하지 않은 경우 대상 생성을 선택한 다음 SNS 주제를 선택합니다. codestar-notifications- 뒤에 주제 이름을 입력한 다음 생성을 선택합니다.

Note

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

됩니다. 이 옵션을 선택하면 채팅 애플리케이션 콘솔에서 Amazon Q Developer가 열립 니다. 자세한 내용은 <u>채팅 애플리케이션에서 알림과 Amazon Q Developer 간의 통합</u> 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 내용 은 <u>알림에 대한 Amazon SNS 주제 구성</u>과 <u>알림 내용 및 보안 이해</u>를 참조하세요.
- 8. 규칙 생성을 완료하려면 제출을 선택합니다.
- 9. 사용자가 알림을 받으려면 먼저 규칙에 대한 Amazon SNS 주제를 사용자가 구독하도록 해야 합니다. 자세한 내용은 <u>대상인 Amazon SNS 주제에 사용자 구독</u>을 참조하세요. 채팅 애플리케이 션에서 알림과 Amazon Q Developer 간의 통합을 설정하여 Amazon Chime 채팅룸 또는 Slack 채널로 알림을 보낼 수도 있습니다. 자세한 내용은 <u>채팅 애플리케이션에서 알림과 Amazon Q</u> Developer 간의 통합 구성을 참조하세요.

알림 규칙을 생성하려면(AWS CLI)

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

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

2. 일반 텍스트 편집기에서 JSON 파일을 열고 규칙에 대해 원하는 리소스, 이벤트 유형 및 대상을 포 함하도록 편집합니다. 다음 예제는 ID가 123456789012인 AWS 계정의 MyDemoPipeline이라 는 파이프라인에 MyNotificationRule 대한 라는 알림 규칙을 보여줍니다. 파이프라인 실행이 시작될 때 codestar-notifications-MyNotificationTopic이라는 Amazon SNS 주제에 전체 세부 정보 유형과 함께 알림이 전송됩니다.

```
{
    "Name": "MyNotificationRule",
    "EventTypeIds": [
        "codepipeline-pipeline-pipeline-execution-started"
    ],
    "Resource": "arn:aws:codebuild:us-east-2:123456789012:MyDemoPipeline",
    "Targets": [
```

```
{
    "TargetType": "SNS",
    "TargetAddress": "arn:aws:sns:us-east-2:123456789012:codestar-
notifications-MyNotificationTopic"
    }
    ],
    "Status": "ENABLED",
    "DetailType": "FULL"
}
```

파일을 저장합니다.

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

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

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

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

소스 작업 및 변경 감지 방법

파이프라인에 소스 작업을 추가하면 해당 작업이 표에 설명된 추가 리소스와 함께 작동합니다.

Note

CodeCommit 및 S3 소스 작업에는 구성된 변경 감지 리소스(EventBridge 규칙)가 필요하거나 리포지토리에서 소스 변경 내용을 폴링하는 옵션을 사용해야 합니다. Bitbucket, GitHub 또는 GitHub Enterprise Server 소스 작업이 포함된 파이프라인의 경우 webhook를 설정하거나 기본 값을 폴링으로 설정할 필요가 없습니다. 연결 작업은 변경 감지를 관리합니다.

소스	추가 리소스 사용 여부	단계
CloudTrail 리소스 가 포함된 Amazon S3	이 소스 작업은 이벤트 규칙과 추가 CloudTrail 리소스를 사용합니다. CLI 또는 CloudFormation을 사용하 여 이 작업을 생성할 때 이러한 리소 스도 생성하고 관리합니다.	파이프라인 스테이지 및 작업 생성 및 EventBridge 및를 사용하는 Amazon S3 소스 작업에 연결 AWS CloudTrai I를 참조하세요.
CloudTrail 리소스 가 없는 Amazon S3	이 소스 작업은 추가 CloudTrail 리 소스 없이 이벤트 규칙이 있는 이벤 트에 대해 활성화된 버킷을 사용합 니다. CLI 또는 CloudFormation을 사용하여 이 작업을 생성할 때 이러 한 리소스도 생성하고 관리합니다.	파이프라인 스테이지 및 작업 생성 및 이벤트에 대해 활성화된 소스를 사용 하여 Amazon S3 소스 작업에 연결를 참조하세요.
Bitbucket Cloud	이 소스 작업은 연결 리소스를 사용 합니다.	<u>Bitbucket Cloud 연결</u> 부분 참조
AWS CodeCommit	Amazon EventBridge(권장). 콘솔에 서 생성 또는 편집된 CodeCommit 소스가 있는 파이프라인의 기본값입 니다.	<u>파이프라인 스테이지 및 작업 생성</u> 및 <u>CodeCommit 소스 작업 및 EventBrid</u> <u>ge</u> 를 참조하세요.

AWS CodePipeline

소스	추가 리소스 사용 여부	단계
Amazon ECR	Amazon EventBridge. 콘솔에서 생 성 또는 편집된 Amazon ECR 소스 가 있는 파이프라인 마법사에 의해 생성됩니다.	<u>파이프라인 스테이지 및 작업 생성</u> 및 <u>Amazon ECR 소스 작업 및 EventBrid</u> ge 리소스 섹션을 참조하세요.
GitHub 또는 GitHub Enterprise Cloud	이 소스 작업은 연결 리소스를 사용 합니다.	<u>GitHub 연결</u> 부분 참조
GitHub Enterprise Server	이 소스 작업은 연결 리소스 및 호스 트 리소스를 사용합니다.	<u>GitHub Enterprise Server 연결</u> 부분 참조
GitLab.com	이 소스 작업은 연결 리소스를 사용 합니다.	<u>GitLab.com 연결</u> 부분 참조
GitLab 자체 관리 형	이 소스 작업은 연결 리소스 및 호스 트 리소스를 사용합니다.	<u>GitLab 자체 관리형을 위한 연결</u> 부분 참조

폴링을 사용하는 파이프라인이 있는 경우, 권장되는 감지 방법을 사용하도록 파이프라인을 업데이트 할 수 있습니다. 자세한 내용은 <u>폴링 파이프라인을 권장되는 변경 감지 방법으로 업데이트</u> 단원을 참조 하십시오.

연결을 사용하는 소스 작업에 대한 변경 감지를 끄려면 <u>Bitbucket Cloud, GitHub, GitHub Enterprise</u> Server, GitLab.com 및 GitLab 자체 관리형 작업용 CodeStarSourceConnection을 참조하세요.

소스 작업을 사용하여 자사 소스 공급자에 연결

AWS CodePipeline 콘솔 또는를 사용하여 CodeCommit 또는 S3와 같은 소스 작업 공급자 AWS CLI 에 연결할 수 있습니다.

Note

콘솔을 사용하여 파이프라인을 생성하거나 편집하면 변경 감지 리소스가 자동으로 생성됩니 다. AWS CLI를 사용하여 파이프라인을 생성하는 경우 추가 리소스를 직접 생성해야 합니다. 자세한 내용은 CodeCommit 소스 작업 및 EventBridge 단원을 참조하십시오.

주제

- Amazon ECR 소스 작업 및 EventBridge 리소스
- 이벤트에 대해 활성화된 소스를 사용하여 Amazon S3 소스 작업에 연결
- EventBridge 및를 사용하는 Amazon S3 소스 작업에 연결 AWS CloudTrail
- CodeCommit 소스 작업 및 EventBridge

Amazon ECR 소스 작업 및 EventBridge 리소스

CodePipeline에 Amazon ECR 소스 작업을 추가하려면 다음 중 하나를 선택할 수 있습니다.

- CodePipeline 콘솔 파이프라인 생성 마법사(<u>사용자 지정 파이프라인 생성(콘솔)</u>) 또는 작업 편집 페 이지를 사용하여 Amazon ECR 공급자 옵션을 선택합니다. 콘솔은 소스가 변경될 때 파이프라인을 시작하는 EventBridge 규칙을 생성합니다.
- CLI를 사용하여 ECR 작업에 대한 작업 구성을 추가하고 다음과 같이 추가 리소스를 생성합니다.
 - <u>Amazon ECR 소스 작업 참조</u>의 ECR 예제 작업 구성을 사용하여 <u>파이프라인 생성(CLI)</u>과 같이 작 업을 생성합니다.
 - 변경 감지 방법은 기본적으로 소스를 폴링하여 파이프라인을 시작합니다. 정기적 확인을 비활성 화하고 감지 규칙 변경을 수동으로 생성해야 합니다. <u>Amazon ECR 소스에 대한 EventBridge 규칙</u> <u>생성(콘솔)</u>, <u>Amazon ECR 소스에 대한 EventBridge 규칙 생성(CLI)</u> 또는 <u>Amazon ECR 소스에 대</u> <u>한 EventBridge 규칙 생성(AWS CloudFormation 템플릿)</u> 방법 중 한 가지를 선택하세요.

주제

- Amazon ECR 소스에 대한 EventBridge 규칙 생성(콘솔)
- Amazon ECR 소스에 대한 EventBridge 규칙 생성(CLI)
- Amazon ECR 소스에 대한 EventBridge 규칙 생성(AWS CloudFormation 템플릿)

Amazon ECR 소스에 대한 EventBridge 규칙 생성(콘솔)

CodePipeline 작업에 사용할 EventBridge 규칙을 생성하려면(Amazon ECR 소스)

- 1. Amazon EventBridge 콘솔(https://console.aws.amazon.com/events/)을 엽니다.
- 2. 탐색 창에서 이벤트를 선택합니다.
- 3. 규칙 생성을 선택한 후 이벤트 소스의 서비스 이름에서 Elastic Container Registry(ECR)를 선택합 니다.
- 4. 이벤트 소스에서 이벤트 패턴을 선택합니다.

편집을 선택한 다음 cli-testing의 이미지 태그가 다음과 같은 eb-test 리포지토리의 이벤트 소스 창에 다음 예제 이벤트 패턴을 붙여넣습니다.

```
{
    "detail-type": [
        "ECR Image Action"
    ],
    "source": [
        "aws.ecr"
    ],
    "detail": {
        "action-type": [
             "PUSH"
        ],
        "image-tag": [
             "latest"
        ],
        "repository-name": [
             "eb-test"
        ],
        "result": [
             "SUCCESS"
        ]
    }
}
```

i Note

Amazon ECR 이벤트에 지원되는 전체 이벤트 패턴을 보려면 <u>Amazon ECR 이벤트 및</u> EventBridge 또는 Amazon Elastic Container Registry Events를 참조하세요.

5. 저장(Save)을 선택합니다.

[Event Pattern Preview] 창에서 규칙을 봅니다.

- 6. 대상 영역에서 CodePipeline을 선택합니다.
- 7. 이 규칙에 의해 시작되는 파이프라인의 파이프라인 ARN을 입력합니다.

1 Note

get-pipeline 명령을 실행한 후 메타데이터 출력에서 파이프라인 ARN을 찾을 수 있습니다. 파이프라인 ARN은 다음 형식으로 구성됩니다. arn:aws:codepipeline:*region:account:pipeline-name* 파이프라인 ARN 샘플: arn:aws:codepipeline:us-east-2:80398EXAMPLE:MyFirstPipeline

- 8. EventBridge에 EventBridge 규칙과 연결된 대상을 호출할 권한을 부여하는 IAM 서비스 역할을 생성하거나 지정합니다(이 경우에는 대상이 CodePipeline).
 - 이 특정 리소스에 대한 새 역할 생성을 선택하여 EventBridge에 파이프라인 실행을 시작할 권한 을 제공하는 서비스 역할을 생성합니다.
 - 기존 역할 사용을 선택하여 EventBridge에 파이프라인 실행을 시작할 권한을 제공하는 서비스 역할을 입력합니다.
- 9. (선택 사항) 특정 이미지 ID로 소스 재정의를 지정하려면 입력 변환기를 사용하여 데이터를 JSON 파라미터로 전달합니다.
 - 추가 설정을 폅니다.

대상 입력 구성에서 입력 변환기 구성을 선택합니다.

대화 창에서 내 이름 입력을 선택합니다. 입력 경로 상자에 다음 키-값 페어를 입력합니다.

{"revisionValue": "\$.detail.image-digest"}

• 템플릿 상자에 다음 키-값 페어를 입력합니다.

{		
		"sourceRevisions": {
		"actionName": "Source",
		"revisionType": "IMAGE_DIGEST",
		"revisionValue": "< <mark>revisionValue</mark> >"
		}
	}	

- 확인을 선택합니다.
- 10. 규칙 설정을 검토하여 요구 사항을 충족하는지 확인합니다.
- 11. 세부 정보 구성을 선택합니다.
- 규칙 세부 정보 구성 페이지에서 해당 규칙의 이름과 설명을 입력한 후 상태를 선택하여 규칙을 활 성화합니다.
- 13. 규칙이 만족스러우면 규칙 생성(Create rule)을 선택하세요.

Amazon ECR 소스에 대한 EventBridge 규칙 생성(CLI)

put-rule 명령을 호출해 다음을 지정합니다.

- 만들려는 규칙을 고유하게 식별하는 이름. 이 이름은 AWS 계정과 연결된 CodePipeline으로 생성하는 모든 파이프라인에서 고유해야 합니다.
- 소스의 이벤트 패턴 및 규칙에서 사용하는 세부 정보 필드. 자세한 내용은 <u>Amazon EventBridge 및</u> 이벤트 패턴을 참조하세요.

Amazon ECR이 이벤트 소스이고 CodePipeline이 대상인 EventBridge 규칙을 만들려면

- 1. CodePipeline을 사용하여 규칙을 호출하도록 EventBridge에 권한을 추가합니다. 자세한 내용은 Amazon EventBridge에 대한 리소스 기반 정책 사용을 참조하세요.
 - a. 다음 예제를 사용하여 EventBridge가 서비스 역할을 담당하도록 허용하는 신뢰 정책을 생성 합니다. 신뢰 정책 이름을 trustpolicyforEB.json으로 지정합니다.

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

```
"Principal": {
    "Service": "events.amazonaws.com"
},
    "Action": "sts:AssumeRole"
}
]
```

b. 다음 명령을 사용하여 Role-for-MyRule 역할을 생성한 후 신뢰 정책에 연결합니다.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
file://trustpolicyforEB.json
```

c. 이 샘플에서 보이는 것처럼 MyFirstPipeline이라는 파이프라인에 대한 권한 정책 JSON 을 생성합니다. 권한 정책 이름을 permissionspolicyforEB.json으로 지정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "codepipeline:StartPipelineExecution"
        ],
            "Resource": [
               "arn:aws:codepipeline:us-west-2:80398EXAMPLE:MyFirstPipeline"
        ]
        }
    ]
}
```

d. 다음 명령을 사용하여 CodePipeline-Permissions-Policy-for-EB 권한 정책을 Role-for-MyRule 역할에 연결합니다.

이렇게 변경하는 이유는 무엇입니까? 이 정책을 역할에 추가하면 EventBridge에 대한 권한이 생성됩니다.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. put-rule 명령을 호출하고 --name, --event-pattern 및 --role-arn 파라미터를 포함시킵니다.

이렇게 변경하는 이유는 무엇입니까? 이미지 푸시가 수행되는 방식을 지정하는 규칙과 이벤트에 서 시작되는 파이프라인의 이름을 지정하는 대상을 사용하여 이벤트를 생성해야 합니다.

다음 샘플 명령은 MyECRRepoRule이라는 역할 별칭을 생성합니다.

```
aws events put-rule --name "MyECRRepoRule" --event-pattern "{\"detail-type\":[\"ECR
Image Action\"],\"source\":[\"aws.ecr\"],\"detail\":{\"action-type\":[\"PUSH\"],
\"image-tag\":[\"latest\"],\"repository-name\":[\"eb-test\"],\"result\":[\"SUCCESS
\"]}}" --role-arn "arn:aws:iam::ACCOUNT_ID:role/Role-for-MyRule"
```

Note

Amazon ECR 이벤트에 지원되는 전체 이벤트 패턴을 보려면 <u>Amazon ECR 이벤트 및</u> EventBridge 또는 <u>Amazon Elastic Container Registry Events</u>를 참조하세요.

- 3. CodePipeline을 대상으로 추가하려면 put-targets 명령을 호출하고 다음 파라미터를 포함합니다.
 - --rule 파라미터는 put-rule을 사용하여 생성한 rule_name에 사용됩니다.
 - --targets 파라미터는 대상 목록에 있는 대상의 목록 Id 및 대상 파이프라인의 ARN에 사용됩니다.

다음 예제 명령은 MyECRRepoRule이라는 규칙에 대해 대상 Id가 숫자 1로 구성됨을 지정하며, 규칙의 대상 목록에서 1로 대상 1로 표시됩니다. 예제 명령은 파이프라인에 대한 예 Arn과 규칙에 대한 예 RoleArn도 지정합니다. 파이프라인은 리포지토리에서 변경이 발생하면 시작됩니다.

```
aws events put-targets --rule MyECRRepoRule --targets
Id=1,Arn=arn:aws:codepipeline:us-
west-2:80398EXAMPLE:TestPipeline,RoleArn=arn:aws:iam::80398EXAMPLE:role/Role-for-
MyRule
```

- 4. (선택 사항) 특정 이미지 ID에 대한 소스 재정의로 입력 변환기를 구성하려면 CLI 명령에서 다음 JSON을 사용합니다. 다음 예제에서는 다음과 같은 재정의를 구성합니다.
 - 이 예제actionNameSource에서는 소스 이벤트에서 파생되지 않은 파이프라인 생성 시 정의 된 동적 값입니다.
 - 이 예제revisionTypeIMAGE_DIGEST에서는 소스 이벤트에서 파생되지 않은 파이프라인 생성 시 정의된 동적 값입니다.

• 이 예제의, revisionValue<revisionValue>는 소스 이벤트 변수에서 파생됩니다.

```
{
    "Rule": "my-rule",
    "Targets": [
        {
            "Id": "MyTargetId",
            "Arn": "ARN",
            "InputTransformer": {
                "InputPathsMap": {
                     "revisionValue": "$.detail.image-digest"
                },
                "InputTemplate": {
                     "sourceRevisions": {
                         "actionName": "Source",
                         "revisionType": "IMAGE_DIGEST",
                         "revisionValue": "<revisionValue>"
                     }
                }
            }
        }
    ]
}
```

Amazon ECR 소스에 대한 EventBridge 규칙 생성(AWS CloudFormation 템 플릿)

AWS CloudFormation 를 사용하여 규칙을 생성하려면 여기에 표시된 템플릿 코드 조각을 사용합니다.

파이프라인 AWS CloudFormation 템플릿을 업데이트하고 EventBridge 규칙을 생성하려면

- 템플릿의에서 AWS::IAM::Role AWS CloudFormation 리소스를 Resources사용하여 이벤트가 파이프라인을 시작하도록 허용하는 IAM 역할을 구성합니다. 이 항목은 두 가지 정책을 사용하는 역할을 만듭니다.
 - 첫 번째 정책은 가 역할을 수임하도록 허용합니다.
 - 두 번째 정책은 파이프라인을 시작할 권한을 부여합니다.

이렇게 변경하는 이유는 무엇입니까? 파이프라인에서 실행을 시작하기 위해 EventBridge가 수임 할 수 있는 역할을 생성해야 합니다.

YAML

```
EventRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
            Effect: Allow
            Principal:
              Service:
                - events.amazonaws.com
            Action: sts:AssumeRole
      Path: /
      Policies:
          PolicyName: eb-pipeline-execution
          PolicyDocument:
            Version: 2012-10-17
            Statement:
                Effect: Allow
                Action: codepipeline:StartPipelineExecution
                Resource: !Sub arn:aws:codepipeline:${AWS::Region}:
${AWS::AccountId}:${AppPipeline}
```

JSON



2. 템플릿의에서 AWS::Events::Rule AWS CloudFormation 리소스를 Resources사용하여 Amazon ECR 소스에 대한 EventBridge 규칙을 추가합니다. 이 이벤트 패턴은 리포지토리에 대한 커밋을 모니터링하는 이벤트를 생성합니다.EventBridge가 리포지토리 상태 변경을 감지하면 해 당 규칙이 대상 파이프라인에서 StartPipelineExecution을 호출합니다.

이렇게 변경하는 이유는 무엇입니까? 이미지 푸시가 수행되는 방식을 지정하는 규칙과 이벤트에 서 시작되는 파이프라인의 이름을 지정하는 대상을 사용하여 이벤트를 생성해야 합니다.

이 조각은 latest 태그와 함께 eb-test라는 이미지를 사용합니다.

YAML

```
EventRule:
  Type: 'AWS::Events::Rule'
  Properties:
    EventPattern:
      detail:
        action-type: [PUSH]
        image-tag: [latest]
        repository-name: [eb-test]
        result: [SUCCESS]
      detail-type: [ECR Image Action]
      source: [aws.ecr]
    Targets:
      - Arn: !Sub arn:aws:codepipeline:${AWS::Region}:${AWS::AccountId}:
${AppPipeline}
        RoleArn: !GetAtt
          - EventRole
          - Arn
        Id: codepipeline-AppPipeline
```

JSON

```
{
    "EventRule": {
        "Type": "AWS::Events::Rule",
        "Properties": {
            "EventPattern": {
                 "detail": {
                     "action-type": [
                         "PUSH"
                     ],
                     "image-tag": [
                         "latest"
                     ],
                     "repository-name": [
                         "eb-test"
                     ],
                     "result": [
                         "SUCCESS"
                     ٦
```



Note

Amazon ECR 이벤트에 지원되는 전체 이벤트 패턴을 보려면 <u>Amazon ECR 이벤트 및</u> <u>EventBridge</u> 또는 <u>Amazon Elastic Container Registry Events</u>를 참조하세요.

- (선택 사항) 특정 이미지 ID에 대한 소스 재정의로 입력 변환기를 구성하려면 다음 YAML 코드 조 각을 사용합니다. 다음 예제에서는 다음과 같은 재정의를 구성합니다.
 - 이 예제actionNameSource에서는 소스 이벤트에서 파생되지 않은 파이프라인 생성 시 정의 된 동적 값입니다.
 - 이 예제revisionTypeIMAGE_DIGEST에서는 소스 이벤트에서 파생되지 않은 파이프라인 생성 시 정의된 동적 값입니다.
 - 이 예제의, revisionValue<*revisionValue*>는 소스 이벤트 변수에서 파생됩니다.

Rule: my-rule
Targets:
- Id: MyTargetId
Arn: ARN
InputTransformer:
InputPathsMap:
<pre>revisionValue: "\$.detail.image-digest"</pre>
InputTemplate:
sourceRevisions:
actionName: Source
revisionType: IMAGE_DIGEST
revisionValue: '< <u>revisionValue</u> >'

- 4. 업데이트된 템플릿을 로컬 컴퓨터에 저장하고 AWS CloudFormation 콘솔을 엽니다.
- 5. 스택을 선택한 후 현재 스택에 대한 변경 세트 만들기를 선택합니다.
- 템플릿을 업로드한 후 AWS CloudFormation에 나열된 변경 사항을 확인합니다. 이는 스택에 적용 될 변경 사항입니다. 목록에 새로운 리소스가 표시됩니다.
- 7. 실행을 선택합니다.

이벤트에 대해 활성화된 소스를 사용하여 Amazon S3 소스 작업에 연결

이 섹션의 지침은 AWS CloudTrail 리소스를 생성하거나 관리할 필요가 없는 S3 소스 작업을 생성하는 단계를 제공합니다.

🛕 Important

AWS CloudTrail 리소스 없이이 작업을 생성하는 절차는 콘솔에서 사용할 수 없습니다. CLI를 사용하려면 여기의 절차를 참조하거나 섹션을 참조하세요<u>이벤트용으로 활성화된 S3 소스를</u> 사용하여 폴링 파이프라인 마이그레이션.

Amazon S3 소스가 있는 파이프라인의 경우, EventBridge와 이벤트 알림이 활성화된 소스 버킷을 통해 변경 감지가 자동화되도록 파이프라인을 수정하세요. CLI를 사용하거나 파이프라인을 마이그레이션 AWS CloudFormation 하는 경우 권장되는 방법입니다.

Note

여기에는 별도의 CloudTrail 트레일을 생성할 필요가 없는 이벤트 알림이 활성화된 버킷을 사용하는 것도 포함됩니다. 콘솔을 사용하는 경우 이벤트 규칙과 CloudTrail 트레일이 자동으로 설정됩니다. 이러한 단계는 <u>S3 소스 및 CloudTrail 트레일을 사용하여 폴링 파이프라인 마이그</u> <u>레이션</u>을 참조하세요.

- CLI: S3 소스 및 CloudTrail 트레일을 사용하여 폴링 파이프라인 마이그레이션(CLI)
- AWS CloudFormation: <u>S3 소스 및 CloudTrail 추적을 사용하여 폴링 파이프라인 마이그레이션(AWS</u> <u>CloudFormation 템플릿)</u>

이벤트에 대해 활성화된 S3 소스를 사용하여 파이프라인 생성(CLI)

다음 단계에 따라 변경 감지를 위해 EventBridge에서 이벤트를 사용하는 S3 소스가 있는 파이프라인을 생성합니다. CLI를 사용하여 파이프라인을 생성하는 전체 단계는 섹션을 참조하세요<u>파이프라인 스테</u> 이지 및 작업 생성.

Amazon S3를 사용하여 이벤트 기반 파이프라인을 빌드하려면 파이프라인의 PollForSourceChanges 파라미터를 편집한 후 다음 리소스를 생성합니다.

- EventBridge 이벤트 규칙
- EventBridge 이벤트가 파이프라인을 시작하도록 허용하는 IAM 역할

Amazon S3가 이벤트 소스이고 CodePipeline이 대상인 EventBridge 규칙을 생성하고 권한 정책을 적 용하려면

- 1. CodePipeline을 사용하여 규칙을 호출하도록 EventBridge에 권한을 부여합니다. 자세한 내용은 Amazon EventBridge에 대한 리소스 기반 정책 사용을 참조하세요.
 - a. 다음 예제를 사용하여 EventBridge가 서비스 역할을 담당하도록 허용하는 신뢰 정책을 생성 합니다. 이름을 trustpolicyforEB.json로 지정합니다.

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

```
"Principal": {
    "Service": "events.amazonaws.com"
    },
    "Action": "sts:AssumeRole"
    }
]
```

b. 다음 명령을 사용하여 Role-for-MyRule 역할을 생성한 후 신뢰 정책에 연결합니다.

이렇게 변경하는 이유는 무엇입니까? 이 신뢰 정책을 역할에 추가하면 EventBridge에 대한 권한이 생성됩니다.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
file://trustpolicyforEB.json
```

c. 이 샘플에서 보이는 것처럼 MyFirstPipeline이라는 파이프라인에 대한 권한 정책 JSON 을 생성합니다. 권한 정책 이름을 permissionspolicyforEB.json으로 지정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "codepipeline:StartPipelineExecution"
        ],
        "Resource": [
              "arn:aws:codepipeline:us-west-2:80398EXAMPLE:MyFirstPipeline"
        ]
        }
    ]
}
```

d. 다음 명령을 사용하여 앞에서 생성한 Role-for-MyRule 역할에 새로운 CodePipeline-Permissions-Policy-for-EB 권한 정책을 연결합니다.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. put-rule 명령을 호출하고 --name, --event-pattern 및 --role-arn 파라미터를 포함시킵니 다. 다음 샘플 명령은 EnabledS3SourceRule이라는 역할 별칭을 생성합니다.

aws events put-rule --name "EnabledS3SourceRule" --event-pattern "{\"source\":
[\"aws.s3\"],\"detail-type\":[\"Object Created\"],\"detail\":{\"bucket\":{\"name\":
[\"amzn-s3-demo-source-bucket\"]}}" --role-arn "arn:aws:iam::ACCOUNT_ID:role/Rolefor-MyRule"

3. CodePipeline을 대상으로 추가하려면 put-targets 명령을 호출하고 --rule 및 --targets 파라 미터를 포함합니다.

다음 명령은 EnabledS3SourceRule이라는 규칙에 대해 대상 Id가 숫자 1로 구성됨을 지정 하며, 규칙에 대한 대상 목록에서 대상 1로 표시됩니다. 이 명령은 또한 파이프라인에 대한 예제 ARN를 지정합니다. 파이프라인은 리포지토리에서 변경이 발생하면 시작됩니다.

```
aws events put-targets --rule EnabledS3SourceRule --targets Id=codepipeline-
AppPipeline,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
```

파이프라인의 PollForSourceChanges 파라미터를 편집하려면

🛕 Important

이 방법으로 파이프라인을 생성할 때 명시적으로 false로 설정되지 않은 경우 PollForSourceChanges 파라미터 기본값은 true입니다. 이벤트 기반 변경 감지를 추가할 때 는 출력에 파라미터를 추가하고 false로 설정하여 폴링을 비활성화해야 합니다. 그렇지 않으면 파이프라인이 단일 소스 변경 시 두 번 시작됩니다. 세부 정보는 <u>PollForSourceChanges 파</u> <u>라미터의 유효한 설정</u>을 참조하세요.

 get-pipeline 명령을 실행하여 파이프라인 구조를 JSON 파일로 복사합니다. 예를 들어, MyFirstPipeline라는 파이프라인의 경우 다음 명령을 입력합니다.

aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json

이 명령은 아무 것도 반환하지 않지만 생성한 파일이 명령을 실행한 디렉터리에 표시되어야 합니 다. 2. 일반 텍스트 편집기에서 JSON 파일을 열고 다음 예에 나와 있는 것처럼 amzn-s3-demosource-bucket 버킷의 PollForSourceChanges 파라미터를 false로 변경하여 소스 단계를 편집합니다.

이렇게 변경하는 이유는 무엇입니까? 이 파라미터를 false로 설정하면 정기적 확인이 비활성화 되어 이벤트 기반 변경 탐지만 사용할 수 있습니다.

```
"configuration": {
    "S3Bucket": "amzn-s3-demo-source-bucket",
    "PollForSourceChanges": "false",
    "S3ObjectKey": "index.zip"
},
```

 get-pipeline 명령을 사용하여 검색한 파이프라인 구조로 작업을 수행할 경우, JSON 파일에서 metadata 행을 제거해야 합니다. 이렇게 하지 않으면 update-pipeline 명령에서 사용할 수 없습니 다. "metadata": { } 행과, "created", "pipelineARN" 및 "updated" 필드를 제거합니다.

예를 들어, 구조에서 다음 행을 삭제합니다.

```
"metadata": {
    "pipelineArn": "arn:aws:codepipeline:region:account-ID:pipeline-name",
    "created": "date",
    "updated": "date"
},
```

파일을 저장합니다.

4. 변경 사항을 적용하려면 파이프라인 JSON 파일을 지정하여 update-pipeline 명령을 실행합니다.

Important

파일 이름 앞에 file://를 포함해야 합니다. 이 명령에 필수적입니다.

aws codepipeline update-pipeline --cli-input-json file://pipeline.json

이 명령은 편집한 파이프라인의 전체 구조를 반환합니다.

Note

update-pipeline 명령을 실행하면 파이프라인이 중지됩니다. update-pipeline 명령을 실행 할 때 파이프라인을 통해 개정을 실행하는 중이라면 해당 실행이 중지됩니다. 업데이트 된 파이프라인을 통해 해당 개정을 실행하려면 파이프라인을 수동으로 시작해야 합니다. start-pipeline-execution 명령을 사용하여 수동으로 파이프라인을 시작합니다.

이벤트에 대해 활성화된 S3 소스를 사용하여 파이프라인 생성(AWS CloudFormation 템플릿)

이 절차는 소스 버킷에 이벤트가 활성화된 파이프라인을 위한 것입니다.

다음 단계에 따라 이벤트 기반 변경 감지를 위해 Amazon S3 소스가 있는 파이프라인을 생성합니다.

Amazon S3를 사용하여 이벤트 기반 파이프라인을 빌드하려면 해당 파이프라인의 PollForSourceChanges 파라미터를 편집한 후 다음 리소스를 템플릿에 추가합니다.

• 이 이벤트가 파이프라인을 시작할 수 있게 하는 EventBridge 규칙 및 IAM 역할.

AWS CloudFormation 를 사용하여 파이프라인을 생성하고 관리하는 경우 템플릿에는 다음과 같은 콘 텐츠가 포함됩니다.

Note

PollForSourceChanges라고 하는 소스 단계의 Configuration 속성. 템플릿에 해당 속성 이 포함되어 있지 않으면 기본적으로 PollForSourceChanges가 true로 설정됩니다.

YAML

```
AppPipeline:
Type: AWS::CodePipeline::Pipeline
Properties:
RoleArn: !GetAtt CodePipelineServiceRole.Arn
Stages:
-
Name: Source
```

Actions: -Name: SourceAction ActionTypeId: Category: Source Owner: AWS Version: 1 Provider: S3 OutputArtifacts: -Name: SourceOutput Configuration: S3Bucket: !Ref SourceBucket S3ObjectKey: !Ref S3SourceObjectKey PollForSourceChanges: true RunOrder: 1

•••

JSON

```
"AppPipeline": {
    "Type": "AWS::CodePipeline::Pipeline",
    "Properties": {
        "RoleArn": {
            "Fn::GetAtt": ["CodePipelineServiceRole", "Arn"]
        },
        "Stages": [
            {
                "Name": "Source",
                "Actions": [
                    {
                        "Name": "SourceAction",
                        "ActionTypeId": {
                             "Category": "Source",
                             "Owner": "AWS",
                             "Version": 1,
                             "Provider": "S3"
                        },
                         "OutputArtifacts": [
                             {
                                 "Name": "SourceOutput"
```

```
}
                                  ],
                                  "Configuration": {
                                      "S3Bucket": {
                                          "Ref": "SourceBucket"
                                      },
                                      "S3ObjectKey": {
                                          "Ref": "SourceObjectKey"
                                      },
                                      "PollForSourceChanges": true
                                  },
                                  "RunOrder": 1
                             }
                         ]
                     },
. . .
```

Amazon S3가 이벤트 소스이고 CodePipeline이 대상인 EventBridge 규칙을 생성하고 권한 정책을 적 용하려면

- 템플릿의에서 AWS::IAM::Role AWS CloudFormation 리소스를 Resources사용하여 이벤트가 파이프라인을 시작하도록 허용하는 IAM 역할을 구성합니다. 이 항목은 두 가지 정책을 사용하는 역할을 만듭니다.
 - 첫 번째 정책은 가 역할을 수임하도록 허용합니다.
 - 두 번째 정책은 파이프라인을 시작할 권한을 부여합니다.

이렇게 변경하는 이유는 무엇입니까? AWS::IAM::Role 리소스를 추가하면 AWS CloudFormation 가 EventBridge에 대한 권한을 생성할 수 있습니다. 이 리소스는 AWS CloudFormation 스택에 추가됩니다.

YAML

```
EventRole:
Type: AWS::IAM::Role
Properties:
AssumeRolePolicyDocument:
Version: 2012-10-17
```

```
Statement:
            Effect: Allow
            Principal:
              Service:
                - events.amazonaws.com
            Action: sts:AssumeRole
     Path: /
     Policies:
         PolicyName: eb-pipeline-execution
         PolicyDocument:
            Version: 2012-10-17
            Statement:
                Effect: Allow
                Action: codepipeline:StartPipelineExecution
                Resource: !Join [ '', [ 'arn:aws:codepipeline:', !Ref
'AWS::Region', ':', !Ref 'AWS::AccountId', ':', !Ref AppPipeline ] ]
. . .
```

```
JSON
```

```
"EventRole": {
  "Type": "AWS::IAM::Role",
  "Properties": {
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": [
              "events.amazonaws.com"
            ]
          },
          "Action": "sts:AssumeRole"
        }
      1
    },
    "Path": "/",
```

```
"Policies": [
        {
          "PolicyName": "eb-pipeline-execution",
          "PolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
              {
                "Effect": "Allow",
                "Action": "codepipeline:StartPipelineExecution",
                "Resource": {
                  "Fn::Join": [
                    "",
                     Γ
                       "arn:aws:codepipeline:",
                       {
                         "Ref": "AWS::Region"
                       },
                       ":",
                       {
                         "Ref": "AWS::AccountId"
                       },
                       ":",
                       {
                         "Ref": "AppPipeline"
                      }
                    ]
                  ]
. . .
```

AWS::Events::Rule AWS CloudFormation 리소스를 사용하여 EventBridge 규칙을 추가합니다. 이 이벤트 패턴은 Amazon S3 원본 버킷의 객체 생성 또는 삭제를 모니터링하는 이벤트를 생성합니다. 또한 파이프라인 대상도 포함하세요. 객체가 생성되면 이 규칙이 대상 파이프라인의 StartPipelineExecution을 호출합니다.

이렇게 변경하는 이유는 무엇입니까? AWS::Events::Rule 리소스를 추가 AWS CloudFormation 하면가 이벤트를 생성할 수 있습니다. 이 리소스는 AWS CloudFormation 스택에 추가됩니다.

YAML

EventRule: Type: AWS::Events::Rule

```
Properties:
     EventBusName: default
     EventPattern:
        source:
          - aws.s3
        detail-type:
          - Object Created
        detail:
          bucket:
            name:
              - !Ref SourceBucket
     Name: EnabledS3SourceRule
     State: ENABLED
     Targets:
        -
          Arn:
            !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region', ':', !Ref
'AWS::AccountId', ':', !Ref AppPipeline ] ]
          RoleArn: !GetAtt EventRole.Arn
          Id: codepipeline-AppPipeline
. . .
```

JSON

```
"EventRule": {
  "Type": "AWS::Events::Rule",
 "Properties": {
"EventBusName": "default",
"EventPattern": {
   "source": [
"aws.s3"
   ],
   "detail-type": [
  "Object Created"
    ],
    "detail": {
  "bucket": {
      "name": [
    "s3-pipeline-source-fra-bucket"
      ]
```

```
}
            }
  },
  "Name": "EnabledS3SourceRule",
        "State": "ENABLED",
        "Targets": [
        {
          "Arn": {
            "Fn::Join": [
               "",
               Ε
                 "arn:aws:codepipeline:",
                 {
                   "Ref": "AWS::Region"
                 },
                 ":",
                 {
                  "Ref": "AWS::AccountId"
                 },
                 ":",
                 {
                   "Ref": "AppPipeline"
                 }
               ]
            1
          },
          "RoleArn": {
            "Fn::GetAtt": [
               "EventRole",
              "Arn"
            ]
          },
          "Id": "codepipeline-AppPipeline"
        }
      ]
    }
  }
},
. . .
```

- 3. 업데이트된 템플릿을 로컬 컴퓨터에 저장하고 AWS CloudFormation 콘솔을 엽니다.
- 4. 스택을 선택한 후 현재 스택에 대한 변경 세트 만들기를 선택합니다.

- 5. 업데이트된 템플릿을 업로드한 후 AWS CloudFormation에 나열된 변경 사항을 확인합니다. 이는 스택에 적용될 변경 사항입니다. 목록에 새로운 리소스가 표시됩니다.
- 6. 실행을 선택합니다.

파이프라인의 PollForSourceChanges 파라미터를 편집하려면

▲ Important

이 방법으로 파이프라인을 생성할 때 명시적으로 false로 설정되지 않은 경우 PollForSourceChanges 파라미터 기본값은 true입니다. 이벤트 기반 변경 감지를 추가할 때 는 출력에 파라미터를 추가하고 false로 설정하여 폴링을 비활성화해야 합니다. 그렇지 않으면 파이프라인이 단일 소스 변경 시 두 번 시작됩니다. 세부 정보는 <u>PollForSourceChanges 파</u> 라미터의 유효한 설정을 참조하세요.

템플릿에서 PollForSourceChanges를 false로 변경합니다. PollForSourceChanges를 파 이프라인 정의에 포함하지 않은 경우 추가하고 false로 설정하세요.

이렇게 변경하는 이유는 무엇입니까? PollForSourceChanges를 false로 변경하면 정기적 확 인이 비활성화되어 이벤트 기반 변경 탐지만 사용할 수 있습니다.

YAML

Name: Source
Actions:
-
Name: SourceAction
ActionTypeId:
Category: Source
Owner: AWS
Version: 1
Provider: S3
OutputArtifacts:
- Name: SourceOutput
Configuration:
S3Bucket: !Ref SourceBucket
S3ObjectKey: !Ref SourceObjectKey
PollForSourceChanges: false
RunOrder: 1

JSON

{

},

{

}],

},

},

"RunOrder": 1

},

}

"S3ObjectKey": {

"Ref": "SourceObjectKey"

"PollForSourceChanges": false

```
"Name": "SourceAction",
"ActionTypeId": {
  "Category": "Source",
  "Owner": "AWS",
 "Version": 1,
 "Provider": "S3"
"OutputArtifacts": [
    "Name": "SourceOutput"
"Configuration": {
 "S3Bucket": {
    "Ref": "SourceBucket"
```

Example

AWS CloudFormation 를 사용하여 이러한 리소스를 생성하면 리포지토리의 파일이 생성되거나 업데 이트될 때 파이프라인이 트리거됩니다.

Note

여기에서 중단하지 마십시오. 파이프라인이 생성되더라도 Amazon S3 파이프라인에 대한 두 번째 AWS CloudFormation 템플릿을 생성해야 합니다. 두 번째 템플릿을 생성하지 않으면 파 이프라인에 변경 감지 기능이 없습니다.

YAML

```
Parameters:
  SourceObjectKey:
    Description: 'S3 source artifact'
    Type: String
    Default: SampleApp_Linux.zip
 ApplicationName:
    Description: 'CodeDeploy application name'
    Type: String
    Default: DemoApplication
  BetaFleet:
    Description: 'Fleet configured in CodeDeploy'
    Type: String
    Default: DemoFleet
Resources:
  SourceBucket:
    Type: AWS::S3::Bucket
    Properties:
      NotificationConfiguration:
        EventBridgeConfiguration:
          EventBridgeEnabled: true
      VersioningConfiguration:
        Status: Enabled
  CodePipelineArtifactStoreBucket:
    Type: AWS::S3::Bucket
  CodePipelineArtifactStoreBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket: !Ref CodePipelineArtifactStoreBucket
      PolicyDocument:
        Version: 2012-10-17
        Statement:
            Sid: DenyUnEncryptedObjectUploads
            Effect: Deny
            Principal: '*'
            Action: s3:PutObject
            Resource: !Join [ '', [ !GetAtt CodePipelineArtifactStoreBucket.Arn, '/
*']]
            Condition:
              StringNotEquals:
                s3:x-amz-server-side-encryption: aws:kms
```

```
Sid: DenyInsecureConnections
          Effect: Deny
          Principal: '*'
          Action: s3:*
          Resource: !Sub ${CodePipelineArtifactStoreBucket.Arn}/*
          Condition:
            Bool:
              aws:SecureTransport: false
CodePipelineServiceRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: 2012-10-17
      Statement:
          Effect: Allow
          Principal:
            Service:
              - codepipeline.amazonaws.com
          Action: sts:AssumeRole
    Path: /
    Policies:
        PolicyName: AWS-CodePipeline-Service-3
        PolicyDocument:
          Version: 2012-10-17
          Statement:
              Effect: Allow
              Action:
                - codecommit:CancelUploadArchive
                - codecommit:GetBranch
                - codecommit:GetCommit
                - codecommit:GetUploadArchiveStatus
                - codecommit:UploadArchive
              Resource: 'resource_ARN'
              Effect: Allow
              Action:
                - codedeploy:CreateDeployment
                - codedeploy:GetApplicationRevision
                - codedeploy:GetDeployment
                - codedeploy:GetDeploymentConfig
```

```
- codedeploy:RegisterApplicationRevision
Resource: 'resource_ARN'
Effect: Allow
Action:
  - codebuild:BatchGetBuilds
  - codebuild:StartBuild
Resource: 'resource_ARN'
Effect: Allow
Action:
  - devicefarm:ListProjects
  - devicefarm:ListDevicePools
  - devicefarm:GetRun
  - devicefarm:GetUpload
  - devicefarm:CreateUpload
  - devicefarm:ScheduleRun
Resource: 'resource_ARN'
Effect: Allow
Action:
  - lambda:InvokeFunction
  - lambda:ListFunctions
Resource: 'resource_ARN'
Effect: Allow
Action:
  - iam:PassRole
Resource: 'resource_ARN'
Effect: Allow
Action:
  - elasticbeanstalk:*
  - ec2:*
  - elasticloadbalancing:*
  - autoscaling:*
  - cloudwatch:*
  - s3:*
  - sns:*
  - cloudformation:*
  - rds:*
  - sqs:*
  - ecs:*
Resource: 'resource_ARN'
```

```
AppPipeline:
  Type: AWS::CodePipeline::Pipeline
  Properties:
    Name: s3-events-pipeline
    RoleArn:
      !GetAtt CodePipelineServiceRole.Arn
    Stages:
        Name: Source
        Actions:
          _
            Name: SourceAction
            ActionTypeId:
              Category: Source
              Owner: AWS
              Version: 1
              Provider: S3
            OutputArtifacts:
              - Name: SourceOutput
            Configuration:
              S3Bucket: !Ref SourceBucket
              S3ObjectKey: !Ref SourceObjectKey
              PollForSourceChanges: false
            RunOrder: 1
        Name: Beta
        Actions:
            Name: BetaAction
            InputArtifacts:
              - Name: SourceOutput
            ActionTypeId:
              Category: Deploy
              Owner: AWS
              Version: 1
              Provider: CodeDeploy
            Configuration:
              ApplicationName: !Ref ApplicationName
              DeploymentGroupName: !Ref BetaFleet
            RunOrder: 1
    ArtifactStore:
      Type: S3
      Location: !Ref CodePipelineArtifactStoreBucket
EventRole:
```

```
Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
       Version: 2012-10-17
       Statement:
           Effect: Allow
           Principal:
             Service:
               - events.amazonaws.com
           Action: sts:AssumeRole
     Path: /
     Policies:
         PolicyName: eb-pipeline-execution
         PolicyDocument:
           Version: 2012-10-17
           Statement:
             _
               Effect: Allow
               Action: codepipeline:StartPipelineExecution
               Resource: !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region',
':', !Ref 'AWS::AccountId', ':', !Ref AppPipeline ] ]
EventRule:
  Type: AWS::Events::Rule
  Properties:
     EventBusName: default
     EventPattern:
       source:
         - aws.s3
      detail-type:
         - Object Created
       detail:
         bucket:
           name:
             - !Ref SourceBucket
     Name: EnabledS3SourceRule
     State: ENABLED
    Targets:
       -
         Arn:
           !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region', ':', !Ref
'AWS::AccountId', ':', !Ref AppPipeline ] ]
         RoleArn: !GetAtt EventRole.Arn
```
Id: codepipeline-AppPipeline

JSON

{

```
"Parameters": {
    "SourceObjectKey": {
        "Description": "S3 source artifact",
        "Type": "String",
        "Default": "SampleApp_Linux.zip"
    },
    "ApplicationName": {
        "Description": "CodeDeploy application name",
        "Type": "String",
        "Default": "DemoApplication"
    },
    "BetaFleet": {
        "Description": "Fleet configured in CodeDeploy",
        "Type": "String",
        "Default": "DemoFleet"
    }
},
"Resources": {
    "SourceBucket": {
        "Type": "AWS::S3::Bucket",
          "Properties": {
            "NotificationConfiguration": {
                "EventBridgeConfiguration": {
                     "EventBridgeEnabled": true
                }
            },
            "VersioningConfiguration": {
                "Status": "Enabled"
            }
        }
    },
    "CodePipelineArtifactStoreBucket": {
        "Type": "AWS::S3::Bucket"
    },
    "CodePipelineArtifactStoreBucketPolicy": {
        "Type": "AWS::S3::BucketPolicy",
        "Properties": {
```

```
"Bucket": {
    "Ref": "CodePipelineArtifactStoreBucket"
},
"PolicyDocument": {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyUnEncryptedObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": {
                "Fn::Join": [
                     "",
                     Γ
                         {
                             "Fn::GetAtt": [
                                 "CodePipelineArtifactStoreBucket",
                                 "Arn"
                             ]
                         },
                         "/*"
                     ]
                ]
            },
            "Condition": {
                "StringNotEquals": {
                     "s3:x-amz-server-side-encryption": "aws:kms"
                }
            }
        },
        {
            "Sid": "DenyInsecureConnections",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": {
                "Fn::Join": [
                     "",
                     Ε
                         {
                             "Fn::GetAtt": [
                                 "CodePipelineArtifactStoreBucket",
                                 "Arn"
```

```
]
                                 },
                                 "/*"
                             ]
                         ]
                    },
                     "Condition": {
                         "Bool": {
                             "aws:SecureTransport": false
                         }
                    }
                }
            ]
        }
    }
},
"CodePipelineServiceRole": {
    "Type": "AWS:::IAM::Role",
    "Properties": {
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                     "Effect": "Allow",
                     "Principal": {
                         "Service": [
                             "codepipeline.amazonaws.com"
                         ]
                    },
                     "Action": "sts:AssumeRole"
                }
            ]
        },
        "Path": "/",
        "Policies": [
            {
                 "PolicyName": "AWS-CodePipeline-Service-3",
                 "PolicyDocument": {
                     "Version": "2012-10-17",
                     "Statement": [
                         {
                             "Effect": "Allow",
                             "Action": [
                                 "codecommit:CancelUploadArchive",
```

```
"codecommit:GetBranch",
        "codecommit:GetCommit",
        "codecommit:GetUploadArchiveStatus",
        "codecommit:UploadArchive"
    ],
    "Resource": "resource_ARN"
},
{
    "Effect": "Allow",
    "Action": [
        "codedeploy:CreateDeployment",
        "codedeploy:GetApplicationRevision",
        "codedeploy:GetDeployment",
        "codedeploy:GetDeploymentConfig",
        "codedeploy:RegisterApplicationRevision"
    ],
    "Resource": "resource_ARN"
},
{
    "Effect": "Allow",
    "Action": [
        "codebuild:BatchGetBuilds",
        "codebuild:StartBuild"
    ],
    "Resource": "resource ARN"
},
{
    "Effect": "Allow",
    "Action": [
        "devicefarm:ListProjects",
        "devicefarm:ListDevicePools",
        "devicefarm:GetRun",
        "devicefarm:GetUpload",
        "devicefarm:CreateUpload",
        "devicefarm:ScheduleRun"
    ],
    "Resource": "resource_ARN"
},
{
    "Effect": "Allow",
    "Action": [
        "lambda:InvokeFunction",
        "lambda:ListFunctions"
    ],
```

```
"Resource": "resource_ARN"
                         },
                         {
                             "Effect": "Allow",
                             "Action": [
                                 "iam:PassRole"
                             ],
                             "Resource": "resource_ARN"
                         },
                         {
                             "Effect": "Allow",
                             "Action": [
                                  "elasticbeanstalk:*",
                                  "ec2:*",
                                  "elasticloadbalancing:*",
                                  "autoscaling:*",
                                 "cloudwatch:*",
                                 "s3:*",
                                  "sns:*",
                                 "cloudformation:*",
                                  "rds:*",
                                  "sqs:*",
                                  "ecs:*"
                             ],
                             "Resource": "resource_ARN"
                         }
                     ]
                }
            }
        ]
   }
},
"AppPipeline": {
    "Type": "AWS::CodePipeline::Pipeline",
    "Properties": {
        "Name": "s3-events-pipeline",
        "RoleArn": {
            "Fn::GetAtt": [
                "CodePipelineServiceRole",
                "Arn"
            ]
        },
        "Stages": [
            {
```

```
"Name": "Source",
    "Actions": [
        {
            "Name": "SourceAction",
            "ActionTypeId": {
                "Category": "Source",
                "Owner": "AWS",
                "Version": 1,
                "Provider": "S3"
            },
            "OutputArtifacts": [
                {
                     "Name": "SourceOutput"
                }
            ],
            "Configuration": {
                "S3Bucket": {
                    "Ref": "SourceBucket"
                },
                "S3ObjectKey": {
                    "Ref": "SourceObjectKey"
                },
                "PollForSourceChanges": false
            },
            "RunOrder": 1
        }
    ]
},
{
    "Name": "Beta",
    "Actions": [
        {
            "Name": "BetaAction",
            "InputArtifacts": [
                {
                     "Name": "SourceOutput"
                }
            ],
            "ActionTypeId": {
                "Category": "Deploy",
                "Owner": "AWS",
                "Version": 1,
                "Provider": "CodeDeploy"
            },
```

```
"Configuration": {
                             "ApplicationName": {
                                 "Ref": "ApplicationName"
                             },
                             "DeploymentGroupName": {
                                 "Ref": "BetaFleet"
                             }
                         },
                         "RunOrder": 1
                    }
                ]
            }
        ],
        "ArtifactStore": {
            "Type": "S3",
            "Location": {
                "Ref": "CodePipelineArtifactStoreBucket"
            }
        }
    }
},
"EventRole": {
    "Type": "AWS::IAM::Role",
    "Properties": {
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                     "Effect": "Allow",
                     "Principal": {
                         "Service": [
                             "events.amazonaws.com"
                         ]
                    },
                     "Action": "sts:AssumeRole"
                }
            ]
        },
        "Path": "/",
        "Policies": [
            {
                "PolicyName": "eb-pipeline-execution",
                "PolicyDocument": {
                     "Version": "2012-10-17",
```

```
"Statement": [
                         {
                             "Effect": "Allow",
                             "Action": "codepipeline:StartPipelineExecution",
                             "Resource": {
                                 "Fn::Join": [
                                     "",
                                      Γ
                                          "arn:aws:codepipeline:",
                                          {
                                              "Ref": "AWS::Region"
                                          },
                                          ":",
                                          {
                                              "Ref": "AWS::AccountId"
                                          },
                                          ":",
                                          {
                                              "Ref": "AppPipeline"
                                         }
                                     ]
                                 ]
                             }
                         }
                    ]
                }
            }
        ]
    }
},
"EventRule": {
    "Type": "AWS::Events::Rule",
    "Properties": {
        "EventBusName": "default",
  "EventPattern": {
      "source": [
    "aws.s3"
 ],
        "detail-type": [
"Object Created"
  ],
  "detail": {
"bucket": {
```

```
"name": [
                            {
                            "Ref": "SourceBucket"
                            }
                        ]
                    }
             }
},
"Name": "EnabledS3SourceRule",
      "State": "ENABLED",
               "Targets": [
                   {
                       "Arn": {
                           "Fn::Join": [
                               "",
                                Ε
                                    "arn:aws:codepipeline:",
                                    {
                                        "Ref": "AWS::Region"
                                    },
                                    ":",
                                    {
                                        "Ref": "AWS::AccountId"
                                    },
                                    ":",
                                    {
                                        "Ref": "AppPipeline"
                                    }
                               ]
                           ]
                       },
                       "RoleArn": {
                           "Fn::GetAtt": [
                                "EventRole",
                               "Arn"
                           ]
                       },
                       "Id": "codepipeline-AppPipeline"
                   }
              ]
          }
      }
  }
  }
```

}

EventBridge 및를 사용하는 Amazon S3 소스 작업에 연결 AWS CloudTrail

이 섹션의 지침은 생성 및 관리해야 하는 AWS CloudTrail 리소스를 사용하는 S3 소스 작업을 생성하 는 단계를 제공합니다. 추가 AWS CloudTrail 리소스가 필요하지 않은 EventBridge에서 S3 소스 작업 을 사용하려면의 CLI 지침을 사용합니다<u>이벤트용으로 활성화된 S3 소스를 사용하여 폴링 파이프라인</u> 마이그레이션.

A Important

이 절차에서는 생성 및 관리해야 하는 AWS CloudTrail 리소스를 사용하는 S3 소스 작업을 생 성하는 단계를 제공합니다. AWS CloudTrail 리소스 없이이 작업을 생성하는 절차는 콘솔에서 사용할 수 없습니다. CLI를 사용하려면 <u>이벤트용으로 활성화된 S3 소스를 사용하여 폴링 파이</u> 프라인 마이그레이션 섹션을 참조하세요.

CodePipeline에 Amazon S3 소스 작업을 추가하려면 다음 중 하나를 선택합니다.

- CodePipeline 콘솔 파이프라인 생성 마법사(<u>사용자 지정 파이프라인 생성(콘솔)</u>) 또는 작업 편집 페 이지를 사용하여 S3 공급자 옵션을 선택합니다. 콘솔은 소스가 변경될 때 파이프라인을 시작하는 EventBridge 규칙 및 CloudTrail 추적을 생성합니다.
- AWS CLI 를 사용하여 작업에 대한 S3 작업 구성을 추가하고 다음과 같이 추가 리소스를 생성합니다.
 - <u>Amazon S3 소스 작업 참조</u>의 S3 예제 작업 구성을 사용하여 <u>파이프라인 생성(CLI)</u>과 같이 작업을 생성합니다.
 - 변경 감지 방법은 기본적으로 소스를 폴링하여 파이프라인을 시작합니다. 정기적 확인을 비 활성화하고 감지 규칙 및 추적 변경을 수동으로 생성해야 합니다. <u>Amazon S3 소스에 대한</u> <u>EventBridge 규칙 생성(콘솔)</u>, <u>Amazon S3 소스에 대한 EventBridge 규칙 생성(CLI)</u> 또는 <u>Amazon</u> <u>S3 소스에 대한 EventBridge 규칙 생성(AWS CloudFormation 템플릿)</u> 방법 중 한 가지를 선택하 세요.

AWS CloudTrail 는 Amazon S3 소스 버킷에서 이벤트를 로깅하고 필터링하는 서비스입니다. 추적 기 록은 필터링된 소스 변경 사항을 EventBridge 규칙으로 전송합니다. EventBridge 규칙은 소스 변경을 감지하고 파이프라인을 시작합니다.

요구 사항:

- 추적을 생성하지 않는 경우 기존 AWS CloudTrail 추적을 사용하여 Amazon S3 소스 버킷에 이벤트 를 로깅하고 필터링된 이벤트를 EventBridge 규칙으로 전송합니다.
- 가 로그 파일을 저장할 AWS CloudTrail 수 있는 기존 S3 버킷을 생성하거나 사용합니다. 에는 Amazon S3 버킷으로 로그 파일을 전송하는 데 필요한 권한이 AWS CloudTrail 있어야 합니다. 버킷 을 <u>요청자 지불</u> 버킷으로 구성할 수 없습니다. 콘솔에서 추적을 생성하거나 업데이트하는 과정에서 Amazon S3 버킷을 생성하면가 필요한 권한을 버킷에 AWS CloudTrail 연결합니다. 자세한 내용은 <u>CloudTrail용 Amazon S3 버킷 정책</u> 단원을 참조하십시오.

Amazon S3 소스에 대한 EventBridge 규칙 생성(콘솔)

EventBridge에서 규칙을 설정하기 전에 AWS CloudTrail 추적을 생성해야 합니다. 자세한 내용은 <u>콘솔</u>에서 추적 생성 단원을 참조하십시오.

🛕 Important

콘솔을 사용하여 파이프라인을 생성하거나 편집하면 EventBridge 규칙 및 AWS CloudTrail 추적이 생성됩니다.

추적을 생성하려면

- 1. AWS CloudTrail 콘솔을 엽니다.
- 2. 탐색 창에서 [Trails]를 선택합니다.
- 3. 추적 생성을 선택합니다. 추적 이름에 추적 이름을 입력합니다.
- [Storage location]에서 로그 파일을 저장하는 데 사용되는 버킷을 생성 또는 지정합니다. 기본적으 로 Amazon S3 버킷 및 객체는 프라이빗입니다. 리소스 소유자(버킷을 생성한 AWS 계정)만 버킷 과 해당 객체에 액세스할 수 있습니다. 버킷에는 버킷의 객체에 액세스할 수 있는 AWS CloudTrail 권한을 허용하는 리소스 정책이 있어야 합니다.
- 5. 추적 로그 버킷 및 폴더에서 Amazon S3 버킷과 객체 접두사(폴더 이름)를 지정하여 폴더의 모든 객체에 대한 데이터 이벤트를 기록합니다. 각 추적의 경우 최대 250개의 Amazon S3 객체를 추가 할 수 있습니다. 필요한 암호화 키 정보를 입력하고 다음을 선택합니다.

- 6. 이벤트 유형에서 관리 이벤트를 선택합니다.
- 7. 관리 이벤트에서 쓰기를 선택합니다. 추적은 지정된 버킷 및 접두사에서 Amazon S3 객체 수준 API 활동을 기록합니다(예: GetObject 및 PutObject).
- 8. 쓰기를 선택합니다.
- 9. 추적이 만족스러운 경우 추적 생성을 선택합니다.

Amazon S3 소스가 포함된 파이프라인을 대상으로 지정하는 EventBridge 규칙을 생성하려면

- 1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
- 탐색 창에서 규칙을 선택합니다. 기본 버스를 선택된 상태로 두거나 이벤트 버스를 선택하세요. 규 칙 생성을 선택합니다.
- 3. 이름에 역할의 이름을 입력합니다.
- 4. 규칙 유형에서 이벤트 패턴이 있는 규칙을 선택합니다. Next(다음)를 선택합니다.
- 5. 이벤트 소스에서 AWS 이벤트 또는 EventBridge 파트너 이벤트를 선택합니다.
- 6. 샘플 이벤트 유형에서 AWS 이벤트를 선택합니다.
- 7. 샘플 이벤트에서 필터링할 키워드로 S3를 입력합니다. CloudTrail을 통해AWS API 직접 호출을 선 택합니다.
- 8. 생성 방법에서 사용자 정의 패턴(JSON 편집기)을 선택합니다.

아래 제공된 이벤트 패턴을 붙여넣습니다. 버킷 이름과 버킷 내 객체를 requestParameters로 고유하게 식별하는 S3 객체 키(또는 키 이름)를 추가해야 합니다. 이 예제에서는 amzn-s3demo-source-bucket이라는 버킷과 my-files.zip의 객체 키에 대해 규칙이 생성됩니다. Edit 창을 사용하여 리소스를 지정할 때 규칙이 업데이트되어 사용자 지정 이벤트 패턴을 사용합 니다.

다음은 복사하여 붙여넣기가 가능한 이벤트 패턴 샘플입니다.

```
{
    "source": [
        "aws.s3"
],
    "detail-type": [
        "AWS API Call via CloudTrail"
],
    "detail": {
        "eventSource": [
        "aws.s3"
        "eventSource": [
        "detail": [
        "eventSource": [
        "even
```

```
"s3.amazonaws.com"
        ],
        "eventName": [
            "CopyObject",
            "CompleteMultipartUpload",
            "PutObject"
        ],
        "requestParameters": {
            "bucketName": [
                 "amzn-s3-demo-source-bucket"
            ],
            "key": [
                 "my-files.zip"
            ]
        }
    }
}
```

- 9. Next(다음)를 선택합니다.
- 10. 대상 유형에서 AWS 서비스를 선택합니다.
- 11. 대상 선택에서 CodePipeline을 선택합니다. Pipeline ARN에서 이 규칙에 의해 시작되는 파이프라 인의 파이프라인 ARN을 입력합니다.

Note

파이프라인 ARN을 확인하려면 get-pipeline 명령을 실행합니다. 출력에 파이프라인 ARN 이 나타납니다. ARN의 형식은 다음과 같습니다.

arn:aws:codepipeline:region:account:pipeline-name

파이프라인 ARN 샘플:

arn:aws:codepipeline:us-east-2:80398EXAMPLE:MyFirstPipeline

- 12. EventBridge에 EventBridge 규칙과 연결된 대상을 호출할 권한을 부여하는 IAM 서비스 역할을 생성하거나 지정하려면(이 경우에는 대상이 CodePipeline):
 - 이 특정 리소스에 대한 새 역할 생성을 선택하여 EventBridge에 파이프라인 실행을 시작할 권한 을 제공하는 서비스 역할을 생성합니다.
 - 기존 역할 사용을 선택하여 EventBridge에 파이프라인 실행을 시작할 권한을 제공하는 서비스 역할을 입력합니다.
- 13. (선택 사항) 특정 이미지 ID로 소스 재정의를 지정하려면 입력 변환기를 사용하여 데이터를 JSON 파라미터로 전달합니다.

• 추가 설정을 폅니다.

대상 입력 구성에서 입력 변환기 구성을 선택합니다.

대화 상자에서 내 이름 입력을 선택합니다. 입력 경로 상자에 다음 키-값 페어를 입력합니다.

```
{"revisionValue": "$.detail.object.version-id"}
```

• 템플릿 상자에 다음 키-값 페어를 입력합니다.



- 확인을 선택합니다.
- 14. Next(다음)를 선택합니다.
- 15. 태그 페이지에서 다음을 선택합니다.
- 16. 검토 및 생성 페이지에서 규칙 구성을 검토합니다. 규칙이 만족스러우면 규칙 생성(Create rule)을 선택하세요.

Amazon S3 소스에 대한 EventBridge 규칙 생성(CLI)

AWS CloudTrail 추적을 생성하고 로깅을 활성화하려면

AWS CLI 를 사용하여 추적을 생성하려면 다음을 지정하여 create-trail 명령을 호출합니다.

- 추적 이름입니다.
- AWS CloudTrail에 대한 버킷 정책을 이미 적용한 버킷입니다.

자세한 내용은 AWS 명령줄 인터페이스를 사용하여 추적 생성을 참조하세요.

1. create-trail 명령을 호출하고 --name 및 --s3-bucket-name 파라미터를 포함시킵니다.

이렇게 변경하는 이유는 무엇입니까? 이렇게 하면 S3 소스 버킷에 필요한 CloudTrail 추적이 생성 됩니다.

다음 명령은 --name 및 --s3-bucket-name을 사용하여 my-trail이라는 추적과 amzn-s3demo-source-bucket이라는 버킷을 생성합니다.

aws cloudtrail create-trail --name my-trail --s3-bucket-name amzn-s3-demo-source-bucket

2. start-logging 명령을 호출하고 - - name 파라미터를 포함시킵니다.

이렇게 변경하는 이유는 무엇입니까? 이 명령은 소스 버킷에 대한 CloudTrail 로깅을 시작하고, EventBridge에 이벤트를 전송합니다.

예제:

다음 명령은 --name을 사용하여 my-trail이라는 추적에서 로깅을 시작합니다.

aws cloudtrail start-logging --name my-trail

 put-event-selectors 명령을 호출하고 --trail-name 및 --event-selectors 파라미터를 포함 시킵니다. 이벤트 선택기를 사용하여 추적에서 소스 버킷에 대한 데이터 이벤트를 기록하고 이벤 트를 EventBridge 규칙으로 전송하도록 지정합니다.

이렇게 변경하는 이유는 무엇입니까? 이 명령은 이벤트를 필터링합니다.

예제:

다음 명령은 --trail-name 및 --event-selectors를 사용하여 소스 버킷 및 amzn-s3demo-source-bucket/myFolder라는 접두사에 대한 데이터 이벤트 관리를 지정합니다.

```
aws cloudtrail put-event-selectors --trail-name my-trail --event-selectors
'[{ "ReadWriteType": "WriteOnly", "IncludeManagementEvents":false,
"DataResources": [{ "Type": "AWS::S3::Object", "Values": ["arn:aws:s3:::amzn-s3-
demo-source-bucket/myFolder/file.zip"] }] }]'
```

Amazon S3가 이벤트 소스이고 CodePipeline이 대상인 EventBridge 규칙을 생성하고 권한 정책을 적 용하려면

- 1. CodePipeline을 사용하여 규칙을 호출하도록 EventBridge에 권한을 부여합니다. 자세한 내용은 Amazon EventBridge에 대한 리소스 기반 정책 사용을 참조하세요.
 - a. 다음 예제를 사용하여 EventBridge가 서비스 역할을 담당하도록 허용하는 신뢰 정책을 생성 합니다. 이름을 trustpolicyforEB.json로 지정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
               "Service": "events.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
        }
    ]
}
```

b. 다음 명령을 사용하여 Role-for-MyRule 역할을 생성한 후 신뢰 정책에 연결합니다.

이렇게 변경하는 이유는 무엇입니까? 이 신뢰 정책을 역할에 추가하면 EventBridge에 대한 권한이 생성됩니다.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
file://trustpolicyforEB.json
```

c. 이 샘플에서 보이는 것처럼 MyFirstPipeline이라는 파이프라인에 대한 권한 정책 JSON 을 생성합니다. 권한 정책 이름을 permissionspolicyforEB.json으로 지정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "codepipeline:StartPipelineExecution"
        ],
            "Resource": [
```

}] }

]

d. 다음 명령을 사용하여 앞에서 생성한 Role-for-MyRule 역할에 새로운 CodePipeline-Permissions-Policy-for-EB 권한 정책을 연결합니다.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

"arn:aws:codepipeline:us-west-2:80398EXAMPLE:MyFirstPipeline"

2. put-rule 명령을 호출하고 --name, --event-pattern 및 --role-arn 파라미터를 포함시킵니 다.

다음 샘플 명령은 MyS3SourceRule이라는 역할 별칭을 생성합니다.

```
aws events put-rule --name "MyS3SourceRule" --event-pattern "{\"source\":
[\"aws.s3\"],\"detail-type\":[\"AWS API Call via CloudTrail\"],\"detail\":
{\"eventSource\":[\"s3.amazonaws.com\"],\"eventName\":[\"CopyObject\",\"PutObject
\",\"CompleteMultipartUpload\"],\"requestParameters\":{\"bucketName\":[\"amzn-s3-
demo-source-bucket\"],\"key\":[\"my-key\"]}}
--role-arn "arn:aws:iam::ACCOUNT_ID:role/Role-for-MyRule"
```

3. CodePipeline을 대상으로 추가하려면 put-targets 명령을 호출하고 --rule 및 --targets 파라 미터를 포함합니다.

다음 명령은 MyS3SourceRule이라는 규칙에 대해 대상 Id가 숫자 1로 구성됨을 지정하며, 규칙 에 대한 대상 목록에서 대상 1로 표시됩니다. 이 명령은 또한 파이프라인에 대한 예제 ARN를 지정 합니다. 파이프라인은 리포지토리에서 변경이 발생하면 시작됩니다.

```
aws events put-targets --rule MyS3SourceRule --targets
Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
```

- 4. (선택 사항) 특정 이미지 ID에 대한 소스 재정의로 입력 변환기를 구성하려면 CLI 명령에서 다음 JSON을 사용합니다. 다음 예제에서는 다음과 같은 재정의를 구성합니다.
 - 이 예제actionNameSource에서는 소스 이벤트에서 파생되지 않은 파이프라인 생성 시 정의 된 동적 값입니다.
 - 이 예제revisionTypeS3_OBJECT_VERSION_ID에서는 소스 이벤트에서 파생되지 않은 파이 프라인 생성 시 정의된 동적 값입니다.

• 이 예제의, revisionValue<revisionValue>는 소스 이벤트 변수에서 파생됩니다.

```
{
    "Rule": "my-rule",
    "Targets": [
        {
            "Id": "MyTargetId",
            "Arn": "ARN",
            "InputTransformer": {
                "InputPathsMap": {
                     "revisionValue": "$.detail.object.version-id"
                },
                "InputTemplate": {
                     "sourceRevisions": {
                         "actionName": "Source",
                         "revisionType": "S3_OBJECT_VERSION_ID",
                         "revisionValue": "<revisionValue>"
                    }
                }
            }
        }
    ]
}
```

파이프라인의 PollForSourceChanges 파라미터를 편집하려면

A Important

이 방법으로 파이프라인을 생성할 때 명시적으로 false로 설정되지 않은 경우 PollForSourceChanges 파라미터 기본값은 true입니다. 이벤트 기반 변경 감지를 추가할 때 는 출력에 파라미터를 추가하고 false로 설정하여 폴링을 비활성화해야 합니다. 그렇지 않으면 파이프라인이 단일 소스 변경 시 두 번 시작됩니다. 세부 정보는 <u>PollForSourceChanges 파</u> 라미터의 유효한 설정을 참조하세요.

1. get-pipeline 명령을 실행하여 파이프라인 구조를 JSON 파일로 복사합니다. 예를 들어, MyFirstPipeline라는 파이프라인의 경우 다음 명령을 입력합니다. aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json

이 명령은 아무 것도 반환하지 않지만 생성한 파일이 명령을 실행한 디렉터리에 표시되어야 합니 다.

2. 일반 텍스트 편집기에서 JSON 파일을 열고 다음 예에 나와 있는 것처럼 amzn-s3-demosource-bucket 버킷의 PollForSourceChanges 파라미터를 false로 변경하여 소스 단계를 편집합니다.

이렇게 변경하는 이유는 무엇입니까? 이 파라미터를 false로 설정하면 정기적 확인이 비활성화 되어 이벤트 기반 변경 탐지만 사용할 수 있습니다.

```
"configuration": {
    "S3Bucket": "amzn-s3-demo-source-bucket",
    "PollForSourceChanges": "false",
    "S3ObjectKey": "index.zip"
},
```

 get-pipeline 명령을 사용하여 검색한 파이프라인 구조로 작업을 수행할 경우, JSON 파일에서 metadata 행을 제거해야 합니다. 이렇게 하지 않으면 update-pipeline 명령에서 사용할 수 없습니 다. "metadata": { } 행과, "created", "pipelineARN" 및 "updated" 필드를 제거합니다.

예를 들어, 구조에서 다음 행을 삭제합니다.

```
"metadata": {
    "pipelineArn": "arn:aws:codepipeline:region:account-ID:pipeline-name",
    "created": "date",
    "updated": "date"
},
```

파일을 저장합니다.

4. 변경 사항을 적용하려면 파이프라인 JSON 파일을 지정하여 update-pipeline 명령을 실행합니다.

```
▲ Important
파일 이름 앞에 file://를 포함해야 합니다. 이 명령에 필수적입니다.
```

aws codepipeline update-pipeline --cli-input-json file://pipeline.json

이 명령은 편집한 파이프라인의 전체 구조를 반환합니다.

Note

update-pipeline 명령을 실행하면 파이프라인이 중지됩니다. update-pipeline 명령을 실행 할 때 파이프라인을 통해 개정을 실행하는 중이라면 해당 실행이 중지됩니다. 업데이트 된 파이프라인을 통해 해당 개정을 실행하려면 파이프라인을 수동으로 시작해야 합니다. start-pipeline-execution 명령을 사용하여 수동으로 파이프라인을 시작합니다.

Amazon S3 소스에 대한 EventBridge 규칙 생성(AWS CloudFormation 템플 릿)

AWS CloudFormation 를 사용하여 규칙을 생성하려면 여기에 표시된 대로 템플릿을 업데이트합니다.

Amazon S3가 이벤트 소스이고 CodePipeline이 대상인 EventBridge 규칙을 생성하고 권한 정책을 적 용하려면

- 템플릿의에서 AWS::IAM::Role AWS CloudFormation 리소스를 Resources사용하여 이벤트가 파이프라인을 시작하도록 허용하는 IAM 역할을 구성합니다. 이 항목은 두 가지 정책을 사용하는 역할을 만듭니다.
 - 첫 번째 정책은 가 역할을 수임하도록 허용합니다.
 - 두 번째 정책은 파이프라인을 시작할 권한을 부여합니다.

이렇게 변경하는 이유는 무엇입니까? AWS::IAM::Role 리소스를 추가하면 AWS CloudFormation 가 EventBridge에 대한 권한을 생성할 수 있습니다. 이 리소스는 AWS CloudFormation 스택에 추가됩니다.

YAML

EventRole: Type: AWS::IAM::Role Properties:

```
AssumeRolePolicyDocument:
       Version: 2012-10-17
       Statement:
           Effect: Allow
           Principal:
             Service:
                - events.amazonaws.com
           Action: sts:AssumeRole
     Path: /
     Policies:
         PolicyName: eb-pipeline-execution
         PolicyDocument:
           Version: 2012-10-17
           Statement:
                Effect: Allow
                Action: codepipeline:StartPipelineExecution
                Resource: !Join [ '', [ 'arn:aws:codepipeline:', !Ref
'AWS::Region', ':', !Ref 'AWS::AccountId', ':', !Ref AppPipeline ] ]
• • •
```

JSON

```
"EventRole": {
  "Type": "AWS::IAM::Role",
  "Properties": {
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": [
              "events.amazonaws.com"
            ]
          },
          "Action": "sts:AssumeRole"
        }
      ]
```

```
},
      "Path": "/",
      "Policies": [
        {
          "PolicyName": "eb-pipeline-execution",
          "PolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
              {
                 "Effect": "Allow",
                 "Action": "codepipeline:StartPipelineExecution",
                 "Resource": {
                   "Fn::Join": [
                     "",
                     Ε
                       "arn:aws:codepipeline:",
                       {
                         "Ref": "AWS::Region"
                       },
                       ":",
                       {
                         "Ref": "AWS::AccountId"
                       },
                       ":",
                       {
                         "Ref": "AppPipeline"
                       }
                     ]
                   ]
. . .
```

2. AWS::Events::Rule AWS CloudFormation 리소스를 사용하여 EventBridge 규칙을 추가합니다. 이 이벤트 패턴은 Amazon S3 소스 버킷의 CopyObject, PutObject 및 CompleteMultipartUpload를 모니터링하는 이벤트를 생성합니다. 또한 파이프라인 대상도 포함하세요. 이 규칙은 CopyObject, PutObject 또는 CompleteMultipartUpload 발생 시 대상 파이프라인에서 StartPipelineExecution을 호출합니다.

이렇게 변경하는 이유는 무엇입니까? AWS::Events::Rule 리소스를 추가 AWS CloudFormation 하면가 이벤트를 생성할 수 있습니다. 이 리소스는 AWS CloudFormation 스택에 추가됩니다.

YAML

```
EventRule:
   Type: AWS::Events::Rule
   Properties:
     EventPattern:
        source:
          - aws.s3
        detail-type:
          - 'AWS API Call via CloudTrail'
        detail:
          eventSource:
            - s3.amazonaws.com
          eventName:
            - CopyObject
            - PutObject
            - CompleteMultipartUpload
          requestParameters:
            bucketName:
              - !Ref SourceBucket
            key:
              - !Ref SourceObjectKey
     Targets:
        -
          Arn:
            !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region', ':', !Ref
'AWS::AccountId', ':', !Ref AppPipeline ] ]
          RoleArn: !GetAtt EventRole.Arn
          Id: codepipeline-AppPipeline
. . .
```

JSON

```
"EventRule": {
   "Type": "AWS::Events::Rule",
   "Properties": {
    "EventPattern": {
        "source": [
            "aws.s3"
```

```
],
  "detail-type": [
    "AWS API Call via CloudTrail"
  ],
  "detail": {
    "eventSource": [
      "s3.amazonaws.com"
    ],
    "eventName": [
      "CopyObject",
      "PutObject",
      "CompleteMultipartUpload"
    ],
    "requestParameters": {
      "bucketName": [
        {
          "Ref": "SourceBucket"
        }
      ],
      "key": [
        {
          "Ref": "SourceObjectKey"
        }
      ]
    }
  }
},
"Targets": [
  {
    "Arn": {
      "Fn::Join": [
        "",
        Ε
          "arn:aws:codepipeline:",
          {
            "Ref": "AWS::Region"
          },
          ":",
          {
            "Ref": "AWS::AccountId"
          },
          ":",
          {
            "Ref": "AppPipeline"
```

```
}
               ]
             ]
           },
           "RoleArn": {
             "Fn::GetAtt": [
               "EventRole",
               "Arn"
             ]
           },
           "Id": "codepipeline-AppPipeline"
        }
      ]
    }
 }
},
. . .
```

3. 이 코드 조각을 첫 번째 템플릿에 추가하여 교차 스택 기능을 허용하세요.

YAML

```
Outputs:
SourceBucketARN:
Description: "S3 bucket ARN that Cloudtrail will use"
Value: !GetAtt SourceBucket.Arn
Export:
Name: SourceBucketARN
```

JSON

```
"Outputs" : {
    "SourceBucketARN" : {
        "Description" : "S3 bucket ARN that Cloudtrail will use",
        "Value" : { "Fn::GetAtt": ["SourceBucket", "Arn"] },
        "Export" : {
            "Name" : "SourceBucketARN"
        }
    }
....
```

- (선택 사항) 특정 이미지 ID에 대한 소스 재정의로 입력 변환기를 구성하려면 다음 YAML 코드 조 각을 사용합니다. 다음 예제에서는 다음과 같은 재정의를 구성합니다.
 - 이 예제actionNameSource에서는 소스 이벤트에서 파생되지 않은 파이프라인 생성 시 정의 된 동적 값입니다.
 - 이 예제revisionTypeS3_OBJECT_VERSION_ID에서는 소스 이벤트에서 파생되지 않은 파이 프라인 생성 시 정의된 동적 값입니다.
 - 이 예제의, revisionValue<revisionValue>는 소스 이벤트 변수에서 파생됩니다.

```
---
Rule: my-rule
Targets:
- Id: MyTargetId
Arn: pipeline-ARN
InputTransformer:
InputPathsMap:
revisionValue: "$.detail.object.version-id"
InputTemplate:
sourceRevisions:
actionName: Source
revisionType: S3_OBJECT_VERSION_ID
revisionValue: '<revisionValue>'
```

- 5. 업데이트된 템플릿을 로컬 컴퓨터에 저장하고 AWS CloudFormation 콘솔을 엽니다.
- 6. 스택을 선택한 후 현재 스택에 대한 변경 세트 만들기를 선택합니다.
- 업데이트된 템플릿을 업로드한 후 AWS CloudFormation에 나열된 변경 사항을 확인합니다. 이는 스택에 적용될 변경 사항입니다. 목록에 새로운 리소스가 표시됩니다.
- 8. 실행을 선택합니다.

파이프라인의 PollForSourceChanges 파라미터를 편집하려면

A Important

이 방법으로 파이프라인을 생성할 때 명시적으로 false로 설정되지 않은 경우 PollForSourceChanges 파라미터 기본값은 true입니다. 이벤트 기반 변경 감지를 추가할 때 는 출력에 파라미터를 추가하고 false로 설정하여 폴링을 비활성화해야 합니다. 그렇지 않으면 파이프라인이 단일 소스 변경 시 두 번 시작됩니다. 세부 정보는 <u>PollForSourceChanges 파</u> 라미터의 유효한 설정을 참조하세요.

템플릿에서 PollForSourceChanges를 false로 변경합니다. PollForSourceChanges를 파 이프라인 정의에 포함하지 않은 경우 추가하고 false로 설정하세요.

이렇게 변경하는 이유는 무엇입니까? PollForSourceChanges를 false로 변경하면 정기적 확 인이 비활성화되어 이벤트 기반 변경 탐지만 사용할 수 있습니다.

YAML

```
Name: Source
Actions:
-
Name: SourceAction
ActionTypeId:
Category: Source
Owner: AWS
Version: 1
Provider: S3
OutputArtifacts:
- Name: SourceOutput
Configuration:
S3Bucket: !Ref SourceBucket
S3ObjectKey: !Ref SourceObjectKey
PollForSourceChanges: false
RunOrder: 1
```

JSON

```
{
    "Name": "SourceAction",
    "ActionTypeId": {
        "Category": "Source",
        "Owner": "AWS",
        "Version": 1,
        "Provider": "S3"
    },
    "OutputArtifacts": [
        {
            "Name": "SourceOutput"
            "Name": "SourceOutput"
            "Name": "SourceOutput"
            "SourceOutput"
            "SourceOutput"
            "Name": "SourceOutput"
            "
            "SourceOutput"
            "SourceOutput"
            "
            "SourceOutput"
            "SourceOutput"
            "SourceOutput"
```

```
}
],
"Configuration": {
    "S3Bucket": {
        "Ref": "SourceBucket"
    },
    "S3ObjectKey": {
        "Ref": "SourceObjectKey"
    },
    "PollForSourceChanges": false
},
"RunOrder": 1
}
```

Amazon S3 파이프라인의 CloudTrail 리소스를 위한 두 번째 템플릿을 만들려면

 별도의 템플릿에서 Resources, AWS::S3::Bucket AWS::S3::BucketPolicy및 AWS::CloudTrail::Trail AWS CloudFormation 리소스를 사용하여 CloudTrail에 대한 간단 한 버킷 정의 및 추적을 제공합니다.

이렇게 변경하는 이유는 무엇입니까? 현재 계정당 5개의 추적으로 제한되어 있으므로 CloudTrail 추적을 별도로 생성하고 관리해야 합니다. (<u>의 제한 AWS CloudTrail</u> 참조) 하지만 단일 추적에 많 은 Amazon S3 버킷을 포함할 수 있으므로 추적을 한 번 생성한 다음 필요에 따라 다른 파이프라 인용 Amazon S3 버킷을 추가할 수 있습니다. 다음 내용을 두 번째 샘플 템플릿 파일에 붙여넣습 니다.

YAML

```
AWSCloudTrailBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket: !Ref AWSCloudTrailBucket
      PolicyDocument:
        Version: 2012-10-17
        Statement:
            Sid: AWSCloudTrailAclCheck
            Effect: Allow
            Principal:
              Service:
                - cloudtrail.amazonaws.com
            Action: s3:GetBucketAcl
            Resource: !GetAtt AWSCloudTrailBucket.Arn
            Sid: AWSCloudTrailWrite
            Effect: Allow
            Principal:
              Service:
                - cloudtrail.amazonaws.com
            Action: s3:PutObject
            Resource: !Join [ '', [ !GetAtt AWSCloudTrailBucket.Arn, '/
AWSLogs/', !Ref 'AWS::AccountId', '/*' ] ]
            Condition:
              StringEquals:
                s3:x-amz-acl: bucket-owner-full-control
 AWSCloudTrailBucket:
    Type: AWS::S3::Bucket
    DeletionPolicy: Retain
 AwsCloudTrail:
    DependsOn:

    AWSCloudTrailBucketPolicy

    Type: AWS::CloudTrail::Trail
    Properties:
      S3BucketName: !Ref AWSCloudTrailBucket
      EventSelectors:
          DataResources:
              Type: AWS::S3::Object
              Values:
                - !Join [ '', [ !ImportValue SourceBucketARN, '/', !Ref
 SourceObjectKey ] ]
```

```
ReadWriteType: WriteOnly
IncludeManagementEvents: false
IncludeGlobalServiceEvents: true
IsLogging: true
IsMultiRegionTrail: true
```

JSON

• • •

```
{
  "Parameters": {
    "SourceObjectKey": {
      "Description": "S3 source artifact",
      "Type": "String",
      "Default": "SampleApp_Linux.zip"
    }
  },
  "Resources": {
    "AWSCloudTrailBucket": {
      "Type": "AWS::S3::Bucket",
        "DeletionPolicy": "Retain"
    },
    "AWSCloudTrailBucketPolicy": {
      "Type": "AWS::S3::BucketPolicy",
      "Properties": {
        "Bucket": {
          "Ref": "AWSCloudTrailBucket"
        },
        "PolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Sid": "AWSCloudTrailAclCheck",
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  "cloudtrail.amazonaws.com"
                ]
              },
              "Action": "s3:GetBucketAcl",
              "Resource": {
```

```
"Fn::GetAtt": [
              "AWSCloudTrailBucket",
              "Arn"
            ]
          }
        },
        {
          "Sid": "AWSCloudTrailWrite",
          "Effect": "Allow",
          "Principal": {
            "Service": [
              "cloudtrail.amazonaws.com"
            ]
          },
          "Action": "s3:PutObject",
          "Resource": {
            "Fn::Join": [
              "",
              Ε
                {
                   "Fn::GetAtt": [
                     "AWSCloudTrailBucket",
                     "Arn"
                  ]
                },
                 "/AWSLogs/",
                 {
                  "Ref": "AWS::AccountId"
                },
                "/*"
              ]
            ]
          },
          "Condition": {
            "StringEquals": {
              "s3:x-amz-acl": "bucket-owner-full-control"
            }
          }
        }
      ]
    }
 }
},
"AwsCloudTrail": {
```

```
"DependsOn": [
        "AWSCloudTrailBucketPolicy"
      ],
      "Type": "AWS::CloudTrail::Trail",
      "Properties": {
        "S3BucketName": {
          "Ref": "AWSCloudTrailBucket"
        },
        "EventSelectors": [
          {
            "DataResources": [
              {
                "Type": "AWS::S3::Object",
                "Values": [
                  {
                     "Fn::Join": [
                       "",
                       Ε
                         {
                           "Fn::ImportValue": "SourceBucketARN"
                         },
                         "/",
                         {
                           "Ref": "SourceObjectKey"
                         }
                       ]
                     ]
                   }
                ]
              }
            ],
            "ReadWriteType": "WriteOnly",
            "IncludeManagementEvents": false
          }
        ],
        "IncludeGlobalServiceEvents": true,
        "IsLogging": true,
        "IsMultiRegionTrail": true
      }
    }
 }
}
```

•••

CodeCommit 소스 작업 및 EventBridge

CodePipeline에 CodeCommit 소스 작업을 추가하려면 다음 중 하나를 선택할 수 있습니다.

- CodePipeline 콘솔 파이프라인 생성 마법사(<u>사용자 지정 파이프라인 생성(콘솔)</u>) 또는 작업 편집 페 이지를 사용하여 CodeCommit 공급자 옵션을 선택합니다. 콘솔은 소스가 변경될 때 파이프라인을 시작하는 EventBridge 규칙을 생성합니다.
- AWS CLI 를 사용하여 작업에 대한 CodeCommit 작업 구성을 추가하고 다음과 같이 추가 리소스를 생성합니다.
 - <u>CodeCommit 소스 작업 참조</u>의 CodeCommit 예제 작업 구성을 사용하여 <u>파이프라인 생성(CLI)</u>과 같이 작업을 생성합니다.
 - 변경 감지 방법은 기본적으로 소스를 폴링하여 파이프라인을 시작합니다. 정기적 확인을 비활성 화하고 감지 규칙 변경을 수동으로 생성해야 합니다. <u>CodeCommit 소스에 대한 EventBridge 규칙</u> <u>생성(콘솔), CodeCommit 소스에 대한 EventBridge 규칙 생성(CLI)</u> 또는 <u>CodeCommit 소스에 대</u> <u>한 EventBridge 규칙 생성(AWS CloudFormation 템플릿)</u> 방법 중 한 가지를 선택하세요.

주제

- CodeCommit 소스에 대한 EventBridge 규칙 생성(콘솔)
- CodeCommit 소스에 대한 EventBridge 규칙 생성(CLI)
- CodeCommit 소스에 대한 EventBridge 규칙 생성(AWS CloudFormation 템플릿)

CodeCommit 소스에 대한 EventBridge 규칙 생성(콘솔)

A Important

콘솔을 사용하여 파이프라인을 생성하거나 편집하면 EventBridge 규칙이 생성됩니다.

CodePipeline 작업에 사용할 EventBridge 규칙을 생성하려면

- 1. <u>https://console.aws.amazon.com/events/</u>에서 Amazon EventBridge 콘솔을 엽니다.
- 탐색 창에서 규칙을 선택합니다. 기본 버스를 선택된 상태로 두거나 이벤트 버스를 선택하세요. 규 칙 생성을 선택합니다.

- 3. 이름에 역할의 이름을 입력합니다.
- 4. 규칙 유형에서 이벤트 패턴이 있는 규칙을 선택합니다. Next(다음)를 선택합니다.
- 5. 이벤트 소스에서 AWS 이벤트 또는 EventBridge 파트너 이벤트를 선택합니다.
- 6. 샘플 이벤트 유형에서 AWS 이벤트를 선택합니다.
- 7. 샘플 이벤트에서 필터링할 키워드로 CodeCommit을 입력합니다. CodeCommit 리포지토리 상태 변경을 선택합니다.
- 8. 생성 방법에서 사용자 정의 패턴(JSON 편집기)을 선택합니다.

아래 제공된 이벤트 패턴을 붙여넣습니다. 다음은 이름이 main인 브랜치가 있는 MyTestRepo 리 포지토리에 대한 이벤트 창의 샘플 CodeCommit 이벤트 패턴입니다.

```
{
  "source": [
    "aws.codecommit"
  ],
  "detail-type": [
    "CodeCommit Repository State Change"
  ],
  "resources": [
    "arn:aws:codecommit:us-west-2:80398EXAMPLE:MyTestRepo"
  ],
  "detail": {
    "referenceType": [
      "branch"
    ],
    "referenceName": [
      "main"
    ]
  }
}
```

- 9. 대상 영역에서 CodePipeline을 선택합니다.
- 10. 이 규칙에 의해 시작되는 파이프라인의 파이프라인 ARN을 입력합니다.

Note

get-pipeline 명령을 실행한 후 메타데이터 출력에서 파이프라인 ARN을 찾을 수 있습니다. 파이프라인 ARN은 다음 형식으로 구성됩니다.

arn:aws:codepipeline:region:account:pipeline-name

파이프라인 ARN 샘플:

arn:aws:codepipeline:us-east-2:80398EXAMPLE:MyFirstPipeline

- 11. EventBridge에 EventBridge 규칙과 연결된 대상을 호출할 권한을 부여하는 IAM 서비스 역할을 생성하거나 지정하려면(이 경우에는 대상이 CodePipeline):
 - 이 특정 리소스에 대한 새 역할 생성을 선택하여 EventBridge에 파이프라인 실행을 시작할 권한 을 제공하는 서비스 역할을 생성합니다.
 - 기존 역할 사용을 선택하여 EventBridge에 파이프라인 실행을 시작할 권한을 제공하는 서비스 역할을 입력합니다.
- 12. (선택 사항) 특정 이미지 ID로 소스 재정의를 지정하려면 입력 변환기를 사용하여 데이터를 JSON 파라미터로 전달합니다.
 - 추가 설정을 폅니다.

대상 입력 구성에서 입력 변환기 구성을 선택합니다.

대화 창에서 내 이름 입력을 선택합니다. 입력 경로 상자에 다음 키-값 페어를 입력합니다.

{"revisionValue": "\$.detail.image-digest",
"branchName": "\$.detail.referenceName"}

• 템플릿 상자에 다음 키-값 페어를 입력합니다.

- 확인을 선택합니다.
- 13. Next(다음)를 선택합니다.

- 14. 태그 페이지에서 다음을 선택합니다.
- 15. 검토 및 생성 페이지에서 규칙 구성을 검토합니다. 규칙이 만족스러우면 규칙 생성(Create rule)을 선택하세요.

CodeCommit 소스에 대한 EventBridge 규칙 생성(CLI)

put-rule 명령을 호출해 다음을 지정합니다.

- 만들려는 규칙을 고유하게 식별하는 이름. 이 이름은 AWS 계정과 연결된 CodePipeline으로 생성하는 모든 파이프라인에서 고유해야 합니다.
- 소스의 이벤트 패턴 및 규칙에서 사용하는 세부 정보 필드. 자세한 내용은 <u>Amazon EventBridge 및</u> <u>이벤트 패턴</u>을 참조하세요.

CodeCommit이 이벤트 소스이고 CodePipeline이 대상인 EventBridge 규칙을 만들려면

- CodePipeline을 사용하여 규칙을 호출하도록 EventBridge에 권한을 추가합니다. 자세한 내용은 Amazon EventBridge에 대한 리소스 기반 정책 사용을 참조하세요.
 - a. 다음 예제를 사용하여 EventBridge가 서비스 역할을 담당하도록 허용하는 신뢰 정책을 생성 합니다. 신뢰 정책 이름을 trustpolicyforEB.json으로 지정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
               "Service": "events.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
        }
   ]
}
```

b. 다음 명령을 사용하여 Role-for-MyRule 역할을 생성한 후 신뢰 정책에 연결합니다.

aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
file://trustpolicyforEB.json
c. 이 샘플에서 보이는 것처럼 MyFirstPipeline이라는 파이프라인에 대한 권한 정책 JSON 을 생성합니다. 권한 정책 이름을 permissionspolicyforEB.json으로 지정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "codepipeline:StartPipelineExecution"
        ],
            "Resource": [
               "arn:aws:codepipeline:us-west-2:80398EXAMPLE:MyFirstPipeline"
        ]
        }
    ]
}
```

d. 다음 명령을 사용하여 CodePipeline-Permissions-Policy-for-EB 권한 정책을 Role-for-MyRule 역할에 연결합니다.

이렇게 변경하는 이유는 무엇입니까? 이 정책을 역할에 추가하면 EventBridge에 대한 권한이 생성됩니다.

aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json

2. put-rule 명령을 호출하고 --name, --event-pattern 및 --role-arn 파라미터를 포함시킵니 다.

이렇게 변경하는 이유는 무엇입니까? 이 명령은 AWS CloudFormation 에서 이벤트를 생성할 수 있게 합니다.

다음 샘플 명령은 MyCodeCommitRepoRule이라는 역할 별칭을 생성합니다.

```
aws events put-rule --name "MyCodeCommitRepoRule" --event-pattern "{\"source\":
[\"aws.codecommit\"],\"detail-type\":[\"CodeCommit Repository State Change\"],
\"resources\":[\"repository-ARN\"],\"detail\":{\"referenceType\":[\"branch\"],
\"referenceName\":[\"main\"]}}" --role-arn "arn:aws:iam::ACCOUNT_ID:role/Role-for-
MyRule"
```

3. CodePipeline을 대상으로 추가하려면 put-targets 명령을 호출하고 다음 파라미터를 포함합니다.

- --rule 파라미터는 put-rule을 사용하여 생성한 rule_name에 사용됩니다.
- --targets 파라미터는 대상 목록에 있는 대상의 목록 Id 및 대상 파이프라인의 ARN에 사용됩니다.

다음 예제 명령은 MyCodeCommitRepoRule이라는 규칙에 대해 대상 Id가 숫자 1로 구성됨을 지 정하며, 규칙의 대상 목록에서 1로 대상 1로 표시됩니다. 이 예제 명령은 또한 파이프라인에 대한 예제 ARN를 지정합니다. 파이프라인은 리포지토리에서 변경이 발생하면 시작됩니다.

```
aws events put-targets --rule MyCodeCommitRepoRule --targets
Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
```

- 4. (선택 사항) 특정 이미지 ID에 대한 소스 재정의로 입력 변환기를 구성하려면 CLI 명령에서 다음 JSON을 사용합니다. 다음 예제에서는 다음과 같은 재정의를 구성합니다.
 - 이 예제actionNameSource에서는 소스 이벤트에서 파생되지 않은 파이프라인 생성 시 정의 된 동적 값입니다.
 - 이 예제revisionTypeCOMMIT_ID에서는 소스 이벤트에서 파생되지 않고 파이프라인 생성 시 정의된 동적 값입니다.
 - 이 예제의, revisionValue<revisionValue>는 소스 이벤트 변수에서 파생됩니다.

```
{
    "Rule": "my-rule",
    "Targets": [
        {
            "Id": "MyTargetId",
            "Arn": "pipeline-ARN",
            "InputTransformer": {
                "sourceRevisions": {
                     "actionName": "Source",
                     "revisionType": "COMMIT_ID",
                     "revisionValue": "<revisionValue>"
                },
                "variables": [
                    {
                         "name": "Branch_Name",
                         "value": "value"
                     }
                1
```

} } }

파이프라인의 PollForSourceChanges 파라미터를 편집하려면

A Important

이 방법으로 파이프라인을 생성할 때 명시적으로 false로 설정되지 않은 경우 PollForSourceChanges 파라미터 기본값은 true입니다. 이벤트 기반 변경 감지를 추가할 때 는 출력에 파라미터를 추가하고 false로 설정하여 폴링을 비활성화해야 합니다. 그렇지 않으면 파이프라인이 단일 소스 변경 시 두 번 시작됩니다. 세부 정보는 <u>PollForSourceChanges 파</u> 라미터의 유효한 설정을 참조하세요.

1. get-pipeline 명령을 실행하여 파이프라인 구조를 JSON 파일로 복사합니다. 예를 들어, MyFirstPipeline라는 파이프라인의 경우 다음 명령을 입력합니다.

aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json

이 명령은 아무 것도 반환하지 않지만 생성한 파일이 명령을 실행한 디렉터리에 표시되어야 합니 다.

2. 일반 텍스트 편집기에서 JSON 파일을 열고 이 예제에 나와 있는 것처럼 PollForSourceChanges 파라미터를 false로 변경하여 소스 단계를 편집합니다.

이렇게 변경하는 이유는 무엇입니까? 이 파라미터를 false로 변경하면 정기적 확인이 비활성화 되어 이벤트 기반 변경 탐지만 사용할 수 있습니다.

```
"configuration": {
    "PollForSourceChanges": "false",
    "BranchName": "main",
    "RepositoryName": "MyTestRepo"
},
```

 get-pipeline 명령을 사용하여 검색한 파이프라인 구조로 작업을 수행할 경우, JSON 파일에서 metadata 행을 제거하십시오. 이렇게 하지 않으면 update-pipeline 명령에서 사용할 수 없습니다. "metadata": { } 행과, "created", "pipelineARN" 및 "updated" 필드를 제거합니다. 예를 들어, 구조에서 다음 행을 삭제합니다.

```
"metadata": {
    "pipelineArn": "arn:aws:codepipeline:region:account-ID:pipeline-name",
    "created": "date",
    "updated": "date"
},
```

파일을 저장합니다.

4. 변경 사항을 적용하려면 파이프라인 JSON 파일을 지정하여 update-pipeline 명령을 실행합니다.

▲ Important

파일 이름 앞에 file://를 포함해야 합니다. 이 명령에 필수적입니다.

aws codepipeline update-pipeline --cli-input-json file://pipeline.json

이 명령은 편집한 파이프라인의 전체 구조를 반환합니다.

Note

update-pipeline 명령을 실행하면 파이프라인이 중지됩니다. update-pipeline 명령을 실행 할 때 파이프라인을 통해 개정을 실행하는 중이라면 해당 실행이 중지됩니다. 업데이트 된 파이프라인을 통해 해당 개정을 실행하려면 파이프라인을 수동으로 시작해야 합니다. start-pipeline-execution 명령을 사용하여 수동으로 파이프라인을 시작합니다.

CodeCommit 소스에 대한 EventBridge 규칙 생성(AWS CloudFormation 템 플릿)

AWS CloudFormation 를 사용하여 규칙을 생성하려면 여기에 표시된 대로 템플릿을 업데이트합니다.

파이프라인 AWS CloudFormation 템플릿을 업데이트하고 EventBridge 규칙을 생성하려면

- 템플릿의에서 AWS::IAM::Role AWS CloudFormation 리소스를 Resources사용하여 이벤트가 파이프라인을 시작하도록 허용하는 IAM 역할을 구성합니다. 이 항목은 두 가지 정책을 사용하는 역할을 만듭니다.
 - 첫 번째 정책은 가 역할을 수임하도록 허용합니다.
 - 두 번째 정책은 파이프라인을 시작할 권한을 부여합니다.

이렇게 변경하는 이유는 무엇입니까? AWS : : IAM : : Role 리소스를 추가 AWS CloudFormation 하 면가 EventBridge에 대한 권한을 생성할 수 있습니다. 이 리소스는 AWS CloudFormation 스택에 추가됩니다.

YAML

```
EventRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: 2012-10-17
      Statement:
           Effect: Allow
           Principal:
             Service:
               - events.amazonaws.com
           Action: sts:AssumeRole
    Path: /
    Policies:
         PolicyName: eb-pipeline-execution
         PolicyDocument:
           Version: 2012-10-17
           Statement:
               Effect: Allow
               Action: codepipeline:StartPipelineExecution
               Resource: !Join [ '', [ 'arn:aws:codepipeline:', !Ref
'AWS::Region', ':', !Ref 'AWS::AccountId', ':', !Ref AppPipeline ] ]
```

JSON

```
"EventRole": {
  "Type": "AWS:::IAM::Role",
  "Properties": {
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": [
              "events.amazonaws.com"
            ]
          },
          "Action": "sts:AssumeRole"
        }
      ]
    },
    "Path": "/",
    "Policies": [
      {
        "PolicyName": "eb-pipeline-execution",
        "PolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Effect": "Allow",
              "Action": "codepipeline:StartPipelineExecution",
              "Resource": {
                "Fn::Join": [
                  "",
                  Ε
                     "arn:aws:codepipeline:",
                     {
                      "Ref": "AWS::Region"
                    },
                     ":",
                     {
                      "Ref": "AWS::AccountId"
                    },
                     ":",
                     ſ
```

```
"Ref": "AppPipeline"
}
]
```

2. 템플릿의에서 AWS::Events::Rule AWS CloudFormation 리소스를 Resources사용하여 EventBridge 규칙을 추가합니다. 이 이벤트 패턴은 리포지토리에 대한 푸시 변경 사항을 모니터링 하는 이벤트를 생성합니다. EventBridge가 리포지토리 상태 변경을 감지하면 해당 규칙이 대상 파 이프라인에서 StartPipelineExecution을 호출합니다.

이렇게 변경하는 이유는 무엇입니까? AWS::Events::Rule 리소스를 추가 AWS CloudFormation 하면가 이벤트를 생성할 수 있습니다. 이 리소스는 AWS CloudFormation 스택에 추가됩니다.

YAML

```
EventRule:
  Type: AWS::Events::Rule
  Properties:
     EventPattern:
       source:
         - aws.codecommit
       detail-type:
         - 'CodeCommit Repository State Change'
       resources:
         - !Join [ '', [ 'arn:aws:codecommit:', !Ref 'AWS::Region', ':', !Ref
'AWS::AccountId', ':', !Ref RepositoryName ] ]
       detail:
         event:
           - referenceCreated
           - referenceUpdated
         referenceType:
           - branch
         referenceName:
           - main
    Targets:
         Arn:
           !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region', ':', !Ref
'AWS::AccountId', ':', !Ref AppPipeline ] ]
         RoleArn: !GetAtt EventRole.Arn
```

Id: codepipeline-AppPipeline

JSON

```
"EventRule": {
  "Type": "AWS::Events::Rule",
  "Properties": {
    "EventPattern": {
      "source": [
        "aws.codecommit"
      ],
      "detail-type": [
        "CodeCommit Repository State Change"
      ],
      "resources": [
        {
          "Fn::Join": [
            "",
            Ε
              "arn:aws:codecommit:",
              {
                "Ref": "AWS::Region"
              },
              ":",
              {
                "Ref": "AWS::AccountId"
              },
              ":",
              {
                "Ref": "RepositoryName"
              }
            ]
          ]
        }
      ],
      "detail": {
        "event": [
          "referenceCreated",
          "referenceUpdated"
        ],
        "referenceType": [
          "branch"
```

```
],
        "referenceName": [
          "main"
        ]
      }
    },
    "Targets": [
      {
        "Arn": {
          "Fn::Join": [
            "",
            Ε
              "arn:aws:codepipeline:",
              {
                "Ref": "AWS::Region"
              },
              ":",
               {
                "Ref": "AWS::AccountId"
              },
              ":",
              {
                "Ref": "AppPipeline"
              }
            ]
          ]
        },
        "RoleArn": {
          "Fn::GetAtt": [
            "EventRole",
            "Arn"
          ]
        },
        "Id": "codepipeline-AppPipeline"
      }
    ]
  }
},
```

 (선택 사항) 특정 이미지 ID에 대한 소스 재정의로 입력 변환기를 구성하려면 다음 YAML 코드 조 각을 사용합니다. 다음 예제에서는 다음과 같은 재정의를 구성합니다.

- 이 예제actionNameSource에서는 소스 이벤트에서 파생되지 않은 파이프라인 생성 시 정의 된 동적 값입니다.
- 이 예제revisionTypeCOMMIT_ID에서는 소스 이벤트에서 파생되지 않은 파이프라인 생성 시 정의된 동적 값입니다.
- 이 예제의, revisionValue<revisionValue>는 소스 이벤트 변수에서 파생됩니다.
- BranchName 및에 대한 출력 변수가 지정Value됩니다.

```
Rule: my-rule
Targets:
- Id: MyTargetId
Arn: pipeline-ARN
InputTransformer:
   sourceRevisions:
    actionName: Source
    revisionType: COMMIT_ID
    revisionValue: <revisionValue>
   variables:
    - name: BranchName
   value: value
```

- 4. 업데이트된 템플릿을 로컬 컴퓨터에 저장한 다음 AWS CloudFormation 콘솔을 엽니다.
- 5. 스택을 선택한 후 현재 스택에 대한 변경 세트 만들기를 선택합니다.
- 템플릿을 업로드한 후 AWS CloudFormation에 나열된 변경 사항을 확인합니다. 이는 스택에 적용 될 변경 사항입니다. 목록에 새로운 리소스가 표시됩니다.
- 7. 실행을 선택합니다.

파이프라인의 PollForSourceChanges 파라미터를 편집하려면

A Important

많은 경우 파이프라인을 생성할 때 PollForSourceChanges 파라미터 기본값은 true입니다. 이벤트 기반 변경 감지를 추가할 때는 출력에 파라미터를 추가하고 false로 설정하여 폴링을 비 활성화해야 합니다. 그렇지 않으면 파이프라인이 단일 소스 변경 시 두 번 시작됩니다. 세부 정 보는 PollForSourceChanges 파라미터의 유효한 설정을 참조하세요. 템플릿에서 PollForSourceChanges를 false로 변경합니다. PollForSourceChanges를 파 이프라인 정의에 포함하지 않은 경우 추가하고 false로 설정하세요.

이렇게 변경하는 이유는 무엇입니까? 이 파라미터를 false로 변경하면 정기적 확인이 비활성화 되어 이벤트 기반 변경 탐지만 사용할 수 있습니다.

YAML

Name: Source
Actions:
-
Name: SourceAction
ActionTypeId:
Category: Source
Owner: AWS
Version: 1
Provider: CodeCommit
OutputArtifacts:
- Name: SourceOutput
Configuration:
BranchName: !Ref BranchName
RepositoryName: !Ref RepositoryName
PollForSourceChanges: false
RunOrder: 1

JSON

```
{
    "Name": "Source",
    "Actions": [
      {
        "Name": "SourceAction",
        "ActionTypeId": {
            "Category": "Source",
            "Owner": "AWS",
            "Version": 1,
            "Provider": "CodeCommit"
        },
        "OutputArtifacts": [
        {
            "Name": "SourceOutput"
        }
}
```

```
],
   "Configuration": {
      "BranchName": {
        "Ref": "BranchName"
      },
      "RepositoryName": {
        "Ref": "RepositoryName"
      },
      "PollForSourceChanges": false
      },
      "RunOrder": 1
      }
  ]
},
```

CodeConnections를 사용하여 파이프라인에 타사 소스 공급 자 추가

AWS CodePipeline 콘솔 또는를 사용하여 파이프라인 AWS CLI 을 타사 리포지토리에 연결할 수 있습니다.

Note

콘솔을 사용하여 파이프라인을 생성하거나 편집하면 변경 감지 리소스가 자동으로 생성됩니 다. AWS CLI를 사용하여 파이프라인을 생성하는 경우 추가 리소스를 직접 생성해야 합니다. 자세한 내용은 <u>CodeCommit 소스 작업 및 EventBridge</u> 단원을 참조하십시오.

주제

- <u>Bitbucket Cloud 연결</u>
- <u>GitHub 연결</u>
- <u>GitHub Enterprise Server 연결</u>
- <u>GitLab.com 연결</u>
- GitLab 자체 관리형을 위한 연결
- 다른 계정과 공유된 연결 사용

Bitbucket Cloud 연결

연결을 사용하면 타사 공급자를 AWS 리소스와 연결하는 구성을 승인하고 설정할 수 있습니다. 타사 리포지토리를 파이프라인의 소스로 연결하려면 연결을 사용합니다.

Note

계정에서 기존 연결을 생성하거나 사용하는 대신 다른 계정 간의 공유 연결을 사용할 수 있습 니다 AWS 계정. 다른 계정과 공유된 연결 사용을 참조하세요.

이 기능은 아시아 태평양(홍콩), 아시아 태평양(하이데라바드), 아시아 태평양(자카르타), 아시아 태평양(멜버른), 아시아 태평양(오사카), 아프리카(케이프타운), 중동(바레인), 중동 (UAE), 유럽(스페인), 유럽(취리히), 이스라엘(텔아비브) 또는 AWS GovCloud(미국 서부) 리전에서는 사용할 수 없습니다. 사용 가능한 다른 작업을 참조하려면 <u>CodePipeline과 제</u> <u>품 및 서비스 통합</u>을 참조하세요. 유럽(밀라노) 리전에서 이 조치를 고려할 경우 <u>Bitbucket</u> <u>Cloud, GitHub, GitHub Enterprise Server, GitLab.com 및 GitLab 자체 관리형 작업용</u> <u>CodeStarSourceConnection</u>의 참고 사항을 참조하세요.

CodePipeline에 Bitbucket Cloud 소스 작업을 추가하려면 다음 중 하나를 선택할 수 있습니다.

 CodePipeline 콘솔 파이프라인 생성 마법사 또는 작업 편집 페이지를 사용하여 Bitbucket 공급자 옵 션을 선택합니다. 작업을 추가하려면 <u>Bitbucket Cloud에 대한 연결 생성(콘솔)</u>을 참조하세요. 콘솔을 사용하면 연결 리소스를 만들 수 있습니다.

Note

Bitbucket Cloud 리포지토리에 대한 연결을 생성할 수 있습니다. Bitbucket Server와 같은 설 치된 Bitbucket 공급자 유형은 지원되지 않습니다.

- 다음과 같이 CLI를 사용하여 Bitbucket 공급자와 함께 CreateSourceConnection 작업에 대한 작업 구성을 추가합니다.
 - 연결 리소스를 생성하려면 <u>Bitbucket Cloud에 대한 연결 생성(CLI)</u>을 참조하여 CLI를 사용하여 연 결 리소스를 생성합니다.
 - <u>Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com 및 GitLab 자체 관리형 작업용</u> <u>CodeStarSourceConnection</u>의 CreateSourceConnection 예제 작업 구성을 사용하여 <u>파이프</u> <u>라인 생성(CLI)</u>과 같이 작업을 추가합니다.

Note

설정의 개발자 도구 콘솔을 사용하여 연결을 생성할 수도 있습니다. <u>연결 생성</u>을 참조하세요.

시작하기 전:

Bitbucket Cloud 연결

- Bitbucket Cloud 같은 타사 리포지토리 공급자를 사용하여 계정을 생성해야 합니다.
- Bitbucket Cloud 리포지토리 같은 타사 코드 리포지토리를 미리 생성해야 합니다.

Bitbucket Cloud 연결은 연결을 만드는 데 사용된 Bitbucket Cloud 계정이 소유한 리포지토리에 대한 액세스 권한만 제공합니다. Bitbucket Cloud WorkSpace에 애플리케이션을 설치하는 경우 관리자 WorkSpace 권한이 필 요합니다. 그렇지 않은 경우 앱 설치 옵션이 표시되지 않습니다.

주제

- <u>Bitbucket Cloud에 대한 연결 생성(콘솔)</u>
- Bitbucket Cloud에 대한 연결 생성(CLI)

Bitbucket Cloud에 대한 연결 생성(콘솔)

다음 단계를 사용하여 CodePipeline 콘솔을 사용하여 Bitbucket 리포지토리에 연결 작업을 추가할 수 있습니다.

Note

Bitbucket Cloud 리포지토리에 대한 연결을 생성할 수 있습니다. Bitbucket Server와 같은 설치 된 Bitbucket 공급자 유형은 지원되지 않습니다.

1단계: 파이프라인 생성 또는 편집

파이프라인을 생성 또는 편집하려면

- 1. CodePipeline 콘솔에 로그인합니다.
- 2. 다음 중 하나를 선택합니다.
 - 파이프라인을 생성하려면 선택합니다. 파이프라인 생성의 단계에 따라 첫 화면을 완료하고 다 음을 선택합니다. 소스 페이지의 소스 공급자에서 Bitbucket을 선택합니다.

- 기존 파이프라인을 편집하려면 선택합니다. 편집을 선택하고 단계 편집을 선택합니다. 소스 작 업을 추가 또는 편집하려면 선택합니다. 작업 편집 페이지의 작업 이름에 작업 이름을 입력합니다. 작업 공급자에서 Bitbucket을 선택합니다.
- 3. 다음 중 하나를 수행합니다.
 - 연결에서 공급자와의 연결을 아직 생성하지 않은 경우 Bitbucket에 연결을 선택합니다. 2단계: Bitbucket에 대한 연결 생성으로 이동합니다.
 - 연결에서 공급자와의 연결을 이미 생성한 경우 연결을 선택합니다. 3단계: 연결에 대한 소스 작 업 저장으로 이동합니다.

2단계: Bitbucket Cloud에 대한 연결 생성

Bitbucket Cloud에 대한 연결을 생성하려면

1. Bitbucket에 연결 설정 페이지에서 연결 이름을 입력하고 Bitbucket에 연결을 선택합니다.

Create Bitbucket connection	
Connection name	
	Connect to Bitbucket

Bitbucket 앱 필드가 나타납니다.

2. [Bitbucket 앱(Bitbucket apps)]에서 앱 설치를 선택하거나 [새 앱 설치(Install a new app)]을 선택하 여 앱을 새로 만듭니다.



onnect to Bitbucket		
Bitbucket connection settings	Info	
Connection name		
a-connection		
Bitbucket apps Bitbucket apps create a link for your connection	with Bitbucket. To start, install a new app ar	nd save this connection.
Q	or Install a ne	ew app
		Connect

- 3. Bitbucket Cloud의 로그인 페이지가 표시되면 자격 증명으로 로그인한 다음 계속하도록 선택합니다.
- 4. 앱 설치 페이지에서 AWS CodeStar 앱이 Bitbucket 계정에 연결을 시도하고 있다는 메시지가 표시 됩니다.

Bitbucket WorkSpace를 사용하는 경우 Authorization for(권한 부여 대상) 옵션을 WorkSpace로 변 경합니다. 관리자 권한이 있는 WorkSpace만 표시됩니다.

액세스 권한 부여를 선택합니다.

5. [Bitbucket 앱(Bitbucket apps)]을 선택하면 새 설치의 연결 ID가 표시됩니다. 연결을 선택합니다. 생성된 연결이 연결 목록에 표시됩니다.

onnection name MyConnection itbucket apps tbucket apps create a link for your connection with Bitbucket. To start, install a new app and save this connection. Q ari:cloud:bitbucket::app/{c26d1f3 - X or Install a new app	Bitbucket connection setting	S Info				
MyConnection itbucket apps tbucket apps create a link for your connection with Bitbucket. To start, install a new app and save this connection. Q ari:cloud:bitbucket::app/{c26d1f3 - X or Install a new app	Connection name					
itbucket apps tbucket apps create a link for your connection with Bitbucket. To start, install a new app and save this connection. Q ari:cloud:bitbucket::app/{c26d1f3 - X or Install a new app	MyConnection					
Q ari:cloud:bitbucket::app/{c26d1f3 - X or Install a new app	fitbucket apps litbucket apps create a link for your connect	ion with Bitbucket. To	o start, in	stall a new app and save th	is connection.	
	Q ari:cloud:bitbucket::app/{c26d1f3	3 - >	< or	Install a new app		

3단계: Bitbucket Cloud 소스 작업 저장

마법사 또는 작업 편집 페이지에서 다음 단계를 사용하여 소스 작업을 연결 정보와 함께 저장합니다.

연결을 통해 소스 작업을 완료하고 저장하려면

- 1. 리포지토리 이름에서 타사 리포지토리의 이름을 선택합니다.
- 작업이 CodeConnections 작업인 경우 파이프라인 트리거에서 트리거를 추가할 수 있습니다. 파이 프라인 트리거 구성을 구성하고 선택적으로 트리거로 필터링하려면 <u>코드 푸시 또는 풀 요청 이벤</u> 트 유형을 사용하여 트리거 추가에서 자세한 내용을 참조하세요.
- 3. Output artifact format(출력 아티팩트 형식)에서 아티팩트의 형식을 선택해야 합니다.
 - 기본 방법을 사용하여 Bitbucket Cloud 작업의 출력 아티팩트를 저장하려면 CodePipeline default(CodePipeline 기본 방법)를 선택합니다. 그러면 Bitbucket Cloud 리포지토리의 파일에 액세스하여 파이프라인 아티팩트 스토어에 ZIP 파일로 아티팩트가 저장됩니다.
 - 다운스트림 작업이 Git 명령을 직접 수행할 수 있도록 리포지토리에 대한 URL 참조가 포함된 JSON 파일을 저장하려면 Full clone(전체 복제)을 선택합니다. 이 옵션은 CodeBuild 다운스트림 작업에서만 사용할 수 있습니다.

이 옵션을 선택하면 <u>Bitbucket, GitHub, GitHub Enterprise Server 또는 GitLab.com 연결을 위한</u> <u>CodeBuild GitClone 권한 추가</u>과 같이 CodeBuild 프로젝트 서비스 역할에 대한 권한을 업데이 트해야 합니다.

4. 마법사에서 다음을 선택하거나 작업 편집 페이지에서 저장을 선택합니다.

Bitbucket Cloud에 대한 연결 생성(CLI)

AWS Command Line Interface (AWS CLI)를 사용하여 연결을 생성할 수 있습니다.

Note

Bitbucket Cloud 리포지토리에 대한 연결을 생성할 수 있습니다. Bitbucket Server와 같은 설치 된 Bitbucket 공급자 유형은 지원되지 않습니다.

이렇게 하려면 create-connection 명령을 사용합니다.

▲ Important

AWS CLI 또는를 통해 생성된 연결 AWS CloudFormation 은 기본적으로 PENDING 상태입니 다. CLI 또는 와의 연결을 생성한 후 콘솔을 AWS CloudFormation사용하여 연결을 편집하여 상태를 로 설정합니다AVAILABLE.

연결을 생성하는 방법

 터미널(Linux, macOS, Unix) 또는 명령 프롬프트(Windows)를 엽니다. AWS CLI 를 사용하여 연결을 --connection-name 위해 --provider-type 및를 지정하여 create-connection 명령을 실행합니다. 이 예제에서 타사 공급자 이름은 Bitbucket이고 지정된 연결 이름은 MyConnection입니다.

aws codestar-connections create-connection --provider-type Bitbucket --connectionname MyConnection

이 명령이 제대로 실행되면 다음과 비슷한 연결 ARN 정보가 반환됩니다.

{ "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/ aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f" }

- 2. 콘솔을 사용하여 연결을 완료합니다. 자세한 내용은 <u>보류 중인 연결 업데이트</u>를 참조하세요.
- 파이프라인은 기본적으로 연결 소스 리포지토리로 코드를 푸시할 때 변경 사항을 감지합니다. 수 동 릴리스 또는 Git 태그에 대한 파이프라인 트리거 구성을 구성하려면 다음 중 하나를 수행합니다.
 - 수동 릴리스로만 시작하도록 파이프라인 트리거 구성을 구성하려면 구성에 다음 줄을 추가하세
 요.

"DetectChanges": "false",

 트리거로 필터링하도록 파이프라인 트리거 구성을 구성하려면 <u>코드 푸시 또는 풀 요청 이벤</u> <u>트 유형을 사용하여 트리거 추가</u>에서 자세한 내용을 참조하세요. 예를 들어, 다음은 파이프 라인 JSON 정의의 파이프라인 수준에 Git 태그를 추가합니다. 이 예제에서, release-v0 및 release-v1은 포함할 Git 태그이고 release-v2는 제외할 Git 태그입니다.



GitHub 연결

연결을 사용하여 타사 공급자를 AWS 리소스와 연결하는 구성을 승인하고 설정합니다.

Note

계정에서 기존 연결을 생성하거나 사용하는 대신 다른 계정 간의 공유 연결을 사용할 수 있습니다 AWS 계정. 다른 계정과 공유된 연결 사용을 참조하세요.

Note

이 기능은 아시아 태평양(홍콩), 아시아 태평양(하이데라바드), 아시아 태평양(자카르타), 아시아 태평양(멜버른), 아시아 태평양(오사카), 아프리카(케이프타운), 중동(바레인), 중동 (UAE), 유럽(스페인), 유럽(취리히), 이스라엘(텔아비브) 또는 AWS GovCloud(미국 서부) 리전에서는 사용할 수 없습니다. 사용 가능한 다른 작업을 참조하려면 <u>CodePipeline과 제</u> <u>품 및 서비스 통합</u>을 참조하세요. 유럽(밀라노) 리전에서 이 조치를 고려할 경우 <u>Bitbucket</u> <u>Cloud, GitHub, GitHub Enterprise Server, GitLab.com 및 GitLab 자체 관리형 작업용</u> CodeStarSourceConnection의 참고 사항을 참조하세요.

CodePipeline에서 GitHub 또는 GitHub Enterprise Cloud 리포지토리의 소스 작업을 추가하려면 다음 중 하나를 선택할 수 있습니다.

CodePipeline 콘솔 파이프라인 생성 마법사 또는 작업 편집 페이지를 사용하여 GitHub(GitHub 앱 사용) 공급자 옵션을 선택합니다. 작업을 추가하려면 <u>GitHub Enterprise Server에 대한 연결 생성(콘</u>솔)을 참조하세요. 콘솔을 사용하면 연결 리소스를 만들 수 있습니다.

Note

GitHub 연결을 추가하고 파이프라인에서 전체 복제 옵션을 사용하여 메타데이터를 복제하 는 방법을 안내하는 자습서는 <u>자습서: GitHub 파이프라인 소스와 함께 전체 복제 사용</u> 섹션 을 참조하세요.

• <u>파이프라인 생성(CLI)</u>에 표시된 CLI 단계를 통해 CLI를 사용하여 GitHub 공급자와 함께 CodeStarSourceConnection 작업에 대한 작업 구성을 추가합니다.

Note

설정의 개발자 도구 콘솔을 사용하여 연결을 생성할 수도 있습니다. 연결 생성을 참조하세요.

시작하기 전:

- GitHub 계정이 생성되어 있어야 합니다.
- GitHub 코드 리포지토리가 이미 생성되어 있어야 합니다.
- CodePipeline 서비스 역할이 2019년 12월 18일 이전에 생성된 경우, AWS CodeStar 연결에 대해 codestar-connections:UseConnection을 사용하도록 권한을 업데이트해야 할 수 있습니다. 지침은 CodePipeline 서비스 역할에 권한 추가 단원을 참조하십시오.

연결을 생성하려면 GitHub 조직 소유자여야 합니다. 조직 소속이 아닌 리포지토리의 경우 리포 지토리 소유자여야 합니다.

주제

- GitHub에 대한 연결 생성(콘솔)
- GitHub에 대한 연결 생성(CLI)

GitHub에 대한 연결 생성(콘솔)

다음 단계를 사용하여 CodePipeline 콘솔을 사용하여 GitHub 또는 GitHub Enterprise Cloud 리포지토 리에 연결 작업을 추가할 수 있습니다.

Note

이 단계에서는 리포지토리 액세스에서 특정 리포지토리를 선택할 수 있습니다. 선택되지 않은 리포지토리는 CodePipeline에서 접근하거나 볼 수 없습니다.

1단계: 파이프라인 생성 또는 편집

- 1. CodePipeline 콘솔에 로그인합니다.
- 2. 다음 중 하나를 선택합니다.
 - 파이프라인을 생성하려면 선택합니다. 파이프라인 생성의 단계에 따라 첫 화면을 완료하고 다 음을 선택합니다. 소스 페이지의 소스 공급자에서 GitHub(GitHub 앱을 통해)를 선택합니다.
 - 기존 파이프라인을 편집하려면 선택합니다. 편집을 선택하고 단계 편집을 선택합니다. 소스 작 업을 추가 또는 편집하려면 선택합니다. 작업 편집 페이지의 작업 이름에 작업 이름을 입력합니다. 작업 공급자에서 GitHub(GitHub 앱을 통해)를 선택합니다.
- 3. 다음 중 하나를 수행합니다.
 - 연결에서 공급자와의 연결을 아직 생성하지 않은 경우 GitHub에 연결을 선택합니다. 2단계: GitHub에 대한 연결 생성으로 이동합니다.
 - 연결에서 공급자와의 연결을 이미 생성한 경우 연결을 선택합니다. 3단계: 연결에 대한 소스 작 업 저장으로 이동합니다.

2단계: GitHub에 대한 연결 생성

연결을 생성하도록 선택하면 GitHub에 연결 페이지가 표시됩니다.

reate a connection Info	
Create GitHub App connection	
Connection name	
githubc-connectid	

GitHub에 대한 연결을 생성하려면

- 1. GitHub connection settings(GitHub 연결 설정) 아래의 Connection name(연결 이름)에 연결 이름 이 표시됩니다. GitHub에 연결을 선택합니다. 액세스 요청 페이지가 표시됩니다.
- 2. GitHub용 AWS 커넥터 권한 부여를 선택합니다. 연결 페이지가 나타나고 [GitHub 앱(GitHub Apps)] 필드가 표시됩니다.

GitHub connection settings	Info		
Connection name			
githubc-connection			
SitHub Apps SitHub Apps create a link for your connectio	n with GitHub. To start, install a new app ar	id save tl	his connection.
Q		or	Install a new app

3. [GitHub 앱(GitHub apps)]에서 앱 설치를 선택하거나 [새 앱 설치(Install a new app)]을 선택하여 앱을 새로 만듭니다.

특정 공급자에 대한 모든 연결에 대해 하나의 앱을 설치합니다. AWS Connector for GitHub 앱을 이미 설치한 경우 해당 앱을 선택하고이 단계를 건너뜁니다.

<u>사용자 액세스 토큰</u>을 생성하려면 GitHub용 AWS 커넥터 앱을 이미 설치했는지 확인한 다 음 앱 설치 필드를 비워 둡니다. CodeConnections는 연결에 사용자 액세스 토큰을 사용합 니다.

4. GitHub용 AWS 커넥터 설치 페이지에서 앱을 설치할 계정을 선택합니다.

Note

각 GitHub 계정마다 앱을 한 번만 설치합니다. 이전에 앱을 설치한 경우 [구성(Configure)] 을 선택하여 앱 설치의 수정 페이지로 이동하거나 뒤로 버튼을 사용하여 콘솔로 돌아갈 수 있습니다.

- 5. GitHub용 AWS 커넥터 설치 페이지에서 기본값을 그대로 두고 설치를 선택합니다.
- 6. [GitHub에 연결(Connect to GitHub)] 페이지의 [GitHub 앱(GitHub Apps)]에 새 설치의 연결 ID가 표시됩니다. 연결을 선택합니다.

3단계: GitHub 소스 작업 저장

작업 편집 페이지에서 다음 단계를 사용하여 소스 작업을 연결 정보와 함께 저장합니다.

GitHub 소스 작업을 저장하려면

- 1. 리포지토리 이름에서 타사 리포지토리의 이름을 선택합니다.
- 작업이 CodeConnections 작업인 경우 파이프라인 트리거에서 트리거를 추가할 수 있습니다. 파이 프라인 트리거 구성을 구성하고 선택적으로 트리거로 필터링하려면 <u>코드 푸시 또는 풀 요청 이벤</u> 트 유형을 사용하여 트리거 추가에서 자세한 내용을 참조하세요.
- 3. Output artifact format(출력 아티팩트 형식)에서 아티팩트의 형식을 선택해야 합니다.
 - 기본 방법을 사용하여 GitHub 작업의 출력 아티팩트를 저장하려면 CodePipeline 기본 방법을 선택합니다. 그러면 GitHub 리포지토리의 파일에 액세스하여 파이프라인 아티팩트 스토어에 ZIP 파일로 아티팩트가 저장됩니다.
 - 다운스트림 작업이 Git 명령을 직접 수행할 수 있도록 리포지토리에 대한 URL 참조가 포함된 JSON 파일을 저장하려면 Full clone(전체 복제)을 선택합니다. 이 옵션은 CodeBuild 다운스트림 작업에서만 사용할 수 있습니다.

이 옵션을 선택하면 <u>Bitbucket, GitHub, GitHub Enterprise Server 또는 GitLab.com 연결을 위한</u> <u>CodeBuild GitClone 권한 추가</u>과 같이 CodeBuild 프로젝트 서비스 역할에 대한 권한을 업데이 트해야 합니다. 전체 복제 옵션을 사용하는 방법을 보여주는 자습서는 <u>자습서: GitHub 파이프라</u> 인 소스와 함께 전체 복제 사용을 참조하세요.

4. 마법사에서 다음을 선택하거나 작업 편집 페이지에서 저장을 선택합니다.

GitHub에 대한 연결 생성(CLI)

AWS Command Line Interface (AWS CLI)를 사용하여 연결을 생성할 수 있습니다.

이렇게 하려면 create-connection 명령을 사용합니다.

▲ Important

AWS CLI 또는를 통해 생성된 연결 AWS CloudFormation 은 기본적으로 PENDING 상태입니 다. CLI 또는 와의 연결을 생성한 후 콘솔을 AWS CloudFormation사용하여 연결을 편집하여 상태를 로 설정합니다AVAILABLE.

연결을 생성하는 방법

1. 터미널(Linux, macOS, Unix) 또는 명령 프롬프트(Windows)를 엽니다. AWS CLI 를 사용하여 연결 을 --connection-name 위해 --provider-type 및를 지정하여 create-connection 명령을 실 행합니다. 이 예제에서 타사 공급자 이름은 GitHub이고 지정된 연결 이름은 MyConnection입니 다.

aws codestar-connections create-connection --provider-type GitHub --connection-name
MyConnection

이 명령이 제대로 실행되면 다음과 비슷한 연결 ARN 정보가 반환됩니다.

```
{
    "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/
aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f"
}
```

2. 콘솔을 사용하여 연결을 완료합니다. 자세한 내용은 <u>보류 중인 연결 업데이트</u>를 참조하세요.

- 파이프라인은 기본적으로 연결 소스 리포지토리로 코드를 푸시할 때 변경 사항을 감지합니다. 수 동 릴리스 또는 Git 태그에 대한 파이프라인 트리거 구성을 구성하려면 다음 중 하나를 수행합니다.
 - 수동 릴리스로만 시작하도록 파이프라인 트리거 구성을 구성하려면 구성에 다음 줄을 추가하세
 요.

"DetectChanges": "false",

 트리거로 필터링하도록 파이프라인 트리거 구성을 구성하려면 <u>코드 푸시 또는 풀 요청 이벤트</u> <u>유형을 사용하여 트리거 추가</u>에서 자세한 내용을 참조하세요. 예를 들어, 다음은 파이프라인 JSON 정의의 파이프라인 수준에 추가됩니다. 이 예제에서, release-v0 및 release-v1은 포 함할 Git 태그이고 release-v2는 제외할 Git 태그입니다.



GitHub Enterprise Server 연결

연결을 사용하면 타사 공급자를 AWS 리소스와 연결하는 구성을 승인하고 설정할 수 있습니다. 타사 리포지토리를 파이프라인의 소스로 연결하려면 연결을 사용합니다.

계정에서 기존 연결을 생성하거나 사용하는 대신 다른 계정 간의 공유 연결을 사용할 수 있습니다 AWS 계정. 다른 계정과 공유된 연결 사용을 참조하세요.

Note

이 기능은 아시아 태평양(홍콩), 아시아 태평양(하이데라바드), 아시아 태평양(자카르타), 아시아 태평양(멜버른), 아시아 태평양(오사카), 아프리카(케이프타운), 중동(바레인), 중동 (UAE), 유럽(스페인), 유럽(취리히), 이스라엘(텔아비브) 또는 AWS GovCloud(미국 서부) 리전에서는 사용할 수 없습니다. 사용 가능한 다른 작업을 참조하려면 <u>CodePipeline과 제</u> <u>품 및 서비스 통합</u>을 참조하세요. 유럽(밀라노) 리전에서 이 조치를 고려할 경우 <u>Bitbucket</u> <u>Cloud, GitHub, GitHub Enterprise Server, GitLab.com 및 GitLab 자체 관리형 작업용</u> <u>CodeStarSourceConnection</u>의 참고 사항을 참조하세요.

CodePipeline에 GitHub Enterprise Server 소스 작업을 추가하려면 다음 중 하나를 선택할 수 있습니다.

- CodePipeline 콘솔 파이프라인 생성 마법사 또는 작업 편집 페이지를 사용하여 GitHub Enterprise Server 공급자 옵션을 선택합니다. 작업을 추가하려면 <u>GitHub Enterprise Server에 대한 연결 생성</u> (콘솔)을 참조하세요. 콘솔을 사용하면 호스트 리소스와 연결 리소스를 만들 수 있습니다.
- CLI를 사용하여 GitHubEnterpriseServer 공급자와 함께 CreateSourceConnection 작업에 대한 작업 구성을 추가하고 리소스를 생성합니다.
 - 연결 리소스를 생성하려면 <u>GitHub Enterprise Server에 대한 연결 및 호스트 생성(CLI)</u>을 참조하여 CLI를 사용하여 호스트 리소스 및 연결 리소스를 생성합니다.
 - <u>Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com 및 GitLab 자체 관리형 작업용</u> <u>CodeStarSourceConnection</u>의 CreateSourceConnection 예제 작업 구성을 사용하여 <u>파이프</u> 라인 생성(CLI)과 같이 작업을 추가합니다.

Note

설정의 개발자 도구 콘솔을 사용하여 연결을 생성할 수도 있습니다. <u>연결 생성</u>을 참조하세요.

시작하기 전:

• GitHub Enterprise Server에 계정을 생성하고 인프라에 GitHub Enterprise Server 인스턴스를 설치해 야 합니다.

Note

각 VPC는 한 번에 하나의 호스트(GitHub 엔터프라이즈 서버 인스턴스)에만 연결 가능합니다.

• GitHub Enterprise Server가 있는 코드 리포지토리가 생성되어 있어야 합니다.

주제

- GitHub Enterprise Server에 대한 연결 생성(콘솔)
- GitHub Enterprise Server에 대한 연결 및 호스트 생성(CLI)

GitHub Enterprise Server에 대한 연결 생성(콘솔)

다음 단계를 사용하여 CodePipeline 콘솔을 사용하여 GitHub Enterprise Server 리포지토리에 연결 작 업을 추가할 수 있습니다.

Note

GitHub Enterprise Server 연결은 연결을 만드는 데 사용된 GitHub Enterprise Server 계정이 소유한 리포지토리에 대한 액세스 권한만 제공합니다.

시작하기 전:

GitHub Enterprise Server에 대한 호스트 연결의 경우 연결에 대한 호스트 리소스를 생성하기 위한 단계를 완료해야 합니다. <u>연결을 위한 호스트 관리</u>를 참조하세요.

1단계: 파이프라인 생성 또는 편집

파이프라인을 생성 또는 편집하려면

- 1. CodePipeline 콘솔에 로그인합니다.
- 2. 다음 중 하나를 선택합니다.

- 파이프라인을 생성하려면 선택합니다. 파이프라인 생성의 단계에 따라 첫 화면을 완료하고 다 음을 선택합니다. 소스 페이지의 소스 공급자에서 GitHub Enterprise Server를 선택합니다.
- 기존 파이프라인을 편집하려면 선택합니다. 편집을 선택하고 단계 편집을 선택합니다. 소스 작 업을 추가 또는 편집하려면 선택합니다. 작업 편집 페이지의 작업 이름에 작업 이름을 입력합니다. 작업 공급자에서 GitHub Enterprise Server를 선택합니다.
- 3. 다음 중 하나를 수행합니다.
 - 연결에서 공급자와의 연결을 아직 생성하지 않은 경우 GitHub Enterprise Server에 연결을 선택 합니다. 2단계: GitHub Enterprise Server에 대한 연결 생성으로 이동합니다.
 - 연결에서 공급자와의 연결을 이미 생성한 경우 연결을 선택합니다. 3단계: 연결에 대한 소스 작 업 저장으로 이동합니다.

GitHub Enterprise Server에 대한 연결 생성

연결을 생성하도록 선택하면 GitHub Enterprise Server에 연결 페이지가 표시됩니다.

A Important

AWS CodeConnections 는 릴리스에서 알려진 문제로 인해 GitHub Enterprise Server 버전 2.22.0을 지원하지 않습니다. 연결하려면 버전 2.22.1 또는 사용 가능한 최신 버전으로 업그레 이드하세요.

GitHub Enterprise Server에 연결하려면

- 1. [연결 이름(Connection name)]에 연결 이름을 입력합니다.
- 2. [URL]에 서버의 엔드포인트를 입력합니다.

Note

제공된 URL이 연결을 위해 GitHub Enterprise Server를 설정하는 데 이미 사용된 경우, 해 당 엔드포인트에 대해 이전에 생성된 호스트 리소스 ARN을 선택하라는 메시지가 표시됩 니다.

3. Amazon VPC로 서버를 시작한 후 VPC에 연결하려는 경우 [VPC 사용(Use a VPC)]을 선택하고 다음을 완료합니다.

- a. [VPC ID]에서 VPC ID를 선택합니다. GitHub Enterprise Server 인스턴스가 설치된 인프라의 VPC를 선택하거나 VPN 또는 Direct Connect를 통해 GitHub Enterprise Server 인스턴스에 액세스할 수 있는 VPC를 선택해야 합니다.
- b. [서브넷 ID(Subnet ID)]를 선택하고 [추가(Add)]를 선택합니다. 해당 필드에서 호스트에 사용 할 서브넷 ID를 선택합니다. 서브넷은 최대 10개까지 선택할 수 있습니다.

GitHub Enterprise Server 인스턴스가 설치된 인프라의 서브넷를 선택하거나 VPN 또는 Direct Connect를 통해 설치된 GitHub Enterprise Server 인스턴스에 액세스할 수 있는 서브넷 을 선택해야 합니다.

c. [보안 그룹 ID(Security group IDs)]에서 [추가(Add)]를 선택합니다. 해당 필드에서 호스트에 사 용할 보안 그룹을 선택합니다. 보안 그룹은 최대 10개까지 선택할 수 있습니다.

GitHub Enterprise Server 인스턴스가 설치된 인프라의 보안 그룹을 선택하거나 VPN 또는 Direct Connect를 통해 설치된 GitHub Enterprise Server 인스턴스에 액세스할 수 있는 보안 그룹을 선택해야 합니다.

d. 프라이빗 VPC 가 구성되어 있고 퍼블릭이 아닌 인증 기관을 사용하여 TLS 검증을 수행하도 록 GitHub Enterprise Server 인스턴스를 구성한 경우 [TLS 인증서(TLS certificate)]에 인증서 ID를 입력합니다. TLS 인증서 값은 인증서의 퍼블릭 키여야 합니다.

Q vpc-09	×	
Subnet IDs Choose the subnet or subnets for the VPC in which your Git	Hub Enterprise Server is confi	igured.
jubnet ID		
v subnet-/d	X	Remove
Security group IDs Thoose the security group or groups for the VPC in which y Security group ID	our GitHub Enterprise Server i	is configured.
Choose the security group IDs Choose the security group or groups for the VPC in which y Security group ID Q sg-00	our GitHub Enterprise Server	is configured. Remove

- [GitHub Enterprise Server Server에 연결(Connect to GitHub Enterprise Server)]을 선택합니다. 생 성된 연결은 [대기 중(Pending)] 상태로 표시됩니다. 사용자가 제공한 서버 정보를 사용하여 연결 을 위한 호스트 리소스가 생성됩니다. 호스트 이름으로 URL이 사용됩니다.
- 5. [보류 중인 연결을 업데이트(Update pending connection)]를 선택합니다.
- 6. 메시지가 표시되면 GitHub Enterprise 로그인 페이지에서 GitHub Enterprise 자격 증명으로 로그 인합니다.
- 7. [GitHub 앱 생성(Create GitHub App)] 페이지에서 앱의 이름을 선택합니다.
- 8. GitHub 승인 페이지에서 [<app-name> 승인(Authorize <app-name>)]을 선택합니다.
- 앱 설치 페이지에, 커넥터 앱을 설치할 준비가 되었다는 메시지가 표시됩니다. 여러 조직이 있는 경우 앱을 설치할 조직을 선택하라는 메시지가 표시될 수 있습니다.

앱을 설치할 리포지토리 설정을 선택합니다. 설치를 선택합니다.

10. 연결 페이지에 생성된 연결이 [사용 가능(Available)] 상태로 표시됩니다.

3단계: GitHub Enterprise Server 소스 작업 저장

마법사 또는 작업 편집 페이지에서 다음 단계를 사용하여 소스 작업을 연결 정보와 함께 저장합니다.

연결을 통해 소스 작업을 완료하고 저장하려면

- 1. 리포지토리 이름에서 타사 리포지토리의 이름을 선택합니다.
- 작업이 CodeConnections 작업인 경우 파이프라인 트리거에서 트리거를 추가할 수 있습니다. 파이 프라인 트리거 구성을 구성하고 선택적으로 트리거로 필터링하려면 <u>코드 푸시 또는 풀 요청 이벤</u> 트 유형을 사용하여 트리거 추가에서 자세한 내용을 참조하세요.
- 3. Output artifact format(출력 아티팩트 형식)에서 아티팩트의 형식을 선택해야 합니다.
 - 기본 방법을 사용하여 GitHub Enterprise Server 작업의 출력 아티팩트를 저장하려면 CodePipeline 기본 방법을 선택합니다. 그러면 GitHub Enterprise Server 리포지토리의 파일에 액세스하여 파이프라인 아티팩트 스토어에 ZIP 파일로 아티팩트가 저장됩니다.
 - 다운스트림 작업이 Git 명령을 직접 수행할 수 있도록 리포지토리에 대한 URL 참조가 포함된 JSON 파일을 저장하려면 Full clone(전체 복제)을 선택합니다. 이 옵션은 CodeBuild 다운스트림 작업에서만 사용할 수 있습니다.
- 4. 마법사에서 다음을 선택하거나 작업 편집 페이지에서 저장을 선택합니다.

GitHub Enterprise Server에 대한 연결 및 호스트 생성(CLI)

AWS Command Line Interface (AWS CLI)를 사용하여 연결을 생성할 수 있습니다.

이렇게 하려면 create-connection 명령을 사용합니다.

A Important

AWS CLI 또는를 통해 생성된 연결 AWS CloudFormation 은 기본적으로 PENDING 상태입니 다. CLI 또는 와의 연결을 생성한 후 콘솔을 AWS CloudFormation사용하여 연결을 편집하여 상태를 로 설정합니다AVAILABLE.

AWS Command Line Interface (AWS CLI)를 사용하여 설치된 연결을 위한 호스트를 생성할 수 있습니다.

GitHub Enterprise Server 계정별로 한 번만 호스트를 생성합니다. 특정 GitHub Enterprise Server 계정에 대한 모든 연결은 동일한 호스트를 사용합니다.

호스트를 사용하여 서드 파티 공급자가 설치된 인프라의 엔드포인트를 나타냅니다. CLI를 사용하여 호 스트 생성을 완료하면 호스트가 보류 중 상태입니다. 그러면 호스트를 설정 또는 등록하여 사용 가능 상태로 전환합니다. 호스트를 사용할 수 있게 되면 연결을 생성하는 단계를 완료합니다.

이렇게 하려면 create-host 명령을 사용합니다.

A Important

를 통해 생성된 호스트 AWS CLI 는 기본적으로 Pending 상태입니다. CLI를 사용하여 호스트 를 생성한 후 콘솔 또는 CLI를 통해 호스트를 설정하여 호스트를 상태를 Available로 전환합 니다.

호스트를 생성하는 방법

 터미널(Linux, macOS, Unix) 또는 명령 프롬프트(Windows)를 엽니다. AWS CLI 를 사용하여 연결 을 --provider-endpoint 위해 --name, --provider-type및를 지정하여 create-host 명령 을 실행합니다. 이 예제에서 서드 파티 공급자 이름은 GitHubEnterpriseServer이고 엔드포인 트는 my-instance.dev입니다.

aws codestar-connections create-host --name MyHost --provider-type GitHubEnterpriseServer --provider-endpoint "https://my-instance.dev"

이 명령이 제대로 실행되면 다음과 비슷한 호스트 Amazon 리소스 이름(ARN) 정보가 반환됩니다.

```
{
    "HostArn": "arn:aws:codestar-connections:us-west-2:account_id:host/My-
Host-28aef605"
}
```

이 단계가 끝나면 호스트는 PENDING 상태입니다.

2. 콘솔을 사용하여 호스트 설정을 완료하고 호스트를 Available 상태로 전환합니다.

GitHub Enterprise Server에 대한 연결을 생성하려면

 터미널(Linux, macOS, Unix) 또는 명령 프롬프트(Windows)를 엽니다. AWS CLI 를 사용하여 연결 을 --connection-name 위해 --host-arn 및를 지정하여 create-connection 명령을 실행합니 다.

```
aws codestar-connections create-connection --host-arn arn:aws:codestar-
connections:us-west-2:account_id:host/MyHost-234EXAMPLE --connection-name
MyConnection
```

이 명령이 제대로 실행되면 다음과 비슷한 연결 ARN 정보가 반환됩니다.

{
 "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/
aEXAMPLE-8aad"
}

- 2. 콘솔을 사용하여 보류 중인 연결을 설정합니다.
- 파이프라인은 기본적으로 연결 소스 리포지토리로 코드를 푸시할 때 변경 사항을 감지합니다. 수 동 릴리스 또는 Git 태그에 대한 파이프라인 트리거 구성을 구성하려면 다음 중 하나를 수행합니다.
 - 수동 릴리스로만 시작하도록 파이프라인 트리거 구성을 구성하려면 구성에 다음 줄을 추가하세 요.

"DetectChanges": "false",

 트리거로 필터링하도록 파이프라인 트리거 구성을 구성하려면 <u>코드 푸시 또는 풀 요청 이벤트</u> <u>유형을 사용하여 트리거 추가</u>에서 자세한 내용을 참조하세요. 예를 들어, 다음은 파이프라인 JSON 정의의 파이프라인 수준에 추가됩니다. 이 예제에서, release-v0 및 release-v1은 포 함할 Git 태그이고 release-v2는 제외할 Git 태그입니다.



GitLab.com 연결

연결을 사용하면 타사 공급자를 AWS 리소스와 연결하는 구성을 승인하고 설정할 수 있습니다. 타사 리포지토리를 파이프라인의 소스로 연결하려면 연결을 사용합니다.

Note

계정에서 기존 연결을 생성하거나 사용하는 대신 다른 계정 간의 공유 연결을 사용할 수 있습니다 AWS 계정. 다른 계정과 공유된 연결 사용을 참조하세요.

Note

이 기능은 아시아 태평양(홍콩), 아시아 태평양(하이데라바드), 아시아 태평양(자카르타), 아시아 태평양(멜버른), 아시아 태평양(오사카), 아프리카(케이프타운), 중동(바레인), 중동 (UAE), 유럽(스페인), 유럽(취리히), 이스라엘(텔아비브) 또는 AWS GovCloud(미국 서부) 리전에서는 사용할 수 없습니다. 사용 가능한 다른 작업을 참조하려면 <u>CodePipeline과 제</u> <u>품 및 서비스 통합</u>을 참조하세요. 유럽(밀라노) 리전에서 이 조치를 고려할 경우 <u>Bitbucket</u> <u>Cloud, GitHub, GitHub Enterprise Server, GitLab.com 및 GitLab 자체 관리형 작업용</u> <u>CodeStarSourceConnection</u>의 참고 사항을 참조하세요.

CodePipeline에 GitLab.com 소스 작업을 추가하려면 다음 중 하나를 선택할 수 있습니다.

- CodePipeline 콘솔 파이프라인 생성 마법사 또는 작업 편집 페이지를 사용하여 GitLab 공급자 옵션 을 선택합니다. 작업을 추가하려면 <u>GitLab.com에 대한 연결 생성(콘솔)</u>을 참조하세요. 콘솔을 사용 하면 연결 리소스를 만들 수 있습니다.
- 다음과 같이 CLI를 사용하여 GitLab 공급자와 함께 CreateSourceConnection 작업에 대한 작 업 구성을 추가합니다.
 - 연결 리소스를 생성하려면 <u>GitLab.com에 대한 연결 생성(CLI)</u>을 참조하여 CLI를 사용하여 연결 리 소스를 생성합니다.
 - <u>Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com 및 GitLab 자체 관리형 작업용</u> <u>CodeStarSourceConnection</u>의 CreateSourceConnection 예제 작업 구성을 사용하여 <u>파이프</u> <u>라인 생성(CLI)</u>과 같이 작업을 추가합니다.

설정의 개발자 도구 콘솔을 사용하여 연결을 생성할 수도 있습니다. <u>연결 생성</u>을 참조하세요.

1 Note

GitLab.com에서 이 연결 설치를 승인하면 계정에 액세스하여 데이터를 처리할 수 있는 권한을 서비스에 부여하고 언제든지 애플리케이션을 제거하여 권한을 취소할 수 있습니다.

시작하기 전:

• GitLab.com 계정이 이미 생성되어 있어야 합니다.

1 Note

연결은 연결을 만들고 권한을 부여하는 데 사용된 계정이 소유한 리포지토리에 대한 액세스 권한만 제공합니다.
1 Note

GitLab에서 소유자 역할을 가진 리포지토리에 대한 연결을 생성한 다음 CodePipeline과 같 은 리소스가 있는 리포지토리에서 해당 연결을 사용할 수 있습니다. 그룹 내 리포지토리의 경우 그룹 소유자가 아니어도 됩니다.

• 파이프라인 소스를 지정하려면 gitlab.com에 리포지토리가 이미 생성되어 있어야 합니다.

주제

- GitLab.com에 대한 연결 생성(콘솔)
- GitLab.com에 대한 연결 생성(CLI)

GitLab.com에 대한 연결 생성(콘솔)

다음 단계를 사용하여 CodePipeline 콘솔을 사용하여 GitLab의 프로젝트(리포지토리)에 연결 작업을 추가할 수 있습니다.

파이프라인을 생성 또는 편집하려면

- 1. CodePipeline 콘솔에 로그인합니다.
- 2. 다음 중 하나를 선택합니다.
 - 파이프라인을 생성하려면 선택합니다. 파이프라인 생성의 단계에 따라 첫 화면을 완료하고 다 음을 선택합니다. 소스 페이지의 소스 공급자에서 GitLab을 선택합니다.
 - 기존 파이프라인을 편집하려면 선택합니다. 편집을 선택하고 단계 편집을 선택합니다. 소스 작 업을 추가 또는 편집하려면 선택합니다. 작업 편집 페이지의 작업 이름에 작업 이름을 입력합니다. 작업 공급자에서 GitLab을 선택합니다.
- 3. 다음 중 하나를 수행합니다.
 - 연결에서 공급자와의 연결을 아직 생성하지 않은 경우 GitLab에 연결을 선택합니다. 4단계로 이 동하여 연결을 생성합니다.
 - 연결에서 공급자와의 연결을 이미 생성한 경우 연결을 선택합니다. 9단계로 이동합니다.

Note

GitLab.com 연결이 생성되기 전에 팝업 창을 닫으면 페이지를 새로 고쳐야 합니다.

4. GitLab.com 리포지토리에 대한 연결을 생성하려면 공급자 선택에서 GitLab을 선택합니다. [연결 이름(Connection name)]에 생성하려는 연결의 이름을 입력합니다. GitLab에 연결을 선택합니다.

Developer Tools > Connections > Create connection	
Create a connection Info	
Create GitLab connection Info	
Connection name	
► Tags - optional	
	Connect to GitLab

- 5. GitLab.com의 로그인 페이지가 표시되면 로그인 보안 인증 정보를 입력한 다음 로그인을 선택합 니다.
- 처음 연결을 승인하는 경우에는 GitLab.com 계정에 액세스하기 위한 연결 승인을 요청하는 메시 지와 함께 권한 부여 페이지가 표시됩니다.

Authorize를 선택합니다.

Authorize codestar-connections to use your account?

An application called codestar-connections 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:

Gran	ts complete r	ead/write acco	ess to the API, incl	uding all		
grou regis	ps and projec try.	ts, the contair	ier registry, and th	е раскаде		
Read the authenticated user's personal information						
Gran throu publi endp	ts read-only a 1gh the /user 1c email, and 1 100ints under /	access to the a API endpoint, full name. Also users.	authenticated user which includes us o grants access to	's profile ername, read-only A		
Read						
Gran the c	ts read acces container regi	s to the API, in stry, and the p	ncluding all groups ackage registry.	and projec		
Allo	ws read-only	access to the	repository			
Gran using	ts read-only a g Git-over-HT	access to repo TP or the Repo	ositories on private ository Files API.	projects		
Allo	ws read-write	e access to the	e repository			
Gran using	ts read-write g Git-over-HT	access to rep TP (not using	ositories on privat the API).	e projects		
			Denv	Authoriz		

- 7. 브라우저가 연결 콘솔 페이지로 돌아갑니다. GitLab 연결 생성에서 연결 이름에 새 연결이 표시됩 니다.
- 8. GitLab에 연결을 선택합니다.

CodePipeline 콘솔로 돌아가게 됩니다.

Note

GitLab.com 연결이 성공적으로 생성되면 성공 배너가 기본 창에 표시됩니다. 현재 시스템에서 이전에 GitLab에 로그인한 적이 없다면 팝업 창을 수동으로 닫아야 합니 다.

9. 리포지토리 이름에서 네임스페이스와 함께 프로젝트 경로를 지정하여 GitLab의 프로젝트 이 름을 선택합니다. 예를 들어 그룹 수준 리포지토리의 경우 리포지토리 이름을 group-name/ repository-name 형식으로 입력합니다. 경로와 네임스페이스에 대한 자세한 내용은 <u>https:// docs.gitlab.com/ee/api/projects.html#get-single-project</u>의 path_with_namespace 필드를 참조하세요. GitLab의 네임스페이스에 대한 자세한 내용은 <u>https://docs.gitlab.com/ee/user/</u> <u>namespace/</u>를 참조하세요.

Note

GitLab에 있는 그룹의 경우 네임스페이스를 사용하여 프로젝트 경로를 수동으로 지정해야 합니다. 예를 들어 그룹 mygroup의 myrepo라는 리포지토리의 경우 mygroup/myrepo 형식으로 입력합니다. GitLab의 URL에서 네임스페이스를 사용하여 프로젝트 경로를 찾을 수 있습니다.

- 작업이 CodeConnections 작업인 경우 파이프라인 트리거에서 트리거를 추가할 수 있습니다. 파이 프라인 트리거 구성을 구성하고 선택적으로 트리거로 필터링하려면 <u>코드 푸시 또는 풀 요청 이벤</u> 트 유형을 사용하여 트리거 추가에서 자세한 내용을 참조하세요.
- 11. 브랜치 이름에서, 파이프라인에서 소스 변경 사항을 감지할 브랜치를 선택합니다.

Note

브랜치 이름이 자동으로 채워지지 않으면 리포지토리에 대한 소유자 액세스 권한이 없습 니다. 프로젝트 이름이 유효하지 않거나 사용된 연결에 프로젝트/리포지토리에 대한 액세 스 권한이 없습니다.

- 12. Output artifact format(출력 아티팩트 형식)에서 아티팩트의 형식을 선택해야 합니다.
 - 기본 방법을 사용하여 GitLab.com 작업의 출력 아티팩트를 저장하려면 CodePipeline 기본 방 법을 선택합니다. 그러면 GitLab.com 리포지토리의 파일에 액세스하여 파이프라인 아티팩트 스 토어에 ZIP 파일로 아티팩트가 저장됩니다.

 다운스트림 작업이 Git 명령을 직접 수행할 수 있도록 리포지토리에 대한 URL 참조가 포함된 JSON 파일을 저장하려면 Full clone(전체 복제)을 선택합니다. 이 옵션은 CodeBuild 다운스트림 작업에서만 사용할 수 있습니다.

이 옵션을 선택하면 <u>Bitbucket, GitHub, GitHub Enterprise Server 또는 GitLab.com 연결을 위한</u> <u>CodeBuild GitClone 권한 추가</u>과 같이 CodeBuild 프로젝트 서비스 역할에 대한 권한을 업데이 트해야 합니다. 전체 복제 옵션을 사용하는 방법을 보여주는 자습서는 <u>자습서: GitHub 파이프라</u> 인 소스와 함께 전체 복제 사용을 참조하세요.

13. 소스 작업을 저장하고 계속하도록 선택합니다.

GitLab.com에 대한 연결 생성(CLI)

AWS Command Line Interface (AWS CLI)를 사용하여 연결을 생성할 수 있습니다.

이렇게 하려면 create-connection 명령을 사용합니다.

A Important

AWS CLI 또는를 통해 생성된 연결 AWS CloudFormation 은 기본적으로 PENDING 상태입니 다. CLI 또는 와의 연결을 생성한 후 콘솔을 AWS CloudFormation사용하여 연결을 편집하여 상태를 로 설정합니다AVAILABLE.

연결을 생성하는 방법

 터미널(Linux, macOS, Unix) 또는 명령 프롬프트(Windows)를 엽니다. AWS CLI 를 사용하여 연결 을 --connection-name 위해 --provider-type 및를 지정하여 create-connection 명령을 실 행합니다. 이 예제에서 타사 공급자 이름은 GitLab이고 지정된 연결 이름은 MyConnection입니 다.

aws codestar-connections create-connection --provider-type GitLab --connection-name
MyConnection

이 명령이 제대로 실행되면 다음과 비슷한 연결 ARN 정보가 반환됩니다.

"ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/ aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f"

{

}

- 2. 콘솔을 사용하여 연결을 완료합니다. 자세한 내용은 보류 중인 연결 업데이트를 참조하세요.
- 파이프라인은 기본적으로 연결 소스 리포지토리로 코드를 푸시할 때 변경 사항을 감지합니다. 수 동 릴리스 또는 Git 태그에 대한 파이프라인 트리거 구성을 구성하려면 다음 중 하나를 수행합니다.
 - 수동 릴리스로만 시작하도록 파이프라인 트리거 구성을 구성하려면 구성에 다음 줄을 추가하세 요.

"DetectChanges": "false",

 트리거로 필터링하도록 파이프라인 트리거 구성을 구성하려면 <u>코드 푸시 또는 풀 요청 이벤트</u> <u>유형을 사용하여 트리거 추가</u>에서 자세한 내용을 참조하세요. 예를 들어, 다음은 파이프라인 JSON 정의의 파이프라인 수준에 추가됩니다. 이 예제에서, release-v0 및 release-v1은 포 함할 Git 태그이고 release-v2는 제외할 Git 태그입니다.

```
"triggers": [
            {
                "providerType": "CodeStarSourceConnection",
                "gitConfiguration": {
                     "sourceActionName": "Source",
                     "push": [
                         {
                             "tags": {
                                 "includes": [
                                     "release-v0", "release-v1"
                                 ],
                                 "excludes": [
                                     "release-v2"
                                 1
                             }
                         }
                    ]
                }
            }
        ]
```

GitLab 자체 관리형을 위한 연결

연결을 사용하면 타사 공급자를 AWS 리소스와 연결하는 구성을 승인하고 설정할 수 있습니다. 타사 리포지토리를 파이프라인의 소스로 연결하려면 연결을 사용합니다.

Note

계정에서 기존 연결을 생성하거나 사용하는 대신 다른 계정 간의 공유 연결을 사용할 수 있습니다 AWS 계정. 다른 계정과 공유된 연결 사용을 참조하세요.

Note

이 기능은 아시아 태평양(홍콩), 아시아 태평양(하이데라바드), 아시아 태평양(자카르타), 아시아 태평양(멜버른), 아시아 태평양(오사카), 아프리카(케이프타운), 중동(바레인), 중동 (UAE), 유럽(스페인), 유럽(취리히), 이스라엘(텔아비브) 또는 AWS GovCloud(미국 서부) 리전에서는 사용할 수 없습니다. 사용 가능한 다른 작업을 참조하려면 <u>CodePipeline과 제</u> <u>품 및 서비스 통합</u>을 참조하세요. 유럽(밀라노) 리전에서 이 조치를 고려할 경우 <u>Bitbucket</u> <u>Cloud, GitHub, GitHub Enterprise Server, GitLab.com 및 GitLab 자체 관리형 작업용</u> <u>CodeStarSourceConnection</u>의 참고 사항을 참조하세요.

CodePipeline에 GitLab 자체 관리형 소스 작업을 추가하려면 다음 중 하나를 선택할 수 있습니다.

- CodePipeline 콘솔 파이프라인 생성 마법사 또는 작업 편집 페이지를 사용하여 GitLab 자체 관리형 공급자 옵션을 선택합니다. 작업을 추가하려면 <u>GitLab 자체 관리형에 대한 연결 생성(콘솔)</u>을 참조하 세요. 콘솔을 사용하면 호스트 리소스와 연결 리소스를 만들 수 있습니다.
- CLI를 사용하여 GitLabSelfManaged 공급자와 함께 CreateSourceConnection 작업에 대한 작업 구성을 추가하고 리소스를 생성합니다.
 - 연결 리소스를 생성하려면 <u>호스트를 만들고 GitLab 자체 관리형에 연결(CLI)</u>을 참조하여 CLI를 사 용하여 호스트 리소스 및 연결 리소스를 생성합니다.
 - <u>Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com 및 GitLab 자체 관리형 작업용</u> <u>CodeStarSourceConnection</u>의 CreateSourceConnection 예제 작업 구성을 사용하여 <u>파이프</u> 라인 생성(CLI)과 같이 작업을 추가합니다.

Note

설정의 개발자 도구 콘솔을 사용하여 연결을 생성할 수도 있습니다. 연결 생성을 참조하세요.

시작하기 전:

 이미 GitLab에 계정을 만들고 자체 관리형 설치가 가능한 GitLab Enterprise Edition 또는 GitLab Community Edition을 보유하고 있어야 합니다. 자세한 내용은 <u>https://docs.gitlab.com/ee/</u> subscriptions/self_managed/을 참조하십시오.

Note

연결은 연결을 만들고 권한을 부여하는 데 사용된 계정에 대한 액세스 권한만 제공합니다.

Note

GitLab에서 소유자 역할을 가진 리포지토리에 대한 연결을 생성한 다음 CodePipeline과 같 은 리소스에서 해당 연결을 사용할 수 있습니다. 그룹 내 리포지토리의 경우 그룹 소유자가 아니어도 됩니다.

 범위가 축소된 권한(api.)만 있는 GitLab 개인용 액세스 토큰(PAT)을 이미 생성했어야 합니다. 자세한 내용은 <u>https://docs.gitlab.com/ee/user/profile/personal_access_tokens.html</u>을 참조하십시오. PAT를 생성하고 사용하려면 관리자여야 합니다.

Note

PAT는 호스트를 인증하는 데 사용되며 연결에 달리 저장되거나 사용되지 않습니다. 호스트 를 설정하려면 임시 PAT를 만든 다음 호스트를 설정한 후 PAT를 삭제하면 됩니다.

• 호스트를 미리 설정하도록 선택할 수 있습니다. VPC가 있든 없든 호스트를 설정할 수 있습니다. VPC 구성에 대한 세부 정보 및 호스트 생성에 대한 추가 정보는 호스트 만들기를 참조하십시오.

주제

- <u>GitLab 자체 관리형에 대한</u> 연결 생성(콘솔)
- <u>호스트를 만들고 GitLab 자체</u> 관리형에 연결(CLI)

GitLab 자체 관리형에 대한 연결 생성(콘솔)

다음 단계를 사용하여 CodePipeline 콘솔을 사용하여 GitLab 자체 관리형 리포지토리에 대한 연결 작 업을 추가할 수 있습니다.

Note

GitLab 자체 관리형 연결은 연결을 만드는 데 사용된 GitLab 자체 관리형 계정이 소유한 리포지 토리에 대한 액세스 권한만 제공합니다.

시작하기 전:

GitLab 자체 관리형에 대한 호스트 연결의 경우 연결에 대한 호스트 리소스를 생성하기 위한 단계를 완 료해야 합니다. <u>연결을 위한 호스트 관리를</u> 참조하세요.

1단계: 파이프라인 생성 또는 편집

파이프라인을 생성 또는 편집하려면

- 1. CodePipeline 콘솔에 로그인합니다.
- 2. 다음 중 하나를 선택합니다.
 - 파이프라인을 생성하려면 선택합니다. 파이프라인 생성의 단계에 따라 첫 화면을 완료하고 다 음을 선택합니다. 소스 페이지의 소스 공급자에서 GitLab 자체 관리형을 선택합니다.
 - 기존 파이프라인을 편집하려면 선택합니다. 편집을 선택하고 단계 편집을 선택합니다. 소스 작 업을 추가 또는 편집하려면 선택합니다. 작업 편집 페이지의 작업 이름에 작업 이름을 입력합니다. 작업 공급자에서 GitLab 자체 관리형을 선택합니다.
- 3. 다음 중 하나를 수행합니다.
 - 연결에서 공급자와의 연결을 아직 생성하지 않은 경우 GitLab 자체 관리형에 연결을 선택합니다. 2단계: GitLab 자체 관리형에 대한 연결 생성으로 이동합니다.
 - 연결에서 공급자에 대한 연결을 이미 생성한 경우 연결을 선택한 다음 3단계: GitLab 자체 관리 형 소스 저장 작업으로 진행합니다.

2단계: GitLab 자체 관리형에 대한 연결 생성

연결을 생성하도록 선택하면 GitLab 자체 관리형에 연결 페이지가 표시됩니다.

GitLab 자체 관리형에 연결하려면

- 1. [연결 이름(Connection name)]에 연결 이름을 입력합니다.
- 2. [URL]에 서버의 엔드포인트를 입력합니다.

1 Note

제공된 URL이 연결을 위해 호스트를 설정하는 데 이미 사용된 경우, 해당 엔드포인트에 대해 이전에 생성된 호스트 리소스 ARN을 선택하라는 메시지가 표시됩니다.

- Amazon VPC로 서버를 시작한 후 VPC에 연결하려는 경우 VPC 사용을 선택하고 VPC를 위한 정 보를 작성합니다.
- GitLab 자체 관리형에 연결을 선택합니다. 생성된 연결은 [대기 중(Pending)] 상태로 표시됩니다. 사용자가 제공한 서버 정보를 사용하여 연결을 위한 호스트 리소스가 생성됩니다. 호스트 이름으 로 URL이 사용됩니다.
- 5. [보류 중인 연결을 업데이트(Update pending connection)]를 선택합니다.
- 공급자로 계속 이동할지 확인하는 리디렉션 메시지가 포함된 페이지가 열리면 계속을 선택합니다.
 다. 공급자에 대한 승인을 입력합니다.
- host_name 설정 페이지가 표시됩니다. 개인용 액세스 토큰 제공에서 GitLab PAT에 다음과 같은 범위 축소 권한인 api만 제공하십시오.

Note

관리자만 PAT를 생성하고 사용할 수 있습니다.

Continue(계속)을 선택합니다.

Set up myhostgl
Provide personal access token
To set up GitLab self-managed, provide your personal access token from GitLab. The personal access token is required to have the following scoped-down permissions only: api.
Cancel Continue

8. 연결 페이지에 생성된 연결이 [사용 가능(Available)] 상태로 표시됩니다.

3단계: GitLab 자체 관리형 소스 작업 저장

마법사 또는 작업 편집 페이지에서 다음 단계를 사용하여 소스 작업을 연결 정보와 함께 저장합니다.

연결을 통해 소스 작업을 완료하고 저장하려면

- 1. 리포지토리 이름에서 타사 리포지토리의 이름을 선택합니다.
- 작업이 CodeConnections 작업인 경우 파이프라인 트리거에서 트리거를 추가할 수 있습니다. 파이 프라인 트리거 구성을 구성하고 선택적으로 트리거로 필터링하려면 <u>코드 푸시 또는 풀 요청 이벤</u> 트 유형을 사용하여 트리거 추가에서 자세한 내용을 참조하세요.
- 3. Output artifact format(출력 아티팩트 형식)에서 아티팩트의 형식을 선택해야 합니다.
 - 기본 방법을 사용하여 GitLab 자체 관리형 작업의 출력 아티팩트를 저장하려면 CodePipeline 기 본 방법을 선택합니다. 그러면 리포지토리의 파일에 액세스하여 파이프라인 아티팩트 스토어에 ZIP 파일로 아티팩트가 저장됩니다.
 - 다운스트림 작업이 Git 명령을 직접 수행할 수 있도록 리포지토리에 대한 URL 참조가 포함된 JSON 파일을 저장하려면 Full clone(전체 복제)을 선택합니다. 이 옵션은 CodeBuild 다운스트림 작업에서만 사용할 수 있습니다.
- 4. 마법사에서 다음을 선택하거나 작업 편집 페이지에서 저장을 선택합니다.

호스트를 만들고 GitLab 자체 관리형에 연결(CLI)

AWS Command Line Interface (AWS CLI)를 사용하여 연결을 생성할 수 있습니다.

이렇게 하려면 create-connection 명령을 사용합니다.

🛕 Important

AWS CLI 또는를 통해 생성된 연결 AWS CloudFormation 은 기본적으로 PENDING 상태입니 다. CLI 또는 와의 연결을 생성한 후 콘솔을 AWS CloudFormation사용하여 연결을 편집하여 상태를 로 설정합니다AVAILABLE.

AWS Command Line Interface (AWS CLI)를 사용하여 설치된 연결을 위한 호스트를 생성할 수 있습니 다.

호스트를 사용하여 서드 파티 공급자가 설치된 인프라의 엔드포인트를 나타냅니다. CLI를 사용하여 호 스트 생성을 완료하면 호스트가 보류 중 상태입니다. 그러면 호스트를 설정 또는 등록하여 사용 가능 상태로 전환합니다. 호스트를 사용할 수 있게 되면 연결을 생성하는 단계를 완료합니다.

이렇게 하려면 create-host 명령을 사용합니다.

A Important

를 통해 생성된 호스트 AWS CLI 는 기본적으로 Pending 상태입니다. CLI를 사용하여 호스트 를 생성한 후 콘솔 또는 CLI를 통해 호스트를 설정하여 호스트를 상태를 Available로 전환합 니다.

호스트를 생성하는 방법

 터미널(Linux, macOS, Unix) 또는 명령 프롬프트(Windows)를 엽니다. AWS CLI 를 사용하여 연결 을 --provider-endpoint 위해 --name, --provider-type및를 지정하여 create-host 명령 을 실행합니다. 이 예제에서 서드 파티 공급자 이름은 GitLabSelfManaged이고 엔드포인트는 my-instance.dev입니다.

aws codestar-connections create-host --name MyHost --provider-type GitLabSelfManaged --provider-endpoint "https://my-instance.dev"

이 명령이 제대로 실행되면 다음과 비슷한 호스트 Amazon 리소스 이름(ARN) 정보가 반환됩니다.

```
{
    "HostArn": "arn:aws:codestar-connections:us-west-2:account_id:host/My-
Host-28aef605"
```

}

이 단계가 끝나면 호스트는 PENDING 상태입니다.

2. 콘솔을 사용하여 호스트 설정을 완료하고 호스트를 Available 상태로 전환합니다.

GitLab 자체 관리형에 대한 연결을 생성하려면

 터미널(Linux, macOS, Unix) 또는 명령 프롬프트(Windows)를 엽니다. AWS CLI 를 사용하여 연결 을 --connection-name 위해 --host-arn 및를 지정하여 create-connection 명령을 실행합니 다.

```
aws codestar-connections create-connection --host-arn arn:aws:codestar-
connections:us-west-2:account_id:host/MyHost-234EXAMPLE --connection-name
MyConnection
```

이 명령이 제대로 실행되면 다음과 비슷한 연결 ARN 정보가 반환됩니다.

```
{
    "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/
aEXAMPLE-8aad"
}
```

- 2. 콘솔을 사용하여 보류 중인 연결을 설정합니다.
- 파이프라인은 기본적으로 연결 소스 리포지토리로 코드를 푸시할 때 변경 사항을 감지합니다. 수 동 릴리스 또는 Git 태그에 대한 파이프라인 트리거 구성을 구성하려면 다음 중 하나를 수행합니다.
 - 수동 릴리스로만 시작하도록 파이프라인 트리거 구성을 구성하려면 구성에 다음 줄을 추가하세
 요.

"DetectChanges": "false",

 트리거로 필터링하도록 파이프라인 트리거 구성을 구성하려면 <u>코드 푸시 또는 풀 요청 이벤트</u> <u>유형을 사용하여 트리거 추가</u>에서 자세한 내용을 참조하세요. 예를 들어, 다음은 파이프라인 JSON 정의의 파이프라인 수준에 추가됩니다. 이 예제에서, release-v0 및 release-v1은 포 함할 Git 태그이고 release-v2는 제외할 Git 태그입니다.

```
"triggers": [
{
```

```
"providerType": "CodeStarSourceConnection",
        "gitConfiguration": {
            "sourceActionName": "Source",
            "push": [
                 {
                     "tags": {
                         "includes": [
                             "release-v0", "release-v1"
                         ],
                         "excludes": [
                             "release-v2"
                         ]
                     }
                }
            ]
        }
    }
]
```

다른 계정과 공유된 연결 사용

를 사용하여 공유 연결을 생성하고 관리할 수 있습니다 AWS RAM. 이렇게 하면 타사 리포지토리에 액 세스하기 AWS 계정 위해 간에 연결을 공유할 수 있습니다. 이를 통해 계정 간 CodePipeline 파이프라 인에서 단일 연결을 사용하는 동시에 사용자가 각 계정에서 별도의 연결을 관리하고 관리할 필요가 줄 어듭니다.

CodePipeline에서 공유 연결을 사용하려면 다음을 수행합니다.

- 설정에서 개발자 도구 콘솔을 사용하여 연결을 생성합니다. 연결 생성을 참조하세요.
- 를 사용하여 리소스 공유를 설정합니다 AWS RAM. <u>연결 공유를 AWS 계정</u> 참조하세요.
- CodePipeline 콘솔 파이프라인 생성 마법사 또는 작업 편집 페이지를 사용하여 Bitbucket 공급자 옵 션과 같은 연결 공급자를 선택할 때 대상 계정과 공유된 연결을 선택할 수 있습니다.

트리거 및 필터링을 사용하여 시작 파이프라인 자동화

트리거를 사용하면 특정 브랜치 또는 풀 요청에 대한 변경 사항이 감지되는 경우와 같이 특정 이벤트 유형 또는 필터링된 이벤트 유형에서 시작하도록 파이프라인을 구성할 수 있습니다. 트리거는 GitHub, Bitbucket 및 GitLab과 같이 CodePipeline에서 CodeStarSourceConnection 작업을 사용하는 연 결을 통해 소스 작업에 대해 구성할 수 있습니다. 연결을 사용하는 소스 작업에 대한 자세한 내용은 CodeConnections를 사용하여 파이프라인에 타사 소스 공급자 추가 섹션을 참조하세요.

CodeCommit 및 S3와 같은 소스 작업은 자동 변경 감지를 사용하여 변경 시 파이프라인을 시작합니다. 자세한 내용은 CodeCommit 소스 작업 및 EventBridge 단원을 참조하십시오.

콘솔 또는 CLI를 사용하여 트리거를 지정합니다.

다음과 같이 필터 유형을 지정합니다.

• 필터 없음

이 트리거 구성은 작업 구성의 일부로 지정된 기본 브랜치로 푸시할 때마다 파이프라인을 시작합니 다.

• 필터 지정

코드 푸시의 브랜치 이름과 같은 특정 필터에서 파이프라인을 시작하고 정확한 커밋을 가져오는 필 터를 추가합니다. 또한 변경 시 파이프라인이 자동으로 시작되지 않도록 구성합니다.

- 푸시
 - 유효한 필터 조합은 다음과 같습니다.
 - Git 태그

포함 또는 제외

브랜치

포함 또는 제외

• 브랜치 + 파일 경로

포함 또는 제외

- 풀 요청
 - 유효한 필터 조합은 다음과 같습니다.
 - 브랜치

포함 또는 제외

• 브랜치 + 파일 경로

포함 또는 제외

• 변경 사항을 감지하지 않음

이렇게 해도 트리거가 추가되지 않으며 파이프라인은 변경 시 자동으로 시작되지 않습니다.

다음 표에서는 각 이벤트 유형에 유효한 필터 옵션을 제공합니다. 이 표에는 작업 구성에서 자동 변경 감지를 위해 기본적으로 true 또는 false로 설정된 트리거 구성도 나와 있습니다.

트리거 구성	이벤트 유형	필터 옵션	변경 사항 감지
트리거 추가 - 필터 없 음	없음	없음	true
트리거 추가 - 코드 푸 시 필터링	푸시 이벤트	Git 태그, 브랜치, 파일 경로	false
트리거 추가 - 풀 요청 필터링	풀 요청	브랜치, 파일 경로	false
트리거 없음 - 감지하 지 않음	없음	없음	false

1 Note

이 트리거 유형은 자동 변경 감지를 사용합니다(Webhook 트리거 유형으로). 이 트리거 유형을 사용하는 소스 작업 제공자는 코드 푸시를 위해 구성된 연결(Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com, 및 GitLab 자체 관리형)입니다.

트리거에 대한 필드 정의 및 추가 참조는 섹션을 참조하세요.

JSON 구조의 필드 정의 목록은 triggers 섹션을 참조하세요.

필터링의 경우 glob 형식의 정규식 패턴은 <u>구문에서 glob 패턴 작업</u>에 설명된 대로 지원됩니다.

Note

경우에 따라 파일 경로에서 필터링되는 트리거가 있는 파이프라인의 경우 파일 경로 필터가 있 는 브랜치가 처음 생성될 때 파이프라인이 시작되지 않을 수 있습니다. 자세한 내용은 <u>파일 경</u> 로별 트리거 필터링을 사용하는 연결이 있는 파이프라인은 브랜치 생성 시 시작되지 않을 수 있습니다. 단원을 참조하십시오.

트리거 필터 고려 사항

트리거 사용 시 다음 사항을 고려하세요.

- 소스 작업당 트리거를 두 개 이상 추가할 수 없습니다.
- 트리거에 여러 필터 유형을 추가할 수 있습니다. 예시는 <u>4: 충돌하는 두 개의 푸시 필터 유형이 있는</u> <u>트리거에는가 포함되고 제외됩니다.</u>에서 확인하십시오.
- 브랜치 및 파일 경로 필터가 있는 트리거의 경우 브랜치를 처음 푸시할 때 새로 생성된 브랜치에 대 해 변경된 파일 목록에 액세스할 수 없으므로 파이프라인이 실행되지 않습니다.
- 풀 요청을 병합하면 푸시(브랜치 필터) 및 풀 요청(브랜치 필터) 트리거 구성이 교차하는 경우 두 개 의 파이프라인 실행이 트리거될 수 있습니다.
- 풀 요청 이벤트에서 파이프라인을 트리거하는 필터의 경우 닫힌 풀 요청 이벤트 유형의 경우 연결의 타사 리포지토리 공급자가 병합 이벤트에 대해 별도의 상태를 가질 수 있습니다. 예를 들어 Bitbucket 에서 병합에 대한 Git 이벤트는 풀 요청 종료 이벤트가 아닙니다. 그러나 GitHub에서 풀 요청을 병합 하는 것은 종료 이벤트입니다. 자세한 내용은 <u>공급자별 트리거에 대한 풀 요청 이벤트</u> 단원을 참조하 십시오.

공급자별 트리거에 대한 풀 요청 이벤트

다음 표에는 풀 요청 종료와 같이 공급자별로 풀 요청 이벤트 유형을 생성하는 Git 이벤트의 요약이 나 와 있습니다.

	연결을 위한 리포지토리 공급자				
트리거에 대한 PR 이벤트	Bitbucket	GitHub	GHES	GitLab	
열기 -이 옵션은 브랜치/파일 경로 에 대한 풀 요청 이 생성될 때 파 이프라인을 트리 거합니다.	풀 요청을 생성하 면 Git 이벤트가 열립니다.	풀 요청을 생성하 면 Git 이벤트가 열립니다.	풀 요청을 생성하 면 Git 이벤트가 열립니다.	풀 요청을 생성하 면 열린 Git 이벤 트가 발생합니다.	
업데이트 -이 옵 션은 브랜치/파일 경로에 대한 풀 요청 개정이 게시 될 때 파이프라인 을 트리거합니다.	업데이트를 게시 하면 업데이트된 Git 이벤트가 발 생합니다.	업데이트를 게시 하면 업데이트된 Git 이벤트가 발 생합니다.	업데이트를 게시 하면 업데이트된 Git 이벤트가 발 생합니다.	업데이트를 게시 하면 업데이트된 Git 이벤트가 발 생합니다.	
Closed -이 옵션 은 브랜치/파일 경로에 대한 풀 요청이 닫힐 때 파이프라인을 트 리거합니다.	Bitbucket에서 풀 요청을 병합하면 닫힌 Git 이벤트 가 발생합니다. 중요: 병합하지 않고 Bitbucket 에서 풀 요청을 수동으로 닫으면 닫힌 Git 이벤트 가 발생하지 않습 니다.	풀 요청을 병합하 거나 수동으로 닫 으면 닫힌 Git 이 벤트가 발생합니 다.	풀 요청을 병합하 거나 수동으로 닫 으면 닫힌 Git 이 벤트가 발생합니 다.	풀 요청을 병합하 거나 수동으로 닫 으면 닫힌 Git 이 벤트가 발생합니 다.	

트리거 필터의 예

푸시 및 풀 요청 이벤트 유형에 대한 필터가 있는 Git 구성의 경우 지정된 필터가 서로 충돌할 수 있습니 다. 다음은 푸시 및 풀 요청 이벤트에 유효한 필터 조합의 예입니다. 트리거에는 트리거 구성의 두 푸시 필터 유형과 같은 여러 필터 유형이 포함될 수 있으며, 푸시 및 풀 요청 필터 유형은 둘 사이의 OR 작업

을 사용합니다. 즉, 일치하는 모든가 파이프라인을 시작합니다. 마찬가지로 각 필터 유형에는 filePaths 및 브랜치와 같은 여러 필터가 포함될 수 있습니다. 이러한 필터는 AND 작업을 사용합니다. 즉, 전체 일치 항목만 파이프라인을 시작합니다. 각 필터 유형에는 포함 및 제외가 포함될 수 있으며, 이러한 필 터 유형은 이들 간에 AND 작업을 사용합니다. 즉, 전체 일치 항목만 파이프라인을 시작합니다. 브랜치 이름과 같은 포함/제외 내의 이름은 OR 작업을 사용합니다. 브main랜치를 포함하고 제외하는와 같은 두 푸시 필터 간에 충돌이 있는 경우 기본값은 제외입니다. 다음 목록에는 Git 구성 객체의 각 부분에 대 한 작업이 요약되어 있습니다.

JSON 구조의 필드 정의 목록과 포함 및 제외에 대한 자세한 참조는 섹션을 참조하세요triggers.

Example 1: 브랜치 및 파일 경로에 대한 필터가 있는 필터 유형(AND 작업)

풀 요청과 같은 단일 필터 유형의 경우 필터를 결합할 수 있으며, 이러한 필터는 AND 작업을 사용 합니다. 즉, 전체 일치 항목만 파이프라인을 시작합니다. 다음 예제는 두 개의 필터(filePaths 및) 가 있는 푸시 이벤트 유형에 대한 Git 구성을 보여줍니다branches. 다음 예제에서는 filePaths가 branches와 함께 AND가 됩니다.

```
{
   "filePaths": {
     "includes": ["common/**/*.js"]
   },
   "branches": {
     "includes": ["feature/**"]
   }
}
```

위의 Git 구성으로 이 예제에서는 AND 작업이 성공하므로 파이프라인 실행이 시작되는 이벤트를 보여 줍니다. 즉, 지정된 common/app.js 브랜치에 영향이 refs/heads/feature/triggers 없더라 도 파이프라인을 AND로 시작하는 필터에 파일 경로가 포함됩니다.

```
{
   "ref": "refs/heads/feature/triggers",
   ...
   "commits": [
      {
        ...
        "modified": [
           "common/app.js"
      ]
        ...
   }
```

}

]

다음 예제는 브랜치가 필터링할 수 있지만 파일 경로는 필터링할 수 없기 때문에 파이프라인 실행을 시 작하지 않는 위 구성의 트리거 이벤트를 보여줍니다.

Example 2: AND 작업의 사용 포함 및 제외

단일 풀 요청 이벤트 유형의 브랜치와 같은 트리거 필터는 포함과 제외 사이에 AND 작업을 사용합니 다. 이렇게 하면 동일한 파이프라인에 대한 실행을 시작하도록 여러 트리거를 구성할 수 있습니다. 다 음 예제는 푸시 이벤트에 대한 구성 객체에 단일 필터 유형(branches)이 있는 트리거 구성을 보여 줍니다. Includes 및 Excludes 작업은 AND로 표시됩니다. 즉, 브랜치가 제외 패턴(예: featurebranch 예제)과 일치하면 포함도 일치하지 않는 한 파이프라인이 트리거되지 않습니다. main 브랜치 와 같은 포함 패턴이 일치하면 파이프라인이 트리거됩니다.

다음 예제 JSON의 경우:

- main 브랜치에 커밋을 푸시하면 파이프라인이 트리거됩니다.
- feature-branch 브랜치에 커밋을 푸시해도 파이프라인이 트리거되지 않습니다.

```
{
    "branches": {
        "Includes": [
            "main"
        ],
        "Excludes": [
```

]

"feature-branch"

}

Example 3: 푸시 및 풀 요청 필터 유형(OR 작업), 파일 경로 및 브 랜치용 필터(AND 작업), 포함/제외(AND 작업)가 있는 트리거

푸시 이벤트 유형과 풀 요청 이벤트 유형이 포함된 트리거와 같은 트리거 구성 객체는 두 이벤트 유 형 간에 OR 작업을 사용합니다. 다음 예제에서는 main브랜치가 포함된 푸시 이벤트 유형과 동일한 브랜치가 main 제외된 풀 요청 이벤트 유형 하나가 있는 트리거 구성을 보여줍니다. 또한 푸시 이 벤트 유형에는 파일 경로 하나가 LICENSE.txt 제외되고 파일 경로 하나가 README.MD 포함됩니 다. 두 번째 이벤트 유형의 경우 feature-branch브랜치(포함됨)Created에 있는 Closed 풀 요청 이 파이프라인을 시작하고feature-branch-2, 또는 main브랜치(제외됨)에서 풀 요청을 생성하거 나 닫을 때 파이프라인이 시작되지 않습니다.Includes 및 Excludes 작업은 AND'd이며 충돌은 기 본적으로 제외로 설정됩니다. 예를 들어 feature-branch브랜치가 푸시 이벤트 유형에 대해 제외 된 동안 브랜feature-branch치의 풀 요청 이벤트(풀 요청에 포함됨)의 경우 기본값은 제외입니다.

다음 예제에서는

README . MD 파일 경로(포함됨)의 main브랜치(포함 됨)에 커밋을 푸시하면 파이프라인이 트리거됩니다.

- feature-branch 브랜치(제외됨)에서 커밋을 푸시해도 파이프라인이 트리거되지 않습니다.
- 포함된 브랜치에서 README.MD 파일 경로(포함됨)를 편집하면 파이프라인이 트리거됩니다.
- 포함된 브랜치에서 LICENSE . TXT 파일 경로(제외 됨)를 편집해도 파이프라인이 트리거되지 않습니다.
- feature-branch 브랜치에서 README.MD 파일 경로(푸시 이벤트에 포함)에 대
 한 풀 요청을 닫으면 푸시 이벤트 유형이 feature-branch브랜치를 제외로 지정
 하므로 파이프라인이 트리거되지 않으므로 충돌은 기본적으로 제외로 설정됩니다.

다음 이지미는 해당 구성을 보여줍니다.

dit: Triggers	lit: Triggers				
For source action: Source					
Filters					
Push Include branches: main Exclude branches: feature-branch, feature-b IIII Include file paths: README.md	(i)	Pull request Events: Closed Created Include branches: feature-branch Exclude branches: feature-branch-2, main	1		
Exclude file paths: LICENSE.txt					

다음은 구성에 대한 JSON 예제입니다.

```
"triggers": [
            {
                "providerType": "CodeStarSourceConnection",
                "gitConfiguration": {
                    "sourceActionName": "Source",
                    "push": [
                        {
                             "branches": {
                                 "includes": [
                                     "main"
                                 ],
                                 "excludes": [
                                     "feature-branch",
                                     "feature-branch-2"
                                 ]
                            },
                             "filePaths": {
                                 "includes": [
                                     "README.md"
                                 ],
                                 "excludes": [
                                     "LICENSE.txt"
                                 ]
                            }
                        }
                    ],
                    "pullRequest": [
                        {
                             "events": [
```

```
"CLOSED",
                              "OPEN"
                          ],
                          "branches": {
                              "includes": [
                                   "feature-branch"
                              ],
                              "excludes": [
                                   "feature-branch-2",
                                   "main"
                              ]
                          }
                     }
                 ]
             }
        }
    ]
},
```

Example 4: 충돌하는 두 개의 푸시 필터 유형이 있는 트리거에는가 포함되고 제외됩니다.

다음 이미지는 태그release-1(포함)를 기준으로 필터링하도록 지정하는 푸시 필 터 유형을 보여줍니다. 두 번째 푸시 필터 유형은 브랜치를 필터링하고main(포 함) feature*브랜치에 대한 푸시를 시작하지 않도록(제외) 지정하는를 추가합니다.

추가 예제:

feature-branch 브랜치의 태그release-1(첫 번째 푸시 필터에 포 함됨)에서 릴리스를 푸시하면(두 번째 푸시 필터feature*에 로 제외됨) 두 이벤트 유형이 AND'd이기 때문에 파이프라인이 트리거되지 않습니다.

• main 브랜치에서 릴리스를 푸시하면(두 번째 푸시 필터에 포함됨) 파이프라인이 시작됩니다.

편집 페이지의 다음 예제에서는 두 푸시 필터 유형과에 대한 구성에 포함 및 제외가 표시됩니다.

dit: Triggers					Cancel Done
For source action: Source					Remove
Filters					
Push Include tags: release-1	0 ×	Push Include branches: main* Exclude branches: feature*	0 Ø X	+ Add filter	

다음은 구성에 대한 JSON 예제입니다.



Example 5: 기본 작업 구성인 BranchName을 수동 시작에 사용하는 동안 구성된 트리거

작업 구성 기본 BranchName 필드는 파이프라인을 수동으로 시작할 때 사용할 단일 브 랜치를 정의하는 반면, 필터가 있는 트리거는 지정한 모든 브랜치에 사용할 수 있습니다.

다음은 BranchName 필드를 보여주는 작업 구성의 예제 JSON입니다.

```
{
                "name": "Source",
                "actions": [
                    {
                         "name": "Source",
                         "actionTypeId": {
                             "category": "Source",
                             "owner": "AWS",
                             "provider": "CodeStarSourceConnection",
                             "version": "1"
                         },
                         "runOrder": 1,
                         "configuration": {
                             "BranchName": "main",
                             "ConnectionArn": "ARN",
                             "DetectChanges": "false",
                             "FullRepositoryId": "owner-name/my-bitbucket-repo",
                             "OutputArtifactFormat": "CODE_ZIP"
                         },
                         "outputArtifacts": [
                             {
                                 "name": "SourceArtifact"
                             }
                         ],
                         "inputArtifacts": [],
                         "region": "us-west-2",
                         "namespace": "SourceVariables"
                    }
                ],
```

다음 예제 작업 출력은 파이프라인이 수동으로 시작될 때 사용된 기본 브랜치 기본을 보여줍니다.

Action execution details × Action name: Source Status: Succeeded					
Summary Inpu	nt Output				
Output artifact	SourceArtifact 🖪				
AuthorDate	2024-11-08T00:16:03Z				
AuthorDisplayName					
AuthorEmail	r@amazon.com				
Authorid					
BranchName	main				
CommitId	e87b1678ec5c50b2addf20f213cc7				
CommitMessage	Dockerfile created online with Bitbucket				
ConnectionArn	arn:aws:codestar-connections:us-west-2: connection/cdacd948-8633-4409-a4e f-				
FullRepositoryName	'dk-bitbucket-repo				
ProviderType	Bitbucket				
	Done				

다음 예제 작업 출력은 풀 요청으로 필터링될 때 트리거에 사용된 풀 요청 및 브랜치를 보여줍니다.

ction execution deta ction name: Source Statu:	ils s: Succeeded	×
Summary Input	Output	
Output artifact	SourceArtifact 🔼	
AuthorDate	2024-10-23T19:12:43Z	
AuthorDisplayName		
AuthorEmail	@amazon.com	
Authorid	{c26d1f33-2013-46e8-b193 }	
CommitId	32b96d37d12c53680a16029cc3	
CommitMessage	README.md edited online with Bitbucket	
ConnectionArn	arn:aws:codestar-connections:us-west-2: -a4	tion/cdacd948-8633-4409
DestinationBranchName	feature-branch	
FullRepositoryName	/dk-bitbucket-repo	
ProviderType	Bitbucket	
PullRequestId	6	
PullRequestTitle	Feature branch 2	
SourceBranchName	feature-branch-2	
		Done

필터 없이 코드 푸시 시 트리거 추가

트리거를 사용하면 코드 푸시 또는 풀 요청과 같은 특정 이벤트 유형에서 시작하도록 파이 프라인을 구성할 수 있습니다. 트리거는 GitHub, Bitbucket, GitLab과 같이 CodePipeline CodeStarSourceConnection 작업을 사용하는 연결을 통해 소스 작업에 대해 구성할 수 있습니다.

트리거 추가(콘솔)

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> <u>codepipeline/home</u>://https//https://https://https://https://https://https://https://https://https://https://https://https://https://https://https://https://https://https://https://https//http

AWS 계정과 연결된 모든 파이프라인의 이름과 상태가 표시됩니다.

- [Name]에서 편집할 파이프라인의 이름을 선택합니다. 그렇지 않으면 파이프라인 생성 마법사에서 다음 단계를 사용합니다.
- 3. 파이프라인 세부 정보 페이지에서 [Edit]를 선택합니다.
- 편집 페이지에서 편집하려는 소스 작업을 선택합니다. 트리거 편집을 선택합니다. 트리거를 추가 하려면를 선택합니다.
- 5. 트리거 유형에서 필터 없음을 선택합니다.

코드 푸시 또는 풀 요청 이벤트 유형을 사용하여 트리거 추가

태그 또는 브랜치 푸시, 특정 파일 경로 변경, 특정 브랜치로 열린 풀 요청 등과 같은 다양한 Git 이벤트 에 대해 파이프라인 실행을 시작하도록 파이프라인 트리거에 대한 필터를 구성할 수 있습니다. AWS CodePipeline 콘솔 또는의 create-pipeline 및 update-pipeline 명령을 사용하여 트리거 필터를 AWS CLI 구성할 수 있습니다.

Note

작업 구성 BranchName 필드는 단일 브랜치를 정의하는 반면 필터가 있는 트리거는 지정한 모 든 브랜치에 사용할 수 있습니다. 트리거를 사용하여 푸시 또는 풀 요청으로 브랜치를 필터링 하는 파이프라인의 경우 파이프라인은 작업 구성에서 기본 BranchName 필드 브랜치를 사용 하지 않습니다. 그러나 파이프라인을 수동으로 시작할 때 작업 구성의 BranchName 필드에 있 는 브랜치는 기본값입니다. 예시는 <u>5: 기본 작업 구성인 BranchName을 수동 시작에 사용하는</u> <u>동안 구성된 트리거</u>에서 확인하십시오.

다음 트리거 유형에 대한 필터를 지정할 수 있습니다.

• 푸시

푸시 트리거는 변경 사항이 소스 리포지토리로 푸시될 때 파이프라인을 시작합니다. 실행은 푸시하 려는 브랜치(즉, 대상 브랜치)의 커밋을 사용합니다. 브랜치, 파일 경로 또는 Git 태그에서 푸시 트리 거를 필터링할 수 있습니다.

• 풀 요청

풀 요청 트리거는 소스 리포지토리에서 풀 요청이 열리거나 업데이트되거나 닫힐 때 파이프라인을 시작합니다. 실행은 가져오려는 소스 브랜치(즉, 소스 브랜치)에서 커밋을 사용합니다. 브랜치 및 파 일 경로에서 풀 요청 트리거를 필터링할 수 있습니다.

풀 요청에 지원되는 이벤트 유형은 다음과 같습니다. 다른 모든 풀 요청 이벤트는 무시됩니다.

- Opened(열어봄)
- Updated
- 종료됨(병합됨)

Note

특정 풀 요청 이벤트 동작은 공급자마다 다를 수 있습니다. 세부 정보는 <u>공급자별 트리거에</u> 대한 풀 요청 이벤트을 참조하세요.

파이프라인 정의를 사용하면 동일한 푸시 트리거 구성 내에서 서로 다른 필터를 결합할 수 있습니다. 파이프라인 정의에 대한 자세한 내용은 <u>푸시 및 풀 요청 이벤트 유형에 대한 필터 추가(CLI)</u> 섹션을 참 조하세요. 필드 정의 목록은이 가이드의 파이프라인 구조 참조에서 트리거를 참조하세요.

주제

- 푸시 및 풀 요청 이벤트 유형에 대한 필터 추가(콘솔)
- <u>푸시 및 풀 요청 이벤트 유형에 대한 필터 추가(CLI)</u>
- 푸시 및 풀 요청 이벤트 유형에 대한 필터 추가(AWS CloudFormation 템플릿)

푸시 및 풀 요청 이벤트 유형에 대한 필터 추가(콘솔)

콘솔을 사용하여 푸시 이벤트에 대한 필터를 추가하고 브랜치 또는 파일 경로를 포함하거나 제외할 수 있습니다.

푸시 및 풀 요청 이벤트 유형에 대한 필터 추가(콘솔)

필터 추가(콘솔)

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https//https/

AWS 계정과 연결된 모든 파이프라인의 이름과 상태가 표시됩니다.

- [Name]에서 편집할 파이프라인의 이름을 선택합니다. 그렇지 않으면 파이프라인 생성 마법사에서 다음 단계를 사용합니다.
- 3. 파이프라인 세부 정보 페이지에서 [Edit]를 선택합니다.
- 편집 페이지에서 편집하려는 소스 작업을 선택합니다. 트리거 편집을 선택합니다. 필터 지정을 선 택합니다.
- 5. 이벤트 유형의 다음 옵션에서 푸시를 선택합니다.
 - 변경을 소스 리포지토리로 푸시할 때 파이프라인을 시작하려면 푸시를 선택합니다. 이 옵션을 선택하면 필드에서 브랜치 및 파일 경로 또는 Git 태그에 대한 필터를 지정할 수 있습니다.
 - 소스 리포지토리에서 풀 요청이 열리거나, 업데이트되거나, 닫힐 때 파이프라인을 시작하려면 풀 요청을 선택합니다. 이 옵션을 선택하면 필드에서 대상 브랜치 및 파일 경로에 대한 필터를 지정할 수 있습니다.
- 6. 푸시의 필터 유형에서 다음 옵션 중 하나를 선택합니다.
 - 워크플로 실행을 시작할 시기를 알아보기 위해 브랜치를 선택하여 트리거가 모니터링하는 소스 리포지토리의 브랜치를 지정합니다. 포함에서 지정된 브랜치 변경 시 파이프라인을 시작하도록 트리거 구성에 지정할 브랜치 이름 패턴을 glob 형식으로 입력합니다. 제외에서 지정된 브랜치 변경 시 파이프라인을 무시하고 시작하지 않도록 트리거 구성에 지정할 브랜치 이름의 정규식 패턴을 glob 형식으로 입력합니다. 자세한 내용은 구문에서 glob 패턴 작업 섹션을 참조하세요.

Note

포함과 제외 항목 모두 동일한 패턴을 갖는 경우 기본값은 패턴을 제외하는 것입니다.

glob 패턴을 사용하여 브랜치 이름을 정의할 수 있습니다. 예를 들어 main*을 사용하여 main으 로 시작하는 모든 브랜치를 일치시킵니다. 자세한 내용은 <u>구문에서 glob 패턴 작업</u> 섹션을 참조 하세요.

푸시 트리거의 경우 푸시할 브랜치, 즉 대상 브랜치를 지정합니다. 풀 요청 트리거의 경우 풀 요 청을 열고 있는 대상 브랜치를 지정합니다. • (선택 사항) 파일 경로에서 트리거의 파일 경로를 지정합니다. 포함 및 제외에 이름을 알맞게 입 력합니다.

glob 패턴을 사용하여 파일 경로 이름을 정의할 수 있습니다. 예를 들어 prod*을 사용하여 prod으로 시작하는 모든 파일 경로를 일치시킵니다. 자세한 내용은 <u>구문에서 glob 패턴 작업</u> 섹 션을 참조하세요.

- Git 태그로 시작하도록 파이프라인 트리거 구성을 구성하려면 태그를 선택합니다. 포함에서 지 정된 태그가 릴리스되면 파이프라인을 시작하도록 트리거 구성에 지정할 태그 이름 패턴을 glob 형식으로 입력합니다. 제외에서 지정된 태그가 릴리스되면 파이프라인을 무시하거나 시작하지 않도록 트리거 구성에 지정할 태그 이름의 정규식 패턴을 glob 형식으로 입력합니다. 포함과 제 외 항목 모두 동일한 태그 패턴을 갖는 경우 기본값은 태그 패턴을 제외하는 것입니다.
- 7. 푸시의 필터 유형에서 다음 옵션 중 하나를 선택합니다.
 - 워크플로 실행을 시작할 시기를 알아보기 위해 브랜치를 선택하여 트리거가 모니터링하는 소스 리포지토리의 브랜치를 지정합니다. 포함에서 지정된 브랜치 변경 시 파이프라인을 시작하도록 트리거 구성에 지정할 브랜치 이름 패턴을 glob 형식으로 입력합니다. 제외에서 지정된 브랜치 변경 시 파이프라인을 무시하고 시작하지 않도록 트리거 구성에 지정할 브랜치 이름의 정규식 패턴을 glob 형식으로 입력합니다. 자세한 내용은 구문에서 glob 패턴 작업 섹션을 참조하세요.

Note

포함과 제외 항목 모두 동일한 패턴을 갖는 경우 기본값은 패턴을 제외하는 것입니다.

glob 패턴을 사용하여 브랜치 이름을 정의할 수 있습니다. 예를 들어 main*을 사용하여 main으 로 시작하는 모든 브랜치를 일치시킵니다. 자세한 내용은 <u>구문에서 glob 패턴 작업</u> 섹션을 참조 하세요.

푸시 트리거의 경우 푸시할 브랜치, 즉 대상 브랜치를 지정합니다. 풀 요청 트리거의 경우 풀 요 청을 열고 있는 대상 브랜치를 지정합니다.

• (선택 사항) 파일 경로에서 트리거의 파일 경로를 지정합니다. 포함 및 제외에 이름을 알맞게 입 력합니다.

glob 패턴을 사용하여 파일 경로 이름을 정의할 수 있습니다. 예를 들어 prod*을 사용하여 prod으로 시작하는 모든 파일 경로를 일치시킵니다. 자세한 내용은 <u>구문에서 glob 패턴 작업</u> 섹 션을 참조하세요. 풀 요청을 선택하여 지정한 풀 요청 이벤트로 시작하도록 파이프라인 트리거 구성을 구성합니다.

푸시 및 풀 요청 이벤트 유형에 대한 필터 추가(CLI)

파이프라인 JSON을 업데이트하여 트리거 필터를 추가할 수 있습니다.

를 사용하여 파이프라인 AWS CLI 을 생성하거나 업데이트하려면 create-pipeline 또는 updatepipeline 명령을 사용합니다.

다음 예제 JSON 구조는 create-pipeline 아래의 필드 정의에 대한 참조를 제공합니다.

필드 정의 목록은이 가이드의 파이프라인 구조 참조에서 트리거를 참조하세요.

```
{
    "pipeline": {
        "name": "MyServicePipeline",
        "triggers": [
            {
                "provider": "Connection",
                 "gitConfiguration": {
                     "sourceActionName": "ApplicationSource",
                     "push": [
                         {
                             "filePaths": {
                                 "includes": [
                                      "projectA/**",
                                      "common/**/*.js"
                                 ],
                                 "excludes": [
                                      "**/README.md",
                                      "**/LICENSE",
                                      "**/CONTRIBUTING.md"
                                 ]
                             },
                             "branches": {
                                 "includes": [
                                      "feature/**",
                                      "release/**"
                                 ],
                                 "excludes": [
                                      "mainline"
```

```
]
                     },
                     "tags": {
                         "includes": [
                             "release-v0", "release-v1"
                         ],
                         "excludes": [
                             "release-v2"
                         ]
                    }
                }
            ],
            "pullRequest": [
                {
                     "events": [
                         "CLOSED"
                     ],
                     "branches": {
                         "includes": [
                             "feature/**",
                             "release/**"
                         ],
                         "excludes": [
                             "mainline"
                         ]
                    },
                     "filePaths": {
                         "includes": [
                             "projectA/**",
                             "common/**/*.js"
                         ],
                         "excludes": [
                             "**/README.md",
                             "**/LICENSE",
                             "**/CONTRIBUTING.md"
                         ]
                    }
                }
            ]
        }
   }
],
"stages": [
    {
```



푸시 및 풀 요청 이벤트 유형에 대한 필터 추가(AWS CloudFormation 템플 릿)

에서 파이프라인 리소스를 업데이트 AWS CloudFormation 하여 트리거 필터링을 추가할 수 있습니다.

다음 예제 템플릿 조각은 트리거 필드 정의에 대한 YAML 참조를 제공합니다. 필드 정의 목록은이 가이 드의 파이프라인 구조 참조에서 트리거를 참조하세요.

연결 소스 및 트리거 필터 구성에 대한 전체 템플릿 예제는 AWS CloudFormation 사용 설명서의 Pipeline with two stage and trigger configuration을 참조하세요.

```
- '**/README.md'
                - '**/LICENSE'
                - '**/CONTRIBUTING.md'
            branches:
              includes:
                - feature/**
                - release/**
              excludes:
                - mainline
          - tags:
              includes:
                - release-v0
                - release-v1
              excludes:
                - release-v2
        pullRequest:
          - events:
              - CLOSED
            branches:
              includes:
                - feature/**
                - release/**
              excludes:
                - mainline
            filePaths:
              includes:
                - projectA/**
                - common/**/*.js
              excludes:
                - '**/README.md'
                - '**/LICENSE'
                - '**/CONTRIBUTING.md'
  stages:
    - name: Source
      actions:
        - name: ApplicationSource
          configuration:
            BranchName: mainline
            ConnectionArn: arn:aws:codestar-connections:eu-
central-1:111122223333:connection/fe9ff2e8-ee25-40c9-829e-65f85EXAMPLE
            FullRepositoryId: monorepo-example
            OutputArtifactFormat: CODE_ZIP
```

변경 감지를 끄는 트리거 추가

트리거를 사용하면 코드 푸시 또는 풀 요청과 같은 특정 이벤트 유형에서 시작하도록 파이프 라인을 구성할 수 있습니다. 트리거는 GitHub, Bitbucket 및 GitLab과 같이 CodePipeline에서 CodeStarSourceConnection 작업을 사용하는 연결을 통해 소스 작업에 대해 구성할 수 있습니다.

변경 감지를 끄는 트리거 추가(콘솔)

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> <u>codepipeline/home</u>://https//https/

AWS 계정과 연결된 모든 파이프라인의 이름과 상태가 표시됩니다.

- 2. [Name]에서 편집할 파이프라인의 이름을 선택합니다. 그렇지 않으면 파이프라인 생성 마법사에서 다음 단계를 사용합니다.
- 3. 파이프라인 세부 정보 페이지에서 [Edit]를 선택합니다.
- 편집 페이지에서 편집하려는 소스 작업을 선택합니다. 트리거 편집을 선택합니다. 트리거를 추가 하려면를 선택합니다.
- 5. 트리거 유형에서 변경 사항을 감지하지 않음을 선택합니다.

파이프라인 수동 시작 및 중지

AWS CodePipeline 콘솔 또는를 사용하여 파이프라인을 수동으로 시작하고 중지 AWS CLI 할 수 있습 니다. 다음에 설명된 대로 이벤트 기반 변경 감지를 사용하거나 설명된 대로 구성을 트리거하여 파이프 라인을 시작하는 자동화된 방법이 있습니다.

주제

- CodePipeline에서 파이프라인 시작
- CodePipeline에서 파이프라인 실행 중지

CodePipeline에서 파이프라인 시작

각 파이프라인 실행은 다른 트리거를 기반으로 시작할 수 있습니다. 파이프라인 시작 방식에 따라 각 파이프라인 실행에 다른 유형의 트리거가 있을 수 있습니다. 각 실행의 트리거 유형은 파이프라인의 실 행 기록에 표시됩니다. 트리거 유형은 다음과 같이 소스 작업 제공자에 따라 달라질 수 있습니다.

Note

소스 작업당 트리거를 한 개 이상 지정할 수 없습니다.

- 파이프라인 생성: 파이프라인이 생성되면 파이프라인 실행이 자동으로 시작됩니다. 이는 실행 내 역의 CreatePipeline 트리거 유형입니다.
- 수정된 객체의 변경 사항: 이 범주는 실행 내역의 PutActionRevision 트리거 유형을 나타냅니다.
- 브랜치에서의 변경 감지 및 코드 푸시에 대한 커밋: 이 범주는 실행 내역의 CloudWatchEvent 트리 거 유형을 나타냅니다. 소스 리포지토리의 소스 커밋 및 브랜치에 대한 변경이 감지되면 파이프라인 이 시작됩니다. 이 트리거 유형은 자동 변경 감지를 사용합니다. 이 트리거 유형을 사용하는 소스 작 업 제공자는 S3와 CodeCommit입니다. 이 유형은 파이프라인을 시작하는 일정에도 사용됩니다. 일 정에 따라 파이프라인 시작을 참조하세요.
- 소스 변경에 대한 폴링: 이 카테고리는 실행 내역의 PollForSourceChanges 트리거 유형을 나타 냅니다. 폴링을 통한 소스 리포지토리의 소스 커밋 및 브랜치에 대한 변경이 감지되면 파이프라인이 시작됩니다. 이 트리거 유형은 권장되지 않으므로 자동 변경 감지를 사용하도록 마이그레이션해야 합니다. 이 트리거 유형을 사용하는 소스 작업 제공자는 S3와 CodeCommit입니다.
- 타사 소스에 대한 Webhook 이벤트: 이 카테고리는 실행 내역의 Webhook 트리거 유형을 나타냅니다. Webhook 이벤트로 변경이 감지되면 파이프라인이 시작됩니다. 이 트리거 유형은 자동 변경
감지를 사용합니다. 이 트리거 유형을 사용하는 소스 작업 제공자는 코드 푸시를 위해 구성된 연결 (Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com, 및 GitLab 자체 관리형)입니다.

- 타사 소스에 대한 WebhookV2 이벤트: 이 카테고리는 실행 내역의 WebhookV2 트리거 유형을 나타 냅니다. 이 유형은 파이프라인 정의에 정의된 트리거를 기반으로 트리거되는 실행에 사용됩니다. 지 정된 Git 태그가 있는 릴리스가 감지되면 파이프라인이 시작됩니다. Git 태그를 사용하여 다른 리포 지토리 사용자가 중요성을 파악할 수 있도록 커밋을 이름이나 다른 식별자로 표시할 수 있습니다. 또 한 Git 태그를 사용하여 리포지토리의 기록에서 특정 커밋을 식별할 수 있습니다. 이 트리거 유형은 자동 변경 감지를 비활성화합니다. 이 트리거 유형을 사용하는 소스 작업 제공자는 Git 태그를 위해 구성된 연결(Bitbucket Cloud, GitHub, GitHub Enterprise Server 및 GitLab.com)입니다.
- 파이프라인 수동 시작: 이 범주는 실행 내역의 StartPipelineExecution 트리거 유형을 나타냅니다. 콘솔 또는 AWS CLI 를 사용하여 파이프라인을 수동으로 시작할 수 있습니다. 자세한 내용은 수동으로 파이프라인 시작을 참조하세요.
- RollbackStage: 이 범주는 실행 내역의 RollbackStage 트리거 유형을 나타냅니다. 콘솔 또는를 사용하여 스테이지를 수동 또는 자동으로 롤백 AWS CLI 할 수 있습니다. 자세한 내용은 <u>스테이지 롤백</u> <u>구성</u>을 참조하세요.

자동화된 변경 감지 트리거 유형을 사용하는 소스 작업을 파이프라인에 추가하면 작업이 추가 리소스 와 함께 작동합니다. 변경 감지를 위한 이러한 추가 리소스로 인해 각 소스 작업을 생성하는 방법은 별 도의 섹션에 자세히 설명되어 있습니다. 자동화된 변경 감지에 필요한 각 소스 공급자와 변경 감지 방 법에 대한 자세한 내용은 <u>소스 작업 및 변경 감지 방법</u> 섹션을 참조하세요.

주제

- 수동으로 파이프라인 시작
- 일정에 따라 파이프라인 시작
- 소스 개정 재정의로 파이프라인 시작

수동으로 파이프라인 시작

기본적으로 파이프라인은 소스 리포지토리에서 생성된 후 변경이 이루어질 때마다 자동으로 시작됩니다. 하지만 파이프라인을 통해 한 번 더 최신 개정을 다시 실행해야 할 수 있습니다. CodePipeline 콘솔이나 AWS CLI 및 start-pipeline-execution 명령을 사용하여 파이프라인을 통해 최신 개정을 수동으로 다시 실행할 수 있습니다.

주제

• 수동으로 파이프라인 시작(콘솔)

• <u>수동으로 파이프라인 시작(CLI)</u>

수동으로 파이프라인 시작(콘솔)

파이프라인을 수동으로 시작하고 파이프라인을 통해 가장 최근의 개정을 실행하려면

- 1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> <u>codepipeline/home</u>://https//https//ht
- 2. [Name]에서 시작할 파이프라인의 이름을 선택합니다.
- 파이프라인 세부 정보 페이지에서 변경 사항 릴리스를 선택합니다. 파이프라인이 파라미터(파이 프라인 변수)를 전달하도록 구성된 경우 변경 사항 릴리스를 선택하면 변경 사항 릴리스 창이 열립 니다. 파이프라인 변수의 파이프라인 수준에서 필드 또는 변수에 대한 필드에 이 파이프라인 실행 에 전달하려는 값을 하나 이상 입력합니다. 자세한 내용은 변수 참조 단원을 참조하십시오.

이렇게 하면 소스 작업에 지정된 각 소스 위치에서 사용 가능한 가장 최근의 개정이 파이프라인을 통해 시작됩니다.

수동으로 파이프라인 시작(CLI)

파이프라인을 수동으로 시작하고 파이프라인을 통해 가장 최근의 아티팩트 버전을 실행하려면

 터미널(Linux, macOS 또는 Unix) 또는 명령 프롬프트(Windows)를 열고 AWS CLI 를 사용하여 시 작하려는 파이프라인의 이름을 지정하고 start-pipeline-execution 명령을 실행합니다. 예를 들어 MyFirstPipeline이라는 파이프라인을 통해 마지막 변경 사항 실행을 시작하려면 다음과 같이 합니다.

aws codepipeline start-pipeline-execution --name MyFirstPipeline

파이프라인 수준에서 변수가 구성된 파이프라인을 시작하려면 start-pipeline-execution 명령을 선 택적 --variables 인수와 함께 사용하여 파이프라인을 시작하고 실행에 사용할 변수를 추가합니다. 예를 들어, 값이 1인 var1 변수를 추가하려면 다음 명령을 사용하세요.

aws codepipeline start-pipeline-execution --name MyFirstPipeline --variables
name=var1,value=1

2. 성공을 확인하려면 반환된 객체를 봅니다. 이 명령은 다음과 같이 실행 ID 객체를 반환합니다.

{

}

"pipelineExecutionId": "c53dbd42-This-Is-An-Example"

Note

파이프라인을 시작했으면 CodePipeline 콘솔에서 진행 상황을 모니터링하거나 getpipeline-state 명령을 실행합니다. 자세한 내용은 <u>파이프라인 보기(콘솔)</u> 및 <u>파이프라인 세</u> 부 정보 및 이력 보기(CLI) 단원을 참조하세요.

일정에 따라 파이프라인 시작

일정에 따라 파이프라인을 시작하도록 EventBridge의 규칙을 설정할 수 있습니다.

파이프라인 시작을 예약하는 EventBridge 규칙 생성(콘솔)

일정이 이벤트 소스인 EventBridge 규칙을 만들려면

- 1. <u>https://console.aws.amazon.com/events/</u>에서 Amazon EventBridge 콘솔을 엽니다.
- 2. 탐색 창에서 규칙을 선택합니다.
- 3. 규칙 생성을 선택한 후 규칙 세부 정보 아래에서 일정을 선택합니다.
- 고정 비율이나 표현식을 사용하여 일정을 설정합니다. 자세한 내용은 <u>규칙에 대한 일정 표현식</u>을 참조하십시오.
- 5. 대상 영역에서 CodePipeline을 선택합니다.
- 6. 이 일정에 대한 파이프라인 실행의 파이프라인 ARN을 입력합니다.

Note

콘솔의 설정에서 파이프라인 ARN을 찾을 수 있습니다. <u>파이프라인 ARN 및 서비스 역할</u> <u>ARN 보기(콘솔)</u>을 참조하세요.

- 7. 다음 중 하나를 선택하여 EventBridge에 EventBridge 규칙과 연결된 대상을 호출할 권한을 제공하는 IAM 서비스 역할을 만들거나 지정합니다(이 경우에는 대상이 CodePipeline).
 - 이 특정 리소스에 대한 새 역할 생성을 선택하여 EventBridge에 파이프라인 실행을 시작할 권한 을 부여하는 서비스 역할을 생성합니다.

- 기존 역할 사용을 선택하여 EventBridge에 파이프라인 실행을 시작할 권한을 부여하는 서비스 역할을 입력합니다.
- 8. 세부 정보 구성을 선택합니다.
- 규칙 세부 정보 구성 페이지에서 해당 규칙의 이름과 설명을 입력한 후 상태를 선택하여 규칙을 활 성화합니다.
- 10. 규칙이 만족스러우면 규칙 생성(Create rule)을 선택하세요.

파이프라인 시작을 예약하는 EventBridge 규칙 생성(CLI)

AWS CLI 를 사용하여 규칙을 생성하려면 다음을 지정하여 put-rule 명령을 호출합니다.

- 만들려는 규칙을 고유하게 식별하는 이름. 이 이름은 AWS 계정과 연결된 CodePipeline으로 생성하는 모든 파이프라인에서 고유해야 합니다.
- 해당 규칙의 일정 표현식.

일정이 이벤트 소스인 EventBridge 규칙을 만들려면

1. put-rule 명령을 호출하고 --name 및 --schedule-expression 파라미터를 포함시킵니다.

예시:

다음 예제 명령은 --schedule-expression을 사용하여 일정에 따라 EventBridge를 필터링하는 MyRule2라는 규칙을 만듭니다.

```
aws events put-rule --schedule-expression 'cron(15 10 ? * 6L 2002-2005)' --name
MyRule2
```

- 2. CodePipeline을 대상으로 추가하려면 put-targets 명령을 호출하고 다음 파라미터를 포함합니다.
 - --rule 파라미터는 put-rule을 사용하여 생성한 rule_name에 사용됩니다.
 - --targets 파라미터는 대상 목록에 있는 대상의 목록 Id 및 대상 파이프라인의 ARN에 사용됩니다.

다음 예제 명령은 MyCodeCommitRepoRule이라는 규칙에 대해 대상 Id가 숫자 1로 구성됨을 지 정하며, 규칙의 대상 목록에서 1로 대상 1로 표시됩니다. 이 예제 명령은 또한 파이프라인에 대한 예제 ARN를 지정합니다. 파이프라인은 리포지토리에서 변경이 발생하면 시작됩니다.

```
aws events put-targets --rule MyCodeCommitRepoRule --targets
Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
```

- 3. CodePipeline을 사용하여 규칙을 호출하도록 EventBridge에 권한을 부여합니다. 자세한 내용은 Amazon EventBridge에 대한 리소스 기반 정책 사용을 참조하세요.
 - a. 다음 예제를 사용하여 EventBridge가 서비스 역할을 담당하도록 허용하는 신뢰 정책을 생성 합니다. 이름을 trustpolicyforEB.json로 지정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
               "Service": "events.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

b. 다음 명령을 사용하여 Role-for-MyRule 역할을 생성한 후 신뢰 정책에 연결합니다.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
file://trustpolicyforEB.json
```

c. MyFirstPipeline이라는 파이프라인에 대한 이 샘플에 표시된 대로 권한 정책 JSON을 만 듭니다. 권한 정책 이름을 permissionspolicyforEB.json으로 지정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "codepipeline:StartPipelineExecution"
        ],
            "Resource": [
               "arn:aws:codepipeline:us-west-2:80398EXAMPLE:MyFirstPipeline"
        ]
```

] } }

d. 다음 명령을 사용하여 앞에서 생성한 Role-for-MyRule 역할에 새로운 CodePipeline-Permissions-Policy-for-EB 권한 정책을 연결합니다.

aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-Permissions-Policy-For-EB --policy-document file://permissionspolicyforCWE.json

소스 개정 재정의로 파이프라인 시작

재정의를 사용하여 파이프라인 실행을 위해 제공한 특정 소스 개정 ID로 파이프라인을 시작할 수 있습니다. 예를 들어 CodeCommit 소스의 특정 커밋 ID를 처리하는 파이프라인을 시작하려는 경우 파이프 라인을 시작할 때 커밋 ID를 재정의로 추가할 수 있습니다.

1 Note

입력 변환 항목을 사용하여 소스 재정의를 생성하여 파이프라인 이벤트에 EventBridgerevisionValue의를 사용할 수도 있습니다. 여기서 revisionValue는 객체 키, 커밋 또는 이미지 ID에 대한 소스 이벤트 변수에서 파생됩니다. 자세한 내용은 , <u>Amazon ECR 소스 작업 및 EventBridge 리소스 이벤트에 대해 활성화된 소스를 사용하여</u> <u>Amazon S3 소스 작업에 연결</u>또는의 절차에 포함된 입력 변환 항목의 선택적 단계를 참조하세 요CodeCommit 소스 작업 및 EventBridge.

revisionType에 대한 소스 개정에는 네 가지 유형이 있습니다.

- COMMIT_ID
- IMAGE_DIGEST
- S3_OBJECT_VERSION_ID
- S3_OBJECT_KEY

1 Note

COMMIT_ID 및 IMAGE_DIGEST 유형의 소스 개정의 경우 소스 개정 ID는 모든 분기의 리포지 토리 내 모든 컨텐츠에 적용됩니다.

Note

S3_OBJECT_VERSION_ID및 S3_OBJECT_KEY 유형의 소스 개정의 경우, 두 유형 중 하나를 독립적으로 사용하거나 함께 사용하여 특정 ObjectKey 및 VersionID로 소스를 재정의할 수 있습니다. S3_OBJECT_KEY의 경우 구성 파라미터 AllowOverrideForS30bjectKey를 true로 설정해야 합니다. S3 소스 구성 파라미터에 대한 자세한 내용은 <u>구성 파라미터</u> 섹션을 참조하세요.

주제

- 소스 개정 재정의로 파이프라인 시작(콘솔)
- 소스 개정 재정의(CLI)로 파이프라인 시작

소스 개정 재정의로 파이프라인 시작(콘솔)

파이프라인을 수동으로 시작하고 파이프라인을 통해 가장 최근의 개정을 실행하려면

- 1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https//https
- 2. [Name]에서 시작할 파이프라인의 이름을 선택합니다.
- 파이프라인 세부 정보 페이지에서 변경 사항 릴리스를 선택합니다. 릴리스 변경을 선택하면 릴리 스 변경 창이 열립니다. 소스 개정 재정의의 경우 화살표를 선택하여 필드를 확장합니다. 소스에 소스 개정 ID를 입력합니다. 예를 들어 파이프라인에 CodeCommit 소스가 있는 경우 사용하려는 필드에서 커밋 ID를 선택합니다.

Release change	×
Source revision override A source revision is the version with all the changes to your application code, or source artifact for the pipeline execution. Choose the source revision, or version of your source artifact, with changes that you want to run in the pipeline execution.	t, the
Source Commit ID	
Q	
Cancel Releas	e

소스 개정 재정의(CLI)로 파이프라인 시작

파이프라인을 수동으로 시작하고 파이프라인을 통해 아티팩트의 지정된 소스 개정 ID를 실행하려면

 터미널(Linux, macOS 또는 Unix) 또는 명령 프롬프트(Windows)를 열고 AWS CLI 를 사용하 여 시작하려는 파이프라인의 이름을 지정하고 start-pipeline-execution 명령을 실행합니다. 또한 --source-revisions 인수를 사용하여 소스 개정 ID를 제공할 수도 있습니다. 소스 개정은 actionName, revisionType, revisionValue로 구성됩니다. 유효한 revisionType 값은 COMMIT_ID | IMAGE_DIGEST | S3_0BJECT_VERSION_ID | S3_0BJECT_KEY입니다.

다음 예제에서 이름이 codecommit-pipeline인 파이프라인을 통해 지정된 변경 사항의 실행을 시작 하려면 다음 명령을 사용하여 소스 작업 이름을 Source로, 개정 유형을 COMMIT_ID로, 커밋 ID를 78a25c18755ccac3f2a9eec099dEXAMPLE로 지정합니다.

```
aws codepipeline start-pipeline-execution --name codecommit-pipeline --source-
revisions
actionName=Source,revisionType=COMMIT_ID,revisionValue=78a25c18755ccac3f2a9eec099dEXAMPLE
--region us-west-1
```

2. 성공을 확인하려면 반환된 객체를 봅니다. 이 명령은 다음과 같이 실행 ID 객체를 반환합니다.

"pipelineExecutionId": "c53dbd42-This-Is-An-Example"

{

}

Note

파이프라인을 시작했으면 CodePipeline 콘솔에서 진행 상황을 모니터링하거나 getpipeline-state 명령을 실행합니다. 자세한 내용은 <u>파이프라인 보기(콘솔)</u> 및 <u>파이프라인 세</u> 부 정보 및 이력 보기(CLI) 단원을 참조하세요.

CodePipeline에서 파이프라인 실행 중지

파이프라인 실행이 파이프라인을 통해 실행되기 시작하면 파이프라인 실행은 한 번에 한 단계로 들어 가고 단계의 모든 작업 실행이 실행되는 동안 단계를 잠급니다. 이러한 진행 중인 작업은 파이프라인 실행이 중지될 때 작업을 완료하거나 중단할 수 있도록 허용하는 방식으로 처리해야 합니다.

파이프라인 실행을 중지하는 방법은 두 가지입니다.

 중지 및 대기: 진행 중인 모든 작업이 완료될 때까지(즉, 작업이 Succeeded 또는 Failed 상태일 때까지) 실행을 중지할 때까지 AWS CodePipeline 기다립니다. 이 옵션을 선택하면 진행 중인 작업 이 보존됩니다. 실행은 진행 중인 작업이 완료될 때까지 Stopping 상태입니다. 그런 다음 실행은 Stopped 상태입니다. 작업이 완료된 후 단계가 잠금 해제됩니다.

중지하고 대기하도록 선택한 경우 실행이 여전히 Stopping 상태인 동안 생각이 바뀌면 중단하도록 선택할 수 있습니다.

중지 및 중단: 진행 중인 작업이 완료될 때까지 기다리지 않고 실행을 AWS CodePipeline 중지합니다. 실행은 진행 중인 작업이 중단되는 매우 짧은 시간 동안 Stopping 상태입니다. 실행이 중지된후 파이프라인 실행이 Stopped 상태인 동안 작업 실행은 Abandoned 상태입니다. 단계가 잠금 해제됩니다.

파이프라인 실행이 Stopped 상태인 경우 실행이 중지된 단계의 작업은 다시 시도할 수 있습니다.

A Warning

이 옵션을 선택하면 작업이 실패하거나 작업 순서가 뒤바뀔 수 있습니다.

주제

• <u>파이프라인 실행 중지(콘솔)</u>

파이프라인 실행 중지

- 인바운드 실행 중지(콘솔)
- 파이프라인 실행 중지(CLI)
- 인바운드 실행 중지(CLI)

파이프라인 실행 중지(콘솔)

콘솔을 사용하여 파이프라인 실행을 중지할 수 있습니다. 실행을 선택한 다음 파이프라인 실행을 중지 할 방법을 선택합니다.

Note

인바운드 실행인 파이프라인 실행을 중지할 수도 있습니다. 인바운드 실행 중지에 대한 자세한 내용은 <u>인바운드 실행 중지(콘솔)</u>을 참조하세요.

- 1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https//https//https://https//ht
- 2. 다음 중 하나를 수행합니다.

Note

실행을 중지하기 전에 단계 앞의 전환을 비활성화하는 것이 좋습니다. 이렇게 하면 실행 중지로 인해 단계가 잠금 해제될 때 단계에서 후속 파이프라인 실행이 허용되지 않습니다.

- 이름에서 중지하려는 실행이 있는 파이프라인의 이름을 선택합니다. 파이프라인 세부 정보 페 이지에서 실행 중지를 선택합니다.
- 내역 보기를 선택합니다. 기록 페이지에서 실행 중지를 선택합니다.
- 3. 실행 중지 페이지의 실행 선택에서 중지할 실행을 선택합니다.



실행은 아직 진행 중인 경우에만 표시됩니다. 이미 완료된 실행은 표시되지 않습니다.

top execution		>
elect execution hoose the pipeline execution you w	ant to stop.	
	•	
hoose a stop mode for the exe you choose to stop and wait, and y tate, you can choose to abandon.	ecution ou change your mind while your execution is still in a s	topping
• Stop and wait Wait until all in-progress actions are complete.	 Stop and abandon Don't wait until the inprogress actions are complete. Warning: This option can lead to failed actions. 	
itop execution comments - opt	ional	
Cancel Stop		

- 4. 실행에 적용할 작업 선택에서 다음 중 하나를 선택합니다.
 - 진행 중인 모든 작업이 완료될 때까지 실행이 중지되지 않도록 하려면 중지 및 대기를 선택합니다.

(i) Note 실행이 이미 중지 상태인 경우에는 중지하고 대기하도록 선택할 수 없지만 중지하고 중 단하도록 선택할 수 있습니다.

• 진행 중인 작업이 완료될 때까지 대기하지 않고 중지하려면 중지 및 중단을 선택합니다.

🔥 Warning

이 옵션을 선택하면 작업이 실패하거나 작업 순서가 뒤바뀔 수 있습니다.

5. (선택 사항) 설명을 입력합니다. 이러한 설명은 실행 상태와 함께 실행 기록 페이지에 표시됩니다.

6. 중지를 선택합니다.

A Important

이 작업은 실행을 취소할 수 없습니다.

- 7. 다음과 같이 파이프라인 시각화에서 실행 상태를 봅니다.
 - 중지하고 대기하도록 선택하면 진행 중인 작업이 완료될 때까지 선택한 실행이 계속됩니다.
 - 성공 배너 메시지가 콘솔 상단에 표시됩니다.
 - 현재 단계에서 진행 중인 작업은 InProgress 상태로 계속됩니다. 작업이 진행 중인 동안 파 이프라인 실행은 Stopping 상태입니다.

작업이 완료된 후(즉, 작업이 실패하거나 성공한 후) 파이프라인 실행은 Stopped 상태로 변 경되고 작업은 Failed 또는 Succeeded 상태로 변경됩니다. 실행 세부 정보 페이지에서도 작업 상태를 볼 수 있습니다. 실행 기록 페이지 또는 실행 세부 정보 페이지에서 실행 상태를 볼 수 있습니다.

- 파이프라인 실행은 잠시 Stopping 상태로 변경된 다음, Stopped 상태로 변경됩니다. 실행 기록 페이지 또는 실행 세부 정보 페이지에서 실행 상태를 볼 수 있습니다.
- 중지하고 중단하도록 선택하면 실행은 진행 중인 작업이 완료될 때까지 대기하지 않습니다.
 - 성공 배너 메시지가 콘솔 상단에 표시됩니다.
 - 현재 단계에서 진행 중인 작업은 Abandoned 상태로 변경됩니다. 실행 세부 정보 페이지에서 도 작업 상태를 볼 수 있습니다.
 - 파이프라인 실행은 잠시 Stopping 상태로 변경된 다음, Stopped 상태로 변경됩니다. 실행 기록 페이지 또는 실행 세부 정보 페이지에서 실행 상태를 볼 수 있습니다.

실행 기록 보기와 세부 기록 보기에서 파이프라인 실행 상태를 볼 수 있습니다.

인바운드 실행 중지(콘솔)

콘솔을 사용하여 인바운드 실행을 중지할 수 있습니다. 인바운드 실행은 전환이 비활성화된 단계로 들 어가기 위해 대기 중인 파이프라인 실행입니다. 전환이 활성화되면 InProgress인 인바운드 실행이 계속해서 단계에 진입합니다. Stopped인 인바운드 실행은 단계에 들어가지 않습니다. Note

인바운드 실행이 중지된 후에는 다시 시도할 수 없습니다.

인바운드 실행이 보이지 않으면 비활성화된 단계 전환에서 보류 중인 실행이 없는 것입니다.

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> <u>codepipeline/home</u>://https//https://https//h

AWS 계정과 연결된 모든 파이프라인의 이름이 표시됩니다.

- 2. 인바운드 실행을 중지할 파이프라인 이름을 선택하고 다음 중 하나를 수행하세요.
 - 파이프라인 보기에서 인바운드 실행 ID를 선택한 다음 실행을 중지하도록 선택합니다.
 - 파이프라인을 선택하고 기록 보기를 선택합니다. 실행 기록에서 인바운드 실행 ID를 선택한 다 음 실행을 중지하도록 선택합니다.
- 3. 실행 중지 모달에서 위 섹션의 단계에 따라 실행 ID를 선택하고 중지 방법을 지정합니다.

get-pipeline-state 명령을 사용하여 인바운드 실행 상태를 확인합니다.

파이프라인 실행 중지(CLI)

AWS CLI 를 사용하여 파이프라인을 수동으로 중지하려면 다음 파라미터와 함께 stop-pipelineexecution 명령을 사용합니다.

- 실행 ID(필수)
- 설명(선택 사항)
- 파이프라인 이름(필수)
- 중단 플래그(선택 사항, 기본값은 false)

명령 형식:

aws codepipeline stop-pipeline-execution --pipeline-name Pipeline_Name --pipelineexecution-id Execution_ID [--abandon | --no-abandon] [--reason STOP_EXECUTION_REASON]

- 1. 터미널(Linux, macOS, Unix) 또는 명령 프롬프트(Windows)를 엽니다.
- 2. 파이프라인 실행을 중지하려면 다음 중 하나를 선택합니다.

진행 중인 모든 작업이 완료될 때까지 실행이 중지되지 않도록 하려면 중지하고 대기하도록 선택합니다. no-abandon 파라미터를 포함하면 이 작업을 수행할 수 있습니다. 파라미터를 지정하지 않으면 명령은 기본적으로 중지하고 대기하도록 설정됩니다. AWS CLI 를 사용하여 파이프라인의 이름과 실행 ID를 지정하여 stop-pipeline-execution 명령을 실행합니다. 예를 들어, 중지 및 대기 옵션을 지정하여 MyFirstPipeline이라는 파이프라인을 중지하려면 다음과 같이 하세요.

aws codepipeline stop-pipeline-execution --pipeline-name *MyFirstPipeline* -- pipeline-execution-id d-EXAMPLE --no-abandon

예를 들어, 기본적으로 중지 및 대기 옵션을 설정하여 *MyFirstPipeline*이라는 파이프라인을 중지하려면 다음과 같이 하세요.

aws codepipeline stop-pipeline-execution --pipeline-name *MyFirstPipeline* -pipeline-execution-id d-EXAMPLE --reason "Stopping execution after the build action is done"

Note

실행이 이미 중지 상태인 경우 중지하고 대기하도록 선택할 수 없습니다. 이미 중지 상 태인 실행을 중지하고 중단하도록 선택할 수 있습니다.

• 진행 중인 작업이 완료될 때까지 대기하지 않고 중지하려면 중지하고 중단하도록 선택합니다. abandon 파라미터를 포함합니다. AWS CLI 를 사용하여 파이프라인의 이름과 실행 ID를 지정 하여 stop-pipeline-execution 명령을 실행합니다.

예를 들어, 준단 옵션을 지정하고 설명을 포함하도록 선택하여 *MyFirstPipeline*이라는 파이 프라인을 중지하려면 다음과 같이 하세요.

```
aws codepipeline stop-pipeline-execution --pipeline-name MyFirstPipeline --
pipeline-execution-id d-EXAMPLE --abandon --reason "Stopping execution for a bug
fix"
```

인바운드 실행 중지(CLI)

CLI를 사용하여 인바운드 실행을 중지할 수 있습니다. 인바운드 실행은 전환이 비활성화된 단계로 들 어가기 위해 대기 중인 파이프라인 실행입니다. 전환이 활성화되면 InProgress인 인바운드 실행이 계속해서 단계에 진입합니다. Stopped인 인바운드 실행은 단계에 들어가지 않습니다.

1 Note

인바운드 실행이 중지된 후에는 다시 시도할 수 없습니다.

인바운드 실행이 보이지 않으면 비활성화된 단계 전환에서 보류 중인 실행이 없는 것입니다.

AWS CLI 를 사용하여 인바운드 실행을 수동으로 중지하려면 다음 파라미터와 함께 stop-pipelineexecution 명령을 사용합니다.

- 인바운드 실행 ID(필수)
- 설명(선택 사항)
- 파이프라인 이름(필수)
- 중단 플래그(선택 사항, 기본값은 false)

명령 형식:

```
aws codepipeline stop-pipeline-execution --pipeline-name Pipeline_Name --
pipeline-execution-id Inbound_Execution_ID [--abandon | --no-abandon] [--
reason STOP_EXECUTION_REASON]
```

위 절차의 단계에 따라 명령을 입력하고 중지 방법을 지정합니다.

get-pipeline-state 명령을 사용하여 인바운드 실행 상태를 확인합니다.

파이프라인 실행에 대한 기록 보기 및 모드 설정

파이프라인 진행 상황을 분석하기 위해 파이프라인 및 작업 실행 기록을 볼 수 있습니다. 파이프라인 실행을 처리하기 위한 릴리스 방법을 구성하려면 실행 모드를 설정합니다.

주제

- CodePipeline에서 실행 보기
- 파이프라인 실행 모드 설정 또는 변경

CodePipeline에서 실행 보기

AWS CodePipeline 콘솔 또는를 사용하여 실행 상태를 보고, 실행 기록을 보고, 실패한 단계 또는 작업 을 다시 시도 AWS CLI 할 수 있습니다.

주제

- 파이프라인 실행 내역 보기(콘솔)
- 실행상태보기(콘솔)
- 인바운드 실행 보기(콘솔)
- 파이프라인 실행 소스 개정 보기(콘솔)
- 작업 실행 보기(콘솔)
- 작업 아티팩트 및 아티팩트 스토어 정보 보기(콘솔)
- <u>파이프라인 세부 정보 및 이력 보기(CLI)</u>

파이프라인 실행 내역 보기(콘솔)

CodePipeline 콘솔을 사용하여 계정 내 모든 파이프라인의 목록을 볼 수 있습니다. 파이프라인에서 마 지막 작업을 실행한 시기, 단계 간 전환이 활성화 상태인지 비활성화 상태인지, 실패한 작업이 있는지 등 각 파이프라인의 세부 정보와 기타 정보도 볼 수 있습니다. 내역이 기록된 모든 파이프라인 실행에 대한 세부 정보를 보여 주는 내역 페이지를 볼 수도 있습니다.

Note

특정 실행 모드 간에 전환할 때 파이프라인 보기와 기록이 변경될 수 있습니다. 자세한 내용은 <u>파이프라인 실행 모드 설정 또는 변경</u> 단원을 참조하십시오. 실행 내역은 최대 12개월 동안 보존됩니다.

콘솔을 사용하여 상태. 소스 개정 및 각 실행에 대한 시간 세부 정보를 포함하여 파이프라인에서 실행 내역을 볼 수 있습니다.

1. 에 로그인 AWS Management Console 하고 http://console.aws.amazon.com/codesuite/ codepipeline/home://https://https://https://ittps//i

AWS 계정과 연결된 모든 파이프라인의 이름이 상태와 함께 표시됩니다.

- 이름에서 파이프라인의 이름을 선택합니다. 2.
- 3. 내역 보기를 선택합니다.

Note

PARALLEL 실행 모드의 파이프라인의 경우 기본 파이프라인 보기에 파이프라인 구조 또 는 진행 중인 실행이 표시되지 않습니다. PARALLEL 실행 모드의 파이프라인의 경우 실행 기록 페이지에서 보려는 실행의 ID를 선택하여 파이프라인 구조에 액세스합니다. 왼쪽 탐 색에서 기록을 선택하고 병렬 실행의 실행 ID를 선택한 다음 시각화 탭에서 파이프라인을 봅니다.

xec	ution history	Info		Rerun St	op execution	View details	Release change
Q							<pre>1 > 6</pre>
	Execution ID	Status	Source revisions	Trigger	Started	Duration	Completed
)	33bdf70c Rollback	⊘ Succeeded	Source – <u>73ae512c</u> : Added README.txt	-	Apr 16, 2024 2:51 AM (UTC-7:00)	1 second	Apr 16, 2024 2:51 AM (UTC-7:00)
)	3f658bd1 Rollback	Succeeded Succeeded	Source – <u>73ae512c</u> : Added README.txt	-	Apr 16, 2024 2:16 AM (UTC-7:00)	1 second	Apr 16, 2024 2:16 AM (UTC-7:00)
)	4f47bed9	⊘ Succeeded	Source – <u>73ae512c</u> : Added README.txt	StartPipelineExec ution	Apr 16, 2024 2:14 AM (UTC-7:00)	5 seconds	Apr 16, 2024 2:14 AM (UTC-7:00)
)	eb7ebd36	⊘ Succeeded	Source – <u>73ae512c</u> : Added README.txt	-	Apr 16, 2024 2:00 AM	1 second	Apr 16, 2024 2:00 AM

- 상태, 소스 버전, 변경 세부 정보, 파이프라인의 각 실행과 관련된 트리거를 확인합니다. 롤백된 파 이프라인 실행은 콘솔의 세부 정보 화면에 실행 유형 롤백을 표시합니다. 자동 롤백을 트리거한 실 패한 실행의 경우 실패한 실행 ID가 표시됩니다.
- 실행을 선택합니다. 세부 정보 보기에는 실행 세부 정보, 타임라인 탭, 시각화 탭 및 변수 탭이 표시 됩니다. 파이프라인 수준의 변수 값은 파이프라인 실행 시 확인되며 각 실행의 실행 기록에서 확인 할 수 있습니다.

Note

파이프라인 작업의 출력 변수는 각 작업 실행 기록 아래의 출력 변수 탭에서 볼 수 있습니 다.

실행상태보기(콘솔)

실행 내역 페이지에서 상태에서 파이프라인 상태를 볼 수 있습니다. 실행 ID 링크를 선택한 다음 작업 상태를 봅니다.

다음은 파이프라인, 스테이지 및 작업에 대한 유효한 상태입니다.

Note

다음 파이프라인 상태는 인바운드 실행인 파이프라인 실행에도 적용됩니다. 인바운드 실행 및 상태를 보려면 인바운드 실행 보기(콘솔)을 참조하세요.

파이프라인 수준 상태

파이프라인 상태	설명
InProgress	파이프라인 실행이 현재 실행 중입니다.
Stopping	파이프라인 실행을 중지하고 대기하거나 중지하고 중단하라는 요청으로 인해 파이프라인 실행이 중지되는 중입니다.
Stopped	중지 중인 프로세스가 완료되고 파이프라인 실행이 중지됩니다.
성공	파이프라인 실행이 성공적으로 완료되었습니다.

파이프라인 상태	설명
대체됨	이 파이프라인 실행이 다음 단계가 완료되기를 기다리는 동안 새로운 파이 프라인 실행이 진행되고 해당 파이프라인에서 계속되었습니다.
Failed	파이프라인 실행이 성공적으로 완료되지 않았습니다.

단계 수준 상태

단계 상태	설명
InProgress	단계가 현재 실행 중입니다.
Stopping	파이프라인 실행을 중지하고 대기하거나 중지하고 중단하라는 요청으로 인해 단계 실행이 중지되는 중입니다.
Stopped	중지 중인 프로세스가 완료되고 단계 실행이 중지됩니다.
성공	단계가 성공적으로 완료되었습니다.
Failed	단계가 성공적으로 완료되지 않았습니다.

작업 수준 상태

작업 상태	설명
InProgress	작업이 현재 실행 중입니다.
중단됨	파이프라인 실행을 중지하고 중단하라는 요청으로 인해 작업이 중단됩니 다.
성공	작업이 성공적으로 완료되었습니다.
Failed	승인 작업의 경우 FAILED 상태는 검토자가 작업을 거부했거나 잘못된 작 업 구성으로 인해 작업이 실패했음을 의미합니다.

인바운드 실행 보기(콘솔)

콘솔을 사용하여 인바운드 실행의 상태 및 세부 정보를 볼 수 있습니다. 전환이 활성화되거나 단계를 사용할 수 있게 되면 InProgress인 인바운드 실행이 계속되고 단계에 진입합니다. Stopped 상태인 인바운드 실행은 단계에 들어가지 않습니다. 파이프라인이 편집되면 인바운드 실행 상태가 Failed로 변경됩니다. 파이프라인을 편집하면 진행 중인 모든 실행이 계속되지 않고 실행 상태가 Failed로 변 경됩니다.

인바운드 실행이 보이지 않으면 비활성화된 단계 전환에서 보류 중인 실행이 없는 것입니다.

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https//https/

AWS 계정과 연결된 모든 파이프라인의 이름이 표시됩니다.

- 2. 인바운드 실행을 보려는 파이프라인 이름을 선택하고 다음 중 하나를 수행하세요.
 - 보기를 선택합니다. 파이프라인 다이어그램의 비활성화된 전환 앞에 있는 인바운드 실행 ID 필 드에서 인바운드 실행 ID를 볼 수 있습니다.

Details 🖬			
6793d5e5 Source: Edited docume	ntdb_full_stack.yml		0
Inbound execution ID: 28d5a641-43	23-	View summary	
Disable transition]		8
Approval Pending Pipeline execution ID: baf5ecbd-0	4d2-4		
Approval	0	2	

요약 보기를 선택하면 실행 ID, 소스 트리거, 다음 단계 이름과 같은 실행 세부 정보를 볼 수 있습니다.

• 파이프라인을 선택하고 기록 보기를 선택합니다.

파이프라인 실행 소스 개정 보기(콘솔)

파이프라인 실행에 사용되는 소스 아티팩트(파이프라인의 첫 스테이지에 시작된 출력 아티팩트)에 대 한 세부 정보를 볼 수 있습니다. 세부 정보에는 커밋 ID, 체크인 설명, 사용자가 CLI를 사용하는 시점, 파이프라인 빌드 작업의 버전 번호 등 식별자가 포함됩니다. 일부 개정 유형의 경우, 커밋 URL을 보고 열 수 있습니다. 소스 개정은 다음으로 구성됩니다.

- 요약: 아티팩트의 최신 개정에 대한 요약 정보. GitHub 및 CodeCommit 리포지토리의 경우 커밋 메 시지. Amazon S3 버킷 또는 작업의 경우 객체 메타데이터에 지정된 codepipeline-artifact-revisionsummary 키의 사용자가 제공한 컨텐츠.
- revisionUrl: 아티팩트 개정 버전의 개정 URL(예: 외부 리포지토리 URL).
- revisionId: 아티팩트 개정의 개정 ID. 예를 들어 CodeCommit 또는 GitHub 리포지토리의 소스 변경 의 경우 커밋 ID입니다. GitHub 또는 CodeCommit 리포지토리에 저장된 아티팩트의 경우 커밋 ID가 커밋 세부 정보 페이지에 연결됩니다.
- 1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https://https://https://https://i/httpsCodePipeline.com.com.com.com.

사용자의 AWS 계정에 연결된 모든 파이프라인의 이름이 표시됩니다.

- 2. 소스 개정 세부 정보를 보려는 파이프라인의 이름을 선택합니다. 다음 중 하나를 수행합니다.
 - 내역 보기를 선택합니다. Source revisions(소스 개정)에서 각 실행에 대한 소스 변경 사항이 나 열됩니다.
 - 다음과 같이 소스 개정 세부 정보를 보려는 작업을 찾은 다음, 단계의 하단에서 개정 정보를 찾 습니다.



View current revisions(현재 개정 보기)를 선택하여 소스 정보를 확인합니다. Amazon S3 버킷 에 저장된 아티팩트를 제외하고, 이 세부 정보 보기에 있는 커밋 ID와 같은 식별자는 아티팩트의 소스 정보 페이지로 연결됩니다.

"Source" current revisions These are the source artifact changes that started the current pipeline execution.	×
Source Amazon S3 version id: ZqY_zLkx	
Execution ID	
4e49e3b7-7dcc-4d93-b91a-	
Don	e

작업 실행 보기(콘솔)

작업 실행 ID, 입력 아티팩트, 출력 아티팩트 및 상태와 같은 파이프라인에 대한 작업 세부 정보를 볼 수 있습니다. 콘솔에서 파이프라인을 선택한 다음 실행 ID를 선택하여 작업 세부 정보를 볼 수 있습니다.

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https//https/

AWS 계정과 연결된 모든 파이프라인의 이름이 표시됩니다.

- 작업 세부 정보를 보려는 파이프라인의 이름을 선택한 다음 View history(내역 보기)를 선택합니다.
- 3. 실행 ID에서 작업 실행 세부 정보를 보고자 하는 실행 ID를 선택합니다.
- 4. 타임라인 탭에서 다음 정보를 볼 수 있습니다.
 - a. 작업 이름에서 링크를 선택하여 상태, 스테이지 이름, 작업 이름, 구성 데이터 및 아티팩트 정 보를 볼 수 있는 작업의 세부 정보 페이지를 엽니다.
 - b. 공급자에서 링크를 선택하여 작업 공급자 세부 정보를 봅니다. 예를 들어, 앞의 파이프라인 예에서, 스테이징이나 프로덕션 단계 중 하나에서 CodeDeploy를 선택했다면 해당 단계에 대해 구성된 CodeDeploy 애플리케이션의 CodeDeploy 콘솔 페이지가 표시됩니다.

작업 아티팩트 및 아티팩트 스토어 정보 보기(콘솔)

작업에 대한 입력 및 출력 아티팩트 세부 정보를 볼 수 있습니다. 해당 작업의 아티팩트 정보로 연결되 는 링크를 선택할 수도 있습니다. 아티팩트 스토어에서는 버전 관리를 사용하므로 각 작업 실행에는 고 유한 입력 및 출력 아티팩트 위치가 있습니다.

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https//https

AWS 계정과 연결된 모든 파이프라인의 이름이 표시됩니다.

- 작업 세부 정보를 보려는 파이프라인의 이름을 선택한 다음 View history(내역 보기)를 선택합니다.
- 3. 실행 ID에서 작업 세부 정보를 보고자 하는 실행 ID를 선택합니다.
- 4. 타임라인 탭의 작업 이름에서 링크를 선택하여 이 작업에 대한 세부 정보 페이지를 엽니다.
- 5. 세부 정보 페이지의 실행 탭에서 작업 실행의 상태 및 시간을 봅니다.
- 6. 구성 탭에서 작업에 대한 리소스 구성(예: CodeBuild 빌드 프로젝트 이름)을 봅니다.
- 7. 아티팩트 탭에서 아티팩트 유형 및 아티팩트 공급자의 아티팩트 세부 정보를 봅니다. Artifact name(결과물 이름) 아래의 링크를 선택하여 결과물 스토어에 있는 결과물을 봅니다.
- 8. 출력 변수 탭에서 작업 실행을 위해 파이프라인의 작업에서 해결된 변수를 확인합니다.

파이프라인 세부 정보 및 이력 보기(CLI)

다음 명령을 실행하면 파이프라인 및 파이프라인 실행에 대한 세부 정보를 볼 수 있습니다.

- list-pipelines 명령을 사용하여와 연결된 모든 파이프라인의 요약을 봅니다 AWS 계정.
- get-pipeline 명령을 실행하면 단일 파이프라인의 세부 정보를 검토할 수 있습니다.
- list-pipeline-executions를 실행하면 파이프라인의 가장 최근 실행 내역에 대한 요약 정보가 표시됩니다.
- get-pipeline-execution을 실행하면 아티팩트의 세부 정보, 파이프라인 실행 ID, 파이프라인의 이름과 버전 및 상태 등 파이프라인 실행에 대한 정보가 표시됩니다.
- get-pipeline-state 명령으로 파이프라인, 스테이지 및 작업 상태를 봅니다.
- list-action-executions로 파이프라인 실행 세부 정보를 봅니다.

주제

작업 아티팩트 및 아티팩트 스토어 정보 보기(콘솔)

- list-pipeline-executions(CLI)를 사용하여 실행 기록 보기
- get-pipeline-state(CLI)를 사용하여 파이프라인 상태 보기
- get-pipeline-state(CLI)를 사용하여 인바운드 실행 상태 보기
- get-pipeline-execution(CLI)을 사용하여 상태 및 소스 개정 보기
- list-action-executions(CLI)를 사용하여 작업 실행 보기

list-pipeline-executions(CLI)를 사용하여 실행 기록 보기

파이프라인 실행 내역을 볼 수 있습니다.

 파이프라인의 지난 실행 내역에 대한 세부 정보를 보려면 파이프라인의 고유 이름을 지정한 채 <u>list-pipeline-executions</u> 명령을 실행합니다. 예를 들어 MyFirstPipeline이라는 파이프라인의 현재 상태에 대한 세부 정보를 보려면 다음을 입력합니다.

aws codepipeline list-pipeline-executions --pipeline-name MyFirstPipeline

이 명령은 내역이 기록된 모든 파이프라인 실행에 대한 요약 정보를 반환합니다. 이 요약에는 시작 및 종료 시간, 실행 기간 및 상태가 나와 있습니다.

롤백된 파이프라인 실행에는 Rollback 실행 유형이 표시됩니다. 자동 롤백을 트리거한 실패한 실행의 경우 실패한 실행 ID가 표시됩니다.

다음 예에서는 3번의 실행 이력이 있는 *MyFirstPipeline*이라는 파이프라인에 대해 반환된 데 이터를 보여 줍니다.

```
],
    "trigger": {
        "triggerType": "StartPipelineExecution",
        "triggerDetail": "trigger_ARN"
    },
    "executionMode": "SUPERSEDED"
},
{
    "pipelineExecutionId": "fcd61d8b-4532-4384-9da1-2aca1EXAMPLE",
    "status": "Succeeded",
    "startTime": "2024-04-16T08:58:56.601000+00:00",
    "lastUpdateTime": "2024-04-16T08:59:04.274000+00:00",
    "sourceRevisions": [
        {
            "actionName": "Source",
            "revisionId": "revision_ID",
            "revisionSummary": "Added README.txt",
            "revisionUrl": "console_URL"
        }
    ],
    "trigger": {
        "triggerType": "StartPipelineExecution",
        "triggerDetail": "trigger_ARN"
    },
    "executionMode": "SUPERSEDED"
}
```

파이프라인 실행에 대한 세부 정보를 보려면 파이프라인 실행의 고유 ID를 지정한 채 <u>get-pipeline-</u> <u>execution</u> 명령을 실행합니다. 예를 들어 이전 예의 첫 번째 실행에 대한 보다 세부적인 정보를 보 려면 다음을 입력합니다.

```
aws codepipeline get-pipeline-execution --pipeline-name MyFirstPipeline --pipeline-
execution-id 7cf7f7cb-3137-539g-j458-d7eu3EXAMPLE
```

이 명령은 아티팩트의 세부 정보, 파이프라인 실행 ID, 파이프라인의 이름과 버전 및 상태 등 파이 프라인 실행에 대한 요약 정보를 반환합니다.

다음 예에서는 MyFirstPipeline이라는 파이프라인에 대해 반환된 데이터를 보여 줍니다.

```
"pipelineExecution": {
    "pipelineExecutionId": "3137f7cb-7cf7-039j-s831-d7eu3EXAMPLE",
```

{



get-pipeline-state(CLI)를 사용하여 파이프라인 상태 보기

CLI를 사용하여 파이프라인, 스테이지 및 작업 상태를 볼 수 있습니다.

 파이프라인의 현재 상태에 대한 세부 정보를 보려면 파이프라인의 고유 이름을 지정한 채 getpipeline-state 명령을 실행합니다. 예를 들어 MyFirstPipeline이라는 파이프라인의 현재 상태 에 대한 세부 정보를 보려면 다음을 입력합니다.

aws codepipeline get-pipeline-state --name MyFirstPipeline

이 명령은 파이프라인의 전체 단계의 현재 상태 및 해당 단계 내 작업의 상태를 반환합니다.

다음 예에서는 *MyFirstPipeline*이라는 세 단계 파이프라인에 대해 반환된 데이터를 보여 줍니 다. 여기서 처음 두 단계 및 작업에는 성공이 표시되고, 세 번째 단계 및 작업에는 실패가 표시되며, 두 번째와 세 번째 단계 사이의 전환은 비활성화된 것으로 표시됩니다.

```
"entityUrl": "https://console.aws.amazon.com/s3/home?#",
                    "latestExecution": {
                         "status": "Succeeded",
                         "lastStatusChange": 1427298837.768
                    }
                }
            ],
            "stageName": "Source"
        },
        {
            "actionStates": [
                {
                    "actionName": "Deploy-CodeDeploy-Application",
                    "entityUrl": "https://console.aws.amazon.com/codedeploy/home?
#",
                    "latestExecution": {
                         "status": "Succeeded",
                         "lastStatusChange": 1427298939.456,
                         "externalExecutionUrl": "https://console.aws.amazon.com/?
#",
                         "externalExecutionId": ""c53dbd42-This-Is-An-Example"",
                         "summary": "Deployment Succeeded"
                    }
                }
            ],
            "inboundTransitionState": {
                "enabled": true
            },
            "stageName": "Staging"
        },
        {
            "actionStates": [
                {
                    "actionName": "Deploy-Second-Deployment",
                    "entityUrl": "https://console.aws.amazon.com/codedeploy/home?
#",
                    "latestExecution": {
                         "status": "Failed",
                         "errorDetails": {
                             "message": "Deployment Group is already deploying
 deployment ....",
                             "code": "JobFailed"
                        },
                         "lastStatusChange": 1427246155.648
```



get-pipeline-state(CLI)를 사용하여 인바운드 실행 상태 보기

CLI를 사용하여 인바운드 실행 상태를 확인할 수 있습니다. 전환이 활성화되거나 단계를 사용할 수 있 게 되면 InProgress인 인바운드 실행이 계속되고 단계에 진입합니다. Stopped 상태인 인바운드 실 행은 단계에 들어가지 않습니다. 파이프라인이 편집되면 인바운드 실행 상태가 Failed로 변경됩니다. 파이프라인을 편집하면 진행 중인 모든 실행이 계속되지 않고 실행 상태가 Failed로 변경됩니다.

• 파이프라인의 현재 상태에 대한 세부 정보를 보려면 파이프라인의 고유 이름을 지정한 채 <u>get-</u> pipeline-state 명령을 실행합니다. 예를 들어 *MyFirstPipeline*이라는 파이프라인의 현재 상태 에 대한 세부 정보를 보려면 다음을 입력합니다.

aws codepipeline get-pipeline-state --name MyFirstPipeline

이 명령은 파이프라인의 전체 단계의 현재 상태 및 해당 단계 내 작업의 상태를 반환합니다. 또한 출력에는 각 단계의 파이프라인 실행 ID와 전환이 비활성화된 단계에 대한 인바운드 실행 ID가 있 는지 여부도 표시됩니다.

다음 예제는 *MyFirstPipeline*이라는 2단계 파이프라인에 대해 반환된 데이터를 보여줍니다. 여기서 첫 번째 단계는 활성화된 전환과 성공적인 파이프라인 실행을 보여주고, Beta라는 두 번 째 단계는 비활성화된 전환과 인바운드 실행 ID를 보여줍니다. 인바운드 실행은 InProgress, Stopped 또는 FAILED 상태일 수 있습니다.

```
"pipelineName": "MyFirstPipeline",
"pipelineVersion": 2,
```

{

```
"stageStates": [
   {
        "stageName": "Source",
        "inboundTransitionState": {
            "enabled": true
        },
        "actionStates": [
            {
                "actionName": "SourceAction",
                "currentRevision": {
                    "revisionId": "PARcnxX_u0SMRBnKh83pHL09.zPRLLMu"
                },
                "latestExecution": {
                    "actionExecutionId": "14c8b311-0e34-4bda-EXAMPLE",
                    "status": "Succeeded",
                    "summary": "Amazon S3 version id: PARcnxX_u0EXAMPLE",
                    "lastStatusChange": 1586273484.137,
                    "externalExecutionId": "PARcnxX_u0EXAMPLE"
                },
                "entityUrl": "https://console.aws.amazon.com/s3/home?#"
            }
        ],
        "latestExecution": {
            "pipelineExecutionId": "27a47e06-6644-42aa-EXAMPLE",
            "status": "Succeeded"
        }
   },
    {
        "stageName": "Beta",
        "inboundExecution": {
            "pipelineExecutionId": "27a47e06-6644-42aa-958a-EXAMPLE",
            "status": "InProgress"
        },
        "inboundTransitionState": {
            "enabled": false,
            "lastChangedBy": "USER_ARN",
            "lastChangedAt": 1586273583.949,
            "disabledReason": "disabled"
        },
                "currentRevision": {
        "actionStates": [
            {
                "actionName": "BetaAction",
                "latestExecution": {
```

```
"actionExecutionId": "a748f4bf-0b52-4024-98cf-EXAMPLE",
                         "status": "Succeeded",
                         "summary": "Deployment Succeeded",
                         "lastStatusChange": 1586272707.343,
                         "externalExecutionId": "d-KFGF3EXAMPLE",
                        "externalExecutionUrl": "https://us-
west-2.console.aws.amazon.com/codedeploy/home?#/deployments/d-KFGF3WTS2"
                    },
                    "entityUrl": "https://us-west-2.console.aws.amazon.com/
codedeploy/home?#/applications/my-application"
                }
            ],
            "latestExecution": {
                "pipelineExecutionId": "f6bf1671-d706-4b28-EXAMPLE",
                "status": "Succeeded"
            }
        }
    ],
    "created": 1585622700.512,
    "updated": 1586273472.662
}
```

get-pipeline-execution(CLI)을 사용하여 상태 및 소스 개정 보기

파이프라인 실행에 사용되는 소스 아티팩트(파이프라인의 첫 스테이지에 시작된 출력 결과물)에 대한 세부 정보를 볼 수 있습니다. 세부 정보에는 커밋 ID, 체크인 설명, 아티팩트가 생성되었거나 업데이트 된 이후 경과한 시간, 사용자가 CLI를 사용하는 시점, 빌드 작업의 버전 번호 등 식별자가 포함됩니다. 일부 개정 유형의 경우, 아티팩트 버전의 커밋 URL을 확인하고 열 수 있습니다. 소스 개정은 다음으로 구성됩니다.

- 요약: 아티팩트의 최신 개정에 대한 요약 정보. GitHub 및 AWS CodeCommit 리포지토리의 경우 커밋 메시지입니다. Amazon S3 버킷 또는 작업의 경우 객체 메타데이터에 지정된 codepipelineartifact-revision-summary 키의 사용자가 제공한 컨텐츠.
- revisionUrl: 아티팩트 개정의 커밋 ID. GitHub 또는 AWS CodeCommit 리포지토리에 저장된 아티팩 트의 경우 커밋 ID는 커밋 세부 정보 페이지에 연결됩니다.

get-pipeline-execution 명령을 실행하여 파이프라인 실행에 포함되었던 최신 소스 개정에 대한 정보 를 볼 수 있습니다. 파이프라인의 모든 스테이지에 대한 세부 정보를 얻기 위해 처음으로 get-pipelinestate 명령을 실행한 후에 소스 개정 세부 정보를 원하는 스테이지에 적용되는 실행 ID를 식별합니다. 그런 다음 get-pipeline-execution 명령에서 실행 ID를 사용합니다. (파이프라인의 스테이지가 서로 다 른 파이프라인 실행 중 성공적으로 완료된 마지막 스테이지였을 수 있으므로 실행 ID가 서로 다를 수 있습니다.)

달리 말해, 현재 스테이징 단계에 있는 아티팩트에 관한 세부 정보를 보려면 get-pipeline-state 명령을 실행하고 스테이징 단계의 현재 실행 ID를 식별한 다음, 실행 ID를 사용하여 get-pipeline-execution 명 령을 실행합니다.

파이프라인의 상태 및 소스 개정을 보려면

 터미널(Linux, macOS, Unix) 또는 명령 프롬프트(Windows)를 열고 AWS CLI 를 사용하여 <u>get-</u> pipeline-state 명령을 실행합니다. *MyFirstPipeline*이라는 파이프라인의 경우 다음을 입력합 니다.

aws codepipeline get-pipeline-state --name MyFirstPipeline

이 명령은 각 단계의 최근 파이프라인 실행 ID를 포함하여 파이프라인의 최근 상태를 반환합니다.

 파이프라인 실행에 대한 세부 정보를 보려면 get-pipeline-execution 명령을 실행하여 파이프라 인의 고유한 이름 및 아티팩트 세부 정보를 보고자 하는 실행의 파이프라인 실행 ID를 지정합니 다. 예를 들어 *MyFirstPipeline*이라는 파이프라인 실행에 대한 세부 정보를 보려면 실행 ID가 3137f7cb-7cf7-039j-s83I-d7eu3EXAMPLE인 경우, 다음과 같이 입력합니다.

```
aws codepipeline get-pipeline-execution --pipeline-name MyFirstPipeline --pipeline-execution-id 3137f7cb-7cf7-039j-s83l-d7eu3EXAMPLE
```

이 명령은 파이프라인 실행의 일부인 각 소스 개정에 대한 정보와 파이프라인에 대한 식별 정보를 반환합니다. 해당 실행에 포함된 파이프라인 단계에 대한 정보만 포함됩니다. 해당 파이프라인 실 행의 일부가 아닌 다른 파이프라인 단계가 있을 수 있습니다.

다음 예에서는 *MyFirstPipeline*이라는 파이프라인의 일부에 대해 반환된 데이터를 보여 줍니다. 이 파이프라인에는 "MyApp"이라는 아티팩트가 GitHub 리포지토리에 저장되어 있습니다.

3.

{

```
"pipelineExecution": {
    "artifactRevisions": [
        {
            "created": 1427298837.7689769,
            "name": "MyApp",
            "revisionChangeIdentifier": "1427298921.3976923",
            "revisionId": "7636d59f3c461cEXAMPLE8417dbc6371",
```

```
"revisionSummary": "Updating the application for feature 12-4820",
                "revisionUrl": "https://api.github.com/repos/anycompany/MyApp/git/
commits/7636d59f3c461cEXAMPLE8417dbc6371"
            }
        ],
        "pipelineExecutionId": "3137f7cb-7cf7-039j-s83l-d7eu3EXAMPLE",
        "pipelineName": "MyFirstPipeline",
        "pipelineVersion": 2,
        "status": "Succeeded",
        "executionMode": "SUPERSEDED",
        "executionType": "ROLLBACK",
        "rollbackMetadata": {
            "rollbackTargetPipelineExecutionId": "4f47bed9-6998-476c-a49d-
e60beEXAMPLE"
         }
    }
}
```

list-action-executions(CLI)를 사용하여 작업 실행 보기

작업 실행 ID, 입력 아티팩트, 출력 아티팩트, 실행 결과 및 상태와 같은 파이프라인에 대한 작업 실행 세부 정보를 볼 수 있습니다. 실행 ID 필터를 제공하여 파이프라인 실행에서 작업 목록을 반환합니다.

Note

자세한 실행 내역은 2019년 2월 21일 또는 그 이후에 실행되는 실행에 사용할 수 있습니다.

- 파이프라인에 대한 작업 실행을 보려면 다음 중 하나를 수행하십시오.
 - 파이프라인의 모든 작업 실행에 대한 세부 정보를 보려면 list-action-executions 명령을 실행하 여 파이프라인의 고유 이름을 지정합니다. 예를 들어, *MyFirstPipeline*이라는 파이프라인의 작업 실행 세부 정보를 보려면 다음을 입력합니다.

aws codepipeline list-action-executions --pipeline-name MyFirstPipeline

다음은 이 명령에 대한 샘플 출력의 일부입니다.

"actionExecutionDetails": [

{



• 파이프라인 실행에서 모든 작업 실행을 보려면 list-action-executions 명령을 실행하여 파이프라 인의 고유 이름 및 실행 ID를 지정합니다. 예를 들어 *Execution_ID*에 대한 작업 실행을 보려 면 다음을 입력합니다.

aws codepipeline list-action-executions --pipeline-name MyFirstPipeline --filter
pipelineExecutionId=Execution_ID

• 다음은 이 명령에 대한 샘플 출력의 일부입니다.

```
{
    "actionExecutionDetails": [
        {
            "stageName": "Beta",
            "pipelineVersion": 8,
            "actionName": "Build",
            "status": "Failed",
            "lastUpdateTime": 1552958312.034,
            "input": {
                "configuration": {
                    "ProjectName": "java-project"
                },
                "region": "us-east-1",
                "actionTypeId": {
                    "owner": "AWS",
                    "category": "Build",
                    "provider": "CodeBuild",
                    "version": "1"
                },
                "inputArtifacts": [
                    {
                         "s3location": {
                             "bucket": "codepipeline-us-east-1-ID",
                             "key": "MyFirstPipeline/MyApp/Object.zip"
                         },
                         "name": "MyApp"
                    }
                ]
            },
 . .
```

파이프라인 실행 모드 설정 또는 변경

파이프라인의 실행 모드를 설정하여 여러 실행을 처리하는 방법을 지정할 수 있습니다.

파이프라인 실행 모드에 대한 자세한 내용은 파이프라인 실행 작동 방식 섹션을 참조하세요.

▲ Important

PARALLEL 모드의 파이프라인의 경우 파이프라인 실행 모드를 QUEUED 또는 SUPERSEDED로 편집하면 파이프라인 상태에 업데이트된 상태가 PARALLEL로 표시되지 않 습니다. 자세한 내용은 <u>PARALLEL 모드에서 변경된 파이프라인이 이전 실행 모드를 표시합니</u> 다. 단원을 참조하십시오.

▲ Important

PARALLEL 모드의 파이프라인의 경우 파이프라인 실행 모드를 QUEUED 또는 SUPERSEDED로 편집하면 각 모드의 파이프라인에 대한 파이프라인 정의가 업데이트되지 않 습니다. 자세한 내용은 <u>QUEUED 또는 SUPERSEDED 모드로 변경할 때 편집하면 PARALLEL</u> 모드의 파이프라인에 오래된 파이프라인 정의가 있습니다. 단원을 참조하십시오.

A Important

PARALLEL 모드의 파이프라인의 경우 스테이지 롤백을 사용할 수 없습니다. 마찬가지로 롤백 결과 유형이 있는 실패 조건은 PARALLEL 모드 파이프라인에 추가할 수 없습니다.

실행 모드 보기에 대한 고려 사항

특정 실행 모드에서 파이프라인을 볼 때 고려해야 할 사항이 있습니다.

SUPERSEDED 및 QUEUED 모드의 경우 파이프라인 보기를 사용하여 진행 중인 실행을 확인하고 실 행 ID를 클릭하여 세부 정보 및 기록을 확인합니다. PARALLEL 모드에서 실행 ID를 클릭하여 시각화 탭에서 진행 중인 실행을 확인합니다.

다음은 CodePipeline의 SUPERSEDED 모드에 대한 보기를 보여줍니다.

MyPipeline Pipeline type: V2 Execution mode: SUPERSEDED	♦ Notify ▼ Edit	Stop execution	Clone pipeline	Release change	
Source Succeeded Pipeline execution ID: <u>3ff0e57c-e595-407c-8668-</u>					
Source <u>GitHub (Version 2)</u> 2 Succeeded - <u>1 minute ago</u> <u>77cc2e44</u> 2 View details <u>77cc2e44</u> 2 Source: Merge pull request #5 from //feature-bu	ranch •••				0
Disable transition					
Build In progress Pipeline execution ID: <u>3ff0e57c-e595-407c-866</u>					
Build					

다음은 CodePipeline의 QUEUED 모드에 대한 보기를 보여줍니다.
yPipeline	tify 🔻 Edit	Stop execution	Clone pipeline	Release change
line type: V2 Execution mode: QUEUE	D			
-				
✓ Source Succeeded Pipeline execution ID: <u>100f7c0e-4545-485a-88ea-(</u>	;			
Source				
GitHub (Version 2)				
Succeeded - Just now <u>77cc2e44</u>				
View details				
77cc2e44 🗹 Source: Merge pull request #5 from	/feature-branch ••••			
Disable transition				
V				
Build In progress				
Pipeline execution ID: 100f7c0e-4545-485a-88ea-				
Build				
AWS CodeBuild				

다음은 CodePipeline의 PARALLEL 모드에 대한 보기를 보여줍니다.



Visualization Timeline Va	ariables Revisions
Source Succeeded	
Source <u>GitHub (Version 2)</u>	
View details	 ✓ ○
Build In progress	
Build	
AWS CodeBuild In progress - Just now View details	

실행 모드 간 전환 시 고려 사항

다음은 파이프라인의 모드를 변경할 때 파이프라인에 대한 고려 사항입니다. 편집 모드에서 한 실행 모 드에서 다른 실행 모드로 전환한 다음 변경 사항을 저장하면 특정 뷰 또는 상태가 조정될 수 있습니다.

예를 들어 PARALLEL 모드에서 QUEUED 또는 SUPERSEDED 모드로 전환하면 PARALLEL 모드에 서 시작된 실행이 계속 실행됩니다. 이는 실행 기록 페이지에서 볼 수 있습니다. 파이프라인 보기에는 QUEUED 또는 SUPERSEDED 모드에서 이전에 실행된 실행이 표시되고 그렇지 않으면 빈 상태가 표 시됩니다.

또 다른 예로 QUEUED 또는 SUPERSEDED에서 PARALLEL 모드로 전환하면 파이프라인 보기/상태 페이지가 더 이상 표시되지 않습니다. PARALLEL 모드에서 실행을 보려면 실행 세부 정보 페이지의 시 각화 탭을 사용합니다. SUPERSEDED 또는 QUEUED 모드에서 시작된 실행은 취소됩니다.

다음 표에 자세한 내용이 나와 있습니다.

모드 변경	보류 중 및 활성 실행 세부 정보	파이프라인 상태 세부 정보
SUPERSEDED에서 SUPERSEDED로/SUPERS EDED에서 QUEUED로	 진행 중인 작업이 완료된 후 활성 실행이 취소됩니다. 보류 중인 실행이 취소됩니다. 다. 	취소됨과 같은 파이프라인 상 태는 첫 번째 모드의 버전과 두 번째 모드 사이에 보존됩니다.
QUEUED에서 QUEUED로/Q UEUED에서 SUPERSEDED로	 진행 중인 작업이 완료된 후 활성 실행이 취소됩니다. 보류 중인 실행이 취소됩니 다. 	취소됨과 같은 파이프라인 상 태는 첫 번째 모드의 버전과 두 번째 모드 사이에 보존됩니다.
PARALLEL에서 PARALLEL로	모든 실행은 파이프라인 정의 업데이트와 독립적으로 실행할 수 있습니다.	비어 있습니다. 병렬 모드에는 파이프라인 상태가 없습니다.
SUPERSEDED에서 PARALLEL로/QUEUED에서 PARALLEL로	 진행 중인 작업이 완료된 후 활성 실행이 취소됩니다. 보류 중인 실행이 취소됩니다. 다. 	비어 있습니다. 병렬 모드에는 파이프라인 상태가 없습니다.

파이프라인 실행 모드 설정 또는 변경(콘솔)

콘솔을 사용하여 파이프라인 실행 모드를 설정할 수 있습니다.

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> <u>codepipeline/home</u>://https//https/

AWS 계정과 연결된 모든 파이프라인의 이름과 상태가 표시됩니다.

- 2. [Name]에서 편집할 파이프라인의 이름을 선택합니다.
- 3. 파이프라인 세부 정보 페이지에서 [Edit]를 선택합니다.
- 4. 편집 페이지에서 편집: 파이프라인 속성을 선택합니다.
- 5. 파이프라인의 모드를 선택합니다.
 - 대체됨
 - 대기 중(파이프라인 유형 V2 필요)

• 병렬(파이프라인 유형 V2 필요)

6. 편집 페이지에서 완료를 선택합니다.

파이프라인 실행 모드 설정(CLI)

AWS CLI 를 사용하여 파이프라인 실행 모드를 설정하려면 create-pipeline 또는 updatepipeline 명령을 사용합니다.

 터미널 세션(Linux, macOS 또는 Unix) 또는 명령 프롬프트(Windows)를 열고 get-pipeline 명령을 실행하여 파이프라인 구조를 JSON 파일에 복사합니다. 예를 들어, MyFirstPipeline라는 파이 프라인에서는 다음 명령을 입력합니다.

aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json

이 명령은 아무 것도 반환하지 않지만 생성한 파일이 명령을 실행한 디렉터리에 표시되어야 합니 다.

 일반 텍스트 편집기에서 JSON 파일을 열고 파일의 구조를 수정하여 QUEUED와 같이 설정하려는 파이프라인 실행 모드를 반영합니다.

"executionMode": "QUEUED"

다음 예제는 두 스테이지가 있는 예제 파이프라인에서 실행 모드를 QUEUED로 설정하는 방법을 보여줍니다.

```
"actionTypeId": {
                "category": "Source",
                "owner": "AWS",
                "provider": "CodeCommit",
                "version": "1"
            },
            "runOrder": 1,
            "configuration": {
                "BranchName": "main",
                "OutputArtifactFormat": "CODE_ZIP",
                "PollForSourceChanges": "true",
                "RepositoryName": "MyDemoRepo"
            },
            "outputArtifacts": [
                {
                    "name": "SourceArtifact"
                }
            ],
            "inputArtifacts": [],
            "region": "us-east-1",
            "namespace": "SourceVariables"
        }
    ]
},
{
    "name": "Build",
    "actions": [
        {
            "name": "Build",
            "actionTypeId": {
                "category": "Build",
                "owner": "AWS",
                "provider": "CodeBuild",
                "version": "1"
            },
            "runOrder": 1,
            "configuration": {
                "ProjectName": "MyBuildProject"
            },
            "outputArtifacts": [
                {
                    "name": "BuildArtifact"
                }
            ],
```

3. get-pipeline 명령을 사용하여 검색한 파이프라인 구조로 작업 중인 경우, JSON 파일의 구조를 수정해야 합니다. update-pipeline 명령이 JSON 파일을 사용할 수 있도록 하려면 이 파일에서 metadata 라인을 삭제해야 합니다. JSON 파일의 파이프라인 구조에서 단원("metadata":

 {] 행과 "created", "pipelineARN" 및 "updated" 필드)을 삭제합니다.

예를 들어, 구조에서 다음 행을 삭제합니다.

```
"metadata": {
   "pipelineArn": "arn:aws:codepipeline:region:account-ID:pipeline-name",
   "created": "date",
   "updated": "date"
}
```

파일을 저장합니다.

4. 변경 사항을 적용하려면 파이프라인 JSON 파일을 지정하여 update-pipeline 명령을 실행합니다.

▲ Important 파일 이름 앞에 file://를 포함해야 합니다. 이 명령에 필수적입니다.

aws codepipeline update-pipeline --cli-input-json file://pipeline.json

이 명령은 편집한 파이프라인의 전체 구조를 반환합니다.

Note

update-pipeline 명령을 실행하면 파이프라인이 중지됩니다. update-pipeline 명령을 실행 할 때 파이프라인을 통해 개정을 실행하는 중이라면 해당 실행이 중지됩니다. 업데이트된 파이프라인을 통해 해당 개정을 실행하려면 파이프라인을 수동으로 시작해야 합니다.

롤백 또는 재시도 스테이지

AWS CodePipeline 콘솔 또는를 사용하여 스테이지 또는 스테이지의 작업을 수동으로 롤백하거나 재 시도 AWS CLI 할 수 있습니다. 롤백 또는 재시도를 구성된 결과로 사용하려는 단계의 조건을 구성하려 면 스테이지에 대한 조건 구성 섹션을 참조하세요.

주제

- 실패한 스테이지 또는 실패한 작업에 대해 스테이지 재시도 구성
- 스테이지 롤백 구성

실패한 스테이지 또는 실패한 작업에 대해 스테이지 재시도 구성

파이프라인을 처음부터 다시 실행하지 않고 실패한 단계에서 재시도할 수 있습니다. 이렇게 하려면 단 계에서 실패한 작업을 다시 시도하거나 단계의 첫 번째 작업부터 시작하여 단계의 모든 작업을 다시 시 도하면 됩니다. 한 단계에서 실패한 작업을 다시 시도하면 아직 진행 중인 모든 작업이 계속 작동하고 실패한 작업이 다시 트리거됩니다. 단계의 첫 번째 작업부터 실패한 단계를 다시 시도하면 해당 단계에 는 진행 중인 작업이 있을 수 없습니다. 단계를 재시도하려면 먼저 모든 작업이 실패하거나 일부 작업 이 실패하고 일부는 성공해야 합니다.

A Important

실패한 단계를 다시 시도하면 단계의 첫 번째 작업부터 해당 단계의 모든 작업을 다시 시도하 고, 실패한 작업을 다시 시도하면 단계에서 실패한 모든 작업이 다시 시도됩니다. 이렇게 하면 동일한 실행에서 이전에 성공한 작업의 출력 아티팩트가 재정의됩니다. 아티팩트가 재정의될 수 있지만 이전에 성공한 작업의 실행 기록은 계속 보존됩니다.

콘솔을 이용해 파이프라인을 보는 경우 단계 재시도 버튼 또는 실패한 작업 재시도 버튼을 누르면 재시 도할 수 있는 단계가 나타납니다.

AWS CLI를 사용하는 경우 get-pipeline-state 명령을 사용하여 작업이 실패했는지 확인할 수 있습니다.

Note

다음의 경우 단계를 재시도할 수 없을 수도 있습니다.

• 단계의 모든 작업이 성공했으므로 해당 단계는 실패 상태가 아닙니다.

- 단계 실패 후 전체 파이프라인 구조가 변경되었습니다.
- 그 단계에서 또 다른 재시도가 이미 진행 중입니다.

주제

- 스테이지 재시도 시 고려 사항
- 실패한 스테이지를 수동으로 재시도
- 실패 시 자동 재시도를 위한 스테이지 구성

스테이지 재시도 시 고려 사항

스테이지 재시도에 대한 고려 사항은 다음과 같습니다.

- 재시도 횟수 한 번에 스테이지 실패 시 자동 재시도를 구성할 수 있습니다.
- Source 작업을 포함한 모든 작업에 대해 스테이지 실패 시 자동 재시도를 구성할 수 있습니다.

실패한 스테이지를 수동으로 재시도

콘솔 또는 CLI를 사용하여 실패한 단계를 수동으로 재시도할 수 있습니다.

실패 시 자동 재시도를 위한 스테이지 구성에 설명된 대로 실패 시 자동으로 재시도하도록 스테이지를 구성할 수도 있습니다.

실패한 단계 수동으로 재시도(콘솔)

실패한 단계 또는 단계에서 실패한 작업을 재시도하려면 - 콘솔

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https//https//https://https//ht

AWS 계정과 연결된 모든 파이프라인의 이름이 표시됩니다.

- 2. 이름에서 파이프라인의 이름을 선택합니다.
- 3. 실패한 작업이 있는 단계를 찾은 다음, 다음 중 하나를 선택합니다.
 - 단계의 모든 작업을 재시도하려면 단계 재시도를 선택합니다.
 - 단계에서 실패한 작업만 다시 시도하려면 실패한 작업 재시도를 선택합니다.

Retry stage Retry failed actions	;
build	G
AWS CodeBuild	0
Succeeded - 12 minutes and	
Details	
View in AWS CodeBuild	
\checkmark	
Deploy	۵
AWS CodeDeploy	U
Realled - Just now	
Details	
View in AWS CodeDeploy	

단계의 다시 시도된 모든 작업이 성공적으로 완료되면 파이프라인이 계속하여 실행됩니다.

실패한 스테이지를 수동으로 재시도(CLI)

실패한 단계 또는 단계에서 실패한 작업을 재시도하려면 - CLI

AWS CLI 를 사용하여 모든 작업 또는 실패한 모든 작업을 재시도하려면 다음 파라미터와 함께 retrystage-execution 명령을 실행합니다.

--pipeline-name <value>

```
사용자 가이드
```

```
--stage-name <value>
```

```
--pipeline-execution-id <value>
```

--retry-mode ALL_ACTIONS/FAILED_ACTIONS

1 Note

retry-mode에 사용할 수 있는 값은 FAILED_ACTIONS 및 ALL_ACTIONS입니다.

1. 터미널(Linux, macOS 또는 Unix) 또는 명령 프롬프트(Windows)에서, MyPipeline이라는 파이프 라인의 다음 예제에서와 같이 retry-stage-execution 명령을 실행합니다.

aws codepipeline retry-stage-execution --pipeline-name MyPipeline --stage-name Deploy --pipeline-execution-id b59babff-5f34-EXAMPLE --retry-mode FAILED_ACTIONS

출력이 실행 ID를 반환합니다.

```
{
    "pipelineExecutionId": "b59babff-5f34-EXAMPLE"
}
```

- 2. JSON 입력 파일을 사용하여 명령을 실행할 수도 있습니다. 파이프라인, 실패한 작업이 들어 있는 단계, 해당 단계의 마지막 파이프라인 실행을 식별하는 JSON 파일을 먼저 만듭니다. --cli-input-json 파라미터와 함께 retry-stage-execution 명령을 실행합니다. JSON 파일에 필요한 세부 정보를 가져오려면 get-pipeline-state 명령을 사용하는 것이 가장 쉽습니다.
 - a. 터미널(Linux, macOS 또는 Unix) 또는 명령 프롬프트(Windows)에서, 파이프라인에 <u>get-</u> <u>pipeline-state</u> 명령을 실행합니다. 예를 들어 MyFirstPipeline이라는 파이프라인의 경우 다음 과 유사한 명령을 입력합니다.

aws codepipeline get-pipeline-state --name MyFirstPipeline

명령에 대한 응답에는 각 단계의 파이프라인 상태 정보가 포함됩니다. 다음 예에서는 응답이 하나 이상의 작업이 스테이징 단계에서 실패했음을 나타냅니다.

```
{
    "updated": 1427245911.525,
    "created": 1427245911.525,
    "pipelineVersion": 1,
```

```
"pipelineName": "MyFirstPipeline",
    "stageStates": [
        {
            "actionStates": [...],
            "stageName": "Source",
            "latestExecution": {
                "pipelineExecutionId": "9811f7cb-7cf7-SUCCESS",
                "status": "Succeeded"
            }
        },
        ſ
            "actionStates": [...],
            "stageName": "Staging",
            "latestExecution": {
                "pipelineExecutionId": "3137f7cb-7cf7-EXAMPLE",
                "status": "Failed"
            }
        }
    ]
}
```

b. 일반 텍스트 편집기에서, 다음 내용을 기록할 파일을 JSON 형식으로 생성합니다.

- 실패한 작업이 들어 있는 파이프라인의 이름
- 실패한 작업이 들어 있는 단계의 이름
- 단계의 마지막 파이프라인 실행 ID
- 다시 시도 모드.

앞의 MyFirstPipeline 예의 경우 파일이 다음과 비슷해야 합니다.

```
{
    "pipelineName": "MyFirstPipeline",
    "stageName": "Staging",
    "pipelineExecutionId": "3137f7cb-7cf7-EXAMPLE",
    "retryMode": "FAILED_ACTIONS"
}
```

- c. retry-failed-actions.json과 같은 이름으로 파일을 저장합니다.
- d. retry-stage-execution 명령을 실행할 때 생성한 파일을 호출합니다. 예시:

Important

파일 이름 앞에 file://를 포함해야 합니다. 이 명령에 필수적입니다.

aws codepipeline retry-stage-execution --cli-input-json file://retry-failedactions.json

e. 다시 시도한 결과를 보려면 CodePipeline 콘솔을 열고 실패한 작업이 들어 있는 파이프라인을 선택하거나 get-pipeline-state 명령을 다시 사용합니다. 자세한 내용은 <u>CodePipeline에서 파</u> 이프라인 및 세부 정보 보기 단원을 참조하십시오.

실패 시 자동 재시도를 위한 스테이지 구성

실패 시 자동 재시도를 위한 스테이지를 구성할 수 있습니다. 스테이지가 한 번 재시도를 시도하고 실 패한 스테이지의 재시도 상태를 파이프라인 보기 페이지에 표시합니다.

실패한 스테이지의 모든 작업을 자동으로 재시도하거나 스테이지에서 실패한 작업만 재시도하도록 지 정하여 재시도 모드를 구성할 수 있습니다.

실패 시 자동 재시도를 위한 스테이지 구성(콘솔)

콘솔을 사용하여 자동 재시도를 위한 스테이지를 구성할 수 있습니다.

자동 재시도를 위한 스테이지 구성(콘솔)

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https//https

AWS 계정과 연결된 모든 파이프라인의 이름과 상태가 표시됩니다.

- 2. [Name]에서 편집할 파이프라인의 이름을 선택합니다.
- 3. 파이프라인 세부 정보 페이지에서 [Edit]를 선택합니다.
- 4. 편집 페이지에서 편집하려는 작업에 대해 스테이지 편집을 선택합니다.
- 자동화된 스테이지 구성:을 선택한 다음 스테이지 장애 시 자동 재시도 활성화를 선택합니다. 파이 프라인에 변경 사항을 저장합니다.
- 6. 자동화된 스테이지 구성:에서 다음 재시도 모드 중 하나를 선택합니다.

- 모드가 스테이지의 모든 작업을 재시도하도록 지정하려면 실패한 단계 재시도를 선택합니다.
- 모드가 스테이지에서 실패한 작업만 재시도하도록 지정하려면 실패한 작업 재시도를 선택합니다.

파이프라인에 변경 사항을 저장합니다.

Add entry condition 🔻	Add success condition 🔻	Add failure condi	tion	
+ Add action group]			
Build AWS CodeBuild	Ū	+ Add action		
	Enable automatic r	ollback on stage failure		
+ Add action group	Enable automatic r	etry on stage failure 🗸		
utomated stage configura	tion: Enable automatic r	etry on stage failure 🔺	Retry mode:	Retry failed stage

 파이프라인이 실행된 후 스테이지 실패가 발생하면 자동 재시도가 수행됩니다. 다음 예제에서는 자동으로 재시도된 빌드 단계를 보여줍니다.

Source			
GitHub (Version 2)			
Succeeded - <u>1 minu</u>	ute ago		
<u>d4002ba3</u>			
View details			
d4002ba3 🗹 Source: Mak	ing an update to Update	README.md	
Disable trans	ition		
🛞 Build Failed 🕖	Auto retry attempt Vie	w retry metadata	
Start rollback	Auto retry attempt Vie Retry stage	Retry failed actions	
Build Failed Start rollback Pipeline execution ID	Auto retry attempt Vie Retry stage : <u>ee4b9da8-62b4-</u>	w retry metadata Retry failed actions	
Build Failed Start rollback Pipeline execution ID	Auto retry attempt Vie Retry stage : <u>ee4b9da8-62b4-</u>	w retry metadata Retry failed actions	
Build Failed Start rollback Pipeline execution ID Build	Auto retry attempt Vie Retry stage : <u>ee4b9da8-62b4-</u>	w retry metadata Retry failed actions	
 Build Failed Start rollback Pipeline execution ID Build AWS CodeBuild 	Auto retry attempt Vie Retry stage : <u>ee4b9da8-62b4-</u>	w retry metadata Retry failed actions	
 Build Failed Start rollback Pipeline execution ID Build AWS CodeBuild Failed - <u>1 minute and 1000000000000000000000000000000000000</u>	Auto retry attempt Vie Retry stage : <u>ee4b9da8-62b4-</u>	w retry metadata Retry failed actions	
 Build Failed Start rollback Pipeline execution ID Build <u>AWS CodeBuild</u> Failed - <u>1 minute and</u> View details 	Auto retry attempt Vie Retry stage : <u>ee4b9da8-62b4-</u>	w retry metadata Retry failed actions	

8. 재시도에 대한 세부 정보를 보려면 다음을 선택합니다. 창이 표시됩니다.

Retry stage metada	ita	×
Pipeline Execution Id	ee4b9da8-62b4-	
Latest Retry Trigger	AutomatedStageRetry	
Auto Retry Attempt	1	
		Done

자동 재시도를 위한 스테이지 구성(CLI)

AWS CLI 를 사용하여 실패 시 자동으로 재시도하도록 단계를 구성하려면 <u>파이프라인 스테이지 및 작</u> <u>업 생성</u> 및에 설명된 대로 명령을 사용하여 파이프라인을 생성하거나 업데이트합니다<u>CodePipeline에</u> 서 파이프라인 편집. 터미널(Linux, macOS 또는 Unix) 또는 명령 프롬프트(Windows)를 열고 AWS CLI 를 사용하여 update-pipeline 명령을 실행함으로써 파이프라인 구조의 실패 조건을 지정합니다. 다음 예제 에서는 S3Deploy라는 이름의 스테이지에 대한 자동 재시도를 구성합니다.

```
{
                 "name": "S3Deploy",
                "actions": [
                     {
                         "name": "s3deployaction",
                         "actionTypeId": {
                             "category": "Deploy",
                             "owner": "AWS",
                             "provider": "S3",
                             "version": "1"
                         },
                         "runOrder": 1,
                         "configuration": {
                             "BucketName": "static-website-bucket",
                             "Extract": "false",
                             "ObjectKey": "SampleApp.zip"
                         },
                         "outputArtifacts": [],
                         "inputArtifacts": [
                             {
                                 "name": "SourceArtifact"
                             }
                         ],
                         "region": "us-east-1"
                     }
                ],
                  "onFailure": {
                     "result": "RETRY",
                     "retryConfiguration": {
                         "retryMode": "ALL_ACTIONS",
                     },
            }
```

자동 재시도를 위한 스테이지 구성(AWS CloudFormation)

AWS CloudFormation 를 사용하여 실패 시 자동 재시도를 위한 스테이지를 구성하려면 OnFailure 스테이지 수명 주기 파라미터를 사용합니다. RetryConfiguration 파라미터를 사용하여 재시도 모 드를 구성합니다.

OnFailure: Result: RETRY RetryConfiguration: RetryMode: ALL_ACTIONS

 다음 조각과 같이 템플릿을 업데이트합니다. 다음 예제에서는 Release라는 스테이지에 대한 자 동 재시도를 구성합니다.

```
AppPipeline:
 Type: AWS::CodePipeline::Pipeline
  Properties:
    RoleArn:
      Ref: CodePipelineServiceRole
    Stages:
        Name: Source
        Actions:
            Name: SourceAction
            ActionTypeId:
              Category: Source
              Owner: AWS
              Version: 1
              Provider: S3
            OutputArtifacts:
                Name: SourceOutput
            Configuration:
              S3Bucket:
                Ref: SourceS3Bucket
              S3ObjectKey:
                Ref: SourceS30bjectKey
            RunOrder: 1
        Name: Release
        Actions:
```

Name: ReleaseAction InputArtifacts: Name: SourceOutput ActionTypeId: Category: Deploy Owner: AWS Version: 1 Provider: CodeDeploy Configuration: ApplicationName: Ref: ApplicationName DeploymentGroupName: Ref: DeploymentGroupName RunOrder: 1 OnFailure: Result: RETRY RetryConfiguration: RetryMode: ALL_ACTIONS ArtifactStore: Type: S3 Location: Ref: ArtifactStoreS3Location EncryptionKey: Id: arn:aws:kms:useast-1:ACCOUNT-ID:key/KEY-ID Type: KMS DisableInboundStageTransitions: StageName: Release Reason: "Disabling the transition until integration tests are completed" Tags: - Key: Project Value: ProjectA - Key: IsContainerBased Value: 'true'

실패 시 스테이지 재시도 구성에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 StageDeclaration에서 <u>OnFailure</u>를 참조하세요.

스테이지 롤백 구성

해당 스테이지에서 성공한 실행으로 스테이지를 롤백할 수 있습니다. 실패 시 롤백을 위해 스테이지를 사전 구성하거나 스테이지를 수동으로 롤백할 수 있습니다. 롤백 작업을 수행하면 새 실행이 수행됩니 다. 롤백을 위해 선택한 대상 파이프라인 실행은 소스 개정 및 변수를 검색하는 데 사용됩니다.

표준 또는 롤백 실행 유형은 파이프라인 기록, 파이프라인 상태 및 파이프라인 실행 세부 정보에 표시 됩니다.

주제

- 롤백고려사항
- 스테이지를 수동으로 롤백
- 자동 롤백을 위한 스테이지 구성
- 실행 목록에서 롤백 상태 보기
- 롤백상태세부정보보기

롤백 고려 사항

스테이지 롤백에 대한 고려 사항은 다음과 같습니다.

- 소스 스테이지는 롤백할 수 없습니다.
- 파이프라인은 현재 파이프라인 구조 버전에서 이전 실행이 시작된 경우에만 이전 실행으로 롤백할 수 있습니다.
- 롤백 실행 유형인 대상 실행 ID로 롤백할 수 없습니다.
- CodePipeline은 롤백되는 실행의 변수와 아티팩트를 사용합니다.

스테이지를 수동으로 롤백

콘솔 또는 CLI를 사용하여 스테이지를 수동으로 롤백할 수 있습니다. 파이프라인은 현재 파이프라인 구조 버전에서 이전 실행이 시작된 경우에만 이전 실행으로 롤백할 수 있습니다.

<u>자동 롤백을 위한 스테이지 구성</u>에 설명된 대로 실패 시 자동으로 롤백하도록 스테이지를 구성할 수도 있습니다.

스테이지를 수동으로 롤백(콘솔)

콘솔을 사용하여 스테이지를 대상 파이프라인 실행으로 수동으로 롤백할 수 있습니다. 스테이지가 롤 백되면 콘솔의 파이프라인 시각화에 롤백 레이블이 표시됩니다.

스테이지를 수동으로 롤백(콘솔)

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home CodePipeline 콘솔을 엽니다.

AWS 계정과 연결된 모든 파이프라인의 이름과 상태가 표시됩니다.

2. 이름에서 롤백할 스테이지가 있는 파이프라인의 이름을 선택합니다.

Source AWS CodeCommit	
Succeeded - Just now 10cb9a83	
10cb9a83 Source: update	
Disable transition	
Odeploys3 Succeeded Pipeline execution ID: <u>d1b8bf31-1d2f-4133-98f8-6a104fee1b4f</u>	Start rollback
s3deploy	

3. 스테이지에서 롤백 시작을 선택합니다. 페이지로 롤백이 표시됩니다.

4. 스테이지를 롤백할 대상 실행을 선택합니다.

Note 사용 가능한 대상 파이프라인 실행 목록은 2024년 2월 1일부터 현재 파이프라인 버전의 모든 실행입니다.

13					×
Rolli	oack to				Start rollback
	Execution ID	Source revisions	Started	Duration	Completed
0	07c0ff1d	Source – <u>04b7e235</u> : Edited README.txt	4 days ago	5 seconds	4 days ago
0	f15b38f7	Source – <u>f823f4a3</u> : Edited README.txt	5 days ago	5 seconds	5 days ago
0	4f47bed9	Source – <u>73ae512c</u> : Added README.txt	8 days ago	5 seconds	8 days ago
0	fcd61d8b	Source – <u>73ae512c</u> : Added README.txt	8 days ago	7 seconds	8 days ago

다음 다이어그램은 새 실행 ID로 롤백된 스테이지의 예를 보여줍니다.



스테이지를 수동으로 롤백(CLI)

AWS CLI 를 사용하여 스테이지를 수동으로 롤백하려면 rollback-stage 명령을 사용합니다.

스테이지를 수동으로 롤백에 설명된 대로 스테이지를 수동으로 롤백할 수도 있습니다.

Note

사용 가능한 대상 파이프라인 실행 목록은 2024년 2월 1일부터 현재 파이프라인 버전의 모든 실행입니다.

스테이지를 수동으로 롤백하려면(CLI)

1. 수동 롤백을 위한 CLI 명령에는 스테이지에서 이전에 성공한 파이프라인 실행의 실행 ID가 필요합니다. 지정할 대상 파이프라인 실행 ID를 가져오려면 스테이지에서 성공적인 실행을 반환하는 필터와 함께 list-pipeline-executions 명령을 사용합니다. 터미널(Linux, macOS 또는 Unix) 또는 명령 프롬프트(Windows)를 열고를 사용하여 list-pipeline-executions 명령을 실행하고 파이프 라인의 이름과 단계에서 성공적으로 실행 AWS CLI 하기 위한 필터를 지정합니다. 이 예제에서는 출력에 MyFirstPipeline이라는 파이프라인의 파이프라인 실행과 deploys3라는 스테이지에서의 성공적인 실행이 나열됩니다.

aws codepipeline list-pipeline-executions --pipeline-name MyFirstPipeline --filter
succeededInStage={stageName=deploys3}

출력에서 롤백에 지정하려는 이전에 성공한 실행의 실행 ID를 복사합니다. 다음 단계에서는 이를 대상 실행 ID로 사용합니다.

2. 터미널(Linux, macOS 또는 Unix) 또는 명령 프롬프트(Windows)를 열고 AWS CLI 를 사용하여 rollback-stage 명령을 실행하고, 파이프라인의 이름과 스테이지의 이름, 롤백하려는 대상 실 행을 지정합니다. 예를 들어 MyFirstPipeline이라는 파이프라인에 배포라는 스테이지를 롤백 하려면 다음을 입력합니다.

aws codepipeline rollback-stage --pipeline-name MyFirstPipeline --stage-name Deploy
 --target-pipeline-execution-id bc022580-4193-491b-8923-9728dEXAMPLE

출력은 새 롤백 실행의 실행 ID를 반환합니다. 이는 지정된 대상 실행의 소스 개정 및 파라미터를 사용하는 별도의 ID입니다.

자동 롤백을 위한 스테이지 구성

실패 시 자동으로 롤백하도록 파이프라인의 단계를 구성할 수 있습니다. 스테이지가 실패하면 스테이 지가 가장 최근에 성공한 실행으로 롤백됩니다. 파이프라인은 현재 파이프라인 구조 버전에서 이전 실 행이 시작된 경우에만 이전 실행으로 롤백할 수 있습니다. 자동 롤백 구성은 파이프라인 정의의 일부이 므로 파이프라인 단계에서 파이프라인이 성공적으로 실행된 후에만 파이프라인 단계가 자동으로 롤백 됩니다.

자동 롤백을 위한 스테이지 구성(콘솔)

스테이지를 지정된 이전에 성공한 실행으로 롤백할 수 있습니다. 자세한 내용은 CodePipeline API 가 이드의 RollbackStage를 참조하세요.

자동 롤백을 위한 스테이지 구성(콘솔)

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> <u>codepipeline/home</u> CodePipeline 콘솔을 엽니다.

AWS 계정과 연결된 모든 파이프라인의 이름과 상태가 표시됩니다.

- 2. [Name]에서 편집할 파이프라인의 이름을 선택합니다.
- 3. 파이프라인 세부 정보 페이지에서 [Edit]를 선택합니다.
- 4. 편집 페이지에서 편집하려는 작업에 대해 스테이지 편집을 선택합니다.
- 5. 자동화된 스테이지 구성:을 선택한 다음 스테이지 장애 시 자동 롤백 구성을 선택합니다. 파이프라 인에 변경 사항을 저장합니다.

Edit: deploys3			Cancel	Delete Done
+ Add action group				
s3deploy Amazon S3	١	+ Add action		
	∠ ×			
Add action group Configure automatic rollb	ack on stage failure			

자동 롤백을 위한 스테이지 구성(CLI)

AWS CLI 를 사용하여 가장 최근에 성공한 실행으로 자동 롤백하도록 실패한 단계를 구성하려면 <u>파이</u> <u>프라인 스테이지 및 작업 생성</u> 및에 설명된 대로 명령을 사용하여 파이프라인을 생성하거나 업데이트 합니다CodePipeline에서 파이프라인 편집.

• 터미널(Linux, macOS 또는 Unix) 또는 명령 프롬프트(Windows)를 열고 AWS CLI 를 사용하여 update-pipeline 명령을 실행함으로써 파이프라인 구조의 실패 조건을 지정합니다. 다음 예제 에서는 S3Deploy라는 이름의 스테이지에 대한 자동 롤백을 구성합니다.

```
{
                "name": "S3Deploy",
                "actions": [
                     ſ
                         "name": "s3deployaction",
                         "actionTypeId": {
                             "category": "Deploy",
                             "owner": "AWS",
                             "provider": "S3",
                             "version": "1"
                         },
                         "runOrder": 1,
                         "configuration": {
                             "BucketName": "static-website-bucket",
                             "Extract": "false",
                             "ObjectKey": "SampleApp.zip"
                         },
                         "outputArtifacts": [],
                         "inputArtifacts": [
                             {
                                 "name": "SourceArtifact"
                             }
                         ],
                         "region": "us-east-1"
                     }
                ],
                "onFailure": {
                     "result": "ROLLBACK"
                }
            }
```

스테이지 롤백의 실패 조건 구성에 대한 자세한 내용은 CodePipeline API 참조의 FailureConditions을 참조하세요.

자동 롤백을 위한 스테이지 구성(AWS CloudFormation)

AWS CloudFormation 를 사용하여 실패 시 자동으로 롤백하도록 스테이지를 구성하려면 0nFailure 파라미터를 사용합니다. 실패 시 스테이지는 가장 최근에 성공한 실행으로 자동으로 롤백됩니다.

```
OnFailure:
Result: ROLLBACK
```

 다음 조각과 같이 템플릿을 업데이트합니다. 다음 예제에서는 Release라는 이름의 스테이지에 대한 자동 롤백을 구성합니다.

```
AppPipeline:
  Type: AWS::CodePipeline::Pipeline
  Properties:
    RoleArn:
      Ref: CodePipelineServiceRole
    Stages:
        Name: Source
        Actions:
            Name: SourceAction
            ActionTypeId:
              Category: Source
              Owner: AWS
              Version: 1
              Provider: S3
            OutputArtifacts:
                Name: SourceOutput
            Configuration:
              S3Bucket:
                Ref: SourceS3Bucket
              S3ObjectKey:
                Ref: SourceS30bjectKey
            RunOrder: 1
        Name: Release
```

```
Actions:
        Name: ReleaseAction
        InputArtifacts:
            Name: SourceOutput
        ActionTypeId:
          Category: Deploy
          Owner: AWS
          Version: 1
          Provider: CodeDeploy
        Configuration:
          ApplicationName:
            Ref: ApplicationName
          DeploymentGroupName:
            Ref: DeploymentGroupName
        RunOrder: 1
   OnFailure:
        Result: ROLLBACK
ArtifactStore:
  Type: S3
  Location:
    Ref: ArtifactStoreS3Location
  EncryptionKey:
    Id: arn:aws:kms:useast-1:ACCOUNT-ID:key/KEY-ID
    Type: KMS
DisableInboundStageTransitions:
    StageName: Release
    Reason: "Disabling the transition until integration tests are completed"
Tags:
  - Key: Project
    Value: ProjectA
  - Key: IsContainerBased
    Value: 'true'
```

스테이지 롤백의 실패 조건 구성에 대한 자세한 내용은 AWS CloudFormation 사용 설명서에서 StageDeclaration 아래의 OnFailure를 참조하세요.

실행 목록에서 롤백 상태 보기

롤백 실행의 상태 및 대상 실행 ID를 볼 수 있습니다.

실행 목록에서 롤백 상태 보기(콘솔)

콘솔을 사용하여 실행 목록에서 롤백 실행의 상태 및 대상 실행 ID를 볼 수 있습니다.

실행 목록에서 롤백 실행 상태 보기(콘솔)

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> <u>codepipeline/home</u> CodePipeline 콘솔을 엽니다.

사용자의 AWS 계정 에 연결된 모든 파이프라인의 이름과 상태가 표시됩니다.

- 2. 이름에서 확인하려는 파이프라인의 이름을 선택합니다.
- 3. History(기록)를 선택합니다. 실행 목록에 롤백 레이블이 표시됩니다.

Exec	ution history	Info		Rerun Sto	p execution	View details	Release change
٩	I						< 1 > ©
	Execution ID	Status	Source revisions	Trigger	Started	Duration	Completed
0	5cd064ca Rollback	🛞 Failed	Source – 04b7e235: Edited README.txt	Automated Rollback FailedPipeli neExecution Id - b2e77fa5	Apr 24, 2024 12:19 PM (UTC-7:00)	1 second	Apr 24, 2024 12:19 PM (UTC-7:00)
0	b2e77fa5	⊗ Failed	Source – 10cb9a83: update	StartPipelin eExecution	Apr 24, 2024 12:19 PM (UTC-7:00)	5 seconds	Apr 24, 2024 12:19 PM (UTC-7:00)
0	5efcfa68 Rollback	⊘ Succeeded	Source – 04b7e235: Edited README.txt	ManualRoll back -	Apr 24, 2024 12:16 PM (UTC-7:00)	2 seconds	Apr 24, 2024 12:16 PM (UTC-7:00)
0	d1b8bf31	⊘ Succeeded	Source – 10cb9a83: update	StartPipelin eExecution	Apr 24, 2024 12:14 PM (UTC-7:00)	6 seconds	Apr 24, 2024 12:14 PM (UTC-7:00)

세부 정보를 보고자 하는 실행 ID를 선택합니다.

list-pipeline-executions(CLI)를 사용하여 롤백 상태 보기

CLI를 사용하여 롤백 실행의 상태 및 대상 실행 ID를 볼 수 있습니다.

• 터미널(Linux, macOS, Unix) 또는 명령 프롬프트(Windows)를 열고 AWS CLI 를 사용하여 listpipeline-executions 명령을 실행합니다.

aws codepipeline list-pipeline-executions --pipeline-name MyFirstPipeline

이 명령은 파이프라인과 연결되어 있는 완료된 모든 실행의 목록을 반환합니다.

다음 예에서는 롤백 실행으로 파이프라인이 시작된 *MyFirstPipeline*이라는 파이프라인에 대해 반환된 데이터를 보여 줍니다.

```
{
    "pipelineExecutionSummaries": [
        {
            "pipelineExecutionId": "eb7ebd36-353a-4551-90dc-18ca5EXAMPLE",
            "status": "Succeeded",
            "startTime": "2024-04-16T09:00:28.185000+00:00",
            "lastUpdateTime": "2024-04-16T09:00:29.665000+00:00",
            "sourceRevisions": [
                {
                    "actionName": "Source",
                    "revisionId": "revision_ID",
                    "revisionSummary": "Added README.txt",
                    "revisionUrl": "console-URL"
                }
            ],
            "trigger": {
                "triggerType": "ManualRollback",
                "triggerDetail": "{arn:aws:sts::<account_ID>:assumed-role/<role>"}"
            },
            "executionMode": "SUPERSEDED",
            "executionType": "ROLLBACK",
            "rollbackMetadata": {
                "rollbackTargetPipelineExecutionId":
 "f15b38f7-20bf-4c9e-94ed-2535eEXAMPLE"
            }
        },
        {
            "pipelineExecutionId": "fcd61d8b-4532-4384-9da1-2aca1EXAMPLE",
            "status": "Succeeded",
            "startTime": "2024-04-16T08:58:56.601000+00:00",
            "lastUpdateTime": "2024-04-16T08:59:04.274000+00:00",
            "sourceRevisions": [
                {
                    "actionName": "Source",
                    "revisionId": "revision_ID",
                    "revisionSummary": "Added README.txt",
                    "revisionUrl": "console_URL"
                }
            ],
```

```
"trigger": {
                "triggerType": "StartPipelineExecution",
                "triggerDetail": "arn:aws:sts::<account_ID>:assumed-role/<role>"
            },
            "executionMode": "SUPERSEDED"
       },
        {
            "pipelineExecutionId": "5cd064ca-bff7-425f-8653-f41d9EXAMPLE",
            "status": "Failed",
            "startTime": "2024-04-24T19:19:50.781000+00:00",
            "lastUpdateTime": "2024-04-24T19:19:52.119000+00:00",
            "sourceRevisions": [
                {
                    "actionName": "Source",
                    "revisionId": "<revision_ID>",
                    "revisionSummary": "Edited README.txt",
                    "revisionUrl": "<revision_URL>"
                }
            ],
            "trigger": {
                "triggerType": "AutomatedRollback",
                "triggerDetail": "{\"FailedPipelineExecutionId\":
\"b2e77fa5-9285-4dea-ae66-4389EXAMPLE\"}"
            },
            "executionMode": "SUPERSEDED",
            "executionType": "ROLLBACK",
            "rollbackMetadata": {
                "rollbackTargetPipelineExecutionId": "5efcfa68-d838-4ca7-
a63b-4a743EXAMPLE"
            }
         },
```

롤백 상태 세부 정보 보기

롤백 실행의 상태 및 대상 실행 ID를 볼 수 있습니다.

세부 정보 페이지에서 롤백 상태 보기(콘솔)

콘솔을 사용하여 롤백 실행의 상태 및 대상 파이프라인 실행 ID를 볼 수 있습니다.

롤백된 파이프라인 실행은 파이프라인 실행을 가져오기 위한 출력에 표시됩니다.

get-pipeline-execution(CLI)을 사용하여 롤백 세부 정보 보기

Pipeline execution:	01cc ⁻	Rerun Stop	execution	Previous execution	Next execution	>
Execution summary						
Status	Started 1 hour ago	Complet 1 hour ag	ed go	Duration 1 second		
ManualRollback - Latest action execution message Deployment Succeeded Pipeline execution ID	Z					
Olccf652-ab11-4d4b-898c-94 Execution type ROLLBACK	173ef8521ba					
Target pipeline execution ID Target pipeline execution ID f15b38f7-20bf-4c9e-94ed-25	35ee02					0
Visualization Timeline	Variables Revisions					
O Source Didn't Run						
Source (i) AWS CodeCommit						
No executions yet						
Ļ						
⊘ deploys3 Succeeded					Start rollback	

Developer Tools > CodePipeline > Pipelines > rbtest > Execution history > 01ccf

파이프라인에 대한 세부 정보를 보려면 파이프라인의 고유 이름을 지정한 채 <u>get-pipeline-</u> <u>execution</u> 명령을 실행합니다. 예를 들어, *MyFirstPipeline*이라는 파이프라인의 세부 정보를 보려면 다음을 입력합니다.

aws codepipeline get-pipeline-execution --pipeline-name MyFirstPipeline --pipelineexecution-id 3f658bd1-69e6-4448-ba3e-79007EXAMPLE

이 명령은 파이프라인의 구조를 반환합니다.

다음 예제에서는 롤백 실행 ID와 메타데이터가 표시되는 *MyFirstPipeline*이라는 파이프라인 의 일부에 대해 반환된 데이터를 보여줍니다.

```
{
    "pipelineExecution": {
        "pipelineName": "MyFirstPipeline",
        "pipelineVersion": 6,
        "pipelineExecutionId": "2004a94e-8b46-4c34-a695-c8d20EXAMPLE",
        "status": "Succeeded",
        "artifactRevisions": [
            {
                "name": "SourceArtifact",
                "revisionId": "<ID>",
                "revisionSummary": "Added README.txt",
                "revisionUrl": "<console_URL>"
            }
        ],
        "trigger": {
            "triggerType": "ManualRollback",
            "triggerDetail": "arn:aws:sts::<account_ID>:assumed-role/<role>"
        },
        "executionMode": "SUPERSEDED",
        "executionType": "ROLLBACK",
        "rollbackMetadata": {
            "rollbackTargetPipelineExecutionId": "4f47bed9-6998-476c-a49d-
e60beEXAMPLE"
        }
    }
}
```

get-pipeline-state(CLI)를 사용하여 롤백 상태 보기

롤백된 파이프라인 실행은 파이프라인 상태를 가져오기 위한 출력에 표시됩니다.

 파이프라인에 대한 세부 정보를 보려면 파이프라인의 고유 이름을 지정한 채 get-pipeline-state 명 령을 실행합니다. 예를 들어, *MyFirstPipeline*이라는 파이프라인의 세부 정보를 보려면 다음 을 입력합니다.

aws codepipeline get-pipeline-state --name MyFirstPipeline

다음 예제에서는 롤백 실행 유형이 있는 반환된 데이터를 보여줍니다.

```
{
    "pipelineName": "MyFirstPipeline",
    "pipelineVersion": 7,
    "stageStates": [
        {
            "stageName": "Source",
            "inboundExecutions": [],
            "inboundTransitionState": {
                "enabled": true
            },
            "actionStates": [
                {
                    "actionName": "Source",
                    "currentRevision": {
                        "revisionId": "<Revision_ID>"
                    },
                    "latestExecution": {
                        "actionExecutionId": "13bbd05d-
b439-4e35-9c7e-887cb789b126",
                        "status": "Succeeded",
                        "summary": "update",
                        "lastStatusChange": "2024-04-24T20:13:45.799000+00:00",
                        "externalExecutionId": "10cbEXAMPLEID"
                    },
                    "entityUrl": "console-url",
                    "revisionUrl": "console-url"
                }
            ],
            "latestExecution": {
                "pipelineExecutionId": "cf95a8ca-0819-4279-ae31-03978EXAMPLE",
```

```
"status": "Succeeded"
            }
        },
        {
            "stageName": "deploys3",
            "inboundExecutions": [],
            "inboundTransitionState": {
                "enabled": true
            },
            "actionStates": [
                {
                    "actionName": "s3deploy",
                    "latestExecution": {
                         "actionExecutionId":
 "3bc4e3eb-75eb-45b9-8574-8599aEXAMPLE",
                        "status": "Succeeded",
                        "summary": "Deployment Succeeded",
                        "lastStatusChange": "2024-04-24T20:14:07.577000+00:00",
                        "externalExecutionId": "mybucket/SampleApp.zip"
                    },
                    "entityUrl": "console-URL"
                }
            ],
            "latestExecution": {
                "pipelineExecutionId": "fdf6b2ae-1472-4b00-9a83-1624eEXAMPLE",
                "status": "Succeeded",
                "type": "ROLLBACK"
            }
        }
    ],
    "created": "2024-04-15T21:29:01.635000+00:00",
    "updated": "2024-04-24T20:12:24.480000+00:00"
}
```

스테이지에 대한 조건 구성

파이프라인 실행에서 특정 변수를 확인하는 등 스테이지에 대한 조건을 지정한 다음, 스테이지를 건너 뛰거나 스테이지를 실패하는 등의 조건에 대한 결과를 적용할 수 있습니다. 파이프라인을 설정하여 실 행 중에 스테이지 조건을 확인할 수 있습니다. 여기서 스테이지에 대한 검사를 지정한 다음 특정 조건 이 충족될 때 스테이지를 계속하는 방법을 지정합니다. 조건에는 CodePipeline의 규칙 목록에서 사용 할 수 있는 하나 이상의 규칙이 포함됩니다. 조건의 모든 규칙이 성공하면 조건이 충족됩니다. 기준이 충족되지 않으면 지정된 결과가 적용되도록 조건을 구성할 수 있습니다.

각 조건에는 함께 평가되는 규칙의 순서 집합인 규칙 집합이 있습니다. 따라서 조건에서 하나의 규칙이 실패하면 조건이 실패합니다. 파이프라인 런타임 시 규칙 조건을 재정의할 수 있습니다.

조건은 특정 유형의 표현식에 사용되며 각 표현식에는 다음과 같이 사용할 수 있는 결과에 대한 특정 옵션이 있습니다.

- 입력 충족되는 경우 스테이지에 대한 진입을 허용하는 검사 수행 조건입니다. 규칙은 실패 또는 건 너뛰기 결과 옵션과 함께 사용됩니다.
- 실패 시 스테이지가 실패할 때 스테이지를 확인하기 위한 조건입니다. 규칙은 롤백 결과 옵션과 함 께 사용됩니다.
- 성공 시 스테이지가 성공할 때 스테이지를 확인하기 위한 조건입니다. 규칙은 롤백 또는 실패와 같
 은 결과 옵션과 함께 사용됩니다.

조건은 각 조건 유형에 대한 규칙 집합에서 지원됩니다.

각 조건 유형에는 조건에 따라 설정된 특정 작업이 있습니다. 작업은 성공 또는 실패한 조건 확인의 결 과입니다. 예를 들어 입력 조건(입력 조건)에 경보(규칙)가 발생하면 확인이 성공하고 결과(작업)는 스 테이지 항목이 차단된 것입니다.

AWS CodePipeline 콘솔 또는를 사용하여 단계의 단계 또는 작업을 수동으로 롤백하거나 재시도 AWS CLI 할 수도 있습니다. 스테이지에 대한 조건 구성을(를) 참조하세요.

주제

- 스테이지 조건에 대한 사용 사례
- 스테이지 조건에 대해 구성된 결과 고려 사항
- 스테이지 조건에 대해 구성된 규칙 고려 사항
- 입력 조건 생성
- 실패 시 조건 생성
- 성공 시 조건 생성
- 스테이지 조건 삭제
- 스테이지 조건 재정의

스테이지 조건에 대한 사용 사례

스테이지 조건에는 파이프라인에서 릴리스 및 변경 안전을 설정하는 데 사용되는 여러 사용 사례가 있 습니다. 다음은 스테이지 조건에 대한 샘플 사용 사례입니다.

- 입력 조건을 사용하여 CloudWatch 경보 상태를 확인하는 조건을 정의하면 프로덕션 환경이 정상 상 태가 아닌 경우 변경 사항이 차단됩니다.
- 대기 시간이 60인 입력 조건을 사용하여 단계의 모든 작업이 성공적으로 완료되었을 때 평가할 조건 을 정의한 다음, CloudWatch 경보가 60분 이내에 ALARM 상태가 되면 변경 사항을 롤백합니다.
- 성공 시 조건을 사용하여 스테이지가 성공적으로 완료되면 규칙이 현재 시간이 배포 창에 있는지 확 인한 다음, 규칙이 성공하면 배포되도록 조건을 정의합니다.

스테이지 조건에 대해 구성된 결과 고려 사항

스테이지 조건에 대한 고려 사항은 다음과 같습니다.

- onFailure 조건에서는 자동 스테이지 재시도를 사용할 수 없습니다.
- 롤백 결과로 조건을 구성할 때 현재 파이프라인 구조 버전에서 사용 가능한 경우에만 스테이지가 이 전 실행으로 롤백될 수 있습니다.
- 롤백 결과로 조건을 구성할 때는 롤백 실행 유형인 대상 실행 ID로 롤백할 수 없습니다.
- 조건이 실패하면 건너뛰기 결과를 사용하여 스테이지를 건너뛰는 입력 조건의 경우 LambdaInvoke 및 VariableCheck 규칙만 지원됩니다.
- 건너뛴 상태의 스테이지에서는 수동 스테이지 재시도를 수행할 수 없습니다.
- 건너뛴 상태의 스테이지로 수동 롤백을 수행할 수 없습니다.
- 조건이 건너뛰기 결과로 구성된 경우 조건을 재정의할 수 없습니다.
- 건너뛰기 결과를 제외하고 파이프라인 실행을 시작할 때 스테이지 조건을 재정의할 수 있습니다. 재 정의를 사용하는 스테이지 조건의 경우 실행은 다음 표에 설명된 대로 수행됩니다.

유형	조건 실패 시 구성된 결과	스테이지 상태	동작 재정의
입력	Fail	진행 중	스테이지가 진행됩니 다.
입력	건너뛰기	건너뜀	해당 사항 없음.
OnFailure	롤백	Failed	스테이지가 실패했습 니다.
OnSuccess	롤백	성공	스테이지가 진행됩니 다.
OnSuccess	Fail	Failed	스테이지가 진행됩니 다.

스테이지 조건에 대해 구성된 규칙 고려 사항

스테이지 조건에 대해 가능한 규칙의 고려 사항은 다음과 같습니다.

- LambdaInvoke 규칙의 경우 먼저 규칙에 사용할 Lambda 함수를 구성해야 합니다. 규칙을 구성할 때 Lambda 함수 ARN을 제공할 준비를 합니다.
- CloudWatchAlarm 규칙의 경우 먼저 규칙에 사용할 CloudWatch Events 이벤트를 구성해야 합니다. 규칙을 구성할 때 이벤트 ARN을 제공할 준비를 합니다.

입력 조건 생성

콘솔 또는 CLI를 사용하여 스테이지의 입력 조건을 구성할 수 있습니다. 각 조건에 해당하는 규칙과 결 과를 구성합니다. 롤백 결과의 경우 파이프라인은 현재 파이프라인 구조 버전에서 이전 실행이 시작된 경우에만 이전 실행으로 롤백할 수 있습니다.

이 단계에서는 모니터 규칙을 사용하는 예제 입력 조건을 제공합니다.

자세한 내용은 CodePipeline API 가이드의 <u>조건</u>, <u>RuleTypeId</u> 및 <u>RuleExecution</u>을 참조하세요.

입력 조건 생성 - CloudWatchAlarm 규칙 예제(콘솔)

조건이 충족될 때 스테이지가 수행할 규칙 및 결과와 함께 스테이지에 대한 입력 조건을 구성할 수 있 습니다.

입력 조건 구성(콘솔)

- 1. AWS CloudWatchAlarm과 같이 리소스가 제공되는 규칙에 대한 리소스 및 ARN 생성과 같은 사전 조건을 완료합니다.
- 2. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home CodePipeline 콘솔을 엽니다.

사용자의 AWS 계정 에 연결된 모든 파이프라인의 이름과 상태가 표시됩니다.

- 3. [Name]에서 편집할 파이프라인의 이름을 선택합니다.
- 4. 파이프라인 세부 정보 페이지에서 [Edit]를 선택합니다.
- 5. 편집 페이지에서 편집하려는 작업에 대해 스테이지 편집을 선택합니다.
- 입력 조건 추가를 선택합니다. 스테이지 진입 전 조건 카드가 이 조건에 사용할 수 있는 실패 옵션 과 함께 표시됩니다.
- 7. 규칙 추가를 선택하고 다음을 완료합니다.
 - a. 규칙 이름에 규칙의 이름을 입력합니다. 이 예에서는 MyAlarmRule를 입력합니다.
 - b. 규칙 공급자에서 조건에 추가할 사전 구성된 규칙 공급자를 선택합니다. 이 예에서는 AWS CloudWatchAlarm을 선택하고, 다음 단계를 완료합니다.
 - c. 리전에서 조건에 맞는 리전을 선택하거나 기본값을 그대로 둡니다.
 - d. 경보 이름에서 규칙에 사용할 CloudWatch 리소스를 선택합니다. 계정에 리소스를 이미 생성 했어야 합니다.
 - e. (선택 사항) 대기 시간에 경보가 처음 평가될 때 경보가 ALARM 상태인 경우 CodePipeline이 대기하는 시간을 입력합니다. 규칙을 처음 확인할 때 경보가 OK 상태인 경우 규칙이 즉시 성 공합니다.
 - f. (선택 사항) 모니터링할 특정 경보 상태를 입력하고 필요한 경우 역할 ARN을 입력합니다.
 - g. 스테이지 편집을 완료하면 완료를 선택합니다. 파이프라인 편집 페이지에서 저장을 선택합니다.
- 8. 실행 후 결과를 확인합니다.

건너뛰기 결과 및 VariableCheck 규칙을 사용하여 입력 조건 생성(콘솔)

입력 조건이 충족되지 않으면 스테이지를 건너뛰도록 스테이지에 대한 입력 조건을 구성할 수 있습니 다. 조건이 실패하면 결과가 작동하고 스테이지를 건너뜁니다. 스테이지를 건너뛰면 스테이지 상태는 건너뛰기 이고, 작업 상태는 실행 안 함입니다. 건너뛰기 결과가 있는 스테이지 조건에 대한 고려 사항 은 스테이지 조건에 대해 구성된 결과 고려 사항 섹션을 참조하세요.

다음 예제에서 변수 확인 규칙은 값이 일치하지 않으며 빌드 스테이지가 건너뛰어 있음을 확인합니다.

건너뛰기 결과를 사용하여 입력 조건 구성(콘솔)

- 1. AWS CloudWatchAlarm과 같이 리소스가 제공되는 규칙에 대한 리소스 및 ARN 생성과 같은 사전 조건을 완료합니다.
- 2. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home CodePipeline 콘솔을 엽니다.

사용자의 AWS 계정 에 연결된 모든 파이프라인의 이름과 상태가 표시됩니다.

- 3. [Name]에서 편집할 파이프라인의 이름을 선택합니다.
- 4. 파이프라인 세부 정보 페이지에서 [Edit]를 선택합니다.
- 5. 편집 페이지에서 편집하려는 작업에 대해 스테이지 편집을 선택합니다.
- 6. 입력 조건 추가를 선택한 다음 결과로 건너뛰기를 선택합니다.
- 7. 규칙 추가를 선택하고 다음을 완료합니다.
 - a. 규칙 이름에 규칙의 이름을 입력합니다. 이 예에서는 MyAlarmRule를 입력합니다.
 - b. 규칙 공급자에서 조건에 추가할 사전 구성된 규칙 공급자를 선택합니다. 이 예제에서는 VariableCheck를 선택하고, 다음 단계를 완료합니다.

Add entry condition ▼	Add s	success condition v	A	dd failure condition
Before stage entry o	onditio	'n		Delete condition
MyEntrySkipRule AWS VariableCheck	()	+ Add rule		

- c. 리전에서 조건에 맞는 리전을 선택하거나 기본값을 그대로 둡니다.
- d. 변수에서 GitHub(GitHub 앱 사용) 소스 작업이 있는
 #{SourceVariables.FullRepositoryName} 파이프라인과 같이 비교할 변수를 선택합
 니다. 리포지토리 이름을 입력하고 Equals와 같은 연산자를 선택합니다.
- e. 스테이지 편집을 완료하면 완료를 선택합니다. 파이프라인 편집 페이지에서 저장을 선택합니다.
- 8. 실행 후 결과를 확인합니다.

reason: Job Tailed. Job I Intry conditions with Skip	99df30f7-634c-44ba-	led. Skipping stage release.		
Deploy Skipped Pipeline execution ID: 1	a4b8096-35a1-			
Deploy	0			
Amazon S3	0			

 세부 정보를 검토하려면 검토를 선택합니다. 다음 예제의 세부 정보는 조건에 대해 구성된 결과가 건너뛰기이며 재정의할 수 없음을 보여줍니다. 조건이 충족되지 않아 규칙 상태가 실패입니다.

dition type: Be	foreEntry Result: SKIP Status: 🛞 Fa	iled	
Rule states	Rule Configuration		
Name	Rule Execution ID	Status	Reason
myruleforskip	2d28d804-20d3-4357-8ebe- 2	Failed	Job failed. Job ID d51899c9-44f4-499c- a12

입력 조건 생성(CLI)

AWS CLI 를 사용하여 입력 조건을 구성하려면 <u>파이프라인 스테이지 및 작업 생성</u> 및에 설명된 대로 명 령을 사용하여 파이프라인을 생성하거나 업데이트합니다CodePipeline에서 파이프라인 편집.

조건 및 규칙 또는 규칙 구성(CLI)

• 터미널(Linux, macOS 또는 Unix) 또는 명령 프롬프트(Windows)를 열고 AWS CLI 를 사용하여 update-pipeline 명령을 실행함으로써 파이프라인 구조의 실패 조건을 지정합니다. 다음 예제 에서는 Deploy라는 이름의 스테이지에 대한 입력 조건을 구성합니다.

```
{
    "name": "Deploy",
    "actions": [
        {
            "name": "Deploy",
            "actionTypeId": {
                "category": "Deploy",
                "owner": "AWS",
                "provider": "S3",
                "version": "1"
            },
            "runOrder": 1,
            "configuration": {
                "BucketName": "MyBucket",
                "Extract": "false",
                "ObjectKey": "object.xml"
            },
```

```
"outputArtifacts": [],
            "inputArtifacts": [
                {
                     "name": "SourceArtifact"
                }
            ],
            "region": "us-east-1",
            "namespace": "DeployVariables"
        }
    ],
    "beforeEntry": {
        "conditions": [
            {
                "result": "FAIL",
                "rules": [
                    {
                         "name": "MyAlarmRule",
                         "ruleTypeId": {
                             "category": "Rule",
                             "owner": "AWS",
                             "provider": "CloudWatchAlarm",
                             "version": "1"
                         },
                         "configuration": {
                             "AlarmName": "CWAlarm",
                             "WaitTime": "1"
                         },
                         "inputArtifacts": [],
                         "region": "us-east-1"
                    }
                ]
           }
        ]
   }
}
```

스테이지 롤백의 성공 조건 구성에 대한 자세한 내용은 CodePipeline API 참조의 SuccessConditions을 참조하세요.

입력 조건 생성(CFN)

AWS CloudFormation 를 사용하여 입력 조건을 구성하려면 beforeEntry 파라미터를 사용합니다. 입력 시 스테이지는 규칙을 실행하고 결과를 수행합니다.

```
beforeEntry:
Result: FAIL
```

• 다음 조각과 같이 템플릿을 업데이트합니다. 다음 예제에서는 MyMonitorRule이라는 규칙으로 입력 조건을 구성합니다.

```
Name: Deploy
Actions:
- Name: Deploy
 ActionTypeId:
    Category: Deploy
    Owner: AWS
    Provider: S3
    Version: '1'
  RunOrder: 1
  Configuration:
    BucketName: MyBucket
    Extract: 'false'
    ObjectKey: object.xml
  OutputArtifacts: []
  InputArtifacts:
  - Name: SourceArtifact
  Region: us-east-1
  Namespace: DeployVariables
BeforeEntry:
  Conditions:
  - Result: FAIL
    Rules:
    - Name: MyMonitorRule
      RuleTypeId:
        Category: Rule
        Owner: AWS
        Provider: CloudWatchAlarm
        Version: '1'
      Configuration:
        AlarmName: CWAlarm
        WaitTime: '1'
```

```
InputArtifacts: []
Region: us-east-1
```

beforeEntry 조건 구성에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 StageDeclaration에서 AWS::CodePipeline::Pipeline BeforeEntryConditions을 참조하세요.

실패 시 조건 생성

콘솔 또는 CLI를 사용하여 스테이지의 실패 시 조건을 구성할 수 있습니다. 각 조건에 해당하는 규칙과 결과를 구성합니다. 롤백 결과의 경우 파이프라인은 현재 파이프라인 구조 버전에서 이전 실행이 시작 된 경우에만 이전 실행으로 롤백할 수 있습니다.

실패 시 조건 생성(콘솔)

조건이 충족될 때 스테이지가 수행할 규칙 및 결과와 함께 스테이지에 대한 실패 시 조건을 구성할 수 있습니다.

실패 시 조건 구성(콘솔)

- 1. LambdaInvoke 규칙과 같이 리소스가 제공되는 규칙에 대한 리소스 및 ARN 생성과 같은 사전 조 건을 완료합니다.
- 2. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home CodePipeline 콘솔을 엽니다.

사용자의 AWS 계정 에 연결된 모든 파이프라인의 이름과 상태가 표시됩니다.

- 3. [Name]에서 편집할 파이프라인의 이름을 선택합니다.
- 4. 파이프라인 세부 정보 페이지에서 [Edit]를 선택합니다.
- 5. 편집 페이지에서 편집하려는 작업에 대해 스테이지 편집을 선택합니다.
- 실패 조건 추가를 선택합니다. 실패 조건 카드가 이 조건에 사용할 수 있는 롤백 옵션과 함께 표시 됩니다.
- 7. 규칙 추가를 선택하고 다음을 완료합니다.
 - a. 규칙 이름에 규칙의 이름을 입력합니다. 이 예에서는 MyLambdaRule를 입력합니다.
 - b. 규칙 공급자에서 조건에 추가할 사전 구성된 규칙 공급자를 선택합니다. 이 예에서는 AWS LambdaInvoke를 선택하고, 다음 단계를 완료합니다.
 - c. 리전에서 조건에 맞는 리전을 선택하거나 기본값을 그대로 둡니다.

- d. 입력 아티팩트에서 소스 아티팩트를 선택합니다.
- e. 함수 이름에서 규칙에 사용할 Lambda 리소스를 선택합니다. 계정에 리소스를 이미 생성했어 야 합니다.
- f. (선택 사항) 사용자 파라미터에 추가 구성을 위한 파라미터를 나타내는 모든 페어를 입력합니
 다.
- g. (선택 사항) 역할 ARN에서 구성된 경우 역할 ARN을 입력합니다.
- h. (선택 사항) 제한 시간(분)에 규칙이 제한 시간 전에 기다려야 하는 시간을 분 단위로 입력합니
 다.
- i. 스테이지 편집을 완료하면 완료를 선택합니다. 파이프라인 편집 페이지에서 저장을 선택합니다.
 다.

재시도 결과 예제를 사용하여 onFailure 조건 생성(콘솔)

입력 조건이 충족되지 않으면 스테이지가 재시도되도록 스테이지에 대한 onFailure 조건을 구성할 수 있습니다. 이 결과의 일부로 실패한 작업을 다시 시도할지 또는 실패한 단계를 다시 시도할지 지정하여 재시도 모드를 구성합니다.

재시도 결과로 onFailure 조건 구성(콘솔)

- 1. AWS CloudWatchAlarm과 같이 리소스가 제공되는 규칙에 대한 리소스 및 ARN 생성과 같은 사전 조건을 완료합니다.
- 2. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home CodePipeline 콘솔을 엽니다.

사용자의 AWS 계정 에 연결된 모든 파이프라인의 이름과 상태가 표시됩니다.

- 3. [Name]에서 편집할 파이프라인의 이름을 선택합니다.
- 4. 파이프라인 세부 정보 페이지에서 [Edit]를 선택합니다.
- 5. 편집 페이지에서 편집하려는 작업에 대해 스테이지 편집을 선택합니다.
- 6. 스테이지 하단의 자동화된 스테이지 구성:에서 스테이지 장애 시 자동 재시도 활성화를 선택합니
 다. 재시도 모드에서 실패한 단계 재시도 또는 실패한 작업 재시도 중 하나를 선택합니다.

s3deploy 🕕	
Amazon 53 🖸	+ Add action
\mathscr{O} \times	Enable automatic rollback on stage failu
	Enable automatic retry on stage failure
+ Add action group	None
tomated stage configuration:	None 🔺

- 7. onFailure 조건을 추가하도록 선택한 다음 규칙 추가를 선택하고 조건에 대한 규칙을 입력합니다.
 - a. 규칙 이름에 규칙의 이름을 입력합니다. 이 예에서는 MyAlarmRule를 입력합니다.
 - b. 규칙 공급자에서 조건에 추가할 사전 구성된 규칙 공급자를 선택합니다. 이 예에서는 CloudWatchAlarm을 선택하고, 다음 단계를 완료합니다.
 - c. 리전에서 조건에 맞는 리전을 선택하거나 기본값을 그대로 둡니다.
 - d. 경보 이름에서 경보에 대해 구성된 리소스를 선택합니다.
 - e. 스테이지 편집을 완료하면 완료를 선택합니다. 파이프라인 편집 페이지에서 저장을 선택합니다.
- 8. 실행 후 결과를 확인합니다.

실패 시 조건 생성(CLI)

AWS CLI 를 사용하여 실패 시 조건을 구성하려면 <u>파이프라인 스테이지 및 작업 생성</u> 및에 설명된 대로 명령을 사용하여 파이프라인을 생성하거나 업데이트합니다CodePipeline에서 파이프라인 편집.

조건 및 규칙 또는 규칙 구성(CLI)

• 터미널(Linux, macOS 또는 Unix) 또는 명령 프롬프트(Windows)를 열고 AWS CLI 를 사용하여 update-pipeline 명령을 실행함으로써 파이프라인 구조의 실패 조건을 지정합니다. 다음 예제 에서는 Deploy라는 이름의 스테이지에 대한 실패 시 조건을 구성합니다.

```
"owner": "AWS",
            "provider": "S3",
            "version": "1"
        },
        "runOrder": 1,
        "configuration": {
            "BucketName": "MyBucket",
            "Extract": "false",
            "ObjectKey": "object.xml"
        },
        "outputArtifacts": [],
        "inputArtifacts": [
            {
                "name": "SourceArtifact"
            }
        ],
        "region": "us-east-1",
        "namespace": "DeployVariables"
    }
],
"onFailure": {
    "conditions": [
        {
            "result": "ROLLBACK",
            "rules": [
                {
                     "name": "MyLambdaRule",
                     "ruleTypeId": {
                         "category": "Rule",
                         "owner": "AWS",
                         "provider": "LambdaInvoke",
                         "version": "1"
                    },
                     "configuration": {
                         "FunctionName": "my-function"
                    },
                     "inputArtifacts": [
                         {
                             "name": "SourceArtifact"
                         }
                    ],
                     "region": "us-east-1"
                }
            ]
```

}] } }

실패 조건 구성에 대한 자세한 내용은 CodePipeline API 참조의 FailureConditions을 참조하세요.

실패 시 조건 생성(CFN)

AWS CloudFormation 를 사용하여 실패 시 조건을 구성하려면 OnFailure 파라미터를 사용합니다. 성공 시 스테이지는 규칙을 실행하고 결과를 수행합니다.

OnFailure: Result: ROLLBACK

• 다음 조각과 같이 템플릿을 업데이트합니다. 다음 예제에서는 MyMonitorRule이라는 규칙으로 OnFailure 조건을 구성합니다.

```
name: Deploy
actions:
- name: Deploy
  actionTypeId:
    category: Deploy
    owner: AWS
    provider: S3
    version: '1'
  runOrder: 1
  configuration:
    BucketName: MyBucket
    Extract: 'false'
    ObjectKey: object.xml
  outputArtifacts: []
  inputArtifacts:
  - name: SourceArtifact
  region: us-east-1
  namespace: DeployVariables
OnFailure:
  conditions:
  - result: ROLLBACK
    rules:
    - name: MyMonitorRule
```

```
ruleTypeId:
    category: Rule
    owner: AWS
    provider: CloudWatchAlarm
    version: '1'
configuration:
    AlarmName: AlarmOnHelloWorldInvocation
    AlarmStates: ALARM
    WaitTime: '1'
inputArtifacts: []
region: us-east-1
```

실패 조건 구성에 대한 자세한 내용은 AWS CloudFormation 사용 설명서에서 StageDeclaration 아래의 <u>OnFailure</u>를 참조하세요.

성공 시 조건 생성

콘솔 또는 CLI를 사용하여 스테이지의 성공 시 조건을 구성할 수 있습니다. 각 조건에 해당하는 규칙과 결과를 구성합니다. 롤백 결과의 경우 파이프라인은 현재 파이프라인 구조 버전에서 이전 실행이 시작 된 경우에만 이전 실행으로 롤백할 수 있습니다.

이 단계에서는 배포 창 규칙을 사용하는 성공 시 조건의 예를 제공합니다.

자세한 내용은 CodePipeline API 가이드의 조건, RuleTypeId 및 RuleExecution을 참조하세요.

성공 시 조건 생성(콘솔)

조건이 충족될 때 스테이지가 수행할 규칙 및 결과와 함께 스테이지에 대한 성공 시 조건을 구성할 수 있습니다.

성공 시 조건 구성(콘솔)

- a AWS LambdaRule과 같이 리소스가 제공되는 규칙에 대한 리소스 및 ARN 생성과 같은 사전 조 건을 완료합니다.
- 2. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> <u>codepipeline/home</u> CodePipeline 콘솔을 엽니다.

사용자의 AWS 계정 에 연결된 모든 파이프라인의 이름과 상태가 표시됩니다.

3. [Name]에서 편집할 파이프라인의 이름을 선택합니다.

- 4. 파이프라인 세부 정보 페이지에서 [Edit]를 선택합니다.
- 5. 편집 페이지에서 편집하려는 작업에 대해 스테이지 편집을 선택합니다.
- 성공 조건 추가를 선택합니다. 온스테이지 성공 조건 카드가 표시됩니다. 이 조건 유형에 사용 가 능한 결과로 표시된 롤백 또는 실패 옵션을 선택합니다.
- 7. 규칙 추가를 선택하고 다음을 완료합니다.
 - a. 규칙 이름에 조건의 이름을 입력합니다. 이 예에서는 MyDeploymentRule를 입력합니다.
 - b. 규칙 공급자에서 조건에 추가할 사전 구성된 규칙을 선택합니다. 이 예에서는 AWS DeploymentWindow를 선택하고, 다음 단계를 완료합니다.
 - c. 리전에서 조건에 맞는 리전을 선택하거나 기본값을 그대로 둡니다.
 - d. Cron에 배포 창의 cron 표현식을 입력합니다. cron 표현식은 배포가 허용되어야 하는 날짜와 시간을 정의합니다. cron 표현식에 대한 참조 정보는 <u>cron 및 rate 표현식을 사용하여 규칙 예</u> 약을 참조하세요.
 - e. (선택 사항) TimeZone에 배포 창의 시간대를 입력합니다.
- 8. 실행 후 결과를 확인합니다.

134det6d Source: Added simpleCov-report.json	
Disable transition	
Deploy (i) Succeeded	Start rollback
ipeline execution ID: <u>8a6fe20b-9670-4a41-a5a8-be04d6750</u>	1 <u>1c</u>
Deploy	
Deploy Amazon S3 🖸	
Deploy <u>Amazon S3</u>	
Deploy <u>Amazon S3</u> Succeeded - <u>Just now</u> <u>View details</u>	
Deploy <u>Amazon S3</u> Succeeded - <u>Just now</u> <u>View details</u> <u>S4def6d</u> Source: Added simpleCov-report.json	

성공 시 조건 생성(CLI)

AWS CLI 를 사용하여 성공 시 조건을 구성하려면 <u>파이프라인 스테이지 및 작업 생성</u> 및에 설명된 대로 명령을 사용하여 파이프라인을 생성하거나 업데이트합니다CodePipeline에서 파이프라인 편집.

조건 및 규칙 또는 규칙 구성(CLI)

• 터미널(Linux, macOS 또는 Unix) 또는 명령 프롬프트(Windows)를 열고 AWS CLI 를 사용하여 update-pipeline 명령을 실행함으로써 파이프라인 구조의 실패 조건을 지정합니다. 다음 예제 에서는 규칙 이름이 MyDeploymentRule인 Deploy라는 이름의 스테이지에 대한 성공 시 조건을 구성합니다.

```
{
    "name": "Deploy",
    "actions": [
        {
            "name": "Deploy",
            "actionTypeId": {
                "category": "Deploy",
                "owner": "AWS",
                "provider": "S3",
                "version": "1"
            },
            "runOrder": 1,
            "configuration": {
                "BucketName": "MyBucket",
                "Extract": "false",
                "ObjectKey": "object.xml"
            },
            "outputArtifacts": [],
            "inputArtifacts": [
                {
                    "name": "SourceArtifact"
                }
            ],
            "region": "us-east-1",
            "namespace": "DeployVariables"
        }
    ],
    "onSuccess": {
       "conditions": [
            ſ
                "result": "FAIL",
                "rules": [
                    {
                         "name": "MyAlarmRule",
                         "ruleTypeId": {
                             "category": "Rule",
                             "owner": "AWS",
                             "provider": "CloudWatchAlarm",
                             "version": "1"
                        },
                         "configuration": {
                             "AlarmName": "CWAlarm",
                             "WaitTime": "1"
```



성공 조건 구성에 대한 자세한 내용은 CodePipeline API 참조의 <u>SuccessConditions</u>을 참조하세 요.

성공 시 조건 생성(CFN)

AWS CloudFormation 를 사용하여 성공 시 조건을 구성하려면 OnSuccess 파라미터를 사용합니다. 성공 시 스테이지는 규칙을 실행하고 결과를 수행합니다.

```
OnSuccess:
Result: ROLLBACK
```

• 다음 조각과 같이 템플릿을 업데이트합니다. 다음 예제에서는 MyDeploymentWindowRule이라 는 규칙으로 OnSuccess 조건을 구성합니다.

```
name: Deploy
actions:
- name: Deploy
 actionTypeId:
    category: Deploy
    owner: AWS
    provider: S3
    version: '1'
 runOrder: 1
 configuration:
    BucketName: MyBucket
    Extract: 'false'
    ObjectKey: object.xml
 outputArtifacts: []
 inputArtifacts:
  - name: SourceArtifact
 region: us-east-1
```

```
namespace: DeployVariables
onSuccess:
  conditions:
  - result: FAIL
    rules:
    - name: MyMonitorRule
      ruleTypeId:
        category: Rule
        owner: AWS
        provider: CloudWatchAlarm
        version: '1'
      configuration:
        AlarmName: CWAlarm
        WaitTime: '1'
      inputArtifacts: []
      region: us-east-1
```

스테이지 롤백의 실패 조건 구성에 대한 자세한 내용은 AWS CloudFormation 사용 설명서에서 StageDeclaration 아래의 OnFailure를 참조하세요.

스테이지 조건 삭제

파이프라인에 대해 구성된 스테이지 조건을 삭제할 수 있습니다.

스테이지 조건을 삭제하려면

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home CodePipeline 콘솔을 엽니다.

와 연결된 모든 파이프라인의 이름과 상태가 AWS 계정 표시됩니다.

- 2. [Name]에서 편집할 파이프라인의 이름을 선택합니다.
- 3. 파이프라인 세부 정보 페이지에서 [Edit]를 선택합니다.
- 4. 편집 페이지에서 편집하려는 조건에 대해 스테이지 편집을 선택합니다.
- 5. 삭제하려는 조건 옆에 있는 조건 삭제를 선택합니다.

스테이지 조건 재정의

파이프라인에 대해 구성된 스테이지 조건을 재정의할 수 있습니다. 콘솔에서 스테이지와 규칙이 실행 중일 때 스테이지 조건을 재정의하도록 선택할 수 있습니다. 그러면 스테이지가 실행됩니다.

스테이지 조건을 재정의하려면

1. 이 예제에서는 파이프라인 단계가 조건과 함께 실행됩니다. 재정의 버튼이 활성화됩니다.

Disable transition	
Deploy In progress Pipeline execution ID: <u>8a6fe20b-9670-4a41-a5a8-be04d6750d1c</u>	 Image: Second sec
Deploy <u>Amazon S3</u> Succeeded - <u>3 minutes ago</u> View details d34def6d Source: Added simpleCov-report.json	
OnSuccess condition: In progress Execution ID: 8at Override condition Review Reason: Waiting for the time window to open. This is estimated to happen in >1 week. Provide the time window to open. This is estimated to happen in >1 week.	

2. 재정의하려는 조건 옆에 있는 재정의를 선택합니다.

Disable transition		
Deploy Succeeded	Start rollback	
Pipeline execution ID: <u>8a6fe20b-9670-4a41-a5a8-be04d6750d1c</u>		
Deploy		
Amazon S3 G Succeeded - <u>5 minutes ago</u>		
View details		
d34def6d Source: Added simpleCov-report.json		
OnSuccess condition: V Overridden Execution ID: 8a6fe20b	Review	Ĩ

 세부 정보를 검토하려면 검토를 선택합니다. 다음 예제의 세부 정보는 조건에 대해 구성된 결과가 실패이며 재정의되었음을 보여줍니다. 규칙 상태는 재정의로 인해 중단됨 상태입니다.

Condition execution details Execution ID: 8a6fe20b-9670-4a41-a5a8-be04d6750d1c				
Condition type: OnSuccess Rule states:	Result: FAIL Status: V Overridden			
Name	Rule Execution ID	Status	Reason	
MyDeploymentRule	2e5989ea-b59d-4ae8-93fb-5a47538956bb	Abandoned	-	

Done

작업 유형, 사용자 지정 작업 및 승인 작업 사용

에서 AWS CodePipeline작업은 파이프라인의 단계에 있는 시퀀스의 일부입니다. 이는 해당 단계의 아 티팩트에서 수행된 작업입니다. 파이프라인 작업은 단계 구성에서 정의한 대로 지정된 순서나 연속 또 는 병렬로 발생합니다.

CodePipeline은 6가지 유형의 작업에 대한 지원을 제공합니다.

- 소스
- 빌드
- 테스트
- 배포
- 승인
- 간접 호출

작업 유형에 따라 파이프라인에 통합할 수 있는 AWS 서비스 및 파트너 제품 및 서비스에 대한 자세한 내용은 섹션을 참조하세요CodePipeline 작업 유형과의 통합.

주제

- 작업 유형 처리
- CodePipeline에서 사용자 지정 작업 생성 및 추가
- CodePipeline에서 사용자 지정 작업 태그 지정
- CodePipeline의 파이프라인에서 AWS Lambda 함수 호출
- 스테이지에 수동 승인 작업 추가
- CodePipeline에 교차 리전 작업 추가
- <u>변수 작업</u>

작업 유형 처리

작업 유형은 공급자가 AWS CodePipeline에서 지원되는 통합 모델 중 하나를 사용하여 고객을 위해 생성하는 미리 구성된 작업입니다.

작업 유형을 요청, 확인 및 업데이트할 수 있습니다. 계정의 작업 유형이 소유자로 생성된 경우 AWS CLI 를 사용하여 작업 유형 속성 및 구조를 보거나 업데이트할 수 있습니다. 작업 유형의 공급자 또는 소유자인 경우, CodePipeline에서 사용할 수 있게 되면 고객이 해당 작업을 선택하고 파이프라인에 추 가할 수 있습니다.

Note

작업자와 함께 실행할 custom 작업을 owner 필드에 생성합니다. 통합 모델로 만들 수는 없습니다. 사용자 지정 작업에 대한 자세한 내용은 <u>CodePipeline에서 사용자 지정 작업 생성 및 추</u>가 섹션을 참조하세요.

작업 유형 구성 요소

작업 유형을 구성하는 구성 요소는 다음과 같습니다.

• 작업 유형 ID - ID는 범주, 소유자, 공급자 및 버전으로 구성됩니다. 다음 예제는 ThirdParty 소유 자, Test 범주, TestProvider라는 이름의 공급자, 1 버전인 작업 유형 ID를 보여줍니다.

```
{
    "Category": "Test",
    "Owner": "ThirdParty",
    "Provider": "TestProvider",
    "Version": "1"
},
```

- 실행기 구성 작업이 생성될 때 지정된 통합 모델 또는 작업 엔진입니다. 작업 유형의 실행기를 지정 할 때는 다음 두 가지 유형 중 하나를 선택합니다.
 - Lambda: 작업 유형 소유자는 Lambda 함수로 통합을 작성합니다. 이 함수는 작업에 사용할 수 있 는 작업이 있을 때마다 CodePipeline에 의해 호출됩니다.
 - JobWorker: 작업 유형 소유자는 고객 파이프라인에서 사용 가능한 작업을 폴링하는 작업자로 통 합을 작성합니다. 그러면 작업자는 CodePipeline API를 사용하여 작업을 실행하고 CodePipeline 에 작업 결과를 다시 제출합니다.

Note
 작업자 통합 모델은 선호되는 통합 모델이 아닙니다.

입력 및 출력 아티팩트: 작업 유형 소유자가 작업의 고객에 대해 지정하는 아티팩트에 대한 제한입니다.

작업 유형 처리

- 권한: 타사 작업 유형에 액세스할 수 있는 고객을 지정하는 권한 전략입니다. 사용 가능한 권한 전략
 은 작업 유형에 대해 선택한 통합 모델에 따라 다릅니다.
- URL: 고객이 상호 작용할 수 있는 리소스로 연결되는 딥 링크입니다(예: 작업 유형 소유자의 구성 페이지).

주제

- 작업 유형 요청
- 파이프라인에 사용 가능한 작업 유형 추가(콘솔)
- 작업 유형 보기
- 작업 유형 업데이트

작업 유형 요청

타사 공급자가 새 CodePipeline 작업 유형을 요청하면 CodePipeline에서 작업 유형 소유자를 위한 작 업 유형이 생성되고 소유자는 작업 유형을 관리하고 볼 수 있습니다.

작업 유형은 프라이빗 또는 퍼블릭 작업일 수 있습니다. 작업 유형을 생성하면 프라이빗입니다. 작업 유형을 퍼블릭 작업으로 변경하도록 요청하려면 CodePipeline 서비스 팀에 문의하세요.

CodePipeline 팀을 위한 작업 정의 파일, 실행자 리소스 및 작업 유형 요청을 생성하기 전에 통합 모델 을 선택해야 합니다.

1단계: 통합 모델 선택

통합 모델을 선택한 다음 해당 모델의 구성을 생성합니다. 통합 모델을 선택한 후에는 통합 리소스를 구성해야 합니다.

- Lambda 통합 모델의 경우, Lambda 함수를 생성하고 권한을 추가합니다. CodePipeline 서비스 보안 주체: codepipeline.amazonaws.com을 사용하여 CodePipeline 서비스를 호출할 수 있는 권한 을 제공하는 권한을 통합자 Lambda 함수에 추가합니다. AWS CloudFormation 또는 명령줄을 사용 하여 권한을 추가할 수 있습니다.
 - AWS CloudFormation을 사용하여 권한을 추가하는 예:

```
CodePipelineLambdaBasedActionPermission:
Type: 'AWS::Lambda::Permission'
Properties:
Action: 'lambda:invokeFunction'
```

FunctionName: {"Fn::Sub": "arn:\${AWS::Partition}:lambda:\${AWS::Region}:
\${AWS::AccountId}:function:function-name"}
Principal: codepipeline.amazonaws.com

- 명령줄 설명서
- 작업자 통합 모델의 경우 작업자가 CodePipeline API로 작업을 폴링하는 허용된 계정 목록을 사용하 여 통합을 생성합니다.

2단계: 작업 유형 정의 파일 생성

JSON을 사용하여 작업 유형 정의 파일에서 작업 유형을 정의합니다. 파일에는 작업 범주, 작업 유형을 관리하는 데 사용되는 통합 모델 및 구성 속성이 포함됩니다.

Note

퍼블릭 작업을 만든 후에는 properties의 작업 유형 속성을 optional에서 required로 변 경할 수 없습니다. 또한 owner도 변경할 수 없습니다.

작업 유형 정의 파일 파라미터에 대한 자세한 내용은 <u>CodePipeline API 참조</u>의 <u>ActionTypeDeclaration</u> 및 UpdateActionType을 참조하세요.

작업 유형 정의 파일에는 다음과 같은 8개의 섹션이 있습니다.

- description: 업데이트할 작업 유형에 대한 설명입니다.
- executor: 지원되는 통합 모델(Lambda 또는 job worker)을 사용하여 만든 작업 유형의 실 행기에 대한 정보입니다. 실행기 유형에 따라 jobWorkerExecutorConfiguration 또는 lambdaExecutorConfiguration 중 하나만 제공할 수 있습니다.
 - configuration: 선택한 통합 모델을 기반으로 하는 작업 유형 구성을 위한 리소스입니다.
 Lambda 통합 모델의 경우 Lambda 함수 ARN을 사용하세요. 작업자 통합 모델의 경우 작업자가 실행하는 계정 또는 계정 목록을 사용하세요.
 - jobTimeout: 작업에 대한 제한 시간(초)입니다. 작업 실행은 여러 작업으로 구성될 수 있습니다. 이는 전체 작업 실행에 대한 제한 시간이 아니라 단일 작업에 대한 제한 시간입니다.

Note

Lambda 통합 모델의 경우 최대 제한 시간은 15분입니다.

- policyStatementsTemplate: CodePipeline 고객 계정에서 작업 실행을 성공적으로 실행하는 데 필요한 권한을 지정하는 정책 설명입니다.
- type: 작업 유형을 생성하고 업데이트하는 데 사용되는 통합 모델(Lambda 또는 JobWorker)입니다.
- id: 작업 유형의 범주, 소유자, 공급자 및 버전 ID입니다.
 - category: 단계에서 수행할 수 있는 작업의 종류는 소스, 빌드, 배포, 테스트, 호출 또는 승인입니다.
 - provider: 호출되는 작업 유형의 공급자(예: 공급자 회사 또는 제품 이름)입니다. 공급자 이름은 작업 유형이 생성될 때 제공됩니다.
 - owner: 호출되는 작업 유형의 생성자는 AWS 또는 ThirdParty입니다.
 - version: 작업 유형의 버전을 지정하는 데 사용되는 문자열입니다. 첫 번째 버전의 경우 버전 번 호를 1로 설정합니다.
- inputArtifactDetails: 파이프라인의 이전 단계에서 예상할 수 있는 아티팩트 수입니다.
- outputArtifactDetails:작업 유형 단계의 결과에서 예상할 수 있는 아티팩트 수입니다.
- permissions: 작업 유형을 사용할 권한이 있는 계정을 식별하는 세부 정보입니다.
- properties: 프로젝트 작업을 완료하는 데 필요한 파라미터입니다.
 - description: 사용자에게 표시되는 속성에 대한 설명입니다.
 - optional: 구성 속성이 선택 사항인지 여부를 나타냅니다.
 - noEcho: 고객이 입력한 필드 값이 로그에서 생략되었는지 여부를 나타냅니다. true인 경우 GetPipeline API 요청과 함께 반환되면 값이 삭제됩니다.
 - key: 구성 속성이 키인지 여부를 나타냅니다.
 - queryable: 속성이 폴링에 사용되는지 여부를 나타냅니다. 작업 유형에는 쿼리가 가능한 속성이 최대 1개 있을 수 있습니다. 이러한 속성이 하나 있을 경우 해당 속성은 필수여야 하고 보안 항목이 아니어야 합니다.
 - name: 사용자에게 표시되는 속성 이름입니다.
- urls: URL CodePipeline 목록이 사용자에게 표시됩니다.
 - entityUrlTemplate: 작업 유형의 외부 리소스에 대한 URL(예: 구성 페이지)입니다.
 - executionUrlTemplate: 작업의 최신 실행에 대한 세부 정보 URL입니다.
 - revisionUrlTemplate: CodePipeline 콘솔에 표시되는 URL로, 고객이 외부 작업의 구성을 업 데이트 또는 변경할 수 있는 페이지로 연결됩니다.
 - thirdPartyConfigurationUrl: 페이지의 URL로, 여기서 사용자는 외부 서비스에 가입하고 서비스에서 제공하는 작업의 초기 구성을 수행할 수 있습니다.

다음 코드는 예제 작업 유형 정의 파일을 보여줍니다.

```
{
   "actionType": {
      "description": "string",
      "executor": {
         "configuration": {
            "jobWorkerExecutorConfiguration": {
               "pollingAccounts": [ "string" ],
               "pollingServicePrincipals": [ "string" ]
            },
            "lambdaExecutorConfiguration": {
               "lambdaFunctionArn": "string"
            }
         },
         "jobTimeout": number,
         "policyStatementsTemplate": "string",
         "type": "string"
      },
      "id": {
         "category": "string",
         "owner": "string",
         "provider": "string",
         "version": "string"
      },
      "inputArtifactDetails": {
         "maximumCount": number,
         "minimumCount": number
      },
      "outputArtifactDetails": {
         "maximumCount": number,
         "minimumCount": number
      },
      "permissions": {
         "allowedAccounts": [ "string" ]
      },
      "properties": [
         {
            "description": "string",
            "key": boolean,
            "name": "string",
            "noEcho": boolean,
            "optional": boolean,
            "queryable": boolean
```

```
}
],
"urls": {
    "configurationUrl": "string",
    "entityUrlTemplate": "string",
    "executionUrlTemplate": "string",
    "revisionUrlTemplate": "string"
}
}
```

3단계: CodePipeline과의 통합 등록

CodePipeline에 작업 유형을 등록하려면 CodePipeline 서비스 팀에 문의하여 요청을 보내야 합니다.

CodePipeline 서비스 팀은 서비스 코드베이스를 변경하여 새 작업 유형 통합을 등록합니다. CodePipeline은 퍼블릭 작업과 프라이빗 작업이라는 두 개의 새로운 액션을 등록합니다. 프라이빗 작 업을 테스트에 사용한 다음 준비가 되면 퍼블릭 작업을 활성화하여 고객 트래픽을 처리합니다.

Lambda 통합 요청을 등록하려면

• 다음 양식을 사용하여 CodePipeline 서비스 팀에 요청을 보내세요.

This issue will track the onboarding of [Name] in CodePipeline.
[Contact engineer] will be the primary point of contact for this integration.
Name of the action type as you want it to appear to customers: Example.com Testing
Initial onboard checklist:
1. Attach an action type definition file in JSON format. This includes the schema
for the action type
2. A list of test accounts for the allowlist which can access the new action type
[{account, account_name}]
3. The Lambda function ARN
4. List of AWS ## where your action will be available

5. Will this be available as a public action?

작업자 통합 요청을 등록하려면

다음 양식을 사용하여 CodePipeline 서비스 팀에 요청을 보내세요.

```
This issue will track the onboarding of [Name] in CodePipeline.
[Contact engineer] will be the primary point of contact for this integration.
Name of the action type as you want it to appear to customers: Example.com Testing
Initial onboard checklist:
1. Attach an action type definition file in JSON format. This includes the schema
for the action type.
2. A list of test accounts for the allowlist which can access the new action type
[{account, account_name}]
3. URL information:
Website URL: https://www.example.com/%TestThirdPartyName%/%TestVersionNumber%
Example URL pattern where customers will be able to review their configuration
information for the action: https://www.example.com/%TestThirdPartyName%/
%customer-ID%/%CustomerActionConfiguration%
Example runtime URL pattern: https://www.example.com/%TestThirdPartyName%/
%customer-ID%/%TestRunId%
4. List of AWS ## where your action will be available
5. Will this be available as a public action?
```

4단계: 새 통합 활성화

새 통합을 공개적으로 사용할 준비가 되면 CodePipeline 서비스 팀에 문의하세요.

파이프라인에 사용 가능한 작업 유형 추가(콘솔)

테스트할 수 있도록 파이프라인에 작업 유형을 추가합니다. 새 파이프라인을 만들거나 기존 파이프라 인을 편집하여 이 작업을 수행할 수 있습니다.

Note

작업 유형이 소스, 빌드 또는 배포 카테고리 작업인 경우 파이프라인을 생성하여 추가할 수 있 습니다. 작업 유형이 테스트 범주에 있는 경우 기존 파이프라인을 편집하여 추가해야 합니다.

CodePipeline 콘솔에서 기존 파이프라인에 작업 유형을 추가하려면

- 1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https://https://https://https://i/httpsCodePipeline.com.com.com.com.
- 2. 파이프라인 목록에서 작업 유형을 추가할 파이프라인을 선택합니다.
- 3. 파이프라인의 요약 보기 페이지에서 편집을 선택합니다.
- 단계를 편집하려면 선택합니다. 작업 유형을 추가하려는 단계에서 작업 그룹 추가를 선택합니다.
 작업 편집 페이지가 표시됩니다.
- 5. 작업 편집 페이지의 작업 이름에 작업 이름을 입력합니다. 파이프라인의 단계에 표시되는 이름입 니다.
- 6. 작업 공급자의 목록에서 작업 유형을 선택합니다.

목록의 값은 작업 유형 정의 파일에 지정된 provider를 기반으로 한다는 점에 유의하세요.

7. 입력 아티팩트에 이 형식으로 아티팩트 이름을 입력합니다.

Artifactname::FileName

단, 허용되는 최소 및 최대 수량은 작업 유형 정의 파일에 지정된 inputArtifactDetails을 기 준으로 정의됩니다.

8. <Action_Name>에 연결을 선택합니다.

브라우저 창이 열리고 작업 유형에 맞게 생성한 웹 사이트로 연결됩니다.

 웹 사이트에 고객으로 로그인하고 고객이 작업 유형을 사용하기 위해 수행하는 단계를 완료하세
 요. 단계는 작업 범주, 웹 사이트 및 구성에 따라 다르지만 일반적으로 고객을 작업 편집 페이지로 돌아가게 하는 완료 작업이 포함됩니다. CodePipeline 작업 편집 페이지에는 작업에 대한 추가 구성 필드가 표시됩니다. 표시되는 필드는 작업 정의 파일에 지정한 구성 속성입니다. 작업 유형에 맞게 사용자 지정된 필드에 정보를 입력합 니다.

예를 들어, 작업 정의 파일에서 Host라는 이름의 속성을 지정한 경우 해당 작업에 대한 작업 편집 페이지에 호스트라는 레이블이 있는 필드가 표시됩니다.

11. 출력 아티팩트에 이 형식으로 아티팩트 이름을 입력합니다.

Artifactname::FileName

단, 허용되는 최소 및 최대 수량은 작업 유형 정의 파일에 지정된 outputArtifactDetails을 기준으로 정의됩니다.

12. 완료를 선택하여 파이프라인 세부 정보 페이지로 돌아가십시오.

Note

고객은 선택적으로 CLI를 사용하여 파이프라인에 작업 유형을 추가할 수 있습니다.

13. 작업을 테스트하려면 파이프라인의 소스 단계에서 지정된 소스에 변경 사항을 커밋하거나 <u>수동으</u> 로 파이프라인 시작의 단계를 따르세요.

작업 유형으로 파이프라인을 생성하려면 <u>파이프라인 스테이지 및 작업 생성</u>의 단계에 따라 테스트할 여러 단계에서 작업 유형을 선택합니다.

작업 유형 보기

CLI를 사용하여 작업 유형을 확인할 수 있습니다. get-action-type 명령을 사용하면 통합 모델을 사용하여 생성된 작업 유형을 볼 수 있습니다.

작업 유형을 보려면

1. 입력 JSON 파일을 만들고 파일 이름을 file.json으로 지정합니다. 다음 예와 같이 JSON 형식 으로 작업 유형 ID를 추가합니다.

```
{
    "category": "Test",
    "owner": "ThirdParty",
    "provider": "TestProvider",
    "version": "1"
```

}

2. 터미널 창 또는 명령줄에서 get-action-type 명령을 실행합니다.

```
aws codepipeline get-action-type --cli-input-json file://file.json
```

이 명령은 작업 유형에 대한 작업 정의 출력을 반환합니다. 이 예제는 Lambda 통합 모델로 생성된 작업 유형을 보여줍니다.

```
{
    "actionType": {
        "executor": {
            "configuration": {
                "lambdaExecutorConfiguration": {
                     "lambdaFunctionArn": "arn:aws:lambda:us-west-2:<account-
id>:function:my-function"
                }
            },
            "type": "Lambda"
        },
        "id": {
            "category": "Test",
            "owner": "ThirdParty",
            "provider": "TestProvider",
            "version": "1"
        },
        "inputArtifactDetails": {
            "minimumCount": 0,
            "maximumCount": 1
        },
        "outputArtifactDetails": {
            "minimumCount": 0,
            "maximumCount": 1
        },
        "permissions": {
            "allowedAccounts": [
                "<account-id>"
            ٦
        },
        "properties": []
    }
}
```

작업 유형 업데이트

CLI를 사용하여 통합 모델로 만든 작업 유형을 편집할 수 있습니다.

퍼블릭 작업 유형의 경우 소유자를 업데이트할 수 없고, 선택적 속성을 필수로 변경할 수 없으며, 새 선 택적 속성만 추가할 수 있습니다.

- get-action-type 명령을 사용하여 작업 유형의 구조를 가져올 수 있습니다. 구조를 복사합니다.
 다.
- 입력 JSON 파일을 생성하고 이름을 action.json으로 지정합니다. 이전 단계에서 복사한 작업 유형 구조를 해당 구조에 붙여넣습니다. 변경할 파라미터를 업데이트합니다. 또한 옵션 파라미터 를 추가할 수 있습니다.

입력 파일의 파라미터에 대한 자세한 내용은 <u>2단계: 작업 유형 정의 파일 생성</u>의 작업 정의 파일 설 명을 참조하세요.

다음 예제는 Lambda 통합 모델로 생성된 예제 작업 유형을 업데이트하는 방법을 보여줍니다. 이 예제에서는 다음을 변경합니다.

- provider 이름을 TestProvider1으로 변경합니다.
- 작업 제한 시간을 900초로 추가합니다.
- 작업을 사용하는 고객에게 표시되는 Host로 이름이 지정된 작업 구성 속성을 추가합니다.

```
{
    "actionType": {
        "executor": {
            "configuration": {
                "lambdaExecutorConfiguration": {
                     "lambdaFunctionArn": "arn:aws:lambda:us-west-2:<account-
id>:function:my-function"
                }
            },
            "type": "Lambda",
            "jobTimeout": 900
        },
        "id": {
            "category": "Test",
            "owner": "ThirdParty",
            "provider": "TestProvider1",
            "version": "1"
        },
```

```
"inputArtifactDetails": {
            "minimumCount": 0,
            "maximumCount": 1
        },
        "outputArtifactDetails": {
            "minimumCount": 0,
            "maximumCount": 1
        },
        "permissions": {
            "allowedAccounts": [
                "account-id"
            ٦
        },
        "properties": {
         "description": "Owned build action parameter description",
         "optional": true,
         "noEcho": false,
         "key": true,
         "queryable": false,
         "name": "Host"
         7
    }
}
```

3. 터미널 또는 명령줄에서 update-action-type 명령 실행

aws codepipeline update-action-type --cli-input-json file://action.json

이 명령은 업데이트된 파라미터와 일치하는 작업 유형 출력을 반환합니다.

CodePipeline에서 사용자 지정 작업 생성 및 추가

AWS CodePipeline 에는 자동 릴리스 프로세스에 대한 빌드, 테스트 및 배포 리소스를 구성하는 데 도 움이 되는 여러 작업이 포함되어 있습니다. 릴리스 프로세스에 내부적으로 개발된 빌드 프로세스 또는 테스트 제품군 등의 기본 작업에 포함되지 않은 작업이 포함되어 있는 경우 해당 목적에 대한 사용자 지정 작업을 생성하고 이를 파이프라인에 포함할 수 있습니다. AWS CLI 를 사용하여 AWS 계정과 연 결된 파이프라인에서 사용자 지정 작업을 생성할 수 있습니다.

다음 작업 범주에 대한 사용자 지정 AWS CodePipeline 작업을 생성할 수 있습니다.

• 항목을 빌드 또는 변환하는 고객 빌드 작업

• 하나 이상의 서버, 웹 사이트 또는 리포지토리에 항목을 배포하는 고객 배포 작업

- 자동화된 테스트를 구성 및 실행하는 고객 테스트 작업
- 함수를 실행하는 고객 호출 작업

사용자 지정 작업을 생성하면 이 사용자 지정 작업의 작업 요청에 대해 CodePipeline을 폴링하고 작업을 실행하며 상태 결과를 CodePipeline에 반환하는 작업자도 생성해야 합니다. 이 작업자는 CodePipeline의 퍼블릭 엔드포인트에 액세스할 수 있는 경우에 한해 컴퓨터 또는 리소스에서 찾을 수 있습니다. 액세스 및 보안을 손쉽게 관리하려면 Amazon EC2 인스턴스에 작업자를 호스팅합니다.

다음 다이어그램에서는 사용자 지정 빌드 작업이 포함된 파이프라인을 세부적으로 보여 줍니다.



CodePipeline

파이프라인에 단계의 일부로 사용자 지정 작업이 포함되어 있으면 파이프라인에서 작업 요청이 생성 됩니다. 사용자 지정 작업자가 해당 요청을 감지하고 해당 작업을 수행합니다(이 예에서는 타사 빌드 소프트웨어를 사용하는 사용자 지정 프로세스). 작업이 완료되면 작업자가 성공 결과 또는 실패 결과를 반환합니다. 성공 결과가 수신된 경우 파이프라인은 개정 사항과 아티팩트를 다음 작업에 제공합니다. 실패가 반환된 경우 파이프라인은 수정 사항을 파이프라인의 다음 작업에 제공하지 않습니다.

Note

이러한 지침에서는 CodePipeline 시작하기의 단계를 이미 완료한 것으로 가정합니다.

주제

- <u>사용자 지정 작업 생성</u>
- 사용자 지정 작업에 대한 작업자 생성
- 파이프라인에 사용자 지정 작업 추가

사용자 지정 작업 생성

를 사용하여 사용자 지정 작업을 생성하려면 AWS CLI

 텍스트 편집기를 열고 작업 범주, 작업 공급자 및 사용자 지정 작업에 필요한 모든 설정이 들어 있 는 사용자 지정 작업용 JSON 파일을 생성합니다. 예를 들어, 하나의 속성만 필요로 하는 사용자 지정 빌드 작업을 생성하려면 다음과 같은 JSON 파일을 생성합니다.

```
{
    "category": "Build",
    "provider": "My-Build-Provider-Name",
    "version": "1",
    "settings": {
        "entityUrlTemplate": "https://my-build-instance/job/{Config:ProjectName}/",
        "executionUrlTemplate": "https://my-build-instance/job/
{Config:ProjectName}/lastSuccessfulBuild/{ExternalExecutionId}/"
    },
    "configurationProperties": [{
        "name": "ProjectName",
        "required": true,
        "key": true,
        "secret": false,
        "queryable": false,
        "description": "The name of the build project must be provided when this
 action is added to the pipeline.",
        "type": "String"
    }],
    "inputArtifactDetails": {
        "maximumCount": integer,
        "minimumCount": integer
    },
    "outputArtifactDetails": {
        "maximumCount": integer,
        "minimumCount": integer
    },
    "tags": [{
```
```
"key": "Project",
    "value": "ProjectA"
}]
}
```

이 예제는 사용자 지정 작업에 Project 태그 키와 ProjectA 값을 포함하여 사용자 지정 작업에 태그 지정을 추가합니다. CodePipeline의 리소스 태깅에 대한 자세한 내용은 <u>리소스에 태그 지정</u> 섹션을 참조하세요.

JSON 파일에는 entityUrlTemplate과 executionUrlTemplate의 두 가지 속성이 있습니 다. 구성 속성이 필수이며 보안 사항이 아닌 경우, {Config:*name*}의 형식을 따라 URL 템플릿 내 사용자 지정 작업의 구성 속성에 있는 이름을 참조할 수 있습니다. 예를 들어, 위의 샘플에서 entityUrlTemplate 값은 구성 속성 *ProjectName*을 참조합니다.

- entityUrlTemplate: 작업의 서비스 공급자에 대한 정보를 제공하는 정적 링크입니다. 예에 서는 빌드 시스템에 각 빌드 프로젝트에 대한 정적 링크가 포함되어 있습니다. 링크 형식은 빌드 제공자에 따라 다를 수 있습니다(또는 테스트나 다른 서비스 공급자와 같이 서로 다른 작업 유형 을 생성하는 경우). 사용자 지정 작업이 추가될 때 사용자가 이 링크를 선택하여 빌드 프로젝트 (또는 테스트 환경)에 대한 세부 사항을 제공하는 웹 사이트의 페이지로 브라우저를 열 수 있도 록 이 링크 형식을 제공해야 합니다.
- executionUrlTemplate: 작업의 현재 실행 또는 가장 최근의 실행에 대한 정보로 업데이트 되는 동적 링크입니다. 사용자 지정 작업 작업자가 작업의 상태를 업데이트하면(예: 성공, 실패 또는 진행 중), 링크를 완성하는 데 사용되는 externalExecutionId도 제공됩니다. 이 링크 를 사용하면 작업 실행에 대한 세부 정보를 제공할 수 있습니다.

예를 들어, 파이프라인의 작업을 확인하면, 다음과 같은 두 개의 링크가 표시됩니다.



1>

이 정적 링크는 사용자 지정 작업을 추가하면 표시되며 entityUrlTemplate의 주소를 가리킵니 다. 이 주소는 사용자 지정 작업을 생성할 때 지정합니다.

2

이 동적 링크는 작업을 실행할 때마다 업데이트되며 executionUrlTemplate의 주소를 가리킵 니다. 이 주소는 사용자 지정 작업을 생성할 때 지정합니다.

이러한 링크 유형과 RevisionURLTemplate 및 ThirdPartyURL에 대한 자세한 내용은 <u>CodePipeline API 참조</u>의 <u>ActionTypeSettings</u> 및 <u>CreateCustomActionType</u>을 참조하세요. 작업 구조 요구사항 및 작업을 생성하는 방법에 대한 자세한 내용은 <u>CodePipeline 파이프라인 구조 참</u> 조 단원을 참조하십시오.

- 2. JSON 파일을 저장하고 기억하기 쉬운 이름을 지정합니다(예: MyCustomAction.json).
- 3. AWS CLI를 설치한 컴퓨터에서 터미널 세션(Linux, OS X, Unix)이나 명령 프롬프트(Windows)를 입니다.
- 4. AWS CLI 를 사용하여 방금 생성한 JSON 파일의 이름을 지정하여 aws codepipeline createcustom-action-type 명령을 실행합니다.

예를 들어, 빌드 사용자 지정 작업을 만들려면 다음과 같이 입력합니다.

Important

파일 이름 앞에 file://를 포함해야 합니다. 이 명령에 필수적입니다.

```
aws codepipeline create-custom-action-type --cli-input-json
file://MyCustomAction.json
```

5. 이 명령은 생성한 사용자 지정 작업의 전체 구조뿐만 아니라 자동으로 추가된 JobList 작업 구성 속성도 반환합니다. 파이프라인에 사용자 지정 작업을 추가하면 JobList를 사용하여 작업에 대 해 폴링할 수 있는 공급자의 프로젝트를 지정할 수 있습니다. 이를 구성하지 않으면 사용자 지정 작업 작업자가 작업에 대해 폴링할 때 사용 가능한 모든 작업이 반환됩니다.

예를 들어 이전 명령은 다음과 유사한 구조를 반환할 수 있습니다.

```
{
    "actionType": {
        "inputArtifactDetails": {
            "maximumCount": 1,
            "minimumCount": 1
       },
       "actionConfigurationProperties": [
            {
                "secret": false,
                "required": true,
                "name": "ProjectName",
                "key": true,
                "description": "The name of the build project must be provided when
 this action is added to the pipeline."
            }
        ],
        "outputArtifactDetails": {
            "maximumCount": 0,
            "minimumCount": 0
        },
        "id": {
            "category": "Build",
            "owner": "Custom",
            "version": "1",
            "provider": "My-Build-Provider-Name"
        },
```

```
"settings": {
    "entityUrlTemplate": "https://my-build-instance/job/
{Config:ProjectName}/",
    "executionUrlTemplate": "https://my-build-instance/job/mybuildjob/
lastSuccessfulBuild/{ExternalExecutionId}/"
    }
}
```

id 섹션에는 create-custom-action-type 명령 출력의 일부로 "owner": "Custom"이 포 함됩니다. CodePipeline은 Custom을 사용자 지정 작업 유형의 소유자로 자동으로 할당합 니다. 이 값은 create-custom-action-type 명령이나 update-pipeline 명령을 사용할 때는 할 당하거나 변경할 수 없습니다.

사용자 지정 작업에 대한 작업자 생성

사용자 지정 작업을 수행하려면 사용자 지정 작업에 대한 작업 요청에 대해 CodePipeline을 폴링하고 작업을 실행하고 상태 결과를 CodePipeline에 반환하는 작업자가 필요합니다. 작업자는 CodePipeline 의 퍼블릭 엔드포인트에 액세스할 수 있는 경우에 한해 컴퓨터 또는 리소스에서 찾을 수 있습니다.

작업자를 설계하는 방법에는 여러 가지가 있습니다. 다음 단원에서는 CodePipeline에 대한 사용자 지 정 작업자를 개발하기 위한 몇 가지 실용적 지침을 제공합니다.

주제

- 작업자에 대한 권한 관리 전략 선택 및 구성
- 사용자 지정 작업에 대한 작업자 개발
- 사용자 지정 작업자 아키텍처 및 예

작업자에 대한 권한 관리 전략 선택 및 구성

CodePipeline에서 사용자 지정 작업에 대한 사용자 지정 작업자를 개발하려면 사용자 및 권한 관리의 통합을 위한 전략이 필요합니다.

가장 간단한 전략은 통합에 필요한 리소스를 손쉽게 스케일 업할 수 있도록 IAM 인스턴스 역할을 사용 하여 Amazon EC2 인스턴스를 생성함으로써 사용자 지정 작업자에 필요한 인프라를 추가하는 것입니 다. 와의 기본 통합을 사용하여 사용자 지정 작업자와 CodePipeline 간의 상호 작용을 간소화 AWS 할 수 있습니다.

Amazon EC2 인스턴스를 설정하려면

- Amazon EC2에 대해 자세히 알아보고 해당 통합 사례에 적합한 선택인지 확인합니다. 자세한 내 용은 Amazon EC2 - 가상 서버 호스팅을 참조하세요.
- 2. Amazon EC2 인스턴스 생성을 시작합니다. 자세한 내용은 Amazon EC2 Linux 인스턴스 시작하

 기를 참조하세요.

고려해야 할 다른 전략은 IAM에 ID 페더레이션을 사용하여 기존의 자격 증명 공급자 시스템 및 리소스 를 통합하는 것입니다. 이 전략은 기업 자격 증명 공급자가 이미 있거나 웹 자격 증명 공급자를 사용하 여 사용자를 지원하도록 이미 구성되어 있는 경우에 특히 유용합니다. ID 페더레이션을 사용하면 IAM 사용자를 생성하거나 관리할 필요 없이 CodePipeline을 포함한 AWS 리소스에 대한 보안 액세스 권한 을 부여할 수 있습니다. 암호 보안 요구 사항 및 자격 증명 교체에 대한 기능과 정책을 사용할 수 있습니 다. 샘플 애플리케이션을 고유한 설계를 위한 템플릿으로 사용할 수 있습니다.

ID 페더레이션을 설정하려면

- 1. IAM ID 페더레이션에 대해 자세히 알아봅니다. 자세한 내용은 연동 관리를 참조하십시오.
- <u>임시 액세스 부여를 위한 시나리오</u>의 예를 검토하여 사용자 지정 작업의 요구에 가장 잘 맞는 시나 리오를 식별하고 일시적으로 액세스합니다.
- 3. 다음과 같이 자신의 인프라와 관련 있는 자격 증명 연동의 코드 예제를 살펴봅니다.
 - Active Directory 사용 사례를 위한 자격 증명 연동 샘플 애플리케이션
- 자격 증명 연동 구성을 시작합니다. 자세한 내용은 IAM 사용 설명서의 <u>자격 증명 공급자 및 페더레</u> 이션 섹션을 참조하세요.

사용자 지정 작업 및 작업자를 실행할 때 다음 중 하나를 생성하여 AWS 계정 에서 사용합니다.

사용자는 AWS 외부에서와 상호 작용하려는 경우 프로그래밍 방식의 액세스가 필요합니다 AWS Management Console. 프로그래밍 방식 액세스를 부여하는 방법은에 액세스하는 사용자 유형에 따라 다릅니다 AWS.

사용자에게 프로그래밍 방식 액세스 권한을 부여하려면 다음 옵션 중 하나를 선택합니다.

프로그래밍 방식 액세스가 필 요한 사용자는 누구인가요?	То	액세스 권한을 부여하는 사용 자	
작업 인력 ID (IAM Identity Center가 관리하	임시 자격 증명을 사용하여 AWS CLI, AWS SDKs 또는	사용하고자 하는 인터페이스에 대한 지침을 따릅니다.	
는 사용자)	AWS APIS.	 자세한 AWS CLI내용 은 AWS Command Line Interface 사용 설명서의 <u>AWS CLI 를 사용하도록</u> 구성을 AWS IAM Identity <u>Center</u> 참조하세요. AWS SDKs, 도구 및 AWS APIs의 경우 SDK 및 도구 참조 안내서의 IAM Identity <u>Center 인증을</u> 참조하세요. AWS SDKs 	
IAM	임시 자격 증명을 사용하여 AWS CLI, AWS SDKs 또는 AWS APIs.	IAM 사용 설명서의 <u>AWS 리소</u> <u>스에서 임시 자격 증명 사용</u> 의 지침을 따릅니다.	
ΙΑΜ	(권장되지 않음) 장기 자격 증명을 사용하여 AWS CLI, AWS SDKs 또는 AWS APIs.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • 자세한 AWS CLI내용은 사 용 AWS Command Line Interface 설명서의 IAM 사용 자 자격 증명을 사용하여 인 증을 참조하세요. • AWS SDKs 및 도구의 경우 SDK 및 도구 참조 안내서 의 <u>장기 자격 증명을 사용하</u> 여 인증을 참조하세요. AWS SDKs	

프로그래밍 방식 액세스가 필 요한 사용자는 누구인가요?	То	액세스 권한을 부여하는 사용 자
		• AWS APIs 경우 <u>IAM 사용 설</u> 명서의 IAM 사용자의 액세스 키 관리를 참조하세요.

다음은 사용자 지정 작업자와 함께 사용하기 위해 생성할 수 있는 예제 정책입니다. 이 정책은 예로만 사용되며 있는 그대로 제공됩니다.



Note

AWSCodePipelineCustomActionAccess 관리형 정책을 사용해 보세요.

사용자 지정 작업에 대한 작업자 개발

권한 관리 전략을 선택한 후 작업자가 CodePipeline과 상호 작용하는 방법을 고려해야 합니다. 다음의 상위 수준 다이어그램에서는 빌드 프로세스에 대한 사용자 지정 작업과 작업자의 워크플로우를 보여 줍니다.



- 1. 작업자가 PollForJobs를 사용하여 작업에 대해 CodePipeline을 폴링합니다.
- 2. 파이프라인이 소스 단계의 변경 사항으로 인해 트리거되면(예: 개발자가 변경 사항을 커밋하는 경우) 자동화된 릴리스 프로세스가 시작됩니다. 프로세스는 사용자 지정 작업이 구성된 단계에 이를 때까지 계속 진행됩니다. 이 단계의 작업에 도달하면 CodePipeline이 작업을 대기열에 넣 습니다. 이 작업은 작업자가 PollForJobs를 다시 호출하여 상태를 가져오는 경우 나타납니다. PollForJobs에서 작업 세부 정보를 가져와서 작업자에 다시 전달합니다.
- 3. 작업자가 AcknowledgeJob을 호출하여 CodePipeline에 작업 승인을 전송합니다. CodePipeline이 작업자가 작업을 계속 수행해야 함을 나타내는 승인(InProgress)을 반환하거나 작업에 대해 폴링 하는 작업자가 둘 이상 있고 다른 작업자가 이미 작업을 신청한 경우 InvalidNonceException 오류 응답이 반환됩니다. InProgress 승인 이후에 CodePipeline은 결과가 반환될 때까지 기다립 니다.

- 4. 작업자가 수정에 대한 사용자 지정 작업을 시작하면 작업이 실행됩니다. 다른 작업과 함께 사용자 지정 작업이 작업자에 결과를 반환합니다. 빌드 사용자 지정 작업의 예에서는 작업이 Amazon S3 버 킷에서 아티팩트를 가져와서 빌드하고 성공적으로 빌드된 아티팩트를 Amazon S3 버킷으로 다시 푸시합니다.
- 5. 작업이 실행 중인 동안 작업자는 executionUrlTemplate에 링크를 채우는 데 사용되는 ExternalExecutionId 정보뿐만 아니라 연속 토큰(작업자에 의해 생성된 작업 상태 직렬화, 예를 들어 JSON 형식의 빌드 식별자 또는 Amazon S3 객체 키)을 사용하여 PutJobSuccessResult를 호출할 수 있습니다. 그러면 작동 링크가 있는 파이프라인의 콘솔 보기가 진행 중일 때 특정 작업 세 부 정보로 업데이트됩니다. 필수는 아니지만 사용자가 실행 중인 사용자 지정 작업의 상태를 볼 수 있게 해주므로 모범 사례입니다.

Put JobSuccessResult가 호출되면 작업이 완료된 것으로 간주됩니다. 새 작업이 연속 토큰을 포 함한 CodePipeline에서 생성됩니다. 이 작업은 작업자가 PollFor Jobs를 다시 호출하는 경우 나타 납니다. 이 새 작업은 작업의 상태를 확인하는 데 사용할 수 있으며 연속 토큰을 사용하여 반환하거 나 작업이 완료되면 연속 토큰 없이 반환합니다.

Note

작업자가 사용자 지정 작업에 필요한 모든 작업을 수행하는 경우 처리 중인 작업자를 최소 두 단계로 나눠야 합니다. 첫 단계에서는 작업에 대한 세부 정보 페이지를 설정합니다. 세부 정보 페이지를 생성했으면 작업자의 상태를 직렬화하고 크기 제한(in AWS CodePipeline 할 당량 참조)이 적용된 연속 토큰으로 반환할 수 있습니다. 예를 들어 연속 토큰으로 사용하는 문자열에 작업의 상태를 작성할 수 있습니다. 처리 중인 작업자의 두 번째 단계(및 후속 단 계)에서는 작업의 실제 작업을 수행합니다. 마지막 단계에서는 마지막 단계에 연속 토큰 없 이 CodePipeline에 성공 또는 실패를 반환합니다.

연속 토큰 사용에 대한 자세한 내용은 <u>CodePipeline API 참조</u>에서 PutJobSuccessResult의 사양 을 참조하세요.

- 사용자 지정 작업이 완료되면 작업자가 두 API 중 하나를 호출하여 사용자 지정 작업의 결과를 CodePipeline에 반환합니다.
 - PutJobSuccessResult(연속 토큰 없음), 사용자 지정 작업이 성공적으로 실행되었음을 나타냄
 - PutJobFailureResult, 사용자 지정 작업이 성공적으로 실행되지 않았음을 나타냄

결과에 따라 파이프라인은 다음 작업으로 계속 진행되거나(성공) 중지됩니다(실패).

사용자 지정 작업자 아키텍처 및 예

상위 수준 워크플로우를 매핑한 후 작업자를 생성할 수 있습니다. 결국 사용자 지정 작업의 세부 사항 이 작업자에 필요한 사항을 결정하지만 사용자 지정 작업의 대다수 작업자에는 다음 기능이 포함되어 있습니다.

- PollForJobs를 사용하여 CodePipeline의 작업을 폴링합니다.
- AcknowledgeJob, PutJobSuccessResult 및 PutJobFailureResult를 사용하여 작업 승인 및 CodePipeline에 결과 반환.
- 파이프라인에 대한 아티팩트를 Amazon S3 버킷에서 검색 및/또는 버킷에 배치. Amazon S3 버킷에 서 아티팩트를 다운로드하려면 서명 버전 4 서명(Sig V4)을 사용하는 Amazon S3 클라이언트를 생성해야 합니다. 에는 Sig V4가 필요합니다 AWS KMS.

Amazon S3 버킷에 아티팩트를 업로드하려면 암호화를 사용하도록 Amazon S3 <u>PutObject</u> 요청을 추가로 구성해야 합니다. 현재 encryption. AWS KMS uses에는 AWS Key Management Service(AWS KMS)만 지원됩니다 AWS KMS keys. AWS 관리형 키 또는 고객 관리형 키를 사용하 여 아티팩트를 업로드할지 여부를 확인하려면 사용자 지정 작업자가 <u>작업 데이터를</u> 살펴보고 <u>암호</u> <u>화 키</u> 속성을 확인해야 합니다. 속성이 설정된 경우 구성할 때 해당 고객 관리형 키 ID를 사용해야 합니다 AWS KMS. 키 속성이 null인 경우 AWS 관리형 키를 사용합니다. CodePipeline은 달리 구성 AWS 관리형 키 되지 않은 한를 사용합니다.

Java 또는 .NET에서 AWS KMS 파라미터를 생성하는 방법을 보여주는 예제는 <u>SDK를 사용하여</u> <u>Amazon S3 AWS Key Management Service 에서 지정을 AWS SDKs</u>. CodePipeline의 Amazon S3 버킷에 대한 자세한 내용은 CodePipeline 개념 섹션을 참조하세요.

사용자 지정 작업자의 더 복잡한 예는 GitHub에서 사용할 수 있습니다. 이 샘플은 오픈 소스이며 있는 그대로 제공됩니다.

• CodePipeline에 대한 샘플 작업자: GitHub 리포지토리에서 샘플을 다운로드합니다.

파이프라인에 사용자 지정 작업 추가

작업자가 있으면 파이프라인 생성 마법사를 사용할 때 새 파이프라인을 생성하고 선택하여 파이프라 인에 사용자 지정 작업을 추가하거나, 기존 파이프라인을 편집하고 사용자 지정 작업을 추가하거나, , AWS CLI SDKs 또는 APIs.

빌드 또는 배포 작업인 경우 사용자 지정 작업이 포함된 파이프라인 생성 마법사에서 파이프라 인을 생성할 수 있습니다. 사용자 지정 작업이 테스트 범주에 있는 경우 기존 파이프라인을 편 집하여 추가해야 합니다.

주제

• <u>기존 파이프라인에 사용자 지정 작업 추가(CLI)</u>

기존 파이프라인에 사용자 지정 작업 추가(CLI)

를 사용하여 기존 파이프라인 AWS CLI 에 사용자 지정 작업을 추가할 수 있습니다.

 터미널 세션(Linux, macOS 또는 Unix) 또는 명령 프롬프트(Windows)를 열고 get-pipeline 명령을 실행하여 편집하려는 파이프라인 구조를 JSON 파일에 복사합니다. 예를 들어 MyFirstPipeline이라는 파이프라인에 대해 다음 명령을 입력합니다.

aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json

이 명령은 아무 것도 반환하지 않지만 생성한 파일이 명령을 실행한 디렉터리에 표시되어야 합니 다.

 텍스트 편집기에서 JSON 파일을 열고 파일 구조를 수정하여 사용자 지정 작업을 기존 단계에 추 가합니다.

Note

해당 단계의 다른 작업과 동시에 작업을 실행하려면 작업에 해당 작업과 동일한 run0rder 값을 지정해야 합니다.

예를 들어, 파이프라인의 구조를 수정하여 Build라는 이름의 단계를 추가하고 해당 단계에 빌드 사 용자 지정 작업을 추가하려는 경우, 다음과 같이 JSON을 수정하여 배포 단계 전에 Build 단계를 추가해야 할 수 있습니다.

{

파이프라인에 사용자 지정 작업 추가

```
"name": "MyBuildStage",
"actions": [
        {
          "inputArtifacts": [
          {
             "name": "MyApp"
           }
             ],
              "name": "MyBuildCustomAction",
              "actionTypeId": {
                  "category": "Build",
                  "owner": "Custom",
                  "version": "1",
                  "provider": "My-Build-Provider-Name"
              },
              "outputArtifacts": [
                  {
                    "name": "MyBuiltApp"
                  }
              ],
              "configuration": {
                  "ProjectName": "MyBuildProject"
              },
              "runOrder": 1
          }
      ]
 },
 {
      "name": "Staging",
      "actions": [
              {
                  "inputArtifacts": [
                      {
                           "name": "MyBuiltApp"
                      }
                  ],
                  "name": "Deploy-CodeDeploy-Application",
                  "actionTypeId": {
                      "category": "Deploy",
                      "owner": "AWS",
                      "version": "1",
                      "provider": "CodeDeploy"
                  },
                  "outputArtifacts": [],
```



3. 변경 사항을 적용하려면 다음과 유사하게 파이프라인 JSON 파일을 지정하여 update-pipeline 명 령을 실행합니다.

\Lambda Important

파일 이름 앞에 file://를 포함해야 합니다. 이 명령에 필수적입니다.

aws codepipeline update-pipeline --cli-input-json file://pipeline.json

이 명령은 편집한 파이프라인의 전체 구조를 반환합니다.

4. CodePipeline 콘솔을 열고 방금 편집한 파이프라인의 이름을 선택합니다.

파이프라인에 변경 사항이 표시됩니다. 다음에 사용자가 소스 위치를 변경할 경우, 파이프라인의 개정된 구조를 통해 해당 개정이 실행됩니다.

CodePipeline에서 사용자 지정 작업 태그 지정

태그는 AWS 리소스와 연결된 키-값 페어입니다. CodePipeline에서 콘솔 또는 CLI를 사용하여 사용자 지정 작업에 태그를 적용할 수 있습니다. CodePipeline 리소스 태그 지정, 사용 사례, 태그 키 및 값 제 약, 지원되는 리소스 유형에 대한 자세한 내용은 <u>리소스에 태그 지정</u> 단원을 참조하세요.

사용자 지정 작업의 태그 값을 추가, 제거 및 업데이트할 수 있습니다. 각 사용자 지정 작업에 최대 50 개의 태그를 추가할 수 있습니다.

주제

- 사용자 지정 작업에 태그 추가
- 사용자 지정 작업에 대한 태그 보기

- 사용자 지정 작업에 대한 태그 편집
- 사용자 지정 작업에서 태그 제거

사용자 지정 작업에 태그 추가

다음 단계에 따라 AWS CLI 를 사용하여 사용자 지정 작업에 태그를 추가합니다. 사용자 지정 작업을 생성할 때 태그를 추가하려면 CodePipeline에서 사용자 지정 작업 생성 및 추가 단원을 참조하십시오.

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

터미널이나 명령줄에서 tag-resource 명령을 실행하여, 태그를 추가할 사용자 지정 작업의 Amazon 리 소스 이름(ARN)과 추가할 태그의 키와 값을 지정합니다. 사용자 지정 작업에 두 개 이상의 태그를 추가 할 수 있습니다. 예를 들어, 사용자 정의 작업에 태그 값이 UnitTest인 TestActionType이라는 태 그 키와 MyApplication이라는 태그 값을 가진 ApplicationName이라는 태그 키라는 두 개의 태그 를 사용하여 태그를 지정하려면 다음과 같이 하세요.

aws codepipeline tag-resource --resource-arn arn:aws:codepipeline:us-west-2:accountid:actiontype:Owner/Category/Provider/Version --tags key=TestActionType,value=UnitTest key=ApplicationName,value=MyApplication

성공한 경우 이 명령은 아무 것도 반환하지 않습니다.

사용자 지정 작업에 대한 태그 보기

다음 단계에 따라 AWS CLI 를 사용하여 사용자 지정 작업의 AWS 태그를 확인합니다. 태그가 추가되 지 않은 경우 반환되는 목록은 비어 있습니다.

터미널 또는 명령줄에서 list-tags-for-resource 명령을 실행합니다. 예

를 들어, ARN arn:aws:codepipeline:us-west-2:account-

id:actiontype: *Owner/Category/Provider/Version*을 사용하여 사용자 지정 작업에 대한 태 그 키 및 태그 값 목록을 보려면 다음을 수행하십시오.

aws codepipeline list-tags-for-resource --resource-arn arn:aws:codepipeline:uswest-2:account-id:actiontype:Owner/Category/Provider/Version

이 명령이 제대로 실행되면 다음과 비슷한 정보를 반환합니다.

```
"tags": {
    "TestActionType": "UnitTest",
    "ApplicationName": "MyApplication"
}
```

사용자 지정 작업에 대한 태그 편집

다음 단계에 따라 AWS CLI 를 사용하여 사용자 지정 작업의 태그를 편집합니다. 기존 키의 값을 변경 하거나 다른 키를 추가할 수 있습니다. 다음 단원에서 설명하는 것처럼 사용자 지정 작업에서 태그를 제거할 수도 있습니다.

터미널이나 명령줄에서 tag-resource 명령을 실행하여, 태그를 업데이트하고 태그 키 및 태그 값을 지 정할 사용자 지정 작업의 Amazon 리소스 이름(ARN)을 지정합니다.

```
aws codepipeline tag-resource --resource-arn arn:aws:codepipeline:us-
west-2:account-id:actiontype:Owner/Category/Provider/Version --tags
key=TestActionType,value=IntegrationTest
```

사용자 지정 작업에서 태그 제거

다음 단계에 따라 AWS CLI 를 사용하여 사용자 지정 작업에서 태그를 제거합니다. 연결된 리소스에서 태그를 제거하면 태그가 삭제됩니다.

Note

사용자 지정 작업을 삭제하면 삭제된 사용자 지정 작업에서 모든 태그 연결이 제거됩니다. 사 용자 지정 작업을 삭제하기 전에 태그를 제거할 필요가 없습니다.

터미널이나 명령줄에서 untag-resource 명령을 실행하여, 태그를 제거할 사용자 지정 작업 의 ARN과 제거할 태그의 태그 키를 지정합니다. 예를 들어, 사용자 지정 작업에서 태그 키가 *TestActionType*인 태그를 제거하려면 다음과 같이 하십시오.

aws codepipeline untag-resource --resource-arn arn:aws:codepipeline:us-west-2:accountid:actiontype:Owner/Category/Provider/Version --tag-keys TestActionType

성공한 경우 이 명령은 아무 것도 반환하지 않습니다. 사용자 지정 작업과 연결된 태그를 확인하려면 list-tags-for-resource 명령을 실행하십시오.

사용자 지정 작업에 대한 태그 편집

CodePipeline의 파이프라인에서 AWS Lambda 함수 호출

AWS Lambda은(는) 서버를 프로비저닝하거나 관리하지 않고도 코드를 실행할 수 있게 해주는 컴퓨팅 서비스입니다. Lambda 함수를 생성하고 파이프라인에서 작업으로 추가할 수 있습니다. Lambda를 통 해 함수를 작성하여 거의 모든 작업을 수행할 수 있으므로 파이프라인이 작동하는 방식을 사용자 지정 할 수 있습니다.

▲ Important

CodePipeline이 Lambda로 전송하는 JSON 이벤트를 기록하지 마십시오. 이렇게 하면 사용자 보안 인증이 CloudWatch Logs에 기록될 수 있기 때문입니다. CodePipeline 역할은 JSON 이 벤트를 사용하여 artifactCredentials 필드의 Lambda에 임시 보안 인증을 전달합니다. 예제 이벤트는 예제 JSON 이벤트을 참조하세요.

파이프라인에서 Lambda 함수를 사용할 수 있는 몇 가지 방법은 다음과 같습니다.

- 를 사용하여 파이프라인의 한 단계에서 온디맨드 리소스를 AWS CloudFormation 생성하고 다른 단 계에서 삭제합니다.
- CNAME 값을 스왑하는 Lambda 함수 AWS Elastic Beanstalk 를 사용하여 가동 중지 시간이 없는 애 플리케이션 버전을에 배포합니다.
- Amazon ECS Docker 인스턴스에 배포합니다.
- AMI 스냅샷을 생성하여 빌드하거나 배포하기 전에 리소스를 백업합니다.
- IRC 클라이언트에 메시지를 게시하는 등 타사 제품과의 통합을 파이프라인에 추가합니다.

Note

Lambda 함수를 생성하고 실행하면 AWS 계정에 요금이 부과될 수 있습니다. 자세한 내용은 <u>요</u> 금을 참조하세요.

이 주제에서는 AWS CodePipeline 및에 익숙 AWS Lambda 하고 파이프라인, 함수, 파이프라인, 함수 가 의존하는 IAM 정책 및 역할을 생성하는 방법을 알고 있다고 가정합니다. 이 주제에서는 다음 작업을 하는 방법을 보여줍니다.

• 웹 페이지가 성공적으로 배포되었는지 여부를 테스트하는 Lambda 함수를 생성합니다.

- 파이프라인의 일부로 함수를 실행하는 데 필요한 CodePipeline 및 Lambda 실행 역할과 권한을 구성 합니다.
- 파이프라인을 편집하여 작업으로 Lambda 함수를 추가합니다.
- 변경 사항을 수동으로 배포하여 작업을 테스트합니다.

CodePipeline에서 리전 간 Lambda 간접 호출 작업을 사용하는 경우 <u>PutJobSuccessResult</u> 및 <u>PutJobFailureResult</u>를 사용한 lambda 실행 상태는 Lambda 함수가 있는 AWS 리전으로 전송 해야 하며 CodePipeline이 있는 리전으로 전송해서는 안 됩니다.

이 주제에는 CodePipeline에서 Lambda 함수 작업의 유연성을 보여주는 샘플 함수가 포함되어 있습니다.

- Basic Lambda function
 - CodePipeline에서 사용할 기본 Lambda 함수 생성.
 - 작업에 대한 세부 정보 링크의 CodePipeline에 성공 또는 실패 결과 반환.
- AWS CloudFormation 템플릿을 사용하는 샘플 Python 함수
 - JSON으로 인코딩된 사용자 파라미터를 사용하여 함수에 여러 구성 값 전달 (get_user_params).
 - 아티팩트 버킷의 .zip 아티팩트와 상호 작용(get_template).
 - 연속 토큰을 사용하여 장기 실행 비동기 프로세스 모니터링(continue_job_later). 이를 통해 15분 런타임(Lambda 제한)을 초과한 경우에도 작업을 계속 진행하고 함수를 성공적으로 실행할 수 있습니다.

각 샘플 함수에는 역할에 추가해야 하는 권한에 대한 정보가 포함되어 있습니다. 의 제한에 대 한 자세한 내용은 AWS Lambda 개발자 안내서의 제한을 AWS Lambda참조하세요. <u>https://</u> docs.aws.amazon.com/lambda/latest/dg/limits.html

A Important

이 주제에 포함된 샘플 코드, 역할 및 정책은 예제에 불과하며, 있는 그대로 제공됩니다.

주제

- 1단계: 파이프라인 생성
- 2단계: Lambda 함수 생성
- 3단계: CodePipeline 콘솔의 파이프라인에 Lambda 함수 추가
- 4단계: Lambda 함수로 파이프라인 테스트
- 5단계: 다음 절차
- 예제 JSON 이벤트
- <u>추가 샘플 함수</u>

1단계: 파이프라인 생성

이 단계에서는 나중에 Lambda 함수를 추가할 파이프라인을 생성합니다. 이는 <u>CodePipeline 자습서</u>에 서 생성한 동일한 파이프라인입니다. 해당 파이프라인이 여전히 계정에 대해 구성되어 있고 Lambda 함수를 생성할 동일한 리전에 있는 경우 이 단계를 건너뛸 수 있습니다.

파이프라인을 생성하려면

- <u>자습서: 간단한 파이프라인 생성(S3 버킷)</u>의 처음 세 단계를 따라 Amazon S3 버킷, CodeDeploy 리소스 및 두 단계 파이프라인을 생성합니다. 인스턴스 유형으로 Amazon Linux 옵션을 선택합니다. 파이프라인 이름은 원하는 이름을 사용할 수 있지만, 이 주제의 단계에서는 MyLambdaTestPipeline을 사용합니다.
- 파이프라인의 상태 페이지의 CodeDeploy 작업에서 세부 정보를 선택합니다. 배포 그룹의 배포 세 부 정보 페이지에 있는 목록에서 인스턴스 ID를 선택합니다.
- 3. Amazon EC2 콘솔에서 해당 인스턴스의 세부 정보 탭에 있는 퍼블릭 IPv4 주소(예: **192.0.2.4**) 의 IP 주소를 복사합니다. 이 주소는 AWS Lambda에서 함수의 대상으로 사용합니다.

CodePipeline의 기본 서비스 역할 정책에는 함수를 호출하는 데 필요한 Lambda 권한이 포함 되어 있습니다. 그러나 기본 서비스 역할을 수정하거나 다른 서비스 역할을 선택한 경우에는 역할에 대한 정책이 lambda:InvokeFunction 및 lambda:ListFunctions 권한을 허용 하는지 확인하십시오. 그렇지 않으면 Lambda 작업이 포함된 파이프라인이 실패합니다.

Note

2단계: Lambda 함수 생성

이 단계에서는 HTTP 요청을 하고 웹 페이지의 텍스트 행을 확인하는 Lambda 함수를 생성합니다. 이 단계의 일부로 IAM 정책 및 Lambda 실행 역할도 생성해야 합니다. 자세한 내용은 AWS Lambda 개발 자 안내서의 권한 모델 단원을 참조하세요.

실행 역할을 만들려면

- 1. 에 로그인 AWS Management Console 하고 <u>https://console.aws.amazon.com/iam/</u>://https://https:// https://https://://https://://https://://https://://https://://
- 정책을 선택한 후 정책 생성을 선택합니다. JSON 탭에서 다음과 같은 정책을 필드에 붙여 넣습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Action": [
        "codepipeline:PutJobSuccessResult",
        "codepipeline:PutJobFailureResult"
        ],
        "Effect": "Allow",
        "Resource": "*"
     }
  ]
}
```

- 3. 정책 검토를 선택합니다.
- 정책 검토 페이지의 이름에서 정책 이름(예: CodePipelineLambdaExecPolicy)을 입력합니다. 설명에 Enables Lambda to execute code을 입력합니다.

정책 생성(Create Policy)을 선택합니다.

이러한 권한은 Lambda 함수가 CodePipeline 및 Amazon CloudWatch와 상호 작용하는 데 필요한 최소 권한입니다. 다른 AWS 리소스와 상호 작용하는 함수를 허용하도록이 정책을 확장하려면 해당 Lambda 함수에 필요한 작업을 허용하도록이 정책을 수정해야 합니다.

- 5. 정책 대시보드 페이지에서 역할을 선택한 다음, 역할 만들기를 선택합니다.
- 6. 역할 생성생성 페이지에서 AWS 서비스를 선택합니다. Lambda를 선택한 후 다음: 권한을 선택합니다.
- 7. 권한 정책 연결 페이지에서 CodePipelineLambdaExecPolicy 옆에 있는 확인란을 선택한 후 다음: 태그를 선택합니다. 다음: 검토를 선택합니다.
- 8. 검토 페이지의 역할 이름에 이름을 입력하고 역할 만들기를 선택합니다.

샘플 Lambda 함수를 생성하여 CodePipeline과 함께 사용하려면

- 1. 에 로그인 AWS Management Console 하고 <u>https://console.aws.amazon.com/lambda/</u>://https:// https://https://https://https://https://https://https://https://
- 2. 함수 페이지에서 함수 생성을 선택합니다.

Note

Lambda 페이지 대신 시작 페이지가 표시되는 경우 지금 시작하기를 선택하세요.

- 3. 함수 생성 페이지에서 처음부터 새로 작성을 선택합니다. 함수 이름에 Lambda 함수의 이름을 입 력합니다(예: **MyLambdaFunctionForAWSCodePipeline**). 런타임에서 Node.js 20.x를 선택합 니다.
- 역할에서 기존 역할 선택을 선택합니다. 기존 역할에서 역할을 선택한 다음 함수 생성을 선택합니다.
 다.

생성된 기능에 대한 세부 정보 페이지가 열립니다.

5. 함수 코드 상자에 다음 코드를 붙여 넣습니다.

CodePipeline.job 키 아래의 이벤트 객체에는 <u>작업 세부 정보</u>가 들어 있습니다. CodePipeline이 Lambda를 반환하는 JSON 이벤트의 전체 예제는 <u>예제 JSON 이벤트</u> 단 원을 참조하세요.

```
import { CodePipelineClient, PutJobSuccessResultCommand,
PutJobFailureResultCommand } from "@aws-sdk/client-codepipeline";
import http from 'http';
import assert from 'assert';
export const handler = (event, context) => {
   const codepipeline = new CodePipelineClient();
   // Retrieve the Job ID from the Lambda action
   const jobId = event["CodePipeline.job"].id;
   // Retrieve the value of UserParameters from the Lambda action configuration in
CodePipeline, in this case a URL which will be
   // health checked by this function.
   const url =
 event["CodePipeline.job"].data.actionConfiguration.configuration.UserParameters;
   // Notify CodePipeline of a successful job
    const putJobSuccess = async function(message) {
        const command = new PutJobSuccessResultCommand({
            jobId: jobId
       });
        try {
             await codepipeline.send(command);
             context.succeed(message);
         } catch (err) {
             context.fail(err);
         }
   };
   // Notify CodePipeline of a failed job
    const putJobFailure = async function(message) {
        const command = new PutJobFailureResultCommand({
```

```
jobId: jobId,
           failureDetails: {
               message: JSON.stringify(message),
               type: 'JobFailed',
               externalExecutionId: context.awsRequestId
           }
       });
       await codepipeline.send(command);
       context.fail(message);
   };
  // Validate the URL passed in UserParameters
   if(!url || url.indexOf('http://') === -1) {
       putJobFailure('The UserParameters field must contain a valid URL address to
test, including http:// or https://');
       return;
   }
  // Helper function to make a HTTP GET request to the page.
  // The helper will test the response and succeed or fail the job accordingly
   const getPage = function(url, callback) {
       var pageObject = {
           body: '',
           statusCode: 0,
           contains: function(search) {
               return this.body.indexOf(search) > -1;
           }
       };
       http.get(url, function(response) {
           pageObject.body = '';
           pageObject.statusCode = response.statusCode;
           response.on('data', function (chunk) {
               pageObject.body += chunk;
           });
           response.on('end', function () {
               callback(pageObject);
           });
           response.resume();
       }).on('error', function(error) {
           // Fail the job if our request failed
           putJobFailure(error);
```

```
});
    };
    getPage(url, function(returnedPage) {
        try {
            // Check if the HTTP response has a 200 status
            assert(returnedPage.statusCode === 200);
            // Check if the page contains the text "Congratulations"
            // You can change this to check for different text, or add other tests
 as required
            assert(returnedPage.contains('Congratulations'));
            // Succeed the job
            putJobSuccess("Tests passed.");
        } catch (ex) {
            // If any of the assertions failed then fail the job
            putJobFailure(ex);
        }
    });
};
```

- 핸들러를 기본값 그대로 두고 역할도 기본값 CodePipelineLambdaExecRole로 그대로 둡니 다.
- 7. Basic settings(기본 설정)의 제한 시간에 20초를 입력합니다.
- 8. 저장(Save)을 선택합니다.

3단계: CodePipeline 콘솔의 파이프라인에 Lambda 함수 추가

이 단계에서는 새 단계를 파이프라인에 추가한 다음 함수를 호출하는 Lambda 작업을 해당 단계에 추 가합니다.

단계를 추가하려면

- 1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> <u>codepipeline/home</u>://https//https://https
- 2. Welcome(시작) 페이지에서, 생성한 파이프라인을 선택합니다.
- 3. 파이프라인 보기 페이지에서 편집을 선택합니다.
- 4. 편집 페이지에서 + 단계 추가를 선택하여 CodeDeploy 작업으로 배포 단계 다음에 단계를 추가합 니다. 역할 이름을 입력(예: LambdaStage)한 후, Add stage(단계 추가)를 선택합니다.

기존 단계에 Lambda 작업을 추가하도록 선택할 수도 있습니다. 데모용으로, 여기서는 Lambda 함수를 단계의 유일한 작업으로 추가하여 파이프라인을 통해 아티팩트의 진행 상 황을 쉽게 확인할 수 있도록 합니다.

5. + Add action group(작업 그룹 추가)을 선택합니다. 작업 편집의 작업 이름에 Lambda 작업 이름을 입력합니다(예: MyLambdaAction). 공급자에서 AWS Lambda를 선택합니다. 함수 이름에 Lambda 함수의 이름(예: MyLambdaFunctionForAWSCodePipeline)을 선택하 거나 입력합니다. 사용자 파라미터에서 이전에 복사한 Amazon EC2 인스턴스의 IP 주소(예: http://192.0.2.4)를 지정한 다음 완료를 선택합니다.

Note

이 주제에서는 IP 주소를 사용하지만, 실제 시나리오에서는 사용자의 등록된 웹 사이트 이름을 대신 제공할 수 있습니다(예: http://www.example.com). 의 이벤트 데이터 및 핸들러에 대한 자세한 내용은 AWS Lambda 개발자 안내서의 <u>프로그래밍 모델을</u> AWS Lambda참조하세요.

6. 작업 편집 페이지에서 저장을 선택합니다.

4단계: Lambda 함수로 파이프라인 테스트

함수를 테스트하려면 파이프라인을 통해 최근 변경 사항을 배포합니다.

콘솔을 사용하여 가장 최근의 아티팩트 버전을 파이프라인을 통해 실행하려면

- 파이프라인 세부 정보 페이지에서 변경 사항 릴리스를 선택합니다. 이렇게 하면 소스 작업에 지정 된 각 소스 위치에서 사용 가능한 가장 최근의 개정이 파이프라인을 통해 실행됩니다.
- Lambda 작업이 완료되면, 세부 정보 링크를 선택하여 Amazon CloudWatch의 함수에 대한 로그 스트림(이벤트의 청구 기간 등)을 확인합니다. 함수가 실패하면 CloudWatch 로그에서 원인에 대 한 정보가 제공됩니다.

5단계: 다음 절차

이제 Lambda 함수를 성공적으로 생성하여 파이프라인에서 작업으로 추가했으므로 다음 작업을 수행 할 수 있습니다.

- Lambda 작업을 단계에 더 추가하여 다른 웹 페이지를 확인합니다.
- Lambda 함수를 수정하여 다른 텍스트 문자열을 확인합니다.
- Lambda 함수를 살펴보고 고유한 Lambda 함수를 생성하여 파이프라인에 추가합니다.

⊘ Source				View current re	visions
Source CodeCommit Succeeded - 1 minute ago	١				
Source: Added codepipeline-codecomn	nit-events.ym	N dbb74de			
Disable transition					
♥ ⊙ Deploy				View current re	visions
Deploy CodeDeploy Succeeded - 1 minute ago Details	١				
Source: Added codepipeline-codecomn	nit-events.ym	nl dbb74de			
Disable transition					
> LambdaStage				View current re	visions
MyLambdaAction1	١	MyLambdaAction2	١	MyLambdaAction3	٤
Lambda 🖾		Lambda 🖸		Lambda 🖸	

Lambda 함수 실험을 완료한 후에는 요금이 부과되지 않도록 파이프라인에서 제거하고, 삭제하고 AWS Lambda, IAM에서 역할을 삭제하는 것이 좋습니다. 자세한 내용은 <u>CodePipeline에서 파이프라인</u> 편집, <u>CodePipeline에서 파이프라인 삭제</u>, <u>역할 또는 인스턴스 프로파일 삭제</u>를 참조하십시오.

예제 JSON 이벤트

다음 예에서는 CodePipeline을 통해 Lambda에 전송된 샘플 JSON 이벤트를 보여 줍니다. 이 이벤트 의 구조는 <u>GetJobDetails API</u>에 대한 응답과 비슷하지만 actionTypeId 및 pipelineContext 데이터 형식이 없습니다. 두 가지 작업 구성 세부 정보, 즉 FunctionName 및 UserParameters는 JSON 이벤트와 GetJobDetails API에 대한 응답 둘 다에 포함되어 있습니다. ### #### ###의 값 은 예 또는 설명이고 실제 값이 아닙니다.

```
{
    "CodePipeline.job": {
        "id": "11111111-abcd-1111-abcd-111111abcdef",
        "accountId": "111111111111",
        "data": {
            "actionConfiguration": {
                "configuration": {
                    "FunctionName": "MyLambdaFunctionForAWSCodePipeline",
                    "UserParameters": "some-input-such-as-a-URL"
                }
            },
            "inputArtifacts": [
                {
                    "location": {
                        "s3Location": {
                            "bucketName": "the name of the bucket configured as the
pipeline artifact store in Amazon S3, for example codepipeline-us-east-2-1234567890",
                            "objectKey": "the name of the application, for example
CodePipelineDemoApplication.zip"
                        },
                        "type": "S3"
                    },
                    "revision": null,
                    "name": "ArtifactName"
                }
            ],
            "outputArtifacts": [],
            "artifactCredentials": {
                "secretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
                "sessionToken": "MIICiTCCAfICCQD6m7oRw0uX0jANBgkghkiG9w
ØBAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZ
WF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBqNVBAsTC01BTSBDb25zb2x1MRIw
EAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5
jb20wHhcNMTEwNDI1MjA0NTIxWhcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBh
MCVVMxCzAJBgNVBAgTA1dBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBb
```

WF6b24xFDASBgNVBAsTC01BTSBDb25zb2x1MRIwEAYDVQQDEw1UZXN0Q21sYWMx HzAdBqkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wqZ8wDQYJKoZIhvcNAQE BBQADqY0AMIGJAoGBAMaK0dn+a4GmWIWJ21uUSfwfEvySWtC2XADZ4nB+BLYqVI k60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wqc/MbQ ITx0USQv7c7ugFFDzQGBzZswY6786m86gpEIbb30hjZnzcvQAaRHhdlQWIMm2nr AqMBAAEwDQYJKoZIhvcNAQEFBQADqYEAtCu4nUhVVxYUntneD9+h8Mq9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZqXL0FkbFFBjvSfpJIlJ00zbhNYS5f6Guo EDmFJl0ZxBHjJnyp3780D8uTs7fLvjx79LjSTbNYiytVbZPQUQ5Yaxu2jXnimvw 3rrszlaEXAMPLE=", "accessKeyId": "AKIAIOSFODNN7EXAMPLE" }, "continuationToken": "A continuation token if continuing job", "encryptionKey": { "id": "arn:aws:kms:uswest-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab", "type": "KMS" } } } }

추가 샘플 함수

다음 샘플 Lambda 함수는 CodePipeline의 파이프라인에서 사용할 수 있는 추가 기능을 보여 줍니다. 이러한 함수를 사용하려면 각 샘플의 소개에 명시된 대로 Lambda 실행 역할에 대한 정책을 수정해야 할 수도 있습니다.

주제

• <u>AWS CloudFormation 템플릿을 사용하는 샘플 Python 함수</u>

AWS CloudFormation 템플릿을 사용하는 샘플 Python 함수

다음 샘플은 제공된 AWS CloudFormation 템플릿을 기반으로 스택을 생성하거나 업데이트하는 함수 를 보여줍니다. 템플릿이 Amazon S3 버킷을 생성합니다. 이는 비용을 최소화하기 위한 데모용일 뿐입 니다. 이상적으로는 버킷에 어떤 것을 업로드하기 전에 스택을 삭제해야 합니다. 버킷에 파일을 업로드 한 경우에는 스택을 삭제할 때 버킷을 삭제할 수 없습니다. 버킷 자체를 삭제하기 전에 버킷에 있는 모 든 것을 수동으로 삭제해야 합니다.

이 Python 샘플에서는 Amazon S3 버킷을 소스 작업으로 사용하거나 파이프라인과 함께 사용할 수 있는 버전이 지정된 Amazon S3 버킷에 액세스할 수 있는 파이프라인이 있다고 가정합니다. AWS CloudFormation 템플릿을 생성하고 압축한 다음 해당 버킷에 .zip 파일로 업로드합니다. 그런 다음 버 킷에서 이 .zip 파일을 검색하는 소스 작업을 파이프라인에 추가해야 합니다.

Note

Amazon S3가 파이프라인의 소스 공급자인 경우, 소스 파일을 .zip 하나로 압축하고 그 .zip을 소스 버킷에 업로드할 수 있습니다. 압축이 풀린 단일 파일을 업로드할 수도 있지만 .zip 파일을 예상하는 다운스트림 작업은 실패합니다.

이 샘플에서 보여 주는 작업은 다음과 같습니다.

- JSON으로 인코딩된 사용자 파라미터를 사용하여 함수에 여러 구성 값 전달(get_user_params).
- 아티팩트 버킷의 .zip 아티팩트와 상호 작용(get_template).
- 연속 토큰을 사용하여 장기 실행 비동기 프로세스 모니터링(continue_job_later). 이를 통해 15 분 런타임(Lambda 제한)을 초과한 경우에도 작업을 계속 진행하고 함수를 성공적으로 실행할 수 있 습니다.

이 샘플 Lambda 함수를 사용하려면이 샘플 정책에 표시된 대로 Lambda 실행 역할에 대한 정책에 AWS CloudFormation Amazon S3 및 CodePipeline에 대한 Allow 권한이 있어야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "logs:*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:logs:*:*:*"
        },
        {
            "Action": [
                "codepipeline:PutJobSuccessResult",
                 "codepipeline:PutJobFailureResult"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
```

```
"Action": [
                 "cloudformation:DescribeStacks",
                "cloudformation:CreateStack",
                "cloudformation:UpdateStack"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "s3:*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

AWS CloudFormation 템플릿을 생성하려면 일반 텍스트 편집기를 열고 다음 코드를 복사하여 붙여 넣 습니다.

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Description" : "CloudFormation template which creates an S3 bucket",
  "Resources" : {
    "MySampleBucket" : {
      "Type" : "AWS::S3::Bucket",
      "Properties" : {
      }
    }
  },
  "Outputs" : {
    "BucketName" : {
      "Value" : { "Ref" : "MySampleBucket" },
      "Description" : "The name of the S3 bucket"
    }
  }
}
```

이름이 template.json인 JSON 파일로 지정한 다음 이를 template-package 디렉터리에 저장합 니다. 이 디렉터리의 압축된(.zip) 파일과 template-package.zip이라는 파일을 생성하고 압축된 파 일을 버전이 지정된 Amazon S3 버킷에 업로드합니다. 파이프라인에 대해 구성된 버킷이 이미 있는 경 우 해당 버킷을 사용할 수 있습니다. 그런 다음 파이프라인을 편집하여 .zip 파일을 검색하는 소스 작업 을 추가합니다. 이 작업 *MyTemp1ate*의 출력 이름을 지정합니다. 자세한 내용은 <u>CodePipeline에서 파</u> 이프라인 편집 단원을 참조하십시오.

Note

샘플 Lambda 함수는 이러한 파일 이름과 압축된 구조가 필요합니다. 그러나이 샘플로 자 체 AWS CloudFormation 템플릿을 대체할 수 있습니다. 자체 템플릿을 사용하는 경우 AWS CloudFormation 템플릿에 필요한 추가 기능을 허용하도록 Lambda 실행 역할에 대한 정책을 수정해야 합니다.

다음 코드를 함수로 Lambda에 추가하려면

- 1. Lambda 콘솔을 열고 함수 생성을 선택합니다.
- 함수 생성 페이지에서 처음부터 새로 작성을 선택합니다. 함수 이름에 Lambda 함수의 이름을 입 력합니다.
- 3. 런타임에서 Python 2.7을 선택합니다.
- 실행 역할 선택 또는 생성에서 기존 역할 사용을 선택합니다. 기존 역할에서 역할을 선택한 다음 함수 생성을 선택합니다.

생성된 기능에 대한 세부 정보 페이지가 열립니다.

5. 함수 코드 상자에 다음 코드를 붙여 넣습니다.

```
from __future__ import print_function
from boto3.session import Session
import json
import urllib
import boto3
import zipfile
import tempfile
import traceback
print('Loading function')
cf = boto3.client('cloudformation')
code_pipeline = boto3.client('codepipeline')
```

```
def find_artifact(artifacts, name):
    """Finds the artifact 'name' among the 'artifacts'
   Args:
        artifacts: The list of artifacts available to the function
        name: The artifact we wish to use
    Returns:
        The artifact dictionary found
    Raises:
        Exception: If no matching artifact is found
    .....
   for artifact in artifacts:
        if artifact['name'] == name:
            return artifact
   raise Exception('Input artifact named "{0}" not found in event'.format(name))
def get_template(s3, artifact, file_in_zip):
    """Gets the template artifact
    Downloads the artifact from the S3 artifact store to a temporary file
    then extracts the zip and returns the file containing the CloudFormation
    template.
   Args:
        artifact: The artifact to download
        file_in_zip: The path to the file within the zip containing the template
    Returns:
        The CloudFormation template as a string
    Raises:
        Exception: Any exception thrown while downloading the artifact or unzipping
it
    .....
   tmp_file = tempfile.NamedTemporaryFile()
    bucket = artifact['location']['s3Location']['bucketName']
    key = artifact['location']['s3Location']['objectKey']
   with tempfile.NamedTemporaryFile() as tmp_file:
        s3.download_file(bucket, key, tmp_file.name)
        with zipfile.ZipFile(tmp_file.name, 'r') as zip:
```

```
return zip.read(file_in_zip)
def update_stack(stack, template):
    """Start a CloudFormation stack update
   Args:
        stack: The stack to update
        template: The template to apply
    Returns:
       True if an update was started, false if there were no changes
        to the template since the last update.
    Raises:
        Exception: Any exception besides "No updates are to be performed."
    .....
    try:
        cf.update_stack(StackName=stack, TemplateBody=template)
        return True
    except botocore.exceptions.ClientError as e:
        if e.response['Error']['Message'] == 'No updates are to be performed.':
            return False
        else:
            raise Exception('Error updating CloudFormation stack
 "{0}"'.format(stack), e)
def stack_exists(stack):
    """Check if a stack exists or not
   Args:
        stack: The stack to check
    Returns:
       True or False depending on whether the stack exists
    Raises:
       Any exceptions raised .describe_stacks() besides that
        the stack doesn't exist.
    .....
    try:
        cf.describe_stacks(StackName=stack)
```

```
return True
    except botocore.exceptions.ClientError as e:
        if "does not exist" in e.response['Error']['Message']:
            return False
        else:
            raise e
def create_stack(stack, template):
    """Starts a new CloudFormation stack creation
   Args:
        stack: The stack to be created
        template: The template for the stack to be created with
   Throws:
        Exception: Any exception thrown by .create_stack()
    .....
    cf.create_stack(StackName=stack, TemplateBody=template)
def get_stack_status(stack):
    """Get the status of an existing CloudFormation stack
   Args:
        stack: The name of the stack to check
    Returns:
        The CloudFormation status string of the stack such as CREATE_COMPLETE
    Raises:
        Exception: Any exception thrown by .describe_stacks()
    .....
    stack_description = cf.describe_stacks(StackName=stack)
    return stack_description['Stacks'][0]['StackStatus']
def put_job_success(job, message):
    """Notify CodePipeline of a successful job
   Args:
        job: The CodePipeline job ID
        message: A message to be logged relating to the job status
    Raises:
        Exception: Any exception thrown by .put_job_success_result()
```

```
.....
    print('Putting job success')
    print(message)
    code_pipeline.put_job_success_result(jobId=job)
def put_job_failure(job, message):
    """Notify CodePipeline of a failed job
   Args:
        job: The CodePipeline job ID
       message: A message to be logged relating to the job status
    Raises:
        Exception: Any exception thrown by .put_job_failure_result()
    .....
   print('Putting job failure')
    print(message)
    code_pipeline.put_job_failure_result(jobId=job, failureDetails={'message':
message, 'type': 'JobFailed'})
def continue_job_later(job, message):
    """Notify CodePipeline of a continuing job
   This will cause CodePipeline to invoke the function again with the
    supplied continuation token.
   Args:
        job: The JobID
       message: A message to be logged relating to the job status
        continuation_token: The continuation token
    Raises:
        Exception: Any exception thrown by .put_job_success_result()
    .....
    # Use the continuation token to keep track of any job execution state
    # This data will be available when a new job is scheduled to continue the
 current execution
    continuation_token = json.dumps({'previous_job_id': job})
   print('Putting job continuation')
```

```
print(message)
    code_pipeline.put_job_success_result(jobId=job,
 continuationToken=continuation_token)
def start_update_or_create(job_id, stack, template):
    """Starts the stack update or create process
   If the stack exists then update, otherwise create.
   Args:
        job_id: The ID of the CodePipeline job
        stack: The stack to create or update
        template: The template to create/update the stack with
    .....
   if stack_exists(stack):
        status = get_stack_status(stack)
        if status not in ['CREATE_COMPLETE', 'ROLLBACK_COMPLETE',
 'UPDATE_COMPLETE']:
            # If the CloudFormation stack is not in a state where
            # it can be updated again then fail the job right away.
            put_job_failure(job_id, 'Stack cannot be updated when status is: ' +
 status)
            return
       were_updates = update_stack(stack, template)
        if were_updates:
            # If there were updates then continue the job so it can monitor
            # the progress of the update.
            continue_job_later(job_id, 'Stack update started')
        else:
            # If there were no updates then succeed the job immediately
            put_job_success(job_id, 'There were no stack updates')
    else:
        # If the stack doesn't already exist then create it instead
        # of updating it.
        create_stack(stack, template)
        # Continue the job so the pipeline will wait for the CloudFormation
        # stack to be created.
        continue_job_later(job_id, 'Stack create started')
def check_stack_update_status(job_id, stack):
```

```
"""Monitor an already-running CloudFormation update/create
    Succeeds, fails or continues the job depending on the stack status.
   Args:
        job_id: The CodePipeline job ID
        stack: The stack to monitor
    .....
    status = get_stack_status(stack)
    if status in ['UPDATE_COMPLETE', 'CREATE_COMPLETE']:
        # If the update/create finished successfully then
        # succeed the job and don't continue.
        put_job_success(job_id, 'Stack update complete')
    elif status in ['UPDATE_IN_PROGRESS', 'UPDATE_ROLLBACK_IN_PROGRESS',
    'UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS', 'CREATE_IN_PROGRESS',
    'ROLLBACK_IN_PROGRESS', 'UPDATE_COMPLETE_CLEANUP_IN_PROGRESS']:
        # If the job isn't finished yet then continue it
        continue_job_later(job_id, 'Stack update still in progress')
    else:
        # If the Stack is a state which isn't "in progress" or "complete"
        # then the stack update/create has failed so end the job with
        # a failed result.
        put_job_failure(job_id, 'Update failed: ' + status)
def get_user_params(job_data):
    """Decodes the JSON user parameters and validates the required properties.
   Args:
        job_data: The job data structure containing the UserParameters string which
 should be a valid JSON structure
    Returns:
        The JSON parameters decoded as a dictionary.
    Raises:
        Exception: The JSON can't be decoded or a property is missing.
    .....
   try:
       # Get the user parameters which contain the stack, artifact and file
 settings
```
```
user_parameters = job_data['actionConfiguration']['configuration']
['UserParameters']
        decoded_parameters = json.loads(user_parameters)
   except Exception as e:
        # We're expecting the user parameters to be encoded as JSON
        # so we can pass multiple values. If the JSON can't be decoded
        # then fail the job with a helpful message.
        raise Exception('UserParameters could not be decoded as JSON')
   if 'stack' not in decoded_parameters:
        # Validate that the stack is provided, otherwise fail the job
        # with a helpful message.
        raise Exception('Your UserParameters JSON must include the stack name')
   if 'artifact' not in decoded_parameters:
        # Validate that the artifact name is provided, otherwise fail the job
        # with a helpful message.
        raise Exception('Your UserParameters JSON must include the artifact name')
    if 'file' not in decoded_parameters:
        # Validate that the template file is provided, otherwise fail the job
        # with a helpful message.
       raise Exception('Your UserParameters JSON must include the template file
 name')
   return decoded_parameters
def setup_s3_client(job_data):
    """Creates an S3 client
   Uses the credentials passed in the event by CodePipeline. These
    credentials can be used to access the artifact bucket.
   Args:
        job_data: The job data structure
    Returns:
       An S3 client with the appropriate credentials
    .....
    key_id = job_data['artifactCredentials']['accessKeyId']
    key_secret = job_data['artifactCredentials']['secretAccessKey']
    session_token = job_data['artifactCredentials']['sessionToken']
```

```
session = Session(aws_access_key_id=key_id,
        aws_secret_access_key=key_secret,
        aws_session_token=session_token)
   return session.client('s3',
config=botocore.client.Config(signature_version='s3v4'))
def lambda_handler(event, context):
    """The Lambda function handler
   If a continuing job then checks the CloudFormation stack status
    and updates the job accordingly.
   If a new job then kick of an update or creation of the target
    CloudFormation stack.
   Args:
        event: The event passed by Lambda
        context: The context passed by Lambda
    .....
    try:
        # Extract the Job ID
        job_id = event['CodePipeline.job']['id']
        # Extract the Job Data
        job_data = event['CodePipeline.job']['data']
        # Extract the params
        params = get_user_params(job_data)
        # Get the list of artifacts passed to the function
        artifacts = job_data['inputArtifacts']
        stack = params['stack']
        artifact = params['artifact']
        template_file = params['file']
        if 'continuationToken' in job_data:
            # If we're continuing then the create/update has already been triggered
            # we just need to check if it has finished.
            check_stack_update_status(job_id, stack)
        else:
            # Get the artifact details
```

```
artifact_data = find_artifact(artifacts, artifact)
        # Get S3 client to access artifact with
        s3 = setup_s3_client(job_data)
        # Get the JSON template file out of the artifact
        template = get_template(s3, artifact_data, template_file)
        # Kick off a stack update or create
        start_update_or_create(job_id, stack, template)
except Exception as e:
   # If any other exceptions which we didn't expect are raised
   # then fail the job and log the exception message.
   print('Function failed due to exception.')
   print(e)
   traceback.print_exc()
   put_job_failure(job_id, 'Function exception: ' + str(e))
print('Function complete.')
return "Complete."
```

- 6. 핸들러는 기본값으로 두고, 역할은 이전에 선택하거나 생성한 이름, CodePipelineLambdaExecRole 그대로 둡니다.
- 7. Basic settings(기본 설정)의 제한 시간에서 기본값 3초를 20으로 바꿉니다.
- 8. 저장(Save)을 선택합니다.
- CodePipeline 콘솔에서, 파이프라인을 편집하여 함수를 작업으로 파이프라인의 한 단계에 추가합 니다. 변경하려는 파이프라인 단계에서 편집을 선택하고 작업 그룹 추가를 선택합니다. 작업 편집 페이지의 작업 이름에 작업 이름을 입력합니다. 작업 공급자에서 Lambda를 선택합니다.

입력 아티팩트에서 MyTemplate을 선택합니다. UserParameters에서는 JSON 문자열에 다음과 같은 3개의 파라미터를 제공해야 합니다.

- 스택 이름
- AWS CloudFormation 템플릿 이름 및 파일 경로
- 입력 아티팩트

중괄호({ })를 사용하고 쉼표로 파라미터를 구분합니다. 예를 들어 *MyTestStack*이라는 스택 을 생성하려면, 입력 아티팩트 *MyTemplate*이 있는 파이프라인에 대해 UserParameters에서 {"stack":"*MyTestStack*","file":"template-package/template.json","artifact":"*MyTemplate*"}을 입력 합니다.

Note

UserParameters에 입력 아티팩트를 지정했더라도 입력 아티팩트의 작업에 대해서도 이 입력 아티팩트를 지정해야 합니다.

10. 변경 사항을 파이프라인에 저장한 다음 변경 사항을 수동으로 릴리스하여 작업 및 Lambda 함수를 테스트합니다.

스테이지에 수동 승인 작업 추가

에서는 필요한 AWS Identity and Access Management 권한이 있는 사용자가 작업을 승인하거나 거부 할 AWS CodePipeline수 있도록 파이프라인 실행을 중지하려는 시점에 파이프라인의 단계에 승인 작 업을 추가할 수 있습니다.

작업이 승인된 경우 파이프라인 실행이 재개됩니다. 작업이 거부되거나 파이프라인이 작업에 도달하 여 중지된 지 7일 이내에 아무도 작업을 승인하거나 거부하지 않는 경우 결과는 실패하는 작업과 동일 하며 파이프라인 실행이 계속 진행되지 않습니다.

다음과 같은 목적으로 수동 승인을 사용할 수 있습니다.

- 파이프라인의 다음 단계에 수정이 허용되기 전에 다른 사용자가 코드 검토를 수행하거나 관리 검토 를 변경하도록 합니다.
- 릴리스되기 전에 다른 사용자가 최신 버전의 애플리케이션에 대한 수동 품질 보증 테스트를 수행하 거나 빌드 아티팩트의 무결성을 확인하도록 합니다.
- 회사 웹 사이트에 게시되기 전에 다른 사용자가 신규 또는 업데이트된 텍스트를 검토하도록 합니다.

CodePipeline에서 수동 승인 작업에 대한 구성 옵션

CodePipeline은 승인자에게 승인 작업에 대해 알리는 데 사용할 수 있는 세 가지 구성 옵션을 제공합니 다.

승인 알림 게시 파이프라인이 작업에서 중지될 때 Amazon Simple Notification Service 주제에 메시지 를 게시하도록 승인 작업을 구성할 수 있습니다. Amazon SNS는 주제를 구독한 모든 엔드포인트에 메 시지를 전달합니다. 승인 작업을 포함할 파이프라인과 동일한 AWS 리전에서 생성된 주제를 사용해야 합니다. 주제를 생성하면 목적을 식별하는 이름을 MyFirstPipeline-us-east-2-approval 등의 형식으로 지정하는 것이 좋습니다. 승인 알림을 Amazon SNS 주제에 게시하면 이메일 또는 SMS 수신자, SQS 대기열, HTTP/HTTPS 엔 드포인트나 Amazon SNS를 사용하여 호출한 AWS Lambda 함수 등의 형식 중에서 선택할 수 있습니 다. Amazon SNS 주제 알림에 대한 자세한 내용은 다음 주제를 참조하세요.

- Amazon Simple Notification Service란 무엇인가요?
- <u>Amazon SNS 주제 생성</u>
- Amazon SQS 대기열로 Amazon SNS 메시지 전송
- 대기열에서 Amazon SNS 주제 구독
- HTTP/HTTPS 엔드포인트로 Amazon SNS 메시지 전송
- Amazon SNS 알림을 사용하여 Lambda 함수 호출

승인 작업 알림에 대해 생성된 JSON 데이터의 구조는 <u>CodePipeline에서 수동 승인 알림에 대한 JSON</u> <u>데이터 형식</u> 단원을 참조하십시오.

검토용 URL 지정 승인 작업 구성의 일부로 검토할 URL을 지정할 수 있습니다. URL은 승인자가 테스 트할 웹 애플리케이션의 링크이거나 승인 요청에 대한 더 많은 정보가 포함된 페이지일 수 있습니다. URL은 Amazon SNS 주제에 게시된 알림에 포함되어 있습니다. 승인자는 콘솔이나 CLI를 사용하여 볼 수 있습니다.

승인자를 위한 설명 입력 승인 작업을 생성하면 알림을 수신하는 승인자나 콘솔 또는 CLI 응답에서 작 업을 보는 승인자에게 표시되는 설명을 추가할 수도 있습니다.

구성 옵션 없음 이러한 세 가지 옵션을 구성하지 않도록 선택할 수도 있습니다. 예를 들어 작업을 검토 할 준비가 된 다른 사용자에게 직접 알리거나 작업을 직접 승인할 때까지 파이프라인이 중지될 경우 이 옵션이 필요하지 않을 수 있습니다.

CodePipeline에서 승인 작업의 설정 및 워크플로우 개요

다음은 수동 승인 설정 및 사용에 대한 개요입니다.

- 1. 승인 작업을 승인하거나 거부하는 데 필요한 IAM 권한을 조직 내 하나 이상의 IAM 역할에게 부여합 니다.
- 2. (선택 사항) Amazon SNS 알림을 사용하는 경우 CodePipeline 작업에서 사용하는 서비스 역할이 Amazon SNS 리소스에 액세스할 수 있는지 확인합니다.
- 3. (선택 사항) Amazon SNS 알림을 사용하는 경우 Amazon SNS 주제를 생성하고 하나 이상의 구독자 또는 엔트포인트를 추가합니다.

4. AWS CLI를 사용하여 파이프라인을 생성하거나 CLI 또는 콘솔을 사용하여 파이프라인을 생성한 후 파이프라인의 단계에 승인 작업을 추가합니다.

알림을 사용하는 경우 작업의 구성에 Amazon SNS 주제의 Amazon 리소스 이름(ARN)을 포 함합니다. (ARN은 Amazon 리소스의 고유 식별자입니다. Amazon SNS 주제에 대한 ARNs은 arn:aws:sns:us-east-2:80398EXAMPLE:MyApprovalTopic과 같이 구조화됩니다. 자세한 내용은의 <u>Amazon 리소스 이름(ARNs) 및 AWS 서비스 네임스페이스</u>를 참조하세요Amazon Web Services 일반 참조.)

- 5. 파이프라인은 승인 작업에 도달할 때 중지됩니다. Amazon SNS 주제 ARN이 작업의 구성에 포함된 경우 알림이 Amazon SNS 주제에 게시되고 콘솔에서 승인 작업을 검토하기 위한 링크가 포함된 메 시지가 주제 또는 구독된 엔드포인트의 구독자에게 전달됩니다.
- 6. 승인자는 대상 URL을 검사하고 설명을 검토합니다(있는 경우).
- 7. 콘솔, CLI 또는 SDK를 사용하여 승인자는 요약 설명을 제공하고 응답을 제출합니다.
 - 승인됨: 파이프라인 실행이 재개됩니다.
 - 거부됨: 단계 상태가 "실패"로 변경되고 파이프라인 실행이 재개되지 않습니다.

응답이 7일 이내에 제출되지 않은 경우 작업이 "실패"로 표시됩니다.

CodePipeline에서 IAM 사용자에게 승인 권한 부여

조직 내 IAM 사용자가 승인 작업을 승인하거나 거부하기 전에 이들에게 파이프라

인에 액세스하고 승인 작업의 상태를 업데이트할 수 있는 권한을 부여해야 합니다.

AWSCodePipelineApproverAccess 관리형 정책을 IAM 사용자, 역할 또는 그룹에 연결하여 계정에 서 모든 파이프라인과 승인 작업에 액세스할 수 있는 권한을 부여하거나 IAM 사용자, 역할 또는 그룹에 서 액세스할 수 있는 개별 리소스를 지정하여 제한된 권한을 부여할 수 있습니다.

Note

이 주제에서 설명된 권한은 매우 제한된 액세스를 허용합니다. 사용자, 역할 또는 그룹이 승인 작업의 승인이나 거부 이외의 작업을 수행할 수 있도록 하려면 다른 관리형 정책을 연결할 수 있습니다. CodePipeline에서 사용 가능한 관리형 정책에 대한 자세한 내용은 <u>AWS 에 대한 관</u> 리형 정책 AWS CodePipeline 단원을 참조하세요.

모든 파이프라인 및 승인 작업에 대한 승인 권한 부여

CodePipeline에서 승인 작업을 수행해야 하는 사용자의 경우 AWSCodePipelineApproverAccess 관리형 정책을 사용하세요.

액세스 권한을 제공하려면 사용자, 그룹 또는 역할에 권한을 추가하세요:

• 의 사용자 및 그룹 AWS IAM Identity Center:

권한 세트를 생성합니다. AWS IAM Identity Center 사용 설명서의 <u>권한 세트 생성</u>의 지침을 따릅니다.

• 보안 인증 공급자를 통해 IAM에서 관리되는 사용자:

ID 페더레이션을 위한 역할을 생성합니다. IAM 사용 설명서의 <u>Create a role for a third-party identity</u> provider (federation)의 지침을 따릅니다.

- IAM 사용자:
 - 사용자가 맡을 수 있는 역할을 생성합니다. IAM 사용 설명서에서 <u>Create a role for an IAM user</u>의 지침을 따릅니다.
 - (권장되지 않음)정책을 사용자에게 직접 연결하거나 사용자를 사용자 그룹에 추가합니다. IAM 사용 설명서에서 사용자(콘솔)에 권한 추가의 지침을 따르세요.

특정 파이프라인 및 승인 작업에 대해 승인 권한 지정

CodePipeline에서 승인 작업을 수행해야 하는 사용자의 경우 다음과 같은 사용자 지정 정책을 사용하 세요. 아래 정책에서 사용자가 액세스할 수 있는 개별 리소스를 지정하세요. 예를 들어, 다음 정책은 미 국 동부(오하이오) 리전(us-east-2)의 MyFirstPipeline 파이프라인에 있는 MyApprovalAction이 라는 작업만 승인하거나 거부할 수 있는 권한을 사용자에게 부여합니다.

Note

IAM 사용자가 CodePipeline 대시보드에 액세스하여 파이프라인 목록을 확인해야 하는 경우 에만 codepipeline:ListPipelines 권한이 필요합니다. 콘솔 액세스가 필요하지 않으면 codepipeline:ListPipelines를 생략해도 됩니다. 1. 에 로그인 AWS Management Console 하고 <u>https://console.aws.amazon.com/iam/</u>://https://https:// https://https://://https://://https://://https://://https://://https://://

2. 왼쪽의 탐색 창에서 정책을 선택합니다.

정책을 처음으로 선택하는 경우 관리형 정책 소개 페이지가 나타납니다. 시작을 선택합니다.

- 3. 페이지 상단에서 정책 생성을 선택합니다.
- 4. 정책 편집기 섹션에서 JSON 옵션을 선택합니다.
- 5. 다음 JSON 정책 문서를 입력합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:ListPipelines"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:GetPipeline",
                "codepipeline:GetPipelineState",
                "codepipeline:GetPipelineExecution"
            ],
            "Resource": "arn:aws:codepipeline:us-
east-2:80398EXAMPLE:MyFirstPipeline"
        },
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:PutApprovalResult"
            ],
            "Resource": "arn:aws:codepipeline:us-
east-2:80398EXAMPLE:MyFirstPipeline/MyApprovalStage/MyApprovalAction"
        }
```

}

6. Next(다음)를 선택합니다.

Note

]

언제든지 시각적 편집기 옵션과 JSON 편집기 옵션 간에 전환할 수 있습니다. 그러나 변경 을 적용하거나 시각적 편집기에서 다음을 선택한 경우 IAM은 시각적 편집기에 최적화되 도록 정책을 재구성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 <u>정책 재구성</u>을 참조 하세요.

- 검토 및 생성 페이지에서 생성하는 정책에 대한 정책 이름과 설명(선택 사항)을 입력합니다. 이 정 책에 정의된 권한을 검토하여 정책이 부여한 권한을 확인합니다.
- 8. 정책 생성을 선택하고 새로운 정책을 저장합니다.

CodePipeline 서비스 역할에 Amazon SNS 권한 부여

승인 작업에 검토가 필요할 때 Amazon SNS를 사용하여 알림을 주제에 게시할 경우 CodePipeline 작 업에서 사용한 서비스 역할에 Amazon SNS 리소스에 액세스할 수 있는 권한을 부여해야 합니다. IAM 콘솔을 사용하여 이 권한을 서비스 역할에 추가할 수 있습니다.

아래 정책에서 SNS를 통한 게시 정책을 지정하세요. 다음 정책의 경우 SNSPublish로 이름을 지정할 수 있습니다. 다음 정책을 서비스 역할에 연결하여 사용하세요.

🛕 Important

에서 사용한 AWS Management Console 것과 동일한 계정 정보로에 로그인했는지 확인합니 다CodePipeline 시작하기.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sns:Publish",
            "Resource": "*"
        }
```

]

}

JSON 정책 편집기를 사용하여 정책을 생성하려면

- 1. 에 로그인 AWS Management Console 하고 <u>https://console.aws.amazon.com/iam/</u>://https://https:// https://https://://https://://https://://https://://https://://
- 2. 왼쪽의 탐색 창에서 정책을 선택합니다.

정책을 처음으로 선택하는 경우 관리형 정책 소개 페이지가 나타납니다. 시작을 선택합니다.

- 3. 페이지 상단에서 정책 생성을 선택합니다.
- 4. 정책 편집기 섹션에서 JSON 옵션을 선택합니다.
- 5. JSON 정책 문서를 입력하거나 붙여 넣습니다. IAM 정책 언어에 대한 자세한 내용은 <u>IAM JSON</u> 정책 참조를 참조하세요.
- 6. 정책 검증 중에 생성되는 모든 보안 경고, 오류 또는 일반 경고를 해결하고 다음을 선택합니다.

Note

언제든지 시각적 편집기 옵션과 JSON 편집기 옵션을 서로 전환할 수 있습니다. 그러나 변 경을 적용하거나 시각적 편집기에서 다음을 선택한 경우 IAM은 시각적 편집기에 최적화 되도록 정책을 재구성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 <u>정책 재구성</u>을 참 조하세요.

7. (선택 사항)에서 정책을 생성하거나 편집할 때 템플릿에 사용할 AWS Management Console수 있는 JSON 또는 YAML 정책 AWS CloudFormation 템플릿을 생성할 수 있습니다.

이렇게 하려면 정책 편집기에서 작업을 선택한 다음, CloudFormation 템플릿 생성을 선택합니 다. 자세한 AWS CloudFormation내용은 AWS CloudFormation 사용 설명서의 <u>AWS Identity and</u> <u>Access Management 리소스 유형 참조</u>를 참조하세요.

- 8. 정책에 권한 추가를 완료했으면 다음을 선택합니다.
- 검토 및 생성 페이지에서 생성하는 정책의 정책 이름과 설명(선택 사항)을 입력합니다. 이 정책에 정의된 권한을 검토하여 정책이 부여한 권한을 확인합니다.
- 10. (선택 사항) 태그를 키 값 페어로 연결하여 메타데이터를 정책에 추가합니다. IAM에서 태그를 사용하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 <u>AWS Identity and Access Management 리</u>소스에 대한 태그를 참조하세요.
- 11. 정책 생성을 선택하고 새로운 정책을 저장합니다.

CodePipeline의 파이프라인에 수동 승인 작업 추가

다른 사용자가 작업을 수동으로 승인하거나 거부할 수 있도록 파이프라인을 중지할 지점에서 CodePipeline 파이프라인의 단계에 승인 작업을 추가할 수 있습니다.

Note

승인 작업은 소스 단계에 추가할 수 없습니다. 소스 단계는 소스 작업만 포함할 수 있습니다.

승인 작업을 검토할 준비가 되었을 때 Amazon SNS를 사용하여 알림을 전송할 경우 먼저 다음 사전 조 건을 완료해야 합니다.

- CodePipeline 서비스 역할에 Amazon SNS 리소스에 액세스할 수 있는 권한을 부여합니다. 자세한 내용은 CodePipeline 서비스 역할에 Amazon SNS 권한 부여을 참조하세요.
- 조직 내 하나 이상의 IAM 자격 증명에 승인 작업의 상태를 업데이트할 수 있는 권한을 부여합니다. 자세한 내용은 CodePipeline에서 IAM 사용자에게 승인 권한 부여을 참조하세요.

이 예시에서는 새 승인 단계를 생성하고 단계에 수동 승인 작업을 추가합니다. 다른 작업이 포함된 기 존 단계에 수동 승인 작업을 추가할 수도 있습니다.

CodePipeline 파이프라인에 수동 승인 작업 추가(콘솔)

CodePipeline 콘솔을 사용하여 승인 작업을 기존 CodePipeline 파이프라인에 추가할 수 있습니다. 새 파이프라인을 생성할 때 승인 작업을 추가하려면 AWS CLI를 사용해야 합니다.

- 1. https://console.aws.amazon.com/codepipeline/에서 CodePipeline 콘솔을 엽니다.
- 2. 이름에서 파이프라인을 선택합니다.
- 3. 파이프라인 세부 정보 페이지에서 [Edit]를 선택합니다.
- 새로운 단계에 승인 작업을 추가하려면 승인 요청을 추가하려는 파이프라인의 지점에서 +Add stage(단계 추가)를 선택하고 단계의 이름을 입력합니다. Add stage(단계 추가) 페이지의 Stage name(단계 이름)에 새 단계 이름을 입력합니다. 예를 들어 새 단계를 추가하고 이름을 Manual_Approval로 지정합니다.

기존 단계에 승인 작업을 추가하려면 Edit stage(단계 편집)를 선택합니다.

- 5. 승인 작업을 추가하려는 단계에서 + Add action group(작업 그룹 추가)을 선택합니다.
- 6. Edit action(작업 편집) 페이지에서 다음을 수행합니다.

- 1. Action name(작업 이름)에 작업을 식별할 이름을 입력합니다.
- 2. 작업 공급자의 승인에서 수동 승인을 선택합니다.
- 3. (선택 사항) SNS topic ARN(SNS 주제 ARN)에서 승인 작업에 대한 알림을 보낼 때 사용할 주제 이름을 선택합니다.
- (선택 사항) URL for review(검토할 URL)에서 승인자의 검사를 받으려는 애플리케이션이나 페 이지의 URL을 입력합니다. 승인자는 파이프라인의 콘솔 보기에 포함되어 있는 링크를 통해 이 URL에 액세스할 수 있습니다.
- 5. (선택 사항) Comments(설명)에 검토자와 공유하려는 그 밖의 모든 정보를 입력합니다.
- 6. 저장(Save)을 선택합니다.

CodePipeline 파이프라인에 수동 승인 작업 추가(CLI)

파이프라인을 생성하거나 승인 작업을 기존 파이프라인에 추가하는 데 CLI를 사용할 수 있습니다. 이 렇게 하려면 생성하거나 편집하고 있는 단계에서 수동 승인 유형과 함께 승인 작업을 포함합니다.

파이프라인 생성 및 편집에 대한 자세한 내용은 <u>파이프라인 스테이지 및 작업 생성</u> 및 <u>CodePipeline에</u> 서 파이프라인 편집 단원을 참조하십시오.

승인 작업만 포함하는 파이프라인에 단계를 추가하려면 파이프라인을 생성하거나 업데이트할 때 다음 예와 비슷한 항목을 포함합니다.

Note

configuration 섹션은 선택 사항입니다. 이것은 파일의 전체 구조가 아닌 일부입니다. 자세 한 내용은 CodePipeline 파이프라인 구조 참조 단원을 참조하십시오.

```
},
    "inputArtifacts": [],
    "outputArtifacts": [],
    "configuration": {
        "NotificationArn": "arn:aws:sns:us-
east-2:80398EXAMPLE:MyApprovalTopic",
        "ExternalEntityLink": "http://example.com",
        "CustomData": "The latest changes include feedback from Bob."},
        "runOrder": 1
     }
   ]
}
```

승인 작업이 다른 작업과 함께 단계에 있는 경우 단계를 포함한 JSON 파일의 섹션은 다음 예와 비슷할 수 있습니다.

Note

configuration 섹션은 선택 사항입니다. 이것은 파일의 전체 구조가 아닌 일부입니다. 자세 한 내용은 <u>CodePipeline 파이프라인 구조 참조</u> 단원을 참조하십시오.

```
,
{
    "name": "Production",
    "actions": [
        {
            "inputArtifacts": [],
            "name": "MyApprovalAction",
            "actionTypeId": {
                "category": "Approval",
                "owner": "AWS",
                "version": "1",
                "provider": "Manual"
            },
            "outputArtifacts": [],
            "configuration": {
                "NotificationArn": "arn:aws:sns:us-
east-2:80398EXAMPLE:MyApprovalTopic",
                "ExternalEntityLink": "http://example.com",
                "CustomData": "The latest changes include feedback from Bob."
            },
```

```
"runOrder": 1
        },
        {
            "inputArtifacts": [
                {
                     "name": "MyApp"
                }
            ],
            "name": "MyDeploymentAction",
            "actionTypeId": {
                 "category": "Deploy",
                "owner": "AWS",
                "version": "1",
                "provider": "CodeDeploy"
            },
            "outputArtifacts": [],
            "configuration": {
                 "ApplicationName": "MyDemoApplication",
                "DeploymentGroupName": "MyProductionFleet"
            },
            "runOrder": 2
        }
    ]
}
```

CodePipeline에서 승인 작업 승인 또는 거부

파이프라인에 승인 작업이 포함되어 있으면 작업이 추가된 지점에서 파이프라인 실행이 중지됩니다. 다른 사용자가 작업을 수동으로 승인하지 않는 한 파이프라인 실행이 재개되지 않습니다. 승인자가 작 업을 거부하거나 승인 작업에 대해 파이프라인이 중지된 지 7일 이내에 승인 응답을 수신하지 않은 경 우 파이프라인 상태는 "실패"가 됩니다.

승인 작업을 파이프라인에 추가한 사람이 알림을 구성한 경우 파이프라인 정보와 승인 상태가 포함된 이메일을 받을 수 있습니다.

승인 작업 승인 또는 거부(콘솔)

승인 작업에 대한 직접 링크를 포함한 알림을 수신하는 경우 Approve or reject(승인 또부 거부) 링크를 선택하고 필요한 경우 콘솔에 로그인한 후 다음 7단계를 계속 진행합니다. 그렇지 않은 경우 다음의 모 든 단계를 따르십시오.

1. https://console.aws.amazon.com/codepipeline/에서 CodePipeline 콘솔을 엽니다.

- 2. [All Pipelines] 페이지에서 파이프라인의 이름을 선택합니다.
- 3. 승인 작업이 있는 단계를 찾습니다. 검토를 선택합니다.

검토 대화 상자가 표시됩니다. 세부 정보 탭에는 리뷰 내용과 의견이 표시됩니다.

Details Revisions		
Trigger		
StartPipelineExecution - assumed-role/	2	
Comments about this action		
Comments for reviewer/approver		
URL for review		
https://review-url 🔼		
Decision		
 Approve Approving will resume the pipeline execution. 		
 Reject Rejecting will stop the pipeline execution with a f 	ailed status.	
Comments - optional	Preview markdow	vn Learn more 🗹
Comments from reviewer/approver		
		,
	Consol	Cultural

개정 탭에는 실행에 대한 소스 개정이 표시됩니다.

Review Action name: Approval Status: Waiting for approval		
Details	Revisions	
Source Source		
Revision 2de8579a	: Add files via upload	
	Cancel	Submit

- 4. 세부 정보 탭에서 의견과 URL(있는 경우)을 확인하십시오. 메시지에도 사용자가 검토할 수 있도록 콘텐츠의 URL이 표시됩니다(포함되어 있는 경우).
- 5. URL이 표시되는 경우, 작업에서 리뷰용 URL 링크를 선택하여 대상 웹 페이지를 연 다음, 콘텐츠 를 검토합니다.
- 검토 창에 작업을 승인 또는 거부하는 이유와 같은 검토 의견을 입력한 다음, 승인 또는 거부 버튼 을 선택합니다.
- 7. 제출을 선택합니다.

승인 요청 승인 또는 거부(CLI)

CLI를 사용하여 승인 작업에 응답하려면 먼저 get-pipeline-state명령을 사용하여 승인 작업의 최신 실행과 연결된 토큰을 검색해야 합니다.

1. 터미널(Linux, macOS 또는 Unix) 또는 명령 프롬프트(Windows)에서, 승인 작업이 들어 있는 파이 프라인에 <u>get-pipeline-state</u> 명령을 실행합니다. 예를 들어, *MyFirstPipeline*이라는 파이프라 인에서 다음을 입력합니다.

aws codepipeline get-pipeline-state --name *MyFirstPipeline*

2. 명령에 대한 응답으로 token 값의 위치로 이동하는데, 이 값은 다음과 같이 승인 작업의 actionStates 단원의 latestExecution에 있습니다.

```
{
    "created": 1467929497.204,
    "pipelineName": "MyFirstPipeline",
    "pipelineVersion": 1,
    "stageStates": [
        {
            "actionStates": [
                {
                    "actionName": "MyApprovalAction",
                    "currentRevision": {
                        "created": 1467929497.204,
                        "revisionChangeId": "CEM7d6Tp7zfelUSLCPPwo234xEXAMPLE",
                        "revisionId": "HYGp7zmwbCPPwo23xCMdTeqIlEXAMPLE"
                    },
                    "latestExecution": {
                        "lastUpdatedBy": "identity",
                        "summary": "The new design needs to be reviewed before
 release.",
                        "token": "1a2b3c4d-573f-4ea7-a67E-XAMPLETOKEN"
                    }
                }
//More content might appear here
}
```

- 3. 일반 텍스트 편집기에서 다음 내용을 추가할 파일을 JSON 형식으로 생성합니다.
 - 승인 작업이 들어 있는 파이프라인의 이름.
 - 승인 작업이 들어 있는 단계의 이름.
 - 승인 작업의 이름.
 - 이전 단계에서 수집한 토큰 값.
 - 작업에 대한 응답(승인됨 또는 거부됨). 응답은 대문자여야 합니다.
 - 요약 설명.

이전 예제 MyFirstPipeline에서 파일은 다음과 같은 형식이어야 합니다.

```
"pipelineName": "MyFirstPipeline",
```

{

```
"stageName": "MyApprovalStage",
"actionName": "MyApprovalAction",
"token": "1a2b3c4d-573f-4ea7-a67E-XAMPLETOKEN",
"result": {
    "status": "Approved",
    "summary": "The new design looks good. Ready to release to customers."
}
```

- 4. approvalstage-approved.json과 같은 이름으로 파일을 저장합니다.
- 5. 다음과 유사하게 승인 JSON 파일의 이름을 지정하여 put-approval-result 명령을 실행합니다.

▲ Important 파일 이름 앞에 file://를 포함해야 합니다. 이 명령에 필수적입니다.

```
aws codepipeline put-approval-result --cli-input-json file://approvalstage-
approved.json
```

CodePipeline에서 수동 승인 알림에 대한 JSON 데이터 형식

Amazon SNS 알림을 사용하는 승인 작업의 경우 파이프라인이 중지되면 작업에 대한 JSON 데이터가 생성되고 Amazon SNS에 게시됩니다. JSON 출력을 사용하여 메시지를 Amazon SQS 대기열로 전송 하거나 AWS Lambda에서 함수를 호출할 수 있습니다.

Note

이 설명서에서는 JSON을 사용하여 알림을 구성하는 방법을 다루지 않습니다. 자세한 내용 은 Amazon SNS 개발자 안내서에서 <u>Amazon SQS 대기열로 Amazon SNS 메시지 전송</u> 및 <u>Amazon SNS 알림을 사용하여 Lambda 함수 호출</u>을 참조하세요.

다음 예에서는 CodePipeline 승인에서 사용할 수 있는 JSON 출력의 구조를 보여 줍니다.

```
{
    "region": "us-east-2",
    "consoleLink": "https://console.aws.amazon.com/codepipeline/home?region=us-east-2#/
view/MyFirstPipeline",
```

"approval": {
"pipelineName": "MyFirstPipeline",
"stageName": "MyApprovalStage",
"actionName": "MyApprovalAction",
"token": "1a2b3c4d-573f-4ea7-a67E-XAMPLETOKEN",
"expires": "2016-07-07T20:22Z",
<pre>"externalEntityLink": "http://example.com",</pre>
"approvalReviewLink": "https://console.aws.amazon.com/codepipeline/
home?region=us-east-2#/view/MyFirstPipeline/MyApprovalStage/MyApprovalAction/
approve/1a2b3c4d-573f-4ea7-a67E-XAMPLETOKEN",
"customData": "Review the latest changes and approve or reject within seven
days."
}
}

CodePipeline에 교차 리전 작업 추가

AWS CodePipeline 에는 자동 릴리스 프로세스에 대한 빌드, 테스트 및 배포 리소스를 구성하는 데 도 움이 되는 여러 작업이 포함되어 있습니다. 파이프라인과 다른 AWS 리전에 있는 파이프라인에 작업을 추가할 수 있습니다. AWS 서비스 가 작업의 공급자이고이 작업 유형/공급자 유형이 파이프라인과 다 른 AWS 리전에 있는 경우 이는 리전 간 작업입니다.

Note

교차 리전 작업은 지원되며 CodePipeline이 지원되는 AWS 리전에서만 생성할 수 있습니다. CodePipeline에 지원되는 AWS 리전 목록은 섹션을 참조하세요<u>in AWS CodePipeline 할당량</u>.

콘솔 AWS CLI또는 AWS CloudFormation 를 사용하여 파이프라인에 리전 간 작업을 추가할 수 있습니 다.

Note

CodePipeline의 특정 작업 유형은 특정 AWS 리전에서만 사용할 수 있습니다. 또한 작업 유형 을 사용할 수 있지만 해당 작업 유형의 특정 AWS 공급자를 사용할 수 없는 AWS 리전이 있을 수 있습니다. 파이프라인을 생성하거나 편집할 때 파이프라인 리전에 아티팩트 버킷이 있어야 하며 작업을 실행하 려는 리전마다 아티팩트 버킷 하나가 있어야 합니다. ArtifactStores 파라미터에 대한 자세한 내용 은 CodePipeline 파이프라인 구조 참조를 참조하세요.

Note

CodePipeline은 리전 간 작업을 수행할 때 한 AWS 리전에서 다른 리전으로의 아티팩트 복사 를 처리합니다.

콘솔을 사용하여 파이프라인 또는 교차 리전 작업을 생성하는 경우 기본 아티팩트 버킷은 작업이 있는 리전의 CodePipeline에 의해 구성됩니다. AWS CLI AWS CloudFormation, 또는 SDK를 사용하여 파이 프라인 또는 리전 간 작업을 생성할 때 작업이 있는 각 리전에 대한 아티팩트 버킷을 제공합니다.

Note

리전 간 작업과 동일한 AWS 리전 및 파이프라인과 동일한 계정에 아티팩트 버킷 및 암호화 키 를 생성해야 합니다.

다음 작업 유형에는 교차 리전 작업을 생성할 수 없습니다.

- 소스 작업
- 타사 작업
- 사용자 지정 작업
 - Note

CodePipeline에서 리전 간 Lambda 간접 호출 작업을 사용하는 경우 <u>PutJobSuccessResult</u> 및 <u>PutJobFailureResult</u>를 사용한 lambda 실행 상태는 Lambda 함수가 있는 AWS 리전으로 전송 해야 하며 CodePipeline이 있는 리전으로 전송해서는 안 됩니다.

파이프라인에 단계의 일부로 교차 리전 작업이 포함될 경우 CodePipeline은 파이프라인 리전에서 작업 리전으로 교차 리전 작업의 입력 아티팩트만 복제합니다.

Note

CloudWatch Events 변경 감지 리소스가 있는 리전과 파이프라인 리전은 동일하게 유지됩니다. 파이프라인이 호스팅되는 리전은 변경되지 않습니다.

파이프라인에서 교차 리전 작업 관리(콘솔)

CodePipeline 콘솔을 사용하여 교차 리전 작업을 기존 파이프라인에 추가할 수 있습니다. 파이프라인 생성 마법사를 사용하여 교차 리전 작업으로 새로운 파이프라인을 생성하려면 <u>사용자 지정 파이프라</u> 인 생성(콘솔) 항목을 참조하십시오.

콘솔에서 작업 공급자와 해당 공급자에 대해 해당 리전에서 생성한 리소스를 나열하는 리전 필드를 선 택하여 파이프라인 단계에서 교차 리전 작업을 생성합니다. 교차 리전 작업을 추가하면 CodePipeline 은 작업 리전에서 별도의 아티팩트 버킷을 사용합니다. 교차 리전 아티팩트 버킷에 대한 자세한 내용은 CodePipeline 파이프라인 구조 참조 항목을 참조하십시오.

파이프라인 단계에 교차 리전 작업 추가(콘솔)

콘솔을 사용하여 파이프라인에 교차 리전 작업을 추가합니다.

Note

변경 사항이 저장된 파이프라인이 실행 중일 경우 실행이 완료되지 않습니다.

교차 리전 작업을 추가하려면

- 1. http://console.aws.amazon.com/codesuite/codepipeline/home에서 콘솔에 로그인합니다.
- 2. 파이프라인을 선택한 다음 편집을 선택합니다.
- 3. 다이어그램의 하단에서 새 단계를 추가하려면 + Add stage(+ 단계 추가)를 선택하고 기존 단계에 작업을 추가하려면 Edit stage(단계 편집)을 선택합니다.
- 4. 편집: <단계>에서 + Add action group(+ 작업 그룹 추가)를 선택하여 연속 작업을 추가합니다. 또 는 + Add action(+ 작업 추가)를 선택하여 병렬 작업을 추가합니다.
- 5. 작업 편집 페이지에서 다음 작업을 수행하십시오.
 - a. 작업 이름에 교차 리전 작업 이름을 입력합니다.

- b. Action provider(작업 공급자)에서 작업 공급자를 선택합니다.
- c. 리전에서 작업에 대한 리소스를 생성했거나 생성할 계획인 AWS 리전을 선택합니다. 리전을 선택하면 해당 리전의 사용 가능한 리소스가 선택 목록에 나열됩니다. 리전 필드는이 작업 유 형 및 공급자 유형에 대해 AWS 리소스가 생성되는 위치를 지정합니다. 이 필드는 작업 공급 자가 AWS 서비스인 작업에 대해서만 표시합니다. 리전 필드의 기본값은 파이프라인과 동일 한 AWS 리전 입니다.
- d. 입력 아티팩트에서 이전 단계의 적절한 입력을 선택합니다. 예를 들어 이전 단계가 소스 단계
 인 경우 SourceArtifact를 선택합니다.
- e. 구성 중인 작업 공급자에 대한 필수 필드를 모두 작성합니다.
- f. 출력 아티팩트에서 다음 단계의 적절한 출력을 선택합니다. 예를 들어 다음 단계가 배포 단계 인 경우 BuildArtifact를 선택합니다.
- g. 저장(Save)을 선택합니다.
- 6. 편집: <단계>에서 완료를 선택합니다.
- 7. 저장(Save)을 선택합니다.

파이프라인 단계에서 교차 리전 작업 편집(콘솔)

콘솔을 사용하여 파이프라인에서 기존의 교차 리전 작업을 편집합니다.

Note

변경 사항이 저장된 파이프라인이 실행 중일 경우 실행이 완료되지 않습니다.

교차 리전 작업 편집 방법

- 1. https://console.aws.amazon.com/codesuite/codepipeline/home.에서 콘솔에 로그인합니다.
- 2. 파이프라인을 선택한 다음 편집을 선택합니다.
- 3. Edit stage(단계 편집)을 선택합니다.
- 4. 편집: <단계>에서 아이콘을 선택하여 기존 작업을 편집합니다.
- 5. 작업 편집 페이지에서 해당 필드를 적절하게 변경합니다.
- 6. 편집: <단계>에서 완료를 선택합니다.
- 7. 저장(Save)을 선택합니다.

파이프라인 단계에서 교차 리전 작업 삭제(콘솔)

콘솔을 사용하여 파이프라인에서 기존의 교차 리전 작업을 삭제합니다.

Note

변경 사항이 저장된 파이프라인이 실행 중일 경우 실행이 완료되지 않습니다.

교차 리전 작업 삭제 방법

- 1. http://console.aws.amazon.com/codesuite/codepipeline/home에서 콘솔에 로그인합니다.
- 2. 파이프라인을 선택한 다음 편집을 선택합니다.
- 3. Edit stage(단계 편집)을 선택합니다.
- 4. 편집: <단계>에서 아이콘을 선택하여 기존 작업을 삭제합니다.
- 5. 편집: <단계>에서 완료를 선택합니다.
- 6. 저장(Save)을 선택합니다.

파이프라인에 교차 리전 작업 추가(CLI)

를 사용하여 기존 파이프라인 AWS CLI 에 리전 간 작업을 추가할 수 있습니다.

를 사용하여 파이프라인 단계에서 리전 간 작업을 생성하려면 선택적 region 필드와 함께 구성 작업 을 AWS CLI추가합니다. 작업의 리전에 아티팩트 버킷이 이미 생성되어 있어야 합니다. 단일 리전 파이 프라인의 artifactStore 파라미터를 제공하는 대신 artifactStores 파라미터를 사용하여 각 리 전의 아티팩트 버킷 목록을 포함시킵니다.

1 Note

이 연습과 예제에서 *RegionA*는 파이프라인이 생성되는 리전입니다. CodePipeline에서 사용 된 파이프라인 아티팩트와 서비스 역할을 저장하는 데 사용된 *RegionA* Amazon S3버킷에 액 세스할 수 있습니다. *RegionB*는 CodeDeploy에서 사용하는 CodeDeploy 애플리케이션, 배포 그룹, 서비스 역할이 생성된 리전입니다.

사전 조건

다음을 생성해야 합니다.

- *RegionA*의 파이프라인.
- RegionB의 Amazon S3 아티팩트 버킷입니다.
- *RegionB*의 CodeDeploy 애플리케이션 및 교차 리전 배포 작업을 위한 배포 그룹과 같은 작업에 대한 리소스입니다.

파이프라인에 교차 리전 작업 추가(CLI)

AWS CLI 를 사용하여 교차 리전 작업을 파이프라인에 추가합니다.

교차 리전 작업을 추가하려면

1. *RegionA*의 파이프라인의 경우, get-pipeline 명령을 실행하여 파이프라인 구조를 JSON 파일로 복사합니다. 예를 들어, MyFirstPipeline라는 파이프라인의 경우 다음 명령을 입력합니다.

aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json

이 명령은 아무 것도 반환하지 않지만 생성한 파일이 명령을 실행한 디렉터리에 표시되어야 합니 다.

 region 필드를 추가하여 작업을 위한 리전 및 리소스가 포함된 교차 리전 작업으로 새 단계를 추 가합니다. 다음 JSON 예제에서는 공급자가 CodeDeploy인 교차 리전 배포 작업으로 배포 단계를 새 리전 us-east-1에 추가합니다.

```
"name": "Deploy",
"actions": [
    {
        "inputArtifacts": [
            {
                 "name": "SourceArtifact"
            }
        ],
        "name": "Deploy",
        "region": "RegionB",
        "actionTypeId": {
            "category": "Deploy",
            "owner": "AWS",
            "version": "1",
            "provider": "CodeDeploy"
        },
```

{

```
"outputArtifacts": [],
    "configuration": {
        "ApplicationName": "name",
        "DeploymentGroupName": "name"
    },
    "runOrder": 1
}
```

 파이프라인 구조에서 artifactStore 필드를 제거하고, 새로운 교차 리전 작업을 위한 artifactStores 맵을 추가합니다. 매핑에는 작업이 있는 각 AWS 리전에 대한 항목이 포함되 어야 합니다. 매핑의 각 항목에 대해 리소스는 해당 AWS 리전에 있어야 합니다. 아래 예에서 ID-A는 RegionA의 암호화 키 ID이며 ID-B는 RegionB의 암호화 키 ID입니다.

```
"artifactStores":{
   "RegionA":{
      "encryptionKey":{
         "id":"ID-A",
         "type":"KMS"
      },
      "location":"Location1",
      "type":"S3"
   },
   "RegionB":{
      "encryptionKey":{
         "id":"ID-B",
         "type":"KMS"
      },
      "location":"Location2",
      "type":"S3"
   }
}
```

다음 JSON 예제는 us-west-2 버킷을 my-storage-bucket으로 표시하고 이름이 mystorage-bucket-us-east-1으로 지정된 us-east-1 버킷을 새로 추가합니다.

```
"artifactStores": {
    "us-west-2": {
        "type": "S3",
        "location": "my-storage-bucket"
    },
    "us-east-1": {
        "type": "S3",
```

"location": "my-storage-bucket-us-east-1"
}

get-pipeline 명령을 사용하여 검색한 파이프라인 구조로 작업을 수행할 경우, JSON 파일에서 metadata 행을 제거하십시오. 이렇게 하지 않으면 update-pipeline 명령에서 사용할 수 없습니다.
 "metadata": { } 행과, "created", "pipelineARN" 및 "updated" 필드를 제거합니다.

예를 들어, 구조에서 다음 행을 삭제합니다.

```
"metadata": {
   "pipelineArn": "arn:aws:codepipeline:region:account-ID:pipeline-name",
   "created": "date",
   "updated": "date"
}
```

파일을 저장합니다.

},

5. 변경 사항을 적용하려면 파이프라인 JSON 파일을 지정하여 update-pipeline 명령을 실행합니다.

▲ Important 파일 이름 앞에 file://를 포함해야 합니다. 이 명령에 필수적입니다.

aws codepipeline update-pipeline --cli-input-json file://pipeline.json

이 명령은 편집한 파이프라인의 전체 구조를 반환합니다. 출력 결과는 다음과 비슷합니다.

```
"category": "Source",
                 "owner": "AWS",
                "version": "1",
                "provider": "CodeCommit"
            },
            "outputArtifacts": [
                {
                     "name": "SourceArtifact"
                }
            ],
            "configuration": {
                "PollForSourceChanges": "false",
                "BranchName": "main",
                "RepositoryName": "MyTestRepo"
            },
            "runOrder": 1
        }
    ]
},
{
    "name": "Deploy",
    "actions": [
        {
            "inputArtifacts": [
                {
                     "name": "SourceArtifact"
                }
            ],
            "name": "Deploy",
            "region": "us-east-1",
            "actionTypeId": {
                "category": "Deploy",
                "owner": "AWS",
                "version": "1",
                "provider": "CodeDeploy"
            },
            "outputArtifacts": [],
            "configuration": {
                "ApplicationName": "name",
                 "DeploymentGroupName": "name"
            },
            "runOrder": 1
        }
    ]
```

```
}
],
],
"name": "AnyCompanyPipeline",
"artifactStores": {
    "us-west-2": {
        "type": "S3",
        "location": "my-storage-bucket"
        },
        "us-east-1": {
            "type": "S3",
            "location": "my-storage-bucket-us-east-1"
        }
    }
}
```

Note

update-pipeline 명령을 실행하면 파이프라인이 중지됩니다. update-pipeline 명령을 실행 할 때 파이프라인을 통해 개정을 실행하는 중이라면 해당 실행이 중지됩니다. 업데이트 된 파이프라인을 통해 해당 개정을 실행하려면 파이프라인을 수동으로 시작해야 합니다. start-pipeline-execution 명령을 사용하여 수동으로 파이프라인을 시작합니다.

6. 파이프라인을 업데이트하면 교차 리전 작업이 콘솔에 표시됩니다.

Source	٢	
Succeeded - 29 days ago dbb74de1		
Source: Added codepipeline-codecomm	t-events.yml dbb74de1 🖸	
Disable transition		
⊘ Deploy		View current revisions
Deploy Ø Cross-region action O Succeeded - 29 days ago	٥	
Details 2		

파이프라인에 교차 리전 작업 추가(AWS CloudFormation)

AWS CloudFormation 를 사용하여 기존 파이프라인에 리전 간 작업을 추가할 수 있습니다.

를 사용하여 리전 간 작업을 추가하려면 AWS CloudFormation

1. 다음 예와 같이 Region 파라미터를 템플릿의 ActionDeclaration 리소스에 추가합니다.

ActionDeclaration:
Type: Object
Properties:
ActionTypeId:
Type: ActionTypeId
Required: true
Configuration:
Туре: Мар
InputArtifacts:
Type: Array
ItemType:
Type: InputArtifact
Name:
Type: String
Required: true
OutputArtifacts:
Type: Array
ItemType:
Type: OutputArtifact
RoleArn:
Type: String
RunOrder:
Type: Integer
Region:
Type: String

2. 이 예에 나온 것처럼 Mappings 아래에서 RegionA 및 RegionB 키에 값을 매핑하는 SecondRegionMap라는 매핑을 위한 리전 맵을 추가합니다. 다음과 같이 Pipeline 리소스의 artifactStore 필드에 새로운 교차 리전 작업을 위한 artifactStores 맵을 추가합니다.

```
Mappings:
SecondRegionMap:
RegionA:
SecondRegion: "RegionB"
RegionB:
SecondRegion: "RegionA"
```

```
Properties:
ArtifactStores:
-
Region: RegionB
ArtifactStore:
Type: "S3"
Location: test-cross-region-artifact-store-bucket-RegionB
-
Region: RegionA
ArtifactStore:
Type: "S3"
Location: test-cross-region-artifact-store-bucket-RegionA
```

다음 YAML 예제는 *RegionA* 버킷을 us-west-2로 표시하고, 새로운 *RegionB* 버킷 eucentral-1을 추가합니다.

```
Mappings:
  SecondRegionMap:
    us-west-2:
      SecondRegion: "eu-central-1"
    eu-central-1:
      SecondRegion: "us-west-2"
. . .
          Properties:
            ArtifactStores:
                Region: eu-central-1
                ArtifactStore:
                  Type: "S3"
                  Location: test-cross-region-artifact-store-bucket-eu-central-1
                Region: us-west-2
                ArtifactStore:
                  Type: "S3"
                  Location: test-cross-region-artifact-store-bucket-us-west-2
```

- 3. 업데이트된 템플릿을 로컬 컴퓨터에 저장하고 AWS CloudFormation 콘솔을 엽니다.
- 4. 스택을 선택한 후 현재 스택에 대한 변경 세트 만들기를 선택합니다.

- 5. 템플릿을 업로드한 후 AWS CloudFormation에 나열된 변경 사항을 확인합니다. 이는 스택에 적용 될 변경 사항입니다. 목록에 새로운 리소스가 표시됩니다.
- 6. 실행을 선택합니다.

변수 작업

CodePipeline의 일부 작업은 변수를 생성합니다. 변수를 사용하려면 다음을 수행합니다.

- 작업에 네임스페이스를 할당하여 생성된 변수를 다운스트림 작업 구성에서 사용할 수 있도록 합니다.
- 작업에서 생성된 변수를 사용하도록 다운스트림 작업을 구성합니다.

각 작업 실행에 대한 세부 정보를 보고 실행 시간에 작업에서 생성된 각 출력 변수의 값을 확인할 수 있습니다.

변수 사용에 대한 단계별 예제를 보려면

- 업스트림 작업(CodeCommit)의 변수를 사용하고 출력 변수를 생성하는 Lambda 작업에 대한 자습서 는 자습서: Lambda 호출 작업과 함께 변수 사용 섹션을 참조하세요.
- 업스트림 CloudFormation AWS CloudFormation 작업의 스택 출력 변수를 참조하는 작업이 포함된 자습서는 섹션을 참조하세요<u>자습서: AWS CloudFormation 배포 작업의 변수를 사용하는 파이프라</u> 인 생성.
- CodeCommit 커밋 ID 및 커밋 메시지로 확인되는 출력 변수를 참조하는 메시지 텍스트가 포함된 수 동 승인 작업의 예는 예: 수동 승인에 변수 사용 섹션을 참조하세요.
- GitHub 브랜치 이름으로 확인되는 환경 변수를 포함하는 CodeBuild 작업의 예는 <u>예: CodeBuild 환경</u> 변수와 함께 BranchName 변수 사용 섹션을 참조하세요.
- CodeBuild 작업은 빌드의 일부로 내보낸 모든 환경 변수를 변수로 생성합니다. 자세한 내용은 <u>CodeBuild 작업 출력 변수</u> 단원을 참조하십시오. CodeBuild에서 사용할 수 있는 환경 변수 목록은 AWS CodeBuild 사용 설명서의 <u>빌드 환경의 환경 변수</u>를 참조하세요.

주제

- 변수에 대한 작업 구성
- 출력 변수 보기
- 예: 수동 승인에 변수 사용

• 예: CodeBuild 환경 변수와 함께 BranchName 변수 사용

변수에 대한 작업 구성

파이프라인에 작업을 추가할 때 해당 작업에 네임스페이스를 할당하고 이전 작업의 변수를 사용하도 록 구성할 수 있습니다.

변수를 사용하여 작업 구성(콘솔)

이 예제에서는 CodeCommit 소스 작업과 CodeBuild 빌드 작업에서 파이프라인을 생성합니다. CodeBuild 작업은 CodeCommit 작업에서 생성된 변수를 소비하도록 구성됩니다.

네임스페이스를 지정하지 않으면 작업 구성에서 변수를 참조할 수 없습니다. 콘솔을 사용하여 파이프 라인을 생성하면 각 작업의 네임스페이스가 자동으로 생성됩니다.

변수를 사용하여 파이프라인을 생성하려면

- 1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> <u>codepipeline/home</u>://https//https//ht
- 2. [파이프라인 생성]을 선택합니다. 파이프라인의 이름을 입력한 후 다음을 선택합니다.
- 소스의 공급자에서 CodeCommit을 선택합니다. 소스 작업에 대한 CodeCommit 리포지토리와 분 기를 선택한 후 다음을 선택합니다.
- 4. 빌드의 공급자에서 CodeBuild를 선택합니다. 기존 CodeBuild 빌드 프로젝트 이름을 선택하거나 프로젝트 생성을 선택합니다. Create build project(빌드 프로젝트 생성)에서 빌드 프로젝트를 생성 한 다음, Return to CodePipeline(CodePipeline으로 돌아가기)을 선택합니다.

환경 변수에서 Add environment variables(환경 변수 추가)를 선택합니다. 예를 들면, 변수 구 문 #{codepipeline.PipelineExecutionId}를 사용하여 실행 ID를 입력하고 변수 구문 #{SourceVariables.CommitId}를 사용하여 커밋 ID를 입력합니다.

Note

마법사의 모든 작업 구성 필드에 변수 구문을 입력할 수 있습니다.

- 5. 생성(Create)을 선택합니다.
- 파이프라인을 생성하고 나면 마법사에서 생성된 네임스페이스를 확인할 수 있습니다. 파이프라인 에서 네임스페이스를 보려는 단계에 대한 아이콘을 선택합니다. 이 예제에서는 소스 작업의 자동 생성 네임스페이스인 SourceVariables이 표시됩니다.

Configuration		×
Variable namespace	SourceVariables	
Output artifact	SourceArtifact	
BranchName	main	
PollForSourceChanges	false	
RepositoryName	repo	
		Done

기존 작업의 네임스페이스를 편집하려면

- 1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> <u>codepipeline/home://https://https://https://ittps//ittps</u>
- 편집할 파이프라인을 선택한 다음 편집을 선택합니다. 소스 단계에서 Edit stage(단계 편집)를 선 택합니다. CodeCommit 작업을 추가합니다.
- 작업 편집에서 Variable namespace(변수 네임스페이스) 필드를 확인합니다. 기존 작업이 이전 에 생성되었거나 마법사를 사용하지 않고 생성된 경우에는 네임스페이스를 추가해야 합니다. Variable namespace(변수 네임스페이스)에 네임스페이스 이름을 입력한 다음, 저장을 선택합니 다.

출력 변수를 보려면

- 1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https//https/
- 파이프라인이 생성되어 성공적으로 실행되면 Action execution details(작업 실행 세부 정보) 페이 지에서 변수를 확인할 수 있습니다. 자세한 내용은 변수 보기(콘솔)을 참조하세요.

변수에 대한 작업 구성(CLI)

create-pipeline 명령을 사용하여 파이프라인을 생성하거나 update-pipeline 명령을 사용하여 파이프라 인을 편집하면 작업 구성에서 변수를 참조하거나 사용할 수 있습니다.

네임스페이스가 지정되어 있지 않으면 작업에서 생성된 변수를 다운스트림 작업 구성에서 참조할 수 없습니다.

네임스페이스를 사용하여 작업을 구성하려면

 <u>파이프라인 스테이지 및 작업 생성</u>의 단계에 따라 CLI를 사용하여 파이프라인을 생성합니다. 입력 파일을 시작하여 create-pipeline 명령에 --cli-input-json 파라미터를 제공합니다. 파이프라 인 구조에서 namespace 파라미터를 추가하고 이름(예: SourceVariables)을 지정합니다.

```
{
    "inputArtifacts": [],
    "name": "Source",
    "region": "us-west-2",
    "namespace": "SourceVariables",
    "actionTypeId": {
        "category": "Source",
        "owner": "AWS",
        "version": "1",
        "provider": "CodeCommit"
      },
      "outputArtifacts": [
```

- 2. MyPipeline.json과 같은 이름으로 파일을 저장합니다.
- 3. 터미널(Linux, macOS 또는 Unix) 또는 명령 프롬프트(Windows)에서, <u>create-pipeline</u> 명령을 실행 하고 파이프라인을 생성합니다.

create-pipeline 명령을 실행할 때 생성한 파일을 호출합니다. 예시:

aws codepipeline create-pipeline --cli-input-json file://MyPipeline.json

변수를 소비하도록 다운스트림 작업을 구성하려면

 update-pipeline 명령에 --cli-input-json 파라미터를 제공하도록 입력 파일을 편집합니다. 다 운스트림 작업에서 해당 작업에 대한 구성에 변수를 추가합니다. 변수는 마침표로 구분된 네임스 페이스와 키로 구성됩니다. 예를 들어 파이프라인 실행 ID와 소스 커밋 ID에 변수를 추가하려면 변 수 #{codepipeline.PipelineExecutionId}에 네임스페이스 codepipeline를 지정합니 다. 변수 #{SourceVariables.CommitId}에 네임스페이스 SourceVariables를 지정합니 다.

```
"name": "Build",
    "actions": [
        {
            "outputArtifacts": [
                {
                     "name": "BuildArtifacts"
                }
            ],
            "name": "Build",
            "configuration": {
                "EnvironmentVariables": "[{\"name\":\"Execution_ID\",\"value
\":\"#{codepipeline.PipelineExecutionId}\",\"type\":\"PLAINTEXT\"},{\"name\":
\"Commit_ID\", \"value\": \"#{SourceVariables.CommitId}\", \"type\": \"PLAINTEXT\"}]",
                "ProjectName": "env-var-test"
            },
            "inputArtifacts": [
                {
                     "name": "SourceArtifact"
                }
            ],
            "region": "us-west-2",
            "actionTypeId": {
                "provider": "CodeBuild",
                "category": "Build",
                "version": "1",
                "owner": "AWS"
            },
            "runOrder": 1
        }
    ]
},
```

- 2. MyPipeline.json과 같은 이름으로 파일을 저장합니다.
- 3. 터미널(Linux, macOS 또는 Unix) 또는 명령 프롬프트(Windows)에서, <u>create-pipeline</u> 명령을 실행 하고 파이프라인을 생성합니다.

create-pipeline 명령을 실행할 때 생성한 파일을 호출합니다. 예시:

aws codepipeline create-pipeline --cli-input-json file://MyPipeline.json

출력 변수 보기

작업 실행 세부 정보를 확인하여 각 실행에 특정한 해당 작업에 대한 변수를 확인할 수 있습니다.

변수 보기(콘솔)

콘솔을 사용하여 작업에 대한 변수를 확인할 수 있습니다.

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https//https://https://https://https//http

AWS 계정과 연결된 모든 파이프라인의 이름이 표시됩니다.

- 2. 이름에서 파이프라인의 이름을 선택합니다.
- 3. 내역 보기를 선택합니다.
- 파이프라인이 성공적으로 실행되면 소스 작업에서 생성된 변수를 확인할 수 있습니다. 내역 보 기를 선택합니다. 파이프라인 실행에 대한 작업 목록에서 소스를 선택하여 CodeCommit 작업에 대한 작업 실행 세부 정보를 확인합니다. 작업 세부 정보 화면의 Output variables(출력 변수) 아래 에 있는 변수를 확인합니다.

Output variables		
Кеу	Value	
AuthorDate	2019-10-29T03:32:21Z	
BranchName	master	
CommitId	8cf40f22b935b306f06d214517e98aet	
CommitMessage	Added LICENSE.txt	
CommitterDate	2019-10-29T03:32:21Z	
RepositoryName	repo	

5. 파이프라인이 성공적으로 실행되면 빌드 작업에서 사용된 변수를 확인할 수 있습니다. 내역 보 기를 선택합니다. 파이프라인 실행에 대한 작업 목록에서 빌드를 선택하여 CodeBuild 작업에 대한 작업 실행 세부 정보를 확인합니다. 작업 세부 정보 페이지의 Action configuratio(작업 구성)에서 변수를 확인합니다. 자동 생성된 네임스페이스가 표시됩니다.
Show resolved configuration

Action configuration

EnvironmentVariables [{"name":"Execution_ID","value":"# {codepipeline.PipelineExecutionId]","type":"PLAINTEXT"), {"name":"Commit_ID","value":"#{SourceVariables.CommitId]","type":"PLAINTEXT"}]

기본적으로 Action configuratio(작업 구성)에는 변수 구문이 표시됩니다. Show resolved configuration(확인된 구성 표시)를 선택하여 작업 실행 중에 생성된 값을 표시하도록 목록을 토글 할 수 있습니다.

ProjectName

dk-var-build-proj

Action configuration		Show resolved configuration
EnvironmentVariables [{"name":"Execution_ID","value":"ab9f6ead-a64c-4fd5-b6aa- 3bf ","type":"PLAINTEXT"), {"name":"Commit_ID","value":"8cf40f22b935b306f06d214517es ","type" :"PLAINTEXT"}]	ProjectName var-build-proj	

변수 보기(CLI)

list-action-executions 명령을 사용하여 작업에 대한 변수를 확인할 수 있습니다.

1. 다음 명령을 사용합니다.

aws codepipeline list-action-executions

여기에 나온 것처럼 출력에 outputVariables 파라미터가 표시됩니다.

```
"outputVariables": {
    "BranchName": "main",
    "CommitMessage": "Updated files for test",
    "AuthorDate": "2019-11-08T22:24:34Z",
    "CommitId": "d99b0083cc10EXAMPLE",
    "CommitterDate": "2019-11-08T22:24:34Z",
    "RepositoryName": "variables-repo"
},
```

2. 다음 명령을 사용합니다.

aws codepipeline get-pipeline --name <pipeline-name>

CodeBuild 작업에 대한 작업 구성에서 변수를 확인할 수 있습니다.

```
{
    "name": "Build",
    "actions": [
        ſ
            "outputArtifacts": [
                {
                    "name": "BuildArtifact"
                }
            ],
            "name": "Build",
            "configuration": {
                "EnvironmentVariables": "[{\"name\":\"Execution_ID\",\"value
\":\"#{codepipeline.PipelineExecutionId}\",\"type\":\"PLAINTEXT\"},{\"name\":
\"Commit_ID\",\"value\":\"#{SourceVariables.CommitId}\",\"type\":\"PLAINTEXT\"}]",
                "ProjectName": "env-var-test"
            },
            "inputArtifacts": [
                {
                    "name": "SourceArtifact"
                }
            ],
            "region": "us-west-2",
            "actionTypeId": {
                "provider": "CodeBuild",
                "category": "Build",
                "version": "1",
                "owner": "AWS"
            },
            "runOrder": 1
        }
    ]
},
```

예: 수동 승인에 변수 사용

작업에 대한 네임스페이스를 지정하고 해당 작업으로 출력 변수가 생성되는 경우 승인 메시지에 변수 를 표시하는 수동 승인을 추가할 수 있습니다. 이 예에서는 수동 승인 메시지에 변수 구문을 추가하는 방법을 보여 줍니다. 1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https//https/

AWS 계정과 연결된 모든 파이프라인의 이름이 표시됩니다. 승인을 추가할 파이프라인을 선택합 니다.

- 파이프라인을 편집하기 위해 편집을 선택합니다. 소스 작업 뒤에 수동 승인을 추가합니다. 작업 이 름에 승인 작업의 이름을 입력합니다.
- 3. Action provider(작업 공급자)에서 수동 승인을 선택합니다.
- 4. 검토용 URL에서 CodeCommit URL에 CommitId의 변수 구문을 추가합니다. 소스 작업에 할 당된 네임스페이스를 사용해야 합니다. 예를 들어 기본 네임스페이스가 SourceVariables인 CodeCommit 작업의 변수 구문은 #{SourceVariables.CommitId}입니다.

주석에서 Commit Message에 커밋 메시지를 입력합니다.

Please approve this change. Commit message: #{SourceVariables.CommitMessage}

5. 파이프라인이 성공적으로 실행되면 승인 메시지에서 변수 값을 확인할 수 있습니다.

예: CodeBuild 환경 변수와 함께 BranchName 변수 사용

CodeBuild 작업을 파이프라인에 추가할 때 CodeBuild 환경 변수를 사용하여 업스트림 소스 작업의 BranchName 출력 변수를 참조할 수 있습니다. CodePipeline의 작업에서 출력 변수를 사용하여 빌드 명령에 사용할 고유한 CodeBuild 환경 변수를 만들 수 있습니다.

이 예제에서는 GitHub 소스 작업의 출력 변수 구문을 CodeBuild 환경 변수에 추가하는 방법을 보여줍 니다. 이 예제의 출력 변수 구문은 BranchName에 대한 GitHub 소스 작업 출력 변수를 나타냅니다. 작 업이 성공적으로 실행되면 변수가 확인되어 GitHub 브랜치 이름을 표시합니다.

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https//https://https://https

AWS 계정과 연결된 모든 파이프라인의 이름이 표시됩니다. 승인을 추가할 파이프라인을 선택합 니다.

- 2. 파이프라인을 편집하기 위해 편집을 선택합니다. CodeBuild 작업이 포함된 단계에서 단계 편집을 선택합니다.
- 3. 아이콘을 선택하여 CodeBuild 작업을 편집합니다.
- 4. 작업 편집 페이지에서 환경 변수에 다음을 입력합니다.

- 이름에 환경 변수의 이름을 입력합니다.
- 값에 소스 작업에 할당된 네임스페이스가 포함된 파이프라인 출력 변수의 변수 구문을 입력합니다. 예를 들어 기본 네임스페이스가 SourceVariables인 GitHub 작업의 출력 변수 구문은 #{SourceVariables.BranchName}입니다.
- 유형에서 Plaintext를 선택합니다.
- 파이프라인이 성공적으로 실행되고 나면 확인된 출력 변수가 환경 변수의 값인 것을 확인할 수 있 습니다. 다음 중 하나를 선택합니다.
 - CodePipeline 콘솔: 파이프라인을 선택한 다음 기록을 선택합니다. 최신 파이프라인 실행을 선 택합니다.
 - 타임라인에서 소스 선택기를 선택합니다. 이것은 GitHub 출력 변수를 생성하는 소스 작업입니다. 실행 세부 정보 보기를 선택합니다. 출력 변수에서 이 작업으로 생성된 출력 변수 목록을 확인합니다.
 - 타임라인에서 빌드 선택기를 선택합니다. 이것은 빌드 프로젝트의 CodeBuild 환경 변수를 지 정하는 빌드 작업입니다. 실행 세부 정보 보기를 선택합니다. 작업 구성에서 CodeBuild 환경 변수를 확인합니다. 해결된 구성 보기를 선택합니다. 환경 변수 값은 GitHub 소스 작업에서 해 결된 BranchName 출력 변수입니다. 이 예시에서의 해결된 값은 main입니다.

자세한 내용은 변수 보기(콘솔) 단원을 참조하십시오.

 CodeBuild 콘솔: 빌드 프로젝트를 선택하고 빌드 실행에 사용할 링크를 선택합니다. 환경 변 수에서 확인된 출력 변수는 CodeBuild 환경 변수의 값입니다. 이 예시에서 환경 변수 이름 BranchName이며 값은 GitHub 소스 작업에서 해결된 BranchName 출력 변수입니다. 이 예시 에서의 해결된 값은 main입니다.

Build logs Phase details Report	s Environment variables Build details	Resource utilization
Name	Value	Туре
BranchName	main	PLAINTEXT

CodePipeline에서 단계 전환 작업

전환은 비활성화하거나 활성화할 수 있는 파이프라인 단계 간 링크입니다. 전환은 기본적으로 활성화 되어 있습니다. 비활성화된 전환을 다시 활성화하면 30일 이상이 경과하지 않은 경우 최신 수정이 파이 프라인의 나머지 단계를 통해 실행됩니다. 새 변경이 감지되거나 파이프라인을 수동으로 다시 실행하 지 않은 경우 30일 이상 비활성화된 전환에 대한 파이프라인 실행이 재개되지 않습니다.

AWS CodePipeline 콘솔 또는를 사용하여 파이프라인의 단계 간 전환을 비활성화하거나 활성화 AWS CLI 할 수 있습니다.

Note

계속하도록 수동으로 승인될 때까지 승인 작업을 사용하여 파이프라인의 실행을 일시 중지할 수 있습니다. 자세한 내용은 스테이지에 수동 승인 작업 추가 단원을 참조하십시오.

주제

- 전환 비활성화 또는 활성화(콘솔)
- 전환 비활성화 또는 활성화(CLI)

전환 비활성화 또는 활성화(콘솔)

파이프라인의 전환을 비활성화하거나 활성화하려면

1. 에 로그인 AWS Management Console 하고 <u>http://console.aws.amazon.com/codesuite/</u> codepipeline/home://https//https

사용자의 AWS 계정에 연결된 모든 파이프라인의 이름이 표시됩니다.

- [Name]에서 전환을 활성화하거나 비활성화하려는 파이프라인의 이름을 선택합니다. 이렇게 하면 파이프라인 단계 간 전환을 포함하여 파이프라인의 세부 정보 보기가 열립니다.
- 실행하려는 마지막 단계 이후의 화살표를 찾은 다음 옆에 있는 버튼을 선택합니다. 예를 들어, 다음 예제 파이프라인에서, Staging(스테이징) 단계의 작업은 실행하지만, 프로덕션 단계의 작업은 실행하지 않으려면 해당 두 단계 사이의 전환 비활성화 버튼을 선택합니다.

View current revisions

4. 전환 비활성화 대화 상자에서 전환을 비활성화하는 사유를 입력한 후 비활성화를 선택합니다.

버튼이 변경되어 해당 전환이 화살표 이전 단계와 화살표 이후 단계 사이에서 비활성화되었음을 나타냅니다. 비활성화된 전환 뒤에 나오는 단계에서 이미 실행되고 있던 개정은 파이프라인을 통 해 계속할 수 있지만, 이후 개정은 비활성화된 전환을 계속하여 통과하지 못합니다.

⊙ Deploy	View current revisions
Deploy (j) CodeDeploy Succeeded - 3 minutes ago Details	
Source: Added codepipeline-codecommit-events.yml dbb	74de
⑦ Production	View current revisions
Deploy (j	
Succeeded - 2 minutes ago Details	\searrow

5. 화살표 옆의 전환 활성화 버튼을 선택합니다. [Enable transition] 대화 상자에서 [Enable]을 선택합니다. 파이프라인에서 두 단계 간 전환을 즉시 활성화합니다. 전환이 비활성화된 후 이전 단계를 통해 실행되고 있던 개정이 있는 경우, 잠시 후에, 파이프라인에서 이전에 비활성화된 전환 이후의 단계를 통해 최신 개정의 실행을 시작합니다. 파이프라인은 파이프라인의 남은 모든 단계를 통해 개정을 실행합니다.

Note
전환을 활성화한 후 변경 사항이 CodePipeline 콘솔에 표시되려면 몇 분 정도 걸릴 수 있습니다.

전환 비활성화 또는 활성화(CLI)

를 사용하여 단계 간 전환을 비활성화하려면 disable-stage-transition 명령을 AWS CLI실행합니다. 비 활성화된 전환을 활성화하려면 enable-stage-transition 명령을 실행합니다.

전환을 비활성화하려면

 터미널(Linux, macOS 또는 Unix) 또는 명령 프롬프트(Windows)를 열고 AWS CLI 를 사용하여 파 이프라인의 이름, 전환을 비활성화하려는 단계의 이름, 전환 유형 및 해당 단계로의 전환을 비활성 화하는 이유를 지정하여 disable<u>disable-stage-transition</u> 명령을 실행합니다. 콘솔과 다르게, 전환 을 단계로 비활성화할 것인지(인바운드) 아니면 모든 작업 완료 후에 단계 밖에서 전환을 비활성화 할 것인지(아웃바운드)도 지정해야 합니다.

예를 들어 *MyFirstPipeline*이라는 파이프라인에서 *Staging*이라는 스테이지로의 전환을 비 활성화하려면 다음과 유사한 명령을 입력합니다.

aws codepipeline disable-stage-transition --pipeline-name MyFirstPipeline --stagename Staging --transition-type Inbound --reason "My Reason"

이 명령은 아무 것도 반환하지 않습니다.

 전환이 비활성화되었는지 확인하려면 CodePipeline 콘솔에서 파이프라인을 확인하거나 getpipeline-state 명령을 실행합니다. 자세한 내용은 <u>파이프라인 보기(콘솔)</u> 및 <u>파이프라인 세부 정보</u> 및 이력 보기(CLI) 단원을 참조하세요.

전환을 활성화하려면

1. 터미널(Linux, macOS 또는 Unix) 또는 명령 프롬프트(Windows)를 열고 AWS CLI 를 사용하여 파이프라인의 이름, 전환을 활성화하려는 단계의 이름 및 전환 유형을 지정하여 <u>enable-stage-</u> transition 명령을 실행합니다.

예를 들어 *MyFirstPipeline*이라는 파이프라인에서 *Staging*이라는 스테이지로의 전환을 활성화하려면 다음과 유사한 명령을 입력합니다.

aws codepipeline enable-stage-transition --pipeline-name MyFirstPipeline --stagename Staging --transition-type Inbound

이 명령은 아무 것도 반환하지 않습니다.

 전환이 비활성화되었는지 확인하려면 CodePipeline 콘솔에서 파이프라인을 확인하거나 getpipeline-state 명령을 실행합니다. 자세한 내용은 <u>파이프라인 보기(콘솔)</u> 및 <u>파이프라인 세부 정보</u> 및 이력 보기(CLI) 단원을 참조하세요.

Monitoring pipelines

모니터링은 AWS CodePipeline의 안정성, 가용성 및 성능을 유지하는 데 중요한 부분입니다. 다중 지 점 장애가 발생할 경우 보다 쉽게 디버깅할 수 있도록 AWS 솔루션의 모든 부분에서 모니터링 데이터 를 수집해야 합니다. 모니터링을 시작하기 전에 다음 질문에 대해 답하는 모니터링 계획을 작성해야 합 니다.

- 모니터링의 목표
- 모니터링할 리소스
- 이러한 리소스를 모니터링하는 빈도
- 사용할 수 있는 모니터링 도구
- 모니터링 작업을 수행할 사람
- 문제 발생 시 알려야 할 대상

다음 도구를 사용하여 CodePipeline 파이프라인과 리소스를 모니터링할 수 있습니다.

- EventBridge 이벤트 버스 이벤트 EventBridge에서 CodePipeline 이벤트를 모니터링하여 파이프 라인, 단계 또는 작업 실행 상태의 변경을 감지할 수 있습니다. EventBridge는 해당 데이터를 AWS Lambda 및 Amazon Simple Notification Service와 같은 대상으로 라우팅합니다. EventBridge 이벤 트는 Amazon CloudWatch Events에 나타나는 이벤트와 동일합니다.
- 개발자 도구 콘솔의 파이프라인 이벤트 알림 콘솔에서 설정한 알림으로 CodePipeline 이벤트를 모 니터링한 다음 Amazon Simple Notification Service 주제 및 구독을 생성할 수 있습니다. 자세한 정보 는 개발자 도구 콘솔 사용 설명서의 알림이란 무엇입니까를 참조하세요.
- AWS CloudTrail CloudTrail을 사용하여 AWS 계정에서 CodePipeline을 대신하여 수행된 API 호출 을 캡처하고 로그 파일을 Amazon S3 버킷으로 전송합니다. 신속한 조치를 취하기 위해, 새 로그 파 일이 전송되면 CloudWatch에서 Amazon SNS 알림을 게시하게 할 수 있습니다.
- 콘솔과 CLI CodePipeline 콘솔과 CLI를 이용해 파이프라인이나 특정 파이프라인 실행의 상세 상태 정보를 볼 수 있습니다.

주제

- CodePipeline 이벤트 모니터링
- 이벤트 자리 표시자 버킷 참조
- 를 사용하여 CodePipeline API 호출 로깅 AWS CloudTrail

• CodePipeline CloudWatch 지표

CodePipeline 이벤트 모니터링

자체 애플리케이션, software-as-a-service(SaaS) 애플리케이션 및에서 실시간 데이터 스트림을 제공 하는 EventBridge에서 CodePipeline 이벤트를 모니터링할 수 있습니다 AWS 서비스. EventBridge는 해당 데이터를 AWS Lambda 및 Amazon Simple Notification Service와 같은 대상으로 라우팅합니다. 이러한 이벤트는 Amazon CloudWatch Events에 나타나는 이벤트와 동일하며, AWS 리소스의 변경 사 항을 설명하는 시스템 이벤트의 스트림을 거의 실시간 제공합니다. 자세한 내용은 EventBridge User Guide(EventBridge 사용 설명서)의 What Is EventBridge?(EventBridge란?)를 참조하세요.

Note

Amazon EventBridge는 이벤트를 관리하는 데 선호되는 방법입니다. Amazon CloudWatch Events와 EventBridge는 기본 서비스 및 API가 동일하지만 EventBridge가 더 많은 기능을 제 공합니다. CloudWatch Events 또는 EventBridge에서 변경한 내용은 각 콘솔에 나타납니다.

이벤트는 규칙으로 구성됩니다. 규칙은 다음을 선택하여 구성됩니다.

- 이벤트 패턴. 각 규칙은 모니터링할 이벤트의 소스 및 유형, 이벤트 대상이 포함된 이벤트 패턴으로 표현됩니다. 이벤트를 모니터링하려면 CodePipeline과 같이 모니터링 중인 서비스를 이벤트 소스로 사용하여 규칙을 생성합니다. 예를 들어, 파이프라인, 단계 또는 작업의 상태가 변경될 때 CodePipeline을 이벤트 소스로 사용하여 규칙을 트리거하는 이벤트 패턴이 포함된 규칙을 만들 수 있습니다.
- 대상. 새 규칙은 선택한 서비스를 이벤트 대상으로 수신합니다. 알림을 보내거나, 상태 정보를 캡처 하거나, 교정 작업을 수행하거나, 이벤트를 시작하거나, 기타 작업을 수행하도록 대상 서비스를 설 정할 수 있습니다. 대상을 추가할 때 EventBridge에 선택한 대상 서비스를 호출할 수 있는 권한을 EventBridge에 부여해야 합니다.

각 실행 상태 변경 이벤트 유형은 특정 메시지 내용이 포함된 알림을 내보냅니다. 여기에서:

- 첫 version 항목은 이벤트의 버전 번호를 보여 줍니다.
- 파이프라인 version 아래의 detail 항목은 파이프라인 구조 버전 번호를 보여 줍니다.

- 파이프라인 execution-id 아래의 detail 항목은 상태 변경을 발생시킨 파이프라인 실행의 실행 ID를 보여 줍니다. <u>AWS CodePipeline API 참조</u>의 GetPipelineExecution API 직접 호출을 참조하세 요.
- pipeline-execution-attempt 항목은 특정 실행 ID에 대한 시도 또는 재시도 횟수를 보여줍니다.

CodePipeline은 AWS 계정 의 리소스 상태가 변경될 때마다 EventBridge에 이벤트를 보고합니다. 다음 리소스에 대해 적어도 한 번은 보장된 방식으로 이벤트가 발생합니다.

- 파이프라인 실행
- 단계 실행
- 작업 실행

이벤트는 EventBridge에서 위에 설명된 이벤트 패턴 및 스키마를 사용하여 발생됩니다. 개발자 도구 콘솔에서 구성한 알림을 통해 받는 이벤트와 같이 처리된 이벤트의 경우 이벤트 메시지에는 일부 변형 이 있는 이벤트 패턴 필드가 포함됩니다. 예를 들어, detail-type 필드는 detailType으로 변환됩 니다. 자세한 내용은 <u>Amazon EventBridge API 참조</u>의 PutEvents API 직접 호출을 참조하세요.

다음은 CodePipeline 이벤트의 예제입니다. 가능한 경우, 각 예제는 처리된 이벤트에 대한 스키마와 함 께 생성된 이벤트의 스키마를 보여줍니다.

주제

- 세부 정보 유형
- 파이프라인 수준 이벤트
- 단계 수준 이벤트
- 작업 수준 이벤트
- 파이프라인 이벤트에 알림을 보내는 규칙 생성

세부 정보 유형

모니터링할 이벤트를 설정할 때 이벤트의 세부 유형을 선택할 수 있습니다.

다음의 상태가 변경될 때 보낼 알림을 구성할 수 있습니다.

• 지정된 파이프라인 또는 모든 파이프라인. "detail-type": "CodePipeline Pipeline Execution State Change"를 사용하여 이를 제어합니다.

- 지정된 파이프라인 또는 모든 파이프라인 내의 지정된 단계 또는 모든 단계. "detail-type": "CodePipeline Stage Execution State Change"를 사용하여 이를 제어합니다.
- 지정된 파이프라인 또는 모든 파이프라인 내의 지정된 단계 또는 모든 단계에 속한 지정된 작업 또는 모든 작업. "detail-type": "CodePipeline Action Execution State Change"를 사용 하여 이를 제어합니다.

Note

EventBridge에서 발생하는 이벤트는 이벤트가 처리될 때 detailType으로 변환되는 detail-type 파라미터를 포함합니다.

세부 정보 유형	State	설명
CodePipeline 파 이프라인 실행 상	CANCELED	파이프라인 구조가 업데이트되지 않아 파이프라인 실행이 취소되었습니다.
태면경	FAILED	파이프라인 실행이 성공적으로 완료되지 않았습니다.
	RESUMED	실패한 파이프라인 실행이 RetryStageExecution API 호출에 대한 응답으로 다시 시도되었습니다.
	STARTED	파이프라인 실행이 현재 실행 중입니다.
	STOPPED	중지 중인 프로세스가 완료되고 파이프라인 실행이 중지됩 니다.
	중지 중 (STOPPING)	파이프라인 실행을 중지하고 대기하거나 중지하고 중단하 라는 요청으로 인해 파이프라인 실행이 중지되는 중입니 다.
	SUCCEEDED	파이프라인 실행이 성공적으로 완료되었습니다.
	SUPERSEDED	이 파이프라인 실행이 다음 단계가 완료되기를 기다리는 동안 새로운 파이프라인 실행이 진행되고 해당 파이프라인 에서 계속되었습니다.

세부 정보 유형	State	설명	
CodePipeline 단 계 실행 상태 변	CANCELED	파이프라인 구조가 업데이트되지 않아 단계가 취소되었습 니다.	
3	FAILED	단계가 성공적으로 완료되지 않았습니다.	
	RESUMED	실패한 단계가 RetryStageExecution API 호출에 대 한 응답으로 다시 시도되었습니다.	
	STARTED	단계가 현재 실행 중입니다.	
	STOPPED	중지 중인 프로세스가 완료되고 단계 실행이 중지됩니다.	
	중지 중 (STOPPING)	파이프라인 실행을 중지하고 대기하거나 중지하고 중단하 라는 요청으로 인해 단계 실행이 중지되는 중입니다.	
	SUCCEEDED	단계가 성공적으로 완료되었습니다.	
CodePipeline 작 업 실행 상태 변	ABANDONED	파이프라인 실행을 중지하고 중단하라는 요청으로 인해 작 업이 중단됩니다.	
3	CANCELED	파이프라인 구조가 업데이트되지 않아 작업이 취소되었습 니다.	
	FAILED	승인 작업의 경우 FAILED 상태는 검토자가 작업을 거부했 거나 잘못된 작업 구성으로 인해 작업이 실패했음을 의미 합니다.	
	STARTED	작업이 현재 실행 중입니다.	
	SUCCEEDED	작업이 성공적으로 완료되었습니다.	

파이프라인 수준 이벤트

파이프라인 수준 이벤트는 파이프라인 실행의 상태가 변경될 때 발생합니다.

주제

• <u>파이프라인 STARTED 이벤트</u>

- 파이프라인 STOPPING 이벤트
- 파이프라인 SUCCEEDED 이벤트
- 파이프라인 SUCCEEDED(Git 태그를 사용한 예제)
- 파이프라인 FAILED 이벤트
- 파이프라인 FAILED(Git 태그를 사용한 예제)

파이프라인 STARTED 이벤트

파이프라인 실행이 시작되면 다음 내용이 포함된 알림을 보내는 이벤트를 내보냅니다. 이는 useast-1 리전의 "myPipeline"이라는 파이프라인에 대한 예제입니다. id 필드는 이벤트 ID를 나타 내고 account 필드는 파이프라인이 생성된 계정 ID를 나타냅니다.

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Pipeline Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-24T22:03:07Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:codepipeline:us-east-1:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "execution-trigger": {
            "trigger-type": "StartPipelineExecution",
            "trigger-detail": "arn:aws:sts::123456789012:assumed-role/Admin/my-user"
        },
        "state": "STARTED",
        "version": 1.0,
        "pipeline-execution-attempt": 1.0
   }
}
```

```
{
    "account": "123456789012",
    "detailType": "CodePipeline Pipeline Execution State Change",
    "region": "us-east-1",
    "source": "aws.codepipeline",
    "time": "2021-06-24T00:44:50Z",
    "notificationRuleArn": "arn:aws:codestar-notifications:us-
east-1:123456789012:notificationrule/a69c62c21EXAMPLE",
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "execution-trigger": {
            "trigger-type": "StartPipelineExecution",
            "trigger-detail": "arn:aws:sts::123456789012:assumed-role/Admin/my-user"
        },
        "state": "STARTED",
        "version": 1.0,
        "pipeline-execution-attempt": 1.0
    },
    "resources": [
        "arn:aws:codepipeline:us-east-1:123456789012:myPipeline"
    ],
    "additionalAttributes": {}
}
```

파이프라인 STOPPING 이벤트

파이프라인 실행이 중지되면 다음 내용이 포함된 알림을 보내는 이벤트를 내보냅니다. 이는 uswest-2 리전의 myPipeline이라는 파이프라인에 대한 예제입니다.

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Pipeline Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-24T22:02:20Z",
    "region": "us-west-2",
    "resources": [
```

```
"arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
],
"detail": {
    "pipeline": "myPipeline",
    "execution-id": "12345678-1234-5678-abcd-12345678abcd",
    "start-time": "2023-10-26T13:49:39.208Z",
    "state": "STOPPING",
    "version": 3.0,
    "pipeline-execution-attempt": 1.0
    "stop-execution-comments": "Stopping the pipeline for an update"
}
```

파이프라인 SUCCEEDED 이벤트

파이프라인 실행이 성공하면 다음 내용이 포함된 알림을 보내는 이벤트를 내보냅니다. 이는 useast-1 리전의 myPipeline이라는 파이프라인에 대한 예제입니다.

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Pipeline Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-24T22:03:44Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:codepipeline:us-east-1:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "state": "SUCCEEDED",
        "version": 3.0,
        "pipeline-execution-attempt": 1.0
    }
}
```

```
{
    "account": "123456789012",
    "detailType": "CodePipeline Pipeline Execution State Change",
    "region": "us-east-1",
    "source": "aws.codepipeline",
    "time": "2021-06-30T22:13:51Z",
    "notificationRuleArn": "arn:aws:codestar-notifications:us-
west-2:123456789012:notificationrule/a69c62c21EXAMPLE",
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "state": "SUCCEEDED",
        "version": 1.0,
        "pipeline-execution-attempt": 1.0
    },
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "additionalAttributes": {}
}
```

파이프라인 SUCCEEDED(Git 태그를 사용한 예제)

파이프라인 실행에 재시도하여 성공한 단계가 있으면 다음 내용이 포함된 알림을 보내는 이벤트를 내보냅니다. 이 예제는 execution-trigger가 Git 태그에 대해 구성된 eu-central-1 리전에서 myPipeline이라는 파이프라인에 대한 것입니다.

Note

execution-trigger 필드에는 파이프라인을 트리거한 이벤트의 종류에 따라 tag-name 또 는 branch-name 둘 중 하나가 있습니다.

```
{
    "version": "0",
    "id": "b128b002-09fd-4574-4eba-27152726c777",
    "detail-type": "CodePipeline Pipeline Execution State Change",
    "source": "aws.codepipeline",
```

```
"account": "123456789012",
    "time": "2023-10-26T13:50:53Z",
    "region": "eu-central-1",
    "resources": [
        "arn:aws:codepipeline:eu-central-1:123456789012:BuildFromTag"
    ],
    "detail": {
        "pipeline": "BuildFromTag",
        "execution-id": "e17b5773-cc0d-4db2-9ad7-594c73888de8",
        "start-time": "2023-10-26T13:49:39.208Z",
        "execution-trigger": {
            "author-display-name": "Mary Major",
            "full-repository-name": "mmajor/sample-project",
            "provider-type": "GitLab",
            "author-email": "email_address",
            "commit-message": "Update file README.md",
            "author-date": "2023-08-16T21:08:08Z",
            "tag-name": "gitlab-v4.2.1",
            "commit-id": "commit_ID",
            "connection-arn": "arn:aws:codestar-connections:eu-
central-1:123456789012:connection/0f5b706a-1a1d-46c5-86b6-f177321bcfb2",
            "author-id": "Mary Major"
        },
        "state": "SUCCEEDED",
        "version": 32.0,
        "pipeline-execution-attempt": 1.0
    }
}
```

파이프라인 FAILED 이벤트

파이프라인 실행이 실패하면 다음 내용이 포함된 알림을 보내는 이벤트를 내보냅니다. 이는 uswest-2 리전의 "myPipeline"이라는 파이프라인에 대한 예제입니다.

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Pipeline Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-31T18:55:432",
```

```
"region": "us-west-2",
"resources": [
    "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
],
"detail": {
    "pipeline": "myPipeline",
    "execution-id": "12345678-1234-5678-abcd-12345678abcd",
    "start-time": "2023-10-26T13:49:39.208Z",
    "state": "FAILED",
    "version": 4.0,
    "pipeline-execution-attempt": 1.0
}
```

{
"account": "123456789012",
"detailType": "CodePipeline Pipeline Execution State Change",
"region": "us-west-2",
"source": "aws.codepipeline",
"time": "2021-06-24T00:46:16Z",
"notificationRuleArn": "arn:aws:codestar-notifications:us-
<pre>west-2:123456789012:notificationrule/a69c62c21EXAMPLE",</pre>
"detail": {
"pipeline": "myPipeline",
"execution-id": "12345678-1234-5678-abcd-12345678abcd",
"start-time": "2023-10-26T13:49:39.208Z",
"state": "FAILED",
"version": 1.0,
"pipeline-execution-attempt": 1.0
},
"resources": [
"arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
],
"additionalAttributes": {
"failedActionCount": 1,
"failedActions": [
{
"action": "Deploy",
"additionalInformation": "Deployment <id> failed"</id>
}
1.

}

파이프라인 FAILED(Git 태그를 사용한 예제)

"failedStage": "Deploy"

소스 단계에서 실패하지 않는 한 트리거가 포함된 파이프라인 구성의 경우 다음 내용이 포함된 알 림을 보내는 이벤트가 발생합니다. 이 예제는 execution-trigger가 Git 태그에 대해 구성된 eucentral-1 리전에서 myPipeline이라는 파이프라인에 대한 것입니다.

Note

execution-trigger 필드에는 파이프라인을 트리거한 이벤트의 종류에 따라 tag-name 또 는 branch-name 둘 중 하나가 있습니다.

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Pipeline Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-31T18:55:43Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "execution-trigger": {
            "author-display-name": "Mary Major",
            "full-repository-name": "mmajor/sample-project",
            "provider-type": "GitLab",
            "author-email": "email_address",
            "commit-message": "Update file README.md",
            "author-date": "2023-08-16T21:08:08Z",
            "tag-name": "gitlab-v4.2.1",
            "commit-id": "commit_ID",
```

```
"connection-arn": "arn:aws:codestar-connections:eu-
central-1:123456789012:connection/0f5b706a-1a1d-46c5-86b6-f177321bcfb2",
        "author-id": "Mary Major"
    },
    "state": "FAILED",
        "version": 4.0,
        "pipeline-execution-attempt": 1.0
    }
}
```

```
{
    "account": "123456789012",
    "detailType": "CodePipeline Pipeline Execution State Change",
    "region": "us-west-2",
    "source": "aws.codepipeline",
    "time": "2021-06-24T00:46:16Z",
    "notificationRuleArn": "arn:aws:codestar-notifications:us-
west-2:123456789012:notificationrule/a69c62c21EXAMPLE",
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "execution-trigger": {
            "author-display-name": "Mary Major",
            "full-repository-name": "mmajor/sample-project",
            "provider-type": "GitLab",
            "author-email": "email_address",
            "commit-message": "Update file README.md",
            "author-date": "2023-08-16T21:08:08Z",
            "tag-name": "gitlab-v4.2.1",
            "commit-id": "commit_ID",
            "connection-arn": "arn:aws:codestar-connections:eu-
central-1:123456789012:connection/0f5b706a-1a1d-46c5-86b6-f177321bcfb2",
            "author-id": "Mary Major"
        },
        "state": "FAILED",
        "version": 1.0,
        "pipeline-execution-attempt": 1.0
    },
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
```

```
],
"additionalAttributes": {
    "failedActionCount": 1,
    "failedActions": [
        {
            "action": "Deploy",
            "additionalInformation": "Deployment <ID> failed"
        }
    ],
    "failedStage": "Deploy"
}
```

단계 수준 이벤트

단계 수준 이벤트는 단계 실행의 상태가 변경될 때 발생합니다.

주제

- <u>단계 STARTED 이벤트</u>
- <u>단계 STOPPING 이벤트</u>
- 단계 STOPPED 이벤트
- 단계 재시도 이벤트 이후 Stage RESUMED

단계 STARTED 이벤트

단계 실행이 시작되면 다음 내용이 포함된 알림을 보내는 이벤트를 내보냅니다. 이는 us-east-1 리 전의 "myPipeline"이라는 파이프라인의 단계 Prod에 대한 예제입니다.

```
{
    "version": "0",
    "id": 01234567-EXAMPLE,
    "detail-type": "CodePipeline Stage Execution State Change",
    "source": "aws.codepipeline",
    "account": 123456789012,
    "time": "2020-01-24T22:03:07Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:codepipeline:us-east-1:123456789012:myPipeline"
```

```
],
    "detail": {
        "pipeline": "myPipeline",
        "version": 1.0,
        "execution-id": 12345678-1234-5678-abcd-12345678abcd,
        "start-time": "2023-10-26T13:49:39.208Z",
        "stage": "Prod",
        "state": "STARTED",
        "pipeline-execution-attempt": 1.0
    }
}
```

```
{
    "account": "123456789012",
    "detailType": "CodePipeline Stage Execution State Change",
    "region": "us-east-1",
    "source": "aws.codepipeline",
    "time": "2021-06-24T00:45:40Z",
    "notificationRuleArn": "arn:aws:codestar-notifications:us-
west-2:123456789012:notificationrule/a69c62c21EXAMPLE",
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "stage": "Source",
        "state": "STARTED",
        "version": 1.0,
        "pipeline-execution-attempt": 0.0
    },
    "resources": [
        "arn:aws:codepipeline:us-east-1:123456789012:myPipeline"
    ],
    "additionalAttributes": {
        "sourceActions": [
            {
                "sourceActionName": "Source",
                "sourceActionProvider": "CodeCommit",
                "sourceActionVariables": {
                    "BranchName": "main",
                    "CommitId": "<ID>",
                    "RepositoryName": "my-repo"
```



단계 STOPPING 이벤트

단계 실행이 중지되면 다음 내용이 포함된 알림을 보내는 이벤트를 내보냅니다. 이는 us-west-2 리 전의 myPipeline이라는 파이프라인의 단계 Deploy에 대한 예제입니다.

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Stage Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-24T22:02:20Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "stage": "Deploy",
        "state": "STOPPING",
        "version": 3.0,
        "pipeline-execution-attempt": 1.0
    }
}
```

단계 STOPPED 이벤트

단계 실행이 중단되면 다음 내용이 포함된 알림을 보내는 이벤트를 내보냅니다. 이는 us-west-2 리 전의 myPipeline이라는 파이프라인의 단계 Deploy에 대한 예제입니다.

```
"version": "0",
"id": "01234567-EXAMPLE",
```

{

```
"detail-type": "CodePipeline Stage Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-31T18:21:39Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "stage": "Deploy",
        "state": "STOPPED",
        "version": 3.0,
        "pipeline-execution-attempt": 1.0
    }
}
```

단계 재시도 이벤트 이후 Stage RESUMED

단계 실행이 재개되고 재시도된 단계가 있으면 다음 내용이 포함된 알림을 보내는 이벤트를 내보냅니 다.

단계가 재시도되면 예제와 같이 stage-last-retry-attempt-time 필드가 표시됩니다. 재시도가 수행된 경우 모든 단계 이벤트에 필드가 표시됩니다.

Note

단계가 재시도되면 모든 후속 단계 이벤트에 stage-last-retry-attempt-time 필드가 표시됩니다.

```
{
    "version": "0",
    "id": "38656bcd-a798-5f92-c738-02a71be484e1",
    "detail-type": "CodePipeline Stage Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2023-10-26T14:14:56Z",
    "region": "eu-central-1",
    "resources": [
```

```
"arn:aws:codepipeline:eu-central-1:123456789012:BuildFromTag"
],
"detail": {
    "pipeline": "BuildFromTag",
    "execution-id": "05dafb6a-5a56-4951-a858-968795364846",
    "stage-last-retry-attempt-time": "2023-10-26T14:14:56.305Z",
    "stage": "Build",
    "state": "RESUMED",
    "version": 32.0,
    "pipeline-execution-attempt": 2.0
}
```

작업 수준 이벤트

작업 수준 이벤트는 작업 실행의 상태가 변경될 때 발생합니다.

주제

- <u>작업 STARTED 이벤트</u>
- <u>작업 SUCCEEDED 이벤트</u>
- 작업 FAILED 이벤트
- 작업 ABANDONED 이벤트

작업 STARTED 이벤트

작업 실행이 시작되면 다음 내용이 포함된 알림을 보내는 이벤트를 내보냅니다. 이는 us-east-1 리 전의 myPipeline이라는 파이프라인의 배포 작업 myAction에 대한 예제입니다.

```
{
    "version": "0",
    "id": 01234567-EXAMPLE,
    "detail-type": "CodePipeline Action Execution State Change",
    "source": "aws.codepipeline",
    "account": 123456789012,
    "time": "2020-01-24T22:03:07Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:codepipeline:us-east-1:123456789012:myPipeline"
```

```
],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": 12345678-1234-5678-abcd-12345678abcd,
        "start-time": "2023-10-26T13:51:09.981Z",
        "stage": "Prod",
        "action-execution-id": "47f821c5-a902-44b2-ae61-b878d31ecd21",
        "action": "myAction",
        "state": "STARTED",
        "type": {
            "owner": "AWS",
            "category": "Deploy",
            "provider": "CodeDeploy",
            "version": "1"
        },
        "version": 2.0
        "pipeline-execution-attempt": 1.0
        "input-artifacts": [
            {
                "name": "SourceArtifact",
                "s3location": {
                    "bucket": "codepipeline-us-east-1-BUCKETEXAMPLE",
                    "key": "myPipeline/SourceArti/KEYEXAMPLE"
                }
            }
        ]
    }
}
```

```
{
    "account": "123456789012",
    "detailType": "CodePipeline Action Execution State Change",
    "region": "us-west-2",
    "source": "aws.codepipeline",
    "time": "2021-06-24T00:45:44Z",
    "notificationRuleArn": "arn:aws:codestar-notifications:us-
west-2:123456789012:notificationrule/a69c62c21EXAMPLE",
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:51:09.9812",
    }
}
```

```
"stage": "Deploy",
        "action-execution-id": "47f821c5-a902-44b2-ae61-b878d31ecd21",
        "action": "Deploy",
        "input-artifacts": [
            {
                "name": "SourceArtifact",
                "s3location": {
                    "bucket": "codepipeline-us-east-1-EXAMPLE",
                    "key": "myPipeline/SourceArti/EXAMPLE"
                }
            }
        ],
        "state": "STARTED",
        "region": "us-east-1",
        "type": {
            "owner": "AWS",
            "provider": "CodeDeploy",
            "category": "Deploy",
            "version": "1"
        },
        "version": 1.0,
        "pipeline-execution-attempt": 1.0
    },
    "resources": [
        "arn:aws:codepipeline:us-east-1:123456789012:myPipeline"
    ],
    "additionalAttributes": {}
}
```

작업 SUCCEEDED 이벤트

작업 실행이 성공하면 다음 내용이 포함된 알림을 보내는 이벤트를 내보냅니다. 이는 us-west-2 리 전의 "myPipeline"이라는 파이프라인의 소스 작업 "Source"에 대한 예제입니다. 이 이벤트 유형 에는 두 개의 다른 region 필드가 있습니다. 이벤트 region 필드는 파이프라인 이벤트의 리전을 지 정합니다. detail 섹션 아래의 region 필드는 작업의 리전을 지정합니다.

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Action Execution State Change",
```

```
"source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-24T22:03:11Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:51:09.981Z",
        "stage": "Source",
        "execution-result": {
            "external-execution-url": "https://us-west-2.console.aws.amazon.com/
codecommit/home#/repository/my-repo/commit/8cf40f2EXAMPLE",
            "external-execution-summary": "Added LICENSE.txt",
            "external-execution-id": "8cf40fEXAMPLE"
        },
        "output-artifacts": [
            {
                "name": "SourceArtifact",
                "s3location": {
                    "bucket": "codepipeline-us-west-2-BUCKETEXAMPLE",
                    "key": "myPipeline/SourceArti/KEYEXAMPLE"
                }
            }
        ],
        "action-execution-id": "47f821c5-a902-44b2-ae61-b878d31ecd21",
        "action": "Source",
        "state": "SUCCEEDED",
        "region": "us-west-2",
        "type": {
            "owner": "AWS",
            "provider": "CodeCommit",
            "category": "Source",
            "version": "1"
        },
        "version": 3.0,
        "pipeline-execution-attempt": 1.0
    }
}
```

```
{
    "account": "123456789012",
    "detailType": "CodePipeline Action Execution State Change",
    "region": "us-west-2",
    "source": "aws.codepipeline",
    "time": "2021-06-24T00:45:44Z",
    "notificationRuleArn": "arn:aws:codestar-notifications:us-
west-2:ACCOUNT:notificationrule/a69c62c21EXAMPLE",
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "arn:aws:codepipeline:us-west-2:123456789012:myPipeline",
        "start-time": "2023-10-26T13:51:09.981Z",
        "stage": "Source",
        "execution-result": {
            "external-execution-url": "https://us-west-2.console.aws.amazon.com/
codecommit/home#/repository/my-repo/commit/8cf40f2EXAMPLE",
            "external-execution-summary": "Edited index.html",
            "external-execution-id": "36ab3ab7EXAMPLE"
        },
        "output-artifacts": [
            {
                "name": "SourceArtifact",
                "s3location": {
                    "bucket": "codepipeline-us-west-2-EXAMPLE",
                    "key": "myPipeline/SourceArti/EXAMPLE"
                }
            }
        ],
        "action-execution-id": "47f821c5-a902-44b2-ae61-b878d31ecd21",
        "action": "Source",
        "state": "SUCCEEDED",
        "region": "us-west-2",
        "type": {
            "owner": "AWS",
            "provider": "CodeCommit",
            "category": "Source",
            "version": "1"
        },
        "version": 1.0,
        "pipeline-execution-attempt": 1.0
    },
    "resources": [
```

```
"arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
],
"additionalAttributes": {}
}
```

작업 FAILED 이벤트

작업 실행이 실패하면 다음 내용이 포함된 알림을 보내는 이벤트를 내보냅니다. 이는 us-west-2 리 전의 "myPipeline"이라는 파이프라인의 작업 "Deploy"에 대한 예제입니다.

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Action Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-31T18:55:43Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:51:09.981Z",
        "stage": "Deploy",
        "execution-result": {
            "external-execution-url": "https://us-west-2.console.aws.amazon.com/
codedeploy/home?#/deployments/<ID>",
            "external-execution-summary": "Deployment <ID> failed",
            "external-execution-id": "<ID>",
            "error-code": "JobFailed"
        },
        "action-execution-id": "47f821c5-a902-44b2-ae61-b878d31ecd21",
        "action": "Deploy",
        "state": "FAILED",
        "region": "us-west-2",
        "type": {
            "owner": "AWS",
            "provider": "CodeDeploy",
```

```
"category": "Deploy",
    "version": "1"
    },
    "version": 4.0,
    "pipeline-execution-attempt": 1.0
  }
}
```

```
{
    "account": "123456789012",
    "detailType": "CodePipeline Action Execution State Change",
    "region": "us-west-2",
    "source": "aws.codepipeline",
    "time": "2021-06-24T00:46:16Z",
    "notificationRuleArn": "arn:aws:codestar-notifications:us-
west-2:123456789012:notificationrule/a69c62c21EXAMPLE",
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "stage": "Deploy",
        "execution-result": {
            "external-execution-url": "https://console.aws.amazon.com/codedeploy/
home?region=us-west-2#/deployments/<ID>",
            "external-execution-summary": "Deployment <ID> failed",
            "external-execution-id": "<ID>",
            "error-code": "JobFailed"
        },
        "action-execution-id": "47f821c5-a902-44b2-ae61-b878d31ecd21",
        "action": "Deploy",
        "state": "FAILED",
        "region": "us-west-2",
        "type": {
            "owner": "AWS",
            "provider": "CodeDeploy",
            "category": "Deploy",
            "version": "1"
        },
        "version": 13.0,
        "pipeline-execution-attempt": 1.0
    },
    "resources": [
```

```
"arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
],
"additionalAttributes": {
    "additionalInformation": "Deployment <ID> failed"
}
}
```

작업 ABANDONED 이벤트

작업 실행이 중단되면 다음 내용이 포함된 알림을 보내는 이벤트를 내보냅니다. 이는 us-west-2 리 전의 "myPipeline"이라는 파이프라인의 작업 "Deploy"에 대한 예제입니다.

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Action Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-31T18:21:39Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "stage": "Deploy",
        "action-execution-id": "47f821c5-a902-44b2-ae61-b878d31ecd21",
        "action": "Deploy",
        "state": "ABANDONED",
        "region": "us-west-2",
        "type": {
            "owner": "AWS",
            "provider": "CodeDeploy",
            "category": "Deploy",
            "version": "1"
        },
        "version": 3.0,
        "pipeline-execution-attempt": 1.0
    }
}
```

파이프라인 이벤트에 알림을 보내는 규칙 생성

규칙은 특정 이벤트를 감시한 다음 선택한 AWS 대상으로 라우팅합니다. 다른 AWS 작업이 발생할 때 AWS 작업을 자동으로 수행하는 규칙 또는 설정된 일정에 따라 AWS 작업을 정기적으로 수행하는 규 칙을 생성할 수 있습니다.

주제

- 파이프라인 상태 변경 시 알림 보내기(콘솔)
- 파이프라인 상태 변경 시 알림 보내기(CLI)

파이프라인 상태 변경 시 알림 보내기(콘솔)

다음 단계에서는 EventBridge 콘솔을 사용하여 CodePipeline의 변경에 대한 알림을 보내는 규칙을 만 드는 방법을 보여 줍니다.

Amazon S3 소스가 포함된 파이프라인을 대상으로 지정하는 EventBridge 규칙을 생성하려면

- 1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
- 탐색 창에서 규칙을 선택합니다. 기본 버스를 선택된 상태로 두거나 이벤트 버스를 선택하세요. 규 칙 생성을 선택합니다.
- 3. 이름에 역할의 이름을 입력합니다.
- 4. 규칙 유형에서 이벤트 패턴이 있는 규칙을 선택합니다. 다음을 선택합니다.
- 5. 이벤트 패턴에서 AWS 서비스를 선택합니다.
- 6. [Event Type] 드롭다운 목록에서 알림의 상태 변경 수준을 선택합니다.
 - 파이프라인 수준 이벤트에 적용되는 규칙의 경우 [CodePipeline Pipeline Execution State Change]를 선택합니다.
 - 상태 수준 이벤트에 적용되는 규칙의 경우 [CodePipeline Stage Execution State Change]를 선 택합니다.
 - 작업 수준 이벤트에 적용되는 규칙의 경우 [CodePipeline Action Execution State Change]를 선 택합니다.
- 7. 규칙을 적용할 상태 변경을 지정합니다.
 - 모든 상태 변경에 적용되는 규칙의 경우 [Any state]를 선택합니다.
 - 일부 상태 변경에만 적용되는 규칙의 경우 [Specific state(s)]를 선택한 후 목록에서 상태 값을 한 개 이상 선택합니다.

 선택기에서 허용하는 것보다 더 자세한 이벤트 패턴의 경우 이벤트 패턴 창의 패턴 편집 옵션을 사 용하여 이벤트 패턴을 JSON 형식으로 지정할 수도 있습니다.

1 Note

달리 지정하지 않으면 모든 파이프라인/단계/작업 및 상태에 대한 이벤트 패턴이 생성됩니 다.

더 세부적인 이벤트 패턴의 경우 다음 이벤트 패턴 예제를 복사하여 이벤트 패턴 창에 붙여넣으면 됩니다.

• Example

다음 이벤트 패턴 샘플을 사용하면 모든 파이프라인에서 실패한 배포 및 빌드 작업을 캡처합니 다.

```
{
"source": [
    "aws.codepipeline"
 ],
 "detail-type": [
    "CodePipeline Action Execution State Change"
 ],
  "detail": {
    "state": [
      "FAILED"
    ],
    "type": {
      "category": ["Deploy", "Build"]
    }
  }
}
```

• Example

다음 이벤트 패턴 샘플을 사용하면 모든 파이프라인에서 거부되었거나 실패한 모든 승인 작업 을 캡처합니다.

```
"aws.codepipeline"
],
"detail-type": [
   "CodePipeline Action Execution State Change"
],
"detail": {
   "state": [
    "FAILED"
   ],
   "type": {
      "category": ["Approval"]
   }
}
```

• Example

다음 이벤트 패턴 샘플을 사용하면 지정된 파이프라인에서 모든 이벤트를 캡처합니다.

```
{
"source": [
    "aws.codepipeline"
],
    "detail-type": [
    "CodePipeline Pipeline Execution State Change",
    "CodePipeline Action Execution State Change",
    "CodePipeline Stage Execution State Change"
],
    "detail": {
        "pipeline": ["myPipeline", "my2ndPipeline"]
    }
}
```

- 9. 다음을 선택합니다.
- 10. 대상 유형에서 AWS 서비스를 선택합니다.
- 11. 대상 선택에서 CodePipeline을 선택합니다. Pipeline ARN에서 이 규칙에 의해 시작되는 파이프라 인의 파이프라인 ARN을 입력합니다.

Note

파이프라인 ARN을 확인하려면 get-pipeline 명령을 실행합니다. 출력에 파이프라인 ARN 이 나타납니다. ARN의 형식은 다음과 같습니다.
arn:aws:codepipeline:*region:account:pipeline-name* 파이프라인 ARN 샘플: arn:aws:codepipeline:us-east-2:80398EXAMPLE:MyFirstPipeline

- 12. EventBridge에 EventBridge 규칙과 연결된 대상을 호출할 권한을 부여하는 IAM 서비스 역할을 생성하거나 지정하려면(이 경우에는 대상이 CodePipeline):
 - 이 특정 리소스에 대한 새 역할 생성을 선택하여 EventBridge에 파이프라인 실행을 시작할 권한 을 제공하는 서비스 역할을 생성합니다.
 - 기존 역할 사용을 선택하여 EventBridge에 파이프라인 실행을 시작할 권한을 제공하는 서비스 역할을 입력합니다.
- 13. 다음을 선택합니다.
- 14. 태그 페이지에서 다음을 선택합니다.
- 15. 검토 및 생성 페이지에서 규칙 구성을 검토합니다. 규칙이 만족스러우면 규칙 생성(Create rule)을 선택하세요.

파이프라인 상태 변경 시 알림 보내기(CLI)

다음 단계에서는 CLI를 사용하여 CodePipeline의 변경에 대한 알림을 보내는 CloudWatch Events 규 칙을 만드는 방법을 보여 줍니다.

AWS CLI 를 사용하여 규칙을 생성하려면 다음을 지정하여 put-rule 명령을 호출합니다.

- 만들려는 규칙을 고유하게 식별하는 이름. 이 이름은 AWS 계정과 관련해 CodePipeline에서 만드는 모든 파이프라인에서 고유해야 합니다.
- 소스의 이벤트 패턴 및 규칙에서 사용하는 세부 정보 필드. 자세한 내용은 <u>Amazon EventBridge 및</u> <u>이벤트 패턴</u>을 참조하세요.

CodePipeline이 이벤트 소스인 EventBridge 규칙을 만들려면

1. put-rule 명령을 호출하여 이벤트 패턴을 지정하는 규칙을 만듭니다. (유효한 상태는 앞의 표 참조)

다음 명령 샘플은 --event-pattern을 사용하여 "myPipeline"이라는 파이프라인에 대한 파이프라인 실행이 실패하면 CloudWatch 이벤트를 내보내는 "MyPipelineStateChanges"라는 규칙을 만 듭니다.

```
aws events put-rule --name "MyPipelineStateChanges" --event-pattern "{\"source\":
[\"aws.codepipeline\"],\"detail-type\":[\"CodePipeline Pipeline Execution State
Change\"],\"detail\":{\"pipeline\":[\"myPipeline\"],\"state\":[\"FAILED\"]}}"
```

- 2. put-targets 명령을 호출하고 다음 파라미터를 포함합니다.
 - --rule 파라미터는 put-rule을 사용하여 생성한 rule_name에 사용됩니다.
 - --targets 파라미터는 대상 목록에 있는 대상의 목록 Id 및 Amazon SNS 주제의 ARN에 사용 됩니다.

다음 예제 명령은 MyPipelineStateChanges이라는 규칙에 대해 대상 Id가 숫자 1로 구성됨을 지정하며, 규칙의 대상 목록에서 1로 대상 1로 표시됩니다. 이 예제 명령은 또한 Amazon SNS 주 제에 대한 예제 ARN을 지정합니다.

aws events put-targets --rule MyPipelineStateChanges --targets Id=1,Arn=arn:aws:sns:us-west-2:11111EXAMPLE:MyNotificationTopic

3. 지정된 대상 서비스를 사용하여 알림을 호출할 수 있도록 EventBridge에 권한을 추가합니다. 자세 한 내용은 Amazon EventBridge에 대한 리소스 기반 정책 사용을 참조하세요.

이벤트 자리 표시자 버킷 참조

이 단원은 참조용입니다. 이벤트 감지 리소스가 있는 파이프라인 생성에 대한 자세한 내용은 <u>소스 작업</u> 및 변경 감지 방법 단원을 참조하십시오.

Amazon S3 및 CodeCommit에서 제공하는 소스 작업은 소스 버킷 또는 리포지토리에서 변경이 이 루어질 때 이벤트 기반 변경 감지 리소스를 사용하여 파이프라인을 트리거합니다. 이러한 리소스는 CodeCommit 리포지토리에 대한 코드 변경과 같이 파이프라인 소스의 이벤트에 응답하도록 구성 된 CloudWatch Events 규칙입니다. Amazon S3 소스에 대해 CloudWatch Events를 사용하는 경우 CloudTrail을 켜야 이벤트가 로깅됩니다. CloudTrail에는 해당 다이제스트를 전송할 수 있는 S3 버킷이 필요합니다. 사용자 지정 버킷에서 CloudWatch Events 리소스의 로그 파일에 액세스할 수 있지만 자 리 표시자 버킷의 데이터에는 액세스할 수 없습니다.

• CLI 또는를 사용하여 CloudWatch Events 리소스를 AWS CloudFormation 설정한 경우 파이프라인 을 설정할 때 지정한 버킷에서 CloudTrail 파일을 찾을 수 있습니다. 콘솔을 사용하여 S3 소스로 파이프라인을 설정한 경우, 콘솔은 CloudWatch Events 리소스를 생성 할 때 CloudTrail 자리 표시자 버킷을 사용합니다. CloudTrail 다이제스트는 AWS 리전 파이프라인이 생성된의 자리 표시자 버킷에 저장됩니다.

자리 표시자 버킷 이외의 버킷을 사용하려는 경우 구성을 변경할 수 있습니다.

Note

CloudTrail 자리 표시자 버킷에 기록된 데이터는 하루 후에 자동으로 만료되며 보존되지 않습 니다.

CloudTrail 로그 파일 조사 결과 및 관리에 대한 자세한 내용은 <u>CloudTrail 로그 파일 가져오기 및 보</u> <u>기</u>를 참조하세요.

주제

• <u>리전별 이벤트 자리 표시자 버킷 이름</u>

리전별 이벤트 자리 표시자 버킷 이름

이 표에는 Amazon S3 소스 작업이 있는 파이프라인에 대한 변경 감지 이벤트를 추적하는 로그 파일이 들어 있는 S3 자리 표시자 버킷의 이름이 나와 있습니다.

리전 이름	자리 표시자 버킷 이름	리전 식별자
미국 동부(오하이오)	codepipeline-cloudtrail-pla ceholder-bucket-us-east-2	us-east-2
미국 동부(버지니아 북부)	codepipeline-cloudtrail-pla ceholder-bucket-us-east-1	us-east-1
미국 서부(캘리포니아 북부)	codepipeline-cloudtrail-pla ceholder-bucket-us-west-1	us-west-1
미국 서부(오리건)	codepipeline-cloudtrail-pla ceholder-bucket-us-west-2	us-west-2

AWS CodePipeline

리전 이름	자리 표시자 버킷 이름	리전 식별자
캐나다(중부)	codepipeline-cloudtrail-pla ceholder-bucket-ca-central-1	ca-central-1
유럽(프랑크푸르트)	codepipeline-cloudtrail-pla ceholder-bucket-eu-central-1	eu-central-1
유럽(아일랜드)	codepipeline-cloudtrail-pla ceholder-bucket-eu-west-1	eu-west-1
유럽(런던)	codepipeline-cloudtrail-pla ceholder-bucket-eu-west-2	eu-west-2
유럽(파리)	codepipeline-cloudtrail-pla ceholder-bucket-eu-west-3	eu-west-3
유럽(스톡홀름)	codepipeline-cloudtrail-pla ceholder-bucket-eu-north-1	eu-north-1
아시아 태평양(홍콩)	codepipeline-cloudtrail-pla ceholder-bucket-ap-east-1	ap-east-1
아시아 태평양(하이데라바드)	codepipeline-cloudtrail-pla ceholder-bucket-ap-south-2	ap-south-2
아시아 태평양(자카르타)	codepipeline-cloudtrail-pla ceholder-bucket-ap-southeas t-3	ap-southeast-3
아시아 태평양(멜버른)	codepipeline-cloudtrail-pla ceholder-bucket-ap-southeas t-4	ap-southeast-4
아시아 태평양(뭄바이)	codepipeline-cloudtrail-pla ceholder-bucket-ap-south-1	ap-south-1

AWS CodePipeline

리전 이름	자리 표시자 버킷 이름	리전 식별자
아시아 태평양(오사카)	codepipeline-cloudtrail-pla ceholder-bucket-ap-northeas t-3-prod	ap-northeast-3
아시아 태평양(도쿄)	codepipeline-cloudtrail-pla ceholder-bucket-ap-northeas t-1	ap-northeast-1
아시아 태평양(서울)	codepipeline-cloudtrail-pla ceholder-bucket-ap-northeas t-2	ap-northeast-2
아시아 태평양(싱가포르)	codepipeline-cloudtrail-pla ceholder-bucket-ap-southeas t-1	ap-southeast-1
아시아 태평양(시드니)	codepipeline-cloudtrail-pla ceholder-bucket-ap-southeas t-2	ap-southeast-2
아시아 태평양(도쿄)	codepipeline-cloudtrail-pla ceholder-bucket-ap-northeas t-1	ap-northeast-1
캐나다(중부)	codepipeline-cloudtrail-pla ceholder-bucket-ca-central-1	ca-central-1
유럽(프랑크푸르트)	codepipeline-cloudtrail-pla ceholder-bucket-eu-central-1	eu-central-1
유럽(아일랜드)	codepipeline-cloudtrail-pla ceholder-bucket-eu-west-1	eu-west-1
유럽(런던)	codepipeline-cloudtrail-pla ceholder-bucket-eu-west-2	eu-west-2

AWS CodePipeline

리전 이름	자리 표시자 버킷 이름	리전 식별자
유럽(밀라노)	codepipeline-cloudtrail-pla ceholder-bucket-eu-south-1	eu-south-1
유럽(파리)	codepipeline-cloudtrail-pla ceholder-bucket-eu-west-3	eu-west-3
유럽(스페인)	codepipeline-cloudtrail-pla ceholder-bucket-eu-south-2	eu-south-2
유럽(스톡홀름)	codepipeline-cloudtrail-pla ceholder-bucket-eu-north-1	eu-north-1
유럽(취리히)*	codepipeline-cloudtrail-pla ceholder-bucket-eu-central-2	eu-central-2
이스라엘(텔아비브)	codepipeline-cloudtrail-pla ceholder-bucket-il-central-1	il-central-1
중동(바레인)*	codepipeline-cloudtrail-pla ceholder-bucket-me-south-1	me-south-1
중동(UAE)	codepipeline-cloudtrail-pla ceholder-bucket-me-central-1	me-central-1
남아메리카(상파울루)	codepipeline-cloudtrail-pla ceholder-bucket-sa-east-1	sa-east-1

를 사용하여 CodePipeline API 호출 로깅 AWS CloudTrail

AWS CodePipeline 는 CodePipeline AWS 서비스 에서 사용자 AWS CloudTrail, 역할 또는가 수행한 작업에 대한 레코드를 제공하는 서비스와 통합됩니다. CloudTrail은 CodePipeline에 대한 모든 API 직 접 호출을 이벤트로 캡처합니다. 캡처되는 호출에는 CodePipeline 콘솔로부터의 호출과 CodePipeline API 작업에 대한 코드 호출이 포함됩니다. 추적을 생성하면 CodePipeline 이벤트를 포함한 CloudTrail 이벤트를 지속적으로 Amazon S3 버킷에 배포할 수 있습니다. 트레일을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용 하여 CodePipeline에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

CloudTrail의 CodePipeline 정보

CloudTrail은 계정을 생성할 AWS 계정 때에서 활성화됩니다. CodePipeline에서 활동이 발생하면 해당 활동이 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. AWS 계정에 서 최근 이벤트를 보고 검색하고 다운로드할 수 있습니다. 자세한 정보는 <u>CloudTrail 이벤트 기록을 사</u> 용하여 이벤트 보기를 참조하세요.

CodePipeline에 대한 이벤트를 AWS 계정 포함하여에 이벤트를 지속적으로 기록하려면 추적을 생성합 니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 기본적으 로 콘솔에서 추적을 생성하면 추적이 모든 AWS 리전에 적용됩니다. 추적은 AWS 파티션의 모든 리전 에서 이벤트를 로깅하고 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 또한 CloudTrail 로그 에서 수집된 이벤트 데이터를 추가로 분석하고 조치를 취 AWS 서비스 하도록 다른를 구성할 수 있습 니다. 자세한 내용은 다음 자료를 참조하세요.

- 트레일 생성 개요
- <u>CloudTrail 지원 서비스 및 통합</u>
- CloudTrail에서 Amazon SNS 알림 구성
- 여러 리전으로부터 CloudTrail 로그 파일 받기 및 여러 계정으로부터 CloudTrail 로그 파일 받기

모든 CodePipeline 작업은 CloudTrail에서 로깅되고 <u>CodePipeline API 참조</u>에 기록됩니다. 예를 들어 CreatePipeline, GetPipelineExecution, UpdatePipeline 작업을 직접 호출하면 CloudTrail 로그 파일에 항목이 생성됩니다.

모든 이벤트 또는 로그 항목에는 요청을 생성했던 사용자에 대한 정보가 포함됩니다. 자격 증명을 이용 하면 다음을 쉽게 판단할 수 있습니다.

- 요청이 루트 또는 AWS Identity and Access Management (IAM) 자격 증명으로 이루어졌는지 여부입니다.
- 역할 또는 페더레이션 사용자에 대한 임시 보안 인증을 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청했는지 여부.

자세한 설명은 CloudTrail userIdentity 요소를 참조하세요.

CodePipeline 로그 파일 항목 이해

추적이란 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 입력할 수 있게 하는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스로부터의 단일 요청을 나타내며 요청 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보가 들어 있습니다. CloudTrail 로그 파일은 퍼블릭 API 직접 호출의 주문 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

다음 예는 파이프라인 업데이트 이벤트에 대한 CloudTrail 로그 항목을 보여주는데, 여기에 서 MyFirstPipeline이라는 파이프라인은 계정 ID가 80398EXAMPLE이고 이름이 JaneDoe-CodePipeline인 사용자에 의해 편집되었습니다. 사용자가 파이프라인의 소스 단계 이름을 Source에서 MySourceStage로 변경했습니다. CloudTrail 로그의 requestParameters 및 responseElements 요소 모두에 편집된 파이프라인의 전체 구조가 포함되어 있기 때문에, 이러한 요 소는 다음 예시에서 약어로 사용됩니다. 변경이 발생한 파이프라인의 requestParameters 부분, 파 이프라인의 이전 버전 번호, 버전 번호가 1씩 증가되었음을 보여주는 responseElements 부분에 강 조가 추가되었습니다. 실제 로그 항목에는 더 많은 데이터가 표시된다는 뜻에서 편집된 부분은 줄임표 (...)로 표기됩니다.

```
{
 "eventVersion":"1.03",
  "userIdentity": {
  "type":"IAMUser",
  "principalId":"AKIAI44QH8DHBEXAMPLE",
   "arn":"arn:aws:iam::80398EXAMPLE:user/JaneDoe-CodePipeline",
  "accountId": "80398EXAMPLE",
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "userName":"JaneDoe-CodePipeline",
  "sessionContext": {
  "attributes":{
   "mfaAuthenticated":"false",
    "creationDate":"2015-06-17T14:44:03Z"
   }
  },
 "invokedBy":"signin.amazonaws.com"},
 "eventTime":"2015-06-17T19:12:20Z",
 "eventSource": "codepipeline.amazonaws.com",
 "eventName": "UpdatePipeline",
"awsRegion":"us-east-2",
"sourceIPAddress":"192.0.2.64",
 "userAgent":"signin.amazonaws.com",
 "requestParameters":{
```

```
"pipeline":{
   "version":1,
   "roleArn":"arn:aws:iam::80398EXAMPLE:role/CodePipeline_Service_Role",
   "name":"MyFirstPipeline",
   "stages":[
     {
     "actions":[
      {
          "name": "MySourceStage",
        "actionType":{
        "owner":"AWS",
        "version":"1",
        "category":"Source",
        "provider":"S3"
     },
    "inputArtifacts":[],
    "outputArtifacts":[
     {"name":"MyApp"}
     ],
    "runOrder":1,
    "configuration":{
     "S3Bucket": "amzn-s3-demo-source-bucket",
     "S3ObjectKey":"sampleapp_linux.zip"
      }
      }
     ],
       "name":"Source"
     },
     (...)
               },
 "responseElements":{
   "pipeline":{
     "version":2,
     (...)
           },
     "requestID": "2c4af5c9-7ce8-EXAMPLE",
     "eventID":""c53dbd42-This-Is-An-Example"",
     "eventType":"AwsApiCall",
     "recipientAccountId":"80398EXAMPLE"
     }
      ]
}
```

CodePipeline CloudWatch 지표

CloudWatch의 지표를 사용하여 CodePipeline 파이프라인을 측정할 수 있습니다. 다음 지표를 사용하여 다이프라인 기간과 실패한 파이프라인 실행을 측정할 수 있습니다.

파이프라인은 두 가지 수준에서 모니터링할 수 있습니다.

파이프라인 수준

이러한 지표는 파이프라인 수준에 있습니다. CloudWatch에서 파이프라인별을 선택합니다. 사용 가 능한 파이프라인은 CloudWatch에 나열됩니다.

AWS 계정 수준

이러한 지표는 계정의 모든 파이프라인에 적용됩니다. AWS 계정 수준에서 지표를 보려면 CloudWatch에서 계정 지표를 선택합니다.

지표 보기에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 <u>사용 가능한 지표 보기를</u> 참조하 세요.

주제

- PipelineDuration
- FailedPipelineExecutions

PipelineDuration

PipelineDuration 지표는 파이프라인 실행 기간을 측정합니다. 이 지표는 파이프라인 수준에서만 사용할 수 있습니다.

단위: 초

FailedPipelineExecutions

FailedPipelineExecutions 지표는 실패한 파이프라인 실행 수를 측정합니다. 이 지표는 파이프 라인 수준 및 계정 수준에서 사용할 수 있습니다.

이 지표에는 실패한 작업을 재시도하는 파이프라인 실행에 대한 별도의 수가 포함됩니다. 즉, 실패한 작업이 파이프라인에서 재시도되면 별도의 파이프라인 실행 지표로 계산됩니다. 예를 들어, S3 소스 작업이 처음 실패하여 한 번 실행한 다음 소스 작업이 재시도되고 다시 실패하는 S3 소스 작업이 있는 파이프라인의 경우. 이 예제에서 지표는 다음과 같습니다.

FailedPipelineExecutionAttempts: 2

단위: 개

CodePipeline 문제 해결

다음은 AWS CodePipeline에서 일반적으로 발생하는 문제를 해결하는 데 유용한 정보입니다.

주제

- 파이프라인 오류: AWS Elastic Beanstalk 으로 구성한 파이프라인이 오류 메시지를 반환 했습니다. "Deployment failed. The provided role does not have sufficient permissions: Service:AmazonElasticLoadBalancing"
- <u>배포 오류: "DescribeEvents" 권한이 누락되면 AWS Elastic Beanstalk 배포 작업으로 구성한 파이프</u> 라인이 실패하는 대신 멈춥니다.
- <u>파이프라인 오류: 소스 작업이 권한이 부족하다는 메시지를 반환합니다. "CodeCommit 리포지토리</u> repository-name에 액세스할 수 없습니다. 파이프라인 IAM 역할에 있는 리포지토리 액세스 권한이 부족한지 확인하십시오."
- <u>파이프라인 오류: Jenkins 빌드 또는 테스트 작업을 오래 지속했는데 자격 증명 또는 권한이 없어서</u> 실패했습니다.
- <u>파이프라인 오류: 다른 AWS 리전에서 생성된 버킷을 사용하여 한 AWS 리전에서 생성된 파이프라</u> 인은 코드 "JobFailed"와 함께 "InternalError"를 반환합니다.
- <u>배포 오류: WAR 파일이 포함된 ZIP 파일이에 성공적으로 배포 AWS Elastic Beanstalk되었지만 애</u> 플리케이션 URL이 404 찾을 수 없음 오류를 보고합니다.
- 파이프라인 아티팩트 폴더 이름이 잘린 것처럼 보입니다.
- <u>Bitbucket, GitHub, GitHub Enterprise Server 또는 GitLab.com 연결을 위한 CodeBuild GitClone 권</u> 한 추가
- CodeCommit 소스 작업에 대한 CodeBuild GitClone 권한 추가
- <u>파이프라인 오류: CodeDeployToECS 작업을 통해 배포하면 "<소스 아티팩트 이름>에서 작업 정의</u> 아티팩트 파일을 읽으려고 시도하는 동안 예외가 발생했습니다"라는 오류 메시지가 반환됩니다.
- GitHub(OAuth 앱 사용) 소스 작업: 리포지토리 목록에 다른 리포지토리가 표시됨
- GitHub(GitHub 앱 사용) 소스 작업: 리포지토리에 대한 연결을 완료할 수 없음
- <u>Amazon S3 오류: CodePipeline 서비스 역할 <ARN>에서 S3 버킷<BucketName>에 대한 S3 액세스</u> 가 거부되었습니다.
- <u>Amazon S3, Amazon ECR 또는 CodeCommit 소스를 사용하는 파이프라인은 더 이상 자동으로 시</u> <u>작되지 않음</u>
- <u>GitHub에 연결할 때 연결 오류 발생: "A problem occurred, make sure cookies are enabled in your</u> browser" 또는 "An organization owner must install the GitHub app"

- 실행 모드가 QUEUED 또는 PARALLEL 모드로 변경된 파이프라인은 실행 한도에 도달하면 실패합 니다.
- <u>QUEUED 또는 SUPERSEDED 모드로 변경할 때 편집하면 PARALLEL 모드의 파이프라인에 오래된</u> 파이프라인 정의가 있습니다.
- PARALLEL 모드에서 변경된 파이프라인이 이전 실행 모드를 표시합니다.
- <u>파일 경로별 트리거 필터링을 사용하는 연결이 있는 파이프라인은 브랜치 생성 시 시작되지 않을 수</u> 있습니다.
- <u>파일 경로별 트리거 필터링을 사용하는 연결이 있는 파이프라인은 파일 한도에 도달하면 시작되지</u> 않을 수 있습니다.
- <u>PARALLEL 모드의 CodeCommit 또는 S3 소스 개정이 EventBridge 이벤트와 일치하지 않을 수 있습니다.</u>
- <u>오류 메시지와 함께 EC2 배포 작업이 실패함 No such file</u>
- cluster unreachable 오류 메시지와 함께 EKS 배포 작업이 실패함
- <u>다른 문제에 도움이 필요하십니까?</u>

파이프라인 오류: AWS Elastic Beanstalk 으로 구성한 파이 프라인이 오류 메시지를 반환했습니다. "Deployment failed.

The provided role does not have sufficient permissions:

Service:AmazonElasticLoadBalancing"

문제: CodePipeline의 서비스 역할에 Elastic Load Balancing의 일부 작업을 AWS Elastic Beanstalk포 함하되 이에 국한되지 않는에 대한 충분한 권한이 없습니다. 이 문제를 해결하기 위해 CodePipeline에 대한 서비스 역할을 2015년 8월 6일 업데이트했습니다. 이 날짜 전에 서비스 역할을 만든 고객은 필요 한 권한을 추가하도록 서비스 역할의 정책 설명을 변경해야 합니다.

수정 방법: 가장 간단한 해결 방법은 <u>CodePipeline 서비스 역할에 권한 추가</u>에 자세히 설명된 대로 서 비스 역할에 대한 정책 설명을 편집하는 것입니다.

편집한 정책을 적용한 후 <u>수동으로 파이프라인 시작</u>의 단계를 따라 Elastic Beanstalk를 사용하는 모든 파이프라인을 수동으로 반환하세요.

보안 필요성에 따라 다른 방법으로도 권한을 변경할 수 있습니다.

배포 오류: "DescribeEvents" 권한이 누락되면 AWS Elastic Beanstalk 배포 작업으로 구성한 파이프라인이 실패하는 대신 멈춥 니다.

문제: CodePipeline의 서비스 역할이 AWS Elastic Beanstalk를 사용하는 파이프라인에 대해 "elasticbeanstalk:DescribeEvents" 작업을 포함해야 합니다. 이 권한이 없으면 AWS Elastic Beanstalk 배포 작업이 실패하거나 오류를 표시하지 않고 중단됩니다. 서비스 역할에서이 작업이 누락 된 경우 CodePipeline은 AWS Elastic Beanstalk 사용자를 대신하여에서 파이프라인 배포 단계를 실행 할 권한이 없습니다.

수정 방법: CodePipeline 서비스 역할을 검토합니다. "elasticbeanstalk:DescribeEvents" 작 업이 누락된 경우 <u>CodePipeline 서비스 역할에 권한 추가</u>의 단계에 따라 IAM 콘솔에서 정책 편집 기능 을 사용하여 작업을 추가합니다.

편집한 정책을 적용한 후 <u>수동으로 파이프라인 시작</u>의 단계를 따라 Elastic Beanstalk를 사용하는 모든 파이프라인을 수동으로 반환하세요.

파이프라인 오류: 소스 작업이 권한이 부족하다는 메시지를 반환합 니다. "CodeCommit 리포지토리 **repository-name**에 액세스할 수 없습니다. 파이프라인 IAM 역할에 있는 리포지토리 액세스 권한이 부족한지 확인하십시오."

문제: CodePipeline에 대한 서비스 역할에 CodeCommit 권한이 부족하고 CodeCommit 리포지토리 이 용 지원을 추가한 2016년 4월 18일 이전에 만든 것 같습니다. 이 날짜 전에 서비스 역할을 만든 고객은 필요한 권한을 추가하도록 서비스 역할의 정책 설명을 변경해야 합니다.

수정 방법: CodePipeline 서비스 역할의 정책에 필요한 CodeCommit 권한을 추가합니다. 자세한 내용 은 <u>CodePipeline 서비스 역할에 권한 추가</u> 단원을 참조하십시오.

파이프라인 오류: Jenkins 빌드 또는 테스트 작업을 오래 지속했는데 자격 증명 또는 권한이 없어서 실패했습니다.

문제: Amazon EC2 인스턴스에 Jenkins 서버를 설치한 경우 CodePipeline에 필요한 권한이 있는 인스 턴스 역할로 인스턴스를 만들지 않았을 수 있습니다. 필수 IAM 역할이 없는 채로 만든 Jenkins 서버, 온 프레미스 인스턴스, 혹은 Amazon EC2 인스턴스에서 IAM 사용자를 사용 중인 경우, IAM 사용자에게 필수 권한이 없거나 Jenkins 서버가 해당 서버에서 구성한 프로파일을 통해 그러한 자격 증명에 액세스 할 수가 없습니다.

수정 방법: Amazon EC2 인스턴스 역할 또는 IAM 사용자가

AWSCodePipelineCustomActionAccess 관리형 정책 혹은 동등한 권한으로 구성되었는지 확인합 니다. 자세한 내용은 AWS 에 대한 관리형 정책 AWS CodePipeline 단원을 참조하십시오.

IAM 사용자를 사용하는 경우 인스턴스에 구성된 AWS 프로필이 올바른 권한으로 구성된 IAM 사용자 를 사용하는지 확인합니다. Jenkins와 CodePipeline 간 통합용으로 구성한 IAM 사용자 자격 증명을 Jenkins UI에 바로 제공해야 할 수도 있습니다. 이것은 권장되는 모범 사례는 아닙니다. 이 방법을 꼭 써야 한다면 Jenkins 서버의 보안을 확인하고 HTTP 대신 HTTPS를 사용하십시오.

파이프라인 오류: 다른 AWS 리전에서 생성된 버킷을 사용하여 한 AWS 리전에서 생성된 파이프라인은 코드 "JobFailed"와 함께 "InternalError"를 반환합니다.

문제: 파이프라인과 버킷이 서로 다른 AWS 리전에 생성되면 Amazon S3 버킷에 저장된 아티팩트 다 운로드가 실패합니다.

수정 방법: 아티팩트가 저장된 Amazon S3 버킷이 생성한 파이프라인과 동일한 AWS 리전에 있는지 확인합니다.

배포 오류: WAR 파일이 포함된 ZIP 파일이에 성공적으로 배포 AWS Elastic Beanstalk되었지만 애플리케이션 URL이 404 찾을 수 없음 오류를 보고합니다.

문제: WAR 파일이 AWS Elastic Beanstalk 환경에 성공적으로 배포되었으나 애플리케이션 URL이 404 Not Found 오류를 반환합니다.

가능한 수정 사항: AWS Elastic Beanstalk ZIP 파일의 압축을 풀 수 있지만 ZIP 파일에 포함된 WAR 파 일은 풀 수 없습니다. buildspec.yml 파일에서 WAR 파일을 지정하는 대신 배포할 콘텐츠가 포함된 폴더를 지정하십시오. 예시:

version: 0.2

phases:

post_build:

```
사용자 가이드
```

```
commands:
    - mvn package
    - mv target/my-web-app ./
artifacts:
    files:
    - my-web-app/**/*
discard-paths: yes
```

예시는 <u>CodeBuild용AWS Elastic Beanstalk 샘플</u>을 참조하세요.

파이프라인 아티팩트 폴더 이름이 잘린 것처럼 보입니다.

문제점: 파이프라인 아티팩트 이름을 CodePipeline에서 보면 이름이 잘린 것처럼 보입니다. 이로 인해 이름이 비슷하거나 더 이상 전체 파이프라인 이름을 포함하지 않는 것처럼 보일 수 있습니다.

설명: CodePipeline은 아티팩트 이름을 잘라내어 CodePipeline이 작업자를 위한 임시 자격 증명을 생 성할 때 전체 Amazon S3 경로가 정책 크기 제한을 초과하지 않도록 합니다.

아티팩트 이름이 잘린 것처럼 보이더라도 CodePipeline은 이름이 잘린 아티팩트의 영향을 받지 않는 방식으로 아티팩트 버킷에 매핑합니다. 파이프라인은 정상적으로 작동할 수 있습니다. 이는 폴더 또는 결과물에 문제가 되지 않습니다. 파이프라인 이름은 100자 이내로 제한됩니다. 결과물 폴더 이름이 짧 아 보이더라도 여전히 파이프라인에 고유합니다.

Bitbucket, GitHub, GitHub Enterprise Server 또는 GitLab.com 연결 을 위한 CodeBuild GitClone 권한 추가

소스 작업과 CodeBuild 작업에서 AWS CodeStar 연결을 사용하는 경우 입력 아티팩트를 빌드에 전달 할 수 있는 두 가지 방법이 있습니다.

- 기본 방법: 소스 작업이 CodeBuild가 다운로드하는 코드가 포함된 zip 파일을 생성합니다.
- Git 복제: 빌드 환경에서 소스 코드를 직접 다운로드할 수 있습니다.

Git 복제 모드를 사용하면 작업 Git 리포지토리로서 소스 코드와 상호 작용할 수 있습니다. 이 모드를 사용하려면 연결을 사용할 수 있는 권한을 CodeBuild 환경에 부여해야 합니다.

CodeBuild 서비스 역할 정책에 권한을 추가하려면 CodeBuild 서비스 역할에 연결되는 고객 관리형 정 책을 만듭니다. 다음 단계에서는 action 필드에 UseConnection 권한이 지정되고 Resource 필드 에 연결 ARN이 지정된 정책을 만듭니다.

콘솔을 사용하여 UseConnection 권한을 추가하려면

 파이프라인을 열고 소스 작업에서 (i) 아이콘을 클릭하여 파이프라인의 연결 ARN을 찾습니다. CodeBuild 서비스 역할 정책에 연결 ARN을 추가합니다.

연결 ARN의 예는 다음과 같습니다.

```
arn:aws:codeconnections:eu-central-1:123456789123:connection/
sample-1908-4932-9ecc-2ddacee15095
```

- CodeBuild 서비스 역할을 찾으려면 파이프라인에 사용된 빌드 프로젝트를 열고 [빌드 세부 정보 (Build details)] 탭으로 이동합니다.
- 서비스 역할 링크를 선택합니다. 그러면 연결에 대한 액세스 권한을 부여하는 새 정책을 추가할 수 있는 IAM 콘솔이 열립니다.
- 4. IAM 콘솔에서 Attach policies(정책 연결)를 선택하고 Create policy(정책 생성)를 선택합니다.

다음 샘플 정책 템플릿을 사용합니다. 다음 예와 같이 Resource 필드에 연결 ARN을 추가합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codestar-connections:UseConnection",
            "Resource": "insert connection ARN here"
        }
    ]
}
```

JSON 탭에서 정책을 붙여 넣습니다.

- 5. 정책 검토를 선택합니다. 정책 이름(예: **connection-permissions**)을 입력한 후 Create policy(정책 생성)를 선택합니다.
- 권한을 연결한 페이지로 돌아가서 정책 목록을 새로 고친 다음 방금 생성한 정책을 선택합니다.
 책 연결을 선택합니다.

Attac	h	Permis	sions			
Creat	te p	olicy				2
Filter p	poli	cies ~	Q connection			Showing 1 result
		Policy na	ame 👻	Ту	pe	Used as
	Þ	conn	ection-permissions	Cu	ustomer managed	None

CodeCommit 소스 작업에 대한 CodeBuild GitClone 권한 추가

파이프라인에 CodeCommit 소스 작업이 있는 경우 다음 두 가지 방법으로 입력 아티팩트를 빌드에 전 달할 수 있습니다.

- 기본 방법 소스 작업이 CodeBuild가 다운로드하는 코드가 포함된 zip 파일을 생성합니다.
- 전체 복제 빌드 환경에서 소스 코드를 직접 다운로드할 수 있습니다.

전체 복제 옵션을 사용하면 작업 Git 리포지토리로서 소스 코드와 상호 작용할 수 있습니다. 이 모드 를 사용하려면 리포지토리에서 가져올 수 있는 권한을 CodeBuild 환경에 추가해야 합니다.

CodeBuild 서비스 역할 정책에 권한을 추가하려면 CodeBuild 서비스 역할에 연결되는 고객 관리형 정 책을 만듭니다. 다음 단계는 action 필드의 codecommit:GitPull 권한을 지정하는 정책을 생성합 니다.

콘솔을 사용하여 GitPull 권한을 추가하려면

- 1. CodeBuild 서비스 역할을 찾으려면 파이프라인에 사용된 빌드 프로젝트를 열고 [빌드 세부 정보 (Build details)] 탭으로 이동합니다.
- 서비스 역할 링크를 선택합니다. 그러면 리포지토리에 대한 액세스 권한을 부여하는 새 정책을 추 가할 수 있는 IAM 콘솔이 열립니다.
- 3. IAM 콘솔에서 Attach policies(정책 연결)를 선택하고 Create policy(정책 생성)를 선택합니다.
- 4. JSON 탭에 다음 샘플 정책을 붙여넣습니다.

```
{
    "Action": [
        "codecommit:GitPull"
],
    "Resource": "*",
```

"Effect": "Allow"
},

- 5. 정책 검토를 선택합니다. 정책 이름(예: **codecommit-gitpull**)을 입력한 후 Create policy(정책 생성)를 선택합니다.
- 권한을 연결한 페이지로 돌아가서 정책 목록을 새로 고친 다음 방금 생성한 정책을 선택합니다. 정 책 연결을 선택합니다.

파이프라인 오류: CodeDeployToECS 작업을 통해 배포하면 "<소스 아티팩트 이름>에서 작업 정의 아티팩트 파일을 읽으려고 시도하는 동안 예외가 발생했습니다"라는 오류 메시지가 반환됩니다.

문제:

태스크 정의 파일은 CodeDeploy(CodeDeployToECS 작업)를 통해 Amazon ECS로 CodePipeline을 배포하는 작업에 필요한 아티팩트입니다. CodeDeployToECS 배포 작업의 최대 아티팩트 ZIP 크기는 3MB입니다. 파일을 찾을 수 없거나 아티팩트 크기가 3MB를 초과하면 다음 오류 메시지가 반환됩니다.

<소스 아티팩트 이름>에서 작업 정의 아티팩트 파일을 읽으려고 시도하는 동안 예외가 발생했습니다

수정 방법: 태스크 정의 파일이 아티팩트로 포함되어 있는지 확인하세요. 파일이 이미 있는 경우 압축 된 크기가 3MB 미만인지 확인하세요.

GitHub(OAuth 앱 사용) 소스 작업: 리포지토리 목록에 다른 리포지 토리가 표시됨

문제:

CodePipeline 콘솔에서 GitHub(OAuth 앱 사용) 작업에 대한 권한 부여가 성공하면 GitHub 리포지토리 목록에서 선택할 수 있습니다. 목록에 표시될 것으로 예상했던 리포지토리가 포함되어 있지 않은 경우 권한 부여에 사용된 계정의 문제를 해결할 수 있습니다.

수정 방법: CodePipeline 콘솔에서 제공되는 리포지토리 목록은 승인된 계정이 속한 GitHub 조직을 기 반으로 합니다. GitHub로 승인하기 위해 사용하는 계정이 리포지토리가 생성된 GitHub 조직과 연결된 계정인지 확인하세요.

GitHub(GitHub 앱 사용) 소스 작업: 리포지토리에 대한 연결을 완료 할 수 없음

문제:

GitHub 리포지토리에 대한 연결은 GitHub용 AWS 커넥터를 사용하므로 연결을 생성하려면 리포지토 리에 대한 조직 소유자 권한 또는 관리자 권한이 필요합니다.

가능한 해결 방법: GitHub 리포지토리의 권한 수준에 대한 자세한 내용은 <u>https://docs.github.com/en/</u> <u>free-pro-team@latest/github/setting-up-and-managing-organizations-and-teams/permission-levels-</u> <u>for-an-organization</u> 섹션을 참조하세요.

Amazon S3 오류: CodePipeline 서비스 역할 <ARN>에서 S3 버킷 <BucketName>에 대한 S3 액세스가 거부되었습니다.

문제:

진행 중에 CodePipeline의 CodeCommit 작업은 파이프라인 아티팩트 버킷이 존재하는지 확인합니다. 작업에 검사 권한이 없는 경우 Amazon S3에서 AccessDenied 오류가 발생하고 CodePipeline에 다 음 오류 메시지가 표시됩니다.

CodePipeline 서비스 역할 "arn:aws:iam::*AccountID*:role/service-role/*RoleID*"에서 S3 버킷 "*BucketName*"에 대한 S3 액세스가 거부되었습니다.

작업에 대한 CloudTrail 로그에도 AccessDenied 오류가 기록됩니다.

수정 방법: 다음을 수행합니다.

- CodePipeline 서비스 역할에 연결된 정책의 경우 정책의 작업 목록에 s3:ListBucket을 추가합니다. 서비스 역할 정책을 보는 방법에 대한 지침은 <u>파이프라인 ARN 및 서비스 역할 ARN 보기(콘솔)</u> 단원을 참조하세요. <u>CodePipeline 서비스 역할에 권한 추가</u>에 설명된 대로 서비스 역할에 대한 정책 설명을 편집하세요.
- 파이프라인의 Amazon S3 아티팩트 버킷에 연결된 리소스 기반 정책(아티팩트 버킷 정책이라고도 함)의 경우 CodePipeline 서비스 역할이 s3:ListBucket 권한을 사용할 수 있도록 허용하는 명령 문을 추가합니다.

아티팩트 버킷에 정책을 추가하려면

- 1. <u>파이프라인 ARN 및 서비스 역할 ARN 보기(콘솔)</u>의 단계에 따라 파이프라인 설정 페이지에서 아티팩트 버킷을 선택한 다음 Amazon S3 콘솔에서 확인합니다.
- 2. Permissions를 선택합니다.
- 3. 버킷 정책에서 편집을 선택합니다.
- 4. 정책 텍스트 필드에서 새 버킷 정책을 입력하거나 다음 예와 같이 기존 정책을 편집합니다. 버킷 정책은 JSON 파일이므로 유효한 JSON을 입력해야 합니다.

다음 예제는 서비스 역할의 예제 역할 ID가 *AROAEXAMPLEID*인 아티팩트 버킷의 버킷 정책 설 명을 보여줍니다.

```
{
    "Effect": "Allow",
    "Principal": "*",
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::BucketName",
    "Condition": {
        "StringLike": {
            "aws:userid": "AROAEXAMPLEID:*"
        }
    }
}
```

다음 예제는 권한이 추가된 후의 동일한 버킷 정책 설명을 보여줍니다.

```
},
        {
            "Sid": "DenyUnEncryptedObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*",
            "Condition": {
                "StringNotEquals": {
                     "s3:x-amz-server-side-encryption": "aws:kms"
                }
            }
        },
        {
            "Sid": "DenyInsecureConnections",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*",
            "Condition": {
                "Bool": {
                     "aws:SecureTransport": false
                }
            }
        }
    ]
}
```

자세한 내용은 <u>https://aws.amazon.com/blogs/security/writing-iam-policies-how-to-grant-</u> access-to-an-amazon-s3-bucket/의 단계를 참조하세요.

5. 저장(Save)을 선택합니다.

편집한 정책을 적용한 후 <u>수동으로 파이프라인 시작</u>의 단계를 따라 파이프라인을 수동으로 다시 실행 하세요.

Amazon S3, Amazon ECR 또는 CodeCommit 소스를 사용하는 파이 프라인은 더 이상 자동으로 시작되지 않음

문제:

변경 감지를 위해 이벤트 규칙(EventBridge 또는 CloudWatch Events)을 사용하는 작업에 대한 구성 설 정을 변경한 후 소스 트리거 식별자가 유사하고 첫 문자가 동일한 경우 콘솔이 변경 사항을 감지하지 못할 수 있습니다. 콘솔에서 새 이벤트 규칙을 생성하지 않으므로 파이프라인이 더 이상 자동으로 시작 되지 않습니다.

CodeCommit의 파라미터 이름 끝에 있는 사소한 변경의 예로는 CodeCommit 브랜치 이름 MyTestBranch-1을 MyTestBranch-2로 변경하는 경우를 들 수 있습니다. 브랜치 이름 끝에 변경 사항이 적용되므로 소스 작업의 이벤트 규칙이 업데이트되지 않거나 새 소스 설정에 대한 규칙이 생성 되지 않을 수 있습니다.

이는 다음과 같이 변경 감지에 CWE 이벤트를 사용하는 소스 작업에 적용됩니다.

소스 작업	파라미터/트리거 식별자(콘솔)
Amazon ECR	리포지토리 이름
	이미지 태그
Amazon S3	버킷
	S3 객체 키
CodeCommit	리포지토리 이름
	브랜치 이름

수정 방법:

다음 중 하나를 수행합니다.

• 파라미터 값의 시작 부분이 변경되도록 CodeCommit/S3/ECR 구성 설정을 변경합니다.

예: 브랜치 이름 release-branch를 2nd-release-branch로 변경합니다. 이름의 끝부분은 변경 하지 마십시오(예: release-branch-2).

• 각 파이프라인의 CodeCommit/S3/ECR 구성 설정을 변경합니다.

예: 브랜치 이름 myRepo/myBranch를 myDeployRepo/myDeployBranch로 변경합니다. 이름의 끝부분은 변경하지 마십시오(예: myRepo/myBranch2).

• 콘솔 대신 CLI 또는를 사용하여 변경 감지 이벤트 규칙을 AWS CloudFormation 생성하고 업데이트 합니다. S3 소스 작업에 대한 이벤트 규칙 생성에 대한 지침은 EventBridge 및를 사용하는 Amazon <u>S3 소스 작업에 연결 AWS CloudTrail</u> 단원을 참조하세요. Amazon ECR 작업에 대한 이벤트 규칙 생성에 대한 지침은 <u>Amazon ECR 소스 작업 및 EventBridge 리소스</u> 단원을 참조하세요. CodeCommit 작업에 대한 이벤트 규칙 생성에 대한 지침은 <u>CodeCommit 소스 작업 및 EventBridge</u> 단원을 참조하세요.

콘솔에서 작업 구성을 편집한 후에는 콘솔에서 만든 업데이트된 변경 감지 리소스를 수락합니다.

GitHub에 연결할 때 연결 오류 발생: "A problem occurred, make sure cookies are enabled in your browser" 또는 "An organization owner must install the GitHub app"

문제:

CodePipeline의 GitHub 소스 작업에 대한 연결을 생성하려면 GitHub 조직 소유자여야 합니다. 조직 소 속이 아닌 리포지토리의 경우 리포지토리 소유자여야 합니다. 조직 소유자가 아닌 다른 사람이 연결을 생성하면 조직 소유자에 대한 요청이 생성되고 다음 오류 중 하나가 표시됩니다.

A problem occurred, make sure cookies are enabled in your browser(문제가 발생했습니다. 브라우저에서 쿠키가 활성화되어 있는지 확인하십시오.)

OR

An organization owner must install the GitHub app(조직 소유자가 GitHub 앱을 설치해야 합니다.)

가능한 수정: GitHub 조직의 리포지토리의 경우 조직 소유자가 GitHub 리포지토리에 대한 연결을 생성 해야 합니다. 조직 소속이 아닌 리포지토리의 경우 리포지토리 소유자여야 합니다.

실행 모드가 QUEUED 또는 PARALLEL 모드로 변경된 파이프라인 은 실행 한도에 도달하면 실패합니다.

문제: QUEUED 모드의 파이프라인에 대한 최대 동시 실행 수는 50회입니다. 이 한도에 도달하면 파이 프라인이 상태 메시지 없이 실패합니다.

수정 방법: 실행 모드에 대한 파이프라인 정의를 편집할 때 다른 편집 작업과 별도로 편집합니다.

QUEUED 또는 PARALLEL 실행 모드에 대한 자세한 내용은 CodePipeline 개념 섹션을 참조하세요.

QUEUED 또는 SUPERSEDED 모드로 변경할 때 편집하면 PARALLEL 모드의 파이프라인에 오래된 파이프라인 정의가 있습니 다.

문제: 병렬 모드의 파이프라인의 경우 파이프라인 실행 모드를 QUEUED 또는 SUPERSEDED로 편집 하면 PARALLEL 모드에 대한 파이프라인 정의가 업데이트되지 않습니다. PARALLEL 모드를 업데이 트할 때 업데이트된 파이프라인 정의는 SUPERSEDED 또는 QUEUED 모드에서 사용되지 않습니다.

수정 방법: 병렬 모드의 파이프라인의 경우 파이프라인 실행 모드를 QUEUED 또는 SUPERSEDED로 편집할 때 파이프라인 정의를 동시에 업데이트하지 않습니다.

QUEUED 또는 PARALLEL 실행 모드에 대한 자세한 내용은 <u>CodePipeline 개념</u> 섹션을 참조하세요.

PARALLEL 모드에서 변경된 파이프라인이 이전 실행 모드를 표시합 니다.

문제: PARALLEL 모드의 파이프라인의 경우 파이프라인 실행 모드를 QUEUED 또는 SUPERSEDED 로 편집하면 파이프라인 상태에 업데이트된 상태가 PARALLEL로 표시되지 않습니다. 파이프라인이 PARALLEL에서 QUEUED 또는 SUPERSEDED로 변경된 경우 SUPERSED 또는 QUEUED 모드의 파 이프라인 상태는 해당 모드 중 하나에서 마지막으로 알려진 상태가 됩니다. 파이프라인이 이전에 해당 모드에서 실행된 적이 없는 경우 상태는 비어 있습니다.

수정 방법: 병렬 모드의 파이프라인의 경우 파이프라인 실행 모드를 QUEUED 또는 SUPERSEDED로 편집할 때 실행 모드 디스플레이에 PARALLEL 상태가 표시되지 않는다는 점을 유념하세요.

QUEUED 또는 PARALLEL 실행 모드에 대한 자세한 내용은 CodePipeline 개념 섹션을 참조하세요.

파일 경로별 트리거 필터링을 사용하는 연결이 있는 파이프라인은 브랜치 생성 시 시작되지 않을 수 있습니다.

설명: BitBucket 소스 작업과 같이 연결을 사용하는 소스 작업이 있는 파이프라인의 경우, 파일 경로를 기준으로 필터링하여 파이프라인을 시작할 수 있는 Git 구성으로 트리거를 설정할 수 있습니다. 경우에 따라 파일 경로에서 필터링되는 트리거가 있는 파이프라인의 경우 CodeConnections 연결에서 변경된 파일을 확인할 수 없으므로 파일 경로 필터가 있는 브랜치가 처음 생성될 때 파이프라인이 시작되지 않 을 수 있습니다. 트리거에 대한 Git 구성이 파일 경로를 필터링하도록 설정된 경우 필터가 있는 브랜치 가 소스 리포지토리에서 방금 생성되면 파이프라인이 시작되지 않습니다. 파일 경로 필터링에 대한 자 세한 내용은 코드 푸시 또는 풀 요청 이벤트 유형을 사용하여 트리거 추가 섹션을 참조하세요. 결과: 예를 들어, 브랜치 'B'에 파일 경로 필터가 있는 CodePipeline의 파이프라인은 브랜치 'B'가 생성 될 때 트리거되지 않습니다. 파일 경로 필터가 없는 경우에도 파이프라인은 계속 시작됩니다.

파일 경로별 트리거 필터링을 사용하는 연결이 있는 파이프라인은 파일 한도에 도달하면 시작되지 않을 수 있습니다.

설명: BitBucket 소스 작업과 같이 연결을 사용하는 소스 작업이 있는 파이프라인의 경우, 파일 경로 를 기준으로 필터링하여 파이프라인을 시작할 수 있는 Git 구성으로 트리거를 설정할 수 있습니다. CodePipeline은 처음 100개 파일까지 검색합니다. 따라서 트리거의 Git 구성이 파일 경로에서 필터링 하도록 설정된 경우 파일이 100개 이상이면 파이프라인이 시작되지 않을 수 있습니다. 파일 경로에서 필터링에 대한 자세한 내용은 <u>코드 푸시 또는 풀 요청 이벤트 유형을 사용하여 트리거 추가</u> 섹션을 참 조하세요.

결과: 예를 들어 diff에 150개의 파일이 포함된 경우 CodePipeline은 지정된 파일 경로 필터와 비교하여 처음 100개의 파일(특정 순서 없음)을 확인합니다. 파일 경로 필터와 일치하는 파일이 CodePipeline에 서 검색한 100개의 파일 중 하나가 아닌 경우 파이프라인이 호출되지 않습니다.

PARALLEL 모드의 CodeCommit 또는 S3 소스 개정이 EventBridge 이벤트와 일치하지 않을 수 있습니다.

설명: PARALLEL 모드에서 파이프라인 실행의 경우 CodeCommit 리포지토리 커밋과 같은 가장 최근 변경 사항으로 실행이 시작될 수 있으며, 이는 EventBridge 이벤트의 변경 사항과 동일하지 않을 수 있 습니다. 파이프라인을 시작하는 커밋이나 이미지 태그 사이에 아주 짧은 순간의 간격이 있을 수 있는 데, CodePipeline이 이벤트를 수신하고 해당 실행을 시작했을 때 다른 커밋이나 이미지 태그가 푸시된 경우(예: CodeCommit 작업) CodePipeline은 해당 순간에 HEAD 커밋을 복제합니다.

결과: CodeCommit 또는 S3 소스가 있는 PARALLEL 모드의 파이프라인의 경우 파이프라인 실행을 트 리거한 변경 사항에 관계없이 소스 작업은 시작 시 항상 HEAD를 복제합니다. 예를 들어 PARALLEL 모 드의 파이프라인의 경우 커밋이 푸시되어 실행 1을 위한 파이프라인이 시작되고 두 번째 파이프라인 실행은 두 번째 커밋을 사용합니다.

오류 메시지와 함께 EC2 배포 작업이 실패함 No such file

설명: EC2 배포 작업이 인스턴스의 대상 디렉터리에서 아티팩트의 압축을 푼 후 작업은 스크립트를 실 행합니다. 스크립트가 대상 디렉터리에 있지만 작업이 스크립트를 실행할 수 없는 경우 해당 인스턴스 에서 작업이 실패하고 나머지 인스턴스는 배포에 실패합니다. 대상 디렉터리가 이고 소스 리포지토리 경로가 인 배포의 로그에 다음 오류 메시지/home/ec2user/deploy/와 유사한 오류가 표시됩니다myRepo/postScript.sh.

```
Instance i-0145a2d3f3EXAMPLE is FAILED on event AFTER_DEPLOY, message:
-----ERROR------
chmod: cannot access '/home/ec2-user/deploy/myRepo/postScript.sh': No such file or
directory
/var/lib/<path>/_script.sh: line 2: /home/ec2-user/deploy/myRepo/postScript.sh: No
such file or directory
failed to run commands: exit status 127
```

Executing commands on instances i-0145a2d3f3EXAMPLE, SSM command id <ID>, commands: chmod u+x /home/ec2-user/deploy/script.sh ------ERROR------: No such file or directory

결과: 파이프라인에서 배포 작업이 실패합니다.

가능한 수정 사항: 문제를 해결하려면 다음 단계를 사용합니다.

- 1. 로그를 보고 스크립트가 실패한 인스턴스를 확인합니다.
- 디렉터리(cd)를 인스턴스의 대상 디렉터리로 변경합니다. 인스턴스에서 스크립트를 테스트 실행합 니다.
- 소스 리포지토리에서 스크립트 파일을 편집하여 문제를 일으킬 수 있는 설명이나 명령을 제거합니다.

cluster unreachable 오류 메시지와 함께 EKS 배포 작업이 실 패함

설명: EKS 배포 작업이 실행된 후 cluster unreachable 오류 메시지와 함께 작업이 실패합니다. 이 메시지에는 누락된 권한으로 인한 클러스터의 액세스 문제가 표시됩니다. 파일 유형(Helm 차트 또 는 Kubernetes 매니페스트 파일)에 따라 오류 메시지가 다음과 같이 표시됩니다.

• 차트 Helm을 사용하는 EKS 배포 작업의 경우 다음 오류 메시지와 유사한 오류가 표시됩니다.

```
helm upgrade --install my-release test-chart --wait
```

error message:

Error: Kubernetes cluster unreachable: the server has asked for the client to provide credentials

• Kubernetes 매니페스트 파일을 사용하는 EKS 배포 작업의 경우 다음 오류 메시지와 유사한 오류가 표시됩니다.

kubectl apply -f deployment.yaml
Error: error validating "deployment.yaml": error validating data: failed to download
openapi: the server has asked for the client to provide credentials

결과: 파이프라인에서 배포 작업이 실패합니다.

수정 방법: 기존 역할을 사용하는 경우 EKS 배포 작업을 사용하는 데 필요한 권한으로 CodePipeline 서비스 역할을 업데이트해야 합니다. 또한 클러스터에 대한 CodePipeline 서비스 역할 액세스를 허용 하려면 클러스터에 액세스 항목을 추가하고 액세스 항목에 대한 서비스 역할을 지정해야 합니다.

- 1. CodePipeline 서비스 역할에 EKS 배포 작업에 필요한 권한이 있는지 확인합니다. 권한 참조는 섹션 을 참조하세요서비스 역할 정책 권한.
- 2. 클러스터에 액세스 항목을 추가하고 액세스에 대한 CodePipeline 서비스 역할을 지정합니다. 예시 는 4단계: CodePipeline 서비스 역할에 대한 액세스 항목 생성에서 확인하십시오.

다른 문제에 도움이 필요하십니까?

다른 리소스를 시도해 보십시오.

- AWS Support에 문의하세요.
- CodePipeline 포럼에 질문하세요.
- 할당량 증가를 요청하십시오. 자세한 내용은 in AWS CodePipeline 할당량 단원을 참조하십시오.

Note

할당량 증대 요청을 처리하려면 최대 2주가 걸릴 수 있습니다.

의 보안 AWS CodePipeline

의 클라우드 보안 AWS 이 최우선 순위입니다. AWS 고객은 가장 보안에 민감한 조직의 요구 사항을 충 족하기 위해 bui It인 데이터 센터 및 네트워크 아키텍처의 이점을 누릴 수 있습니다.

보안은 AWS 와 사용자 간의 공동 책임입니다. <u>공동 책임 모델</u>은 이 사항을 클라우드 내 보안 및 클라우 드의 보안으로 설명합니다.

- 클라우드 보안 AWS 는 AWS 서비스 에서 실행되는 인프라를 보호할 책임이 있습니다 AWS 클 라우드. AWS 또한는 안전하게 사용할 수 있는 서비스를 제공합니다. 서드 파티 감사원은 정기적 으로 <u>AWS 규정 준수 프로그램</u>의 일환으로 보안 효과를 테스트하고 검증합니다. 에 적용되는 규 정 준수 프로그램에 대한 자세한 내용은 <u>AWS 서비스 규정 준수 프로그램별 범위 내</u> 섹션을 AWS CodePipeline참조하세요.
- 클라우드의 보안 사용자의 책임은 AWS 서비스 사용하는에 따라 결정됩니다. 또한 사용자는 데이 터의 민감도, 회사 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 CodePipeline 사용 시 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다 음 주제에서는 보안 및 규정 준수 목표를 충족하도록 CodePipeline를 구성하는 방법을 보여줍니다. 또 한 CodePipeline 리소스를 모니터링하고 보호하는 데 도움이 AWS 서비스 되는 다른를 사용하는 방법 을 알아봅니다.

주제

- 의 데이터 보호 AWS CodePipeline
- AWS CodePipeline의 Identity and Access Management(IAM)
- CodePipeline에서 로깅 및 모니터링
- 에 대한 규정 준수 검증 AWS CodePipeline
- <u>의 복원력 AWS CodePipeline</u>
- <u>의 인프라 보안 AWS CodePipeline</u>
- <u>보안 모범 사례</u>

의 데이터 보호 AWS CodePipeline

AWS <u>공동 책임 모델</u>의 데이터 보호에 적용됩니다 AWS CodePipeline. 이 모델에 설명된 대로 AWS 는 모든를 실행하는 글로벌 인프라를 보호할 책임이 있습니다 AWS 클라우드. 사용자는 인프라에서 호스 팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스 의 보안 구성과 관리 태스크에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 <u>데이터 프라이버시 FAQ</u>를 참조하세 요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 <u>AWS 공동 책임 모델 및 GDPR</u> 블 로그 게시물을 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 사 용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데 이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용하세요.
- 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 <u>CloudTrail 추적</u> 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 고 급 관리형 보안 서비스를 사용하세요.
- 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 <u>Federal</u> Information Processing Standard(FIPS) 140-3을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 형식 텍스트 필 드에 입력하지 않는 것이 좋습니다. 여기에는 CodePipeline 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 자격 증명을 URL에 포함해서는 안 됩니다.

다음 보안 모범 사례에서도 CodePipeline의 데이터 보호를 다룹니다.

- CodePipeline의 Amazon S3에 저장된 아티팩트에 대해 서버 측 암호화 구성
- AWS Secrets Manager 를 사용하여 데이터베이스 암호 또는 타사 API 키 추적

인터네트워크 트래픽 개인 정보

Amazon VPC는 정의 AWS 서비스 한 가상 네트워크(가상 프라이빗 클라우드)에서 AWS 리소스 를 시작하는 데 사용할 수 있는 입니다. CodePipeline은 프라이빗 IP 주소와 함께 탄력적 네트워 크 인터페이스를 AWS 서비스 사용하는 간의 프라이빗 통신을 용이하게 하는 AWS 기술인 AWS PrivateLink로 구동되는 Amazon VPC 엔드포인트를 지원합니다. 즉, VPC의 프라이빗 엔드포인트를 통해 CodePipeline에 직접 연결하여 VPC 및 AWS 네트워크 내의 모든 트래픽을 유지할 수 있습니다. 이전에는 VPC 내에서 실행 중인 애플리케이션에서 CodePipeline에 연결하기 위해 인터넷 접속이 필 요했습니다. VPC를 사용하면 다음과 같은 네트워크 설정을 제어할 수 있습니다.

- IP 주소 범위,
- 서브넷,
- 라우팅 테이블,
- 네트워크 게이트웨이

VPC를 CodePipeline에 연결하려면 CodePipeline에 대해 인터페이스 VPC 엔드포인트를 정의하세요. 이 유형의 엔드포인트를 사용하여 VPC를 AWS 서비스에 연결할 수 있습니다. 이 엔드포인트를 이용하 면 인터넷 게이트웨이나 NAT(네트워크 주소 변환) 인스턴스 또는 VPN 연결 없이도 CodePipeline에 안 정적이고 확장 가능하게 연결됩니다. VPC 설정에 대한 자세한 내용은 <u>VPC 사용 설명서</u>를 참조하십시 오.

저장 시 암호화

CodePipeline의 데이터는를 사용하여 저장 시 암호화됩니다 AWS KMS keys. 코드 아티팩트는 고 객 소유 S3 버킷에 저장되며 AWS 관리형 키 또는 고객 관리형 키로 암호화됩니다. 자세한 내용은 CodePipeline의 Amazon S3에 저장된 아티팩트에 대해 서버 측 암호화 구성 단원을 참조하십시오.

전송 중 데이터 암호화

모든 서비스 대 서비스 통신은 SSL/TLS를 사용하여 전송 중에 암호화됩니다.

암호화 키 관리

코드 아티팩트를 암호화하기 위한 기본 옵션을 선택하면 CodePipeline은 AWS 관리형 키를 사용합니다. 이를 변경하거나 삭제할 수 없습니다 AWS 관리형 키. 에서 고객 관리형 키를 사용하여 S3 버킷의 아티팩트를 암호화하거나 AWS KMS 복호화하는 경우 필요에 따라이 고객 관리형 키를 변경하거나 교체할 수 있습니다.

▲ Important

CodePipeline은 대칭 KMS 키만 지원합니다. 비대칭 KMS 키를 사용하여 S3 버킷의 데이터를 암호화하지 마십시오.

CodePipeline의 Amazon S3에 저장된 아티팩트에 대해 서버 측 암호화 구성

Amazon S3 아티팩트에 대해 서버 측 암호화를 구성하는 방법은 2가지입니다.

- CodePipeline은 파이프라인 생성 마법사를 사용하여 파이프라인을 생성할 AWS 관리형 키 때 S3 아 티팩트 버킷과 기본값을 생성합니다. AWS 관리형 키 는 객체 데이터와 함께 암호화되고에서 관리합 니다 AWS.
- 자체 고객 관리형 키를 만들고 관리할 수 있습니다.

A Important

CodePipeline은 대칭 KMS 키만 지원합니다. 비대칭 KMS 키를 사용하여 S3 버킷의 데이터를 암호화하지 마십시오.

기본 S3 키를 사용 중이라면 이 AWS 관리형 키를 변경하거나 삭제할 수 없습니다. 에서 고객 관리형 키를 사용하여 S3 버킷의 아티팩트를 암호화하거나 AWS KMS 해독하는 경우 필요에 따라이 고객 관 리형 키를 변경하거나 교체할 수 있습니다.

Amazon S3는 버킷에 저장된 모든 객체에 대해 서버 측 암호화가 필요할 경우 사용할 수 있는 버킷 정 책을 지원합니다. 예를 들어, 다음 버킷 정책은 요청에 SSE-KMS를 사용한 서버 측 암호화를 요청하 는 s3:PutObject 헤더가 포함되지 않을 경우 모든 사용자에게 객체 업로드(x-amz-server-sideencryption) 권한을 거부합니다.

```
{
    "Version": "2012-10-17",
    "Id": "SSEAndSSLPolicy",
    "Statement": [
        {
            "Sid": "DenyUnEncryptedObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
        }
}
```

```
"Resource": "arn:aws:s3:::codepipeline-us-west-2-89050EXAMPLE/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-server-side-encryption": "aws:kms"
                }
            }
        },
        {
            "Sid": "DenyInsecureConnections",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::codepipeline-us-west-2-89050EXAMPLE/*",
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "false"
                }
            }
        }
    ]
}
```

서버 측 암호화에 대한 자세한 내용은 <u>서버 측 암호화를 사용하여 데이터 보호</u> 및 (SSE-KMS)에 저 장된 KMS 키를 사용한 서버 측 암호화를 사용하여 데이터 보호를 AWS KMS참조하세요. <u>AWS Key</u> Management Service

에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서를 AWS KMS참조하세요.

주제

- 보기 AWS 관리형 키
- AWS CloudFormation 또는를 사용하여 S3 버킷에 대한 서버 측 암호화 구성 AWS CLI

보기 AWS 관리형 키

파이프라인 생성 마법사로 첫 번째 파이프라인을 만들 때 파이프라인을 만든 것과 동일한 리전에 S3 버킷이 생성됩니다. 버킷은 파이프라인 아티팩트를 저장하는 데 사용됩니다. 파이프라인을 실행하면 S3 버킷에서 아티팩트를 검색해서 배치합니다. 기본적으로 CodePipeline은 Amazon S3 AWS 관리형 키 용 (aws/s3키)를 AWS KMS 사용하여와 함께 서버 측 암호화를 사용합니다. AWS 관리형 키 이는 생성되어 AWS 계정에 저장됩니다. S3 버킷에서 아티팩트를 검색하면 CodePipeline이 동일한 SSE-KMS 프로세스를 이용해 아티팩트를 암호화합니다. 에 대한 정보를 보려면 AWS 관리형 키

사용자 가이드

- 1. 에 로그인 AWS Management Console 하고 AWS KMS 콘솔을 엽니다.
- 2. 시작 페이지가 나타나면 지금 시작을 선택합니다.
- 3. 서비스 탐색 창에서 AWS 관리형 키를 선택합니다.
- 파이프라인의 리전을 선택합니다. 예를 들어 파이프라인이 us-east-2에 생성된 경우 필터는 미 국 동부(오하이오)로 설정해야 합니다.

CodePipeline에 사용할 수 있는 리전과 엔드포인트에 대한 자세한 내용은 <u>AWS CodePipeline 엔</u> 드포인트 및 할당량을 참조하세요.

5. 목록에서 파이프라인에 대해 사용된 별칭이 있는 키를 선택합니다(기본값: aws/s3). 키에 대한 기 본 정보가 표시됩니다.

AWS CloudFormation 또는를 사용하여 S3 버킷에 대한 서버 측 암호화 구성 AWS CLI

AWS CloudFormation 또는를 사용하여 파이프라인 AWS CLI 을 생성하는 경우 서버 측 암호화를 수동 으로 구성해야 합니다. 위의 샘플 버킷 정책을 사용한 다음 자체 고객 관리형 키를 생성합니다. AWS 관 리형 키대신 자체 키를 사용해도 됩니다. 자체 키를 선택해야 하는 몇 가지 이유는 다음과 같습니다.

- 조직의 비즈니스 또는 보안 요건에 맞추기 위해 일정에 따라 키를 교체하고 싶습니다.
- 다른 AWS 계정에 연결된 리소스를 사용하는 파이프라인을 만들고 싶습니다. 이때 고객 관리형 키를 사용해야 합니다. 자세한 내용은 <u>CodePipeline에서 다른 AWS 계정의 리소스를 사용하는 파이프라</u> <u>인 생성</u> 단원을 참조하십시오.

암호화 모범 사례에 따르면 암호화 키를 광범위하게 사용하지 않는 것이 좋습니다. 키를 정기적으로 교 체하는 것이 좋습니다. AWS KMS 키에 대한 새 암호화 구성 요소를 생성하려면 고객 관리형 키를 생성 한 다음 새 고객 관리형 키를 사용하도록 애플리케이션 또는 별칭을 변경할 수 있습니다. 또는 기존 고 객 관리형 키에 대해 자동 키 교체를 활성화할 수 있습니다.

고객 관리 키를 교체하려면 키 교체를 참조하세요.

A Important

CodePipeline은 대칭 KMS 키만 지원합니다. 비대칭 KMS 키를 사용하여 S3 버킷의 데이터를 암호화하지 마십시오.

AWS Secrets Manager 를 사용하여 데이터베이스 암호 또는 타사 API 키 추 적

수명 주기 동안 AWS Secrets Manager 를 사용하여 데이터베이스 자격 증명, API 키 및 기타 보안 암 호를 교체, 관리 및 검색하는 것이 좋습니다. Secrets Manager는 코드의 암호를 포함해 하드 코딩된 자 격 증명을 Secrets Manager에서 프로그래밍 방식으로 보안 암호를 검색하도록 하는 API 직접 호출로 바꿀 수 있습니다. 자세한 내용은 <u>AWS Secrets Manager 사용 설명서의 Secrets Manager란 무엇입니</u> 까?AWS 를 참조하세요.

AWS CloudFormation 템플릿에 보안 암호인 파라미터(예: OAuth 자격 증명)를 전달하는 파이프라인의 경우 Secrets Manager에 저장한 보안 암호에 액세스하는 동적 참조를 템플릿에 포함해야 합니다. 참 조 ID 패턴 및 예제는 AWS CloudFormation 사용 설명서의 <u>Secrets Manager 보안 암호</u>를 참조하세요. 파이프라인의 GitHub Webhook에 대한 템플릿 스니펫에 동적 참조를 사용하는 예제의 경우, <u>Webhook</u> <u>리소스 구성</u> 단원을 참조하십시오.

다음 사항도 참조하세요.

다음 표에는 암호를 관리할 때 참조할 수 있는 관련 리소스가 나와 있습니다.

- Secrets Manager는 Amazon RDS 암호 교체와 같이 데이터베이스 자격 증명을 자동으로 교체할 수 있습니다. 자세한 내용은 <u>AWS Secrets Manager 사용 설명서의 Secrets Manager 보안 암호 교체</u>를 참조하세요. AWS
- AWS CloudFormation 템플릿에 Secrets Manager 동적 참조를 추가하는 방법에 대한 지침은 <u>https://</u> <u>aws.amazon.com/blogs/security/how-to-create-and-retrieve-secrets-managed-in-aws-secrets-</u> manager-using-aws-cloudformation-template/을 참조하십시오.

AWS CodePipeline의 Identity and Access Management(IAM)

AWS Identity and Access Management (IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제 어하는 데 도움이 AWS 서비스 되는 입니다. IAM 관리자는 어떤 사용자가 CodePipeline 리소스를 사용 할 수 있는 인증(로그인) 및 권한(권한 있음)을 받을 수 있는지 제어합니다. IAM은 추가 비용 없이 사용 할 수 AWS 서비스 있는 입니다.

주제

- <u>대상</u>
- <u>ID를 통한 인증</u>

- 정책을 사용하여 액세스 관리
- AWS CodePipeline 에서 IAM을 사용하는 방법
- AWS CodePipeline 자격 증명 기반 정책 예시
- AWS CodePipeline 리소스 기반 정책 예제
- AWS CodePipeline ID 및 액세스 문제 해결
- CodePipeline 권한 참조
- CodePipeline 서비스 역할 관리

대상

AWS Identity and Access Management (IAM)를 사용하는 방법은 CodePipeline에서 수행하는 작업에 따라 다릅니다.

서비스 사용자 - CodePipeline 서비스를 사용하여 작업을 수행하는 경우 필요한 자격 증명과 권한을 관 리자가 제공합니다. 더 많은 CodePipeline 기능을 사용하여 작업을 수행하게 되면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방법을 이해하면 관리자에게 올바른 권한을 요청하는 데 도움이 됩니 다. CodePipeline의 기능에 액세스할 수 없는 경우 <u>AWS CodePipeline ID 및 액세스 문제 해결</u> 섹션을 참조하세요.

서비스 관리자 – 회사 내 CodePipeline 리소스를 책임지고 있는 경우 CodePipeline에 대한 전체 액세스 권한을 가지고 있을 것입니다. 서비스 관리자는 서비스 사용자가 액세스해야 하는 CodePipeline 기능 과 리소스를 결정합니다. 그런 다음 IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해 야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해하세요. 회사에서 CodePipeline와 함 께 IAM을 어떻게 사용할 수 있는지 자세히 알아보려면 <u>AWS CodePipeline 에서 IAM을 사용하는 방법</u> 섹션을 참조하세요.

IAM 관리자 - IAM 관리자인 경우 CodePipeline에 대한 액세스 권한 관리 정책을 작성하는 방법에 대해 자세히 알아보고 싶을 수 있습니다. IAM에서 사용할 수 있는 CodePipeline 자격 증명 기반 정책 예제를 보려면 AWS CodePipeline 자격 증명 기반 정책 예시 섹션을 참조하세요.

ID를 통한 인증

인증은 자격 증명 자격 증명을 AWS 사용하여에 로그인하는 방법입니다. IAM 사용자 또는 AWS 계정 루트 사용자 IAM 역할을 수임하여 로 인증(로그인 AWS)되어야 합니다.

자격 증명 소스를 통해 제공된 자격 증명을 사용하여 페더레이션 자격 증명 AWS 으로에 로그인할 수 있습니다. AWS IAM Identity Center (IAM Identity Center) 사용자, 회사의 Single Sign-On 인증 및 Google 또는 Facebook 자격 증명은 페더레이션 자격 증명의 예입니다. 페더레이션형 ID로 로그인
할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 페더레이션을 사용하여 AWS 에 액세스하면 간접적으로 역할을 수임하게 됩니다.

사용자 유형에 따라 AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. 로그 인에 대한 자세한 내용은 AWS 로그인 사용 설명서의에 로그인하는 방법을 AWS참조하세요. <u>AWS 계</u> 정

AWS 프로그래밍 방식으로에 액세스하는 경우는 자격 증명을 사용하여 요청에 암호화 방식으로 서명 할 수 있는 소프트웨어 개발 키트(SDK)와 명령줄 인터페이스(CLI)를 AWS 제공합니다. AWS 도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. 권장 방법을 사용하여 요청에 직접 서명하는 자세한 방법은 IAM 사용 설명서에서 API 요청용AWS Signature Version 4를 참조하세요.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어는 멀티 팩터 인증(MFA)을 사용하여 계정의 보안을 강화할 것을 AWS 권장합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 다중 인증 및 IAM 사용 설명서에서 IAM의AWS 다중 인증을 참조하세요.

AWS 계정 루트 사용자

를 생성할 때 계정의 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 하나의 로그인 자 격 증명으로 AWS 계정시작합니다. 이 자격 증명을 AWS 계정 루트 사용자라고 하며 계정을 생성하는 데 사용한 이메일 주소와 암호로 로그인하여 액세스합니다. 일상적인 작업에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명을 보호하고 루트 사용자만 수행할 수 있는 작업 을 수행하는 데 사용합니다. 루트 사용자로 로그인해야 하는 전체 작업 목록은 IAM 사용 설명서의 <u>루트</u> 사용자 자격 증명이 필요한 작업을 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 사용자 또는 애플리케이션에 대한 특정 권한이 AWS 계정 있는 내의 자격 증명입니 다. 가능하면 암호 및 액세스 키와 같은 장기 자격 증명이 있는 IAM 사용자를 생성하는 대신 임시 자격 증명을 사용하는 것이 좋습니다. 하지만 IAM 사용자의 장기 자격 증명이 필요한 특정 사용 사례가 있는 경우, 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 <u>장기 보안 인증이 필요</u> 한 사용 사례의 경우, 정기적으로 액세스 키 교체를 참조하세요.

IAM 그룹은 IAM 사용자 컬렉션을 지정하는 자격 증명입니다. 사용자는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용 자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어, IAMAdmins라는 그룹이 있고 이 그룹에 IAM 리소스를 관리할 권한을 부여할 수 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 장기 자격 증명을 가지고 있 지만, 역할은 임시 보안 인증만 제공합니다. 자세한 내용은 IAM 사용 설명서에서 <u>IAM 사용자 사용 사</u> 례를 참조하세요.

IAM 역할

IAM 역할은 특정 권한이 AWS 계정 있는 내의 자격 증명입니다. IAM 사용자와 유사하지만, 특정 개 인과 연결되지 않습니다. 에서 IAM 역할을 일시적으로 수임하려면 사용자에서 IAM 역할(콘솔)로 전 환할 AWS Management Console수 있습니다. <u>https://docs.aws.amazon.com/IAM/latest/UserGuide/</u> <u>id_roles_use_switch-role-console.html</u> 또는 AWS API 작업을 호출하거나 사용자 지정 URL을 AWS CLI 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 내용은 IAM 사용 설명서의 <u>역</u> 할 수임 방법을 참조하세요.

임시 보안 인증이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 페더레이션 사용자 액세스 페더레이션 ID에 권한을 부여하려면 역할을 생성하고 해당 역할의 권 한을 정의합니다. 페더레이션 ID가 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페 더레이션 관련 역할에 대한 자세한 내용은 IAM 사용 설명서의 <u>Create a role for a third-party identity</u> <u>provider (federation)</u>를 참조하세요. IAM Identity Center를 사용하는 경우, 권한 집합을 구성합니다. 인증 후 ID가 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 집합을 IAM의 역할 과 연관짓습니다. 권한 집합에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 <u>권한 집</u> 합을 참조하세요.
- 임시 IAM 사용자 권한 IAM 사용자 또는 역할은 IAM 역할을 수임하여 특정 작업에 대한 다양한 권 한을 임시로 받을 수 있습니다.
- 교차 계정 액세스 IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니 다. 그러나 일부 에서는 정책을 리소스에 직접 연결할 AWS 서비스수 있습니다(역할을 프록시로 사 용하는 대신). 교차 계정 액세스에 대한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 교차 계정 리소스 액세스를 참조하세요.
- 교차 서비스 액세스 일부는 다른에서 기능을 AWS 서비스 사용합니다 AWS 서비스. 예를 들어, 서비스에서 호출하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는 직접적으로 호출하는 위탁자의 권한을 사용하거나, 서 비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.
 - 전달 액세스 세션(FAS) IAM 사용자 또는 역할을 사용하여에서 작업을 수행하는 경우 AWS보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우, 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는를 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대 한 요청과 AWS 서비스함께 사용합니다. FAS 요청은 서비스가 완료하기 위해 다른 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 수신할 때만 이루어집니다. 이 경우, 두 작업을 모두

수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 <u>전달 액세스 세션</u>을 참조하 세요.

- 서비스 역할 서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하기 위해 맡는 <u>IAM 역할</u>입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 <u>Create a role to delegate permissions to an AWS 서비스</u>를 참조하세요.
- 서비스 연결 역할 서비스 연결 역할은에 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비 스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은에 표시 AWS 계정 되며 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지 만 편집은 할 수 없습니다.
- Amazon EC2에서 실행되는 애플리케이션 IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 AWS CLI 또는 AWS API 요청을 수행하는 애플리케이션의 임시 자격 증명을 관리할 수 있습니다. 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS 역할을 할 당하고 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로파일 을 생성합니다. 인스턴스 프로필에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그 램이 임시 보안 인증을 얻을 수 있습니다. 자세한 정보는 IAM 사용 설명서의 <u>IAM 역할을 사용하여</u> Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하세요.

정책을 사용하여 액세스 관리

정책을 AWS 생성하고 자격 증명 또는 리소스에 연결하여 AWS 에서 액세스를 제어합니다. 정책은 자 격 증명 또는 리소스와 연결 AWS 될 때 권한을 정의하는의 객체입니다.는 보안 주체(사용자, 루트 사 용자 또는 역할 세션)가 요청할 때 이러한 정책을 AWS 평가합니다. 정책에서 권한은 요청이 허용되거 나 거부되는 지를 결정합니다. 대부분의 정책은에 JSON 문서 AWS 로 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 정보는 IAM 사용 설명서의 JSON 정책 개요를 참조하세요.

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

기본적으로, 사용자 및 역할에는 어떠한 권한도 없습니다. 사용자에게 사용자가 필요한 리소스에서 작 업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 수임할 수 있습니다.

IAM 정책은 작업을 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, iam:GetRole 작업을 허용하는 정책이 있다고 가정합니다. 해당 정책이 있는 사용자는 AWS Management Console AWS CLI, 또는 API에서 역할 정보를 가져올 수 있습니다 AWS .

ID 기반 정책

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서 입니다. 이러한 정책은 사용자 및 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지 를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 <u>고객 관리형</u> 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

ID 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사 용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은의 여러 사용자, 그룹 및 역할에 연결할 수 있는 독립 실행형 정책입니다 AWS 계정. 관리형 정책에는 AWS 관리형 정책 및 고객 관리형 정책이 포함됩 니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 <u>관리형 정책 및</u> 인라인 정책 중에서 선택을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예제는 IAM 역 할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자 는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 위탁자가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니 다. 리소스 기반 정책에서 <u>위탁자를 지정</u>해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사 용자 또는이 포함될 수 있습니다 AWS 서비스.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS 관리 형 정책을 사용할 수 없습니다.

기타 정책 타입

AWS 는 덜 일반적인 추가 정책 유형을 지원합니다. 이러한 정책 타입은 더 일반적인 정책 유형에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 권한 경계는 ID 기반 정책에 따라 IAM 엔터티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻 는 권한은 객체의 ID 기반 정책과 그 권한 경계의 교집합입니다. Principal 필드에서 사용자나 역 할을 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포 함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 정보는 IAM 사용 설명서의 IAM 엔티티에 대한 권한 경계를 참조하세요.
- 서비스 제어 정책(SCPs) SCPs는의 조직 또는 조직 단위(OU)에 대한 최대 권한을 지정하는 JSON 정책입니다 AWS Organizations. AWS Organizations 는 기업이 소유한 여러 AWS 계정 을 그룹화

하고 중앙에서 관리하기 위한 서비스입니다. 조직에서 모든 기능을 활성화할 경우, 서비스 제어 정 책(SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 각각을 포함하여 멤버 계정의 엔 터티에 대한 권한을 제한합니다 AWS 계정 루트 사용자. 조직 및 SCP에 대한 자세한 내용은 AWS Organizations 사용 설명서에서 Service control policies을 참조하세요.

- 리소스 제어 정책(RCP) RCP는 소유한 각 리소스에 연결된 IAM 정책을 업데이트하지 않고 계정의 리소스에 대해 사용 가능한 최대 권한을 설정하는 데 사용할 수 있는 JSON 정책입니다. RCP는 멤버 계정의 리소스에 대한 권한을 제한하며 조직에 속하는지 여부에 AWS 계정 루트 사용자관계없이를 포함한 자격 증명에 대한 유효 권한에 영향을 미칠 수 있습니다. RCP를 AWS 서비스 지원하는 목록 을 포함하여 조직 및 RCPs에 대한 자세한 내용은 AWS Organizations 사용 설명서의 <u>리소스 제어 정</u> 책(RCPs)을 참조하세요.
- 세션 정책 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생 성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 ID 기반 정책의 교차와 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 정보는 IAM 사용 설명 서의 세션 정책을 참조하세요.

AWS CodePipeline 에서 IAM을 사용하는 방법

IAM을 사용하여 CodePipeline에 대한 액세스를 관리하려면 먼저 어떤 IAM 기능을 CodePipeline에 사 용할 수 있는지를 이해해야 합니다. CodePipeline 및 IAM으로 작업 AWS 서비스 하는 기타를 전체적으 로 알아보려면 IAM 사용 설명서의 AWS 서비스 IAM으로 작업하는 섹션을 참조하세요.

주제

- CodePipeline 자격 증명 기반 정책
- CodePipeline 리소스 기반 정책
- CodePipeline 태그 기반 권한 부여
- CodePipeline IAM 역할

CodePipeline 자격 증명 기반 정책

IAM ID 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스뿐 아니라 작업이 허용되거나 거부 되는 조건을 지정할 수 있습니다. CodePipeline은 특정 작업, 리소스 및 조건 키를 지원합니다. JSON 정책에서 사용하는 모든 요소에 대해 알고 싶다면 IAM 사용 설명서의 <u>IAM JSON 정책 요소 참조</u>를 참 조하세요. 작업

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 위탁자가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 Action 요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명 합니다. 정책 작업은 일반적으로 연결된 AWS API 작업과 이름이 동일합니다. 일치하는 API 작업이 없 는 권한 전용 작업 같은 몇 가지 예외도 있습니다. 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니 다. 이러한 추가 작업을 일컬어 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하세요.

CodePipeline의 정책 작업은 작업 앞에 codepipeline: 접두사를 사용합니다.

예를 들어 누군가에게 계정 내에 있는 기존 파이프라인을 볼 수 있는 권한을 부여하려면 정책에 codepipeline:GetPipeline 작업을 포함시킵니다. 정책 문에는 Action 또는 NotAction 요소가 포함되어야 합니다. CodePipeline은 이 서비스로 수행할 수 있는 작업을 설명하는 고유한 작업 세트를 정의합니다.

명령문 하나에 여러 태스크를 지정하려면 다음과 같이 쉼표로 구분합니다.

"Action": ["codepipeline:*action1*", "codepipeline:*action2*"

와일드카드(*)를 사용하여 여러 작업을 지정할 수 있습니다. 예를 들어, Get라는 단어로 시작하는 모든 태스크를 지정하려면 다음 태스크를 포함합니다.

"Action": "codepipeline:Get*"

CodePipeline 작업 목록에 대해서는 IAM 사용 설명서의 <u>AWS CodePipeline에서 정의한 작업</u>을 참조하 세요.

리소스

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

Resource JSON 정책 요소는 작업이 적용되는 하나 이상의 객체를 지정합니다. 문에는 Resource또 는 NotResource요소가 반드시 추가되어야 합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을

사용하여 리소스를 지정합니다. 리소스 수준 권한이라고 하는 특정 리소스 유형을 지원하는 작업에 대 해 이를 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

"Resource": "*"

CodePipeline 리소스 및 작업

CodePipeline에서는 파이프라인이 기본 리소스입니다. 정책에서 Amazon 리소스 이름(ARN)을 사용하 여 정책이 적용되는 리소스를 식별합니다. CodePipeline은 기본 리소스와 함께 사용할 수 있는 다른 리 소스(예: 단계, 작업, 사용자 지정 작업)를 지원합니다. 이러한 리소스를 가리켜 하위 리소스라 합니다. 이러한 리소스와 하위 리소스에는 고유한 Amazon 리소스 이름(ARN)이 연결됩니다. ARNs에 대한 자 세한 내용은의 <u>Amazon 리소스 이름(ARN) 및 AWS 서비스 네임스페이스</u>를 참조하세요Amazon Web Services 일반 참조. 파이프라인과 연결된 파이프라인 ARN을 가져오려면 콘솔의 설정에서 파이프라인 ARN을 찾을 수 있습니다. 자세한 내용은 <u>파이프라인 ARN 및 서비스 역할 ARN 보기(콘솔)</u> 단원을 참 조하십시오.

리소스 유형	ARN 형식
파이프라인	arn:aws:codepipeline: <i>region:account:pipeline-name</i>
단계	<pre>arn:aws:codepipeline:region:account:pipeline-name /stage- name</pre>
작업	<pre>arn:aws:codepipeline:region:account:pipeline-name /stage- name /action-name</pre>
사용자 지정 작업	arn:aws:codepipeline: <i>region:account</i> :actionty pe: <i>owner/category/provider/version</i>
모든 CodePipeline 리소 스	arn:aws:codepipeline:*
지정한 리전에서 지정 한 계정이 소유한 모든 CodePipeline 리소스	arn:aws:codepipeline: <i>region:account</i> :*

Note

의 대부분의 서비스는 콜론(:) 또는 슬래시(/)를 ARN에서 동일한 문자로 AWS 취급합니다. ARNs 그러나 CodePipeline은 이벤트 패턴 및 규칙에서 정확히 일치를 사용합니다. 따라서 이 벤트 패턴을 만들 때 일치시키려는 파이프라인에서 ARN 구문이 일치하도록 정확한 ARN 문자 를 사용해야 합니다.

CodePipeline에는 리소스 수준 권한을 지원하는 API 직접 호출이 있습니다. 리소스 수준 권한은 API 호출이 리소스 ARN을 지정할 수 있는지, 또는 API 호출이 와일드카드를 사용해야만 모든 리소스를 지 정할 수 있는지 나타냅니다. 리소스 수준 권한과 리소스 수준 권한을 지원하는 CodePipeline API 직접 호출 목록에 대한 자세한 설명은 <u>CodePipeline 권한 참조</u> 단원을 참조하세요.

예를 들어 명령문에서 다음과 같이 ARN을 사용하여 특정 파이프라인(*myPipeline*)을 나타낼 수 있습니다.

"Resource": "arn:aws:codepipeline:us-east-2:111222333444:myPipeline"

다음과 같이 (*) 와일드카드 문자를 사용하여 특정 계정에 속하는 모든 파이프라인을 지정할 수도 있습니다.

"Resource": "arn:aws:codepipeline:us-east-2:111222333444:*"

모든 리소스를 지정해야 하거나 특정 API 작업이 ARN을 지원하지 않는 경우 다음과 같이 Resource 요소에 와일드카드 문자(*)를 사용합니다.

"Resource": "*"

Note

IAM 정책을 만들 때 최소 권한 부여의 기본 보안 조언을 따릅니다. 즉, 작업 수행에 필요한 최소 한의 권한만 부여합니다. API 호출이 ARN을 지원하는 경우 리소스 수준 권한을 지원하므로 와 일드카드 문자(*)를 사용할 필요가 없습니다.

일부 CodePipeline API 직접 호출에서는 여러 리소스 사용을 허용합니다(예: GetPipeline). 명령문 하나에 여러 리소스를 지정하려면 다음과 같이 각 ARN을 쉼표로 구분합니다. "Resource": ["arn1", "arn2"]

CodePipeline는 CodePipeline 리소스를 사용하기 위한 일련의 작업을 제공합니다. 사용 가능한 작업 목록은 <u>CodePipeline 권한 참조</u> 섹션을 참조하십시오.

조건 키

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

Condition 요소(또는 Condition 블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. Condition 요소는 옵션입니다. 같거나 작음과 같은 <u>조건 연산자</u>를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다.

한 문에서 여러 Condition 요소를 지정하거나 단일 Condition 요소에서 여러 키를 지정하는 경우, AWS 는 논리적 AND 작업을 사용하여 평가합니다. 단일 조건 키에 여러 값을 지정하는 경우는 논리적 OR 작업을 사용하여 조건을 AWS 평가합니다. 문의 권한을 부여하기 전에 모든 조건을 충족해야 합니 다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 예를 들어, IAM 사용자에게 IAM 사용자 이름으로 태그가 지정된 경우에만 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 자세한 내용 은 IAM 사용 설명서의 IAM 정책 요소: 변수 및 태그를 참조하세요.

AWS 는 전역 조건 키와 서비스별 조건 키를 지원합니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

CodePipeline에서는 자체 조건 키 집합을 정의하고 일부 전역 조건 키 사용도 지원합니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

모든 Amazon EC2 작업은 aws:RequestedRegion 및 ec2:Region 조건 키를 지원합니다. 자세한 내용은 예제: 특정 리전으로 액세스 제한을 참조하세요.

CodePipeline 조건 키 목록을 보려면 IAM 사용자 설명서에서 <u>AWS CodePipeline에 대한 조건</u> <u>키</u>를 참조하세요. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 <u>에서 정의한 작업을 AWS</u> <u>CodePipeline</u> 참조하세요.

예시

CodePipeline 자격 증명 기반 정책 예제를 보려면 <u>AWS CodePipeline 자격 증명 기반 정책 예시</u> 섹션 을 참조하세요.

CodePipeline 리소스 기반 정책

CodePipeline은 리소스 기반 정책을 지원하지 않습니다. 그러나 CodePipeline과 관련된 S3 서비스에 대한 리소스 기반 정책의 예가 제공됩니다.

예시

CodePipeline 리소스 기반 정책의 예는 AWS CodePipeline 리소스 기반 정책 예제 섹션을 참조하세요.

CodePipeline 태그 기반 권한 부여

태그를 CodePipeline 리소스에 연결하거나 요청을 통해 태그를 CodePipeline에 전달할 수 있 습니다. 태그에 근거하여 액세스를 제어하려면 codepipeline:ResourceTag/*key-name*, aws:RequestTag/*key-name* 또는 aws:TagKeys 조건 키를 사용하여 정책의 <u>조건 요소</u>에 태그 정 보를 제공합니다. CodePipeline 리소스 태깅에 대한 자세한 내용은 <u>리소스에 태그 지정</u> 섹션을 참조하 세요.

리소스의 태그를 기반으로 리소스에 대한 액세스를 제한하는 자격 증명 기반 정책의 예시는 <u>태그를 사</u>용하여 CodePipeline 리소스에 대한 액세스 통제에서 확인할 수 있습니다.

CodePipeline IAM 역할

IAM 역할은 특정 권한이 있는 AWS 계정의 엔터티입니다.

CodePipeline에서 임시 자격 증명 사용

임시 보안 인증을 사용하여 페더레이션을 통해 로그인하거나, IAM 역할을 맡거나, 교차 계정 역할을 맡 을 수 있습니다. <u>AssumeRole</u> 또는 <u>GetFederationToken</u>과 같은 AWS STS API 작업을 호출하여 임시 보안 자격 증명을 얻습니다.

CodePipeline에서 임시 자격 증명 사용을 지원합니다.

서비스 역할

CodePipeline을 사용하면 서비스가 사용자를 대신해 <u>서비스 역할</u>을 맡을 수 있습니다. 이 역할을 사용 하면 서비스가 다른 서비스의 리소스에 액세스해 사용자를 대신해 작업을 완료할 수 있습니다. 서비스 역할은 IAM 계정에 나타나고, 해당 계정이 소유합니다. 즉, IAM 관리자가 이 역할에 대한 권한을 변경 할 수 있습니다. 그러나 권한을 변경하면 서비스의 기능이 손상될 수 있습니다.

CodePipeline은 서비스 역할을 지원합니다.

AWS CodePipeline 자격 증명 기반 정책 예시

기본적으로 IAM 사용자 및 역할은 CodePipeline 리소스를 생성하거나 수정할 수 있는 권한이 없습니 다. 또한 AWS Management Console AWS CLI또는 AWS API를 사용하여 작업을 수행할 수 없습니다. IAM 관리자는 지정된 리소스에서 특정 API 작업을 수행할 수 있는 권한을 사용자와 역할에게 부여하는 IAM 정책을 생성해야 합니다. 그런 다음 관리자는 해당 권한이 필요한 IAM 사용자 또는 그룹에 이러한 정책을 연결해야 합니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 JSON 탭에서 정책 생성을 참조하세요.

다른 계정의 리소스를 사용하는 파이프라인을 생성하는 방법과 관련 예제 정책은 <u>CodePipeline에서 다</u> 른 AWS 계정의 리소스를 사용하는 파이프라인 생성 섹션을 참조하세요.

주제

- 정책 모범 사례
- 콘솔에서 리소스 보기
- 사용자가 자신의 고유한 권한을 볼 수 있도록 허용
- 자격 증명 기반 정책(IAM)의 예
- 태그를 사용하여 CodePipeline 리소스에 대한 액세스 통제
- CodePipeline 콘솔 사용에 필요한 권한
- CodePipeline 콘솔에서 컴퓨팅 로그를 보는 데 필요한 권한
- AWS 에 대한 관리형 정책 AWS CodePipeline
- 고객 관리형 정책 예

정책 모범 사례

ID 기반 정책에 따라 계정에서 사용자가 CodePipeline 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하 거나 편집할 때는 다음 지침과 권장 사항을 따릅니다.

 AWS 관리형 정책을 시작하고 최소 권한으로 전환 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반적인 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. 에서 사용할 수 있습니다 AWS 계정. 사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 추가로 줄이는 것 이 좋습니다. 자세한 정보는 IAM 사용 설명서의 <u>AWS 관리형 정책</u> 또는 <u>AWS 직무에 대한 관리형 정</u> 책을 참조하세요.

- 최소 권한 적용 IAM 정책을 사용하여 권한을 설정하는 경우, 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있 는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명 서에 있는 IAM의 정책 및 권한을 참조하세요.
- IAM 정책의 조건을 사용하여 액세스 추가 제한 정책에 조건을 추가하여 작업 및 리소스에 대한 액 세스를 제한할 수 있습니다. 예를 들어, SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정 책 조건을 작성할 수 있습니다. AWS 서비스와 같은 특정를 통해 사용되는 경우 조건을 사용하여 서 비스 작업에 대한 액세스 권한을 부여할 수도 있습니다 AWS CloudFormation. 자세한 정보는 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 확인하여 안전하고 기능적인 권한 보장 IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 새로운 및 기존 정책을 확인합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하 여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 내용은 IAM 사용 설명서의 <u>IAM Access</u> <u>Analyzer에서 정책 검증</u>을 참조하세요.
- 다중 인증(MFA) 필요 -에서 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안 을 위해 MFA를 AWS 계정켭니다. API 작업을 직접 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 내용은 IAM 사용 설명서의 MFA를 통한 보안 API 액세스를 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

콘솔에서 리소스 보기

CodePipeline 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 AWS 계정에서 수행됩니다. 다음 서비스에 표시되는 리소스는 다음과 같습니다.

- AWS CodeBuild: 빌드 프로젝트
- AWS CodeCommit: 리포지토리
- AWS CodeDeploy: 애플리케이션
- AWS CodePipeline: 파이프라인

모든 서비스의 리소스에서 이 검색을 수행하려면 다음 권한이 있어야 합니다.

- CodeBuild: ListProjects
- CodeCommit: ListRepositories

- CodeDeploy: ListApplications
- CodePipeline: ListPipelines

해당 서비스에 대한 권한이 없는 경우 서비스의 리소스에 대한 결과가 반환되지 않습니다. 리소스를 볼 수 있는 권한이 있더라도 해당 리소스 보기에 대한 명시적 Deny가 있으면 일부 리소스가 반환되지 않 습니다.

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예제는 IAM 사용자가 자신의 사용자 ID에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 AWS CLI 또는 AWS API를 사용하여 프로그래밍 방식으로이 작업을 완료할 수 있는 권한이 포함됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            1,
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
```

}

] }

자격 증명 기반 정책(IAM)의 예

정책을 IAM 보안 인증에 연결할 수 있습니다. 예를 들면, 다음을 수행할 수 있습니다:

- 계정 내 사용자 또는 그룹에 권한 정책 연결 CodePipeline 콘솔에서 파이프라인을 볼 수 있는 사용 자 권한을 부여하려면 권한 정책을 사용자 또는 해당 사용자가 속한 그룹에 연결하면 됩니다.
- 역할에 권한 정책 연결(교차 계정 권한 부여) ID 기반 권한 정책을 IAM 역할에 연결하여 교차 계정 권한을 부여할 수 있습니다. 예를 들어 계정 A의 관리자는 다음과 AWS 서비스 같이 다른 AWS 계정 (예: 계정 B) 또는에 교차 계정 권한을 부여하는 역할을 생성할 수 있습니다.
 - 1. 계정 A 관리자는 IAM 역할을 생성하고 계정 A의 리소스에 대한 권한을 부여하는 역할에 권한 정 책을 연결합니다.
 - 2. 계정 A 관리자는 계정 B를 역할을 수임할 보안 주체로 식별하는 역할에 신뢰 정책을 연결합니다.
 - 3. 그런 다음 계정 B 관리자는 계정 B의 사용자에게 역할을 수임할 권한을 위임할 수 있습니다. 이렇 게 하면 계정 B의 사용자가 계정 A에서 리소스를 생성하거나 액세스할 수 있습니다. 역할을 수임 할 AWS 서비스 권한을 부여하려는 경우 신뢰 정책의 보안 주체가 보안 AWS 서비스 주체가 될 수 도 있습니다.

IAM을 사용하여 권한을 위임하는 방법에 대한 자세한 설명은 IAM 사용자 가이드의 <u>액세스 관리</u> 섹 션을 참조하세요.

다음은 권한 정책의 예시입니다. 이 정책은 사용자는 us-west-2 region에서 MyFirstPipeline이라는 이름의 파이프라인에서 모든 단계 간 전환을 활성화하고 비활성화할 권한 을 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement" : [
        {
          "Effect" : "Allow",
          "Action" : [
          "codepipeline:EnableStageTransition",
          "codepipeline:DisableStageTransition"
        ],
          "Resource" : [
          "arn:aws:codepipeline:us-west-2:111222333444:MyFirstPipeline/*"
```

] } }

다음 예는 111222333444 계정의 정책을 보여줍니다. 이 정책을 통해 사용자는 CodePipeline 콘솔에서 MyFirstPipeline이라는 이름의 파이프라인을 볼 수 있지만 변경할 수는 없습니다. 이 정책의 기반 은 AWSCodePipeline_ReadOnlyAccess 관리형 정책이지만 MyFirstPipeline 파이프라인에 대 해 특화되어 있으므로 관리형 정책을 바로 사용할 수 없습니다. 특정 파이프라인으로 정책을 제한하고 싶지 않은 경우 CodePipeline으로 생성하고 관리하는 관리형 정책의 사용을 고려해 보세요. 자세한 내 용은 관리형 정책 작업을 참조하세요. 액세스용으로 만든 IAM 역할에 이 정책을 연결해야 합니다(예: CrossAccountPipelineViewers 역할).

```
{
  "Statement": [
    {
      "Action": [
        "codepipeline:GetPipeline",
        "codepipeline:GetPipelineState",
        "codepipeline:GetPipelineExecution",
        "codepipeline:ListPipelineExecutions",
        "codepipeline:ListActionExecutions",
        "codepipeline:ListActionTypes",
        "codepipeline:ListPipelines",
        "codepipeline:ListTagsForResource",
        "iam:ListRoles",
        "s3:ListAllMyBuckets",
        "codecommit:ListRepositories",
        "codedeploy:ListApplications",
        "lambda:ListFunctions",
        "codestar-notifications:ListNotificationRules",
        "codestar-notifications:ListEventTypes",
        "codestar-notifications:ListTargets"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:codepipeline:us-west-2:111222333444:MyFirstPipeline"
    },
    {
      "Action": [
        "codepipeline:GetPipeline",
        "codepipeline:GetPipelineState",
        "codepipeline:GetPipelineExecution",
```

"codepipeline:ListPipelineExecutions", "codepipeline:ListActionExecutions", "codepipeline:ListActionTypes", "codepipeline:ListPipelines", "codepipeline:ListTagsForResource", "iam:ListRoles", "s3:GetBucketPolicy", "s3:GetObject", "s3:ListBucket", "codecommit:ListBranches", "codedeploy:GetApplication", "codedeploy:GetDeploymentGroup", "codedeploy:ListDeploymentGroups", "elasticbeanstalk:DescribeApplications", "elasticbeanstalk:DescribeEnvironments", "lambda:GetFunctionConfiguration", "opsworks:DescribeApps", "opsworks:DescribeLayers", "opsworks:DescribeStacks"], "Effect": "Allow", "Resource": "*" }, { "Sid": "CodeStarNotificationsReadOnlyAccess", "Effect": "Allow", "Action": ["codestar-notifications:DescribeNotificationRule"], "Resource": "*", "Condition": { "StringLike": { "codestar-notifications:NotificationsForResource": "arn:aws:codepipeline:*" } } }], "Version": "2012-10-17"

일단 이 정책을 만들었으면 111222333444 계정에 IAM 역할을 만들어 역할에 이 정책을 연결하세요. 역할의 신뢰 관계에서이 역할을 수임할 AWS 계정을 추가해야 합니다. 다음 예제는 1111111111 AWS 계정의 사용자가 111222333444 계정에 정의된 역할을 수임하도록 허용하는 정책을 보여줍니다.

}

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111111111111:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

다음 예제는 사용자가 111111111111 AWS 계정에서 CrossAccountPipelineViewers라는 역할 을 수임하도록 허용하는 111222333444 계정에서 생성된 정책을 보여줍니다.



IAM 정책을 생성하여 계정의 사용자가 액세스할 수 있는 호출 및 리소스를 제한한 다음, 해당 정책을 관리 사용자에게 연결합니다. IAM 역할 생성 방법 및 CodePipeline용 IAM 정책 명령문 예제를 살펴보 는 방법에 대한 자세한 내용은 고객 관리형 정책 예 단원을 참조하세요.

태그를 사용하여 CodePipeline 리소스에 대한 액세스 통제

IAM 정책 설명의 조건은 CodePipeline 작업에 필요한 리소스에 대한 권한을 지정하는 데 사용하는 구 문의 일부입니다. 조건에 태그를 사용하는 것은 리소스 및 요청에 대한 액세스를 제어하는 하나의 방 법입니다. CodePipeline 리소스 태깅에 대한 자세한 내용은 <u>리소스에 태그 지정</u> 섹션을 참조하세요. 이 주제에서는 태그 기반 액세스 제어를 다룹니다.

IAM 정책을 설계할 때 특정 리소스에 대한 액세스 권한을 부여하여 세부적인 권한을 설정할 수도 있습 니다. 관리하는 리소스의 개수가 늘어날수록 이 작업은 더 어려워집니다. 리소스에 태그를 지정하고 정 책 문 조건에서 태그를 사용하면 이러한 작업이 더 간단해질 수 있습니다. 특정 태그를 사용하여 리소 스에 대량으로 액세스 권한을 부여합니다. 그런 다음 생성 중 또는 나중에 이 태그를 관련 리소스에 반 복해서 적용합니다.

리소스에 태그가 연결되거나 태그 지정을 지원하는 서비스에 대한 요청에서 전달될 수 있습니다. CodePipeline에서 리소스에 태그가 있을 수 있고 일부 작업에 태그가 포함될 수 있습니다. IAM 정책을 생성하면 태그 조건 키를 사용하여 다음을 제어할 수 있습니다.

- 파이프라인 리소스에 이미 있는 태그를 기반으로 해당 리소스에 대해 작업을 수행할 수 있는 사용자
- 작업의 요청에서 전달될 수 있는 태그
- 요청에서 특정 키를 사용할 수 있는지 여부를 통제합니다.

문자열 조건 연산자를 사용하여 키와 문자열 값을 비교한 결과에 따라 액세스를 제한하는 Condition 요소를 생성할 수 있습니다. Null 조건을 제외하고 조건 연산자 이름 끝에 IfExists를 추가할 수 있습 니다. 이렇게 하면 "요청 콘텍스트에 정책 키가 있으면 정책에 지정된 대로 키를 처리하고, 키가 없으면 조건 요소를 true로 평가합니다." 예를 들어 다른 유형의 리소스에는 없을 수도 있는 조건 키로 제한하 는 데 StringEqualsIfExists를 사용할 수 있습니다.

태그 조건 키의 전체 구문 및 의미는 <u>태그를 사용한 액세스 제어</u>를 참조하세요. 조건 키에 대한 자세한 내용은 다음 리소스를 참조하세요. 이 섹션의 CodePipeline 정책 예제는 조건 키에 대한 다음 정보와 일치하며, 이를 리소스 중첩과 같은 CodePipeline의 미묘한 차이에 대한 예시로 확장합니다.

- 문자열 조건 연산자
- AWS 서비스 IAM에서 작동하는
- <u>SCP 구문</u>
- IAM JSON 정책 요소: Condition
- aws:RequestTag/tag-key
- CodePipeline에 사용되는 조건 키

다음 예에서는 CodePipeline 사용자에 정책의 태그 조건을 지정하는 방법을 설명합니다.

Example 1: 요청의 태그 기반 작업 한도 지정

AWSCodePipeline_FullAccess 관리형 사용자 정책은 사용자에게 모든 리소스에서 모든 CodePipeline 작업을 수행할 수 있는 무제한적인 권한을 제공합니다.

다음 정책은 이러한 기능을 제한하고 권한이 없는 사용자가 요청에 특정 태그가 나열된 파이프라 인을 생성할 수 있는 권한을 거부합니다. 이와 관련하여 정책은 요청이 ProjectA 또는 ProjectB 값 중 하나와 함께 Project라는 태그를 지정하는 경우 CreatePipeline 작업을 거부합니다. aws:RequestTag 조건 키는 IAM 요청에서 전달할 수 있는 태그를 제어하는 데 사용됩니다.

다음 예제에서 정책의 목적은 지정된 태그 값으로 파이프라인을 생성할 권한이 없는 사용자의 권한을 거부하는 것입니다. 하지만 파이프라인을 생성하려면 파이프라인 자체 외에도 리소스(예: 파이프라인 작업 및 단계)에 액세스해야 합니다. 정책에 지정된 'Resource'가 '*'이므로 ARN이 있는 모든 리소 스에 대해 정책이 평가되며 파이프라인 생성 시 생성됩니다. 하지만 이러한 추가 리소스에는 태그 조건 키가 없기 때문에 StringEquals 검사는 fail로 끝나고 사용자에게 권한이 부여되지 않아 어떤 파이프 라인도 생성하지 못합니다. 이 문제를 해결하려면 그 대신 StringEqualsIfExists 조건 연산자를 사용해야 합니다. 이렇게 하면 조건 키가 존재하는 경우에만 테스트가 실행됩니다.

그 결과 다음은 이렇게 해석할 수 있습니다. "검사 대상 리소스에 태그 "RequestTag/Project" 조건 키가 있으면 키 값이 projectA로 시작할 때에만 작업을 허용합니다. 검사 대상 리소스에 조건 키가 없 으면 그냥 둡니다."

또한 aws : TagKeys 조건 키를 사용하여 동일한 태그 값을 포함하는 태그 수정 작업을 허용하지 않음 으로써 이러한 권한이 없는 사용자가 리소스를 변경하지 못하게 합니다. 고객의 관리자는 권한이 없는 관리 사용자에게 관리형 사용자 정책 이외에 이 IAM 정책도 연결해야 합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codepipeline:CreatePipeline",
        "codepipeline:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEqualsIfExists": {
          "aws:RequestTag/Project": ["ProjectA", "ProjectB"]
        }
      }
    },
    {
      "Effect": "Deny",
      "Action": [
        "codepipeline:UntagResource"
      ],
      "Resource": "*",
      "Condition": {
```

```
"ForAllValues:StringEquals": {
    "aws:TagKeys": ["Project"]
    }
    }
    ]
}
```

Example 2: 리소스 태그 기반 태그 지정 작업 제한

AWSCodePipeline_FullAccess 관리형 사용자 정책은 사용자에게 모든 리소스에서 모든 CodePipeline 작업을 수행할 수 있는 무제한적인 권한을 제공합니다.

다음 정책은 이러한 기능을 제한하고 권한이 없는 사용자의 지정된 프로젝트 파이프라인에 대한 작업 수행 권한을 거부합니다. 이와 관련하여 정책은 리소스에 ProjectA 또는 ProjectB 값 중 하나가 포 함된 Project 태그가 있으면 일부 작업을 거부합니다. aws:ResourceTag 조건 키는 해당 리소스의 태그를 기반으로 리소스에 대한 액세스를 제어하는 데 사용됩니다. 고객의 관리자는 권한이 없는 IAM 사용자에게 관리형 사용자 정책 이외에 이 IAM 정책도 연결해야 합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codepipeline:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Project": ["ProjectA", "ProjectB"]
        }
      }
    }
  ]
}
```

Example 3: 요청의 태그 기반 작업 허용

다음 정책은 사용자에게 CodePipeline에서 개발 파이프라인을 생성할 수 있는 권한을 부여합니다.

이와 관련하여 정책은 요청이 ProjectA 값이 포함된 Project 태그를 지정하는 경우 CreatePipeline 및 TagResource 작업을 허용합니다. 즉, 지정할 수 있는 유일한 태그 키는 Project이며, 해당 값은 ProjectA이어야 합니다.

aws:RequestTag 조건 키는 IAM 요청에서 전달할 수 있는 태그를 제어하는 데 사 용됩니다. aws:TagKeys 조건은 태그 키의 대/소문자를 구분합니다. 이 정책은 AWSCodePipeline_FullAccess 관리형 사용자 정책이 연결되어 있지 않은 사용자 또는 역할에 유 용합니다. 관리형 정책은 사용자에게 모든 리소스에서 모든 CodePipeline 작업을 수행할 수 있는 무제 한적인 권한을 제공합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codepipeline:CreatePipeline",
        "codepipeline:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/Project": "ProjectA"
        },
        "ForAllValues:StringEquals": {
          "aws:TagKeys": ["Project"]
        }
      }
    }
  ]
}
```

Example 4: 리소스 태그 기반 태그 해제 작업 제한

AWSCodePipeline_FullAccess 관리형 사용자 정책은 사용자에게 모든 리소스에서 모든 CodePipeline 작업을 수행할 수 있는 무제한적인 권한을 제공합니다.

다음 정책은 이러한 기능을 제한하고 권한이 없는 사용자의 지정된 프로젝트 파이프라인에 대한 작업 수행 권한을 거부합니다. 이와 관련하여 정책은 리소스에 ProjectA 또는 ProjectB 값 중 하나가 포 함된 Project 태그가 있으면 일부 작업을 거부합니다. 또한 aws : TagKeys 조건 키를 사용하여 Project 태그를 완전히 제거하는 태그 수정 작업을 허용하 지 않음으로써 이러한 권한이 없는 사용자가 리소스를 변경하지 못하게 합니다. 고객의 관리자는 권한 이 없는 사용자 또는 역할에 관리형 사용자 정책 이외에 이 IAM 정책도 연결해야 합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codepipeline:UntagResource"
      ],
      "Resource": "*",
      "Condition": {
        "ForAllValues:StringEquals": {
          "aws:TagKeys": ["Project"]
        }
      }
    }
  ]
}
```

CodePipeline 콘솔 사용에 필요한 권한

CodePipeline 콘솔에서 CodePipeline을 사용하려면 다음 서비스의 최소 권한이 있어야 합니다.

- · AWS Identity and Access Management
- Amazon Simple Storage Service(S3)

이러한 권한을 통해 계정의 다른 AWS 리소스를 설명할 수 있습니다 AWS .

사용자의 파이프라인에 통합하는 다른 서비스에 따라 다음 중 한 가지 이상의 권한이 필요할 수 있습니 다.

- AWS CodeCommit
- CodeBuild
- AWS CloudFormation
- AWS CodeDeploy
- AWS Elastic Beanstalk

- AWS Lambda
- AWS OpsWorks

최소 필수 권한보다 더 제한적인 IAM 정책을 만들면 콘솔은 해당 IAM 정책에 연결된 사용자 에 대해 의도대로 작동하지 않습니다. 사용자가 CodePipeline 콘솔을 사용할 수 있도록 하려면 AWSCodePipeline_ReadOnlyAccess 관리형 정책을 사용자에게 연결합니다(<u>AWS 에 대한 관리형</u> 정책 AWS CodePipeline 참조).

AWS CLI 또는 CodePipeline API를 호출하는 사용자에게는 최소 콘솔 권한을 허용할 필요가 없습니다.

CodePipeline 콘솔에서 컴퓨팅 로그를 보는 데 필요한 권한

CodePipeline 콘솔의 명령 작업에서 로그를 보려면 콘솔 역할에 권한이 있어야 합니다. 콘솔에서 로그 를 보려면 콘솔 역할에 logs:GetLogEvents 권한을 추가합니다.

콘솔 역할 정책 설명에서 다음 예제와 같이 파이프라인 수준까지 권한의 범위를 좁힙니다.

```
{
    "Effect": "Allow",
    "Action": [
        "Action": "logs:GetLogEvents"
    ],
    "Resource": "arn:aws:logs:*:YOUR_AWS_ACCOUNT_ID:log-group:/aws/
codepipeline/YOUR_PIPELINE_NAME:*"
}
```

AWS 에 대한 관리형 정책 AWS CodePipeline

AWS 관리형 정책은에서 생성하고 관리하는 독립 실행형 정책입니다 AWS. AWS 관리형 정책은 사용 자, 그룹 및 역할에 권한 할당을 시작할 수 있도록 많은 일반적인 사용 사례에 대한 권한을 제공하도록 설계되었습니다.

AWS 관리형 정책은 모든 AWS 고객이 사용할 수 있으므로 특정 사용 사례에 대해 최소 권한을 부여하 지 않을 수 있습니다. 사용 사례에 고유한 <u>고객 관리형 정책</u>을 정의하여 권한을 줄이는 것이 좋습니다.

AWS 관리형 정책에 정의된 권한은 변경할 수 없습니다. 가 관리형 정책에 정의된 권한을 AWS 업데이 트하는 AWS 경우 업데이트는 정책이 연결된 모든 보안 주체 자격 증명(사용자, 그룹 및 역할)에 영향 을 미칩니다. AWS AWS 서비스 는 새가 시작되거나 기존 서비스에 새 API 작업을 사용할 수 있게 될 때 AWS 관리형 정책을 업데이트할 가능성이 높습니다.

자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책을 참조하세요.

🛕 Important

AWS 관리형 정책 AWSCodePipelineFullAccess 및 AWSCodePipelineReadOnlyAccess가 교체되었습니다. AWSCodePipeline_FullAccess 및 AWSCodePipeline_ReadOnlyAccess 정책을 사용 하세요.

AWS 관리형 정책: AWSCodePipeline_FullAccess

CodePipeline에 대한 모든 액세스 권한을 부여하는 정책입니다. IAM 콘솔에서 JSON 정책 문서를 보려 면 AWSCodePipeline_FullAccess를 참조하세요.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- codepipeline CodePipeline에 권한을 부여합니다.
- chatbot 보안 주체가 채팅 애플리케이션에서 Amazon Q Developer의 리소스를 관리할 수 있는 권한을 부여합니다.
- cloudformation 보안 주체가 리소스 스택을 관리할 수 있는 권한을 부여합니다 AWS CloudFormation.
- cloudtrail 보안 주체가 CloudTrail에서 로깅 리소스를 관리할 수 있는 권한을 부여합니다.
- codebuild 보안 주체가 CodeBuild에서 빌드 리소스에 액세스할 수 있는 권한을 부여합니다.
- codecommit 보안 주체가 CodeCommit에서 소스 리소스에 액세스할 수 있는 권한을 부여합니다.
- codedeploy 보안 주체가 CodeDeploy에서 배포 리소스에 액세스할 수 있는 권한을 부여합니다.
- codestar-notifications 보안 주체가 AWS CodeStar 알림의 리소스에 액세스할 수 있는 권 한을 부여합니다.

- ec2 CodeCatalyst의 배포가 Amazon EC2에서 Elastic Load Balancing을 관리할 수 있는 권한을 부여합니다.
- ecr Amazon ECR의 리소스에 액세스할 수 있는 권한을 부여합니다.
- elasticbeanstalk 보안 주체가 Elastic Beanstalk에서 리소스에 액세스할 수 있는 권한을 부여 합니다.
- iam 보안 주체가 IAM에서 역할 및 정책을 관리할 수 있는 권한을 부여합니다.
- 1ambda 보안 주체가 Lambda에서 리소스를 관리할 수 있는 권한을 부여합니다.
- events 보안 주체가 CloudWatch Events에서 리소스를 관리할 수 있는 권한을 부여합니다.
- opsworks 보안 주체가 리소스를 관리할 수 있는 권한을 부여합니다 AWS OpsWorks.
- s3 보안 주체가 Amazon S3에서 리소스를 관리할 수 있는 권한을 부여합니다.
- sns 보안 주체가 Amazon SNS에서 알림 리소스를 관리할 수 있는 권한을 부여합니다.
- states 보안 주체가 상태 시스템을 볼 수 있는 권한을 부여합니다 AWS Step Functions. 상태 머 신은 작업을 관리하고 상태 간 전환을 수행하는 상태 모음으로 구성됩니다.

ł			
	"Statement": [
	{		
	"Action": [
	"codepipeline:*",		
	"cloudformation:DescribeStacks",		
	"cloudformation:ListStacks",		
	"cloudformation:ListChangeSets",		
	"cloudtrail:DescribeTrails",		
	"codebuild:BatchGetProjects",		
	"codebuild:CreateProject",		
	"codebuild:ListCuratedEnvironmentImages",		
	"codebuild:ListProjects",		
	"codecommit:ListBranches",		
	"codecommit:GetReferences",		
	"codecommit:ListRepositories",		
	"codedeploy:BatchGetDeploymentGroups",		
	"codedeploy:ListApplications",		
	"codedeploy:ListDeploymentGroups",		
	<pre>"ec2:DescribeSecurityGroups",</pre>		
	"ec2:DescribeSubnets",		
	"ec2:DescribeVpcs",		
	"ecr:DescribeRepositories",		

```
"ecr:ListImages",
        "ecs:ListClusters",
        "ecs:ListServices",
        "elasticbeanstalk:DescribeApplications",
        "elasticbeanstalk:DescribeEnvironments",
        "iam:ListRoles",
        "iam:GetRole",
        "lambda:ListFunctions",
        "events:ListRules",
        "events:ListTargetsByRule",
        "events:DescribeRule",
        "opsworks:DescribeApps",
        "opsworks:DescribeLayers",
        "opsworks:DescribeStacks",
        "s3:ListAllMyBuckets",
        "sns:ListTopics",
        "codestar-notifications:ListNotificationRules",
        "codestar-notifications:ListTargets",
        "codestar-notifications:ListTagsforResource",
        "codestar-notifications:ListEventTypes",
        "states:ListStateMachines"
    ],
    "Effect": "Allow",
    "Resource": "*",
    "Sid": "CodePipelineAuthoringAccess"
},
{
    "Action": [
        "s3:GetObject",
        "s3:ListBucket",
        "s3:GetBucketPolicy",
        "s3:GetBucketVersioning",
        "s3:GetObjectVersion",
        "s3:CreateBucket",
        "s3:PutBucketPolicy"
    ],
    "Effect": "Allow",
    "Resource": "arn:aws:s3::*:codepipeline-*",
    "Sid": "CodePipelineArtifactsReadWriteAccess"
},
{
    "Action": [
        "cloudtrail:PutEventSelectors",
        "cloudtrail:CreateTrail",
```

```
"cloudtrail:GetEventSelectors",
        "cloudtrail:StartLogging"
    ],
    "Effect": "Allow",
    "Resource": "arn:aws:cloudtrail:*:*:trail/codepipeline-source-trail",
    "Sid": "CodePipelineSourceTrailReadWriteAccess"
},
{
    "Action": [
        "iam:PassRole"
    ],
    "Effect": "Allow",
    "Resource": [
        "arn:aws:iam::*:role/service-role/cwe-role-*"
    ],
    "Condition": {
        "StringEquals": {
            "iam:PassedToService": [
                "events.amazonaws.com"
            ]
        }
    },
    "Sid": "EventsIAMPassRole"
},
{
    "Action": [
        "iam:PassRole"
    ],
    "Effect": "Allow",
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "iam:PassedToService": [
                "codepipeline.amazonaws.com"
            ]
        }
    },
    "Sid": "CodePipelineIAMPassRole"
},
{
    "Action": [
        "events:PutRule",
        "events:PutTargets",
        "events:DeleteRule",
```

```
"events:DisableRule",
               "events:RemoveTargets"
           ],
           "Effect": "Allow",
           "Resource": [
               "arn:aws:events:*:*:rule/codepipeline-*"
           ],
           "Sid": "CodePipelineEventsReadWriteAccess"
       },
       {
           "Sid": "CodeStarNotificationsReadWriteAccess",
           "Effect": "Allow",
           "Action": [
               "codestar-notifications:CreateNotificationRule",
               "codestar-notifications:DescribeNotificationRule",
               "codestar-notifications:UpdateNotificationRule",
               "codestar-notifications:DeleteNotificationRule",
               "codestar-notifications:Subscribe",
               "codestar-notifications:Unsubscribe"
           ],
           "Resource": "*",
           "Condition": {
               "StringLike": {
                   "codestar-notifications:NotificationsForResource":
"arn:aws:codepipeline:*"
               }
           }
       },
       {
           "Sid": "CodeStarNotificationsSNSTopicCreateAccess",
           "Effect": "Allow",
           "Action": [
               "sns:CreateTopic",
               "sns:SetTopicAttributes"
           ],
           "Resource": "arn:aws:sns:*:*:codestar-notifications*"
       },
       {
           "Sid": "CodeStarNotificationsChatbotAccess",
           "Effect": "Allow",
           "Action": [
               "chatbot:DescribeSlackChannelConfigurations",
               "chatbot:ListMicrosoftTeamsChannelConfigurations"
           ],
```

```
"Resource": "*"
}
],
"Version": "2012-10-17"
}
```

AWS 관리형 정책: AWSCodePipeline_ReadOnlyAccess

CodePipeline에 대한 읽기 전용 액세스 권한을 부여하는 정책입니다. IAM 콘솔에서 JSON 정책 문서를 보려면 <u>AWSCodePipeline_ReadOnlyAccess</u>를 참조하세요.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- codepipeline CodePipeline의 작업에 권한을 부여합니다.
- codestar-notifications 보안 주체가 AWS CodeStar 알림의 리소스에 액세스할 수 있는 권 한을 부여합니다.
- s3 보안 주체가 Amazon S3에서 리소스를 관리할 수 있는 권한을 부여합니다.
- sns 보안 주체가 Amazon SNS에서 알림 리소스를 관리할 수 있는 권한을 부여합니다.

```
{
    "Statement": [
        {
            "Action": [
                "codepipeline:GetPipeline",
                "codepipeline:GetPipelineState",
                "codepipeline:GetPipelineExecution",
                "codepipeline:ListPipelineExecutions",
                "codepipeline:ListActionExecutions",
                "codepipeline:ListActionTypes",
                "codepipeline:ListPipelines",
                "codepipeline:ListTagsForResource",
                "s3:ListAllMyBuckets",
                "codestar-notifications:ListNotificationRules",
                "codestar-notifications:ListEventTypes",
                "codestar-notifications:ListTargets"
```

```
],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetBucketPolicy"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3::*:codepipeline-*"
        },
        {
            "Sid": "CodeStarNotificationsReadOnlyAccess",
            "Effect": "Allow",
            "Action": [
                "codestar-notifications:DescribeNotificationRule"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                     "codestar-notifications:NotificationsForResource":
 "arn:aws:codepipeline:*"
                }
            }
        }
    ],
    "Version": "2012-10-17"
}
```

AWS 관리형 정책: AWSCodePipelineApproverAccess

이는 수동 승인 작업을 승인하거나 거부할 권한을 부여하는 정책입니다. IAM 콘솔에서 JSON 정책 문 서를 보려면 AWSCodePipelineApproverAccess를 참조하세요.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

• codepipeline – CodePipeline의 작업에 권한을 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "codepipeline:GetPipeline",
                "codepipeline:GetPipelineState",
                "codepipeline:GetPipelineExecution",
                "codepipeline:ListPipelineExecutions",
                "codepipeline:ListPipelines",
                "codepipeline:PutApprovalResult"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

AWS 관리형 정책: AWSCodePipelineCustomActionAccess

이 정책은 CodePipeline에 사용자 지정 작업을 만들거나 빌드 또는 테스트 작업을 위한 Jenkins 리소스를 통합할 권한을 부여하는 정책입니다. IAM 콘솔에서 JSON 정책 문서를 보려면 AWSCodePipelineCustomActionAccess를 참조하세요.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

• codepipeline – CodePipeline의 작업에 권한을 부여합니다.

```
{
    "Statement": [
        {
            "Action": [
            "codepipeline:AcknowledgeJob",
            "codepipeline:GetJobDetails",
            "codepipeline:PollForJobs",
            "codepipe
```

CodePipeline 관리형 정책 및 알림

CodePipeline에서는 사용자에게 파이프라인에 대한 중요한 변경 사항을 알릴 수 있는 알림을 지원합니다. CodePipeline에 대한 관리형 정책에는 알림 기능에 대한 정책 설명이 포함되어 있습니다. 자세한 내용은 알림이란 무엇입니까?를 참조하세요.

전체 액세스 관리형 정책의 알림과 관련된 권한

이 관리형 정책은 관련 서비스 CodeCommit, CodeBuildCodeBuild CodeDeploy 및 AWS CodeStar Notifications와 함께 CodePipeline에 대한 권한을 부여합니다. 또한 이 정책은 Amazon S3, Elastic Beanstalk, CloudTrail, Amazon EC2 및 AWS CloudFormation등과 같이 파이프라인과 통합되는 다 른 서비스를 사용하는 데 필요한 권한을 부여합니다. 이 관리형 정책이 적용된 사용자는 알림을 위 한 Amazon SNS 주제를 생성 및 관리하고, 주제를 구독 및 구독 취소할 수 있으며, 알림 규칙의 대상 으로 선택할 주제를 나열하고, Slack용으로 구성된 채팅 애플리케이션 클라이언트에서 Amazon Q Developer를 나열할 수도 있습니다.

AWSCodePipeline_FullAccess 관리형 정책에는 알림에 대한 전체 액세스를 허용하는 다음 설명 이 포함되어 있습니다.

```
{
    "Sid": "CodeStarNotificationsReadWriteAccess",
    "Effect": "Allow",
    "Action": [
        "codestar-notifications:CreateNotificationRule",
        "codestar-notifications:DescribeNotificationRule",
        "codestar-notifications:UpdateNotificationRule",
        "codestar-notifications:DeleteNotificationRule",
        "codestar-notifications:Subscribe",
        "codestar-notifications:Unsubscribe"
],
    "Resource": "*",
    "Condition" : {
```

```
"StringLike" : {"codestar-notifications:NotificationsForResource" :
"arn:aws:codepipeline:us-west-2:111222333444:MyFirstPipeline"}
       }
  },
   {
       "Sid": "CodeStarNotificationsListAccess",
       "Effect": "Allow",
       "Action": [
           "codestar-notifications:ListNotificationRules",
           "codestar-notifications:ListTargets",
           "codestar-notifications:ListTagsforResource",
           "codestar-notifications:ListEventTypes"
       ],
       "Resource": "*"
  },
   {
       "Sid": "CodeStarNotificationsSNSTopicCreateAccess",
       "Effect": "Allow",
       "Action": [
           "sns:CreateTopic",
           "sns:SetTopicAttributes"
       ],
       "Resource": "arn:aws:sns:*:*:codestar-notifications*"
  },
   {
       "Sid": "SNSTopicListAccess",
       "Effect": "Allow",
       "Action": [
           "sns:ListTopics"
       ],
       "Resource": "*"
  },
   {
       "Sid": "CodeStarNotificationsChatbotAccess",
       "Effect": "Allow",
       "Action": [
           "chatbot:DescribeSlackChannelConfigurations",
           "chatbot:ListMicrosoftTeamsChannelConfigurations"
         ],
      "Resource": "*"
  }
```

읽기 전용 관리형 정책의 알림과 관련된 권한

AWSCodePipeline_ReadOnlyAccess 관리형 정책에는 알림에 대한 읽기 전용 액세스를 허용하는 다음 설명이 포함되어 있습니다. 이 정책이 적용된 사용자는 리소스에 대한 알림을 볼 수 있지만 리소 스를 생성, 관리 또는 구독할 수는 없습니다.

```
{
       "Sid": "CodeStarNotificationsPowerUserAccess",
       "Effect": "Allow",
       "Action": [
           "codestar-notifications:DescribeNotificationRule"
       ],
       "Resource": "*",
       "Condition" : {
           "StringLike" : {"codestar-notifications:NotificationsForResource" :
"arn:aws:codepipeline:us-west-2:111222333444:MyFirstPipeline"}
       }
  },
   {
       "Sid": "CodeStarNotificationsListAccess",
       "Effect": "Allow",
       "Action": [
           "codestar-notifications:ListNotificationRules",
           "codestar-notifications:ListEventTypes",
           "codestar-notifications:ListTargets"
       ],
       "Resource": "*"
  }
```

IAM 및 알림에 대한 자세한 내용은 <u>AWS CodeStar 알림의 Identity and Access Management</u>를 참조하 세요.

AWS CodePipelineAWS 관리형 정책에 대한 업데이트

이 서비스가 이러한 변경 사항을 추적하기 시작한 이후부터 CodePipeline의 AWS 관리형 정책 업데이 트에 대한 세부 정보를 봅니다. 이 페이지의 변경 사항에 대한 자동 알림을 받으려면 CodePipeline <u>문</u> 서 기록 페이지에서 RSS 피드를 구독하세요.

변경 사항	설명	날짜
<u>AWSCodePipeline_FullAccess</u> – 기존 정책 업데이트	CodePipeline은 AWS CloudFormation에서 ListStacks 를 지원하도록 이 정책에 대한 권한을 추가했 습니다.	2024년 3월 15일
<u>AWSCodePipeline_FullAccess</u> – 기존 정책 업데이트	이 정책은 채팅 애플리케이션 에서 Amazon Q Developer에 대한 권한을 추가하도록 업데 이트되었습니다. 자세한 내용 은 <u>CodePipeline 관리형 정책</u> 및 알림 단원을 참조하십시오.	2023년 6월 21일
AWSCodePipeline_FullAccess 및 <u>AWSCodePipeline_Re</u> <u>adOnlyAccess</u> 관리형 정책 – 기존 정책에 대한 업데이트	CodePipeline은 채팅 애플 리케이션에서 Amazon Q Developer를 사용하여 추가 알림 유형을 지원하는 권한 을 이러한 정책에 추가했습니 다chatbot:ListMicros oftTeamsChannelCon figurations .	2023년 5월 16일
AWSCodePipelineFullAccess - 사용되지 않음	이 정책은 AWSCodePi peline_FullAccess (으) 로 대체되었습니다. 2022년 11월 17일 이후에는 이 정책을 새 사용자, 그룹 또는 역 할에 연결할 수 없습니다. 자세 한 내용은 <u>AWS 에 대한 관리형</u> 정책 AWS CodePipeline 단원 을 참조하십시오.	2022년 11월 17일
AWSCodePipelineRea dOnlyAccess – 사용되지 않음	이 정책은 AWSCodePi peline_ReadOnlyAcc ess (으)로 대체되었습니다.	2022년 11월 17일

변경 사항	설명	날짜
	2022년 11월 17일 이후에는 이 정책을 새 사용자, 그룹 또는 역 할에 연결할 수 없습니다. 자세 한 내용은 <u>AWS 에 대한 관리형</u> 정책 AWS CodePipeline 단원 을 참조하십시오.	
CodePipeline에서 변경 사항 추적 시작	CodePipeline이 AWS 관리형 정책에 대한 변경 사항 추적을 시작했습니다.	2021년 3월 12일

고객 관리형 정책 예

이 섹션에서는 다양한 CodePipeline 작업에 대한 권한을 부여하는 사용자 정책의 예를 제공합니다. 이 러한 정책은 CodePipeline API, AWS SDKs 또는를 사용할 때 작동합니다 AWS CLI. 콘솔을 사용하는 경우 콘솔에 특정한 추가 권한을 부여해야 합니다. 자세한 내용은 <u>CodePipeline 콘솔 사용에 필요한 권</u> 한 단원을 참조하십시오.

Note

모든 예에서는 미국 서부(오리건) 리전(us-west-2)을 사용하며 가상의 계정 ID를 포함합니다.

예시

- 예 1: 파이프라인 상태를 입수하도록 권한 부여
- 예 2: 단계 간 전환을 활성화하고 비활성화할 권한 부여
- 예 3: 사용 가능한 모든 작업 유형의 목록을 가져올 수 있는 권한 부여
- 예 4: 수동 승인 작업을 승인 또는 거부할 권한 부여
- 예 5: 사용자 지정 작업에 대한 작업 폴 권한 부여
- 예제 6: Jenkins와 AWS CodePipeline 통합을 위한 정책 연결 또는 편집
- 예 7: 파이프라인에 대한 교차 계정 액세스 구성
- 예 8: 파이프라인에서 다른 계정과 연결된 AWS 리소스 사용
예 1: 파이프라인 상태를 입수하도록 권한 부여

다음 예는 MyFirstPipeline이라는 이름의 파이프라인 상태를 입수할 권한을 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "codepipeline:GetPipelineState"
        ],
            "Resource": "arn:aws:codepipeline:us-west-2:111222333444:MyFirstPipeline"
        }
    ]
}
```

예 2: 단계 간 전환을 활성화하고 비활성화할 권한 부여

다음 예는 MyFirstPipeline이라는 이름의 파이프라인에서 모든 단계 간 전환을 활성화하고 비활성 화할 권한을 허용합니다.

파이프라인에서 단일 단계의 전환을 활성화하고 비활성화하는 권한을 사용자에게 주려면 해당 단계를 지정해야 합니다. 예를 들어 MyFirstPipeline이라는 파이프라인에서 Staging이라는 단계의 전환 을 활성화 및 비활성화하도록 하려면 다음과 같이 합니다.

"Resource": "arn:aws:codepipeline:us-west-2:111222333444:MyFirstPipeline/Staging"

예 3: 사용 가능한 모든 작업 유형의 목록을 가져올 수 있는 권한 부여

다음 예는 us-west-2 리전에서 파이프라인에 이용할 수 있는 모든 작업 유형의 목록을 가져올 권한 을 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "codepipeline:ListActionTypes"
            ],
            "Resource": "arn:aws:codepipeline:us-west-2:111222333444:actiontype:*"
        }
    ]
}
```

예 4: 수동 승인 작업을 승인 또는 거부할 권한 부여

다음 예는 MyFirstPipeline이라는 파이프라인의 Staging이라는 단계에서 수동 승인 작업을 승인 하거나 거부할 권한을 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "codepipeline:PutApprovalResult"
            ],
            "Resource": "arn:aws:codepipeline:us-west-2:111222333444:MyFirstPipeline/
Staging/*"
        }
    ]
}
```

예 5: 사용자 지정 작업에 대한 작업 폴 권한 부여

다음 예는 TestProvider라는 이름으로 모든 파이프라인을 통틀어 첫 버전에서 Test 작업 유형에 해 당하는 사용자 지정 작업에 대하여 작업 폴 권한을 허용합니다.

Note

다른 AWS 계정 하에 사용자 지정 작업을 다룰 작업자를 구성하거나 특정 IAM 역할을 갖춰야 작업하도록 할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "codepipeline:PollForJobs"
        ],
            "Resource": [
               "arn:aws:codepipeline:us-
west-2:111222333444:actionType:Custom/Test/TestProvider/1"
        ]
      }
   ]
}
```

예제 6: Jenkins와 AWS CodePipeline 통합을 위한 정책 연결 또는 편집

Jenkins를 이용해 빌드하거나 테스트하도록 파이프라인을 구성하는 경우, 해당 통합을 위해 별도의 자 격 증명을 새로 만들어 Jenkins와 CodePipeline 간 통합에 필요한 최소한의 권한을 보유한 IAM 정책을 연결하세요. 이 정책은 AWSCodePipelineCustomActionAccess 관리형 정책과 동일합니다. 다음 은 Jenkins 통합을 위한 정책을 나타낸 예입니다.

```
{
    "Statement": [
        {
         "Effect": "Allow",
         "Action": [
            "codepipeline:AcknowledgeJob",
            "codepipeline:GetJobDetails",
            "codepipeline:PollForJobs",
            "codepipeline:PutJobFailureResult",
            "codepipeline:PutJobSuccessResult"
        ],
        "Resource": "*"
```

```
}
],
"Version": "2012-10-17"
}
```

예 7: 파이프라인에 대한 교차 계정 액세스 구성

다른 AWS 계정의 사용자 및 그룹에 대한 파이프라인 액세스를 구성할 수 있습니다. 권장 방법은 파이 프라인이 생성된 계정에 역할을 만드는 것입니다. 역할은 다른 AWS 계정의 사용자가 해당 역할을 수 임하고 파이프라인에 액세스할 수 있도록 허용해야 합니다. 자세한 내용은 <u>연습: 역할을 사용한 교차</u> <u>계정 액세스</u> 단원을 참조하십시오.

다음 예는 80398EXAMPLE 계정의 정책을 보여줍니다. 이 정책을 통해 사용자는 CodePipeline 콘솔에 서 MyFirstPipeline이라는 이름의 파이프라인을 볼 수 있지만 변경할 수는 없습니다. 이 정책의 기 반은 AWSCodePipeline_ReadOnlyAccess 관리형 정책이지만 MyFirstPipeline 파이프라인에 대해 특화되어 있으므로 관리형 정책을 바로 사용할 수 없습니다. 특정 파이프라인으로 정책을 제한하 고 싶지 않은 경우 CodePipeline으로 생성하고 관리하는 관리형 정책의 사용을 고려해 보세요. 자세한 내용은 <u>관리형 정책 작업</u>을 참조하세요. 액세스용으로 만든 IAM 역할에 이 정책을 연결해야 합니다(예: CrossAccountPipelineViewers 역할).

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:GetPipeline",
                "codepipeline:GetPipelineState",
                "codepipeline:ListActionTypes",
                "codepipeline:ListPipelines",
                "iam:ListRoles",
                "s3:GetBucketPolicy",
                "s3:GetObject",
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "codedeploy:GetApplication",
                "codedeploy:GetDeploymentGroup",
                "codedeploy:ListApplications",
                "codedeploy:ListDeploymentGroups",
                "elasticbeanstalk:DescribeApplications",
                "elasticbeanstalk:DescribeEnvironments",
                "lambda:GetFunctionConfiguration",
                "lambda:ListFunctions"
```

```
],
    "Resource": "arn:aws:codepipeline:us-east-2:80398EXAMPLE:MyFirstPipeline"
    }
    ],
    "Version": "2012-10-17"
}
```

일단 이 정책을 만들었으면 80398EXAMPLE 계정에 IAM 역할을 만들어 역할에 이 정책을 연결하세요. 역할의 신뢰 관계에서이 역할을 수임하는 AWS 계정을 추가해야 합니다. 다음 예제는 1111111111 AWS 계정의 사용자가 80398EXAMPLE 계정에 정의된 역할을 수임하도록 허용하는 정책을 보여줍니 다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111111111111:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

다음 예제는 사용자가 80398EXAMPLE 계정에서 라는 역할을 수임하도록 허용하는 11111111111 AWS 계정CrossAccountPipelineViewers에서 생성된 정책을 보여줍니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::80398EXAMPLE:role/CrossAccountPipelineViewers"
        }
    ]
}
```

예 8: 파이프라인에서 다른 계정과 연결된 AWS 리소스 사용

사용자가 다른 AWS 계정의 리소스를 사용하는 파이프라인을 생성할 수 있도록 허용하는 정책을 구 성할 수 있습니다. 파이프라인을 만드는 계정(AccountA)과 그 파이프라인에서 사용할 리소스를 만 든 계정(AccountB) 양쪽에 정책과 역할을 구성해야 합니다. 또한 교차 계정 액세스에 사용하려면에 서 고객 관리형 키를 생성 AWS Key Management Service 해야 합니다. 자세한 내용과 단계별 예시는 <u>CodePipeline에서 다른 AWS 계정의 리소스를 사용하는 파이프라인 생성</u> 및 <u>CodePipeline의 Amazon</u> S3에 저장된 아티팩트에 대해 서버 측 암호화 구성 단원을 참조하십시오.

다음 예는 AccountA가 파이프라인 아티팩트의 저장에 사용하는 S3 버킷에 구성한 정책입니다. 이 정책은 AccountB에 대한 액세스 권한을 부여합니다. 다음 예에서 AccountB에 대한 ARN은 012ID_ACCOUNT_B입니다. S3 버킷에 대한 ARN은 codepipeline-us-east-2-1234567890입니 다. 이들 ARN을 액세스를 허용하려는 계정 및 S3 버킷에 대한 ARN으로 교체합니다.

```
{
    "Version": "2012-10-17",
    "Id": "SSEAndSSLPolicy",
    "Statement": [
        {
            "Sid": "DenyUnEncryptedObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*",
            "Condition": {
                "StringNotEquals": {
                     "s3:x-amz-server-side-encryption": "aws:kms"
                }
            }
        },
        {
            "Sid": "DenyInsecureConnections",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*",
            "Condition": {
                "Bool": {
                     "aws:SecureTransport": false
                }
            }
        },
```

```
{
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::012ID_ACCOUNT_B:root"
            },
            "Action": [
                "s3:Get*",
                "s3:Put*"
            ],
            "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*"
        },
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::012ID_ACCOUNT_B:root"
            },
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890"
        }
    ]
}
```

다음 예는 AccountB의 역할 수임을 허용하도록 AccountA가 구성한 정책입니다. 이 정책을 CodePipeline에 대한 서비스 역할(CodePipeline_Service_Role)에 적용해야 합니다. IAM의 역 할에 정책을 적용하는 자세한 방법은 <u>역할 변경</u>을 참조하세요. 다음 예에서 012ID_ACCOUNT_B는 AccountB에 대한 ARN입니다.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "sts:AssumeRole",
        "Resource": [
            "arn:aws:iam::012ID_ACCOUNT_B:role/*"
        ]
    }
}
```

다음 예는 AccountB에서 구성하고 CodeDeploy의 <u>EC2 인스턴스 역할</u>에 적용한 정책을 보여줍니 다. 이 정책에서는 AccountA가 S3 버킷에 액세스하여 파이프라인 아티팩트(*codepipeline-us-east-2-1234567890*)를 저장하도록 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:Get*"
            ],
            "Resource": [
                "arn:aws:s3:::codepipeline-us-east-2-1234567890/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::codepipeline-us-east-2-1234567890"
            ]
        }
    ]
}
```

다음 예제는 arn:aws:kms:us-

*east-1:012ID_ACCOUNT_A:key/222222-3333333-4444-556677EXAMPLE*가 AccountA에서 생성되고 AccountB가 사용할 수 있도록 구성된 고객 관리형 키의 ARN AWS KMS 인 정책을 보여줍니 다.

다음 예는 AccountB가 만든 IAM 역할(CrossAccount_Role)에 대한 인라인 정책으로, AccountA에서 파이프라인에 필요한 CodeDeploy 작업에 액세스를 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codedeploy:CreateDeployment",
                "codedeploy:GetDeployment",
                "codedeploy:GetDeploymentConfig",
                "codedeploy:GetApplicationRevision",
                "codedeploy:RegisterApplicationRevision"
            ],
            "Resource": "*"
        }
    ]
}
```

다음 예는 AccountB가 만든 IAM 역할(CrossAccount_Role)에 대한 인라인 정책으로, 입력 아티팩트 를 다운로드하고 출력 아티팩트를 업로드하기 위하여 S3 버킷에 액세스를 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "s3:Get0bject*",
               "s3:Put0bject",
                "s3:Put0bject",
               "s3:Put0bject",
               "s3:Put0bject",
               "s3:Put0bject",
               "s3:Put0bject",
```

```
"s3:PutObjectAcl"

],

"Resource": [

"arn:aws:s3:::codepipeline-us-east-2-1234567890/*"

]

}
```

리소스에 대한 교차 계정 액세스를 위해 파이프라인을 편집하는 방법에 대한 자세한 내용은 <u>2단계: 파</u> <u>이프라인 편집</u> 단원을 참조하십시오.

AWS CodePipeline 리소스 기반 정책 예제

Amazon S3과 같은 다른 서비스도 리소스 기반 권한 정책을 지원합니다. 예를 들어, 정책을 S3 버킷에 연결하여 해당 버킷에 대한 액세스 권한을 관리할 수 있습니다. CodePipeline은 리소스 기반 정책을 지 원하지 않지만, 버전 S3 버킷의 파이프라인에 사용할 아티팩트는 저장합니다.

Example CodePipeline의 아티팩트 스토어로 사용할 정책을 S3 버킷에 새로 만들려면

CodePipeline의 아티팩트 스토어로 모든 버전의 S3 버킷을 사용할 수 있습니다. 파이프라인 생성 마법 사를 사용해 첫 번째 파이프라인을 만드는 경우 사용자에 대한 이 S3 버킷이 생성되므로, 아티팩트 스 토어에 업로드한 모든 객체의 암호화를 확인하고 버킷 연결의 보안을 안심할 수 있습니다. 사용자만의 S3 버킷을 새로 만든 경우, 다음 정책 또는 그 요소를 버킷에 추가하는 것이 좋습니다. 이 정책에서 S3 버킷에 대한 ARN은 codepipeline-us-east-2-1234567890입니다. 이 ARN을 S3 버킷에 대한 ARN으로 교체하십시오.

```
{
    "Version": "2012-10-17",
    "Id": "SSEAndSSLPolicy",
    "Statement": [
        {
            "Sid": "DenyUnEncryptedObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::codepipeline-us-east-2-1234567890/*",
            "Condition": {
               "StringNotEquals": {
                     "StringNotEquals": {
                     "StringNotEquals": {
                     "StringNotEquals": {
                     "StringNotEquals": {
                    "StringNotEquals": {
                     "StringNotEquals": {
                     "StringNotEquals": {
                      "StringNotEquals": {
                      "StringNot
```



AWS CodePipeline ID 및 액세스 문제 해결

다음 정보를 사용하여 CodePipeline 및 IAM 작업 시 발생할 수 있는 공통적인 문제를 진단하고 수정할 수 있습니다.

주제

- CodePipeline에서 작업을 수행할 권한이 없음
- iam:PassRole을 수행하도록 인증되지 않음
- 관리자이며 다른 사용자가 CodePipeline에 액세스할 수 있게 허용하기를 원함
- 내 AWS 계정 외부의 사람이 내 CodePipeline 리소스에 액세스하도록 허용하고 싶습니다.

CodePipeline에서 작업을 수행할 권한이 없음

에서 작업을 수행할 권한이 없다는 AWS Management Console 메시지가 표시되면 관리자에게 문의하 여 도움을 받아야 합니다. 관리자는 사용자 이름과 암호를 제공한 사람입니다.

다음 예의 오류는 mateojackson IAM 사용자가 콘솔을 사용하여 파이프라인에 대한 세부 정보를 보 려고 하지만 codepipeline:GetPipeline 권한이 없는 경우에 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: codepipeline:GetPipeline on resource: my-pipeline
```

이 경우, Mateo는 my-pipeline 작업을 사용하여 codepipeline:GetPipeline 리소스에 액세스 하도록 허용하는 정책을 업데이트하라고 관리자에게 요청합니다.

iam:PassRole을 수행하도록 인증되지 않음

iam:PassRole 작업을 수행할 권한이 없다는 오류가 수신되면 관리자에게 문의하여 도움을 받아야 합니다. 관리자는 사용자 이름과 암호를 제공한 사람입니다. 관리자에게 CodePipeline으로 역할 전달 을 허용하도록 정책을 업데이트해 달라고 요청합니다.

일부 AWS 서비스 에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 기존 역할을 해당 서 비스에 전달할 수 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예제 오류는 marymajor라는 IAM 사용자가 콘솔을 사용하여 CodePipeline에서 작업을 수행하려 고 하는 경우에 발생합니다. 하지만 태스크를 수행하려면 서비스에 서비스 역할이 부여한 권한이 있어 야 합니다. Mary는 서비스에 역할을 전달할 수 있는 권한을 가지고 있지 않습니다.

User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole

이 경우 Mary는 iam: PassRole 태스크를 수행하도록 허용하는 정책을 업데이트하라고 관리자에게 요청합니다.

관리자이며 다른 사용자가 CodePipeline에 액세스할 수 있게 허용하기를 원함

다른 사용자가 CodePipeline에 액세스하도록 허용하려면 액세스가 필요한 사람 또는 애플리케이션에 권한을 부여해야 합니다. AWS IAM Identity Center 를 사용하여 사용자 및 애플리케이션을 관리하는 경우 사용자 또는 그룹에 권한 세트를 할당하여 액세스 수준을 정의합니다. 권한 세트는 IAM 정책을 자동으로 생성하고 사용자 또는 애플리케이션과 연결된 IAM 역할에 할당합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 권한 세트를 참조하세요.

IAM Identity Center를 사용하지 않는 경우 액세스가 필요한 사용자 또는 애플리케이션에 대한 IAM 엔 터티(사용자 또는 역할)를 생성해야 합니다. 그런 다음 CodePipeline에서 올바른 권한을 부여하는 정책 을 엔터티에 연결해야 합니다. 권한이 부여되면 사용자 또는 애플리케이션 개발자에게 자격 증명을 제 공합니다. 이들은 이 자격 증명을 사용하여 AWS에 액세스합니다. IAM 사용자, 그룹, 정책, 권한 생성 에 대한 자세한 내용은 IAM 사용 설명서의 IAM ID 및 IAM 정책과 권한을 참조하세요. 내 AWS 계정 외부의 사람이 내 CodePipeline 리소스에 액세스하도록 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제 어 목록(ACL)을 지원하는 서비스의 경우, 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세 스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하세요.

- CodePipeline에서 이러한 기능을 지원하는지 여부를 알아보려면 <u>AWS CodePipeline 에서 IAM을 사</u>용하는 방법 섹션을 참조하세요.
- 소유 AWS 계정 한의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 <u>IAM 사용 설명서의</u> 소유한 다른의 IAM 사용자에게 액세스 권한 제공을 참조 AWS 계정 하세요.
- 타사에 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 <u>타사가 AWS</u> 계정 소유한에 대한 액세스 권한 제공을 AWS 계정참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 <u>외부에서 인</u> 증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하세요.
- 크로스 계정 액세스에 대한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명 서의 IAM의 크로스 계정 리소스 액세스를 참조하세요.

CodePipeline 권한 참조

IAM 자격 증명에 연결할 수 있는 액세스 제어 및 쓰기 권한 정책(자격 증명 기반 정책)을 설정할 때 다 음 표를 참조로 사용하세요. 표에는 각 CodePipeline API 작업과 이 작업을 수행할 수 있는 권한을 부여 할 수 있는 작업이 나와 있습니다. 리소스 수준 권한을 지원하는 작업의 경우 표에는 권한을 부여할 수 있는 AWS 리소스가 나열되어 있습니다. 정책의 Action 필드에 작업을 지정합니다.

리소스 수준 권한은 사용자가 작업을 수행할 수 있는 리소스를 지정할 수 있는 권한입니다. AWS CodePipeline 는 리소스 수준 권한을 부분적으로 지원합니다. 즉, 일부 AWS CodePipeline API 호출의 경우 사용자가 충족해야 하는 조건 또는 사용자가 사용할 수 있는 리소스에 따라 이러한 작업을 사용할 수 있는 시기를 제어할 수 있습니다. 예를 들어 특정 파이프라인에 대해서만 파이프라인 실행 정보를 나열할 수 있는 권한을 사용자에게 부여할 수 있습니다.

Note

리소스 열에는 리소스 수준 권한을 지원하는 API 호출에 필요한 리소스가 나와 있습니다. 리소 스 수준 권한을 지원하지 않는 API 호출의 경우, 사용자에게 해당 사용 권한을 부여할 때 정책 명령문의 리소스 요소를 와일드카드(*)로 지정해야 합니다.

CodePipeline API 작업 및 작업에 필요한 권한

AcknowledgeJob

작업: codepipeline:AcknowledgeJob

지정 작업에 대한 정보를 보고 작업자가 그 작업을 받았는지 여부를 확인하는 데 필요합니다. 사용 자 지정 작업에만 사용합니다.

리소스: 정책 Resource 요소에는 와일드카드(*)만 지원됩니다.

AcknowledgeThirdPartyJob

작업: codepipeline:AcknowledgeThirdPartyJob

작업자가 지정 작업을 받았는지 확인하는 데 필요합니다. 파트너 작업에만 사용합니다.

리소스: 정책 Resource 요소에는 와일드카드(*)만 지원됩니다.

CreateCustomActionType

작업: codepipeline:CreateCustomActionType

AWS 계정과 연결된 모든 파이프라인에서 사용할 수 있는 새 사용자 지정 작업을 생성하는 데 필요 합니다. 사용자 지정 작업에만 사용합니다.

리소스:

작업 유형

arn:aws:codepipeline:region:account:actiontype:owner/category/provider/version
CreatePipeline

작업: codepipeline:CreatePipeline

파이프라인을 새로 만드는 데 필요합니다.

리소스:

파이프라인

arn:aws:codepipeline:region:account:pipeline-name

DeleteCustomActionType

작업:codepipeline:DeleteCustomActionType

사용자 지정 작업을 삭제된 것으로 표시하는 데 필요합니다. 작업이 삭제로 표시된 후 사용자 지정 작업에 PollForJobs를 시도하면 실패합니다. 사용자 지정 작업에만 사용합니다.

리소스:

작업 유형

arn:aws:codepipeline:region:account:actiontype:owner/category/provider/version

DeletePipeline

작업: codepipeline:DeletePipeline

파이프라인을 삭제하는 데 필요합니다.

리소스:

파이프라인

arn:aws:codepipeline:region:account:pipeline-name

DeleteWebhook

작업: codepipeline:DeleteWebhook

Webhook를 삭제하는 데 필요합니다.

리소스:

Webhook

arn:aws:codepipeline:region:account:webhook:webhook-name

DeregisterWebhookWithThirdParty

작업: codepipeline:DeregisterWebhookWithThirdParty

Webhook를 삭제하기 전에 CodePipeline에서 생성된 Webhook와 이벤트를 감지할 외부 도구 간의 연결을 제거하는 데 필요합니다. 현재 GitHub의 작업 유형을 대상으로 하는 Webhook에 대해서만 지원됩니다.

리소스:

Webhook

arn:aws:codepipeline:region:account:webhook:webhook-name

DisableStageTransition

작업: codepipeline:DisableStageTransition

파이프라인의 아티팩트가 파이프라인 내의 다른 단계로 전환하지 않도록 방지하는 데 필요합니다.

리소스:

파이프라인

arn:aws:codepipeline:region:account:pipeline-name

EnableStageTransition

작업: codepipeline:EnableStageTransition

파이프라인의 아티팩트가 파이프라인 내의 다른 단계로 전환하도록 활성화하는 데 필요합니다.

리소스:

파이프라인

arn:aws:codepipeline:region:account:pipeline-name

GetJobDetails

작업: codepipeline:GetJobDetails

작업에 대한 정보를 검색하는 데 필요합니다. 사용자 지정 작업에만 사용합니다.

리소스: 리소스가 필요하지 않습니다.

GetPipeline

작업:codepipeline:GetPipeline

파이프라인 ARN을 포함하여 파이프라인의 구조, 단계, 작업, 메타데이터를 검색하는 데 필요합니 다.

리소스:

파이프라인

arn:aws:codepipeline:region:account:pipeline-name

GetPipelineExecution

작업: codepipeline:GetPipelineExecution

아티팩트의 상세 정보, 파이프라인 실행 ID, 파이프라인의 이름과 버전 및 상태 등 파이프라인 실행 에 대한 정보 검색에 필요합니다.

리소스:

파이프라인

arn:aws:codepipeline:region:account:pipeline-name

GetPipelineState

작업: codepipeline:GetPipelineState

단계와 작업 등 파이프라인 상태에 대한 정보 검색에 필요합니다.

리소스:

파이프라인

arn:aws:codepipeline:region:account:pipeline-name

<u>GetThirdPartyJobDetails</u>

작업: codepipeline:GetThirdPartyJobDetails

타사 조치에 대한 작업의 세부 정보 요청에 필요합니다. 파트너 작업에만 사용합니다.

리소스: 정책 Resource 요소에는 와일드카드(*)만 지원됩니다.

ListActionTypes

작업:codepipeline:ListActionTypes

사용자 계정에 연결된 모든 CodePipeline 작업 유형의 요약을 작성하는 데 필요합니다.

리소스:

작업 유형

arn:aws:codepipeline:region:account:actiontype:owner/category/provider/version

ListPipelineExecutions

작업: codepipeline:ListPipelineExecutions

파이프라인의 가장 최근 실행에 대한 요약 정보를 생성하는 데 필요합니다.

리소스:

파이프라인

arn:aws:codepipeline:region:account:pipeline-name

ListPipelines

작업: codepipeline:ListPipelines

사용자 계정에 연결된 모든 파이프라인의 요약을 작성하는 데 필요합니다.

리소스:

와일드카드가 포함된 파이프라인 ARN(파이프라인 이름 수준의 리소스 수준 권한은 지원되지 않음)

arn:aws:codepipeline:region:account:*

ListTagsForResource

작업: codepipeline:ListTagsForResource

지정된 리소스의 태그를 나열하는 데 필요합니다.

리소스:

작업 유형

arn:aws:codepipeline:region:account:actiontype:owner/category/provider/version

파이프라인

arn:aws:codepipeline:region:account:pipeline-name

Webhook

arn:aws:codepipeline:region:account:webhook:webhook-name

ListWebhooks

작업: codepipeline:ListWebhooks

해당 리전에 대한 계정의 모든 Webhook를 나열하는 데 필요합니다.

리소스:

Webhook

arn:aws:codepipeline:region:account:webhook:webhook-name

PollForJobs

작업: codepipeline:PollForJobs

CodePipeline의 모든 작업 실행에 대한 정보를 검색하는 데 필요합니다.

리소스:

작업 유형

arn:aws:codepipeline:region:account:actiontype:owner/category/provider/version
PollForThirdPartyJobs

작업: codepipeline:PollForThirdPartyJobs

작업자가 작업할 타사 작업이 있는지 판단하는 데 필요합니다. 파트너 작업에만 사용합니다.

리소스: 정책 Resource 요소에는 와일드카드(*)만 지원됩니다.

PutActionRevision

작업: codepipeline:PutActionRevision

CodePipeline에 소스에 대한 새 버전 정보를 보고하는 데 필요합니다.

리소스:

작업

arn:aws:codepipeline:region:account:pipeline-name/stage-name/action-name
PutApprovalResult

작업: codepipeline:PutApprovalResult

CodePipeline에 수동 승인 요청에 대한 반응을 보고하는 데 필요합니다. 유효한 반응은 Approved 및 Rejected입니다.

리소스:

작업

arn:aws:codepipeline:region:account:pipeline-name/stage-name/action-name

Note

이 API는 리소스 수준 권한을 지원합니다. 하지만 IAM 콘솔이나 정책 생성기에서 리소스 ARN을 지정하는 "codepipeline:PutApprovalResult"를 사용하여 정책을 생성하는 경우 오류가 발생할 수 있습니다. 오류가 발생할 경우 IAM 콘솔의 JSON 탭이나 CLI를 사용 하여 정책을 생성합니다.

PutJobFailureResult

작업: codepipeline:PutJobFailureResult

작업자가 파이프라인에 작업을 반환할 때 작업 실패를 보고하는 데 필요합니다. 사용자 지정 작업 에만 사용합니다.

리소스: 정책 Resource 요소에는 와일드카드(*)만 지원됩니다.

PutJobSuccessResult

작업: codepipeline:PutJobSuccessResult

작업자가 파이프라인에 작업을 반환할 때 작업 성공을 보고하는 데 필요합니다. 사용자 지정 작업에만 사용합니다.

리소스: 정책 Resource 요소에는 와일드카드(*)만 지원됩니다.

PutThirdPartyJobFailureResult

작업: codepipeline:PutThirdPartyJobFailureResult

작업자가 파이프라인에 작업을 반환할 때 타사 작업 실패를 보고하는 데 필요합니다. 파트너 작업 에만 사용합니다.

리소스: 정책 Resource 요소에는 와일드카드(*)만 지원됩니다.

PutThirdPartyJobSuccessResult

작업: codepipeline:PutThirdPartyJobSuccessResult

작업자가 파이프라인에 작업을 반환할 때 타사 작업 성공을 보고하는 데 필요합니다. 파트너 작업 에만 사용합니다.

리소스: 정책 Resource 요소에는 와일드카드(*)만 지원됩니다.

PutWebhook

작업: codepipeline:PutWebhook

Webhook를 생성하는 데 필요합니다.

리소스:

Webhook

arn:aws:codepipeline:region:account:webhook:webhook-name

RegisterWebhookWithThirdParty

작업: codepipeline:RegisterWebhookWithThirdParty

리소스:

Webhook가 생성된 후 생성된 Webhook URL을 호출하도록 지원되는 타사를 구성하는 데 필요합니다.

Webhook

arn:aws:codepipeline:region:account:webhook:webhook-name

RetryStageExecution

작업: codepipeline:RetryStageExecution

단계에서 마지막으로 실패한 작업을 재시도함으로써 파이프라인 실행을 재개하는 데 필요합니다.

리소스:

파이프라인

arn:aws:codepipeline:region:account:pipeline-name

StartPipelineExecution

작업: codepipeline:StartPipelineExecution

지정된 파이프라인(구체적으로 파이프라인의 일부로 지정한 소스 위치에 최신 커밋 처리)을 시작하 는데 필요합니다.

리소스:

파이프라인

arn:aws:codepipeline:region:account:pipeline-name

TagResource

작업: codepipeline:TagResource

지정된 리소스에 태그를 지정하는 데 필요합니다.

리소스:

작업 유형

arn:aws:codepipeline:region:account:actiontype:owner/category/provider/version

파이프라인

arn:aws:codepipeline:region:account:pipeline-name

Webhook

arn:aws:codepipeline:region:account:webhook:webhook-name

UntagResource

작업: codepipeline:UntagResource

지정된 리소스에 태그를 지정하는 데 필요합니다.

리소스:

작업 유형

arn:aws:codepipeline:region:account:actiontype:owner/category/provider/version

파이프라인

arn:aws:codepipeline:region:account:pipeline-name

Webhook

arn:aws:codepipeline:region:account:webhook:webhook-name

UpdatePipeline

작업: codepipeline:UpdatePipeline

지정 파이프라인의 구조에 편집 혹은 변경 사항을 업데이트하는 데 필요합니다.

리소스:

파이프라인

arn:aws:codepipeline:region:account:pipeline-name

CodePipeline 서비스 역할 관리

CodePipeline 서비스 역할은 파이프라인에서 사용하는 AWS 리소스에 대한 액세스를 제어하는 하나 이상의 정책으로 구성됩니다. 이 역할에 더 많은 정책을 연결하거나, 역할에 연결된 정책을 편집하거 나,의 다른 서비스 역할에 대한 정책을 구성할 수 있습니다 AWS. 또한 사용자의 파이프라인으로 교차 계정 액세스를 구성할 때 역할에 정책을 연결해야 할 수도 있습니다.

🛕 Important

정책 설명을 수정하거나 다른 정책을 역할에 연결하면 파이프라인이 작동하지 않도록 방지할 수 있습니다. 어느 쪽이든 CodePipeline에 서비스 역할을 수정하기 전에 관련 문제를 확실히 이해하세요. 서비스 역할을 변경한 후에는 반드시 파이프라인을 테스트하십시오.

Note

콘솔에서 2018년 9월 이전에 생성된 서비스 역할은 oneClick_AWS-CodePipeline-Service_ID-Number라는 이름으로 생성됩니다. 2018년 9월 이후에 생성된 서비스 역할은 서비스 역할 이름 형식으로 AWSCodePipelineServiceRole-*Region-Pipeline_Name*을 사용합니다. 예를 들 어 eu-west-2에서 MyFirstPipeline이라는 파이프라인의 경우 콘솔은 역할과 정책을 AWSCodePipelineServiceRole-eu-west-2-MyFirstPipeline이라고 이름을 지정합 니다.

CodePipeline 서비스 역할 정책

CodePipeline 서비스 역할 정책 문에는 파이프라인을 관리하기 위한 최소 권한이 포함되어 있습니다. 서비스 역할 문을 편집하여 사용하지 않는 리소스에 대한 액세스를 제거하거나 추가할 수 있습니다. CodePipeline이 각 작업에 사용하는 최소 필수 권한은 해당 작업 참조를 참조하세요.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowS3BucketAccess",
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketVersioning",
        "s3:GetBucketAcl",
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::[[pipeArtifactBucketNames]]"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "{{accountId}}"
        }
      }
    },
    {
      "Sid": "AllowS30bjectAccess",
      "Effect": "Allow",
      "Action": [
```

```
"s3:PutObject",
        "s3:PutObjectAcl",
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:PutObjectTagging",
        "s3:GetObjectTagging",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": [
        "arn:aws:s3:::[[pipeArtifactBucketNames]]/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "{{accountId}}"
        }
      }
    }
  ]
}
```

Note 정책에서 소스 버킷의 S3 객체에 태그가 있는 경우 다음 권한이 필요합니다. s3:PutObjectTagging s3:GetObjectTagging

s3:GetObjectVersionTagging

CodePipeline 서비스 역할에서 권한 제거

사용하지 않는 리소스에 액세스를 제거하도록 서비스 역할 설명을 편집할 수 있습니다. 예를 들어 Elastic Beanstalk가 포함된 파이프라인이 없으면 Elastic Beanstalk 리소스 액세스를 허용하는 섹션을 제거하도록 정책 설명을 편집할 수 있습니다.

마찬가지로 CodeDeploy가 포함된 파이프라인이 없으면 CodeDeploy 리소스 액세스를 허용하는 섹션 을 제거하도록 정책 설명을 편집할 수 있습니다.

```
{
    "Action": [
        "codedeploy:CreateDeployment",
```

```
"codedeploy:GetApplicationRevision",
    "codedeploy:GetDeployment",
    "codedeploy:GetDeploymentConfig",
    "codedeploy:RegisterApplicationRevision"
],
    "Resource": "*",
    "Effect": "Allow"
},
```

CodePipeline 서비스 역할에 권한 추가

이것을 파이프라인에 사용하기 전에 기본 서비스 역할 정책 설명에 이미 포함되지 않은 AWS 서비스 용 권한으로 서비스 역할 정책 설명을 업데이트해야 합니다.

이는에 대한 CodePipeline에 지원이 추가되기 전에 파이프라인에 사용하는 서비스 역할이 생성된 경우 특히 중요합니다 AWS 서비스.

다음 표에는 다른 AWS 서비스용으로 지원을 추가하는 경우를 보여줍니다.

AWS 서비스	CodePipeline 지원 날짜
CodePipeline 호출 작업 지원이 추가되었습니 다. <u>CodePipeline 호출 작업에 대한 서비스 역할</u> <u>정책 권한</u> 을(를) 참조하세요.	2025년 3월 14일
EC2 작업 지원이 추가되었습니다. <u>EC2 배포 작</u> <u>업에 대한 서비스 역할 정책 권한</u> 을(를) 참조하 세요.	2025년 2월 21일
EKS 작업 지원이 추가되었습니다. <u>서비스 역할</u> <u>정책 권한</u> 을(를) 참조하세요.	2025년 2월 20일
Amazon Elastic Container Registry ECRBuildA ndPublish 작업 지원이 추가되었습니다. <u>서</u> <u>비스 역할 권한: ECRBuildAndPublish 작</u> <u>업</u> 을(를) 참조하세요.	2024년 11월 22일
Amazon Inspector InspectorScan 작업 지원이 추가되었습니다. <u>서비스 역할 권한:</u> <u>InspectorScan 작업</u> 을(를) 참조하세요.	2024년 11월 22일

AWS 서비스	CodePipeline 지원 날짜
명령 작업 지원이 추가되었습니다. <u>서비스 역할</u> <u>권한: 명령 작업</u> 을(를) 참조하세요.	2024년 10월 3일
AWS CloudFormation 작업 지원이 추가되 었습니다. <u>서비스 역할 권한: CloudForm</u> <u>ationStackSet 작업 및 서비스 역할 권한:</u> <u>CloudFormationStackInstances 작업</u> 섹션을 참조하세요.	2020년 12월 30일
CodeCommit 전체 복제 출력 아티팩트 형식 작 업 지원이 추가되었습니다. <u>서비스 역할 권한:</u> <u>CodeCommit 작업</u> 을(를) 참조하세요.	2020년 11월 11일
CodeBuild 배치 빌드 작업 지원이 추가되었습니 다. <u>서비스 역할 권한: CodeCommit 작업</u> 을(를) 참조하세요.	2020년 7월 30일
AWS AppConfig 작업 지원이 추가되었습니다. <u>서비스 역할 권한: AppConfig 작업</u> 을(를) 참 조하세요.	2020년 6월 22일
AWS Step Functions 작업 지원이 추가되었습 니다. <u>서비스 역할 권한: StepFunctions 작</u> <u>업</u> 을(를) 참조하세요.	2020년 5월 27일
AWS CodeStar 연결 작업 지원이 추가되었습니 다. <u>서비스 역할 권한: CodeConnections 작업</u> 을 (를) 참조하세요.	2019년 12월 18일
S3 배포 작업 지원이 추가되었습니다. <u>서비스 역</u> <u>할 권한: S3 배포 작업</u> 을(를) 참조하세요.	2019년 1월 16일
CodeDeployToECS 작업 지원이 추가되었습 니다. <u>서비스 역할 권한: CodeDeployToECS</u> <u>작업</u> 을(를) 참조하세요.	2018년 11월 27일

AWS 서비스	CodePipeline 지원 날짜
Amazon ECR 작업 지원이 추가되었습니다. <u>서</u> <u>비스 역할 권한: Amazon ECR 작업</u> 을(를) 참조 하세요.	2018년 11월 27일
Service Catalog 작업 지원이 추가되었습니다. <u>서비스 역할 권한: Service Catalog 작업</u> 을(를) 참조하세요.	2018년 10월 16일
AWS Device Farm 작업 지원이 추가되었습니 다. <u>서비스 역할 권한: AWS Device Farm 작업</u> 을 (를) 참조하세요.	2018년 7월 19일
Amazon ECS 작업 지원이 추가되었습니다. <u>서</u> <u>비스 역할 권한: Amazon ECS 표준 작업</u> 을(를) 참조하세요.	2017년 12월 12일 / 2017년 7월 21일에 태그 지 정 권한 부여를 위한 옵트인 업데이트
CodeCommit 작업 지원이 추가되었습니다. <u>서</u> <u>비스 역할 권한: CodeCommit 작업</u> 을(를) 참조하 세요.	2016년 4월 18일
AWS OpsWorks 작업 지원이 추가되었습니다. <u>서비스 역할 권한: AWS OpsWorks 작업</u> 을(를) 참조하세요.	2016년 6월 2일
AWS CloudFormation 작업 지원이 추가되었습 니다. <u>서비스 역할 권한: AWS CloudFormation</u> <u>작업</u> 을(를) 참조하세요.	2016년 11월 3일
AWS CodeBuild 작업 지원이 추가되었습니다. <u>서비스 역할 권한: CodeBuild 작업</u> 을(를) 참조하 세요.	2016년 1월 12일
Elastic Beanstalk 작업 지원이 추가되었습니다. <u>서비스 역할 권한: ElasticBeanstalk 배포</u> <u>작업</u> 을(를) 참조하세요.	초기 서비스 출시

AWS 서비스	CodePipeline 지원 날짜
CodeDeploy 작업 지원이 추가되었습니다. <u>서비</u> <u>스 역할 권한: AWS CodeDeploy 작업</u> 을(를) 참 조하세요.	초기 서비스 출시
S3 소스 작업 지원이 추가되었습니다. <u>서비스 역</u> <u>할 권한: S3 소스 작업</u> 을(를) 참조하세요.	초기 서비스 출시

지원되는 서비스에 대한 권한을 추가하려면 다음 단계를 따르십시오.

- 1. 에 로그인 AWS Management Console 하고 <u>https://console.aws.amazon.com/iam/</u> IAM 콘솔을 엽니 다.
- 2. IAM 콘솔 탐색 창에서 역할을 선택한 다음 역할 목록에서 AWS-CodePipeline-Service 역할을 선택합니다.
- 3. 권한 탭으로 가서 인라인 정책의 서비스 역할 정책 열에서 정책 편집을 선택합니다.
- 4. 정책 문서 상자에 필요한 권한을 추가합니다.

Note

IAM 정책을 만들 때 최소 권한 부여의 기본 보안 조언을 따릅니다. 즉, 작업 수행에 필요한 최소한의 권한만 부여합니다. 일부 API 호출은 리소스 기반 권한을 지원하고 액세스 권한을 제한할 수 있습니다. 예를 들어 이 경우DescribeTasks 및 ListTasks를 호출할 때 권한 을 제한하려면 와일드카드 문자(*)를 리소스 ARN 또는 와일드카드 문자(*)가 포함된 리소 스 ARN으로 바꾸면 됩니다. 최소 권한 액세스 권한을 부여하는 정책 생성에 대한 자세한 내 용은 <u>https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-</u> privilege 단원을 참조하세요.

5. 정책 검토를 선택해 정책에 오류가 없는지 확인합니다. 정책에 오류가 없으면 정책 적용를 선택합니다.

CodePipeline에서 로깅 및 모니터링

의 로깅 기능을 사용하여 사용자가 계정에서 수행한 작업과 사용된 리소스를 AWS 확인할 수 있습니 다. 로그 파일에는 다음 정보가 들어 있습니다.

- 작업의 시간과 날짜
- 작업의 소스 IP 주소
- 부족한 권한으로 인해 실패한 작업

로깅 기능은 다음과 같은 AWS 서비스에서 제공됩니다.

- AWS CloudTrail 는에 의해 또는를 대신하여 수행된 AWS API 호출 및 관련 이벤트를 로깅하는 데 사용할 수 있습니다 AWS 계정. 자세한 내용은 <u>를 사용하여 CodePipeline API 호출 로깅 AWS</u> CloudTrail 단원을 참조하십시오.
- Amazon CloudWatch Events를 사용하여 AWS 클라우드 리소스와 실행 중인 애플리케이션을 모니 터링할 수 있습니다 AWS. 정의한 지표에 기반하여 Amazon CloudWatch Events에서 경보를 생성할 수 있습니다. 자세한 내용은 CodePipeline 이벤트 모니터링 단원을 참조하십시오.

에 대한 규정 준수 검증 AWS CodePipeline

AWS 서비스 가 특정 규정 준수 프로그램의 범위 내에 있는지 알아보려면 <u>AWS 서비스 규정 준수 프로</u> <u>그램 제공 범위</u> 섹션을 참조하고 관심 있는 규정 준수 프로그램을 선택합니다. 일반 정보는 <u>AWS 규정</u> 준수 프로그램.

를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다 AWS Artifact. 자세한 내용은 <u>Downloading</u> Reports inDownloading AWS Artifact을 참조하세요.

사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 AWS 서비스 결정됩니다.는 규정 준수를 지원하기 위해 다음 리소스를 AWS 제공합니다.

- <u>보안 규정 준수 및 거버넌스</u> 이러한 솔루션 구현 가이드에서는 아키텍처 고려 사항을 설명하고 보 안 및 규정 준수 기능을 배포하는 단계를 제공합니다.
- <u>HIPAA 적격 서비스 참조</u> HIPAA 적격 서비스가 나열되어 있습니다. 모두 HIPAA 자격이 AWS 서비 스 있는 것은 아닙니다.
- AWS 규정 준수 리소스 -이 워크북 및 가이드 모음은 산업 및 위치에 적용될 수 있습니다.
- <u>AWS 고객 규정 준수 가이드</u> 규정 준수의 관점에서 공동 책임 모델을 이해합니다. 이 가이드에는 여러 프레임워크(미국 국립 표준 기술 연구소(NIST), 결제 카드 산업 보안 표준 위원회(PCI) 및 국제 표준화 기구(ISO) 포함)의 보안 제어에 대한 지침을 보호하고 AWS 서비스 매핑하는 모범 사례가 요 약되어 있습니다.
- AWS Config 개발자 안내서의 <u>규칙을 사용하여 리소스 평가</u> -이 AWS Config 서비스는 리소스 구성 이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가합니다.

- <u>AWS Security Hub</u> 내 보안 상태에 대한 포괄적인 보기를 AWS 서비스 제공합니다 AWS. Security Hub는 보안 컨트롤을 사용하여 AWS 리소스를 평가하고 보안 업계 표준 및 모범 사례에 대한 규정 준수를 확인합니다. 지원되는 서비스 및 제어 목록은 Security Hub 제어 참조를 참조하세요.
- <u>Amazon GuardDuty</u> 의심스러운 활동과 악의적인 활동이 있는지 환경을 모니터링하여 사용자, AWS 계정워크로드, 컨테이너 및 데이터에 대한 잠재적 위협을 AWS 서비스 탐지합니다. GuardDuty 는 특정 규정 준수 프레임워크에서 요구하는 침입 탐지 요구 사항을 충족하여 PCI DSS와 같은 다양 한 규정 준수 요구 사항을 따르는 데 도움을 줄 수 있습니다.
- <u>AWS Audit Manager</u> 이를 AWS 서비스 통해 AWS 사용량을 지속적으로 감사하여 위험과 규정 및 업계 표준 준수를 관리하는 방법을 간소화할 수 있습니다.

의 복원력 AWS CodePipeline

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다. AWS 리전은 물리적으로 분리 되고 격리된 여러 가용 영역을 제공하며,이 가용 영역은 지연 시간이 짧고 처리량이 높으며 중복성이 높은 네트워킹과 연결됩니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

의 인프라 보안 AWS CodePipeline

관리형 서비스인는 AWS 글로벌 네트워크 보안으로 보호 AWS CodePipeline 됩니다. AWS 보안 서비스 및가 인프라를 AWS 보호하는 방법에 대한 자세한 내용은 <u>AWS 클라우드 보안을</u> 참조하세 요. 인프라 보안 모범 사례를 사용하여 AWS 환경을 설계하려면 보안 원칙 AWS Well-Architected Framework의 인프라 보호를 참조하세요.

AWS 게시된 API 호출을 사용하여 네트워크를 통해 CodePipeline에 액세스합니다. 고객은 다음을 지 원해야 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 지원합니 다.

또한 요청은 액세스 키 ID 및 IAM 위탁자와 관련된 보안 암호 액세스 키를 사용하여 서명해야 합니다. 또는 <u>AWS Security Token Service</u>(AWS STS)를 사용하여 임시 자격 증명을 생성하여 요청에 서명할 수 있습니다.

보안 모범 사례

CodePipeline은 자체 보안 정책을 개발하고 구현할 때 고려해야 할 여러 보안 기능을 제공합니다. 다음 모범 사례는 일반적인 지침이며 완벽한 보안 솔루션을 나타내지는 않습니다. 이러한 모범 사례는 환경 에 적절하지 않거나 충분하지 않을 수 있으므로 참고용으로만 사용해 주세요.

파이프라인에 연결하는 소스 리포지토리에 대해 암호화 및 인증을 사용합니다. 다음은 보안을 위한 CodePipeline 모범 사례입니다.

- 토큰이나 암호와 같은 암호를 포함해야 하는 파이프라인 또는 작업 구성을 생성하는 경우 정보가 로 그에 표시되므로 작업 구성에 직접 암호를 입력하거나 파이프라인 수준 또는 AWS CloudFormation 구성에서 정의된 변수의 기본값을 입력하지 마십시오. <u>AWS Secrets Manager 를 사용하여 데이터베</u> 이스 암호 또는 타사 API 키 추적에 설명된 대로 Secrets Manager를 사용하여 암호를 설정 및 저장 한 다음 파이프라인 및 작업 구성에서 참조된 암호를 사용합니다.
- S3 소스 버킷을 사용하는 파이프라인을 생성할 경우, <u>CodePipeline의 Amazon S3에 저장된 아티팩</u> <u>트에 대해 서버 측 암호화 구성</u>에서 설명하는 대로 AWS KMS keys를 관리함으로써 CodePipeline의 Amazon S3에 저장된 아티팩트에 대해 서버 측 암호화를 구성합니다.
- Jenkins 작업 공급자를 사용하는 경우, 파이프라인 빌드 작업이나 테스트 작업에 Jenkins 빌드 공급 자를 사용할 때 EC2 인스턴스에 Jenkins를 설치하고 별도의 EC2 인스턴스 프로파일을 구성합니다. 인스턴스 프로파일이 Jenkins에 Amazon S3에서 파일을 검색하는 등 프로젝트에 대한 작업을 수행 하는 데 필요한 AWS 권한만 부여해야 합니다. Jenkins 인스턴스 프로파일에 대한 역할을 만드는 방 법은 Jenkins 통합에 사용할 IAM 역할 생성</u>의 단계를 참조하십시오.

CodePipeline 파이프라인 구조 참조

CodePipeline을 사용하여 애플리케이션 소스 코드를 빌드, 테스트 및 배포하는 작업을 수행하는 자동 화된 단계의 CI/CD 파이프라인을 구성할 수 있습니다. 파이프라인을 생성할 때 소스 코드를 포함하고 소스 코드 변경을 커밋할 때 파이프라인을 시작하는 S3 버킷, CodeCommit 리포지토리, Bitbucket 리 포지토리 또는 GitHub 리포지토리와 같은 사용 가능한 소스 작업 및 공급자를 선택합니다. 또한 파이프 라인이 실행될 때 자동으로 포함할 작업 및 공급자를 테스트, 빌드 및 배포하도록 선택할 수도 있습니 다. 애플리케이션을 배포하는 DevOps 파이프라인의 개념 예제는 <u>DevOps 파이프라인 예제</u> 섹션을 참 조하세요.

기본적으로에서 성공적으로 생성한 모든 파이프라인 AWS CodePipeline 은 유효한 구조를 갖습니 다. 그러나 JSON 파일을 수동으로 생성하거나 편집하여 파이프라인을 생성하거나에서 파이프라인 을 업데이트 AWS CLI하는 경우 유효하지 않은 구조를 실수로 생성할 수 있습니다. 다음 참조 정보는 파이프라인 구조의 요구 사항과 문제 해결 방법을 더 잘 이해하는 데 도움이 될 수 있습니다. <u>in AWS</u> CodePipeline 할당량 단원에서 모든 파이프라인에 적용되는 제약 조건을 참조하십시오.

다음 섹션에서는 파이프라인 구조에서 상위 수준 파라미터와 해당 위치를 제공합니다. 파이프라인 구 조 요구 사항은 다음 파이프라인 구성 요소 유형에 대한 각 섹션에 자세히 설명되어 있습니다.

- 파이프라인 선언에 대한 필드 참조
- 스테이지 선언에 대한 필드 참조
- 작업 선언에 대한 필드 참조
- 작업 유형별 CodePipeline에서 유효한 작업 공급자 목록
- PollForSourceChanges 파라미터의 유효한 설정 참조 자료
- 각 작업 유형에 대한 유효한 입력 및 출력 아티팩트 참조 자료
- 각 공급자 유형에 유효한 구성 파라미터에 대한 링크 목록

자세한 내용은 CodePipeline API 가이드의 PipelineDeclaration 객체를 참조하세요.

다음 예제 파이프라인 콘솔 보기는 new-github라는 파이프라인, Source, manual, 라는 단계Build, GitHub의 작업(GitHub 앱을 통해), 수동 승인 및 CodeBuild 작업 공급자의 작업을 보여줍니다.

new-github	🗘 Notify 🔻	Edit	Stop execution	Clone pipeline	Release change	
Pipeline type: V2 Execution m	ode: SUPERSEDED					
Source Succeeded Pipeline execution ID: <u>60a18b</u>	a0-b0d6-4a57-					
Source <u>GitHub (Version 2)</u>						
77cc2e44						
77cc2e44 🗹 Source: Merge pull re	quest #5 from /fe	ature-branch	***			-
Disable transition]					
Succeeded Pipeline execution ID: 60a18b	a0-b0d6-4a57			(Start rollback	0
Approval Manual approval 2 Approved - Just now						
View details						
77cc2e44 🗹 Source: Merge pull re	quest #5 from /fe	ature-branch	***			
Disable transition]					_
Build In progress Pipeline execution ID: 60a18b	a0-b0d6-4a57-9aa2-					
Build AWS CodeBuild						
In progress - Just now						
View details						
77cc2e44 🗹 Source: Merge pull re	quest #5 from //fe	ature-branch				

콘솔 다이어그램에서 볼 때 파이프라인 편집 모드를 사용하면 다음 예제와 같이 소스 재정의, 트리거 및 작업을 편집할 수 있습니다.

Edit: Pipeline properties Pipeline type V2 Execution mode SUPERSEDED Edit: Variables Pedere type V2 regard Name Default value Default value Description No variables No variables No variables No variables Edit: Triggers Edit: Triggers Edit: Source Edi	l Save	Cancel	Delete	1		ub	liting: new-gith
Edit: Pripeline properties Poline type V2 Ecucution mode SUPERSEDED Edit: Variables Poline type V2 required Name Default value Default value Description No variables For source action: Source Filters Pull request © Edit: Source Edit: Source Edit: Source Edit: anual Carce Add entry condition Add success condition Add success condition Add success condition Add success				L			<u> </u>
Pipeline type V2 Eacuation mode SUPERSEDED Edit: Variables Pademe type V2 required Name Default value Description No variables No variables defined at the pipeline level in this pipeline. Edit: Triggers Edit: For source action: Source Filters Pull request O Events: Created Revised Closed include branches: master ⁺ Edit: Source E Edit: Source O Edit: Source O Edit: Source O Edit: Add stage Edit: manual Cancel Detere Add action group	Edit					25	dit: Pipeline properties
Edit: Variables Edit. Predire type V2 required Edit. Name Default value Description No variables No variables For source action: Source Edit. Filters Pull request O Pull request O Edit: Source Edit. Source O Edit: Source Edit. Add entry condition Add success condition ♥ Add action group Add failure condition							ipeline type
Edit: Variables SUPPERSUED Edit: Variables Pedient type V2. regarded Name Default value Default value Description No variables No variables defined at the pipeline level in this pipeline. Edit: Triggers Edit: Triggers Edit: Triggers Edit: Source Edit: Source Edit: Source Edit: Source Edit: Source Edit: Source Edit: Source Edit: Source Edit: Source Edit: Source Edit: Source Edit: Source Edit: Source							2
Edit: Variables Pipeline type V2 required Name Default value Description No variables defined at the pipeline level in this pipeline. Edit: Triggers Edit: Triggers Edit: Triggers Edit: Source Edit: Add stage Edit: manual Add entry condition Add success condition Add attion group							xecution mode UPERSEDED
Parties type V2 required Name Default value Description No variables No variables No variables defined at the pipeline level in this pipeline. Edit: Triggers Edit: Triggers Edit: Source Filters Pull request O Pull request O Econtes: Concel Edit: Source Edit: Source Edit: Source Edit: Source Edit: manual Cancel Detete	variables	Edit va					dit: Variables
Name Default value Description No variables No variables defined at the pipeline level in this pipeline. Edit: Triggers For source action: Source Filters Pull request: © Events:: Created Revised Closed Include branches: master* Edit: Source Edit: Source Edit: manual Add astroge Add failure condition + Add action group							ipeline type V2 required
No variables No variables defined at the pipeline level in this pipeline. Edit: Triggers Edit: Triggers For source action: Source Filters Pull request ① Events: ① Created Revised Closed Include branches: master* Edit: Source ① Source ① Edit: source ① Edit: source ① Edit: source ① Edit: source ① Add stage Edit:				Description		Default value	ame
Edit: Triggers Edit For source action: Source Edit Filters Pull request O Pull request O Events: Created Revised Closed Include branches: master* E Source O Edit: Source E Edit: manual Cancel Edit: manual Cancel Add entry condition Add failure condition					No variables		
Edit: Triggers Edit For source action: Source Filters Pull request Created Revised Closed Include branches: master* Edit: Source Edit: Source Edit: source Edit: add success condition Add failure condition Add action group				ne.	es defined at the pipeline level in this pipeli	No varial	
For source action: Source	it triggers	Edit t					dit: Triggers
Filters Pull request Events: Created Revised Closed Include branches: master* Edit: Source Edit: Source Edit: manual Edit: manual Cancel Delete Add entry condition Add success condition ▼ Add failure condition + Add action group							For source action: Source
Pull request Image: Created Revised Closed Include branches: master* Edit: Source Edit: Source Image: Created Revised Closed Source Image: Created Revised Closed Edit: manual Cancel Edit: manual Cancel Add entry condition Add failure condition + Add action group Image: Created Revised Closed							Filters
Events: Created Revised Closed Include branches: master* Edit: Source E Source Edit: manual Cancel Delete Add entry condition Add success condition Add failure condition + Add failure condition						Ð	Pull request (i)
Created Revised Closed Include branches: master* Edit: Source Source ③ H Add stage Edit: manual Cancel Delete Add entry condition Add success condition ▼ Add failure condition						-	Events:
Include branches: master* Edit: Source Source ③ HADD Stage Edit: manual Cancel Delete Add entry condition Add success condition ▼ Add failure condition + Add action group						d	Created Revised Closed
Edit: Source Source Cancel Delete Add entry condition Add success condition Add failure condition + Add action group							Include branches: master*
Edit: Source Source Cancel Delete Add entry condition Add success condition Add failure condition Add action group							
Source () H Add stage Edit: manual Cancel Delete Add entry condition Add success condition Add failure condition H Add action group	Edit stage	Edi					Edit: Source
Hadd stage Edit: manual Cancel Delete Add entry condition Add failure condition							Source
Edit: manual Cancel Delete Add entry condition Add failure condition Image: Cancel condition + Add action group Add failure condition Image: Cancel condition					+ Add stage		
Add entry condition Add success condition Add failure condition + Add action group Add failure condition 	Done	Delete	ncel D	c			Edit: manual
+ Add action group					Add failure condition	Add success condition 🔻	Add entry condition
							+ Add action group

- 주제
- <u>파이프라인 선언</u>
- <u>스테이지 선언</u>
- <u>작업 선언</u>

- CodePipeline에서 유효한 작업 공급자
- <u>PollForSourceChanges 파라미터의 유효한 설정</u>
- 각 작업 유형에 대한 유효한 입력 및 출력 아티팩트
- 각 공급자 유형에 유효한 구성 파라미터

파이프라인 선언

파이프라인 및 파이프라인의 메타데이터 수준에는 다음 파라미터와 구문이 포함된 기본 구조가 있습 니다. 파이프라인 파라미터는 파이프라인에서 수행할 작업 및 스테이지의 구조를 나타냅니다.

자세한 내용은 CodePipeline API 가이드의 PipelineDeclaration 객체를 참조하세요.

다음 예제는 V2 유형 파이프라인에 대한 JSON 및 YAML 모두에서 파이프라인 구조의 파이프라인 및 메타데이터 수준을 보여줍니다.

YAML

```
pipeline:
  name: MyPipeline
  roleArn: >-
    arn:aws:iam::ACCOUNT_ID:role/service-role/AWSCodePipelineServiceRole-us-west-2-
MyPipeline
  artifactStore:
    type: S3
    location: amzn-s3-demo-bucket
  stages:
    . . .
 version: 6
  executionMode: SUPERSEDED
 pipelineType: V2
  variables:
  - name: MyVariable
    defaultValue: '1'
 triggers:
  - providerType: CodeStarSourceConnection
    gitConfiguration:
      sourceActionName: Source
      push:
      - branches:
          includes:
          - main
```
excludes:
- feature-branch
pullRequest:
- events:
- CLOSED
branches:
includes:
- main*
metadata:
<pre>pipelineArn: 'arn:aws:codepipeline:us-west-2:ACCOUNT_ID:MyPipeline'</pre>
<pre>created: '2019-12-12T06:49:02.733000+00:00'</pre>
<u>updated</u> : '2020-09-10T06:34:07.447000+00:00'
pollingDisabledAt: '2020-09-10T06:34:07.447000+00:00'

JSON

```
{
    "pipeline": {
        "name": "MyPipeline",
        "roleArn": "arn:aws:iam::ACCOUNT_ID:role/service-role/
AWSCodePipelineServiceRole-us-west-2-MyPipeline",
        "artifactStore": {
             "type": "S3",
             "location": "amzn-s3-demo-bucket"
        },
        "<u>stages</u>": {
             . . .
    },
        "version": 6,
        "executionMode": "SUPERSEDED",
                 "pipelineType": "V2",
        "<u>variables</u>": [
            {
                 "name": "MyVariable",
                 "defaultValue": "1"
            }
        ],
        "<u>triggers</u>": [
             {
                 "providerType": "CodeStarSourceConnection",
                 "gitConfiguration": {
                     "sourceActionName": "Source",
                     "push": [
```



name

파이프라인의 이름. 파이프라인을 편집하거나 업데이트할 때 파이프라인 이름은 변경할 수 없습니다.

Note

기존 파이프라인의 이름을 바꾸려는 경우 CLI get-pipeline 명령을 사용하여 파이프라인의 구조를 포함하는 JSON 파일을 빌드하면 됩니다. 그런 다음 CLI create-pipeline 명령을 사용하여 해당 구조가 포함된 새 파이프라인을 생성하고 새 이름을 지정하면 됩니다.

roleArn

arn:aws:iam::80398EXAMPLE:role/CodePipeline_Service_Role과 같은 CodePipeline 서비스 역할에 대한 IAM ARN입니다.

콘솔을 사용하여 JSON 구조 대신 파이프라인 서비스 역할 ARN을 보려면 콘솔에서 파이프라인을 선 택한 다음 설정을 선택합니다. 일반 탭 아래에 서비스 역할 ARN 필드가 표시됩니다.

artifactStore OR artifactStores

artifactStore 필드에는 동일한 AWS 리전의 모든 작업이 있는 파이프라인의 아티팩트 버킷 유형 과 위치가 포함됩니다. 파이프라인과 다른 리전에서 작업을 추가하는 경우 artifactStores 매핑은 작업이 실행되는 각 AWS 리전의 아티팩트 버킷을 나열하는 데 사용됩니다. 파이프라인을 생성하거나 편집할 때 파이프라인 리전에 아티팩트 버킷이 있어야 하며 작업을 실행하려는 리전마다 아티팩트 버 킷 하나가 있어야 합니다.

1 Note

파이프라인 구조에서 파이프라인에 artifactStore 또는 artifactStores가 포함 되어야 하나 둘 다 사용할 수는 없습니다. 파이프라인에서 교차 리전 작업을 생성할 경우 artifactStores를 사용해야 합니다.

다음 예제는 artifactStores 파라미터를 사용하는 교차 리전 작업이 있는 파이프라인의 기본 구조 를 보여줍니다.

```
"pipeline": {
    "name": "YourPipelineName",
    "roleArn": "CodePipeline_Service_Role",
    "artifactStores": {
        "us-east-1": {
            "type": "S3",
```

```
"location": "S3 artifact bucket name, such as amzn-s3-demo-bucket"
},
"us-west-2": {
    "type": "S3",
    "location": "S3 artifact bucket name, such as amzn-s3-demo-bucket"
},
"stages": [
    {
    ...
```

type

Amazon S3로 지정된 아티팩트 버킷의 위치 유형입니다.

location

codepipeline-us-east-2-1234567890과 같은 콘솔을 사용하여 파이프라인을 처음 생성할 때 자동으로 생성된 Amazon S3 버킷의 이름 또는 이 목적으로 프로비저닝한 Amazon S3 버킷의 이름

stages

이 파라미터에는 파이프라인의 각 스테이지 이름이 포함됩니다. 파이프라인 구조의 스테이지 수준에 서 파라미터 및 구문에 대한 자세한 내용은 CodePipeline API 가이드의 <u>StageDeclaration</u> 객체를 참조 하세요.

파이프라인 단계 구조는 다음 요구 사항을 충족해야 합니다.

- 파이프라인에는 두 개 이상의 단계가 포함되어야 합니다.
- 파이프라인의 첫 단계에는 하나 이상의 소스 작업이 포함되어야 합니다. 소스 작업만 포함할 수 있습니다.
- 파이프라인의 첫 단계만 소스 작업을 포함할 수 있습니다.
- 각 파이프라인에서 하나 이상의 단계에 소스 작업이 아닌 작업이 포함되어야 합니다.
- 파이프라인 내의 모든 단계 이름은 고유해야 합니다.
- 단계 이름은 CodePipeline 콘솔에서 편집할 수 없습니다. 를 사용하여 스테이지 이름을 편집 AWS CLI하고 스테이지에 하나 이상의 보안 암호 파라미터(예: OAuth 토큰)가 있는 작업이 포함된 경우 해 당 보안 암호 파라미터의 값은 보존되지 않습니다. 파라미터의 값을 수동으로 입력하고(AWS CLI가 반환한 JSON에 네 개의 별표로 마스킹됨) JSON 구조에 포함시켜야 합니다.

▲ Important

30일 이상 비활성 상태인 파이프라인은 파이프라인에 대해 폴링이 비활성화됩니다. 자세한 내 용은 파이프라인 구조 참조의 <u>pollingDisabledAt</u>를 참조하세요. 파이프라인을 폴링에서 이벤트 기반 변경 감지로 마이그레이션하는 단계는 변경 감지 방법을 참조하세요.

version

파이프라인의 버전 번호는 파이프라인을 업데이트할 때마다 자동으로 생성되고 업데이트됩니다.

executionMode

대기열, 대체 또는 병렬 모드에서 실행과 같은 연속 실행에 대한 파이프라인 동작을 지정할 수 있도록 파이프라인 실행 모드를 설정할 수 있습니다. 자세한 내용은 <u>파이프라인 실행 모드 설정 또는 변경</u> 단 원을 참조하십시오.

A Important

PARALLEL 모드의 파이프라인의 경우 스테이지 롤백을 사용할 수 없습니다. 마찬가지로 롤백 결과 유형이 있는 실패 조건은 PARALLEL 모드 파이프라인에 추가할 수 없습니다.

pipelineType

파이프라인 유형은 V2 유형 파이프라인과 같은 파이프라인에서 사용 가능한 구조와 기능을 지정합니 다. 자세한 내용은 파이프라인 유형 단원을 참조하십시오.

variables

파이프라인 수준의 변수는 파이프라인이 생성되고 파이프라인 런타임에서 해결될 때 정의됩니다. 자 세한 내용은 <u>변수 참조</u> 단원을 참조하십시오. 파이프라인 실행 시 전달되는 파이프라인 수준 변수에 대 한 자습서는 자습서: 파이프라인 수준 변수 사용을 참조하세요.

triggers

트리거를 사용하면 특정 브랜치 또는 풀 요청에 대한 변경 사항이 감지되는 경우와 같이 특정 이벤트 유형 또는 필터링된 이벤트 유형에서 시작하도록 파이프라인을 구성할 수 있습니다. 트리거는 GitHub, Bitbucket 및 GitLab과 같이 CodePipeline에서 CodeStarSourceConnection 작업을 사용하는 연 결을 통해 소스 작업에 대해 구성할 수 있습니다. 연결을 사용하는 소스 작업에 대한 자세한 내용은 CodeConnections를 사용하여 파이프라인에 타사 소스 공급자 추가 섹션을 참조하세요.

자세한 내용과 자세한 예제는 단원을 참조하십시오<u>트리거 및 필터링을 사용하여 시작 파이프라인 자</u> 동화.

필터링의 경우 glob 형식의 정규식 패턴은 구문에서 glob 패턴 작업에 설명된 대로 지원됩니다.

Note

CodeCommit 및 S3 소스 작업에는 구성된 변경 감지 리소스(EventBridge 규칙)가 필요하거나 리포지토리에서 소스 변경 내용을 폴링하는 옵션을 사용해야 합니다. Bitbucket, GitHub 또는 GitHub Enterprise Server 소스 작업이 포함된 파이프라인의 경우 webhook를 설정하거나 기본 값을 폴링으로 설정할 필요가 없습니다. 연결 작업은 변경 감지를 관리합니다.

A Important

30일 이상 비활성 상태인 파이프라인은 파이프라인에 대해 폴링이 비활성화됩니다. 자세한 내 용은 파이프라인 구조 참조의 <u>pollingDisabledAt</u>를 참조하세요. 파이프라인을 폴링에서 이벤트 기반 변경 감지로 마이그레이션하는 단계는 변경 감지 방법을 참조하세요.

gitConfiguration 필드

브랜치, 파일 경로, 태그 또는 풀 요청 이벤트를 기준으로 필터링하기 위한 이벤트 유형 및 파라미터를 포함한 트리거의 Git 구성입니다.

JSON 구조의 필드는 다음과 같이 정의됩니다.

- sourceActionName: Git 구성을 사용한 파이프라인 소스 작업의 이름입니다.
- push: 필터링을 사용하여 이벤트를 푸시합니다. 이러한 이벤트는 다양한 푸시 필터와 필터 내부의 AND 작업 간에 OR 작업을 사용합니다.
 - branches: 필터링할 브랜치입니다. 브랜치는 포함과 제외 간에 AND 작업을 사용합니다.
 - includes: 포함될 브랜치에 대해 필터링할 패턴입니다. 포함은 OR 작업을 사용합니다.
 - excludes: 제외될 브랜치에 대해 필터링할 패턴입니다. 제외는 OR 작업을 사용합니다.
 - filePaths: 필터링할 파일 경로 이름입니다.
 - includes: 포함될 파일 경로에 대해 필터링할 패턴입니다. 포함은 OR 작업을 사용합니다.

- excludes: 제외될 파일 경로에 대해 필터링할 패턴입니다. 제외는 OR 작업을 사용합니다.
- tags: 필터링할 태그 이름입니다.
 - includes: 포함될 태그에 대해 필터링할 패턴입니다. 포함은 OR 작업을 사용합니다.
 - excludes: 제외될 태그에 대해 필터링할 패턴입니다. 제외는 OR 작업을 사용합니다.
- pullRequest: 풀 요청 이벤트 및 풀 요청 필터를 필터링하는 풀 요청 이벤트입니다.
 - events: 지정된 대로 열림, 업데이트 또는 닫힘 풀 요청 이벤트를 필터링합니다.
 - branches: 필터링할 브랜치입니다. 브랜치는 포함과 제외 간에 AND 작업을 사용합니다.
 - includes: 포함될 브랜치에 대해 필터링할 패턴입니다. 포함은 OR 작업을 사용합니다.
 - excludes: 제외될 브랜치에 대해 필터링할 패턴입니다. 제외는 OR 작업을 사용합니다.
 - filePaths: 필터링할 파일 경로 이름입니다.
 - includes: 포함될 파일 경로에 대해 필터링할 패턴입니다. 포함은 OR 작업을 사용합니다.
 - excludes: 제외될 파일 경로에 대해 필터링할 패턴입니다. 제외는 OR 작업을 사용합니다.

다음은 푸시 및 풀 요청 이벤트 유형에 대한 트리거 구성의 예입니다.

```
"triggers": [
            {
                 "provider": "Connection",
                 "gitConfiguration": {
                     "sourceActionName": "ApplicationSource",
                     "push": [
                         {
                             "filePaths": {
                                  "includes": [
                                      "projectA/**",
                                      "common/**/*.js"
                                 ],
                                  "excludes": [
                                      "**/README.md",
                                      "**/LICENSE",
                                      "**/CONTRIBUTING.md"
                                 ]
                             },
                             "branches": {
                                  "includes": [
                                      "feature/**",
                                      "release/**"
                                 ],
```

```
"excludes": [
                         "mainline"
                    ]
                },
                "tags": {
                     "includes": [
                         "release-v0", "release-v1"
                     ],
                     "excludes": [
                         "release-v2"
                     ]
                }
            }
        ],
        "pullRequest": [
            {
                "events": [
                    "CLOSED"
                ],
                "branches": {
                     "includes": [
                         "feature/**",
                         "release/**"
                    ],
                     "excludes": [
                         "mainline"
                     ]
                },
                "filePaths": {
                     "includes": [
                         "projectA/**",
                         "common/**/*.js"
                     ],
                    "excludes": [
                         "**/README.md",
                         "**/LICENSE",
                         "**/CONTRIBUTING.md"
                     ]
                }
            }
        ]
    }
}
```

],

에 대한 이벤트 유형 push 필드 포함 및 제외

푸시 이벤트 유형의 Git 구성 필드 수준에 대한 포함 및 제외 동작은 다음 목록에 나와 있습니다.

push (OR operation is used between push and pullRequest or multiples) **filePaths** (AND operation is used between filePaths, branches, and tags) includes (AND operation is used between includes and excludes) **/FILE.md, **/FILE2 (OR operation is used between file path names) excludes (AND operation is used between includes and excludes) **/FILE.md, **/FILE2 (OR operation is used between file path names) **branches** (AND operation is used between filePaths, branches, and tags) includes (AND operation is used between includes and excludes) BRANCH/**", "BRANCH2/** (OR operation is used between branch names) excludes (AND operation is used between includes and excludes) BRANCH/**", "BRANCH2/** (OR operation is used between branch names) tags (AND operation is used between filePaths, branches, and tags) includes (AND operation is used between includes and excludes) TAG/**", "TAG2/** (OR operation is used between tag names) excludes (AND operation is used between includes and excludes) TAG/**", "TAG2/** (OR operation is used between tag names)

에 대한 이벤트 유형 pull request 필드 포함 및 제외

풀 요청 이벤트 유형의 Git 구성 필드 수준에 대한 포함 및 제외 동작은 다음 목록에 나와 있습니다.

pullRequest (OR operation is used between push and pullRequest or multiples)
 events (AND operation is used between events, filePaths, and branches). Includes/
excludes are N/A for pull request events.
 filePaths (AND operation is used between events, filePaths, and branches)
 includes (AND operation is used between includes and excludes)
 **/FILE.md, **/FILE2 (OR operation is used between file path names)
 excludes (AND operation is used between includes and excludes)
 **/FILE.md, **/FILE2 (OR operation is used between file path names)
 excludes (AND operation is used between events, filePaths, and branches)
 includes (AND operation is used between includes and excludes)
 **/FILE.md, **/FILE2 (OR operation is used between file path names)
 branches (AND operation is used between events, filePaths, and branches)
 includes (AND operation is used between includes and excludes)
 BRANCH/**", "BRANCH2/** (OR operation is used between branch names)
 excludes (AND operation is used between includes and excludes)
 BRANCH/**", "BRANCH2/** (OR operation is used between branch names)

metadata

파이프라인 메타데이터 필드는 파이프라인 구조와 다르며 편집할 수 없습니다. 파이프라인을 업데이 트하면 updated 메타데이터 필드의 데이터가 자동으로 변경됩니다.

pipelineArn

파이프라인의 Amazon 리소스 이름(ARN)입니다.

콘솔을 사용하여 JSON 구조 대신 파이프라인 ARN을 보려면 콘솔에서 파이프라인을 선택한 다음 설 정을 선택합니다. 일반 탭 아래에 파이프라인 ARN 필드가 표시됩니다.

created

파이프라인이 생성된 날짜 및 시간입니다.

updated

파이프라인이 마지막으로 업데이트된 날짜 및 시간입니다.

pollingDisabledAt

변경 감지를 위해 폴링하도록 구성된 파이프라인의 경우 폴링이 비활성화된 날짜와 시간입니다.

30일 이상 비활성 상태인 파이프라인은 파이프라인에 대해 폴링이 비활성화됩니다.

- 비활성 파이프라인은 30일 동안 실행하지 않으면 폴링이 비활성화됩니다.
- EventBridge, CodeStar Connections 또는 웹후크를 사용하는 파이프라인은 영향을 받지 않습니다.
- 활성 파이프라인은 영향을 받지 않습니다.

자세한 내용은 CodePipeline API 가이드의 <u>PipelineMetadata</u> 객체에서 pollingDisabledAt 파라 미터를 참조하세요. 파이프라인을 폴링에서 이벤트 기반 변경 감지로 마이그레이션하는 단계는 <u>변경</u> 감지 방법을 참조하세요.

스테이지 선언

파이프라인의 스테이지 수준에는 다음 파라미터와 구문이 포함된 기본 구조가 있습니다. 자세한 내용 은 CodePipeline API 가이드의 <u>StageDeclaration</u> 객체를 참조하세요. 다음 예제는 JSON 및 YAML 모두에서 파이프라인 구조의 스테이지 수준을 보여줍니다. 이 예제는 Source 및 Build라는 두 스테이지를 보여줍니다. 이 예제에는 두 가지 조건이 포함되어 있습니다. 하 나는 onSuccess용이고 다른 하나는 beforeEntry용입니다.

YAML

```
pipeline:
  name: MyPipeline
  roleArn: >-
    arn:aws:iam::ACCOUNT_ID:role/service-role/AWSCodePipelineServiceRole-us-west-2-
MyPipeline
  artifactStore:
    type: S3
    location: amzn-s3-demo-bucket
  stages:
    - name: Source
      actions:
        - name: Source
          . . .
    - name: Build
      actions:
        - name: Build
          . . .
      onSuccess:
        conditions:
        - result: ROLLBACK
          rules:
          - name: DeploymentWindowRule
          . . .
      beforeEntry:
        conditions:
        - result: FAIL
          rules:
          - name: MyLambdaRule
          . . .
  version: 6
metadata:
  pipelineArn: 'arn:aws:codepipeline:us-west-2:ACCOUNT_ID:MyPipeline'
  created: '2019-12-12T06:49:02.733000+00:00'
  updated: '2020-09-10T06:34:07.447000+00:00'
```

{

JSON

```
"pipeline": {
        "name": "MyPipeline",
        "roleArn": "arn:aws:iam::ACCOUNT_ID:role/service-role/
AWSCodePipelineServiceRole-us-west-2-MyPipeline",
        "artifactStore": {
             "type": "S3",
             "location": "amzn-s3-demo-bucket"
        },
        "stages": [
             {
                 "name": "Source",
                 "actions": [
                     {
                          "name": "Source",
                          . . .
                     }
                 ]
            },
             {
                 "name": "Build",
                 "actions": [
                     {
                          "name": "Build",
                          • • •
                     }
                 ],
                 "onSuccess": {
                     "conditions": [
                          {
                              "result": "ROLLBACK",
                              "<u>rules</u>": [
                                  {
                                       "name": "DeploymentWindowRule",
                                       . . .
                                  }
                              ]
                          }
                     ]
                 },
                 "beforeEntry": {
                     "conditions": [
```



name

단계의 이름입니다.

actions

파이프라인의 작업 수준에는 다음 파라미터와 구문이 포함된 기본 구조가 있습니다. 파라미터와 예제 를 보려면 작업 선언 섹션을 참조하세요.

conditions

조건에는 CodePipeline의 규칙 목록에서 사용할 수 있는 하나 이상의 규칙이 포함됩니다. 조건의 모든 규칙이 성공하면 조건이 충족됩니다. 기준이 충족되지 않으면 지정된 결과가 적용되도록 조건을 구성 할 수 있습니다.

다음과 같은 유형의 조건을 구성할 수 있습니다.

- beforeEntry
- onFailure
- onSuccess

자세한 정보와 지침은 스테이지에 대한 조건 구성 섹션을 참조하세요.

rules

각 조건에는 함께 평가되는 규칙의 순서 집합인 규칙 집합이 있습니다. 따라서 조건에서 하나의 규칙이 실패하면 조건이 실패합니다. 파이프라인 런타임 시 규칙 조건을 재정의할 수 있습니다.

사용 가능한 규칙은 규칙 참조에 나와 있습니다. 자세한 내용은 <u>규칙 구조 참조</u>의 규칙 구조 참조를 참 조하세요.

작업 선언

파이프라인의 작업 수준에는 다음 파라미터와 구문이 포함된 기본 구조가 있습니다. 자세한 내용은 CodePipeline API 가이드의 ActionDeclaration 객체를 참조하세요.

다음 예제는 JSON 및 YAML 모두에서 파이프라인 구조의 작업 수준을 보여줍니다.

YAML

```
stages:
        - name: Source
        <u>actions:</u>
        - <u>name</u>: Source
        <u>actionTypeId</u>:
        <u>category</u>: Source
        <u>owner</u>: AWS
        <u>provider</u>: S3
        <u>version</u>: '1'
        <u>runOrder</u>: 1
        <u>configuration</u>:
        PollForSourceChanges: 'false'
        S3Bucket: amzn-s3-demo-bucket
        S3ObjectKey: codedeploy_linux.zip
```

```
outputArtifacts:
        - name: SourceArtifact
      inputArtifacts: []
      region: us-west-2
     namespace: SourceVariables
- name: Build
  actions:
    - name: Build
      actionTypeId:
        category: Build
        owner: AWS
        provider: CodeBuild
        version: '1'
     runOrder: 1
      configuration:
        EnvironmentVariables: >-
          [{"name":"ETag","value":"#{SourceVariables.ETag}","type":"PLAINTEXT"}]
        ProjectName: my-project
     outputArtifacts:
        - name: BuildArtifact
      inputArtifacts:
        - name: SourceArtifact
      region: us-west-2
     namespace: BuildVariables
     runOrder: 1
     configuration:
        CustomData: >-
          Here are the exported variables from the build action: S3 ETAG:
          #{BuildVariables.ETag}
     outputArtifacts: []
      inputArtifacts: []
      region: us-west-2
```

JSON

```
. .
"stages": [
{
"name": "Source",
"<u>actions</u>": [
{
```

```
"name": "Source",
                         "actionTypeId": {
                             "category": "Source",
                             "owner": "AWS",
                             "provider": "S3",
                             "version": "1"
                         },
                         "runOrder": 1,
                         "configuration": {
                             "PollForSourceChanges": "false",
                             "S3Bucket": "amzn-s3-demo-bucket",
                             "S3ObjectKey": "aws-codepipeline-s3-aws-
codedeploy_linux.zip"
                         },
                         "outputArtifacts": [
                             {
                                 "name": "SourceArtifact"
                             }
                         ],
                         "inputArtifacts": [],
                         "region": "us-west-2",
                         "namespace": "SourceVariables"
                     }
                ]
            },
            {
                "name": "Build",
                "actions": [
                     {
                         "name": "Build",
                         "actionTypeId": {
                             "category": "Build",
                             "owner": "AWS",
                             "provider": "CodeBuild",
                             "version": "1"
                         },
                         "runOrder": 1,
                         "configuration": {
                             "EnvironmentVariables": "[{\"name\":\"ETag\",\"value\":
\"#{SourceVariables.ETag}\",\"type\":\"PLAINTEXT\"}]",
                             "ProjectName": "my-build-project"
                         },
                         "outputArtifacts": [
                             {
```



공급자 유형에 적합한 예제 configuration 세부 정보 목록을 보려면 <u>각 공급자 유형에 유효한 구성</u> <u>파라미터</u> 항목을 참조하십시오.

작업 구조는 다음 요구 사항을 충족해야 합니다.

- 단계 내의 모든 작업 이름은 고유해야 합니다.
- 각 파이프라인에는 소스 작업이 필요합니다.
- 연결을 사용하지 않는 소스 작업은 변경 감지를 구성하거나 변경 감지를 끄도록 구성할 수 있습니다.
 변경 감지 방법을 참조하세요.
- 같은 단계에 있든 다음 단계에 있든 모든 작업에 적용되지만, 입력 아티팩트가 출력 아티팩트를 제공 했던 작업으로부터 순서상 반드시 다음 단계일 필요는 없습니다. 동시 작업이 서로 다른 출력 아티팩 트 번들을 선언할 수 있으며, 그러면 차례로 다른 다음 작업에서 이를 사용하게 됩니다.
- 배포 위치로 Amazon S3 버킷을 사용할 때 객체 키도 지정합니다. 객체 키는 파일 이름(객체) 또는 접 두사(폴더 경로)와 파일 이름의 조합일 수 있습니다. 변수를 사용하여 파이프라인에서 사용할 위치 이름을 지정할 수 있습니다. Amazon S3 배포 작업은 Amazon S3 객체 키에서 다음 변수 사용을 지 원합니다.

Amazon S3에서 변수 사용

변수	콘솔 입력의 예	출력
datetime	js-application/{datetime}.zip	다음 형식의 UTC 타임스탬프: <yyyy>- <mm>-DD>_<hh>-<mm>-<ss></ss></mm></hh></mm></yyyy>

변수	콘솔 입력의 예	출력
		예시
		Js-application/2019-01-10_07-39-57.zip
uuid	js-application/{uuid}.zip	UUID는 다른 식별자와 다른 것으로 보 장되는 전 세계적으로 고유한 식별자 입니다. UUID는 <8자리>-<4자리>-4자 리>-<4자리>-<12자리>의 형식(16진수 형식의 모든 숫자)으로 되어 있습니다.
		예시
		js-application/54a60075-b96a-4bf3-90 13-db3a9EXAMPLE.zip

name

작업의 이름.

region

공급자가 인 작업의 경우 리소스 AWS 서비스 AWS 리전 의 입니다.

교차 리전 작업은 Region 필드를 사용하여 작업이 생성될 AWS 리전 을 지정합니다. 이 작업을 위해 생성된 AWS 리소스는 region 필드에 제공된 것과 동일한 리전에서 생성해야 합니다. 다음 작업 유형 에는 교차 리전 작업을 생성할 수 없습니다.

- 소스 작업
- 타사 공급자 기준 작업
- 사용자 지정 공급자 기준 작업

roleArn

선언된 작업을 수행하는 IAM 서비스의 ARN입니다. 이는 파이프라인 수준에서 지정된 roleArn을 통해 가정됩니다.

namespace

작업은 변수로 구성할 수 있습니다. namespace 필드를 사용하여 실행 변수에 대한 네임스페이스 및 변수 정보를 설정합니다. 실행 변수 및 작업 출력 변수에 대한 참조 정보는 <u>변수 참조</u> 단원을 참조하십 시오.

1 Note

Amazon ECR, Amazon S3 또는 CodeCommit 소스의 경우 입력 변환 항목을 사용하여 소스 재 정의를 생성하여 파이프라인 이벤트에 EventBridgerevisionValue의를 사용할 수도 있습니 다. 여기서 revisionValue는 객체 키, 커밋 또는 이미지 ID의 소스 이벤트 변수에서 파생됩 니다. 자세한 내용은 , <u>Amazon ECR 소스 작업 및 EventBridge 리소스 이벤트에 대해 활성화된</u> <u>소스를 사용하여 Amazon S3 소스 작업에 연결</u>또는의 절차에 포함된 입력 변환 항목의 선택적 단계를 참조하세요<u>CodeCommit 소스 작업 및 EventBridge</u>.

actionTypeId

작업 유형 ID는 다음 4개 필드의 조합으로 식별됩니다.

category

소스 작업과 같은 파이프라인의 작업 또는 단계의 유형입니다. 각 작업 유형에는 유효한 특정 작업 공 급자 세트가 있습니다. 작업 유형별 유효한 공급자 목록은 <u>작업 구조 참조</u> 섹션을 참조하세요.

CodePipeline에 대해 유효한 actionTypeId 범주(작업 유형):

- Source
- Build
- Approval
- Deploy
- Test
- Invoke
- Compute

owner

현재 지원되는 모든 작업 유형에서 유일하게 유효한 소유자 문자열은 AWS, ThirdParty 또는 Custom입니다. 특정 작업에 유효한 소유자 문자열은 작업 구조 참조 섹션을 참조하세요.

자세한 내용은 CodePipeline API 참조를 참조하세요.

version

작업의 버전입니다.

provider

CodeBuild와 같은 작업 공급자입니다.

• 작업 범주별로 유효한 공급자 유형은 범주에 따라 다릅니다. 예를 들어 소스 작업 범주의 경우 유효 한 공급자 유형은 S3, CodeStarSourceConnection, CodeCommit 또는 Amazon ECR입니다. 이 예제는 S3 공급자가 있는 소스 작업의 구조를 보여줍니다.

```
"actionTypeId": {
    "category": "Source",
    "owner": "AWS",
    "version": "1",
    "provider": "S3"},
```

InputArtifacts

이 필드에는 작업 범주에 대해 지원되는 경우 입력 아티팩트 구조가 포함됩니다. 작업의 입력 아티팩트 는 이전 작업에서 선언된 출력 아티팩트와 정확히 일치해야 합니다. 예를 들어 이전 작업에 다음 선언 이 포함되고,

```
"outputArtifacts": [
    {
        "MyApp"
    }
],
```

다른 출력 아티팩트가 없는 경우 다음 작업의 입력 아티팩트는 이와 같아야 합니다.

```
"inputArtifacts": [
```

InputArtifacts

```
{
"MyApp"
}
],
```

예를 들어 소스 작업은 파이프라인의 첫 번째 작업이므로 입력 아티팩트를 가질 수 없습니다. 그러나 소스 작업에는 항상 다음 작업으로 처리되는 출력 아티팩트가 있습니다. 소스 작업의 출력 아티팩트는 소스 리포지토리의 애플리케이션 파일로, 압축되어 아티팩트 버킷을 통해 제공되며 빌드 명령을 사용 하여 애플리케이션 파일에 작용하는 CodeBuild 작업과 같은 다음 작업으로 처리됩니다.

출력 아티팩트를 가질 수 없는 작업의 예로서 배포 작업에는 출력 아티팩트가 없습니다. 이러한 작업은 일반적으로 파이프라인의 마지막 작업이기 때문입니다.

name

작업의 입력 아티팩트에 대한 아티팩트 이름입니다.

outputArtifacts

파이프라인에서 출력 아티팩트 이름은 고유해야 합니다. 예를 들어 한 파이프라인에 "MyApp"이라는 출력 아티팩트가 있는 작업과 "MyBuiltApp"이라는 출력 아티팩트가 있는 또 하나의 작업이 포함될 수 있습니다. 하지만 한 파이프라인에 "MyApp"이라는 출력 아티팩트가 있는 두 개의 작업이 포함될 수 없습니다.

이 필드에는 작업 범주에 대해 지원되는 경우 출력 아티팩트 구조가 포함됩니다. 작업의 출력 아티팩 트는 이전 작업에서 선언된 출력 아티팩트와 정확히 일치해야 합니다. 예를 들어 이전 작업에 다음 선 언이 포함되고,

```
"outputArtifacts": [
    {
        "MyApp"
    }
],
```

다른 출력 아티팩트가 없는 경우 다음 작업의 입력 아티팩트는 이와 같아야 합니다.

```
"inputArtifacts": [
    {
        "MyApp"
    }
],
```

예를 들어 소스 작업은 파이프라인의 첫 번째 작업이므로 입력 아티팩트를 가질 수 없습니다. 그러나 소스 작업에는 항상 다음 작업으로 처리되는 출력 아티팩트가 있습니다. 소스 작업의 출력 아티팩트는 소스 리포지토리의 애플리케이션 파일로, 압축되어 아티팩트 버킷을 통해 제공되며 빌드 명령을 사용 하여 애플리케이션 파일에 작용하는 CodeBuild 작업과 같은 다음 작업으로 처리됩니다.

출력 아티팩트를 가질 수 없는 작업의 예로서 배포 작업에는 출력 아티팩트가 없습니다. 이러한 작업은 일반적으로 파이프라인의 마지막 작업이기 때문입니다.

name

작업의 출력 아티팩트에 대한 아티팩트 이름입니다.

configuration(작업 공급자별)

작업 구성에는 공급자 유형에 적합한 세부 정보와 파라미터가 포함되어 있습니다. 아래 섹션에서 예제 작업 구성 파라미터는 S3 소스 작업에 따라 다릅니다.

작업 구성 및 입력/출력 아티팩트 제한은 작업 공급자에 따라 다를 수 있습니다. 작업 공급자별 작업 구 성 예제 목록은 <u>각 공급자 유형에 유효한 구성 파라미터</u>의 <u>작업 구조 참조</u> 및 표를 참조하세요. 이 표는 각 공급자 유형에 대한 작업 참조 링크를 제공하며, 각 작업에 대한 구성 파라미터를 자세히 나열합니 다. 각 작업 공급자에 대한 입력 및 출력 아티팩트 제한이 있는 표는 <u>각 작업 유형에 대한 유효한 입력</u> 및 출력 아티팩트 섹션을 참조하세요.

다음 고려 사항은 작업 수행에 적용됩니다.

- 소스 작업에는 입력 아티팩트가 없으며 배포 작업에는 출력 아티팩트가 없습니다.
- S3와 같은 연결을 사용하지 않는 소스 작업 공급자의 경우 PollForSourceChanges 파라미터 를 사용하여 변경 사항이 감지될 때 파이프라인을 자동으로 시작할지 여부를 지정해야 합니다. PollForSourceChanges 파라미터의 유효한 설정을(를) 참조하세요.
- 파이프라인을 시작하거나 변경 감지를 비활성화하도록 자동 변경 감지를 구성하려면 <u>소스 작업 및</u> 변경 감지 방법 섹션을 참조하세요.
- 필터링을 사용하여 트리거를 구성하려면 연결에 소스 작업을 사용한 다음 <u>트리거 및 필터링을 사용</u> 하여 시작 파이프라인 자동화 섹션을 참조하세요.
- 각 작업의 출력 변수는 변수 참조 섹션을 참조하세요.

Note

Amazon ECR, Amazon S3 또는 CodeCommit 소스의 경우 입력 변환 항목을 사용하여 소스 재정의를 생성하여 파이프라인 이벤트에 EventBridgerevisionValue의를 사용할 수도 있 습니다. 여기서 revisionValue는 객체 키, 커밋 또는 이미지 ID의 소스 이벤트 변수에서 파생됩니다. 자세한 내용은 , <u>Amazon ECR 소스 작업 및 EventBridge 리소스</u> <u>이벤트에 대해</u> <u>활성화된 소스를 사용하여 Amazon S3 소스 작업에 연결</u>또는의 절차에 포함된 입력 변환 항 목의 선택적 단계를 참조하세요CodeCommit 소스 작업 및 EventBridge.

▲ Important

30일 이상 비활성 상태인 파이프라인은 파이프라인에 대해 폴링이 비활성화됩니다. 자세한 내용은 파이프라인 구조 참조의 <u>pollingDisabledAt</u>를 참조하세요. 파이프라인을 폴링에서 이 벤트 기반 변경 감지로 마이그레이션하는 단계는 <u>변경 감지 방법을</u> 참조하세요.

Note

CodeCommit 및 S3 소스 작업에는 구성된 변경 감지 리소스(EventBridge 규칙)가 필요하거나 리포지토리에서 소스 변경 내용을 폴링하는 옵션을 사용해야 합니다. Bitbucket, GitHub 또는 GitHub Enterprise Server 소스 작업이 포함된 파이프라인의 경우 webhook를 설정하거나 기본 값을 폴링으로 설정할 필요가 없습니다. 연결 작업은 변경 감지를 관리합니다.

runOrder

스테이지 내 작업의 실행 순서를 나타내는 양의 정수입니다. 스테이지의 병렬 작업은 동일한 정수를 갖 는 것으로 표시됩니다. 예를 들어 실행 순서가 2인 두 작업은 스테이지의 첫 번째 작업이 실행된 후 병 렬로 실행됩니다.

작업의 기본 run0rder 값은 1입니다. 이 값은 양의 정수(자연수)여야 합니다. 분수, 소수, 음수나 0을 사용할 수 없습니다. 작업 순서를 지정하려면 첫 번째 작업에 가장 작은 숫자를, 나머지 작업에는 각각 의 순서에 따라 더 큰 숫자를 사용합니다. 동시 작업을 지정하려면 동시에 실행할 각 작업에 동일한 정 수를 사용합니다. 콘솔에서 실행하려는 단계의 수준에서 작업 그룹 추가를 선택하여 작업의 직렬 시퀀 스를 지정하거나 작업 추가를 선택하여 병렬 시퀀스를 지정할 수 있습니다. 작업 그룹이란 동일한 수준 에 있는 하나 이상의 동작으로 구성된 실행 순서를 말합니다.

예를 들어 한 단계에서 세 가지 작업이 순서에 따라 실행되도록 하려면 첫 번째 작업의 run0rder 값으로 1을, 두 번째 작업의 run0rder 값으로 2를, 세 번째 작업의 run0rder 값으로 3을 부여합니다. 하지만 두 번째와 세 번째 작업이 동시에 실행되도록 하려면 첫 번째 작업의 run0rder 값으로 1을, 두 번째와 세 번째 작업의 run0rder 값으로 2를 부여합니다.

Note

연속 작업의 숫자 지정을 정확히 순서대로 할 필요는 없습니다. 예를 들어 순서대로 해야 하는 3개 작업이 있고 두 번째 작업을 제거하기로 한 경우, 세 번째 작업의 runOrder 값에 숫자를 다시 지정할 필요는 없습니다. 작업 (3)의 runOrder 값이 첫 번째 작업 (1)의 runOrder 값보 다 크기 때문에 단계에서 첫 번째 작업 후 순서대로 실행됩니다.

CodePipeline에서 유효한 작업 공급자

파이프라인 구조 형식은 파이프라인에서 작업 및 스테이지 빌드에 사용됩니다. 작업 유형은 작업 범주 및 공급자 유형으로 구성됩니다.

각 작업 범주에는 유효한 작업 공급자 목록이 있습니다. 각 작업 범주에 대해 유효한 작업 공급자를 참 조하려면 <mark>작업 구조 참조</mark> 섹션을 참조하세요.

각 작업 범주에는 지정된 공급자 집합이 있습니다. Amazon S3와 같은 각 작업 공급자에는 파이프라인 구조의 작업 범주에 있는 Provider 필드에서 사용해야 하는 공급자 이름(예: S3)이 있습니다.

파이프라인 구조의 작업 범주 섹션에 있는 Owner 필드에는 세 가지 유효한 값(AWS, ThirdParty 및 Custom)이 있습니다.

작업 공급자의 공급자 이름과 소유자 정보를 찾으려면 <u>작업 구조 참조</u> 또는 <u>각 작업 유형에 대한 유효</u> 한 입력 및 출력 아티팩트 섹션을 참조하세요.

이 표에는 작업 유형별로 유효한 공급자가 나열되어 있습니다.

Note

Bitbucket, GitHub 또는 GitHub Enterprise Server 작업에 대해서는 <u>Bitbucket Cloud</u>, <u>GitHub, GitHub Enterprise Server, GitLab.com 및 GitLab 자체 관리형 작업용</u> CodeStarSourceConnection 작업 참조 주제를 참조하세요.

작업 유형별로 유효한 작업 공급자

작업 범주	유효한 작업 공급자	파이프라인 유형 지 원	작업 참조
소스	Amazon S3	V1, V2	<u>Amazon S3 소스 작업 참</u> <u>조</u>
	Amazon ECR	V1, V2	<u>Amazon ECR 소스 작업</u> <u>참조</u>
	CodeCommit	V1, V2	<u>CodeCommit 소스 작업</u> <u>참조</u>
	CodeStarSourceConn ection(Bitbucket, GitHub, GitHub Enterprise Server 작업 용)	V1, V2	Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com 및 GitLab 자체 관리형 작업 용 CodeStarSourceConn ection
빌드	Amazon ECR ECRBuildA ndPublish 작업	V2 전용	<u>ECRBuildA</u> <u>ndPublish 빌드 작업</u> <u>참조</u>
	CodeBuild	V1, V2	<u>AWS CodeBuild 빌드 및</u> <u>테스트 작업 참조</u>
	명령 작업(컴퓨팅 참조)	V2 전용	
	사용자 지정 CloudBees	V1, V2	<u>각 작업 유형에 대한 유효</u> 한 입력 및 출력 아티팩트
	사용자 지정 Jenkins	V1, V2	<u>각 작업 유형에 대한 유효</u> 한 입력 및 출력 아티팩트
	사용자 지정 TeamCity	V1, V2	<u>각 작업 유형에 대한 유효</u> 한 입력 및 출력 아티팩트

작업 범주	유효한 작업 공급자	파이프라인 유형 지 원	작업 참조
테스트	CodeBuild	V1, V2	<u>AWS CodeBuild 빌드 및</u> <u>테스트 작업 참조</u>
	AWS Device Farm	V1, V2	<u>각 작업 유형에 대한 유효</u> 한 입력 및 출력 아티팩트
	사용자 지정 BlazeMeter	V1, V2	<u>각 작업 유형에 대한 유효</u> 한 입력 및 출력 아티팩트
	ThirdParty GhostInspector		<u>각 작업 유형에 대한 유효</u> 한 입력 및 출력 아티팩트
	사용자 지정 Jenkins		<u>각 작업 유형에 대한 유효</u> 한 입력 및 출력 아티팩트
	ThirdParty Micro Focus StormRunner 로드		<u>각 작업 유형에 대한 유효</u> 한 입력 및 출력 아티팩트
	ThirdParty Nouvola		<u>각 작업 유형에 대한 유효</u> 한 입력 및 출력 아티팩트
	ThirdParty Runscope		<u>각 작업 유형에 대한 유효</u> 한 입력 및 출력 아티팩트
Deploy	Amazon S3		<u>Amazon S3 배포 작업 참</u> <u>조</u>
	AWS CloudFormation		<u>AWS CloudFormation 배</u> <u>포 작업 참조</u>
	CodeDeploy		<u>각 작업 유형에 대한 유효</u> 한 입력 및 출력 아티팩트
	EC2 배포 작업	V2 전용	<u>Amazon EC2 작업 참조</u>
	Amazon ECS		<u>각 작업 유형에 대한 유효</u> 한 입력 및 출력 아티팩트

작업 범주	유효한 작업 공급자	파이프라인 유형 지 원	작업 참조
	Amazon ECS(블루/그린) (CodeDeployToECS 작업)		<u>각 작업 유형에 대한 유효</u> 한 입력 및 출력 아티팩트
	Amazon EKS 작업	V2 전용	<u>???</u>
	Elastic Beanstalk		<u>각 작업 유형에 대한 유효</u> 한 입력 및 출력 아티팩트
	AWS AppConfig		<u>AWS AppConfig 배포 작</u> <u>업 참조</u>
	AWS OpsWorks		<u>각 작업 유형에 대한 유효</u> 한 입력 및 출력 아티팩트
	Service Catalog		<u>각 작업 유형에 대한 유효</u> 한 입력 및 출력 아티팩트
	Amazon Alexa		<u>각 작업 유형에 대한 유효</u> 한 입력 및 출력 아티팩트
	사용자 지정 XebiaLabs		<u>각 작업 유형에 대한 유효</u> 한 입력 및 출력 아티팩트
승인	수동		<u>각 작업 유형에 대한 유효</u> 한 입력 및 출력 아티팩트
간접 호출	CodePipeline 호출 작업		<u>AWS CodePipeline 작업</u> <u>참조 호출</u>
	AWS Lambda		<u>AWS Lambda 작업 참조</u> <u>호출</u>
	AWS Step Functions		<u>AWS Step Functions 작</u> <u>업 참조 호출</u>

작업 범주	유효한 작업 공급자	파이프라인 유형 지 원	작업 참조
	InspectorScan		<u>Amazon Inspector</u> <u>InspectorScan 호출</u> <u>작업 참조</u>
컴퓨팅	명령 작업		명령 작업 참조

CodePipeline의 일부 작업 유형은 일부 AWS 리전에서만 사용할 수 있습니다. AWS 리전에서 작업 유 형을 사용할 수 있지만 해당 작업 유형의 AWS 공급자는 사용할 수 없습니다.

각 작업 공급자에 대한 자세한 내용은 <u>CodePipeline 작업 유형과의 통합</u> 항목을 참조하십시오.

PollForSourceChanges 파라미터의 유효한 설정

PollForSourceChanges 파라미터 기본값은 파이프라인 생성에 사용된 방법에 따라 결정됩니다(아 래 표 참조). 많은 경우 PollForSourceChanges 파라미터 기본값은 true이며 비활성화되어야 합니 다.

PollForSourceChanges 파라미터를 기본값이 true일 때 다음을 수행해야 합니다.

- PollForSourceChanges 파라미터를 JSON 파일 또는 AWS CloudFormation 템플릿에 추가합니다.
- 변경 감지 리소스(CloudWatch Events 규칙(해당 시))를 생성합니다.
- PollForSourceChanges 파라미터를 false로 설정합니다.

CloudWatch 이벤트 규칙 또는 webhook을 생성하는 경우 파라미터를 false로 설정하여 파이 프라인을 한 번 이상 트리거하지 않도록 해야 합니다.

PollForSourceChanges 파라미터는 Amazon ECR 소스 작업에 사용하지 않아야 합니다.

Note

• PollForSourceChanges 파라미터 기본값

소스	생성 방법	예제 '구성' JSON 구조 출력
CodeCommit	파이프라인은 콘솔로 생성되며 변경 감지 리 소스는 콘솔에 의해 생성됩니다. 이 파라미터 는 파이프라인 구조 출력에 표시되고 기본값은 false입니다.	BranchName": "main", "PollForSourceCha nges": "false", "RepositoryName": "my- repo"
	파이프라인은 CLI 또는를 사용하여 생성 AWS CloudFormation되며 PollForSo urceChanges 파라미터는 JSON 출력에 표 시되지 않지만 .2로 설정됩니다true.	BranchName": "main", "RepositoryName": "my- repo"
Amazon S3	파이프라인은 콘솔로 생성되며 변경 감지 리 소스는 콘솔에 의해 생성됩니다. 이 파라미터 는 파이프라인 구조 출력에 표시되고 기본값은 false입니다.	<pre>"S3Bucket": "my-bucke t", "S3ObjectKey": "object.zip", "PollForSourceChanges" : "false"</pre>
	파이프라인은 CLI 또는를 사용하여 생성 AWS CloudFormation되며 PollForSo urceChanges 파라미터는 JSON 출력에 표 시되지 않지만 .2로 설정됩니다true.	"S3Bucket": "my-bucke t", "S3ObjectKey": "object.zip"
GitHub	파이프라인은 콘솔로 생성되며 변경 감지 리 소스는 콘솔에 의해 생성됩니다. 이 파라미터 는 파이프라인 구조 출력에 표시되고 기본값은 false입니다.	<pre>"Owner": "MyGitHubA ccountName ", "Repo": " MyGitHubR epositoryName " "PollForSourceCh anges": "false", "Branch": " main" "OAuthToken": " ****"</pre>

소스	생성 방법	예제 '구성' JSON 구조 출력
	파이프라인은 CLI 또는를 사용하여 생성 AWS CloudFormation되며 PollForSo urceChanges 파라미터는 JSON 출력에 표 시되지 않지만 .2로 설정됩니다true.	<pre>"Owner": "MyGitHubA ccountName ", "Repo": "MyGitHubR epositoryName ", "Branch": " main", "OAuthToken": " ****"</pre>
	2 PollForSourceChanges 가 JSON 구조 또 릿에 추가된 경우 다음과 같이 표시됩니다.	는 AWS CloudFormation 템플

³ 각 소스 공급자에 적용되는 변경 감지 리소스에 대한 자세한 내용은 <u>변경 감지</u> <u>방법</u>을 참조하세요.

각 작업 유형에 대한 유효한 입력 및 출력 아티팩트

작업 유형 및 공급자에 따라 입력 및 출력 아티팩트의 숫자는 다음과 같을 수 있습니다.

작업 유형 아티팩트 제약

소유자	작업 유형	공급자	유효한 입력 아티 팩트 숫자	유효한 출력 아티 팩트 숫자
AWS	소스	S3	0	1
AWS	소스	CodeCommit	0	1
AWS	소스	ECR	0	1
ThirdParty	소스	CodeStarS ourceConn ection	0	1
AWS	빌드	CodeBuild	1~5	0~5

AWS CodePipeline

소유자	작업 유형	공급자	유효한 입력 아티 팩트 숫자	유효한 출력 아티 팩트 숫자
AWS	테스트	CodeBuild	1~5	0~5
AWS	테스트	DeviceFarm	1	0
AWS	승인	ThirdParty	0	0
AWS	배포	S3	1	0
AWS	배포	CloudForm ation	0 ~ 10	0 ~ 1
AWS	배포	CodeDeploy	1	0
AWS	배포	ElasticBe anstalk	1	0
AWS	배포	0psWorks	1	0
AWS	배포	ECS	1	0
AWS	배포	ServiceCa talog	1	0
AWS	간접 호출	Lambda	0~5	0~5
ThirdParty	배포	AlexaSkil lsKit	1~2	0
Custom	빌드	Jenkins	0~5	0~5
Custom	테스트	Jenkins	0~5	0~5
Custom	지원되는 모든 범 주	사용자 정의 작업 에 지정된 대로	0~5	0~5

각 공급자 유형에 유효한 구성 파라미터

이 섹션에는 각 작업 공급자에 대한 유효한 configuration 파라미터가 나열되어 있습니다.

모든 작업에는 유효한 작업 구성이 있어야 하며 이는 해당 작업의 공급자 유형에 따라 다릅니다. 다음 표에는 유효한 공급자 유형별로 필요한 작업 구성 요소가 수록되어 있습니다.

공급자 유형별 작업 구성 속성

공급자 이름	작업 유형의 공급자 이름	구성 속성	필수/선택
Amazon S3(배포 작 업 공급자)	Amazon S3 배포 작업 조하세요 <u>Amazon S3 비</u>	파라미터와 관련된 예제를 포함한 자세함 <u>H포 작업 참조</u> .	한 내용은 섹션을 참
Amazon S3(소스 작 업 공급자)	Amazon S3 소스 작업 파라미터에 관련된 예제를 포함하여 자세한 내용을 보려면 <u>Amazon S3 소스 작업 참조</u> 단원을 참조하세요.		
Amazon ECR	Amazon ECR 파라미터에 관련된 예제를 포함하여 자세한 내용을 보려면 <u>Amazon</u> <u>ECR 소스 작업 참조</u> 단원을 참조하세요.		
CodeCommi t	CodeCommit 파라미터에 관련된 예제를 포함하여 자세한 내용을 보려면 <u>CodeCommit 소스 작업 참조</u> 단원을 참조하세요.		
Bitbucket, GitHub(Gi tHub 앱 사 용), GHES 및 GitLab 에 대한 CodeStarS ourceConn ection 작업	작업 구성의 예를 포함	하여 자세한 내용은 섹션을 참조하세요	<u> ²성 파라미터</u> .
GitHub(OA uth 앱 사용)	GitHub 파라미터에 관련 <u>앱 사용) 소스 작업 참</u> 조	련된 예제를 포함하여 자세한 내용을 보려 도단원을 참조하십시오. 이는 버전 1 Gitl	持면 <u>GitHub(OAuth</u> Hub 작업입니다.

공급자 이름	작업 유형의 공급자 이름	구성 속성	필수/선택		
AWS CloudForm ation	AWS CloudFormation 파라미터와 관련된 예제를 포함한 자세한 내용은 단원을 참 조하십시오 <u>AWS CloudFormation 배포 작업 참조</u> .				
CodeBuild	CodeBuild 파라미터에 관련된 추가 설명 및 예제를 보려면 <u>AWS CodeBuild 빌드 및</u> <u>테스트 작업 참조</u> 단원을 참조하세요.				
CodeDeploy	CodeDeploy 파라미터에 관련된 추가 설명 및 예제를 보려면 <u>AWS CodeDeploy 배</u> <u>포 작업 참조</u> 단원을 참조하세요.				
AWS Device Farm	AWS Device Farm 파라미터와 관련된 자세한 설명 및 예제는 섹션을 참조하세 요 <u>AWS Device Farm 테스트 작업 참조</u> .				
AWS Elastic Beanstalk	ElasticBe anstalk	ApplicationName	필수		
		EnvironmentName	필수		
AWS Lambda	AWS Lambda 파라미터와 관련된 예제를 포함한 자세한 내용은 단원을 참조하십시 오 <u>AWS Lambda 작업 참조 호출</u> .				
AWS OpsWorks Stacks	OpsWorks	Stack	필수		
		Layer	선택 사항		
		Арр	필수		
Amazon ECS	Amazon ECS 파라미터에 관련된 추가 설명 및 예제를 보려면 <u>Amazon Elastic</u> <u>Container Service(Amazon ECS) 배포 작업 참조</u> 단원을 참조하세요.				
Amazon ECS 및 CodeDeplo y(블루/그린)	Amazon ECS 및 CodeDeploy 블루/그린 파라미터에 관련된 추가 설명 및 예제를 보 려면 <u>Amazon Elastic Container Service(Amazon ECS) 및 CodeDeploy 블루-그린</u> <u>배포 작업 참조</u> 단원을 참조하세요.				
Service Catalog	ServiceCatalog	TemplateFilePath	필수		

공급자 이름	작업 유형의 공급자 이름	구성 속성	필수/선택
		ProductVersionName	필수
		ProductType	필수
		ProductVersionDescription	선택 사항
		ProductId	필수
Alexa Skills Kit	AlexaSkillsKit	ClientId	필수
		ClientSecret	필수
		RefreshToken	필수
		SkillId	필수
Jenkins	Jenkins에 대해 CodePipeline 플러그 인에 제공한 작업 이 름(예: <i>MyJenkins</i> <i>ProviderName</i>)	ProjectName	필수
수동 승인	Manual	CustomData	선택 사항
		ExternalEntityLink	선택 사항
		NotificationArn	선택 사항

다음 예는 Alexa Skills Kit를 사용하는 배포 작업에 유효한 구성을 보여줍니다.

```
"configuration": {
   "ClientId": "amzn1.application-oa2-client.aadEXAMPLE",
   "ClientSecret": "****",
   "RefreshToken": "****",
   "SkillId": "amzn1.ask.skill.22649d8f-0451-4b4b-9ed9-bfb6cEXAMPLE"
}
```

다음 예제는 수동 승인에 유효한 구성을 보여줍니다.

```
"configuration": {
    "CustomData": "Comments on the manual approval",
    "ExternalEntityLink": "http://my-url.com",
    "NotificationArn": "arn:aws:sns:us-west-2:12345EXAMPLE:Notification"
}
```

작업 구조 참조

이 단원은 작업 구성에 대한 참조일 뿐입니다. 파이프라인 구조의 개념적 개요는 <u>CodePipeline 파이프</u> 라인 구조 참조 단원을 참조하십시오.

CodePipeline의 각 작업 공급자는 파이프라인 구조에서 필수 및 옵션 구성 필드 세트를 사용합니다. 이 단원에는 다음과 같은 작업 공급자별 참조가 나와 있습니다.

- 파이프라인 구조 작업 블록에 포함된 ActionType 필드의 유효한 값(예: Owner 및 Provider)
- 파이프라인 구조 작업 섹션에 포함된 Configuration 파라미터(필수 및 옵션)에 대한 설명 및 기타 참조 정보
- 유효한 예제 JSON 및 YAML 작업 필드

이 단원은 추가 작업 공급자를 포함하도록 주기적으로 업데이트됩니다. 참조 정보는 현재 다음과 같은 작업 공급자에 대해 사용할 수 있습니다.

주제

- Amazon EC2 작업 참조
- Amazon ECR 소스 작업 참조
- ECRBuildAndPublish 빌드 작업 참조
- Amazon Elastic Container Service(Amazon ECS) 및 CodeDeploy 블루-그린 배포 작업 참조
- Amazon Elastic Container Service(Amazon ECS) 배포 작업 참조
- Amazon Elastic Kubernetes Service EKS 배포 작업 참조
- Amazon S3 배포 작업 참조
- Amazon S3 소스 작업 참조
- AWS AppConfig 배포 작업 참조
- AWS CloudFormation 배포 작업 참조
- AWS CloudFormation StackSets 배포 작업 참조
- AWS CodeBuild 빌드 및 테스트 작업 참조
- AWS CodePipeline 작업 참조 호출
- <u>CodeCommit 소스 작업 참조</u>
- AWS CodeDeploy 배포 작업 참조
- <u>Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com 및 GitLab 자체 관리형 작업용</u> CodeStarSourceConnection
- 명령 작업 참조
- AWS Device Farm 테스트 작업 참조
- Elastic Beanstalk 배포 작업 참조
- Amazon Inspector InspectorScan 호출 작업 참조
- AWS Lambda 작업 참조 호출
- AWS OpsWorks 배포 작업 참조
- AWS Service Catalog 배포 작업 참조
- AWS Step Functions 작업 참조 호출

Amazon EC2 작업 참조

Amazon EC2 EC2 작업을 사용하여 배포 플릿에 애플리케이션 코드를 배포합니다. 배포 플릿은 Amazon EC2 Linux 인스턴스 또는 Linux SSM 관리형 노드로 구성될 수 있습니다. 인스턴스에 SSM 에 이전트가 설치되어 있어야 합니다.

Note

이 작업은 Linux 인스턴스 유형만 지원합니다. 지원되는 최대 플릿 크기는 인스턴스 500개입니다.

작업은 지정된 최대값을 기반으로 인스턴스 수를 선택합니다. 이전 인스턴스에서 실패한 인스턴스가 먼저 선택됩니다. 이전에 작업이 실패한 경우와 같이 인스턴스가 이미 동일한 입력 아티팩트의 배포를 수신한 경우, 작업은 특정 인스턴스에서 배포를 건너뜁니다.

Note

이 작업은 V2 유형 파이프라인에서만 지원됩니다.

주제

- <u>작업 유형</u>
- 구성 파라미터

- 입력 아티팩트
- 출력 아티팩트
- EC2 배포 작업에 대한 서비스 역할 정책 권한
- <u>작업 선언</u>
- 다음 사항도 참조하세요.

작업 유형

- 범주: Deploy
- 소유자: AWS
- 공급자: EC2
- 버전:1

구성 파라미터

InstanceTagKey

필수 여부: 예

와 같이 Amazon EC2에서 생성한 인스턴스의 태그 키입니다Name.

InstanceTagValue

필수 여부: 예

와 같이 Amazon EC2에서 생성한 인스턴스의 태그 값입니다my-instances.

InstanceType

필수 여부: 예

Amazon EC2에서 생성된 인스턴스 또는 SSM 노드의 유형입니다. 유효 값은 EC2 및 SSM_MANAGED_NODE입니다.

모든 인스턴스에 SSM 에이전트를 이미 생성, 태그 지정 및 설치했어야 합니다.

Note

인스턴스를 생성할 때 기존 EC2 인스턴스 역할을 생성하거나 사용합니다. Access Denied 오류를 방지하려면 인스턴스 역할에 S3 버킷 권한을 추가하여 CodePipeline 아티 팩트 버킷에 인스턴스 권한을 부여해야 합니다. 파이프라인 리전의 아티팩트 버킷으로 범 위가 축소된 s3:Get0bject 권한으로 기본 역할을 생성하거나 기존 역할을 업데이트합니 다.

TargetDirectory

필수 여부: 예

Amazon EC2 인스턴스에서 스크립트를 실행하는 데 사용할 디렉터리입니다.

MaxBatch

필수 여부: 아니요

병렬로 배포할 수 있는 최대 인스턴스 수입니다.

MaxError

필수 여부: 아니요

배포 중에 허용되는 최대 인스턴스 오류 수입니다.

TargetGroupNameList

필수 여부: 아니요

배포를 위한 대상 그룹 이름 목록입니다. 대상 그룹을 이미 생성했어야 합니다.

대상 그룹은 특정 요청을 처리하기 위한 인스턴스 세트를 제공합니다. 대상 그룹을 지정하면 배포 전에 대상 그룹에서 인스턴스가 제거되고 배포 후에 대상 그룹에 다시 추가됩니다.

PreScript

필수 여부: 아니요

작업 배포 단계 전에 실행할 스크립트입니다.

PostScript

필수 여부: 예

작업 배포 단계 후에 실행할 스크립트입니다.

다음 이미지는 작업에 대한 편집 페이지의 예를 보여줍니다.

×

Instance type

Choose the instance type that you want to deploy to. Supported types are Amazon EC2 instances or AWS Systems Manager (SSM) managed nodes. You must have already created, tagged, and installed the SSM agent on all instances.

T

EC2

You can add one group of tags for EC2 instances to this deployment group.

One tag group: Any instance identified by the tag group will be deployed to.

Кеу		
ſ	Q Name	×

Value

...

Q my-instances

Matching instances

2 unique matched instances.

Click here for details []

Target directory

Specify the location of the target directory you want to deploy to. Use an absolute path like /home/ec2-user/deploy.

/home/ec2-user/testhelloworld

PreScript - optional

Path to the executable script file that runs BEFORE the Deploy phase. It should start from the root directory of your uploaded source artifact. Use an absolute path like uploadDir/preScript.sh.

PostScript

Path to the executable script file that runs AFTER the Deploy phase. It should start from the root directory of your uploaded source artifact. Use an absolute path like uploadDir/postScript.sh.

test/script.sh

Advanced

Max batch - optional

Specify the number or percentage of targets that can deploy in parallel.

targets

○ percentage

नुष्तुम्ब्रह्महाद्वाराणा 1 to the number of instances you have

입력 아티팩트

- 아티팩트 수: 1
- 설명: 배포 중에 스크립트 작업을 지원하기 위해 제공된 파일입니다.

출력 아티팩트

- 아티팩트 수: 0
- 설명: 이 작업 유형에는 출력 아티팩트가 적용되지 않습니다.

EC2 배포 작업에 대한 서비스 역할 정책 권한

CodePipeline이 작업을 실행할 때 CodePipeline 서비스 역할에는 최소 권한으로 액세스할 수 있도록 범위가 적절하게 축소된 다음 권한이 필요합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "StatementWithAllResource",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeInstances",
                "elasticloadbalancing:DescribeTargetGroupAttributes",
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeTargetHealth",
                "ssm:CancelCommand",
                "ssm:DescribeInstanceInformation",
                "ssm:ListCommandInvocations"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "StatementForLogs",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
```

```
"logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:{{region}}:{{AccountId}}:log-group:/aws/codepipeline/
{{pipelineName}}:*"
            ]
        },
        {
            "Sid": "StatementForElasticloadbalancing",
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:DeregisterTargets",
                "elasticloadbalancing:RegisterTargets"
            ],
            "Resource": [
                "arn:aws:elasticloadbalancing:{{region}}:{{AccountId}}:targetgroup/
[[targetGroupName]]/*"
            1
        },
        {
            "Sid": "StatementForSsmOnTaggedInstances",
            "Effect": "Allow",
            "Action": [
                "ssm:SendCommand"
            ],
            "Resource": [
                "arn:aws:ec2:{{region}}:{{AccountId}}:instance/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/{{tagKey}}": "{{tagValue}}"
                }
            }
        },
        {
            "Sid": "StatementForSsmApprovedDocuments",
            "Effect": "Allow",
            "Action": [
                "ssm:SendCommand"
            ],
            "Resource": [
                "arn:aws:ssm:{{region}}::document/AWS-RunPowerShellScript",
                "arn:aws:ssm:{{region}}::document/AWS-RunShellScript"
```

]
		}	
]		
}			

CloudWatch 로그의 파이프라인에 대한 로그 그룹

CodePipeline이 작업을 실행하면 CodePipeline은 다음과 같이 파이프라인 이름을 사용하여 로그 그룹 을 생성합니다. 이렇게 하면 파이프라인 이름을 사용하여 리소스를 로깅할 수 있는 권한을 범위 아래로 지정할 수 있습니다.

/aws/codepipeline/MyPipelineName

로깅에 대한 다음 권한은 서비스 역할에 대한 위 업데이트에 포함되어 있습니다.

- logs:CreateLogGroup
- logs:CreateLogStream
- logs:PutLogEvents

작업 세부 정보 대화 상자를 사용하여 콘솔에서 로그를 보려면 로그를 볼 수 있는 권한을 콘솔 역할에 추가해야 합니다. 자세한 내용은 <u>CodePipeline 콘솔에서 컴퓨팅 로그를 보는 데 필요한 권한</u>에서 콘솔 권한 정책 예제를 참조하세요.

CloudWatch 로그에 대한 서비스 역할 정책 권한

CodePipeline이 작업을 실행하면 CodePipeline은 다음과 같이 파이프라인 이름을 사용하여 로그 그룹 을 생성합니다. 이렇게 하면 파이프라인 이름을 사용하여 리소스를 로깅할 수 있는 권한을 범위 아래로 지정할 수 있습니다.

/aws/codepipeline/MyPipelineName

작업 세부 정보 대화 상자를 사용하여 콘솔에서 로그를 보려면 로그를 볼 수 있는 권한을 콘솔 역할에 추가해야 합니다. 자세한 내용은 <u>CodePipeline 콘솔에서 컴퓨팅 로그를 보는 데 필요한 권한</u>에서 콘솔 권한 정책 예제를 참조하세요.

사용자 가이드

작업 선언

YAML

```
name: DeployEC2
actions:
- name: EC2
  actionTypeId:
    category: Deploy
    owner: AWS
    provider: EC2
    version: '1'
 runOrder: 1
  configuration:
    InstanceTagKey: Name
    InstanceTagValue: my-instances
    InstanceType: EC2
    PostScript: "test/script.sh",
    TargetDirectory: "/home/ec2-user/deploy"
  outputArtifacts: []
  inputArtifacts:
  - name: SourceArtifact
  region: us-east-1
```

JSON

```
{
    "name": "DeployEC2",
    "actions": [
        {
            "name": "EC2Deploy",
            "actionTypeId": {
                "category": "Deploy",
                "owner": "AWS",
                "provider": "EC2",
                "version": "1"
            },
            "runOrder": 1,
            "configuration": {
                "InstanceTagKey": "Name",
                "InstanceTagValue": "my-instances",
                "InstanceType": "EC2",
                "PostScript": "test/script.sh",
```

다음 사항도 참조하세요.

이 작업을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

- <u>자습서: CodePipeline을 사용하여 Amazon EC2 인스턴스에 배포</u> -이 자습서에서는 EC2 작업을 사용하여 파이프라인을 생성하는 것과 함께 스크립트 파일을 배포할 EC2 인스턴스를 생성하는 방법을 안내합니다.
- <u>오류 메시지와 함께 EC2 배포 작업이 실패함 No such file</u> -이 주제에서는 EC2 작업에서 찾을 수 없는 파일 오류에 대한 문제 해결에 대해 설명합니다.

Amazon ECR 소스 작업 참조

새 이미지가 Amazon ECR 리포지토리로 푸시될 때 파이프라인을 트리거합니다. 이 작업은 Amazon ECR에 푸시된 이미지의 URI를 참조하는 이미지 정의 파일을 제공합니다. 이 소스 작업은 다른 모든 소 스 아티팩트에 대한 소스 위치를 허용하기 위해 CodeCommit 같은 다른 소스 작업과 함께 사용되는 경 우가 많습니다. 자세한 내용은 <u>자습서: Amazon ECR 소스 및 ECS-to-CodeDeploy 배포를 사용하여 파</u> 이프라인 생성 단원을 참조하십시오.

콘솔을 사용하여 파이프라인을 생성하거나 편집할 때 CodePipeline은 리포지토리에서 변경이 발생할 때 파이프라인을 시작하는 EventBridge 규칙을 생성합니다.

Note

Amazon ECR, Amazon S3 또는 CodeCommit 소스의 경우 입력 변환 항목을 사용하여 파이프 라인 이벤트에 EventBridgerevisionValue의를 사용하는 소스 재정의를 생성할 수도 있습니 다. 여기서 revisionValue는 객체 키, 커밋 또는 이미지 ID에 대한 소스 이벤트 변수에서 파 생됩니다. 자세한 내용은 , <u>Amazon ECR 소스 작업 및 EventBridge 리소스</u> <u>이벤트에 대해 활성</u> <u>화된 소스를 사용하여 Amazon S3 소스 작업에 연결</u>또는의 절차에 포함된 입력 변환 항목의 선 택적 단계를 참조하세요CodeCommit 소스 작업 및 EventBridge.

Amazon ECR 작업을 통해 파이프라인을 연결하기 전에 이미 Amazon ECR 리포지토리를 생성하고 이 미지를 푸시해야 합니다.

주제

- <u>작업 유형</u>
- <u>구성 파라미터</u>
- 입력 아티팩트
- 출력 아티팩트
- <u>출력 변수</u>
- <u>서비스 역할 권한: Amazon ECR 작업</u>
- <u>작업 선언(Amazon ECR 예제)</u>
- 다음 사항도 참조하세요.

작업 유형

- 범주: Source
- 소유자: AWS
- 공급자: ECR
- 버전:1

구성 파라미터

RepositoryName

필수 여부: 예

이미지가 푸시된 Amazon ECR 리포지토리의 이름입니다.

ImageTag

필수 여부: 아니요

이미지에 사용되는 태그입니다.

Note

ImageTag의 값을 지정하지 않는 경우, 기본값은 latest입니다.

입력 아티팩트

- 아티팩트 수: 0
- 설명: 이 작업 유형에는 입력 아티팩트가 적용되지 않습니다.

출력 아티팩트

- 아티팩트 수: 1
- 설명: 이 작업은 파이프라인 실행을 트리거한 이미지의 URI를 포함하는 imageDetail.json 파 일을 포함하는 아티팩트를 생성합니다. imageDetail.json 파일에 대한 자세한 내용은 <u>Amazon</u> ECS 블루/그린 배포 작업을 위한 imageDetail.json 파일 단원을 참조하십시오.

출력 변수

이 작업을 구성하면 파이프라인에서 다운스트림 작업의 작업 구성에서 참조할 수 있는 변수가 생성됩 니다. 이 작업은 작업에 네임스페이스가 없는 경우에도 출력 변수로 볼 수 있는 변수를 생성합니다. 이 러한 변수를 다운스트림 작업 구성에서 사용할 수 있도록 네임스페이스를 사용하여 작업을 구성합니 다.

자세한 내용은 변수 참조 단원을 참조하십시오.

RegistryId

리포지토리가 포함된 레지스트리와 연결된 AWS 계정 ID입니다.

RepositoryName

이미지가 푸시된 Amazon ECR 리포지토리의 이름입니다.

ImageTag

이미지에 사용되는 태그입니다.

ImageDigest

이미지 매니페스트의 sha256 다이제스트입니다.

imageURI

```
이미지의 URI입니다.
```

서비스 역할 권한: Amazon ECR 작업

Amazon ECR 지원의 경우 다음을 정책 설명에 추가합니다.

```
{
    "Effect": "Allow",
    "Action": [
        "ecr:DescribeImages"
    ],
    "Resource": "resource_ARN"
},
```

이 작업에 대한 자세한 내용은 섹션을 참조하세요<u>Amazon ECR 소스 작업 참조</u>.

작업 선언(Amazon ECR 예제)

YAML

```
Name: Source
Actions:
  - InputArtifacts: []
  ActionTypeId:
    Version: '1'
    Owner: AWS
    Category: Source
    Provider: ECR
    OutputArtifacts:
        - Name: SourceArtifact
    RunOrder: 1
    Configuration:
        ImageTag: latest
        RepositoryName: my-image-repo
    Name: ImageSource
```

JSON

```
{
    "Name": "Source",
    "Actions": [
        {
            "InputArtifacts": [],
            "ActionTypeId": {
                "Version": "1",
                 "Owner": "AWS",
                "Category": "Source",
                "Provider": "ECR"
            },
            "OutputArtifacts": [
                {
                     "Name": "SourceArtifact"
                }
            ],
            "RunOrder": 1,
            "Configuration": {
                "ImageTag": "latest",
                 "RepositoryName": "my-image-repo"
            },
            "Name": "ImageSource"
        }
    ]
},
```

다음 사항도 참조하세요.

이 작업을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

• <u>자습서: Amazon ECR 소스 및 ECS-to-CodeDeploy 배포를 사용하여 파이프라인 생성</u> - 이 자습서에 서는 Amazon ECS 인스턴스에 배포하는 CodeCommit 및 Amazon ECR 소스가 있는 파이프라인을 생성할 수 있는 샘플 앱 사양 파일과 샘플 CodeDeploy 애플리케이션 및 배포 그룹을 제공합니다.

ECRBuildAndPublish 빌드 작업 참조

이 빌드 작업을 사용하면 소스에서 변경 사항이 발생할 때 새 이미지 빌드 및 푸시를 자동화할 수 있 습니다. 이 작업은 지정된 Docker 파일 위치를 기반으로 빌드되고 이미지를 푸시합니다. 이 빌드 작 업은 Amazon ECR 소스 리포지토리에서 변경이 발생할 때 파이프라인을 트리거하는 CodePipeline 의 Amazon ECR 소스 작업과 동일하지 않습니다. 해당 작업에 대한 자세한 내용은 섹션을 참조하세 요Amazon ECR 소스 작업 참조.

이는 파이프라인을 트리거하는 소스 작업이 아닙니다. 이 작업은 이미지를 빌드하고 Amazon ECR 이 미지 리포지토리로 푸시합니다.

파이프라인에 작업을 추가하기 전에 Amazon ECR 리포지토리를 이미 생성하고 GitHub와 같은 소스 코드 리포지토리에 Dockerfile을 추가해야 합니다.

A Important

이 작업은 CodePipeline 관리형 CodeBuild 컴퓨팅을 사용하여 빌드 환경에서 명령을 실행합니 다. 명령 작업을 실행하면 AWS CodeBuild에서 별도의 요금이 발생합니다.

Note

이 작업은 V2 유형 파이프라인에서만 사용할 수 있습니다.

주제

- <u>작업 유형</u>
- <u>구성 파라미터</u>
- 입력 아티팩트
- 출력 아티팩트
- <u>출력 변수</u>
- <u>서비스 역할 권한: ECRBuildAndPublish 작업</u>
- <u>작업 선언</u>
- <u>다음 사항도 참조하세요.</u>

작업 유형

- 범주: Build
- 소유자: AWS

- 공급자: ECRBuildAndPublish
- 버전:1

구성 파라미터

ECRRepositoryName

필수 여부: 예

이미지가 푸시되는 Amazon ECR 리포지토리의 이름입니다.

DockerFilePath

필수 여부: 아니요

이미지를 빌드하는 데 사용되는 Docker 파일의 위치입니다. 선택적으로 루트 수준이 아닌 경우 대 체 Docker 파일 위치를 제공할 수 있습니다.

Note

의 값을 지정DockerFilePath하지 않으면 값이 기본적으로 소스 리포지토리 루트 수준으 로 설정됩니다.

ImageTags

필수 여부: 아니요

이미지에 사용되는 태그입니다. 여러 태그를 쉼표로 구분된 문자열 목록으로 입력할 수 있습니다.

Note

ImageTags의 값을 지정하지 않는 경우, 기본값은 latest입니다.

RegistryType

필수 여부: 아니요

리포지토리가 퍼블릭인지 프라이빗인지 지정합니다. 유효한 값은 private | public입니다.

Note

RegistryType의 값을 지정하지 않는 경우, 기본값은 private입니다.

입력 아티팩트

- 아티팩트 수: 1
- 설명: 이미지를 빌드하는 데 필요한 Dockerfile이 포함된 소스 작업에서 생성된 아티팩트입니다.

출력 아티팩트

• 아티팩트 수: 0

출력 변수

이 작업을 구성하면 파이프라인에서 다운스트림 작업의 작업 구성에서 참조할 수 있는 변수가 생성됩 니다. 이 작업은 작업에 네임스페이스가 없는 경우에도 출력 변수로 볼 수 있는 변수를 생성합니다. 이 러한 변수를 다운스트림 작업 구성에서 사용할 수 있도록 네임스페이스를 사용하여 작업을 구성합니 다.

자세한 내용은 변수 참조 단원을 참조하십시오.

ECRImageDigestId

```
이미지 매니페스트의 sha256 다이제스트입니다.
```

ECRRepositoryName

```
이미지가 푸시된 Amazon ECR 리포지토리의 이름입니다.
```

서비스 역할 권한: ECRBuildAndPublish 작업

ECRBuildAndPublish 작업 지원을 위해 정책 설명에 다음을 추가합니다.

```
"Effect": "Allow",
        "Action": [
            "ecr:DescribeRepositories",
            "ecr:GetAuthorizationToken",
            "ecr-public:DescribeRepositories",
            "ecr-public:GetAuthorizationToken"
        ],
    "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "ecr:GetAuthorizationToken",
            "ecr:InitiateLayerUpload",
            "ecr:UploadLayerPart",
            "ecr:CompleteLayerUpload",
            "ecr:PutImage",
            "ecr:GetDownloadUrlForLayer",
            "ecr:BatchCheckLayerAvailability"
        ],
        "Resource": "PrivateECR_Resource_ARN"
    },
    {
        "Effect": "Allow",
        "Action": [
            "ecr-public:GetAuthorizationToken",
            "ecr-public:DescribeRepositories",
            "ecr-public:InitiateLayerUpload",
            "ecr-public:UploadLayerPart",
            "ecr-public:CompleteLayerUpload",
            "ecr-public:PutImage",
            "ecr-public:BatchCheckLayerAvailability",
            "sts:GetServiceBearerToken"
        ],
        "Resource": "PublicECR_Resource_ARN"
    },
    {
        "Effect": "Allow",
        "Action": [
            "sts:GetServiceBearerToken"
        ],
        "Resource": "*"
    }
]
```

}

또한 Commands 작업에 아직 추가되지 않은 경우 CloudWatch 로그를 보려면 서비스 역할에 다음 권한 을 추가합니다.

```
{
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
    ],
    "Resource": "resource_ARN"
},
```

Note

서비스 역할 정책 문의 리소스 기반 권한을 사용하여 파이프라인 리소스 수준으로 권한 범위를 좁힙니다.

이 작업에 대한 자세한 내용은 섹션을 참조하세요ECRBuildAndPublish 빌드 작업 참조.

작업 선언

YAML

```
name: ECRBuild
actionTypeId:
   category: Build
   owner: AWS
   provider: ECRBuildAndPublish
   version: '1'
runOrder: 1
configuration:
   ECRRepositoryName: actions/my-imagerepo
outputArtifacts: []
inputArtifacts:
   name: SourceArtifact
region: us-east-1
namespace: BuildVariables
```

JSON

```
{
    "name": "ECRBuild",
    "actionTypeId": {
        "category": "Build",
        "owner": "AWS",
        "provider": "ECRBuildAndPublish",
        "version": "1"
    },
    "runOrder": 1,
    "configuration": {
        "ECRRepositoryName": "actions/my-imagerepo"
    },
    "outputArtifacts": [],
    "inputArtifacts": [
        {
            "name": "SourceArtifact"
        }
    ],
    "region": "us-east-1",
    "namespace": "BuildVariables"
},
```

다음 사항도 참조하세요.

이 작업을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

• <u>자습서: CodePipeline을 사용하여 Docker 이미지를 빌드하고 Amazon ECR에 푸시(V2 유형)</u> -이 자 습서에서는 샘플 Dockerfile과 소스 리포지토리 변경 시 이미지를 ECR에 푸시한 다음 Amazon ECS 에 배포하는 파이프라인을 생성하는 지침을 제공합니다.

Amazon Elastic Container Service(Amazon ECS) 및 CodeDeploy 블루-그린 배포 작업 참조

블루/그린 배포를 사용하여 컨테이너 애플리케이션을 배포 AWS CodePipeline 하는 파이프라인을에서 구성할 수 있습니다. 블루/그린 배포에서는 트래픽을 다시 라우팅하기 전에 이전 버전과 함께 새 버전 의 애플리케이션을 시작하고 새 버전을 테스트할 수 있습니다. 또한 배포 프로세스를 모니터링하고 문 제가 발생할 경우 신속하게 롤백할 수 있습니다. 완료된 파이프라인은 이미지 또는 작업 정의 파일의 변경 사항을 감지하고 CodeDeploy를 사용하여 트 래픽을 Amazon ECS 클러스터 및 로드 밸런서로 라우팅하고 배포합니다. CodeDeploy는 로드 밸런서 에 새 리스너를 생성하여 특수 포트를 통해 새 작업을 대상으로 지정할 수 있습니다. Amazon ECS 작 업 정의가 저장되는 CodeCommit 리포지토리와 같은 소스 위치를 사용하도록 파이프라인을 구성할 수 도 있습니다.

파이프라인을 생성하기 전에 Amazon ECS 리소스, CodeDeploy 리소스, 로드 밸런서 및 대상 그룹을 이미 생성해야 합니다. 이미지 리포지토리에 이미 이미지에 태그를 지정하여 저장하고 작업 정의와 AppSpec 파일을 파일 리포지토리에 업로드했어야 합니다.

1 Note

이 주제에서는 CodePipeline인의 Amazon ECS에서 CodeDeploy로의 블루/그린 배포 작업 에 대해 설명합니다. CodePipeline에서의 Amazon ECS 표준 배포 작업에 대한 참조 정보는 Amazon Elastic Container Service(Amazon ECS) 배포 작업 참조을 참조하세요.

주제

- <u>작업 유형</u>
- <u>구성 파라미터</u>
- <u>입력 아티팩트</u>
- 출력 아티팩트
- 서비스 역할 권한: CodeDeployToECS 작업
- <u>작업 선언</u>
- 다음 사항도 참조하세요.

작업 유형

- 범주: Deploy
- 소유자: AWS
- 공급자: CodeDeployToECS
- 버전:1

구성 파라미터

ApplicationName

필수 여부: 예

CodeDeploy의 애플리케이션 이름. 파이프라인을 생성하기 전에 CodeDeploy에 애플리케이션이 이 미 생성되어 있어야 합니다.

DeploymentGroupName

필수 여부: 예

CodeDeploy 애플리케이션용으로 생성한 Amazon ECS 작업 세트에 대해 지정된 배포 그룹입니다. 파이프라인을 생성하기 전에 CodeDeploy에 배포 그룹이 이미 생성되어 있어야 합니다.

TaskDefinitionTemplateArtifact

필수 여부: 예

배포 작업에 작업 정의 파일을 제공하는 입력 아티팩트의 이름입니다. 일반적으로 소스 작업 의 출력 아티팩트 이름입니다. 콘솔을 사용하는 경우 소스 작업 출력 아티팩트의 기본 이름은 SourceArtifact입니다.

AppSpecTemplateArtifact

필수 여부: 예

배포 작업에 AppSpec 파일을 제공하는 입력 아티팩트의 이름입니다. 파이프라인이 실행되면 이 값이 업데이트됩니다. 일반적으로 소스 작업의 출력 아티팩트 이름입니다. 콘솔을 사용하 는 경우 소스 작업 출력 아티팩트의 기본 이름은 SourceArtifact입니다. AppSpec 파일에서 TaskDefinition의 경우 <u>여기</u>에 나온 것처럼 <TASK_DEFINITION> 자리표시자 텍스트를 유지 할 수 있습니다.

AppSpecTemplatePath

필수 여부: 아니요

파이프라인의 CodeCommit 리포지토리와 같은 파이프라인 소스 파일 위치에 저장된 AppSpec 파 일의 파일 이름입니다. 기본적인 파일 이름은 appspec.yaml입니다. AppSpec 파일의 이름이 같 고 파일 리포지토리의 루트 수준에 저장되어 있는 경우 파일 이름을 제공할 필요가 없습니다. 경로 가 기본 경로가 아닌 경우 경로와 파일 이름을 입력합니다. TaskDefinitionTemplatePath

필수 여부: 아니요

파이프라인의 CodeCommit 리포지토리와 같은 파이프라인 파일 소스 위치에 저장된 작업 정의의 파일 이름입니다. 기본적인 파일 이름은 taskdef.json입니다. 작업 정의 파일의 이름이 같고 파 일 리포지토리의 루트 수준에 저장되어 있는 경우 파일 이름을 제공할 필요가 없습니다. 경로가 기 본 경로가 아닌 경우 경로와 파일 이름을 입력합니다.

Image<Number>ArtifactName

필수 여부: 아니요

배포 작업에 이미지를 제공하는 입력 아티팩트의 이름입니다. 이는 일반적으로 이미지 리포지토리 의 출력 아티팩트(예: Amazon ECR 소스 작업의 출력)입니다.

<Number>의 사용 가능한 값은 1~4입니다.

Image<Number>ContainerName

필수 여부: 아니요

이미지 리포지토리(예: Amazon ECR 소스 리포지토리)에서 사용할 수 있는 이미지의 이름입니다.

<Number>의 사용 가능한 값은 1~4입니다.

입력 아티팩트

- 아티팩트 수:1 to 5
- 설명: CodeDeployToECS 작업은 먼저 소스 파일 리포지토리에서 작업 정의 파일과 AppSpec 파일 을 찾은 다음 이미지 리포지토리에서 이미지를 찾은 다음 동적으로 작업 정의의 새 버전을 생성하고 마지막으로 AppSpec 명령을 실행하여 작업 세트와 컨테이너를 클러스터에 배포합니다.

CodeDeployToECS 작업은 이미지 URI를 이미지에 매핑하는 imageDetail.json 파일을 찾 습니다. Amazon ECR 이미지 리포지토리에 변경을 커밋하면 파이프라인 ECR 소스 작업이 해 당 커밋에 대한 imageDetail.json 파일을 생성합니다. 작업이 자동화되지 않은 파이프라인용 imageDetail.json 파일을 수동으로 추가할 수도 있습니다.imageDetail.json 파일에 대한 자 세한 내용은 <u>Amazon ECS 블루/그린 배포 작업을 위한 imageDetail.json 파일</u> 단원을 참조하십시오.

CodeDeployToECS 작업을 수행하면 작업 정의의 새 개정이 동적으로 생성됩니다. 이 단계에 서 이 작업은 작업 정의 파일의 자리 표시자를 imageDetail.json 파일에서 검색된 이미지 URI로 바꿉니다. 예를 들어 IMAGE1_NAME을 Image1ContainerName 파라미터로 설정하는 경우 자 리 표시자 <IMAGE1_NAME>을 작업 정의 파일의 이미지 필드 값으로 지정해야 합니다. 이 경우 CodeDeploytoECS 작업은 Image1ArtifactName으로 지정한 아티팩트의 imageDetail.json에서 검색 된 실제 이미지 URI로 자리 표시자 <IMAGE1_NAME>을 대체합니다.

작업 정의 업데이트의 경우 CodeDeploy AppSpec.yaml 파일에 TaskDefinition속성이 포함되어 있습니다.

TaskDefinition: <TASK_DEFINITION>

이 속성은 새 작업 정의가 생성된 후 CodeDeployToECS 작업에 의해 업데이트됩니다.

TaskDefinition 필드 값의 경우 자리 표시자 텍스트는 <TASK_DEFINITION>이어야 합니다. CodeDeployToECS 작업은 이 자리 표시자를 동적으로 생성된 작업 정의의 실제 ARN으로 대체합 니다.

출력 아티팩트

- 아티팩트 수: 0
- 설명: 이 작업 유형에는 출력 아티팩트가 적용되지 않습니다.

서비스 역할 권한: CodeDeployToECS 작업

CodeDeployToECS 작업(블루/그린 배포)의 경우, CodeDeploy를 사용하여 Amazon ECS 블루/그린 배포 작업을 위한 파이프라인을 만드는 데 필요한 최소 권한은 다음과 같습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCodeDeployDeploymentActions",
            "Action": [
                "codedeploy:CreateDeployment",
                "codedeploy:GetDeployment"
            ],
            "Resource": [
                "arn:aws:codedeploy:*:{{customerAccountId}}:deploymentgroup:
[[ApplicationName]]/*"
            ],
            "Effect": "Allow"
```

```
},
{
  "Sid": "AllowCodeDeployApplicationActions",
  "Action": [
    "codedeploy:GetApplication",
    "codedeploy:GetApplicationRevision",
    "codedeploy:RegisterApplicationRevision"
  ],
  "Resource": [
    "arn:aws:codedeploy:*:{{customerAccountId}}:application:[[ApplicationName]]",
    "arn:aws:codedeploy:*:{{customerAccountId}}:application:[[ApplicationName]]/*"
  ],
  "Effect": "Allow"
},
{
  "Sid": "AllowCodeDeployDeploymentConfigAccess",
  "Action": [
    "codedeploy:GetDeploymentConfig"
  ],
  "Resource": [
    "arn:aws:codedeploy:*:{{customerAccountId}}:deploymentconfig:*"
  ],
  "Effect": "Allow"
},
{
  "Sid": "AllowECSRegisterTaskDefinition",
  "Action": [
    "ecs:RegisterTaskDefinition"
  ],
  "Resource": [
    "*"
  ],
  "Effect": "Allow"
},
{
  "Sid": "AllowPassRoleToECS",
  "Effect": "Allow",
  "Action": "iam:PassRole",
  "Resource": [
    "arn:aws:iam::{{customerAccountId}}:role/[[PassRoles]]"
  ],
  "Condition": {
    "StringEquals": {
      "iam:PassedToService": [
```

```
"ecs.amazonaws.com",
    "ecs-tasks.amazonaws.com"
    ]
    }
    }
    }
}
```

Amazon ECS에서 태그 지정 권한 부여 사용으로 옵트인할 수 있습니다. 옵트인을 통해 다음 권한: ecs:TagResource를 부여해야 합니다. 옵트인하는 방법과 권한 필요 여부 및 태그 권한 부여 적용 여 부에 대한 자세한 내용은 Amazon Elastic Container Service 개발자 안내서의 <u>태그 지정 권한 부여 타</u> <u>임라인</u>을 참조하세요.

또한 작업에 IAM 역할을 사용하려면 iam:PassRole 권한을 추가해야 합니다. 자세한 내용은 <u>Amazon</u> ECS 태스크 실행 IAM 역할과 태스크에 대한 IAM 역할을 참조하세요.

위 예제ecs-tasks.amazonaws.com와 같이 iam:PassedToService 조건의 서비스 목록에를 추 가할 수도 있습니다.

작업 선언

YAML

```
Name: Deploy
Actions:
  - Name: Deploy
    ActionTypeId:
      Category: Deploy
      Owner: AWS
      Provider: CodeDeployToECS
      Version: '1'
    RunOrder: 1
    Configuration:
      AppSpecTemplateArtifact: SourceArtifact
      ApplicationName: ecs-cd-application
      DeploymentGroupName: ecs-deployment-group
      Image1ArtifactName: MyImage
      Image1ContainerName: IMAGE1_NAME
      TaskDefinitionTemplatePath: taskdef.json
      AppSpecTemplatePath: appspec.yaml
      TaskDefinitionTemplateArtifact: SourceArtifact
```

```
OutputArtifacts: []
InputArtifacts:
    - Name: SourceArtifact
    - Name: MyImage
Region: us-west-2
Namespace: DeployVariables
```

JSON

```
{
    "Name": "Deploy",
    "Actions": [
        {
            "Name": "Deploy",
            "ActionTypeId": {
                "Category": "Deploy",
                "Owner": "AWS",
                "Provider": "CodeDeployToECS",
                "Version": "1"
            },
            "RunOrder": 1,
            "Configuration": {
                "AppSpecTemplateArtifact": "SourceArtifact",
                "ApplicationName": "ecs-cd-application",
                "DeploymentGroupName": "ecs-deployment-group",
                "Image1ArtifactName": "MyImage",
                "Image1ContainerName": "IMAGE1_NAME",
                "TaskDefinitionTemplatePath": "taskdef.json",
                "AppSpecTemplatePath": "appspec.yaml",
                "TaskDefinitionTemplateArtifact": "SourceArtifact"
            },
            "OutputArtifacts": [],
            "InputArtifacts": [
                {
                    "Name": "SourceArtifact"
                },
                {
                    "Name": "MyImage"
                }
            ],
            "Region": "us-west-2",
            "Namespace": "DeployVariables"
        }
```

]

}

다음 사항도 참조하세요.

이 작업을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

 <u>자습서: Amazon ECR 소스 및 ECS-to-CodeDeploy 배포를 사용하여 파이프라인 생성</u> – 이 자습서 에서는 블루/그린 배포에 필요한 CodeDeploy 및 Amazon ECS 리소스를 생성하는 방법을 안내합 니다. 이 자습서에서는 도커 이미지를 Amazon ECR로 푸시하고 도커 이미지 이름, 컨테이너 이름, Amazon ECS 서비스 이름 및 로드 밸런서 구성을 나열하는 Amazon ECS 작업 정의를 생성하는 방 법을 보여줍니다. 그런 다음 자습서에서는 배포를 위한 AppSpec 파일 및 파이프라인을 생성하는 과 정을 안내합니다.

Note

이 주제 및 자습서에서는 CodePipeline인의 CodeDeploy/ECS 블루/그린 작업에 대해 설명 합니다. CodePipeline의 ECS 표준 작업에 대한 자세한 내용은 <u>자습서: CodePipeline을 사용</u> <u>한 지속적 배포</u>를 참조하세요.

- AWS CodeDeploy 사용 설명서 블루/그린 배포에서 로드 밸런서, 프로덕션 리스너, 대상 그룹 및 Amazon ECS 애플리케이션을 사용하는 방법에 대한 자세한 내용은 <u>자습서: Amazon ECS 서비스</u> <u>배포를</u> 참조하세요. AWS CodeDeploy 사용 설명서의이 참조 정보는 Amazon ECS 및를 사용한 블 루/그린 배포에 대한 개요를 제공합니다 AWS CodeDeploy.
- Amazon Elastic Container Service 개발자 안내서 도커 이미지 및 컨테이너, ECS 서비스 및 클러스 터, ECS 작업 세트 사용에 대한 자세한 내용은 Amazon ECS란 무엇입니까?를 참조하세요.

Amazon Elastic Container Service(Amazon ECS) 배포 작업 참조

Amazon ECS 작업을 사용하여 Amazon ECS 서비스 및 작업 세트를 배포할 수 있습니다. Amazon ECS 서비스는 Amazon ECS 클러스터에 배포되는 컨테이너 애플리케이션입니다. Amazon ECS 클러 스터는 클라우드에서 컨테이너 애플리케이션을 호스팅하는 인스턴스 모음입니다. 배포에는 Amazon ECS에서 생성한 작업 정의와 CodePipeline이 이미지를 배포하는 데 사용하는 이미지 정의 파일이 필 요합니다.

▲ Important

CodePipeline의 Amazon ECS 표준 배포 작업은 Amazon ECS 서비스에서 사용하는 수정 버전 을 기반으로 작업 정의의 자체 수정 버전을 생성합니다. Amazon ECS 서비스를 업데이트하지 않고 작업 정의에 대한 새 수정 사항을 생성하면 배포 작업에서 해당 수정 사항을 무시합니다.

파이프라인을 생성하기 전에 이미 Amazon ECS 리소스를 생성하고 이미지에 태그를 지정하여 이미지 리포지토리에 저장한 다음 BuildSpec 파일을 파일 리포지토리에 업로드해야 합니다.

Note

이 참조 항목에서는 CodePipeline의 Amazon ECS 표준 배포 작업에 대해 설명합니다. CodePipeline에서의 CodeDeploy 블루/그린 배포 작업에 대한 Amazon ECS에 대한 참조 정보 는 <u>Amazon Elastic Container Service(Amazon ECS) 및 CodeDeploy 블루-그린 배포 작업 참</u> <u>조</u>을 참조하세요.

주제

- <u>작업 유형</u>
- <u>구성 파라미터</u>
- 입력 아티팩트
- 출력 아티팩트
- 서비스 역할 권한: Amazon ECS 표준 작업
- 작업 선언
- <u>다음 사항도 참조하세요.</u>

작업 유형

- 범주: Deploy
- 소유자: AWS
- 공급자: ECS
- 버전:1

구성 파라미터

ClusterName

필수 여부: 예

Amazon ECS의 Amazon ECS 클러스터.

ServiceName

필수 여부: 예

Amazon ECS에서 생성한 Amazon ECS 서비스.

FileName

필수 여부: 아니요

서비스의 컨테이너 이름과 이미지 및 태그를 설명하는 JSON 파일, 이미지 정의 파일의 이름입니 다. 이 파일은 ECS 표준 배포에 사용합니다. 자세한 내용은 <u>입력 아티팩트</u> 및 <u>Amazon ECS 표준 배</u> <u>포 작업을 위한 imagedefinitions.json 파일</u> 섹션을 참조하세요.

DeploymentTimeout

필수 여부: 아니요

Amazon ECS 배포 작업 제한 시간(분). 제한 시간은 이 작업에 대한 최대 기본 제한 시간까지 구성 할 수 있습니다. 예시:

"DeploymentTimeout": "15"

입력 아티팩트

- 아티팩트 수: 1
- 설명: 작업은 파이프라인의 소스 파일 리포지토리에서 imagedefinitions.json 파일을 찾습니다. 이미지 정의 문서는 Amazon ECS 컨테이너 이름과 이미지 및 태그를 설명하는 JSON 파일입니다. CodePipeline은 Amazon ECR과 같은 이미지 리포지토리에서 파일을 사용하여 이미지를 검색합니다. 작업이 자동화되지 않은 파이프라인용 imagedefinitions.json 파일을 수동으로 추가할수 있습니다. imagedefinitions.json 파일에 대한 자세한 내용은 Amazon ECS 표준 배포 작업을 위한 imagedefinitions.json 파일 단원을 참조하십시오.

작업을 수행하려면 이미지 리포지토리에 이미 푸시된 기존 이미지가 필요합니다. 이미지 매핑은 imagedefinitions.json 파일에서 제공되므로 Amazon ECR 소스를 파이프라인의 소스 작업으 로 포함할 필요는 없습니다.

출력 아티팩트

- 아티팩트 수: 0
- 설명: 이 작업 유형에는 출력 아티팩트가 적용되지 않습니다.

서비스 역할 권한: Amazon ECS 표준 작업

Amazon ECS의 경우 Amazon ECS 배포 작업을 통해 파이프라인을 만드는 데 필요한 최소 권한은 다 음과 같습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    { "Sid": "TaskDefinitionPermissions",
      "Effect": "Allow",
      "Action": [
        "ecs:DescribeTaskDefinition",
        "ecs:RegisterTaskDefinition"
      ],
      "Resource": [
        "*"
      ٦
    },
    { "Sid": "ECSServicePermissions",
      "Effect": "Allow",
      "Action": [
        "ecs:DescribeServices",
        "ecs:UpdateService"
      ],
      "Resource": [
        "arn:aws:ecs:*:{{customerAccountId}}:service/[[clusters]]/*"
      ]
    },
    { "Sid": "ECSTagResource",
      "Effect": "Allow",
```

서비스 역할 권한: Amazon ECS 표준 작업

```
AWS CodePipeline
```

"Action": [

```
"ecs:TagResource"
       ],
       "Resource": [
         "arn:aws:ecs:*:{{customerAccountId}}:task-definition/[[taskDefinitions]]:*"
       ],
       "Condition": {
         "StringEquals": {
           "ecs:CreateAction": [
             "RegisterTaskDefinition"
           ]
         }
       }
     },
     { "Sid": "IamPassRolePermissions",
       "Effect": "Allow",
       "Action": "iam:PassRole",
       "Resource": [
         "arn:aws:iam::{{customerAccountId}}:role/[[passRoles]]"
       ],
       "Condition": {
         "StringEquals": {
           "iam:PassedToService": [
             "ecs.amazonaws.com",
             "ecs-tasks.amazonaws.com"
           ]
         }
       }
     }
   ]
 }
Amazon ECS에서 태그 지정 권한 부여 사용으로 옵트인할 수 있습니다. 옵트인을 통해 다음 권한:
```

Amazon ECS에서 태그 지정 권한 부여 자용으로 곱르인할 수 있습니다. 곱르인을 통해 다음 권한: ecs:TagResource를 부여해야 합니다. 옵트인하는 방법과 권한 필요 여부 및 태그 권한 부여 적용 여 부에 대한 자세한 내용은 Amazon Elastic Container Service 개발자 안내서의 <u>태그 지정 권한 부여 타</u> <u>임라인</u>을 참조하세요.

작업에 IAM 역할을 사용하려면 iam: PassRole 권한을 추가해야 합니다. 자세한 내용은 <u>Amazon</u> ECS 태스크 실행 IAM 역할과 태스크에 대한 IAM 역할을 참조하세요. 다음 정책 텍스트를 사용합니다.

사용자 가이드

작업 선언

YAML

```
Name: DeployECS
ActionTypeId:
  Category: Deploy
  Owner: AWS
  Provider: ECS
  Version: '1'
RunOrder: 2
Configuration:
  ClusterName: my-ecs-cluster
  ServiceName: sample-app-service
  FileName: imagedefinitions.json
  DeploymentTimeout: '15'
OutputArtifacts: []
InputArtifacts:
  - Name: my-image
```

JSON

```
{
    "Name": "DeployECS",
    "ActionTypeId": {
        "Category": "Deploy",
        "Owner": "AWS",
        "Provider": "ECS",
        "Version": "1"
    },
    "RunOrder": 2,
    "Configuration": {
        "ClusterName": "my-ecs-cluster",
        "ServiceName": "sample-app-service",
        "FileName": "imagedefinitions.json",
        "DeploymentTimeout": "15"
    },
    "OutputArtifacts": [],
    "InputArtifacts": [
        {
            "Name": "my-image"
        }
    ]
```

},

다음 사항도 참조하세요.

이 작업을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

- ECRBuildandPublish 작업을 사용하여 이미지를 푸시한 다음 ECS 표준 작업을 사용하여 Amazon ECS에 배포하는 방법을 보여주는 <u>자습서: CodePipeline을 사용하여 Docker 이미지를 빌드하고</u> Amazon ECR에 푸시(V2 유형) 자습서는 섹션을 참조하세요.
- <u>자습서: CodePipeline을 사용한 지속적 배포</u> 이 자습서에서는 CodeCommit과 같은 소스 파일 리 포지토리에 저장하는 Dockerfile을 만드는 방법을 보여줍니다. 다음으로 자습서에서는 도커 이미지 를 빌드하여 Amazon ECR로 푸시하고 imagedefinitions.json 파일을 생성하는 CodeBuild BuildSpec 파일을 통합하는 방법을 보여줍니다. 마지막으로 Amazon ECS 서비스와 작업 정의를 생성한 다음 Amazon ECS 배포 작업을 사용하여 파이프라인을 생성합니다.

Note

이 주제 및 자습서에서는 CodePipeline의 Amazon ECS 표준 배포 작업에 대해 설명합니다. CodePipeline에서의 CodeDeploy 블루/그린 배포 작업에 대한 Amazon ECS에 대한 정보는 <u>자습서: Amazon ECR 소스 및 ECS-to-CodeDeploy 배포를 사용하여 파이프라인 생성</u>을 참 조하세요.

 Amazon Elastic Container Service 개발자 안내서 - 도커 이미지 및 컨테이너, Amazon ECS 서비스 및 클러스터, Amazon ECS 작업 세트 사용에 대한 자세한 내용은 <u>Amazon ECS란 무엇입니까?</u>를 참 조하세요.

Amazon Elastic Kubernetes Service EKS 배포 작업 참조

EKSDeploy 작업을 사용하여 Amazon EKS 서비스를 배포할 수 있습니다. 배포에는 CodePipeline이 이미지를 배포하는 데 사용하는 Kubernetes 매니페스트 파일이 필요합니다.

파이프라인을 생성하기 전에 Amazon EKS 리소스를 이미 생성하고 이미지를 이미지 리포지토리에 저 장해야 합니다. 선택적으로 클러스터에 대한 VPC 정보를 제공할 수 있습니다.

▲ Important

이 작업은 CodePipeline 관리형 CodeBuild 컴퓨팅을 사용하여 빌드 환경에서 명령을 실행합니다. 명령 작업을 실행하면 AWS CodeBuild에서 별도의 요금이 발생합니다.

Note

EKS 배포 작업은 V2 유형 파이프라인에서만 사용할 수 있습니다.

EKS 작업은 퍼블릭 및 프라이빗 EKS 클러스터를 모두 지원합니다. 프라이빗 클러스터는 EKS에서 권 장하는 유형이지만 두 유형 모두 지원됩니다.

EKS 작업은 교차 계정 작업에 대해 지원됩니다. 교차 계정 EKS 작업을 추가하려면 작업 선언의 대상 계정actionRoleArn에서를 추가합니다.

주제

- <u>작업 유형</u>
- 구성 파라미터
- 입력 아티팩트
- 출력 아티팩트
- <u>환경 변수</u>
- 출력 변수
- 서비스 역할 정책 권한
- <u>작업 선언</u>
- <u>다음 사항도 참조하세요.</u>

작업 유형

- 범주: Deploy
- 소유자: AWS
- 공급자: EKS
- 버전:1

구성 파라미터

ClusterName

필수 여부: 예

Amazon EKS의 Amazon EKS 클러스터입니다.

Helm 아래의 옵션

다음은 Helm이 선택한 배포 도구일 때 사용할 수 있는 옵션입니다.

HelmReleaseName

필수: 예(Helm 유형에만 필수)

배포의 릴리스 이름입니다.

HelmChartLocation

필수: 예(Helm 유형에만 필수)

배포의 차트 위치입니다.

HelmValuesFiles

필수: 아니요(Helm 유형의 경우에만 선택 사항)

배포의 차트 위치입니다.

Kubectl의 옵션

다음은 Kubectl이 선택한 배포 도구일 때 사용할 수 있는 옵션입니다.

ManifestFiles

필수: 예(Kubectl 유형에만 필수)

매니페스트 파일의 이름, 서비스의 컨테이너 이름을 설명하는 텍스트 파일, 이미지 및 태그입니 다. 이 파일을 사용하여 이미지 URI 및 기타 정보를 파라미터화합니다. 이 용도로 환경 변수를 사용할 수 있습니다.

파이프라인의 소스 리포지토리에이 파일을 저장합니다.

네임스페이스

필수 여부: 아니요

kubectl 또는 helm 명령에 사용할 kubernetes 네임스페이스입니다.
서브넷

필수 여부: 아니요

클러스터의 VPC에 대한 서브넷입니다. 이는 클러스터에 연결된 동일한 VPC의 일부입니다. 클러스 터에 아직 연결되지 않은 서브넷을 제공하고 여기에 지정할 수도 있습니다.

SecurityGroupIds

필수 여부: 아니요

클러스터의 VPC에 대한 보안 그룹입니다. 이는 클러스터에 연결된 동일한 VPC의 일부입니다. 클 러스터에 아직 연결되지 않은 보안 그룹을 제공하고 여기에 지정할 수도 있습니다.

입력 아티팩트

- 아티팩트 수: 1
- 설명: 작업은 파이프라인의 소스 파일 리포지토리에서 Kubernetes 매니페스트 파일 또는 차트 Helm 을 찾습니다. S3 버킷에 저장된 .tgz 형식의 헬름 차트를 사용하려면 S3 버킷/키를 소스 작업으로 구 성하면 됩니다. 예를 들어 제공된 객체 키는 입니다my-chart-0.1.0.tgz.

출력 아티팩트

- 아티팩트 수: 0
- 설명: 이 작업 유형에는 출력 아티팩트가 적용되지 않습니다.

환경 변수

매니페스트 파일 또는 헬름 차트 값 파일의 이미지 리포지토리 또는 이미지 태그와 같은 변수를 대체하 는 데 사용됩니다.

키

와 같은 키-값 환경 변수 페어의 키입니다\$IMAGE_TAG.

값

와 같은 키-값 페어의 값입니다v1.0. 파이프라인 작업 또는 파이프라인 변수의 출력 변수를 사용하 여 값을 파라미터화할 수 있습니다. 예를 들어 파이프라인에는를 사용하여 ECR 이미지를 생성하는 ECRBuildAndPublish 작업이 있을 수 있으며\${codepipeline.PipelineExecutionId}, EKS 작업은를 환경 변수의 \${codepipeline.PipelineExecutionId} 값으로 사용하여이 이미지 를 사용할 수 있습니다.

출력 변수

EKSClusterName

Amazon EKS의 Amazon EKS 클러스터입니다.

서비스 역할 정책 권한

이 작업을 실행하려면 파이프라인의 서비스 역할 정책에서 다음 권한을 사용할 수 있어야 합니다.

• EC2 작업: CodePipeline이 실행되면 EC2 인스턴스 권한이 필요합니다. 이는 EKS 클러스터를 생성 할 때 필요한 EC2 인스턴스 역할과 동일하지 않습니다.

기존 서비스 역할을 사용하는 경우이 작업을 사용하려면 서비스 역할에 대해 다음 권한을 추가해야 합니다.

- ec2:CreateNetworkInterface
- ec2:DescribeDhcpOptions
- ec2:DescribeNetworkInterfaces
- ec2:DeleteNetworkInterface
- ec2:DescribeSubnets
- ec2:DescribeSecurityGroups
- ec2:DescribeVpcs
- EKS 작업: CodePipeline이 작업을 실행할 때 EKS 클러스터 권한이 필요합니다. 이는 EKS 클러스터 를 생성할 때 필요한 IAM EKS 클러스터 역할과 동일하지 않습니다.

기존 서비스 역할을 사용하는 경우이 작업을 사용하려면 서비스 역할에 대해 다음 권한을 추가해야 합니다.

- eks:DescribeCluster
- 로그 스트림 작업: CodePipeline이 작업을 실행하면 CodePipeline은 다음과 같이 파이프라인 이름을 사용하여 로그 그룹을 생성합니다. 이렇게 하면 파이프라인 이름을 사용하여 리소스를 로깅할 수 있
 는 권한을 범위 아래로 지정할 수 있습니다.

/aws/codepipeline/MyPipelineName

기존 서비스 역할을 사용하는 경우이 작업을 사용하려면 서비스 역할에 대해 다음 권한을 추가해야 합니다.

- logs:CreateLogGroup
- logs:CreateLogStream
- logs:PutLogEvents

서비스 역할 정책 설명에서 다음 예제와 같이 권한 범위를 리소스 수준으로 좁힙니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster"
            ],
            "Resource": "arn:aws:eks:*:YOUR_AWS_ACCOUNT_ID:cluster/YOUR_CLUSTER_NAME"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeVpcs",
                "ec2:DescribeRouteTables"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
```

```
"logs:PutLogEvents"
],
    "Resource": [ "arn:aws:logs:*:YOUR_AWS_ACCOUNT_ID:log-group:/aws/
codepipeline/YOUR_PIPELINE_NAME","arn:aws:logs:*:YOUR_AWS_ACCOUNT_ID:log-group:/aws/
codepipeline/YOUR_PIPELINE_NAME:*"]
    },
]
```

작업 세부 정보 대화 상자를 사용하여 콘솔에서 로그를 보려면 로그를 볼 수 있는 권한을 콘솔 역할에 추가해야 합니다. 자세한 내용은 <u>CodePipeline 콘솔에서 컴퓨팅 로그를 보는 데 필요한 권한</u>에서 콘솔 권한 정책 예제를 참조하세요.

서비스 역할을 클러스터의 액세스 항목으로 추가

파이프라인의 서비스 역할 정책에서 권한을 사용할 수 있게 되면 CodePipeline 서비스 역할을 클러스 터의 액세스 항목으로 추가하여 클러스터 권한을 구성합니다.

업데이트된 권한이 있는 작업 역할을 사용할 수도 있습니다. 자세한 내용은의 자습서 예제를 참조하세 요4단계: CodePipeline 서비스 역할에 대한 액세스 항목 생성.

작업 선언

YAML

```
Name: DeployEKS
ActionTypeId:
  Category: Deploy
  Owner: AWS
  Provider: EKS
  Version: '1'
RunOrder: 2
Configuration:
  ClusterName: my-eks-cluster
  ManifestFiles: ManifestFile.json
OutputArtifacts: []
InputArtifacts:
  - Name: SourceArtifact
```

JSON

{

```
"Name": "DeployECS",
    "ActionTypeId": {
        "Category": "Deploy",
        "Owner": "AWS",
        "Provider": "EKS",
        "Version": "1"
    },
    "RunOrder": 2,
    "Configuration": {
        "ClusterName": "my-eks-cluster",
        "ManifestFiles": "ManifestFile.json"
    },
    "OutputArtifacts": [],
    "InputArtifacts": [
        {
            "Name": "SourceArtifact"
        }
    ]
},
```

다음 사항도 참조하세요.

이 작업을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

• EKS 클러스터 및 Kubernetes 매니페스트 파일을 생성하여 파이프라인에 작업을 추가하는 방법을 보여주는 <u>자습서: CodePipeline을 사용하여 Amazon EKS에 배포</u> 자습서는 섹션을 참조하세요.

Amazon S3 배포 작업 참조

Amazon S3 배포 작업을 사용하여 정적 웹 사이트 호스팅 또는 보관을 위해 Amazon S3 버킷에 파일을 배포합니다. 버킷에 업로드하기 전에 배포 파일을 추출할지 여부를 지정할 수 있습니다.

Note

이 참조 주제에서는 CodePipeline의 Amazon S3 배포 작업에 대해 설명합니다. 여기서 배포 플 랫폼은 호스팅용으로 구성된 Amazon S3 버킷입니다. CodePipeline에서의 Amazon S3 소스 작업에 대한 참조 정보는 Amazon S3 소스 작업 참조을 참조하세요.

주제

다음 사항도 참조하세요.

- 작업 유형
- 구성 파라미터
- 입력 아티팩트
- 출력 아티팩트
- 서비스 역할 권한: S3 배포 작업
- 예제 작업 구성
- 다음 사항도 참조하세요.

작업 유형

- 범주: Deploy
- 소유자: AWS
- 공급자: S3
- 버전: 1

구성 파라미터

BucketName

필수 여부: 예

파일을 배포할 Amazon S3 버킷 이름입니다.

Extract

필수 여부: 예

true인 경우 업로드 전에 파일을 추출하도록 지정합니다. 그렇지 않으면 호스팅된 정적 웹 사 이트의 경우처럼 애플리케이션 파일은 업로드를 위해 압축된 상태로 유지됩니다. false인 경우 ObjectKey는 필수 항목입니다.

ObjectKey

조건부. Extract = false인 경우 필수

S3 버킷의 객체를 고유하게 식별하는 Amazon S3 객체 키의 이름입니다.

KMSEncryptionKeyARN

필수 여부: 아니요

호스트 버킷에 대한 AWS KMS 암호화 키의 ARN입니다. AWS KMS key파라미터는 제공된 KMSEncryptionKeyARN를 사용하여 업로드된 아티팩트를 암호화합니다. KMS 키의 경우 키 ID, 키 ARN 또는 별칭 ARN을 사용할 수 있습니다.

Note

별칭은 KMS 키를 생성한 계정에서만 인식됩니다. 교차 계정 작업의 경우 키 ID 또는 키 ARN만 사용하여 키를 식별할 수 있습니다. 계정 간 작업에는 다른 계정(AccountB)의 역할 을 사용하는 것이 포함되므로 키 ID를 지정하면 다른 계정(AccountB)의 키가 사용됩니다.

🛕 Important

CodePipeline은 대칭 KMS 키만 지원합니다. 비대칭 KMS 키를 사용하여 S3 버킷의 데이터 를 암호화하지 마십시오.

CannedACL

필수 여부: 아니요

CannedACL 파라미터는 Amazon S3에 배포된 객체에 지정된 <u>미리 준비된 ACL</u>을 적용합니다. 이 는 객체에 적용된 기존 ACL을 덮어씁니다.

CacheControl

필수 여부: 아니요

CacheControl 파라미터는 버킷의 객체에 대한 요청/응답의 캐싱 동작을 제어합니다. 유효한 값 목록의 경우 HTTP 작업에 대한 <u>Cache-Control</u> 헤더 필드를 확인합니다. CacheControl에 여 러 값을 입력하려면 각 값 사이에 쉼표를 사용합니다. 이 CLI용 예제와 같이 각 쉼표 뒤에 공백을 추 가할 수 있습니다(선택 사항).

"CacheControl": "public, max-age=0, no-transform"

입력 아티팩트

• 아티팩트 수: 1

입력 아티팩트

• 설명: 배포 또는 보관용 파일은 소스 리포지토리에서 가져와서 CodePipeline으로 압축하고 업로드합 니다.

출력 아티팩트

- 아티팩트 수: 0
- 설명: 이 작업 유형에는 출력 아티팩트가 적용되지 않습니다.

서비스 역할 권한: S3 배포 작업

S3 배포 작업 지원의 경우 정책 설명에 다음을 추가합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectAcl",
        "s3:PutObjectVersionAcl",
        "s3:GetBucketVersioning",
        "s3:GetBucketAcl",
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::[[s3DeployBuckets]]",
        "arn:aws:s3:::[[s3DeployBuckets]]/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "{{customerAccountId}}"
        }
      }
    }
  ]
}
```

S3 배포 작업 지원의 경우 S3 객체에 태그가 있는 경우 정책 설명에 다음 권한도 추가해야 합니다.

```
"s3:GetObjectTagging",
"s3:GetObjectVersionTagging",
"s3:PutObjectTagging"
```

예제 작업 구성

다음은 작업 구성에 대한 예제을 보여 줍니다.

Extract를 false로 설정한 경우의 구성 예제

다음 예제는 Extract 필드가 false로 설정된 상태에서 작업을 만들 때의 기본 작업 구성을 보여줍니 다.

YAML

```
Name: Deploy
Actions:
  - Name: Deploy
    ActionTypeId:
      Category: Deploy
      Owner: AWS
      Provider: S3
      Version: '1'
    RunOrder: 1
    Configuration:
      BucketName: website-bucket
      Extract: 'false'
      ObjectKey: MyWebsite
    OutputArtifacts: []
    InputArtifacts:
      - Name: SourceArtifact
    Region: us-west-2
    Namespace: DeployVariables
```

JSON



Extract를 true로 설정한 경우의 구성 예제

다음 예제는 Extract 필드가 true로 설정된 상태에서 작업을 만들 때의 기본 작업 구성을 보여줍니 다.

YAML

```
Name: Deploy
Actions:
  - Name: Deploy
  ActionTypeId:
    Category: Deploy
    Owner: AWS
    Provider: S3
    Version: '1'
    RunOrder: 1
    Configuration:
    BucketName: website-bucket
    Extract: 'true'
```

```
OutputArtifacts: []
InputArtifacts:
    - Name: SourceArtifact
Region: us-west-2
Namespace: DeployVariables
```

JSON

```
{
    "Name": "Deploy",
    "Actions": [
        {
            "Name": "Deploy",
            "ActionTypeId": {
                "Category": "Deploy",
                "Owner": "AWS",
                "Provider": "S3",
                "Version": "1"
            },
            "RunOrder": 1,
            "Configuration": {
                "BucketName": "website-bucket",
                "Extract": "true"
                },
            "OutputArtifacts": [],
            "InputArtifacts": [
                {
                     "Name": "SourceArtifact"
                }
            ],
            "Region": "us-west-2",
            "Namespace": "DeployVariables"
        }
    ]
},
```

다음 사항도 참조하세요.

이 작업을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

• <u>자습서: Amazon S3를 배포 공급자로 사용하는 파이프라인 생성</u> - 이 자습서에서는 S3 배포 작 업을 사용하여 파이프라인을 생성하는 두 가지 예제를 안내합니다. 샘플 파일을 다운로드하고, CodeCommit 리포지토리에 파일을 업로드하고, S3 버킷을 생성하고, 호스팅용 버킷을 구성합니다. 그런 다음 CodePipeline 콘솔을 사용하여 파이프라인을 생성하고 Amazon S3 배포 구성을 지정합니 다.

 <u>Amazon S3 소스 작업 참조</u> - 이 작업 참조는 CodePipeline의 Amazon S3 소스 작업에 대한 참조 정 보와 예제를 제공합니다.

Amazon S3 소스 작업 참조

구성된 버킷 및 객체 키에 새 객체가 업로드될 때 파이프라인을 트리거합니다.

Note

이 참조 주제에서는 CodePipeline의 Amazon S3 소스 작업에 대해 설명합니다. 여기서 소스 위 치는 버전 관리를 위해 구성된 Amazon S3 버킷입니다. CodePipeline에서의 Amazon S3 배포 작업에 대한 참조 정보는 Amazon S3 배포 작업 참조을 참조하세요.

Amazon S3 버킷을 생성하여 애플리케이션 파일의 원본 위치로 사용할 수 있습니다.

Note

소스 버킷을 생성할 때는 버킷에 대한 버전 관리를 활성화해야 합니다. 기존의 Amazon S3 버 킷을 사용하려면 기존 버킷에 대한 버전 관리를 활성화하기 위한 <u>버전 관리 사용</u>을 참조하세 요.

콘솔을 사용하여 파이프라인을 생성하거나 편집하는 경우 CodePipeline은 S3 소스 버킷에서 변경이 발생할 때 파이프라인을 시작하는 EventBridge 규칙을 생성합니다.

Note

Amazon ECR, Amazon S3 또는 CodeCommit 소스의 경우 입력 변환 항목을 사용하여 파이프 라인 이벤트에 대한 EventBridgerevisionValue의를 사용하는 소스 재정의를 생성할 수도 있습니다. 여기서 revisionValue는 객체 키, 커밋 또는 이미지 ID에 대한 소스 이벤트 변수 에서 파생됩니다. 자세한 내용은, <u>Amazon ECR 소스 작업 및 EventBridge 리소스 이벤트에 대</u> <u>해 활성화된 소스를 사용하여 Amazon S3 소스 작업에 연결</u>또는의 절차에 포함된 입력 변환 항 목의 선택적 단계를 참조하세요CodeCommit 소스 작업 및 EventBridge. Amazon S3 작업을 통해 파이프라인을 연결하기 전에 Amazon S3 소스 버킷을 생성하고 원본 파일을 단일 ZIP 파일로 업로드해야 합니다.

Note

Amazon S3가 파이프라인의 소스 공급자인 경우, 소스 파일을 .zip 하나로 압축하고 그 .zip을 소스 버킷에 업로드할 수 있습니다. 압축이 풀린 단일 파일을 업로드할 수도 있지만 .zip 파일을 예상하는 다운스트림 작업은 실패합니다.

주제

- <u>작업 유형</u>
- 구성 파라미터
- 입력 아티팩트
- <u>출력 아티팩트</u>
- <u>출력 변수</u>
- <u>서비스 역할 권한: S3 소스 작업</u>
- <u>작업 선언</u>
- <u>다음 사항도 참조하세요.</u>

작업 유형

- 범주: Source
- 소유자: AWS
- 공급자: S3
- 버전:1

구성 파라미터

S3Bucket

필수 여부: 예

소스 변경 사항이 감지되는 Amazon S3 버킷의 이름입니다.

S3ObjectKey

필수 여부: 예

소스 변경 사항이 감지되는 Amazon S3 객체 키의 이름입니다.

AllowOverrideForS3ObjectKey

필수 여부: 아니요

AllowOverrideForS30bjectKey는 StartPipelineExecution의 소스 재정의가 소스 작업 에 이미 구성된 S30bjectKey를 재정의할 수 있는지 여부를 제어합니다. S3 객체 키를 사용한 소 스 재정의에 대한 자세한 내용은 <u>소스 개정 재정의로 파이프라인 시작</u> 섹션을 참조하세요.

▲ Important

AllowOverrideForS30bjectKey를 생략하면 CodePipeline은 이 파라미터를 false로 설정하여 소스 작업에서 S3 ObjectKey를 재정의할 수 있는 기능을 기본값으로 설정합니다.

이 파라미터에 유효한 값은 다음과 같습니다.

 true: 설정된 경우 파이프라인 실행 중에 소스 개정 재정의로 사전 구성된 S3 객체 키를 재정의 할 수 있습니다.

Note

모든 CodePipeline 사용자가 새 파이프라인 실행을 시작하는 동안 사전 구성된 S3 객체 키를 재정의할 수 있도록 허용하려면 AllowOverrideForS30bjectKey를 true로 설 정해야 합니다.

• false:

설정하면 CodePipeline은 소스 개정 재정의를 사용하여 S3 객체 키를 재정의하도록 허용하지 않 습니다. 이는 이 파라미터의 기본값이기도 합니다.

PollForSourceChanges

필수 여부: 아니요

PollForSourceChanges은 CodePipeline이 소스 변경 사항에 대한 Amazon S3 소스 버킷을 폴 링할지 여부를 제어합니다. 대신 CloudWatch Events 및 CloudTrail을 사용하여 소스 변경 사항을 감지하는 것이 좋습니다. CloudWatch Events 구성에 대한 자세한 내용은 <u>S3 소스 및 CloudTrail 트</u> <u>레일을 사용하여 폴링 파이프라인 마이그레이션(CLI)</u> 또는 <u>S3 소스 및 CloudTrail 추적을 사용하여</u> 폴링 파이프라인 마이그레이션(AWS CloudFormation 템플릿)을 참조하세요.

🛕 Important

CloudWatch Events를 구성하려는 경우, 중복된 파이프라인 실행이 발생하지 않도록 PollForSourceChanges를 false로 설정해야 합니다.

이 파라미터에 유효한 값은 다음과 같습니다.

• true: 이 값으로 설정된 경우 CodePipeline은 소스 변경 사항에 대해 소스 위치를 폴링합니다.

1 Note

PollForSourceChanges를 생략하면 CodePipeline은 기본적으로 소스 변경 사항에 대해 소스 위치를 폴링합니다. 이러한 동작은 PollForSourceChanges이 포함되었고 true로 설정된 경우와 똑같습니다.

 false: 이 값으로 설정된 경우 CodePipeline은 소스 변경 사항에 대해 소스 위치를 폴링하지 않 습니다. 소스 변경 사항을 감지하도록 CloudWatch Events 규칙을 구성하려면 이 설정을 사용합 니다.

입력 아티팩트

- 아티팩트 수: 0
- 설명: 이 작업 유형에는 입력 아티팩트가 적용되지 않습니다.

출력 아티팩트

- 아티팩트 수: 1
- 설명: 파이프라인에 연결하도록 구성된 소스 버킷에서 사용할 수 있는 아티팩트를 제공합니다. 버 킷에서 생성된 아티팩트는 Amazon S3 작업의 출력 아티팩트입니다. Amazon S3 객체 메타데이터 (ETag 및 버전 ID)는 트리거된 파이프라인 실행의 소스 개정으로 CodePipeline에 표시됩니다.

출력 변수

이 작업을 구성하면 파이프라인에서 다운스트림 작업의 작업 구성에서 참조할 수 있는 변수가 생성됩 니다. 이 작업은 작업에 네임스페이스가 없는 경우에도 출력 변수로 볼 수 있는 변수를 생성합니다. 이 러한 변수를 다운스트림 작업 구성에서 사용할 수 있도록 네임스페이스를 사용하여 작업을 구성합니 다.

CodePipeline의 변수에 대한 자세한 내용은 변수 참조 단원을 참조하세요.

BucketName

파이프라인을 트리거한 소스 변경과 관련된 Amazon S3 버킷의 이름입니다.

ETag

파이프라인을 트리거한 소스 변경과 관련된 개체의 엔터티 태그입니다. ETag는 객체의 MD5 해시 입니다. ETag는 객체의 내용에 대한 변경 사항만 반영하고 메타데이터에 대한 변경은 반영하지 않 습니다.

ObjectKey

파이프라인을 트리거한 소스 변경과 관련된 Amazon S3 객체 키의 이름입니다.

VersionId

파이프라인을 트리거한 소스 변경과 관련된 객체 버전의 버전 ID입니다.

서비스 역할 권한: S3 소스 작업

S3 소스 작업 지원의 경우 정책 설명에 다음을 추가합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
         "Effect": "Allow",
         "Action": [
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:GetBucketVersioning",
            "s3:GetBucketAcl",
            "s3:GetBucketLocation",
            "s3:GetBucketLocation",
```

```
"s3:GetObjectTagging",
"s3:GetObjectVersionTagging"
],
"Resource": [
"arn:aws:s3:::[[S3Bucket]]",
"arn:aws:s3:::[[S3Bucket]]/*"
],
"Condition": {
"StringEquals": {
"aws:ResourceAccount": "{{customerAccountId}}"
}
}
}
}
```

작업 선언

YAML

```
Name: Source
Actions:
  - RunOrder: 1
    OutputArtifacts:
      - Name: SourceArtifact
    ActionTypeId:
      Provider: S3
      Owner: AWS
      Version: '1'
      Category: Source
    Region: us-west-2
    Name: Source
    Configuration:
      S3Bucket: amzn-s3-demo-source-bucket
      S3ObjectKey: my-application.zip
      PollForSourceChanges: 'false'
    InputArtifacts: []
```

JSON

"Name": "Source",

{

```
"Actions": [
        {
            "RunOrder": 1,
            "OutputArtifacts": [
                {
                     "Name": "SourceArtifact"
                }
            ],
            "ActionTypeId": {
                "Provider": "S3",
                "Owner": "AWS",
                "Version": "1",
                "Category": "Source"
            },
            "Region": "us-west-2",
            "Name": "Source",
            "Configuration": {
                "S3Bucket": "amzn-s3-demo-source-bucket",
                "S3ObjectKey": "my-application.zip",
                "PollForSourceChanges": "false"
            },
            "InputArtifacts": []
        }
    ]
},
```

다음 사항도 참조하세요.

이 작업을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

• <u>자습서: 간단한 파이프라인 생성(S3 버킷)</u> - 이 자습서에서는 앱 사양 파일 샘플과 CodeDeploy 애플 리케이션 및 배포 그룹 샘플을 제공합니다. 이 자습서를 사용하여 Amazon EC2 인스턴스에 배포되 는 Amazon S3 소스로 파이프라인을 생성합니다.

AWS AppConfig 배포 작업 참조

AWS AppConfig는의 기능입니다 AWS Systems Manager. AppConfig는 모든 규모의 애플리케이션에 대한 제어된 배포를 지원하며 내장 유효성 검사 및 모니터링을 포함합니다. Amazon EC2 인스턴스, AWS Lambda컨테이너, 모바일 애플리케이션 또는 IoT 디바이스에서 호스팅되는 애플리케이션과 함 께 AppConfig를 사용할 수 있습니다. AppConfig 배포 작업은 파이프라인 소스 위치에 저장된 구성을 지정된 AppConfig 애플리케이션, 환 경 및 구성 프로필에 배포하는 AWS CodePipeline 작업입니다. AppConfig 배포 전략에 정의된 기본 설 정을 사용합니다.

작업 유형

- 범주: Deploy
- 소유자: AWS
- 공급자: AppConfig
- 버전:1

구성 파라미터

Application

필수 여부: 예

구성 및 배포에 대한 세부 정보가 포함된 AWS AppConfig 애플리케이션의 ID입니다. 환경

필수 여부: 예

구성이 배포되는 AWS AppConfig 환경의 ID입니다.

ConfigurationProfile

필수 여부: 예

배포할 AWS AppConfig 구성 프로필의 ID입니다.

InputArtifactConfigurationPath

필수 여부: 예

배포할 입력 아티팩트 내 구성 데이터의 파일 경로입니다.

DeploymentStrategy

필수 여부: 아니요

배포에 사용할 AWS AppConfig 배포 전략입니다.

입력 아티팩트

- 아티팩트 수: 1
- 설명: 배포 작업을 위한 입력 아티팩트입니다.

출력 아티팩트

해당 사항 없음.

서비스 역할 권한: AppConfig 작업

CodePipeline이 작업을 실행할 때 CodePipeline 서비스 역할 정책에는 최소 권한으로 액세스를 유지하 기 위해 리소스 수준으로 범위가 적절하게 축소된 다음 권한이 필요합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Action": [
                "appconfig:StartDeployment",
                "appconfig:StopDeployment",
                "appconfig:GetDeployment"
            ],
            "Resource": [
             "arn:aws:appconfig:*:{{customerAccountId}}:application/[[Application]]",
             "arn:aws:appconfig:*:{{customerAccountId}}:application/[[Application]]/*",
             "arn:aws:appconfig:*:{{customerAccountId}}:deploymentstrategy/*"
         ],
            "Effect": "Allow"
        }
    ]
}
```

예제 작업 구성

YAML

name: Deploy
actions:
 - name: Deploy

actionTypeId: category: Deploy owner: AWS provider: AppConfig version: '1' runOrder: 1 configuration: Application: 2s2qv57 ConfigurationProfile: PvjrpU DeploymentStrategy: frqt7ir Environment: 9tm27yd InputArtifactConfigurationPath: / outputArtifacts: [] inputArtifacts: - name: SourceArtifact region: us-west-2 namespace: DeployVariables

JSON

```
{
    "name": "Deploy",
    "actions": [
        {
            "name": "Deploy",
            "actionTypeId": {
                "category": "Deploy",
                "owner": "AWS",
                "provider": "AppConfig",
                "version": "1"
            },
            "runOrder": 1,
            "configuration": {
                "Application": "2s2qv57",
                "ConfigurationProfile": "PvjrpU",
                "DeploymentStrategy": "frqt7ir",
                "Environment": "9tm27yd",
                "InputArtifactConfigurationPath": "/"
            },
            "outputArtifacts": [],
            "inputArtifacts": [
                {
                    "name": "SourceArtifact"
```

```
}
],
"region": "us-west-2",
"namespace": "DeployVariables"
}
]
}
```

다음 사항도 참조하세요.

이 작업을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

- <u>AWS AppConfig</u> AWS AppConfig 배포에 대한 자세한 내용은 AWS Systems Manager 사용 설명서 를 참조하세요.
- <u>자습서: AWS AppConfig를 배포 공급자로 사용하는 파이프라인 생성</u> -이 자습서에서는 간단한 배포 구성 파일 및 AppConfig 리소스 설정을 시작하고 콘솔을 사용하여 AWS AppConfig 배포 작업을 통 해 파이프라인을 생성하는 방법을 보여줍니다.

AWS CloudFormation 배포 작업 참조

AWS CloudFormation 스택에서 작업을 실행합니다. 스택은 단일 단위로 관리할 수 있는 AWS 리소 스 모음입니다. 스택의 리소스는 스택의 AWS CloudFormation 템플릿으로 정의합니다. 변경 세트는 원래 스택을 변경하지 않고 볼 수 있는 비교를 만듭니다. 스택 및 변경 세트에서 수행할 수 있는 AWS CloudFormation 작업 유형에 대한 자세한 내용은 ActionMode 파라미터를 참조하세요.

스택 작업이 실패한 AWS CloudFormation 작업에 대한 오류 메시지를 구성하기 위해 CodePipeline은 API를 AWS CloudFormation DescribeStackEvents 호출합니다. 작업 IAM 역할에 해당 API에 대한 액세스 권한이 있는 경우 첫 번째 실패 리소스에 대한 세부 정보가 CodePipeline 오류 메시지에 포함 됩니다. 그렇지 않으면 역할 정책에 적절한 권한이 없는 경우 CodePipeline은 API 액세스를 무시하고 대신 일반 오류 메시지를 표시합니다. 이렇게 하려면 파이프라인의 서비스 역할이나 다른 IAM 역할에 cloudformation:DescribeStackEvents 권한을 추가해야 합니다.

리소스 세부 정보가 파이프라인 오류 메시지에 표시되지 않도록 하려면 cloudformation:DescribeStackEvents 권한을 제거하여 작업 IAM 역할에 대한 이 권한을 취소 할 수 있습니다.

주제

• <u>작업 유형</u>

- 구성 파라미터
- 입력 아티팩트
- 출력 아티팩트
- <u>출력 변수</u>
- 서비스 역할 권한: AWS CloudFormation 작업
- <u>작업 선언</u>
- 다음 사항도 참조하세요.

작업 유형

- 범주: Deploy
- 소유자: AWS
- 공급자: CloudFormation
- 버전:1

구성 파라미터

ActionMode

필수 여부: 예

ActionMode는 스택 또는 변경 세트에서 AWS CloudFormation 수행하는 작업의 이름입니다. 사용 할 수 있는 작업 모드는 다음과 같습니다.

- CHANGE_SET_EXECUTE는 지정된 리소스 업데이트 세트를 기반으로 하는 리소스 스택에 대한 변경 세트를 실행합니다. 이 작업을 수행하면가 스택을 변경하기 AWS CloudFormation 시작합니 다.
- CHANGE_SET_REPLACE: 변경 세트가 존재하지 않는 경우 스택 이름 및 제출하는 템플릿을 기반 으로 변경 세트를 생성합니다. 변경 세트가 있는 경우는 변경 세트를 AWS CloudFormation 삭제 한 다음 새 변경 세트를 생성합니다.
- CREATE_UPDATE는 스택이 없는 경우 스택을 생성합니다. 스택이 있는 경우 스택을 AWS CloudFormation 업데이트합니다. 이 작업을 사용하여 기존 스택을 업데이트합니다. REPLACE_ON_FAILURE과 달리 스택이 있으며 실패 상태인 경우 CodePipeline은 스택을 삭제하고 교체하지 않습니다.

- DELETE_ONLY: 스택을 삭제합니다. 존재하지 않는 스택을 지정하면 이 작업이 스택을 삭제하지 않고 성공적으로 완료됩니다.
- REPLACE_ON_FAILURE는 스택이 없는 경우 스택을 생성합니다. 스택이 존재하고 실패 상태인 경우는 스택을 AWS CloudFormation 삭제한 다음 새 스택을 생성합니다. 스택이 실패 상태가 아 닌 경우는 스택을 AWS CloudFormation 업데이트합니다.

AWS CloudFormation에 다음 상태 유형 중 하나가 표시되면 스택이 실패 상태가 됩니다.

- ROLLBACK_FAILED
- CREATE_FAILED
- DELETE_FAILED
- UPDATE_ROLLBACK_FAILED

이 작업을 사용하면 실패한 스택을 복구하거나 문제를 해결하지 않고 실패한 스택을 자동으로 대 체합니다.

Important

REPLACE_ON_FAILURE는 스택을 삭제할 수도 있으므로 테스트용으로만 사용하는 것이 좋습니다.

StackName

필수 여부: 예

StackName은 기존 스택 또는 생성하려는 스택의 이름입니다.

기능

필수 항목 여부: 조건부

Capabilities 사용은 템플릿이 일부 리소스를 자체적으로 생성 및 업데이트할 수 있으며, 이러 한 기능은 템플릿의 리소스 유형에 따라 결정됨을 승인합니다.

이 속성은 스택 템플릿에 IAM 리소스가 있거나 매크로가 포함된 템플릿에서 직접 스택을 만들 경우 필수 항목입니다. 이러한 방식으로 AWS CloudFormation 작업이 성공적으로 작동하려면 다음 기능 중 하나를 사용하여 작업을 수행하려는 것을 명시적으로 승인해야 합니다.

- CAPABILITY_IAM
- CAPABILITY_NAMED_IAM

• CAPABILITY_AUTO_EXPAND

기능 사이에 쉼표(공백 아님)를 사용하여 둘 이상의 기능을 지정할 수 있습니다. <u>작업 선언</u>의 예제 는 CAPABILITY_IAM 및 CAPABILITY_AUTO_EXPAND 속성이 모두 있는 항목을 보여줍니다.

Capabilities에 대한 자세한 내용은 AWS CloudFormation API 참조의 <u>UpdateStack</u>에 있는 속 성을 참조하세요.

ChangeSetName

필수 항목 여부: 조건부

ChangeSetName은 지정된 스택에 대해 생성하려는 새 변경 세트 또는 기존 변경 세트의 이름입니다.

이 속성은 CHANGE_SET_REPLACE 및 CHANGE_SET_EXECUTE 작업 모드의 경우 필수 항목입 니다. 기타 모든 작업 모드의 경우 이 속성은 무시됩니다.

RoleArn

필수 항목 여부: 조건부

RoleArn은 지정된 스택의 리소스에 대해 작업을 수행할 때 AWS CloudFormation 이 맡는 IAM 서 비스 역할의 ARN입니다. 변경 세트를 실행할 때는 RoleArn이 적용되지 않습니다. 변경 세트를 생 성하기 위해 CodePipeline을 사용하지 않는 경우 변경 세트 또는 스택에 연결된 역할이 있는지 확 인해야 합니다.

Note
 이 역할은 작업 선언 RoleArn에 구성된 대로 실행 중인 작업의 역할과 동일한 계정에 있어
 야 합니다.

이 속성은 다음 작업 모드의 경우 필수 항목입니다.

- CREATE_UPDATE
- REPLACE_ON_FAILURE
- DELETE_ONLY
- CHANGE_SET_REPLACE

1 Note

AWS CloudFormation 에는 템플릿에 대한 S3-signed URL이 부여되므로 아티팩트 버킷에 액세스할 수 있는 권한이 필요하지 RoleArn 않습니다. 하지만 서명된 URL을 생성하려면 작업 RoleArn에 아티팩트 버킷에 대한 액세스 권한이 필요합니다.

TemplatePath

필수 항목 여부: 조건부

TemplatePath는 AWS CloudFormation 템플릿 파일을 나타냅니다. 이 작업의 입력 아티팩트에 파일을 포함합니다. 파일 이름은 다음과 같은 형식을 따릅니다.

Artifactname::TemplateFileName

Artifactname은 CodePipeline에 표시될 때 입력 아티팩트 이름입니다. 예를 들어 아티팩트 이름 이 SourceArtifact이며 파일 이름이 template-export.json인 소스 단계는 다음 예제에 표 시된 대로 TemplatePath 이름을 생성합니다.

"TemplatePath": "SourceArtifact::template-export.json"

이 속성은 다음 작업 모드의 경우 필수 항목입니다.

- CREATE_UPDATE
- REPLACE_ON_FAILURE
- CHANGE_SET_REPLACE

기타 모든 작업 모드의 경우 이 속성은 무시됩니다.

Note

AWS CloudFormation 템플릿 본문이 포함된 템플릿 파일의 최소 길이는 1바이트이고 최대 길이는 1MB입니다. CodePipeline의 AWS CloudFormation 배포 작업의 경우 최대 입력 아 티팩트 크기는 항상 256MB입니다. 자세한 내용은 <u>in AWS CodePipeline 할당량</u> 및 <u>AWS</u> <u>CloudFormation 제한</u> 단원을 참조하십시오.

OutputFileName

필수 여부: 아니요

OutputFileName을 사용하여 CreateStackOutput.json과 같은 출력 파일 이름을 지정합니 다. CodePipeline은 이 이름을 이 작업의 파이프라인 출력 아티팩트에 추가합니다. JSON 파일에는 AWS CloudFormation 스택의 Outputs 섹션 내용이 포함되어 있습니다.

이름을 지정하지 않으면 CodePipeline에서는 출력 파일 또는 아티팩트를 생성하지 않습니다.

ParameterOverrides

필수 여부: 아니요

파라미터는 스택 템플릿에 정의되며, 이를 통해 스택 생성 또는 업데이트 시 값을 제공할 수 있습니다. JSON 객체를 사용하여 템플릿에서 파라미터 값을 설정할 수 있습니다. 이러한 값은 템플릿 구성 파일에 설정된 값을 재정의합니다. 파라미터 재정의 사용에 대한 자세한 내용은 <u>구성 속성</u> (JSON 객체)을 참조하십시오.

대부분의 파라미터 값에 템플릿 구성 파일을 사용하는 것이 좋습니다. 파라미터를 사용하면 파이프 라인이 실행될 때까지 알 수 없는 값만 재정의합니다. 자세한 내용은 AWS CloudFormation 사용 설 명서의 CodePipeline 파이프라인에서 파라미터 재정의 함수 사용을 참조하세요.

Note

스택 템플릿에는 모든 파라미터 이름이 존재해야 합니다.

TemplateConfiguration

필수 여부: 아니요

TemplateConfiguration은 템플릿 구성 파일입니다. 이 작업의 입력 아티팩트에 파일을 포함합 니다. 여기에는 템플릿 파라미터 값과 스택 정책이 포함될 수 있습니다. 템플릿 구성 파일 형식에 대 한 자세한 내용은 AWS CloudFormation 아티팩트를 참조하세요.

템플릿 구성 파일 이름은 다음 형식을 따릅니다.

Artifactname::TemplateConfigurationFileName

Artifactname은 CodePipeline에 표시될 때 입력 아티팩트 이름입니다. 예를 들어, 아티팩트 이름 이 SourceArtifact이며 파일 이름이 test-configuration.json인 소스 단계는 다음 예제에 표시된 대로 TemplateConfiguration 이름을 생성합니다.

"TemplateConfiguration": "SourceArtifact::test-configuration.json"

입력 아티팩트

- 아티팩트 수:0 to 10
- 설명: 입력으로 AWS CloudFormation 작업은 다음과 같은 목적으로 선택적으로 아티팩트를 수락합니다.
 - 실행할 스택 템플릿 파일을 제공하기 위해. TemplatePath 파라미터를 참조하십시오.
 - 사용할 템플릿 구성 파일을 제공하기 위해. TemplateConfiguration 파라미터를 참조하십시 오. 템플릿 구성 파일 형식에 대한 자세한 내용은 AWS CloudFormation 아티팩트를 참조하세요.
 - AWS CloudFormation 스택의 일부로 배포할 Lambda 함수에 대한 아티팩트를 제공합니다.

출력 아티팩트

- 아티팩트 수:0 to 1
- 설명: OutputFileName 파라미터를 지정하면 이 작업에 의해 지정된 이름의 JSON 파일이 포함되 어 있는 출력 아티팩트가 생성됩니다. JSON 파일에는 AWS CloudFormation 스택에 있는 출력 섹션 의 콘텐츠가 포함됩니다.

AWS CloudFormation 작업에 대해 생성할 수 있는 출력 섹션에 대한 자세한 내용은 <u>출력</u>을 참조하십 시오.

출력 변수

이 작업을 구성하면 파이프라인에서 다운스트림 작업의 작업 구성에서 참조할 수 있는 변수가 생성됩 니다. 이러한 변수를 다운스트림 작업 구성에서 사용할 수 있도록 네임스페이스를 사용하여 작업을 구 성합니다.

AWS CloudFormation 작업의 경우 변수는 스택 템플릿의 Outputs 섹션에 지정된 모든 값에서 생성됩 니다. 출력을 생성하는 유일한 CloudFormation 작업 모드는 스택 생성, 스택 업데이트 및 변경 세트 실 행과 같이 스택을 생성하거나 업데이트하는 모드입니다. 변수를 생성하는 해당 작업 모드는 다음과 같 습니다.

- CHANGE_SET_EXECUTE
- CHANGE_SET_REPLACE
- CREATE_UPDATE
- REPLACE_ON_FAILURE

자세한 내용은 <u>변수 참조</u> 단원을 참조하십시오. CloudFormation 출력 변수를 사용하는 파이프라인에 서 CloudFormation 배포 작업이 적용된 파이프라인을 생성하는 방법을 보여주는 자습서는 <u>자습서:</u> AWS CloudFormation 배포 작업의 변수를 사용하는 파이프라인 생성 단원을 참조하십시오.

서비스 역할 권한: AWS CloudFormation 작업

CodePipeline이 작업을 실행할 때 CodePipeline 서비스 역할 정책에는 최소 권한으로 액세스를 유지하 기 위해 파이프라인 리소스 ARN으로 범위가 적절하게 축소된 다음 권한이 필요합니다. 예를 들어 정책 설명에 다음을 추가합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCFNStackAccess",
      "Effect": "Allow",
      "Action": [
        "cloudformation:CreateStack",
        "cloudformation:UpdateStack",
        "cloudformation:DeleteStack",
        "cloudformation:DescribeStacks",
        "cloudformation:DescribeStackResources",
        "cloudformation:DescribeStackEvents",
        "cloudformation:GetTemplate",
        "cloudformation:DescribeChangeSet",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:ExecuteChangeSet"
      ],
      "Resource": [
        "arn:aws:cloudformation:*:{{customerAccountId}}:stack/[[cfnDeployStackNames]]/
* "
       ]
    },
    {
      "Sid": "ValidateTemplate",
      "Effect": "Allow",
      "Action": [
        "cloudformation:ValidateTemplate"
      ],
      "Resource": "*"
    },
    {
```

```
"Sid": "AllowIAMPassRole",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::{{customerAccountId}}:role/[[cfnExecutionRoles]]"
      ],
      "Condition": {
        "StringEqualsIfExists": {
          "iam:PassedToService": [
            "cloudformation.amazonaws.com"
          ]
        }
      }
    }
  ]
}
```

단, cloudformation:DescribeStackEvents 권한은 선택 사항이라는 점에 유의하십시오. 작업을 통해 보다 자세한 오류 메시지를 AWS CloudFormation 표시할 수 있습니다. 파이프라인 오류 메시지에 리소스 세부 정보가 표시되지 않도록 하려면 IAM 역할에서 이 권한을 취소할 수 있습니다.

작업 선언

YAML

```
Name: ExecuteChangeSet
ActionTypeId:
  Category: Deploy
 Owner: AWS
  Provider: CloudFormation
 Version: '1'
RunOrder: 2
Configuration:
 ActionMode: CHANGE_SET_EXECUTE
 Capabilities: CAPABILITY_NAMED_IAM, CAPABILITY_AUTO_EXPAND
 ChangeSetName: pipeline-changeset
  ParameterOverrides: '{"ProjectId": "my-project", "CodeDeployRole":
 "CodeDeploy_Role_ARN"}'
 RoleArn: CloudFormation_Role_ARN
 StackName: my-project--lambda
 TemplateConfiguration: 'my-project--BuildArtifact::template-configuration.json'
```

JSON

```
{
    "Name": "ExecuteChangeSet",
    "ActionTypeId": {
        "Category": "Deploy",
        "Owner": "AWS",
        "Provider": "CloudFormation",
        "Version": "1"
    },
    "RunOrder": 2,
    "Configuration": {
        "ActionMode": "CHANGE_SET_EXECUTE",
        "Capabilities": "CAPABILITY_NAMED_IAM, CAPABILITY_AUTO_EXPAND",
        "ChangeSetName": "pipeline-changeset",
        "ParameterOverrides": "{\"ProjectId\": \"my-project\", \"CodeDeployRole\":
 \"CodeDeploy_Role_ARN\"}",
        "RoleArn": "CloudFormation_Role_ARN",
        "StackName": "my-project--lambda",
        "TemplateConfiguration": "my-project--BuildArtifact::template-
configuration.json",
        "TemplatePath": "my-project--BuildArtifact::template-export.yml"
    },
    "OutputArtifacts": [],
    "InputArtifacts": [
        {
             "Name": "my-project-BuildArtifact"
        }
    ]
},
```

다음 사항도 참조하세요.

이 작업을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

• <u>구성 속성 참조</u> - AWS CloudFormation 사용 설명서의 이 참조 장에는 이러한 CodePipeline 파라미 터에 대한 추가 설명과 예제가 나와 있습니다. <u>AWS CloudFormation API 참조</u> - AWS CloudFormation API 참조의 <u>CreateStack</u> 파라미터는 템플릿 의 AWS CloudFormation 스택 파라미터를 설명합니다.

AWS CloudFormation StackSets 배포 작업 참조

CodePipeline은 CI/CD 프로세스의 일부로 AWS CloudFormation StackSets 작업을 수행할 수 있 는 기능을 제공합니다. 스택 세트를 사용하여 단일 AWS CloudFormation 템플릿을 사용하여 AWS 리전 간 AWS 계정에서 스택을 생성합니다. 각 스택에 포함된 모든 리소스는 스택 세트의 AWS CloudFormation 템플릿으로 정의됩니다. 스택 세트를 생성할 때 사용할 템플릿과 템플릿에 필요한 파 라미터 및 기능을 지정합니다.

AWS CloudFormation StackSets 개념에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 StackSets 개념을 참조하세요.

파이프라인을 함께 사용하는 두 가지 작업 유형을 통해 AWS CloudFormation StackSets와 통합합니 다.

- CloudFormationStackSet 작업은 파이프라인 소스 위치에 저장된 템플릿에서 스택 세트 또는 스택 인스턴스를 생성하거나 업데이트합니다. 스택 세트를 생성하거나 업데이트할 때마다 해당 변 경 내용을 지정된 인스턴스에 배포하기 시작합니다. 콘솔에서 파이프라인을 생성하거나 편집할 때 CloudFormation 스택 세트 작업 공급자를 선택할 수 있습니다.
- CloudFormationStackInstances 작업은 CloudFormationStackSet 작업의 변경 내용을 지 정된 인스턴스에 배포하고, 새 스택 인스턴스를 만들고, 지정된 인스턴스에 대한 파라미터 재정의를 정의합니다. 콘솔에서 파이프라인을 생성하거나 편집할 때 CloudFormation Stack Instances 작업 공 급자를 선택할 수 있습니다.

이러한 작업을 사용하여 대상 AWS 계정 또는 대상 AWS Organizations 조직 단위 IDs.

Note

대상 AWS Organizations 계정 또는 조직 단위 IDs에 배포하고 서비스 관리형 권한 모델을 사 용하려면 AWS CloudFormation StackSets와 AWS Organizations 간에 신뢰할 수 있는 액세스 를 활성화해야 합니다. 자세한 내용은 <u>AWS CloudFormation Stacksets에서 신뢰할 수 있는 액</u> 세스 활성화를 참조하세요.

주제

- AWS CloudFormation StackSets 작업 작동 방식
- 파이프라인에서 StackSets 작업을 구성하는 방법
- CloudFormationStackSet 작업
- CloudFormationStackInstances 작업
- 서비스 역할 권한: CloudFormationStackSet 작업
- 서비스 역할 권한: CloudFormationStackInstances 작업
- 스택 세트 작업에 대한 권한 모델
- 템플릿 파라미터 데이터 유형
- 다음 사항도 참조하세요.

AWS CloudFormation StackSets 작업 작동 방식

CloudFormationStackSet 작업은 작업이 처음 실행되는지 여부에 따라 리소스를 생성하거나 업데 이트합니다.

CloudFormationStackSet 작업은 스택 세트를 생성하거나 업데이트하고 해당 변경 내용을 지정된 인스턴스에 배포합니다.

Note

이 작업을 사용하여 스택 인스턴스 추가를 포함하는 업데이트를 수행하면 새 인스턴스가 먼저 배포되고 업데이트가 마지막으로 완료됩니다. 새 인스턴스는 먼저 이전 버전을 받은 다음 모든 인스턴스에 업데이트가 적용됩니다.

- 생성: 지정된 인스턴스가 없고 스택 세트가 존재하지 않는 경우 CloudFormationStackSet 작업을 수 행하면 인스턴스를 생성하지 않고 스택 세트를 생성합니다.
- 업데이트: 이미 생성된 스택 세트에 대해 CloudFormationStackSet 작업을 실행하면 해당 작업이 스택 세트를 업데이트합니다. 지정된 인스턴스가 없고 스택 세트가 이미 있는 경우 모든 인스턴스 가 업데이트됩니다. 이 작업을 사용하여 특정 인스턴스를 업데이트하면 나머지 모든 인스턴스가 OUTDATED 상태로 전환됩니다.

CloudFormationStackSet 작업을 사용하여 다음과 같은 방법으로 스택 세트를 업데이트할 수 있습니다.

• 일부 또는 모든 인스턴스에서 템플릿을 업데이트합니다.

- 일부 또는 모든 인스턴스의 파라미터를 업데이트합니다.
- 스택 세트의 실행 역할을 업데이트합니다(관리자 역할에 지정된 실행 역할과 일치해야 함).
- 권한 모델을 변경합니다(인스턴스가 생성되지 않은 경우에만 해당).
- 스택 세트 권한 모델이 Service Managed인 경우 AutoDeployment를 활성화/비활성화합니다.
- 스택 세트 권한 모델이 Service Managed인 경우 멤버 계정에서 위임된 관리자 역할을 합니다.
- 관리자 역할을 업데이트합니다.
- 스택 세트에 대한 설명을 업데이트합니다.
- 스택 세트 업데이트에 배포 대상을 추가하여 새 스택 인스턴스를 생성합니다.

CloudFormationStackInstances 작업을 수행하면 새 스택 인스턴스가 생성되거나 오래된 스택 인스턴스가 업데이트됩니다. 스택 세트를 업데이트하면 인스턴스가 만료되지만, 스택 세트 내의 모든 인스턴스가 업데이트되지는 않습니다.

- 생성: 스택이 이미 있는 경우 CloudFormationStackInstances 작업은 인스턴스만 업데이트하고 스택 인스턴스는 생성하지 않습니다.
- 업데이트: CloudFormationStackSet 작업이 수행된 후 템플릿 또는 파라미터가 일부 인스 턴스에서만 업데이트된 경우 나머지는 OUTDATED로 표시됩니다. 이후 파이프라인 단계에서 CloudFormationStackInstances는 스택의 나머지 인스턴스를 웨이브 형태로 업데이트하여 모 든 인스턴스가 CURRENT로 표시되도록 합니다. 이 작업을 사용하여 인스턴스를 추가하거나 새 인스 턴스 또는 기존 인스턴스의 파라미터를 재정의할 수도 있습니다.

업데이트의 일환으로 CloudFormationStackSet 및 CloudFormationStackInstances 작업은 새 배포 대상을 지정하여 새 스택 인스턴스를 만들 수 있습니다.

업데이트의 일부로, CloudFormationStackSet 및 CloudFormationStackInstances 작업은 스택 세트, 인스턴스 또는 리소스를 삭제하지 않습니다. 액션이 스택을 업데이트하지만 업데이트할 모 든 인스턴스를 지정하지 않는 경우 업데이트하도록 지정되지 않은 인스턴스는 업데이트에서 제거되고 OUTDATED 상태로 설정됩니다.

배포 중에 스택 인스턴스에는 인스턴스에 대한 배포가 실패한 경우 OUTDATED 상태가 표시될 수도 있 습니다.

파이프라인에서 StackSets 작업을 구성하는 방법

가장 좋은 방법은 스택 세트가 생성되고 처음에 하위 집합 또는 단일 인스턴스에 배포되도록 파이프라 인을 구성하는 것입니다. 배포를 테스트하고 생성된 스택 세트를 확인한 후 나머지 인스턴스가 생성되 고 업데이트되도록 CloudFormationStackInstances 작업을 추가하세요.

콘솔 또는 CLI를 사용하여 다음과 같이 권장 파이프라인 구조를 생성합니다.

- 1. 소스 작업(필수)을 사용하고 CloudFormationStackSet 작업을 배포 작업으로 사용하여 파이프 라인을 생성합니다. 파이프라인을 실행합니다.
- 파이프라인이 처음 실행되면 CloudFormationStackSet 작업을 통해 스택 세트와 하나 이상의 초기 인스턴스가 생성됩니다. 스택 세트 생성을 확인하고 초기 인스턴스로의 배포를 검토하세요. 예 를 들어, us-east-1이 지정된 리전인 계정 Account-A에 대한 초기 스택 세트 생성의 경우 스택 세 트를 사용하여 스택 인스턴스가 생성됩니다.

스택 인스턴스	리전	상태 표시기
StackInstanceID-1	us-east-1	CURRENT

3. 파이프라인을 편집하여 지정한 대상에 대한 스택 인스턴스를 생성/업데이트하는 두 번째 배포 작업으로 CloudFormationStackInstances를 추가하세요. 예를 들어, us-east-2 및 eucentral-1 리전이 지정된 계정 Account-A에 대한 스택 인스턴스 생성의 경우 나머지 스택 인스 턴스가 생성되고 초기 인스턴스는 다음과 같이 업데이트된 상태로 유지됩니다.

스택 인스턴스	리전	상태 표시기
StackInstanceID-1	us-east-1	CURRENT
StackInstanceID-2	us-east-2	CURRENT
StackInstanceID-3	eu-central-1	CURRENT

4. 필요에 따라 파이프라인을 실행하여 스택 세트를 업데이트하고 스택 인스턴스를 업데이트하거나 생성합니다.

작업 구성에서 배포 대상을 제거한 스택 업데이트를 시작하면 업데이트하도록 지정되지 않은 스택 인 스턴스가 배포에서 제거되고 OUTDATED 상태로 이동합니다. 예를 들어 작업 구성에서 us-east-2 리전이 제거된 계정 Account-A에 대한 스택 인스턴스 업데이트의 경우 다음과 같이 나머지 스택 인 스턴스가 생성되고 제거된 인스턴스는 OUTDATE로 설정됩니다.

스택 인스턴스	리전	상태 표시기
StackInstanceID-1	us-east-1	CURRENT
StackInstanceID-2	us-east-2	OUTDATED
StackInstanceID-3	eu-central-1	CURRENT

스택 세트 배포 모범 사례에 대한 자세한 내용은AWS CloudFormation 사용 설명서의 StackSets <u>모범</u> 사례를 참조하세요.

CloudFormationStackSet 작업

이 작업은 파이프라인 소스 위치에 저장된 템플릿에서 스택 세트를 생성하거나 업데이트합니다.

스택 세트를 정의하면 구성 파라미터에서 지정된 대상 계정 및 리전에서 스택을 생성, 업데이트 또는 삭제할 수 있습니다. 스택을 생성, 업데이트 또는 삭제하면 작업이 수행될 리전의 순서, 스택 작업이 중 단되는 내결함성 비율, 스택에 대해 작업이 동시에 수행될 수 있는 계정 수 등의 기타 기본 설정도 지정 할 수 있습니다.

스택 세트는 리전 리소스입니다. 한 AWS 리전에서 스택 세트를 생성하는 경우 다른 리전에서 스택 세 트에 액세스할 수 없습니다.

이 작업을 스택 세트에 대한 업데이트 작업으로 사용하는 경우 하나 이상의 스택 인스턴스를 배포하지 않으면 스택을 업데이트할 수 없습니다.

주제

- <u>작업 유형</u>
- 구성 파라미터
- 입력 아티팩트
- 출력 아티팩트
- <u>출력 변수</u>
- CloudFormationStackSet 작업 구성 예시
작업 유형

- 범주: Deploy
- 소유자: AWS
- 공급자: CloudFormationStackSet
- 버전:1

구성 파라미터

StackSetName

필수 여부: 예

스택 세트와 연결할 이름입니다. 이 이름은 생성한 리전 내에서 고유해야 합니다.

이름에는 영숫자 및 하이픈 문자만 포함될 수 있습니다. 알파벳 문자로 시작해야 하고 128자 이하 여야 합니다.

설명

필수 여부: 아니요

스택 세트에 대한 설명입니다. 이를 사용하여 스택 세트의 용도나 기타 관련 정보를 설명할 수 있습 니다.

TemplatePath

필수 여부: 예

스택 세트의 리소스를 정의하는 템플릿의 위치입니다. 이는 최대 크기가 460,800바이트인 템플릿 을 가리켜야 합니다.

소스 아티팩트 이름 및 템플릿 파일의 경로를 다음 예제와 같이 "InputArtifactName::TemplateFileName" 형식으로 입력합니다.

SourceArtifact::template.txt

파라미터

필수 여부: 아니요

CloudFormationStackSet 작업

배포 중에 업데이트되는 스택 세트의 템플릿 파라미터 목록입니다.

파라미터를 리터럴 목록 또는 파일 경로로 제공할 수 있습니다.

• 다음과 같은 간편 구문 형식

(ParameterKey=string, ParameterValue=string, UsePreviousValue=boolean, Resolved' ParameterKey=string, ParameterValue=string, UsePreviousValue=boolean, ResolvedV 으로 파라미터를 입력할 수 있습니다. 이러한 데이터 형식에 대한 자세한 내용은 <u>템플릿 파라미</u> 터 데이터 유형 단원을 참조하세요.

다음 예제는 amzn-s3-demo-source-bucket 값의 BucketName이라는 이름의 파라미터를 보여줍니다.

ParameterKey=BucketName, ParameterValue=amzn-s3-demo-source-bucket

다음 예제는 여러 파라미터가 있는 항목을 보여줍니다.

```
ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket
ParameterKey=Asset1,ParameterValue=true
ParameterKey=Asset2,ParameterValue=true
```

• 다음 예제와 같이 "InputArtifactName::ParametersFileName" 형식에 입력된 템플릿 파 라미터 재지정 목록이 포함된 파일 위치를 입력할 수 있습니다.

SourceArtifact::parameters.txt

다음 예제에서는 parameters.txt에 대한 파일 콘텐츠를 보여줍니다.

기능

필수 여부: 아니요

템플릿의 리소스 유형에 따라 템플릿이 리소스를 생성하고 업데이트할 수 있음을 나타냅니다.

스택 템플릿에 IAM 리소스가 있거나 매크로가 포함된 템플릿에서 직접 스택을 만들 경우 이 속성을 사용해야 합니다. 이러한 방식으로 AWS CloudFormation 작업이 성공적으로 작동하려면 다음 기능 중 하나를 사용해야 합니다.

- CAPABILITY_IAM
- CAPABILITY_NAMED_IAM

기능 사이에 공백 없이 쉼표를 사용하여 둘 이상의 기능을 지정할 수 있습니다. <u>CloudFormationStackSet 작업 구성 예시</u>의 예제는 여러 기능이 있는 항목을 보여줍니다.

PermissionModel

필수 여부: 아니요

IAM 역할을 생성하고 관리하는 방법을 결정합니다. 필드를 지정하지 않으면 기본값이 사용됩니다. 자세한 내용은 스택 세트 작업에 대한 권한 모델 단원을 참조하세요.

유효한 값은 다음과 같습니다.

- SELF_MANAGED(기본값): 대상 계정에 배포하려면 관리자 및 실행 역할을 생성해야 합니다.
- SERVICE_MANAGED: AWS CloudFormation StackSets는 AWS Organizations에서 관리하는 계 정에 배포하는 데 필요한 IAM 역할을 자동으로 생성합니다. 이를 위해서는 계정이 조직 멤버이어 야 합니다.

Note

이 파라미터는 스택 세트에 스택 인스턴스가 없는 경우에만 변경할 수 있습니다.

AdministrationRoleArn

Note

AWS CloudFormation StackSets는 여러 계정에서 작업을 수행하기 때문에 스택 세트를 생성하기 전에 해당 계정에서 필요한 권한을 정의해야 합니다.

필수 여부: 아니요

Note

이 파라미터는 SELF_MANAGED 권한 모델의 경우 선택 사항이며 SERVICE_MANAGED 권한 모델에는 사용되지 않습니다.

스택 세트 작업을 수행하는 데 사용되는 관리자 계정의 IAM 역할 ARN입니다.

이름에는 영숫자 문자, _+=,.@- 문자 중 하나를 사용할 수 있으며 공백은 사용할 수 없습니다. 이름 은 대/소문자를 구분하지 않습니다. 이 역할 이름은 최소 길이 20자, 최대 길이 2,048자이어야 합니 다. 역할 이름은 계정 내에서 고유해야 합니다. 여기에 지정된 역할 이름은 기존 역할 이름이어야 합 니다. 역할 이름을 지정하지 않으면 AWSCloudFormationStackSetAdministrationRole로 설정됩니 다. ServiceManaged를 지정하는 경우 역할 이름을 정의해서는 안 됩니다.

ExecutionRoleName

Note

AWS CloudFormation StackSets는 여러 계정에서 작업을 수행하기 때문에 스택 세트를 생성하기 전에 해당 계정에서 필요한 권한을 정의해야 합니다.

필수 여부: 아니요

Note

이 파라미터는 SELF_MANAGED 권한 모델의 경우 선택 사항이며 SERVICE_MANAGED 권한 모델에는 사용되지 않습니다.

스택 세트 작업을 수행하는 데 사용되는 대상 계정의 IAM 역할 이름입니다. 이름에는 영숫자 문 자, _+=,.@- 문자 중 하나를 사용할 수 있으며 공백은 사용할 수 없습니다. 이름은 대/소문자를 구 분하지 않습니다. 이 역할 이름은 최소 길이 1자, 최대 길이 64자이어야 합니다. 역할 이름은 계 정 내에서 고유해야 합니다. 여기에 지정된 역할 이름은 기존 역할 이름이어야 합니다. 사용자 정의된 실행 역할을 사용하는 경우 이 역할을 지정하지 마십시오. 역할 이름을 지정하지 않으면 AWSCloudFormationStackSetExecutionRole로 설정됩니다. Service_Managed를 true로 설 정하는 경우 역할 이름을 정의해서는 안 됩니다. OrganizationsAutoDeployment

필수 여부: 아니요

Note

이 파라미터는 SERVICE_MANAGED 권한 모델의 경우 선택 사항이며 SELF_MANAGED 권한 모델에는 사용되지 않습니다.

대상 조직 또는 조직 단위(OU)에 AWS 추가된 Organizations 계정에 AWS CloudFormation StackSets가 자동으로 배포되는지 여부를 설명합니다. OrganizationsAutoDeployment가 지 정된 경우 DeploymentTargets 및 Regions를 지정하지 마십시오.

Note

OrganizationsAutoDeployment에 입력이 제공되지 않은 경우 기본값은 Disabled입니다.

유효한 값은 다음과 같습니다.

• Enabled. 필수 여부: 아니요.

StackSets는 지정된 리전의 대상 조직 또는 조직 단위(OU)에 추가된 AWS Organizations 계정에 자동으로 추가 스택 인스턴스를 배포합니다. 계정이 대상 조직 또는 OU에서 제거되면 AWS CloudFormation StackSets는 지정된 리전의 계정에서 스택 인스턴스를 삭제합니다.

• Disabled. 필수 여부: 아니요.

StackSets는 지정된 리전의 대상 AWS 조직 또는 조직 단위(OU)에 추가된 Organizations 계정에 추가 스택 인스턴스를 자동으로 배포하지 않습니다.

• EnabledWithStackRetention. 필수 여부: 아니요.

대상 조직 또는 OU에서 계정이 제거될 때 스택 리소스가 유지됩니다.

DeploymentTargets

필수 여부: 아니요

CloudFormationStackSet 작업

Note

SERVICE_MANAGED 권한 모델의 경우 배포 대상에 조직 루트 ID 또는 조직 단위 ID를 제 공할 수 있습니다. SELF_MANAGED 권한 모델의 경우 계정만 제공할 수 있습니다.

Note

이 파라미터를 선택한 경우 리전도 선택해야 합니다.

스택 세트 인스턴스를 생성/업데이트해야 하는 AWS 계정 또는 조직 단위 IDs의 목록입니다.

• 계정:

계정을 리터럴 목록 또는 파일 경로로 제공할 수 있습니다.

• 리터럴: 다음 예제와 같이 간편 구문 형식 account_ID, account_ID로 파라미터를 입력합니다.

111111222222,333333444444

• 파일 경로: 스택 세트 인스턴스를 생성/업데이트하고 형식으로 입력해야 하는 AWS 계정 목록 이 포함된 파일의 위치입니다InputArtifactName::AccountsFileName. 파일 경로를 사 용하여 계정 또는 OrganizationalUnitIds를 지정하는 경우 파일 형식은 다음 예제와 같이 JSON 이어야 합니다.

SourceArtifact::accounts.txt

다음 예제에서는 accounts.txt에 대한 파일 콘텐츠를 보여줍니다.

```
[
"111111222222"
]
```

다음 예제는 두 개 이상의 계정을 나열할 때 accounts.txt의 파일 내용을 보여줍니다.

["11111222222","333333444444"] OrganizationalUnitIds:

Note

이 파라미터는 SERVICE_MANAGED 권한 모델의 경우 선택 사항이며 SELF_MANAGED 권한 모델에는 사용되지 않습니다. OrganizationsAutoDeployment를 선택한 경우에는 이 옵션을 사용하지 마십시오.

연결된 스택 인스턴스를 업데이트할 AWS 조직 단위입니다.

조직 단위 ID를 리터럴 목록 또는 파일 경로로 제공할 수 있습니다.

• 리터럴: 다음 예제와 같이 쉼표로 구분하여 문자열 배열을 입력합니다.

ou-examplerootid111-exampleouid111,ou-examplerootid222-exampleouid222

• 파일 경로: 스택 세트 인스턴스를 만들거나 업데이트할 OrganizationalUnitIds 목록이 들어 있 는 파일의 위치입니다. 파일 경로를 사용하여 계정 또는 OrganizationalUnitIds를 지정하는 경 우 파일 형식은 다음 예제와 같이 JSON이어야 합니다.

파일 경로를 InputArtifactName::OrganizationalUnitIdsFileName 형식으로 입력 합니다.

SourceArtifact::OU-IDs.txt

다음 예제에서는 0U-IDs.txt에 대한 파일 콘텐츠를 보여줍니다.

```
[ "ou-examplerootid111-exampleouid111", "ou-examplerootid222-exampleouid222" ]
```

리전

필수 여부: 아니요

Note

이 파라미터를 선택한 경우 DeploymentTargets도 선택해야 합니다.

스택 세트 인스턴스가 생성되거나 업데이트되는 AWS 리전 목록입니다. 리전은 입력한 순서대로 업데이트됩니다.

다음 예제와 Region1, Region2같이 형식으로 유효한 AWS 리전 목록을 입력합니다.

```
us-west-2,us-east-1
```

FailureTolerancePercentage

필수 여부: 아니요

가 해당 리전에서 작업을 AWS CloudFormation 중지하기 전에이 스택 작업이 실패할 수 있 는 리전당 계정의 비율입니다. 리전에서 작업이 중지되면 후속 리전에서 작업을 시도 AWS CloudFormation 하지 않습니다. 지정된 백분율을 기준으로 계정 수를 계산할 때는 다음 정수로 반 올 AWS CloudFormation 림합니다.

MaxConcurrentPercentage

필수 여부: 아니요

한 번에 이 작업을 수행할 최대 계정 백분율입니다. 지정된 백분율을 기준으로 계정 수를 계산할 때 는 다음 정수로 반올 AWS CloudFormation 림합니다. 반내림으로 인해 0이 되는 경우는 대신 숫자 를 1로 AWS CloudFormation 설정합니다. 이 설정을 사용하여 최대값을 지정하지만 대규모 배포의 경우 동시에 실행된 실제 계정 수는 서비스 조절로 인해 낮아질 수 있습니다.

RegionConcurrencyType

필수 여부: 아니요

리전 동시성 배포 파라미터를 구성하여 스택 세트를 AWS 리전 에 순차적으로 배포할지 병렬로 배 포할지 지정할 수 있습니다. 여러에 스택을 병렬 AWS 리전 로 배포하도록 리전 동시성을 지정하면 전체 배포 시간이 빨라질 수 있습니다.

- 병렬: 리전의 배포 실패가 지정된 내결함성을 초과하지 않는 한 스택 세트 배포가 동시에 수행됩니다.
- 순차: 리전의 배포 실패가 지정된 내결함성을 초과하지 않는 한 스택 세트 배포가 동시에 수행됩니다. 기본 선택 항목은 [순차(Sequential)] 배포입니다.

ConcurrencyMode

필수 여부: 아니요

동시성 모드 사용하면 스택 세트 작업 중에 동시성 수준이 작동하는 방식을 선택할 수 있습니다(엄 격한 내결함성 또는 소프트 내결함성 포함). 엄격한 내결함성을 사용하면 스택 세트 작업이 실패할 때마다 동시성이 감소하므로 배포 속도가 느려집니다. 소프트 내결함성은 AWS CloudFormation 안 전 기능을 계속 활용하면서 배포 속도의 우선순위를 지정합니다.

- STRICT_FAILURE_TOLERANCE: 이 옵션은 실패한 계정 수가 특정 내결함성을 초과하지 않도록 동시성 수준을 동적으로 낮춥니다. 이는 기본 설정 동작입니다.
- SOFT_FAILURE_TOLERANCE: 이 옵션은 내결함성을 실제 동시성에서 분리합니다. 이렇게 하면 실패 횟수에 관계없이 설정된 동시성 수준에서 스택 세트 작업을 실행할 수 있습니다.

CallAs

필수 여부: 아니요

Note

이 파라미터는 SERVICE_MANAGED 권한 모델의 경우 선택 사항이며 SELF_MANAGED 권한 모델에는 사용되지 않습니다.

조직의 관리 계정에서 역할을 하는지 아니면 멤버 계정에서 위임된 관리자 역할을 하는지 지정합니 다.

Note

이 파라미터가 DELEGATED_ADMIN으로 설정된 경우 파이프라인 IAM 역할에 organizations:ListDelegatedAdministrators 권한이 있는지 확인합니다. 그렇 지 않으면 실행 중에 다음과 유사한 오류와 함께 작업이 실패합니다. Account used is not a delegated administrator

- SELF: 스택 세트 배포는 관리 계정에 로그인한 상태에서 서비스 관리형 권한을 사용합니다.
- DELEGATED_ADMIN: 스택 세트 배포는 위임 관리자 계정에 로그인한 상태에서 서비스 관리형 권 한을 사용합니다.

입력 아티팩트

CloudFormationStackSet 작업에 설정된 스택용 템플릿이 포함된 입력 아티팩트를 하나 이상 포함 해야 합니다. 배포 대상, 계정, 파라미터 목록에 더 많은 입력 아티팩트를 포함할 수 있습니다.

• 아티팩트 수:1 to 3

CloudFormationStackSet 작업

- 설명: 다음을 제공할 아티팩트를 포함할 수 있습니다.
 - 스택 템플릿 파일. TemplatePath 파라미터를 참조하십시오.
 - 파라미터 파일. Parameters 파라미터를 참조하십시오.
 - 계정 파일. DeploymentTargets 파라미터를 참조하십시오.

출력 아티팩트

- 아티팩트 수: 0
- 설명: 이 작업 유형에는 출력 아티팩트가 적용되지 않습니다.

출력 변수

이 작업을 구성하면 파이프라인에서 다운스트림 작업의 작업 구성에서 참조할 수 있는 변수가 생성됩 니다. 이러한 변수를 다운스트림 작업 구성에서 사용할 수 있도록 네임스페이스를 사용하여 작업을 구 성합니다.

- StackSetId: 스택 세트의 ID입니다.
- OperationId: 스택 세트 작업의 ID입니다.

자세한 내용은 변수 참조 단원을 참조하십시오.

CloudFormationStackSet 작업 구성 예시

다음 예시는 CloudFormationStackSet 작업에 대한 작업 구성을 보여줍니다.

자체 관리형 권한 모델의 예

다음 예제는 입력한 배포 대상이 AWS 계정 ID인 CloudFormationStackSet 작업을 보여줍니다.

YAML

```
Name: CreateStackSet
ActionTypeId:
  Category: Deploy
  Owner: AWS
  Provider: CloudFormationStackSet
  Version: '1'
RunOrder: 1
```

```
Configuration:

DeploymentTargets: '111111222222'

FailureTolerancePercentage: '20'

MaxConcurrentPercentage: '25'

PermissionModel: SELF_MANAGED

Regions: us-east-1

StackSetName: my-stackset

TemplatePath: 'SourceArtifact::template.json'

OutputArtifacts: []

InputArtifacts:

- Name: SourceArtifact

Region: us-west-2

Namespace: DeployVariables
```

JSON

```
{
    "Name": "CreateStackSet",
    "ActionTypeId": {
        "Category": "Deploy",
        "Owner": "AWS",
        "Provider": "CloudFormationStackSet",
        "Version": "1"
    },
    "RunOrder": 1,
    "Configuration": {
        "DeploymentTargets": "111111222222",
        "FailureTolerancePercentage": "20",
        "MaxConcurrentPercentage": "25",
        "PermissionModel": "SELF_MANAGED",
        "Regions": "us-east-1",
        "StackSetName": "my-stackset",
        "TemplatePath": "SourceArtifact::template.json"
    },
    "OutputArtifacts": [],
    "InputArtifacts": [
        {
            "Name": "SourceArtifact"
        }
    ],
    "Region": "us-west-2",
    "Namespace": "DeployVariables"
}
```

서비스 관리형 권한 모델의 예

다음 예제는 스택 보존을 통해 AWS Organizations에 자동 배포하는 옵션이 활성화된 서비스 관리형 권한 모델에 대한 CloudFormationStackSet 작업을 보여줍니다.

YAML

```
Name: Deploy
ActionTypeId:
 Category: Deploy
 Owner: AWS
  Provider: CloudFormationStackSet
  Version: '1'
RunOrder: 1
Configuration:
  Capabilities: 'CAPABILITY_IAM, CAPABILITY_NAMED_IAM'
 OrganizationsAutoDeployment: EnabledWithStackRetention
  PermissionModel: SERVICE_MANAGED
  StackSetName: stacks-orgs
  TemplatePath: 'SourceArtifact::template.json'
OutputArtifacts: []
InputArtifacts:
  - Name: SourceArtifact
Region: eu-central-1
Namespace: DeployVariables
```

JSON

```
{
    "Name": "Deploy",
    "ActionTypeId": {
        "Category": "Deploy",
        "Owner": "AWS",
        "Provider": "CloudFormationStackSet",
        "Version": "1"
    },
    "RunOrder": 1,
    "Configuration": {
        "Capabilities": "CAPABILITY_IAM,CAPABILITY_NAMED_IAM",
        "OrganizationsAutoDeployment": "EnabledWithStackRetention",
        "PermissionModel": "SERVICE_MANAGED",
        "StackSetName": "stacks-orgs",
        "TemplatePath": "SourceArtifact::template.json"
```

```
},
  "OutputArtifacts": [],
  "InputArtifacts": [
        {
            "Name": "SourceArtifact"
        }
    ],
    "Region": "eu-central-1",
    "Namespace": "DeployVariables"
}
```

CloudFormationStackInstances 작업

이 작업은 새 인스턴스를 생성하고 지정된 인스턴스에 스택 세트를 배포합니다. 스택 인스턴스는 리전 내 대상 계정에서 스택에 대한 참조입니다. 스택 인스턴스는 스택 없이 존재할 수 있습니다. 예를 들어, 스택 생성이 성공적이지 않은 경우 스택 인스턴스는 스택 생성 실패에 대한 이유를 표시합니다. 스택 인스턴스는 하나의 스택 세트에만 연결됩니다.

스택 세트를 처음 생성한 후에는 CloudFormationStackInstances를 사용하여 새 스택 인스턴스 를 추가할 수 있습니다. 스택 세트 인스턴스 생성 또는 업데이트 작업 중에 스택 인스턴스 수준에서 템 플릿 파라미터 값을 재정의할 수 있습니다.

각 스택 세트에는 템플릿 하나와 템플릿 파라미터 세트가 있습니다. 템플릿 또는 템플릿 파라미터를 업 데이트하면 전체 세트에 대해 업데이트됩니다. 그러면 변경 내용이 해당 인스턴스에 배포될 때까지 모 든 인스턴스 상태가 OUTDATED로 설정됩니다.

예를 들어 템플릿에 값이 prod인 stage의 파라미터가 포함된 경우 특정 인스턴스의 파라미터 값을 재정의하려면 해당 파라미터의 값을 beta 또는 gamma로 재정의할 수 있습니다.

주제

- 작업 유형
- <u>구성 파라미터</u>
- 입력 아티팩트
- 출력 아티팩트
- <u>출력 변수</u>
- <u>예제 작업 구성</u>

작업 유형

- 범주: Deploy
- 소유자: AWS
- 공급자: CloudFormationStackInstances
- 버전:1

구성 파라미터

StackSetName

필수 여부: 예

스택 세트와 연결할 이름입니다. 이 이름은 생성한 리전 내에서 고유해야 합니다.

이름에는 영숫자 및 하이픈 문자만 포함될 수 있습니다. 알파벳 문자로 시작해야 하고 128자 이하 여야 합니다.

DeploymentTargets

필수 여부: 아니요

1 Note

SERVICE_MANAGED 권한 모델의 경우 배포 대상에 조직 루트 ID 또는 조직 단위 ID를 제 공할 수 있습니다. SELF_MANAGED 권한 모델의 경우 계정만 제공할 수 있습니다.

Note

이 파라미터를 선택한 경우 리전도 선택해야 합니다.

스택 세트 인스턴스를 생성/업데이트해야 하는 AWS 계정 또는 조직 단위 IDs의 목록입니다.

• 계정:

계정을 리터럴 목록 또는 파일 경로로 제공할 수 있습니다.

• 리터럴: 다음 예제와 같이 간편 구문 형식 account_ID, account_ID로 파라미터를 입력합니다.

```
111111222222,333333444444
```

• 파일 경로: 스택 세트 인스턴스를 생성/업데이트하고 형식으로 입력해야 하는 AWS 계정 목록 이 포함된 파일의 위치입니다InputArtifactName::AccountsFileName. 파일 경로를 사 용하여 계정 또는 OrganizationalUnitIds를 지정하는 경우 파일 형식은 다음 예제와 같이 JSON 이어야 합니다.

```
SourceArtifact::accounts.txt
```

다음 예제에서는 accounts.txt에 대한 파일 콘텐츠를 보여줍니다.

```
[
"11111222222"
]
```

다음 예제는 두 개 이상의 계정을 나열할 때 accounts.txt의 파일 내용을 보여줍니다.

```
[
"111111222222","333333444444"
]
```

OrganizationalUnitIds:

```
i) Note이 파라미터는 SERVICE_MA
```

이 파라미터는 SERVICE_MANAGED 권한 모델의 경우 선택 사항이며 SELF_MANAGED 권한 모델에는 사용되지 않습니다. OrganizationsAutoDeployment를 선택한 경우에는 이 옵션을 사용하지 마십시오.

연결된 스택 인스턴스를 업데이트할 AWS 조직 단위입니다.

조직 단위 ID를 리터럴 목록 또는 파일 경로로 제공할 수 있습니다.

• 리터럴: 다음 예제와 같이 쉼표로 구분하여 문자열 배열을 입력합니다.

ou-examplerootid111-exampleouid111,ou-examplerootid222-exampleouid222

• 파일 경로: 스택 세트 인스턴스를 만들거나 업데이트할 OrganizationalUnitIds 목록이 들어 있 는 파일의 위치입니다. 파일 경로를 사용하여 계정 또는 OrganizationalUnitIds를 지정하는 경 우 파일 형식은 다음 예제와 같이 JSON이어야 합니다.

파일 경로를 InputArtifactName::OrganizationalUnitIdsFileName 형식으로 입력 합니다.

SourceArtifact::OU-IDs.txt

다음 예제에서는 0U-IDs.txt에 대한 파일 콘텐츠를 보여줍니다.

"ou-examplerootid111-exampleouid111", "ou-examplerootid222-exampleouid222"

리전

필수 여부: 예

Γ

]

Note

이 파라미터를 선택한 경우 DeploymentTargets도 선택해야 합니다.

스택 세트 인스턴스가 생성되거나 업데이트되는 AWS 리전 목록입니다. 리전은 입력한 순서대로 업데이트됩니다.

다음 예제와 Region1, Region2같이 형식으로 유효한 AWS 리전 목록을 입력합니다.

```
us-west-2,us-east-1
```

ParameterOverrides

필수 여부: 아니요

선택한 스택 인스턴스에서 재정의할 스택 세트 파라미터의 목록입니다. 재정의된 파라미터 값은 지 정된 계정 및 리전의 모든 스택 인스턴스에 적용됩니다.

파라미터를 리터럴 목록 또는 파일 경로로 제공할 수 있습니다.

• 다음과 같은 간편 구문 형식 (ParameterKey=string,ParameterValue=string,UsePreviousValue=boolean,Resolved) ParameterKey=string, ParameterValue=string, UsePreviousValue=boolean, ResolvedV 으로 파라미터를 입력할 수 있습니다. 이러한 데이터 형식에 대한 자세한 내용은 <u>템플릿 파라미</u> 터 데이터 유형 단원을 참조하세요.

다음 예제는 amzn-s3-demo-source-bucket 값의 BucketName이라는 이름의 파라미터를 보여줍니다.

ParameterKey=BucketName, ParameterValue=amzn-s3-demo-source-bucket

다음 예제는 여러 파라미터가 있는 항목을 보여줍니다.

ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket ParameterKey=Asset1,ParameterValue=true ParameterKey=Asset2,ParameterValue=true

• 다음 예제와 같이 InputArtifactName::ParameterOverridessFileName 형식에 입력된 템플릿 파라미터 재지정 목록이 포함된 파일 위치를 입력할 수 있습니다.

SourceArtifact::parameter-overrides.txt

다음 예제에서는 parameter-overrides.txt에 대한 파일 콘텐츠를 보여줍니다.

FailureTolerancePercentage

필수 여부: 아니요

가 해당 리전에서 작업을 AWS CloudFormation 중지하기 전에이 스택 작업이 실패할 수 있는 리전당 계정의 비율입니다. 리전에서 작업이 중지되면 후속 리전에서 작업을 시도 AWS

CloudFormation 하지 않습니다. 지정된 백분율을 기준으로 계정 수를 계산할 때 다음 정수로 반올 AWS CloudFormation 림합니다.

MaxConcurrentPercentage

필수 여부: 아니요

한 번에 이 작업을 수행할 최대 계정 백분율입니다. 지정된 백분율을 기준으로 계정 수를 계산할 때 는 다음 정수로 반올 AWS CloudFormation 림합니다. 반내림으로 인해 0이 되는 경우는 대신 숫자 를 1로 AWS CloudFormation 설정합니다. 최대값을 지정하지만 대규모 배포의 경우 동시에 실행된 실제 계정 수는 서비스 조절로 인해 낮아질 수 있습니다.

RegionConcurrencyType

필수 여부: 아니요

리전 동시성 배포 파라미터를 구성하여 스택 세트를 AWS 리전 에 순차적으로 배포할지 병렬로 배 포할지 지정할 수 있습니다. 여러에 스택을 병렬 AWS 리전 로 배포하도록 리전 동시성을 지정하면 전체 배포 시간이 빨라질 수 있습니다.

- 병렬: 리전의 배포 실패가 지정된 내결함성을 초과하지 않는 한 스택 세트 배포가 동시에 수행됩니다.
- 순차: 리전의 배포 실패가 지정된 내결함성을 초과하지 않는 한 스택 세트 배포가 동시에 수행됩니다. 기본 선택 항목은 [순차(Sequential)] 배포입니다.

ConcurrencyMode

필수 여부: 아니요

동시성 모드 사용하면 스택 세트 작업 중에 동시성 수준이 작동하는 방식을 선택할 수 있습니다(엄 격한 내결함성 또는 소프트 내결함성 포함). 엄격한 내결함성을 사용하면 스택 세트 작업이 실패할 때마다 동시성이 감소하므로 배포 속도가 느려집니다. 소프트 내결함성은 AWS CloudFormation 안 전 기능을 계속 활용하면서 배포 속도의 우선순위를 지정합니다.

- STRICT_FAILURE_TOLERANCE: 이 옵션은 실패한 계정 수가 특정 내결함성을 초과하지 않도록 동시성 수준을 동적으로 낮춥니다. 이는 기본 설정 동작입니다.
- SOFT_FAILURE_TOLERANCE: 이 옵션은 내결함성을 실제 동시성에서 분리합니다. 이렇게 하면 실패 횟수에 관계없이 설정된 동시성 수준에서 스택 세트 작업을 실행할 수 있습니다.

CallAs

필수 여부: 아니요

CloudFormationStackInstances 작업

Note

이 파라미터는 SERVICE_MANAGED 권한 모델의 경우 선택 사항이며 SELF_MANAGED 권한 모델에는 사용되지 않습니다.

조직의 관리 계정에서 역할을 하는지 아니면 멤버 계정에서 위임된 관리자 역할을 하는지 지정합니 다.

Note

이 파라미터가 DELEGATED_ADMIN으로 설정된 경우 파이프라인 IAM 역할에 organizations:ListDelegatedAdministrators 권한이 있는지 확인합니다. 그렇 지 않으면 실행 중에 다음과 유사한 오류와 함께 작업이 실패합니다. Account used is not a delegated administrator

- SELF: 스택 세트 배포는 관리 계정에 로그인한 상태에서 서비스 관리형 권한을 사용합니다.
- DELEGATED_ADMIN: 스택 세트 배포는 위임 관리자 계정에 로그인한 상태에서 서비스 관리형 권 한을 사용합니다.

입력 아티팩트

CloudFormationStackInstances는 배포 대상 및 파라미터를 나열하는 아티팩트를 포함할 수 있 습니다.

- 아티팩트 수: 0 to 2
- 설명: 선택적으로 스택 세트 작업은 다음의 용도로 아티팩트를 입력으로 수락합니다.
 - 사용할 파라미터 파일을 제공합니다. ParameterOverrides 파라미터를 참조하십시오.
 - 사용할 대상 계정 파일을 제공합니다. DeploymentTargets 파라미터를 참조하십시오.

출력 아티팩트

- 아티팩트 수: 0
- 설명: 이 작업 유형에는 출력 아티팩트가 적용되지 않습니다.

출력 변수

이 작업을 구성하면 파이프라인에서 다운스트림 작업의 작업 구성에서 참조할 수 있는 변수가 생성됩 니다. 이러한 변수를 다운스트림 작업 구성에서 사용할 수 있도록 네임스페이스를 사용하여 작업을 구 성합니다.

- StackSetId: 스택 세트의 ID입니다.
- OperationId: 스택 세트 작업의 ID입니다.

자세한 내용은 변수 참조 단원을 참조하십시오.

예제 작업 구성

다음 예시는 CloudFormationStackInstances 작업에 대한 작업 구성을 보여줍니다.

자체 관리형 권한 모델의 예

다음 예제는 입력한 배포 대상이 AWS 계정 ID 인 CloudFormationStackInstances 작업을 보여줍니 다111111222222.

YAML

```
Name: my-instances
ActionTypeId:
   Category: Deploy
   Owner: AWS
   Provider: CloudFormationStackInstances
   Version: '1'
RunOrder: 2
Configuration:
   DeploymentTargets: '111111222222'
   Regions: 'us-east-1,us-east-2,us-west-1,us-west-2'
   StackSetName: my-stackset
OutputArtifacts: []
InputArtifacts:
   - Name: SourceArtifact
Region: us-west-2
```

JSON

{

```
"Name": "my-instances",
    "ActionTypeId": {
        "Category": "Deploy",
        "Owner": "AWS",
        "Provider": "CloudFormationStackInstances",
        "Version": "1"
    },
    "RunOrder": 2,
    "Configuration": {
        "DeploymentTargets": "111111222222",
        "Regions": "us-east-1, us-east-2, us-west-1, us-west-2",
        "StackSetName": "my-stackset"
    },
    "OutputArtifacts": [],
    "InputArtifacts": [
        {
            "Name": "SourceArtifact"
        }
    ],
    "Region": "us-west-2"
}
```

서비스 관리형 권한 모델의 예

다음 예제는 배포 대상이 AWS Organizations 조직 단위 ID인 서비스 관리형 권한 모델에 대한 CloudFormationStackInstances 작업을 보여줍니다ou-1111-1example.

YAML

```
Name: Instances
ActionTypeId:
   Category: Deploy
   Owner: AWS
   Provider: CloudFormationStackInstances
   Version: '1'
RunOrder: 2
Configuration:
   DeploymentTargets: ou-1111-1example
   Regions: us-east-1
   StackSetName: my-stackset
OutputArtifacts: []
InputArtifacts:
```

```
- Name: SourceArtifact
Region: eu-central-1
```

JSON

```
{
    "Name": "Instances",
    "ActionTypeId": {
        "Category": "Deploy",
        "Owner": "AWS",
        "Provider": "CloudFormationStackInstances",
        "Version": "1"
    },
    "RunOrder": 2,
    "Configuration": {
        "DeploymentTargets": "ou-1111-1example",
        "Regions": "us-east-1",
        "StackSetName": "my-stackset"
    },
    "OutputArtifacts": [],
    "InputArtifacts": [
        {
            "Name": "SourceArtifact"
        }
    ],
    "Region": "eu-central-1"
}
```

서비스 역할 권한: CloudFormationStackSet 작업

AWS CloudFormation StackSets 작업의 경우 다음과 같은 최소 권한이 필요합니다.

CloudFormationStackSet 작업의 경우 다음을 정책 설명에 추가합니다.

```
{
    "Effect": "Allow",
    "Action": [
        "cloudformation:CreateStackSet",
        "cloudformation:UpdateStackSet",
        "cloudformation:CreateStackInstances",
        "cloudformation:DescribeStackSetOperation",
        "cloudformation:DescribeStackSet",
```

```
사용자 가이드
```

```
"cloudformation:ListStackInstances"
],
"Resource": "resource_ARN"
},
```

서비스 역할 권한: CloudFormationStackInstances 작업

CloudFormationStackInstances 작업의 경우 다음을 정책 설명에 추가합니다.

```
{
    "Effect": "Allow",
    "Action": [
        "cloudformation:CreateStackInstances",
        "cloudformation:DescribeStackSetOperation"
    ],
    "Resource": "resource_ARN"
},
```

스택 세트 작업에 대한 권한 모델

AWS CloudFormation StackSets는 여러 계정에서 작업을 수행하기 때문에 스택 세트를 생성하기 전에 해당 계정에서 필요한 권한을 정의해야 합니다. 자체 관리형 권한 또는 서비스 관리형 권한을 통해 권 한을 정의할 수 있습니다.

자체 관리형 권한을 사용하면 StackSets에 필요한 두 개의 IAM 역할을 생성할 수 있습니 다. 하나는 스택 세트를 정의하는 계정의 AWSCloudFormationStackSetAdministrationRole 과 같은 관리자 역할이고, 다른 하나는 스택 세트 인스턴스를 배포하는 각 계정에서 AWSCloudFormationStackSetExecutionRole과 같은 실행 역할을 생성합니다. StackSets는이 권한 모 델을 사용하여 사용자에게 IAM 역할을 생성할 권한이 있는 모든 AWS 계정에 배포할 수 있습니다. 자 세한 내용은 AWS CloudFormation 사용 설명서의 <u>자체 관리형 권한 부여</u>를 참조하세요.

Note

AWS CloudFormation StackSets는 여러 계정에서 작업을 수행하기 때문에 스택 세트를 생성 하기 전에 해당 계정에서 필요한 권한을 정의해야 합니다.

서비스 관리형 권한을 사용하면 AWS Organizations에서 관리하는 계정에 스택 인스턴스를 배포할 수 있습니다. 이 권한 모델을 사용하면 필요한 IAM 역할을 만들 필요가 없습니다. StackSets가 사용자를 대신하여 IAM 역할을 생성하기 때문입니다. 이 모델을 사용하면 나중에 조직에 추가되는 계정에 자동 배포를 활성화할 수도 있습니다. AWS CloudFormation 사용 설명서의 <u>AWS Organizations에서 신뢰할</u> 수 있는 액세스 활성화를 참조하세요.

템플릿 파라미터 데이터 유형

스택 세트 작업에 사용되는 템플릿 파라미터에는 다음 데이터 유형이 포함됩니다. 자세한 내용은 DescribeStackSet를 참조하세요.

ParameterKey

- 설명: 파라미터와 연결된 키입니다. 특정 파라미터에 키와 값을 지정하지 않으면 템플릿에 지정 된 기본값을 AWS CloudFormation 사용합니다.
- 예시

"ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket"

ParameterValue

- 설명: 파라미터와 연결된 입력 값입니다.
- 예시

"ParameterKey=BucketName, ParameterValue=amzn-s3-demo-source-bucket"

UsePreviousValue

- 스택 업데이트 중에는 스택에서 지정된 파라미터 키에 대해 사용 중인 기존 파라미터 값을 사용 하세요. true를 지정한 경우 파라미터 값을 지정하지 마십시오.
- 예시

"ParameterKey=Asset1,UsePreviousValue=true"

각 스택 세트에는 템플릿 하나와 템플릿 파라미터 세트가 있습니다. 템플릿 또는 템플릿 파라미터를 업 데이트하면 전체 세트에 대해 업데이트됩니다. 그러면 변경 내용이 해당 인스턴스에 배포될 때까지 모 든 인스턴스 상태가 OUTDATED로 설정됩니다.

예를 들어 템플릿에 값이 prod인 stage의 파라미터가 포함된 경우 특정 인스턴스의 파라미터 값을 재정의하려면 해당 파라미터의 값을 beta 또는 gamma로 재정의할 수 있습니다.

다음 사항도 참조하세요.

이 작업을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

- <u>파라미터 유형</u> AWS CloudFormation 사용 설명서의 이 참조 챕터에서는 CloudFormation 템플릿 파라미터에 대한 자세한 설명과 예제를 제공합니다.
- 모범 사례 스택 세트 배포 모범 사례에 대한 자세한 내용은AWS CloudFormation 사용 설 명서의 <u>https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/stacksets-</u> bestpractices.html을 참조하세요.
- <u>AWS CloudFormation API 참조</u> 스택 세트 작업에 사용되는 파라미터에 대한 자세한 내용은 AWS CloudFormation API 참조에서 다음 CloudFormation 작업을 참조할 수 있습니다.
 - CreateStackSet 작업은 스택 세트를 생성합니다.
 - <u>UpdateStackSet</u> 작업은 지정된 계정 및 리전의 스택 세트 및 관련 스택 인스턴스를 업데이트합니 다. 스택 세트를 업데이트하여 만든 스택 세트 작업이 완전히 또는 부분적으로, 지정된 실패 허용 오차 미만이나 초과로 실패하더라도 스택 세트는 이러한 변경 사항으로 업데이트됩니다. 지정된 스택 세트에 대한 후속 CreateStackInstances 호출에서는 업데이트된 스택 세트를 사용합니다.
 - CreateStackInstances 작업은 자체 관리형 권한 모델의 지정된 모든 계정 내 또는 서비스 관리형 권한 모델의 지정된 모든 배포 대상 내에 지정된 모든 리전에 대한 스택 인스턴스를 만듭니다. 이 작업으로 만든 인스턴스의 파라미터를 재정의할 수 있습니다. 인스턴스가 이미 존재하는 경우, CreateStackInstances는 동일한 입력 파라미터를 사용하여 UpdateStackInstances를 호출합니다. 이 작업을 사용하여 인스턴스를 생성할 때 다른 스택 인스턴스의 상태는 변경되지 않습니다.
 - UpdateStackInstances 작업은 자체 관리형 권한 모델의 지정된 모든 계정 내 또는 서비스 관리형 권한 모델의 지정된 모든 배포 대상 내에서 지정된 모든 리전에 대한 스택 세트를 통해 스택 인스 턴스를 최신 상태로 유지합니다. 이 작업으로 업데이트된 인스턴스의 파라미터를 재정의할 수 있 습니다. 이 작업을 사용하여 인스턴스의 하위 세트를 업데이트할 때 다른 스택 인스턴스의 상태는 변경되지 않습니다.
 - DescribeStackSetOperation 작업은 지정된 스택 세트 작업에 대한 설명을 반환합니다.
 - DescribeStackSet 작업은 지정된 스택 세트에 대한 설명을 반환합니다.

AWS CodeBuild 빌드 및 테스트 작업 참조

파이프라인의 일부로 빌드와 테스트를 실행할 수 있습니다. CodeBuild 빌드 또는 테스트 작업을 실 행하면 buildspec에 지정된 명령이 CodeBuild 컨테이너 내에서 실행됩니다. 명령을 실행하는 컨테 이너 내에서 CodeBuild 작업에 대해 입력 아티팩트로 지정된 모든 아티팩트를 사용할 수 있습니다. CodeBuild는 빌드 또는 테스트 작업을 제공할 수 있습니다. 자세한 내용은 <u>AWS CodeBuild 사용 설명</u> <u>서</u>를 참조하십시오. 콘솔에서 CodePipeline 마법사를 사용하여 빌드 프로젝트를 생성할 때 CodeBuild 빌드 프로젝트 는 소스 공급자가 CodePipeline임을 보여줍니다. CodeBuild 콘솔에서 빌드 프로젝트를 생성할 때 CodePipeline을 소스 공급자로 지정할 수 없지만, 파이프라인에 빌드 작업을 추가하면 CodeBuild 콘솔 에서 소스가 조정됩니다. 자세한 내용은AWS CodeBuild API 참조의 ProjectSource를 참조하세요.

주제

- <u>작업 유형</u>
- <u>구성 파라미터</u>
- 입력 아티팩트
- 출력 아티팩트
- <u>출력 변수</u>
- <u>서비스 역할 권한: CodeBuild 작업</u>
- 작업 선언(CodeBuild 예제)
- <u>다음 사항도 참조하세요.</u>

작업 유형

- 범주: Build 또는 Test
- 소유자: AWS
- 공급자: CodeBuild
- 버전:1

구성 파라미터

ProjectName

필수 여부: 예

ProjectName은 CodeBuild에서 빌드 프로젝트의 이름입니다.

PrimarySource

필수 항목 여부: 조건부

PrimarySource 파라미터 값은 작업에 대한 입력 아티팩트 중 하나의 이름이어야 합니다. CodeBuild는 buildspec 파일을 찾고이 아티팩트의 압축 해제된 버전이 포함된 디렉터리에서 buildspec 명령을 실행합니다. CodeBuild 작업에 대해 여러 입력 아티팩트가 지정되는 경우 이 파라미터는 필수 항목입니다. 작업 에 대해 소스 아티팩트가 하나만 있는 경우 해당 아티팩트의 기본값은 PrimarySource 아티팩트 입니다.

BatchEnabled

필수 여부: 아니요

BatchEnabled 파라미터의 부울 값을 사용하면 작업이 동일한 빌드 실행에서 여러 빌드를 실행할 수 있습니다.

이 옵션을 활성화하면 CombineArtifacts 옵션을 사용할 수 있습니다.

배치 빌드가 활성화된 파이프라인 예제는 <u>CodePipeline과 CodeBuild 및 배치 빌드 통합을 참조하</u> <u>세요</u>.

BuildspecOverride

필수 여부: 아니요

이 빌드에 대해서만 빌드 프로젝트에 정의된 최신 빌드를 재정의하는 인라인 buildspec 정의 또는 buildspec 파일 선언입니다. 프로젝트에 정의된 buildspec은 변경되지 않습니다.

이 값을 설정하면 다음 중 하나일 수 있습니다.

- 인라인 buildspec 정의입니다. 자세한 내용은 Buildspec 구문의 구문 참조를 참조하세요.
- 기본 제공 CODEBUILD_SRC_DIR 환경 변수의 값 또는 S3 버킷의 경로를 기준으로 대체 buildspec 파일의 경로입니다. 버킷은 빌드 프로젝트 AWS 리전 와 동일한에 있어야 합니다. ARN 을 사용하여 buildspec 파일을 지정합니다(예: arn:aws:s3:::my-codebuild-sample2/ buildspec.yml). 이 값이 제공되지 않거나 빈 문자열로 설정된 경우 소스 코드는 루트 디렉터 리에 buildspec 파일을 포함해야 합니다. 경로 추가에 대한 자세한 내용은 <u>Buildspec 파일 이름 및</u> 스토리지 위치를 참조하세요.

Note

이 속성을 사용하면 컨테이너에서 실행할 빌드 명령을 변경할 수 있으므로이 API를 호출하 고이 파라미터를 설정하는 기능을 가진 IAM 보안 주체가 기본 설정을 재정의할 수 있습니 다. 또한 소스 리포지토리의 파일 또는 Amazon S3 버킷과 같은 신뢰할 수 있는 buildspec 위치를 사용하는 것이 좋습니다.

CombineArtifacts

필수 여부: 아니요

CombineArtifacts 파라미터의 부울 값은 배치 빌드의 모든 빌드 아티팩트를 빌드 작업을 위한 단일 아티팩트 파일로 결합합니다.

이 옵션을 사용하려면 BatchEnabled 파라미터를 활성화해야 합니다.

EnvironmentVariables

필수 여부: 아니요

이 파라미터의 값은 파이프라인의 CodeBuild 작업에 대한 환경 변수를 설정하는 데 사용됩니다. EnvironmentVariables 파라미터 값은 환경 변수 객체의 JSON 배열 형식을 취합니다. <u>작업 선</u> 언(CodeBuild 예제)의 예제 파라미터를 참조하십시오.

각 객체에는 세 부분이 있으며 모두 문자열입니다.

- name: 환경 변수의 이름 또는 키입니다.
- value: 환경 변수의 값입니다. PARAMETER_STORE 또는 SECRETS_MANAGER 유형을 사용하는 경우이 값은 AWS Systems Manager Parameter Store에 이미 저장한 파라미터의 이름이거나 Secrets Manager에 AWS 이미 저장한 보안 암호여야 합니다.

Note

환경 변수를 사용하여 민감한 값, 특히 AWS 자격 증명을 저장하지 않는 것이 좋습니다. CodeBuild 콘솔 또는 AWS CLI를 사용하면 환경 변수가 일반 텍스트로 표시됩니다. 중요 한 값의 경우 대신 SECRETS_MANAGER 유형을 사용하는 것이 좋습니다.

• type: (선택 사항) 환경 변수의 유형입니다. 유효한 값은 PARAMETER_STORE, SECRETS_MANAGER또는 PLAINTEXT입니다. 지정하지 않으면 기본적으로 PLAINTEXT가 사용 됩니다.

1 Note

환경 변수 구성에 대해 name, value 및 type을 입력할 때, 특히 환경 변수에 CodePipeline 출력 변수 구문이 포함된 경우 구성 값 필드에 1000자 제한을 초과하지 마십시오. 이 제한 을 초과하면 확인 오류가 반환됩니다. 자세한 내용은 AWS CodeBuild API 참조의 <u>EnvironmentVariable</u>을 참조하세요. GitHub 브랜치 이름으로 확인되는 환경 변수를 포함하는 CodeBuild 작업의 예는 <u>예: CodeBuild 환경 변수와 함께</u> BranchName 변수 사용 섹션을 참조하세요.

입력 아티팩트

- 아티팩트 수:1 to 5
- 설명: CodeBuild는 buildspec 파일을 찾고 기본 소스 아티팩트의 디렉터리에서 buildspec 명령을 실 행합니다. 단일 입력 소스가 지정되거나 CodeBuild 작업에 대해 둘 이상의 입력 소스가 지정된 경우 단일 아티팩트 또는 여러 입력 소스의 경우 기본 아티팩트를 CodePipeline의 PrimarySource 작업 구성 파라미터를 사용하여 설정해야 합니다.

각 입력 아티팩트가 자체 디렉터리로 추출되며 이 위치가 환경 변수에 저장됩니다. 기본 소스 아티팩 트의 디렉터리는 \$CODEBUILD_SRC_DIR에서 사용할 수 있습니다. 다른 모든 입력 아티팩트의 디렉 터리는 \$CODEBUILD_SRC_DIR_yourInputArtifactName에서 사용할 수 있습니다.

Note

CodeBuild 프로젝트에서 구성된 아티팩트가 파이프라인에서는 CodeBuild 작업에 의해 사용 되는 입력 아티팩트가 됩니다.

출력 아티팩트

- 아티팩트 수: 0 to 5
- 설명: CodeBuild buildspec 파일에 정의된 아티팩트를 파이프라인의 후속 작업에 사용할 수 있도록 하는 데 사용할 수 있습니다. 출력 아티팩트가 하나만 정의되면 buildspec 파일의 artifacts 섹션 바로 아래에이 아티팩트를 정의할 수 있습니다. 둘 이상의 출력 아티팩트가 지정되면 참조되는 모든 아티팩트를 buildspec 파일에서 보조 아티팩트로 정의해야 합니다. CodePipeline의 출력 아티팩트 이름은 buildspec 파일의 아티팩트 식별자와 일치해야 합니다.

Note

CodeBuild 프로젝트에서 구성된 아티팩트가 파이프라인 작업에서는 CodePipeline 입력 아 티팩트가 됩니다. CombineArtifacts 파라미터를 배치 빌드용으로 선택한 경우 출력 아티팩트 위치에는 동일한 실 행에서 실행된 여러 빌드의 결합된 아티팩트가 포함됩니다.

출력 변수

이 작업은 빌드의 일부로 내보낸 모든 환경 변수를 변수로 생성합니다. 환경 변수를 내보내는 방법에 대한 자세한 내용은AWS CodeBuild API 가이드의 <u>EnvironmentVariable</u>을 참조하세요.

CodePipeline에서 CodeBuild 환경 변수를 사용하는 방법에 대한 자세한 내용은 <u>CodeBuild 작업 출력</u> 변수의 예제를 참조하세요. CodeBuild에서 사용할 수 있는 환경 변수 목록은AWS CodeBuild 사용 설 명서의 <u>빌드 환경의 환경 변수</u>를 참조하세요.

서비스 역할 권한: CodeBuild 작업

CodeBuild 지원의 경우 다음을 정책 설명에 추가합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "codebuild:BatchGetBuilds",
        "codebuild:StartBuild",
        "codebuild:BatchGetBuildBatches",
        "codebuild:StartBuildBatch"
      ],
      "Resource": [
        "arn:aws:codebuild:*:{{customerAccountId}}:project/[[ProjectName]]"
      ],
      "Effect": "Allow"
    }
  ]
}
```

작업 선언(CodeBuild 예제)

YAML

Name: Build

```
Actions:
  - Name: PackageExport
    ActionTypeId:
      Category: Build
      Owner: AWS
      Provider: CodeBuild
      Version: '1'
    RunOrder: 1
    Configuration:
      BatchEnabled: 'true'
      CombineArtifacts: 'true'
      ProjectName: my-build-project
      PrimarySource: MyApplicationSource1
      EnvironmentVariables:
 '[{"name":"TEST_VARIABLE","value":"TEST_VALUE","type":"PLAINTEXT"},
{"name":"ParamStoreTest", "value":"PARAMETER_NAME", "type":"PARAMETER_STORE"}]'
    OutputArtifacts:
      - Name: MyPipeline-BuildArtifact
    InputArtifacts:
      - Name: MyApplicationSource1
      - Name: MyApplicationSource2
```

JSON

```
{
    "Name": "Build",
    "Actions": [
        {
            "Name": "PackageExport",
            "ActionTypeId": {
                "Category": "Build",
                "Owner": "AWS",
                "Provider": "CodeBuild",
                "Version": "1"
            },
            "RunOrder": 1,
            "Configuration": {
                "BatchEnabled": "true",
                "CombineArtifacts": "true",
                "ProjectName": "my-build-project",
                "PrimarySource": "MyApplicationSource1",
```

```
"EnvironmentVariables": "[{\"name\":\"TEST_VARIABLE\",\"value\":
\"TEST_VALUE\", \"type\":\"PLAINTEXT\"}, {\"name\":\"ParamStoreTest\", \"value\":
\"PARAMETER_NAME\",\"type\":\"PARAMETER_STORE\"}]"
            },
            "OutputArtifacts": [
                {
                     "Name": "MyPipeline-BuildArtifact"
                }
            ],
            "InputArtifacts": [
                {
                     "Name": "MyApplicationSource1"
                },
                {
                     "Name": "MyApplicationSource2"
                }
            ]
        }
    ]
}
```

다음 사항도 참조하세요.

이 작업을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

- <u>AWS CodeBuild 사용 설명서</u> CodeBuild 작업이 포함된 예제 파이프라인은 <u>CodeBuild와 함께</u> <u>CodePipeline을 사용하여 코드 테스트 및 빌드 실행을 참조하세요</u>. 입력 및 출력 CodeBuild 아티팩 트가 여러 개인 프로젝트의 예제는 <u>CodePipeline과 CodeBuild의 통합 및 다중 입력 소스 및 출력 아</u> 티팩트 샘플과 다중 입력 소스 및 출력 아티팩트 샘플을 참조하세요.
- <u>자습서:를 사용하여 Android 앱을 빌드하고 테스트하는 파이프라인 생성 AWS Device Farm</u> -이 자 습서에서는 CodeBuild 및를 사용하여 Android 앱을 빌드하고 테스트하는 GitHub 소스로 파이프라인 을 생성하는 샘플 buildspec 파일 및 샘플 애플리케이션을 제공합니다 AWS Device Farm.
- <u>CodeBuild용 빌드 사양 참조</u> -이 참조 주제에서는 CodeBuild buildspec 파일을 이해하기 위한 정의 와 예제를 제공합니다. CodeBuild에서 사용할 수 있는 환경 변수 목록은AWS CodeBuild 사용 설명 서의 빌드 환경의 환경 변수를 참조하세요.

AWS CodePipeline 작업 참조 호출

CodePipeline 간접 호출 작업을 사용하여 다운스트림 파이프라인 실행을 트리거하고 파이프라인 간에 파이프라인 변수 및 소스 개정을 전달하는 작업을 간소화할 수 있습니다.

Note

이 작업은 V2 유형 파이프라인에서만 지원됩니다.

주제

- <u>작업 유형</u>
- 구성 파라미터
- 입력 아티팩트
- 출력 아티팩트
- CodePipeline 호출 작업에 대한 서비스 역할 정책 권한
- 작업 선언
- <u>다음 사항도 참조하세요.</u>

작업 유형

- 범주: Invoke
- 소유자: AWS
- 공급자: CodePipeline
- 버전:1

구성 파라미터

PipelineName

필수 여부: 예

실행 시 현재 대상 파이프라인을 시작할 파이프라인의 이름입니다. 호출 파이프라인을 이미 생성했 어야 합니다. 작업은 라는 s3-pipeline-test (호출) 파이프라인이 실행을 시작할 때 (대상) 파이 프라인을 my-s3-pipeline 시작합니다.

SourceRevisions

필수 여부: 아니요

간접 호출 파이프라인에 의해 시작될 때 대상 파이프라인이 사용할 소스 개정입니다. 예를 들어 S3 소스 작업은 S3 버전 ID 및 객체 키와 같은 출력 변수를 제공합니다. 파이프라인이 호출될 때 사용 할 개정 값을 지정할 수 있습니다.

CLI의 경우 소스 개정을 직렬화된 JSON 문자열로 지정합니다. 소스 개정 재정의를 사용하는 방법에 대한 자세한 내용은 CodePipeline API 안내서의 SourceRevisionOverride를 참조하세요.

매핑은 다음 예제와 같이 문자열 형식을 사용합니다.

```
[{"actionName":"Source","revisionType":"S3_OBJECT_VERSION_ID","revision
Value":"zq8mjNEXAMPLE"}]
```

변수

필수 여부: 아니요

작업이 지원할 변수의 이름과 값입니다.

CLI의 경우 변수를 직렬화된 JSON 문자열로 지정합니다. 파이프라인 변수 사용에 대한 자세한 내 용은 CodePipeline API 안내서의 <u>PipelineVariable</u>을 참조하세요.

매핑은 다음 예제와 같이 문자열 형식을 사용합니다.

[{"name":"VAR1","value":"VALUE1"}]

다음 이미지는 콘솔의 파이프라인에 추가된 작업의 예를 보여줍니다.

Developer Tools > CodePipeline > Pipelines > s3-	pipeline-t	est	
r⊡ s3-pipeline-test	l actions		Edit
Pipeline Executions Settings Tags	Stag	je	
		00	0
@ Source	\longrightarrow	@ Build	5
a1c23f9e-fadb-	ദ്		٠,
All actions succeeded.		a1c23f9e-fadb	
Source Amazon S3 2 Mar 11, 2025 11:36 AM (UTC-7:00)		Alt actions succeeded. Build Commands Mar 11, 2025 11:36 AM (UTC-7:00)	
Source: Amazon S3 •••		\checkmark	
		AWS CodePipeline Mar 11, 2025 11:36 AM (UTC-7:00)	
		Source: Amazon S3 ••••	

다음 이미지는 작업에 대한 편집 페이지의 예를 보여줍니다. 다음 예제에서 라는 파이프라인s3pipeline-test에는 콘솔에 대해 표시된 대로 구성된 파이프라인 호출 작업이 있습니다. 작업은 라 는 s3-pipeline-test 파이프라인이 실행을 my-s3-pipeline 완료하면 파이프라인을 시작합니 다. 이 예제는 S3_OBJECT_VERSION_ID 소스에 대한 소스 개정 재정의가 지정된 개정 값 로 재정의 됨을 보여줍니다zq8mjNYEexample.

Action name Thoose a name for your action	
Invoke-pipeline	
lo more than 100 characters	
Action provider	
AWS CodePipeline	▼
Region	
United States (N. Virginia)	•
Pipeline name Choose a pipeline that you have already created in the AWS ask. Q my-s3-pipeline	; CodePipeline console. Or create a pipeline in the AWS CodePipeline console and then ret
Source Revision Overrides - <i>optional</i> Choose the action name, revision type, and revision value for CodePipeline.	or your CodePipeline source revisions. In the value field, you can reference variables gener
Action name	Revision type
Source	S3_OBJECT_VERSION_ID
	Remove
Add source revision override	
/ariables - optional hoose the key and value for your CodePipeline pipeline va	riables. In the value field, you can reference variables generated by CodePipeline.
Add pipeline variable	
/ariable namespace - optional hoose a namespace for the output variables from this acti onfiguration. Learn more	on. You must choose a namespace if you want to use the variables this action produces in
1	
'	

• 설명: 이 작업 유형에는 입력 아티팩트가 적용되지 않습니다.
사용자 가이드

출력 아티팩트

- 아티팩트 수: 0
- 설명: 이 작업 유형에는 출력 아티팩트가 적용되지 않습니다.

CodePipeline 호출 작업에 대한 서비스 역할 정책 권한

CodePipeline이 작업을 실행할 때 CodePipeline 서비스 역할 정책에는 최소 codepipeline:StartPipelineExecution 권한으로 액세스를 유지하기 위해 파이프라인 리소스 ARN으로 범위가 적절하게 축소된 권한이 필요합니다.

```
{
    "Sid": "StatementForPipelineInvokeAction",
    "Effect": "Allow",
    "Action": "codepipeline:StartPipelineExecution",
    "Resource": [
        "arn:aws:codepipeline:{{region}}:{{AccountId}}:{{pipelineName}}"
    ]
}
```

작업 선언

YAML

```
name: Invoke-pipeline
actionTypeId:
  category: Invoke
  owner: AWS
  provider: CodePipeline
  version: '1'
runOrder: 2
configuration:
  PipelineName: my-s3-pipeline
  SourceRevisions:
  '[{"actionName":"Source","revisionType":"S3_OBJECT_VERSION_ID","revision
Value":"zq8mjNEXAMPLE"}]'
  Variables: '[{"name":"VAR1","value":"VALUE1"}]'
```

사용자 가이드

```
{
    "name": "Invoke-pipeline",
    "actionTypeId": {
        "category": "Invoke",
        "owner": "AWS",
        "provider": "CodePipeline",
        "version": "1"
    },
    "runOrder": 2,
    "configuration": {
        "PipelineName": "my-s3-pipeline",
        "SourceRevisions": "[{\"actionName\":\"Source\",\"revisionType\":
\"S3_OBJECT_VERSION_ID\", \"revisionValue\":\"zq8mjNEXAMPLE"}]",
        "Variables": "[{\"name\":\"VAR1\",\"value\":\"VALUE1\"}]"
    }
},
```

다음 사항도 참조하세요.

이 작업을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

• <u>소스 개정 재정의로 파이프라인 시작</u> -이 섹션에서는 소스 개정을 사용하여 파이프라인을 수동으로 또는 EventBridge 이벤트 입력 변환기를 통해 시작하는 방법을 설명합니다.

CodeCommit 소스 작업 참조

구성된 CodeCommit 리포지토리 및 분기에서 새 커밋이 수행될 때 파이프라인을 시작합니다.

콘솔을 사용하여 파이프라인을 생성하거나 편집하는 경우 CodePipeline은 리포지토리에서 변경이 발 생할 때 파이프라인을 시작하는 EventBridge 규칙을 생성합니다.

Note

Amazon ECR, Amazon S3 또는 CodeCommit 소스의 경우 입력 변환 항목을 사용하여 소스 재 정의를 생성하여 파이프라인 이벤트에 EventBridgerevisionValue의를 사용할 수도 있습니 다. 여기서 revisionValue는 객체 키, 커밋 또는 이미지 ID의 소스 이벤트 변수에서 파생됩 니다. 자세한 내용은 , Amazon ECR 소스 작업 및 EventBridge 리소스 이벤트에 대해 활성화된 <u>소스를 사용하여 Amazon S3 소스 작업에 연결</u>또는의 절차에 포함된 입력 변환 항목의 선택적 단계를 참조하세요CodeCommit 소스 작업 및 EventBridge.

CodeCommit 작업을 통해 파이프라인을 연결하기 전에 CodeCommit 리포지토리를 이미 생성했어야 합니다.

코드 변경이 감지되면 다음 옵션을 사용하여 코드를 후속 작업에 전달할 수 있습니다.

- 기본값 커밋의 단순 복사본이 포함된 ZIP 파일을 출력하도록 CodeCommit 소스 작업을 구성합니다.
- 전체 복제 후속 작업을 위해 리포지토리에 대한 Git URL 참조를 출력하도록 소스 작업을 구성합니다.

현재 Git URL 참조는 다운스트림 CodeBuild 작업에서만 리포지토리와 관련 Git 메타데이터를 복제 하는 데 사용할 수 있습니다. 비 CodeBuild 작업에 대한 Git URL 참조를 전달하려고 하면 오류가 발 생합니다.

주제

- <u>작업 유형</u>
- 구성 파라미터
- 입력 아티팩트
- 출력 아티팩트
- 출력 변수
- 서비스 역할 권한: CodeCommit 작업
- <u>예제 작업 구성</u>
- <u>다음 사항도 참조하세요.</u>

작업 유형

- 범주: Source
- 소유자: AWS
- 공급자: CodeCommit
- 버전:1

구성 파라미터

RepositoryName

필수 여부: 예

소스 변경 사항을 감지할 리포지토리의 이름입니다.

BranchName

필수 여부: 예

소스 변경 사항을 감지할 분기의 이름입니다.

PollForSourceChanges

필수 여부: 아니요

PollForSourceChanges는 CodePipeline이 CodeCommit 리포지토리에서 소스 변경 사항 을 폴링할지 여부를 제어합니다. 대신 CloudWatch Events를 사용하여 소스 변경 사항을 감지 하는 것이 좋습니다. CloudWatch Events 구성에 대한 자세한 내용은 <u>폴링 파이프라인 마이그</u> <u>레이션(CodeCommit 소스)(CLI)</u> 또는 <u>폴링 파이프라인 마이그레이션(CodeCommit 소스)(AWS</u> <u>CloudFormation 템플릿)</u>을 참조하세요.

A Important

CloudWatch Events 규칙을 구성하려는 경우, 중복된 파이프라인 실행이 발생하지 않도록 PollForSourceChanges를 false로 설정해야 합니다.

이 파라미터에 유효한 값은 다음과 같습니다.

• true: 이 값으로 설정된 경우 CodePipeline은 리포지토리에서 소스 변경 사항을 폴링합니다.

Note

PollForSourceChanges를 생략하면 CodePipeline은 기본적으로 소스 변경 사항에 대 해 리포지토리를 폴링합니다. 이러한 동작은 PollForSourceChanges이 포함되었고 true로 설정된 경우와 똑같습니다.

• false: 이 값으로 설정된 경우 CodePipeline은 리포지토리에서 소스 변경 사항을 폴링하지 않습 니다. 소스 변경 사항을 감지하도록 CloudWatch Events 규칙을 구성하려면 이 설정을 사용합니 다.

OutputArtifactFormat

필수 여부: 아니요

출력 아티팩트 형식. 값은 CODEBUILD_CLONE_REF 또는 CODE_ZIP가 될 수 있습니다. 미지정된 경우 기본값은 CODE_ZIP입니다.

▲ Important

CODEBUILD_CLONE_REF 옵션은 CodeBuild 다운스트림 작업에서만 사용할 수 있습니다. 이 옵션을 선택하면 <u>CodeCommit 소스 작업에 대한 CodeBuild GitClone 권한 추가</u>과 같 이 CodeBuild 서비스 역할에 대한 codecommit:GitPull 권한을 추가해야 합니다. 또 한 <u>CodePipeline 서비스 역할에 권한 추가</u>에 표시된 대로 CodePipeline 서비스 역할에 codecommit:GetRepository 권한을 추가해야 합니다. 전체 복제 옵션을 사용하는 방법 을 보여주는 자습서는 <u>자습서:CodeCommit 파이프라인 소스와 함께 전체 복제 사용</u>을 참 조하세요.

입력 아티팩트

- 아티팩트 수: 0
- 설명: 이 작업 유형에는 입력 아티팩트가 적용되지 않습니다.

출력 아티팩트

- 아티팩트 수: 1
- 설명: 이 작업의 출력 아티팩트는 파이프라인 실행을 위한 소스 개정으로 지정된 커밋에서 구성 된 리포지토리 및 분기의 내용을 포함하는 ZIP 파일입니다. 리포지토리에서 생성된 아티팩트는 CodeCommit 작업에 대한 출력 아티팩트입니다. 소스 코드 커밋 ID는 트리거된 파이프라인 실행의 소스 개정으로 CodePipeline에 표시됩니다.

출력 변수

이 작업을 구성하면 파이프라인에서 다운스트림 작업의 작업 구성에서 참조할 수 있는 변수가 생성됩 니다. 이 작업은 작업에 네임스페이스가 없는 경우에도 출력 변수로 볼 수 있는 변수를 생성합니다. 이 러한 변수를 다운스트림 작업 구성에서 사용할 수 있도록 네임스페이스를 사용하여 작업을 구성합니 다. 자세한 내용은 <u>변수 참조</u> 단원을 참조하십시오.

CommitId

파이프라인 실행을 트리거한 CodeCommit 커밋 ID입니다. 커밋 ID는 커밋의 전체 SHA입니다. CommitMessage

파이프라인 실행을 트리거한 커밋과 연관된 설명 메시지입니다(존재하는 경우).

RepositoryName

파이프라인을 트리거한 커밋이 만들어진 CodeCommit 리포지토리의 이름입니다.

BranchName

소스 변경이 이루어진 CodeCommit 리포지토리의 분기 이름입니다.

AuthorDate

커밋이 작성된 날짜입니다(타임스탬프 형식).

CommitterDate

커밋이 수행된 날짜입니다(타임스탬프 형식).

서비스 역할 권한: CodeCommit 작업

CodePipeline이 작업을 실행할 때 CodePipeline 서비스 역할 정책에는 최소 권한으로 액세스를 유지하 기 위해 파이프라인 리소스 ARN으로 범위가 적절하게 축소된 다음 권한이 필요합니다. 예를 들어 정책 설명에 다음을 추가합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
         "Effect": "Allow",
         "Action": [
            "codecommit:CancelUploadArchive",
            "codecommit:GetBranch",
            "codecommit:GetCommit",
            "codecommit:GetRepository",
            "codecommit:GetUploadArchiveStatus",
            "codecommit:UploadArchiveStatus",
            "codecommit:UploadArchive"
        ],
        "Resource": [
```

```
"arn:aws:codecommit:*:{{customerAccountId}}:[[codecommitRepostories]]"
]
]
}
```

예제 작업 구성

기본 출력 아티팩트 형식의 예

YAML

```
name: Source
actionTypeId:
 category: Source
 owner: AWS
  provider: CodeCommit
  version: '1'
runOrder: 1
configuration:
  BranchName: main
  PollForSourceChanges: 'false'
  RepositoryName: MyWebsite
outputArtifacts:
  - name: Artifact_MyWebsiteStack
inputArtifacts: []
region: us-west-2
namespace: SourceVariables
```

JSON

```
{
    "name": "Source",
    "actionTypeId": {
        "category": "Source",
        "owner": "AWS",
        "provider": "CodeCommit",
        "version": "1"
    },
    "runOrder": 1,
    "configuration": {
        "BranchName": "main",
    }
}
```



전체 복제 출력 아티팩트 형식의 예

YAML

```
name: Source
actionTypeId:
 category: Source
 owner: AWS
  provider: CodeCommit
  version: '1'
runOrder: 1
configuration:
  BranchName: main
 OutputArtifactFormat: CODEBUILD_CLONE_REF
  PollForSourceChanges: 'false'
  RepositoryName: MyWebsite
outputArtifacts:
  - name: SourceArtifact
inputArtifacts: []
region: us-west-2
namespace: SourceVariables
```

JSON

```
{
    "name": "Source",
    "actionTypeId": {
        "category": "Source",
        "owner": "AWS",
```

```
"provider": "CodeCommit",
        "version": "1"
    },
    "runOrder": 1,
    "configuration": {
        "BranchName": "main",
        "OutputArtifactFormat": "CODEBUILD_CLONE_REF",
        "PollForSourceChanges": "false",
        "RepositoryName": "MyWebsite"
    },
    "outputArtifacts": [
        {
            "name": "SourceArtifact"
        }
    ],
    "inputArtifacts": [],
    "region": "us-west-2",
    "namespace": "SourceVariables"
}
```

다음 사항도 참조하세요.

이 작업을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

• <u>자습서: 간단한 파이프라인 생성(CodeCommit 리포지토리)</u> - 이 자습서에서는 앱 사양 파일 샘플과 CodeDeploy 애플리케이션 및 배포 그룹 샘플을 제공합니다. 이 자습서를 사용하여 Amazon EC2 인 스턴스에 배포되는 CodeCommit 소스로 파이프라인을 생성합니다.

AWS CodeDeploy 배포 작업 참조

AWS CodeDeploy 작업을 사용하여 배포 플릿에 애플리케이션 코드를 배포합니다. 배포 플릿은 Amazon EC2 인스턴스, 온프레미스 인스턴스 또는 둘 다로 구성될 수 있습니다.

Note

이 참조 주제에서는 CodePipeline의 CodeDeploy 배포 작업에 대해 설명합니다. 여기서 배 포 플랫폼은 Amazon EC2입니다. CodePipeline에서의 CodeDeploy 블루/그린 배포 작업 에 대한 Amazon Elastic Container Service에 대한 참조 정보는 <u>Amazon Elastic Container</u> <u>Service(Amazon ECS) 및 CodeDeploy 블루-그린 배포 작업 참조</u>을 참조하세요.

주제

- <u>작업 유형</u>
- <u>구성 파라미터</u>
- 입력 아티팩트
- 출력 아티팩트
- <u>서비스 역할 권한: AWS CodeDeploy 작업</u>
- <u>작업 선언</u>
- <u>다음 사항도 참조하세요.</u>

작업 유형

- 범주: Deploy
- 소유자: AWS
- 공급자: CodeDeploy
- 버전:1

구성 파라미터

ApplicationName

필수 여부: 예

CodeDeploy에서 생성한 애플리케이션 이름입니다.

DeploymentGroupName

필수 여부: 예

CodeDeploy 에서 생성한 배포 그룹입니다.

입력 아티팩트

- 아티팩트 수: 1
- 설명: CodeDeploy에서 다음을 결정하는 데 사용하는 AppSpec 파일입니다.
 - Amazon S3 또는 GitHub의 애플리케이션 수정에서 인스턴스에 설치할 항목입니다.

• 배포 수명 주기 이벤트에 대한 응답으로 실행될 수명 주기 이벤트 후크

AppSpec 파일에 대한 자세한 내용은 CodeDeploy AppSpec 파일 참조를 참조하세요.

출력 아티팩트

- 아티팩트 수: 0
- 설명: 이 작업 유형에는 출력 아티팩트가 적용되지 않습니다.

서비스 역할 권한: AWS CodeDeploy 작업

AWS CodeDeploy 지원을 받으려면 정책 설명에 다음을 추가합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codedeploy:CreateDeployment",
        "codedeploy:GetApplication",
        "codedeploy:GetDeployment",
        "codedeploy:RegisterApplicationRevision",
        "codedeploy:ListDeployments",
        "codedeploy:ListDeploymentGroups",
        "codedeploy:GetDeploymentGroup"
      ],
      "Resource": [
        "arn:aws:codedeploy:*:{{customerAccountId}}:application:
[[codedeployApplications]]",
        "arn:aws:codedeploy:*:{{customerAccountId}}:deploymentgroup:
[[codedeployApplications]]/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "codedeploy:GetDeploymentConfig"
      ],
      "Resource": [
```

```
"arn:aws:codedeploy:*:{{customerAccountId}}:deploymentconfig:
[[deploymentConfigs]]"
]
},
{
"Effect": "Allow",
"Action": [
"codedeploy:ListDeploymentConfigs"
],
"Resource": [
"*"
]
}
```

작업 선언

YAML

```
Name: Deploy
Actions:
  - Name: Deploy
   ActionTypeId:
      Category: Deploy
      Owner: AWS
      Provider: CodeDeploy
      Version: '1'
    RunOrder: 1
    Configuration:
      ApplicationName: my-application
      DeploymentGroupName: my-deployment-group
    OutputArtifacts: []
    InputArtifacts:
      - Name: SourceArtifact
    Region: us-west-2
    Namespace: DeployVariables
```

JSON

"Name": "Deploy",

{

```
"Actions": [
        {
            "Name": "Deploy",
            "ActionTypeId": {
                "Category": "Deploy",
                "Owner": "AWS",
                "Provider": "CodeDeploy",
                "Version": "1"
            },
            "RunOrder": 1,
            "Configuration": {
                "ApplicationName": "my-application",
                "DeploymentGroupName": "my-deployment-group"
            },
            "OutputArtifacts": [],
            "InputArtifacts": [
                {
                     "Name": "SourceArtifact"
                }
            ],
            "Region": "us-west-2",
            "Namespace": "DeployVariables"
        }
    ]
},
```

다음 사항도 참조하세요.

이 작업을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

- <u>자습서: 간단한 파이프라인 생성(S3 버킷)</u> 이 자습서에서는 샘플 애플리케이션을 배포하기 위한 소 스 버킷, EC2 인스턴스 및 CodeDeploy 리소스를 생성하는 방법을 안내합니다. 그런 다음 S3 버킷에 서 유지 관리하는 코드를 Amazon EC2 인스턴스에 배포하는 CodeDeploy 배포 작업을 사용하여 파 이프라인을 구축합니다.
- <u>자습서: 간단한 파이프라인 생성(CodeCommit 리포지토리)</u> 이 자습서에서는 샘플 애플리케이션을 배포하기 위한 CodeCommit 소스 리포지토리, EC2 인스턴스 및 CodeDeploy 리소스를 생성하는 방 법을 안내합니다. 그런 다음 CodeCommit 리포지토리의 코드를 Amazon EC2 인스턴스에 배포하는 CodeDeploy 배포 작업을 사용하여 파이프라인을 구축합니다.
- <u>CodeDeploy AppSpec 파일 참조</u> AWS CodeDeploy 사용 설명서의 이 참조 장에서는 CodeDeploy AppSpec 파일에 대한 참조 정보와 예제를 제공합니다.

Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com 및 GitLab 자체 관리형 작업용 CodeStarSourceConnection

연결에 대한 소스 작업은에서 지원합니다 AWS CodeConnections. CodeConnections를 사용하면 AWS 리소스와 GitHub와 같은 타사 리포지토리 간의 연결을 생성하고 관리할 수 있습니다. 타사 소스 코드 리포지토리에서 새 커밋이 수행될 때 파이프라인을 시작합니다. 소스 작업은 파이프라인이 수동 으로 실행되거나 소스 공급자로부터 Webhook 이벤트가 전송될 경우 코드 변경 사항을 검색합니다.

트리거로 파이프라인을 시작할 수 있는 Git 구성을 사용하도록 파이프라인의 작업을 구성할 수 있습니 다. 트리거로 필터링하도록 파이프라인 트리거 구성을 구성하려면 <u>코드 푸시 또는 풀 요청 이벤트 유형</u> <u>을 사용하여 트리거 추가</u>에서 자세한 내용을 참조하세요.

Note

이 기능은 아시아 태평양(홍콩), 아시아 태평양(하이데라바드), 아시아 태평양(자카르타), 아시아 태평양(멜버른), 아시아 태평양(오사카), 아프리카(케이프타운), 중동(바레인), 중동 (UAE), 유럽(스페인), 유럽(취리히), 이스라엘(텔아비브) 또는 AWS GovCloud(미국 서부) 리전에서는 사용할 수 없습니다. 사용 가능한 다른 작업을 참조하려면 <u>CodePipeline과 제</u> <u>품 및 서비스 통합</u>을 참조하세요. 유럽(밀라노) 리전에서 이 조치를 고려할 경우 <u>Bitbucket</u> <u>Cloud, GitHub, GitHub Enterprise Server, GitLab.com 및 GitLab 자체 관리형 작업용</u> <u>CodeStarSourceConnection</u>의 참고 사항을 참조하세요.

연결을 통해 리소스를 다음 타사 리포지토리 AWS 와 연결할 수 있습니다.

 Bitbucket Cloud(CodePipeline 콘솔의 Bitbucket 공급자 옵션 또는 CLI의 Bitbucket 공급자를 통 해)

Note

Bitbucket Cloud 리포지토리에 대한 연결을 생성할 수 있습니다. Bitbucket Server와 같은 설치된 Bitbucket 공급자 유형은 지원되지 않습니다.

Note

Bitbucket WorkSpace를 사용하는 경우 연결을 만들려면 관리자 액세스 권한이 있어야 합니 다. GitHub 및 GitHub Enterprise Cloud(CodePipeline 콘솔의 GitHub(GitHub 앱 사용) 공급자 옵션 또는 CLI의 GitHub 공급자를 통해)

Note

리포지토리가 GitHub 조직에 있는 경우 연결을 생성하려면 조직 소유자여야 합니다. 조직 소속이 아닌 리포지토리를 사용하고 있는 경우 리포지토리 소유자여야 합니다.

- GitHub Enterprise Server(CodePipeline 콘솔의 GitHub Enterprise Server 공급자 옵션 또는 CLI의 GitHub Enterprise Server 공급자를 통해)
- GitLab.com(CodePipeline 콘솔의 GitLab 공급자 옵션 또는 CLI의 GitLab 공급자를 통해)

Note

GitLab에서 소유자 역할을 가진 리포지토리에 대한 연결을 생성한 다음 CodePipeline과 같 은 리소스가 있는 리포지토리에서 해당 연결을 사용할 수 있습니다. 그룹 내 리포지토리의 경우 그룹 소유자가 아니어도 됩니다.

• GitLab용 자체 관리형 설치(엔터프라이즈 에디션 또는 커뮤니티 에디션)(CodePipeline 콘솔의 GitLab 자체 관리형 공급자 옵션 또는 CLI의 GitLabSelfManaged 공급자를 통해)

Note

각 연결은 해당 공급자에 있는 모든 리포지토리를 지원합니다. 각 공급자 유형에 대해 새 연결 만 생성하면 됩니다.

연결을 통해 파이프라인은 타사 공급자의 설치 앱을 통해 소스 변경을 감지할 수 있습니다. 예를 들어 웹후크는 GitHub 이벤트 유형을 구독하는 데 사용되며 조직, 리포지토리 또는 GitHub 앱에 설치할 수 있습니다. 연결하면 GitHub 푸시 유형 이벤트를 구독하는 리포지토리 웹후크가 GitHub 앱에 설치됩니 다.

코드 변경이 감지되면 다음 옵션을 사용하여 코드를 후속 작업에 전달할 수 있습니다.

- 기본값: 다른 기존 CodePipeline 소스 작업과 마찬가지로, CodeStarSourceConnection에서는 커밋의 부분 복사본이 포함된 ZIP 파일을 출력할 수 있습니다.
- 전체 복제: CodeStarSourceConnection은 후속 작업을 위해 리포지토리에 대한 URL 참조를 출 력하도록 구성할 수도 있습니다.

현재 Git URL 참조는 다운스트림 CodeBuild 작업에서만 리포지토리와 관련 Git 메타데이터를 복제 하는 데 사용할 수 있습니다. 비 CodeBuild 작업에 대한 Git URL 참조를 전달하려고 하면 오류가 발 생합니다.

CodePipeline은 연결을 생성할 때 타사 계정에 AWS 커넥터 설치 앱을 추가하라는 메시지를 표시합니다. CodeStarSourceConnection 작업을 통해 연결하려면 먼저 타사 공급자 계정 및 리포지토리를 생성해야 합니다.

Note

AWS CodeStar 연결을 사용하는 데 필요한 권한을 가진 역할에 정책을 생성하거나 연결하 려면 <u>연결 권한 참조를</u> 참조하세요. CodePipeline 서비스 역할이 생성된 시기에 따라 AWS CodeStar 연결을 지원하도록 권한을 업데이트해야 할 수 있습니다. 지침은 <u>CodePipeline 서비</u> <u>스 역할에 권한 추가</u> 섹션을 참조하세요.

1 Note

유럽(밀라노)에서 연결을 사용하려면 다음을 AWS 리전수행해야 합니다.

- 1. 리전별 앱 설치
- 2. 리전 활성화

이 리전별 앱은 유럽(밀라노) 리전 내 연결을 지원합니다. 서드 파티 제공업체 사이트에 게시되 며 다른 리전의 연결을 지원하는 기존 앱과는 별개입니다. 이 앱을 설치하면 서드 파티 제공업 체가 이 리전에서만 서비스와 데이터를 공유할 수 있는 권한을 부여하게 되며 앱을 제거하여 언제든지 권한을 취소할 수 있습니다.

리전을 활성화하지 않으면 서비스에서 데이터를 처리하거나 저장하지 않습니다. 이 리전을 활 성화하면 데이터를 처리하고 저장할 수 있는 권한이 서비스에 부여됩니다.

리전이 활성화되지 않았더라도 리전별 앱이 설치된 상태로 유지되면 서드 파티 제공업체가 서 비스와 데이터를 공유할 수 있으므로 리전을 비활성화한 후에는 앱을 제거해야 합니다. 자세한 내용은 리전 활성화를 참조하세요.

주제

작업 유형

- 구성 파라미터
- 입력 아티팩트
- 출력 아티팩트
- <u>출력 변수</u>
- <u>서비스 역할 권한: CodeConnections 작업</u>
- <u>작업 선언</u>
- 설치 앱 설치 및 연결 생성
- 다음 사항도 참조하세요.

작업 유형

- 범주: Source
- 소유자: AWS
- 공급자: CodeStarSourceConnection
- 버전:1

구성 파라미터

ConnectionArn

필수 여부: 예

소스 공급자에 대해 구성 및 인증된 연결 ARN입니다.

FullRepositoryId

필수 여부: 예

소스 변경 사항을 감지할 리포지토리의 소유자 및 이름입니다.

예시:some-user/my-repo

A Important

FullRepositoryId 값의 대/소문자를 올바르게 유지해야 합니다. 예를 들어 사용자 이름 이 some-user이고 리포지토리 이름이 My-Repo인 경우 FullRepositoryId의 권장 값은 some-user/My-Repo입니다.

BranchName

필수 여부: 예

소스 변경 사항을 감지할 분기의 이름입니다.

OutputArtifactFormat

필수 여부: 아니요

출력 아티팩트 형식을 지정합니다. CODEBUILD_CLONE_REF 또는 CODE_ZIP입니다. 미지정된 경 우 기본값은 CODE_ZIP입니다.

▲ Important

CODEBUILD_CLONE_REF 옵션은 CodeBuild 다운스트림 작업에서만 사용할 수 있습니다. 이 옵션을 선택하면 <u>Bitbucket, GitHub, GitHub Enterprise Server 또는 GitLab.com 연결을</u> <u>위한 CodeBuild GitClone 권한 추가</u>과 같이 CodeBuild 프로젝트 서비스 역할에 대한 권한 을 업데이트해야 합니다. 전체 복제 옵션을 사용하는 방법을 보여주는 자습서는 <u>자습서:</u> <u>GitHub 파이프라인 소스와 함께 전체 복제 사용</u>을 참조하세요.

DetectChanges

필수 여부: 아니요

구성된 리포지토리 및 브랜치에서 새 커밋이 이루어지면 파이프라인이 자동으로 시작되도록 제어 합니다. 지정되지 않은 경우 기본값은 true이며 필드는 기본적으로 표시되지 않습니다. 이 파라미 터에 유효한 값은 다음과 같습니다.

- true: CodePipeline은 새 커밋에서 자동으로 파이프라인을 시작합니다.
- false: CodePipeline은 새 커밋에서 파이프라인을 시작하지 않습니다.

입력 아티팩트

- 아티팩트 수: 0
- 설명: 이 작업 유형에는 입력 아티팩트가 적용되지 않습니다.

출력 아티팩트

- 아티팩트 수: 1
- 설명: 리포지토리에서 생성된 아티팩트는 CodeStarSourceConnection 작업에 대한 출력 아티팩 트입니다. 소스 코드 커밋 ID는 트리거된 파이프라인 실행의 소스 개정으로 CodePipeline에 표시됩 니다. 이 작업의 출력 아티팩트는 다음 파일에서 구성할 수 있습니다.
 - 파이프라인 실행을 위한 소스 개정으로 지정된 커밋에서 구성된 리포지토리 및 브랜치의 내용을 포함하는 ZIP 파일
 - 다운스트림 작업이 Git 명령을 직접 수행할 수 있도록 리포지토리에 대한 URL 참조가 포함된 JSON 파일

▲ Important

이 옵션은 CodeBuild 다운스트림 작업에서만 사용할 수 있습니다.

이 옵션을 선택하면 <u>CodePipeline 문제 해결</u>과 같이 CodeBuild 프로젝트 서비스 역할에 대한 권한을 업데이트해야 합니다. 전체 복제 옵션을 사용하는 방법을 보여주는 자습서는 자습서: GitHub 파이프라인 소스와 함께 전체 복제 사용을 참조하세요.

출력 변수

이 작업을 구성하면 파이프라인에서 다운스트림 작업의 작업 구성에서 참조할 수 있는 변수가 생성됩 니다. 이 작업은 작업에 네임스페이스가 없는 경우에도 출력 변수로 볼 수 있는 변수를 생성합니다. 이 러한 변수를 다운스트림 작업 구성에서 사용할 수 있도록 네임스페이스를 사용하여 작업을 구성합니 다.

자세한 내용은 변수 참조 단원을 참조하십시오.

AuthorDate

커밋이 작성된 날짜입니다(타임스탬프 형식).

BranchName

소스 변경이 이루어진 리포지토리의 분기 이름입니다.

CommitId

파이프라인 실행을 트리거한 커밋 ID입니다.

CommitMessage

파이프라인 실행을 트리거한 커밋과 연관된 설명 메시지입니다(존재하는 경우).

ConnectionArn

소스 공급자에 대해 구성 및 인증된 연결 ARN입니다.

FullRepositoryName

파이프라인을 트리거한 커밋이 만들어진 리포지토리의 이름입니다.

서비스 역할 권한: CodeConnections 작업

CodeConnections의 경우 Bitbucket Cloud와 같은 연결을 사용하는 소스로 파이프라인을 생성하려면 다음 권한이 필요합니다.

```
{
    "Effect": "Allow",
    "Action": [
        "codeconnections:UseConnection"
    ],
    "Resource": "resource_ARN"
},
```

연결의 IAM 권한에 대한 자세한 내용은 <u>연결 권한 참조</u>를 참조하세요.

작업 선언

다음 예제에서 출력 아티팩트는 ARN arn:aws:codestar-connections:region:*account-id*:connection/*connection-id*과의 연결에 대한 CODE_ZIP의 기본 ZIP 형식으로 설정되어 있습니다.

YAML

```
Name: Source
Actions:
    - InputArtifacts: []
    ActionTypeId:
        Version: '1'
        Owner: AWS
```

```
Category: Source

Provider: CodeStarSourceConnection

OutputArtifacts:

- Name: SourceArtifact

RunOrder: 1

Configuration:

ConnectionArn: "arn:aws:codestar-connections:region:account-

id:connection/connection-id"

FullRepositoryId: "some-user/my-repo"

BranchName: "main"

OutputArtifactFormat: "CODE_ZIP"

Name: ApplicationSource
```

JSON

```
{
    "Name": "Source",
    "Actions": [
        {
            "InputArtifacts": [],
            "ActionTypeId": {
                "Version": "1",
                "Owner": "AWS",
                "Category": "Source",
                "Provider": "CodeStarSourceConnection"
            },
            "OutputArtifacts": [
                {
                     "Name": "SourceArtifact"
                }
            ],
            "RunOrder": 1,
            "Configuration": {
                "ConnectionArn": "arn:aws:codestar-connections:region:account-
id:connection/connection-id",
                "FullRepositoryId": "some-user/my-repo",
                "BranchName": "main",
                "OutputArtifactFormat": "CODE_ZIP"
            },
            "Name": "ApplicationSource"
        }
    ]
},
```

설치 앱 설치 및 연결 생성

콘솔을 사용하여 타사 리포지토리에 대한 새 연결을 처음 추가하는 경우 해당 리포지토리에 대한 액세 스 권한을 CodePipeline에 부여해야 합니다. 타사 코드 리포지토리를 만든 계정에 연결하는 데 도움이 되는 설치 앱을 선택하거나 생성합니다.

AWS CLI 또는 AWS CloudFormation 템플릿을 사용하는 경우 이미 설치 핸드셰이크를 거친 연결의 연 결 ARN을 제공해야 합니다. 그러지 않으면 파이프라인이 트리거되지 않습니다.

Note

CodeStarSourceConnection 소스 작업의 경우 webhook를 설정하거나 기본값을 폴링으로 설정할 필요가 없습니다. 연결 작업은 소스 변경 감지를 관리합니다.

다음 사항도 참조하세요.

이 작업을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

- <u>AWS::CodeStarConnections::Connection</u> AWS CodeStar Connections 리소스에 대한 AWS CloudFormation 템플릿 참조는 AWS CloudFormation 템플릿의 연결에 대한 파라미터와 예제를 제 공합니다.
- <u>AWS CodeStar Connections API 참조</u> AWS CodeStar Connections API 참조는 사용 가능한 연결 작업에 대한 참조 정보를 제공합니다.
- 연결에서 지원되는 소스 작업으로 파이프라인을 생성하는 단계를 보려면 다음을 참조하세요.
 - Bitbucket Cloud의 경우 콘솔의 Bitbucket 옵션 또는 CLI의 CodestarSourceConnection 작업 을 사용하세요. Bitbucket Cloud 연결을(를) 참조하세요.
 - GitHub 및 GitHub Enterprise Cloud의 경우 콘솔의 GitHub 공급자 옵션 또는 CLI의 CodestarSourceConnection 작업을 사용하세요. GitHub 연결을(를) 참조하세요.
 - GitHub Enterprise Server의 경우 콘솔의 GitHub Enterprise Server 공급자 옵션 또는 CLI의 CodestarSourceConnection 작업을 사용하세요. <u>GitHub Enterprise Server 연결</u>을(를) 참조하 세요.
 - GitLab.com의 경우 콘솔의 GitLab 공급자 옵션 또는 CLI의 GitLab 공급자와 함께
 CodestarSourceConnection 작업을 사용하세요. <u>GitLab.com 연결</u>을(를) 참조하세요.
- Bitbucket 소스와 CodeBuild 작업으로 파이프라인을 생성하는 시작하기 자습서를 보려면 <u>연결 시작</u> 하기를 참조하세요.

 GitHub 리포지토리에 연결하고 다운스트림 CodeBuild 작업과 함께 전체 복제 옵션을 사용하는 방법 을 보여주는 자습서는 자습서: GitHub 파이프라인 소스와 함께 전체 복제 사용 단원을 참조하세요.

명령 작업 참조

명령 작업을 사용하면 가상 컴퓨팅 인스턴스에서 쉘 명령을 실행할 수 있습니다. 작업을 실행하면 작업 구성에 지정된 명령이 별도의 컨테이너에서 실행됩니다. 명령을 실행하는 컨테이너 내에서 CodeBuild 작업에 대해 입력 아티팩트로 지정된 모든 아티팩트를 사용할 수 있습니다. 이 작업을 사 용하면 CodeBuild 프로젝트를 먼저 생성하지 않고 명령을 지정할 수 있습니다. 자세한 내용은 AWS CodePipeline API 참조의 <u>ActionDeclaration</u> 및 <u>OutputArtifact</u>를 참조하세요.

🛕 Important

이 작업은 CodePipeline 관리형 CodeBuild 컴퓨팅을 사용하여 빌드 환경에서 명령을 실행합니 다. 명령 작업을 실행하면 AWS CodeBuild에서 별도의 요금이 발생합니다.

Note

명령 작업은 V2 유형 파이프라인에서만 사용할 수 있습니다.

주제

- 명령 작업에 대한 고려 사항
- 서비스 역할 정책 권한
- 작업 유형
- 구성 파라미터
- 입력 아티팩트
- 출력 아티팩트
- 환경 변수
- 서비스 역할 권한: 명령 작업
- <u>작업 선언(예)</u>
- 다음 사항도 참조하세요.

명령 작업에 대한 고려 사항

명령 작업에는 다음 고려 사항이 적용됩니다.

• 명령 작업은 CodeBuild 작업과 유사한 CodeBuild 리소스를 사용하는 동시에 빌드 프로젝트를 연결 하거나 생성할 필요 없이 가상 컴퓨팅 인스턴스에서 쉘 환경 명령을 허용합니다.

Note

명령 작업을 실행하면 AWS CodeBuild에서 별도의 요금이 발생합니다.

- CodePipeline의 명령 작업은 CodeBuild 리소스를 사용하기 때문에 작업에서 실행되는 빌드는 CodeBuild에서 계정의 빌드 제한에 귀속됩니다. 명령 작업에서 실행되는 빌드는 해당 계정에 구성된 동시 빌드 제한에 포함됩니다.
- CodeBuild 빌드를 기반으로 할 때 명령 작업을 사용한 빌드의 제한 시간은 55분입니다.
- 컴퓨팅 인스턴스는 CodeBuild에서 격리된 빌드 환경을 사용합니다.

Note

격리된 빌드 환경은 계정 수준에서 사용되므로 인스턴스를 다른 파이프라인 실행에 재사용 할 수 있습니다.

- 다중 라인 형식을 제외한 모든 형식이 지원됩니다. 명령을 입력할 때는 단일 줄 형식을 사용해야 합니다.
- 명령 작업은 교차 계정 작업에 대해 지원됩니다. 교차 계정 명령 작업을 추가하려면 작업 선언의 대 상 계정actionRoleArn에서를 추가합니다.
- 이 작업의 경우 CodePipeline은 파이프라인 서비스 역할을 수행하고 해당 역할을 사용하여 런타임
 시 리소스에 대한 액세스를 허용합니다. 권한 범위가 작업 수준까지 내려가도록 서비스 역할을 구성
 하는 것이 좋습니다.
- CodePipeline 서비스 역할에 추가된 권한은 <u>CodePipeline 서비스 역할에 권한 추가</u>에 자세히 설명되 어 있습니다.
- 콘솔에서 로그를 보는 데 필요한 권한은 <u>CodePipeline 콘솔에서 컴퓨팅 로그를 보는 데 필요한 권</u> 한에 자세히 설명되어 있습니다.
- CodePipeline의 다른 작업과 달리 작업 구성에서는 필드를 설정하지 않고 작업 구성 외부에서 작업 구성 필드를 설정합니다.

서비스 역할 정책 권한

CodePipeline이 작업을 실행하면 CodePipeline은 다음과 같이 파이프라인 이름을 사용하여 로그 그룹 을 생성합니다. 이렇게 하면 파이프라인 이름을 사용하여 리소스를 로깅할 수 있는 권한을 범위 아래로 지정할 수 있습니다.

/aws/codepipeline/MyPipelineName

기존 서비스 역할을 사용하는 경우 명령 작업을 사용하려면 서비스 역할에 대해 다음 권한을 추가해야 합니다.

- logs:CreateLogGroup
- logs:CreateLogStream
- logs:PutLogEvents

서비스 역할 정책 설명에서 다음 예제와 같이 파이프라인 수준까지 권한의 범위를 좁힙니다.

```
{
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
    ],
        "Resource": [
            "arn:aws:logs:*:YOUR_AWS_ACCOUNT_ID:log-group:/aws/
codepipeline/YOUR_PIPELINE_NAME",
            "arn:aws:logs:*:YOUR_AWS_ACCOUNT_ID:log-group:/aws/
codepipeline/YOUR_PIPELINE_NAME:*"
    ]
}
```

작업 세부 정보 대화 상자를 사용하여 콘솔에서 로그를 보려면 로그를 볼 수 있는 권한을 콘솔 역할에 추가해야 합니다. 자세한 내용은 <u>CodePipeline 콘솔에서 컴퓨팅 로그를 보는 데 필요한 권한</u>에서 콘솔 권한 정책 예제를 참조하세요.

작업 유형

• 범주: Compute

- 소유자: AWS
- 공급자: Commands
- 버전:1

구성 파라미터

명령

필수 여부: 예

실행할 Commands 작업에 쉘 명령을 제공할 수 있습니다. 콘솔에서 명령은 별도의 줄에 입력됩니다. CLI에서 명령은 별도의 문자열로 입력됩니다.

Note

다중 라인 형식은 지원되지 않으며 오류 메시지가 표시됩니다. 명령 필드에 명령을 입력하 려면 단일 줄 형식을 사용해야 합니다.

🛕 Important

EnvironmentType 및 ComputeType 값은 CodeBuild의 값과 일치합니다. 사용 가능한 유형 의 하위 집합을 지원합니다. 자세한 내용은 빌드 환경 컴퓨팅 유형을 참조하십시오.

EnvironmentType

필수 여부: 아니요

명령 작업을 지원하는 빌드 환경의 OS 이미지입니다. 다음은 빌드 환경에 유효한 값입니다.

- LINUX_CONTAINER
- WINDOWS_SERVER_2022_컨테이너

그러면 EnvironmentType을 선택하면 ComputeType 필드에서 해당 OS의 컴퓨팅 유형이 허용됩니 다. 이 작업에 사용할 수 있는 CodeBuild 컴퓨팅 유형에 대한 자세한 내용은 CodeBuild 사용 설명서 의 <u>빌드 환경 컴퓨팅 모드 및 유형</u> 참조를 참조하세요.

1 Note

지정하지 않으면 빌드 환경의 컴퓨팅 기본값은 다음과 같습니다.

- 컴퓨팅 유형: BUILD_GENERAL1_SMALL
- 환경 유형: LINUX_CONTAINER

ComputeType

필수 여부: 아니요

EnvironmentType에 대한 선택에 따라 컴퓨팅 유형을 제공할 수 있습니다. 다음은 컴퓨팅에 사용할 수 있는 값이지만 사용 가능한 옵션은 OS에 따라 다를 수 있습니다.

- BUILD_GENERAL1_SMALL
- BUILD_GENERAL1_MEDIUM
- BUILD_GENERAL1_LARGE

▲ Important

일부 컴퓨팅 유형은 특정 환경 유형과 호환되지 않습니다. 예를 들어 WINDOWS_SERVER_2022_CONTAINER는 BUILD_GENERAL1_SMALL과 호환되지 않 습니다. 호환되지 않는 조합을 사용하면 작업이 실패하고 런타임 오류가 발생합니다.

outputVariables

필수 여부: 아니요

내보낼 환경의 변수 이름을 지정합니다. CodeBuild 환경 변수에 대한 참조는 <u>CodeBuild 사용 설명</u> 서의 빌드 환경의 환경 변수를 참조하세요.

파일

필수 여부: 아니요

작업에 대한 출력 아티팩트로 내보내려는 파일을 제공할 수 있습니다.

지원되는 파일 형식은 CodeBuild 파일 패턴과 동일합니다. 예를 들어 모든 파일에 **/를 입력합니다. 자세한 내용은 CodeBuild 사용 설명서의 CodeBuild용 빌드 사양 참조를 참조하세요.

Edit action			×
Action name Choose a name for your action			
Commands			
No more than 100 characters			
Action provider			
Commands		▼	
Input artifacts Choose an input artifact for this action. Learn more	12		
	_	~	
SourceArtifact V			
Defined by: Source			
No more than 100 characters			
Commands			
Specify the shell commands to run with your compu- separate charges in AWS CodeBuild.	te action in CodePipeline. You do not need to create a	ny resources in CodeBuild. Note: Using compute time	for this action will incur
ls			
echo "Hello World"			
Environment variables - optional Key value pair that is supplied to actions with mana	ged compute.		
Name	Value		
bedrock_api_token	api_keys:bedrock		
Type			
Secrets Manager			
Secrets Manager			
	Remove		
Add on vivon mont verifield			
Add environment variable			
VPC ID			
Specify the VPC ID that your compute action will us	2.		
3			
Additional configuration Environment type compute			
Environment Type			
Compute		Ť	
 3 GB memory, 2 vCPUs 			
O 7 GB memory, 4 vCPUs			
15 GB memory, 8 vCPUs			
Variable namerace entioned			
Choose a namespace for the output variables from the output variables	his action. You must choose a namespace if you want	to use the variables this action produces in your confi	guration. Learn more [🛃
buildcommands			
		G	ancel Done

Vpcld

필수 여부: 아니요

리소스의 VPC ID입니다.

```
서브넷
```

필수 여부: 아니요

VPC의 서브넷입니다. 이 필드는 명령이 VPC의 리소스에 연결해야 하는 경우에 필요합니다. SecurityGroupIds

필수 여부: 아니요

VPC의 보안 그룹입니다. 이 필드는 명령이 VPC의 리소스에 연결해야 하는 경우에 필요합니다.

다음은 예제 환경 변수와 함께 환경 및 컴퓨팅 유형에 대한 구성 필드가 표시된 작업의 JSON 예제입니 다.

```
{
           "name": "Commands1",
           "actionTypeId": {
             "category": "Compute",
             "owner": "AWS",
             "provider": "Commands",
             "version": "1"
           },
           "inputArtifacts": [
             {
               "name": "SourceArtifact"
             }
           ],
           "commands": [
             "ls",
             "echo hello",
             "echo $BEDROCK_TOKEN",
           ],
           "configuration": {
             "EnvironmentType": "LINUX_CONTAINER",
             "ComputeType": "BUILD_GENERAL1_MEDIUM"
           },
           "environmentVariables": [
```

```
{
    "name": "BEDROCK_TOKEN",
    "value": "apiTokens:bedrockToken",
    "type": "SECRETS_MANAGER"
    }
],
    "runOrder": 1
}
```

입력 아티팩트

• 아티팩트 수:1 to 10

출력 아티팩트

• 아티팩트 수: 0 to 1

환경 변수

키

와 같은 키-값 환경 변수 페어의 키입니다BEDROCK_TOKEN.

값

와 같은 키-값 페어의 값입니다apiTokens:bedrockToken. 파이프라인 작업 또는 파이프라인 변 수의 출력 변수를 사용하여 값을 파라미터화할 수 있습니다.

SECRETS_MANAGER 유형을 사용할 때이 값은 AWS Secrets Manager에 이미 저장한 보안 암호의 이름이어야 합니다.

유형

환경 변수 값의 사용 유형을 지정합니다. 이때 값은 PLAINTEXT 또는 SECRETS_MANAGER가 될 수 있습니다. 값이 인 경우 EnvironmentVariable 값에 보안 암호 참조를 SECRETS_MANAGER제공 합니다. 지정하지 않으면 기본적으로 PLAINTEXT가 사용됩니다.

Note

민감한 값, 특히 AWS 자격 증명을 저장하기 위해 일반 텍스트 환경 변수를 사용하지 않는 것이 좋습니다. CodeBuild 콘솔 또는를 사용하면 AWS CLI일반 텍스트 환경 변수가 일반

텍스트로 표시됩니다. 중요한 값의 경우 대신 SECRETS_MANAGER 유형을 사용하는 것이 좋습니다.

Note

환경 변수 구성에 대해 name, value 및 type을 입력할 때, 특히 환경 변수에 CodePipeline 출 력 변수 구문이 포함된 경우 구성 값 필드에 1000자 제한을 초과하지 마십시오. 이 제한을 초과 하면 확인 오류가 반환됩니다.

환경 변수를 보여주는 예제 작업 선언은 섹션을 참조하세요구성 파라미터.

1 Note

- SECRETS_MANAGER 유형은 명령 작업에서만 지원됩니다.
- 명령 작업에서 참조된 보안 암호는 CodeBuild와 유사한 빌드 로그에서 수정됩니다. 그러나 파이프라인에 대한 편집 액세스 권한이 있는 파이프라인 사용자는 명령을 수정하여 이러한 보안 암호 값에 계속 액세스할 수 있습니다.
- SecretsManager를 사용하려면 파이프라인 서비스 역할에 다음 권한을 추가해야 합니다.

```
{
    "Effect": "Allow",
    "Action": [
        "secretsmanager:GetSecretValue"
    ],
    "Resource": [
        "SECRET_ARN"
    ]
}
```

서비스 역할 권한: 명령 작업

명령 지원의 경우 정책 설명에 다음을 추가합니다.

```
"Version": "2012-10-17",
```

{

```
"Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:{{region}}:{{customerAccountId}}:log-group:/aws/
codepipeline/{{pipelineName}}",
                "arn:aws:logs:{{region}}:{{customerAccountId}}:log-group:/aws/
codepipeline/{{pipelineName}}:log-stream:*"
            1
        }
    ]
}
```

작업 선언(예)

YAML

```
name: Commands_action
actionTypeId:
 category: Compute
 owner: AWS
  provider: Commands
  version: '1'
runOrder: 1
configuration: {}
commands:
- ls
- echo hello
- 'echo pipeline Execution Id is #{codepipeline.PipelineExecutionId}'
outputArtifacts:
- name: BuildArtifact
 files:
  - **/
inputArtifacts:
- name: SourceArtifact
outputVariables:
- AWS_DEFAULT_REGION
region: us-east-1
```

namespace: compute

JSON

```
{
    "name": "Commands_action",
    "actionTypeId": {
        "category": "Compute",
        "owner": "AWS",
        "provider": "Commands",
        "version": "1"
    },
    "runOrder": 1,
    "configuration": {},
    "commands": [
        "ls",
        "echo hello",
        "echo pipeline Execution Id is #{codepipeline.PipelineExecutionId}"
    ],
    "outputArtifacts": [
        {
            "name": "BuildArtifact",
            "files": [
                "**/"
            ]
        }
    ],
    "inputArtifacts": [
        {
            "name": "SourceArtifact"
        }
    ],
    "outputVariables": [
        "AWS_DEFAULT_REGION"
    ],
    "region": "us-east-1",
    "namespace": "compute"
}
```

다음 사항도 참조하세요.

이 작업을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

• <u>자습서: 컴퓨팅을 사용하여 명령을 실행하는 파이프라인 생성(V2 유형)</u> - 이 자습서에서는 명령 작업 이 포함된 샘플 파이프라인을 제공합니다.

AWS Device Farm 테스트 작업 참조

파이프라인에서가 디바이스에서 애플리케이션을 실행하고 테스트하는 AWS Device Farm 데 사용하 는 테스트 작업을 구성할 수 있습니다. Device Farm은 디바이스의 테스트 풀과 테스트 프레임워크를 사용하여 특정 디바이스에서 애플리케이션을 테스트합니다. Device Farm 작업에서 지원하는 테스트 프레임워크 유형에 대한 자세한 내용은 <u>Work with Test Types in AWS Device Farm</u>을 참조하세요.

주제

- 작업 유형
- 구성 파라미터
- 입력 아티팩트
- <u>출력 아티팩트</u>
- 서비스 역할 권한: AWS Device Farm 작업
- <u>작업 선언</u>
- 다음 사항도 참조하세요.

작업 유형

- 범주: Test
- 소유자: AWS
- 공급자:DeviceFarm
- 버전:1

구성 파라미터

AppType

필수 여부: 예

테스트 중인 애플리케이션의 OS 및 유형입니다. 유효한 값 목록은 다음과 같습니다.

• i0S

- Android
- Web

ProjectId

필수 여부: 예

Device Farm 프로젝트 ID입니다.

프로젝트 ID를 찾으려면 Device Farm 콘솔에서 프로젝트를 선택합니다. 브라우저에 서 새 프로젝트의 URL을 복사합니다. URL에 프로젝트 ID가 포함되어 있습니다. 프 로젝트 ID는 projects/ 뒤에 있는 URL의 값입니다. 다음 예제에서 프로젝트 ID는 eec4905f-98f8-40aa-9afc-4c1cfexample입니다.

```
https://<region-URL>/devicefarm/home?region=us-west-2#/projects/
eec4905f-98f8-40aa-9afc-4c1cfexample/runs
```

앱

필수 여부: 예

입력 아티팩트에 있는 애플리케이션 파일의 이름과 위치입니다. 예: s3-ios-test-1.ipa TestSpec

조건부: 예

입력 아티팩트에 있는 테스트 사양 정의 파일의 위치입니다. 이는 사용자 정의 모드 테스트에 필요 합니다.

DevicePoolArn

필수 여부: 예

Device Farm 디바이스 풀 ARN.

상위 디바이스에 대한 ARNs을 포함하여 프로젝트에 사용 가능한 디바이스 풀 ARN을 가져오려면 AWS CLI를 사용하여 다음 명령을 입력합니다.

```
aws devicefarm list-device-pools --arn arn:aws:devicefarm:us-
west-2:account_ID:project:project_ID
```

TestType

필수 여부: 예

테스트에 지원되는 테스트 프레임워크를 지정합니다. TestType의 유효한 값 목록은 다음과 같습 니다.

- APPIUM_JAVA_JUNIT
- APPIUM_JAVA_TESTNG
- APPIUM_NODE
- APPIUM_RUBY
- APPIUM_PYTHON
- APPIUM_WEB_JAVA_JUNIT
- APPIUM_WEB_JAVA_TESTNG
- APPIUM_WEB_NODE
- APPIUM_WEB_RUBY
- APPIUM_WEB_PYTHON
- BUILTIN_FUZZ
- INSTRUMENTATION
- XCTEST
- XCTEST_UI

1 Note

CodePipeline의 작업에서 지원되지 않는 테스트 유형은 WEB_PERFORMANCE_PROFILE, REMOTE_ACCESS_RECORD 및 REMOTE_ACCESS_REPLAY입니다.

Device Farm 테스트 유형에 대한 자세한 내용은 <u>AWS Device Farm에서 테스트 유형을 사용한 작</u> 업을 참조하세요.

RadioBluetoothEnabled

필수 여부: 아니요

테스트 시작 시 Bluetooth를 활성화할지 여부를 나타내는 부울 값입니다.

RecordAppPerformanceData

필수 여부: 아니요

테스트 중에 CPU, FPS, 메모리 성능과 같은 기기 성능 데이터를 기록할지 여부를 나타내는 부울 값 입니다.
RecordVideo

필수 여부: 아니요

테스트 중에 비디오를 녹화할지 여부를 나타내는 부울 값입니다.

RadioWifiEnabled

필수 여부: 아니요

테스트 시작 시 Wi-Fi를 활성화할지 여부를 나타내는 부울 값입니다.

RadioNfcEnabled

필수 여부: 아니요

테스트 시작 시 NFC를 활성화할지 여부를 나타내는 부울 값입니다.

RadioGpsEnabled

필수 여부: 아니요

테스트 시작 시 GPS를 활성화할지 여부를 나타내는 부울 값입니다.

테스트

필수 여부: 아니요

소스 위치에 있는 테스트 정의 파일의 이름 및 경로입니다. 경로는 테스트용 입력 아티팩트의 루트 와 상대적입니다.

FuzzEventCount

필수 여부: 아니요

fuzz 테스트가 수행할 사용자 인터페이스 이벤트 수로, 1에서 10,000 사이입니다.

FuzzEventThrottle

필수 여부: 아니요

다음 사용자 인터페이스 이벤트를 수행하기 전에 fuzz 테스트가 대기할 시간을 밀리초 단위로 나타 내는 숫자로, 1에서 1,000 사이입니다.

FuzzRandomizerSeed

필수 여부: 아니요

사용자 인터페이스 이벤트를 무작위화하는 데 사용할 fuzz 테스트의 시드입니다. 후속 fuzz 테스트 에 동일한 숫자를 사용하면 동일한 이벤트 시퀀스가 생성됩니다. CustomHostMachineArtifacts

필수 여부: 아니요

사용자 지정 아티팩트가 저장될 호스트 시스템의 위치입니다.

CustomDeviceArtifacts

필수 여부: 아니요

사용자 지정 아티팩트가 저장될 디바이스의 위치입니다.

UnmeteredDevicesOnly

필수 여부: 아니요

이 단계에서 테스트를 실행할 때 무제한 디바이스만 사용할지 여부를 나타내는 부울 값입니다.

JobTimeoutMinutes

필수 여부: 아니요

시간 초과되기 전에 디바이스당 테스트 실행이 실행되는 시간(분)입니다.

위도

필수 여부: 아니요

지리 좌표계 각도로 표시되는 디바이스의 위도입니다.

경도

필수 여부: 아니요

지리 좌표계 각도로 표시되는 디바이스의 경도입니다.

입력 아티팩트

- 아티팩트 수: 1
- 설명: 테스트 작업에서 사용할 수 있도록 할 아티팩트 세트입니다. Device Farm은 사용할 빌드된 애 플리케이션과 테스트 정의를 찾습니다.

출력 아티팩트

• 아티팩트 수: 0

입력 아티팩트

• 설명: 이 작업 유형에는 출력 아티팩트가 적용되지 않습니다.

서비스 역할 권한: AWS Device Farm 작업

CodePipeline이 작업을 실행할 때 CodePipeline 서비스 역할 정책에는 최소 권한으로 액세스를 유지하 기 위해 파이프라인 리소스 ARN으로 범위가 적절하게 축소된 다음 권한이 필요합니다. 예를 들어 정책 설명에 다음을 추가합니다.

```
{
    "Effect": "Allow",
    "Action": [
        "devicefarm:ListProjects",
        "devicefarm:GetRun",
        "devicefarm:GetUpload",
        "devicefarm:CreateUpload",
        "devicefarm:ScheduleRun"
    ],
    "Resource": "resource_ARN"
},
```

작업 선언

YAML

```
Name: Test
Actions:
  - Name: TestDeviceFarm
   ActionTypeId: null
   category: Test
    owner: AWS
    provider: DeviceFarm
    version: '1'
RunOrder: 1
Configuration:
 App: s3-ios-test-1.ipa
 AppType: iOS
  DevicePoolArn: >-
    arn:aws:devicefarm:us-west-2::devicepool:0EXAMPLE-d7d7-48a5-ba5c-b33d66efa1f5
  ProjectId: eec4905f-98f8-40aa-9afc-4c1cfEXAMPLE
 TestType: APPIUM_PYTHON
```

```
TestSpec: example-spec.yml
OutputArtifacts: []
InputArtifacts:
- Name: SourceArtifact
Region: us-west-2
```

JSON

```
{
    "Name": "Test",
    "Actions": [
        {
            "Name": "TestDeviceFarm",
            "ActionTypeId": null,
            "category": "Test",
            "owner": "AWS",
            "provider": "DeviceFarm",
            "version": "1"
        }
    ],
    "RunOrder": 1,
    "Configuration": {
        "App": "s3-ios-test-1.ipa",
        "AppType": "iOS",
        "DevicePoolArn": "arn:aws:devicefarm:us-west-2::devicepool:0EXAMPLE-
d7d7-48a5-ba5c-b33d66efa1f5",
        "ProjectId": "eec4905f-98f8-40aa-9afc-4c1cfEXAMPLE",
        "TestType": "APPIUM_PYTHON",
        "TestSpec": "example-spec.yml"
    },
    "OutputArtifacts": [],
    "InputArtifacts": [
        {
            "Name": "SourceArtifact"
        }
    ],
    "Region": "us-west-2"
},
```

다음 사항도 참조하세요.

이 작업을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

사용자 가이드

- <u>Device Farm의 테스트 유형 작업</u> Device Farm 개발자 안내서의 이 참조 장에서는 Device Farm에 서 지원하는 Android, iOS 및 웹 애플리케이션 테스트 프레임워크에 대한 자세한 설명을 제공합니다.
- <u>Device Farm의 작업</u> Device Farm API 참조의 API 직접 호출 및 파라미터는 Device Farm 프로젝트 작업에 도움이 될 수 있습니다.
- <u>자습서:를 사용하여 Android 앱을 빌드하고 테스트하는 파이프라인 생성 AWS Device Farm</u> 이 자 습서는 CodeBuild 및 Device Farm을 사용하여 Android 앱을 빌드 및 테스트하는 GitHub 소스가 있 는 파이프라인을 생성하기 위한 샘플 빌드 사양 파일과 샘플 애플리케이션을 제공합니다.
- <u>자습서:를 사용하여 iOS 앱을 테스트하는 파이프라인 생성 AWS Device Farm</u> 이 자습서에서는 Device Farm으로 빌드된 iOS 앱을 테스트하는 Amazon S3 소스로 파이프라인을 생성하는 샘플 애 플리케이션을 제공합니다.

Elastic Beanstalk 배포 작업 참조

Elastic Beanstalk는 웹 애플리케이션을 배포하고 확장 AWS 하는 데 사용되는 내의 플랫폼입니다. Elastic Beanstalk 작업을 사용하여 배포 환경에 애플리케이션 코드를 배포합니다.

주제

- <u>작업 유형</u>
- <u>구성 파라미터</u>
- 입력 아티팩트
- 출력 아티팩트
- 서비스 역할 권한: ElasticBeanstalk 배포 작업
- <u>작업 선언</u>
- 다음 사항도 참조하세요.

작업 유형

- 범주: Deploy
- 소유자: AWS
- 공급자: ElasticBeanstalk
- 버전:1

구성 파라미터

ApplicationName

필수 여부: 예

Elastic Beanstalk에서 생성한 애플리케이션의 이름입니다.

EnvironmentName

필수 여부: 예

Elastic Beanstalk에서 생성한 환경의 이름입니다. 환경은 애플리케이션 버전을 실행하는 AWS 리 소스 모음입니다. 각 환경은 한 번에 하나의 애플리케이션 버전만 실행하지만 여러 환경에서 동일 한 애플리케이션 버전 또는 서로 다른 애플리케이션 버전을 동시에 실행할 수 있습니다.

입력 아티팩트

- 아티팩트 수: 1
- 설명: 작업에 대한 입력 아티팩트입니다.

출력 아티팩트

- 아티팩트 수: 0
- 설명: 이 작업 유형에는 출력 아티팩트가 적용되지 않습니다.

서비스 역할 권한: ElasticBeanstalk 배포 작업

Elastic Beanstalk의 경우 ElasticBeanstalk 배포 작업을 통해 파이프라인을 만드는 데 필요한 최소 권한은 다음과 같습니다.

```
{
    "Effect": "Allow",
    "Action": [
        "elasticbeanstalk:*",
        "ec2:*",
        "elasticloadbalancing:*",
        "autoscaling:*",
```

```
"cloudwatch:*",
    "s3:*",
    "sns:*",
    "cloudformation:*",
    "rds:*",
    "sqs:*",
    "ecs:*"
],
    "Resource": "resource_ARN"
},
```

1 Note

리소스 정책의 와일드카드를 액세스를 제한하려는 계정의 리소스로 바꿔야 합니다. 최소 권한 액세스 권한을 부여하는 정책 생성에 대한 자세한 내용은 <u>https://docs.aws.amazon.com/IAM/</u> latest/UserGuide/best-practices.html#grant-least-privilege 단원을 참조하세요.

작업 선언

YAML

```
Name: Deploy
Actions:
  - Name: Deploy
   ActionTypeId:
      Category: Deploy
      Owner: AWS
      Provider: ElasticBeanstalk
      Version: '1'
    RunOrder: 1
    Configuration:
      ApplicationName: my-application
      EnvironmentName: my-environment
    OutputArtifacts: []
    InputArtifacts:
      - Name: SourceArtifact
    Region: us-west-2
    Namespace: DeployVariables
```

JSON

```
{
    "Name": "Deploy",
    "Actions": [
        {
            "Name": "Deploy",
            "ActionTypeId": {
                "Category": "Deploy",
                "Owner": "AWS",
                "Provider": "ElasticBeanstalk",
                "Version": "1"
            },
            "RunOrder": 1,
            "Configuration": {
                "ApplicationName": "my-application",
                "EnvironmentName": "my-environment"
            },
            "OutputArtifacts": [],
            "InputArtifacts": [
                {
                     "Name": "SourceArtifact"
                }
            ],
            "Region": "us-west-2",
            "Namespace": "DeployVariables"
        }
    ]
},
```

다음 사항도 참조하세요.

이 작업을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

• Elastic Beanstalk에 Flask 애플리케이션 배포 -이 자습서에서는 샘플 Flask 애플리케이션을 사용하 여 Elastic Beanstalk에서 애플리케이션 및 환경 리소스를 생성하는 방법을 안내합니다. 그런 다음 소 스 리포지토리에서 Elastic Beanstalk 환경으로 애플리케이션을 배포하는 Elastic Beanstalk 배포 작 업을 사용하여 파이프라인을 빌드할 수 있습니다.

Amazon Inspector **InspectorScan** 호출 작업 참조

Amazon Inspector는 워크로드를 자동으로 검색하고 소프트웨어 취약성 및 의도하지 않은 네트워크 노 출이 있는지 지속적으로 스캔하는 취약성 관리 서비스입니다. CodePipeline의 InspectorScan 작업 은 오픈 소스 코드의 보안 취약성 감지 및 수정을 자동화합니다. 작업은 보안 스캔 기능이 있는 관리형 컴퓨팅 작업입니다. InspectorScan을 GitHub 또는 Bitbucket Cloud와 같은 타사 리포지토리의 애플리 케이션 소스 코드 또는 컨테이너 애플리케이션용 이미지와 함께 사용할 수 있습니다. 작업은 사용자가 구성한 취약성 수준과 경고를 스캔하고 보고합니다.

🛕 Important

이 작업은 CodePipeline 관리형 CodeBuild 컴퓨팅을 사용하여 빌드 환경에서 명령을 실행합니다. 작업을 실행하면 별도의 요금이 발생합니다 AWS CodeBuild.

주제

- <u>작업 유형 ID</u>
- 구성 파라미터
- 입력 아티팩트
- 출력 아티팩트
- 출력 변수
- 서비스 역할 권한: InspectorScan 작업
- <u>작업 선언</u>
- 다음 사항도 참조하세요.

작업 유형 ID

- 범주: Invoke
- 소유자: AWS
- 공급자: InspectorScan
- 버전:1

예시

```
{
    "Category": "Invoke",
    "Owner": "AWS",
    "Provider": "InspectorScan",
    "Version": "1"
},
```

구성 파라미터

InspectorRunMode

(필수) 스캔 모드를 나타내는 문자열입니다. 유효한 값은 SourceCodeScan | ECRImageScan입니다.

ECRRepositoryName

이미지가 푸시된 Amazon ECR 리포지토리의 이름입니다.

ImageTag

이미지에 사용되는 태그입니다.

이 작업의 파라미터는 지정한 취약성 수준을 검사합니다. 취약성 임계값에 사용할 수 있는 수준은 다음 과 같습니다.

CriticalThreshold

CodePipeline이 작업에 실패해야 하는 소스에서 발견된 중요한 심각도 취약성 수입니다. HighThreshold

CodePipeline이 작업에 실패해야 하는 소스에서 발견된 심각도가 높은 취약성 수입니다.

MediumThreshold

CodePipeline이 작업에 실패해야 하는 소스에서 발견된 중간 심각도 취약성 수입니다.

LowThreshold

CodePipeline이 작업에 실패해야 하는 소스에서 발견된 심각도가 낮은 취약성 수입니다.

Action name

Choose a name for your action

Scan

No more than 100 characters

Action provider

AWS InspectorScan

Region

US East (N. Virginia)

Input artifacts

Choose an input artifact for this action. Learn more 🛂

SourceArtifact X Defined by: Source

No more than 100 characters

Inspector Run mode

InspectorRunMode: SourceCodeScan or ECRImageScan.

O Source Code Scan

SourceCodeScan: Scan the source code. Choose an input artifact for this run mode.

ECR Image Scan
 ECRImageScan: Scan the ECR image.

T

Critical Threshold - optional

Threshold value for critical severity vulnerabilities to allow the action to succeed. Default: Integer.MAX_VALUE

High Threshold - optional

Threshold value for high severity vulnerabilities to allow the action to succeed. Default: Integer.MAX_VALUE

Medium Threshold - optional

Threshold value for medium severity vulnerabilities to allow the action to succeed. Default: Integer.MAX_VALUE

Low Threshold - optional

Threshold value for low severity vulnerabilities to allow the action to succeed. Default: Integer.MAX_VALUE

T

입력 아티팩트

- 아티팩트 수: 1
- 설명: 취약성을 검사할 소스 코드입니다. 스캔이 ECR 리포지토리용인 경우이 입력 아티팩트는 필요 하지 않습니다.

출력 아티팩트

- 아티팩트 수: 1
- 설명: 소프트웨어 재료표(SBOM) 파일 형식의 소스 취약성 세부 정보입니다.

출력 변수

이 작업을 구성하면 파이프라인에서 다운스트림 작업의 작업 구성에서 참조할 수 있는 변수가 생성됩 니다. 이 작업은 작업에 네임스페이스가 없는 경우에도 출력 변수로 볼 수 있는 변수를 생성합니다. 이 러한 변수를 다운스트림 작업 구성에서 사용할 수 있도록 네임스페이스를 사용하여 작업을 구성합니 다.

자세한 내용은 변수 참조 단원을 참조하십시오.

HighestScannedSeverity

스캔의 가장 높은 심각도 출력입니다. 유효한 값은 medium | high | critical입니다.

서비스 역할 권한: InspectorScan 작업

InspectorScan 작업 지원을 위해 정책 설명에 다음을 추가합니다.

```
{
    "Effect": "Allow",
    "Action": "inspector-scan:ScanSbom",
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
    "
```

```
"ecr:BatchCheckLayerAvailability"
],
"Resource": "resource_ARN"
},
```

또한 명령 작업에 아직 추가되지 않은 경우 서비스 역할에 다음 권한을 추가하여 CloudWatch 로그를 봅니다.



Note

서비스 역할 정책 문의 리소스 기반 권한을 사용하여 파이프라인 리소스 수준으로 권한 범위를 좁힙니다.

작업 선언

YAML

```
name: Scan
actionTypeId:
   category: Invoke
   owner: AWS
   provider: InspectorScan
   version: '1'
runOrder: 1
configuration:
   InspectorRunMode: SourceCodeScan
outputArtifacts:
- name: output
inputArtifacts:
- name: SourceArtifact
```

region: us-east-1

JSON

```
{
                         "name": "Scan",
                         "actionTypeId": {
                             "category": "Invoke",
                             "owner": "AWS",
                             "provider": "InspectorScan",
                             "version": "1"
                         },
                         "runOrder": 1,
                         "configuration": {
                             "InspectorRunMode": "SourceCodeScan"
                         },
                         "outputArtifacts": [
                             {
                                 "name": "output"
                             }
                         ],
                         "inputArtifacts": [
                             {
                                  "name": "SourceArtifact"
                             }
                         ],
                         "region": "us-east-1"
                     },
```

다음 사항도 참조하세요.

이 작업을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

• Amazon Inspector에 대한 자세한 내용은 Amazon Inspector 사용 설명서를 참조하세요.

AWS Lambda 작업 참조 호출

Lambda 함수를 파이프라인의 작업으로 실행할 수 있습니다. 이 함수에 대한 입력인 이벤트 객체를 사용하면 함수가 작업 구성, 입력 아티팩트 위치, 출력 아티팩트 위치, 및 아티팩트에 액세스하는 데 필요한 기타 정보에 액세스할 수 있습니다. Lambda 간접 호출 함수에 전달되는 예제 이벤트는 예제 JSON 이벤트 단원을 참조하세요. Lambda 함수 구현의 일환으로 <u>PutJobSuccessResult</u> API 또는 <u>PutJobFailureResult</u> API에 대한 호출이 있어야 합니다. 그렇지 않으면 작업이 시간 초과될 때까 지 이 작업의 실행이 중단됩니다. 작업에 대한 출력 아티팩트를 지정하면 함수 구현의 일환으로 해당 아티팩트를 S3 버킷에 업로드해야 합니다.

A Important

CodePipeline이 Lambda로 전송하는 JSON 이벤트를 기록하지 마십시오. 이렇게 하면 사용자 보안 인증이 CloudWatch Logs에 기록될 수 있기 때문입니다. CodePipeline 역할은 JSON 이 벤트를 사용하여 artifactCredentials 필드의 Lambda에 임시 보안 인증을 전달합니다. 예제 이벤트는 예제 JSON 이벤트을 참조하세요.

작업 유형

- 범주: Invoke
- 소유자: AWS
- 공급자:Lambda
- 버전:1

구성 파라미터

FunctionName

필수 여부: 예

FunctionName은 Lambda에서 생성된 함수의 이름입니다.

UserParameters

필수 여부: 아니요

Lambda 함수에서 입력으로 처리할 수 있는 문자열입니다.

입력 아티팩트

- 아티팩트 수:0 to 5
- 설명: Lambda 함수에서 사용할 수 있도록 할 아티팩트 세트입니다.

출력 아티팩트

- 아티팩트 수: 0 to 5
- 설명: Lambda 함수에서 출력으로 생성한 아티팩트 세트입니다.

출력 변수

이 작업을 수행하면 <u>PutJobSuccessResult API</u> 요청의 outputVariables 섹션에 포함된 모든 키-값 페어가 변수로 생성됩니다.

CodePipeline의 변수에 대한 자세한 내용은 변수 참조 단원을 참조하세요.

예제 작업 구성

YAML

```
Name: Lambda
Actions:
    - Name: Lambda
    ActionTypeId:
        Category: Invoke
        Owner: AWS
        Provider: Lambda
        Version: '1'
    RunOrder: 1
    Configuration:
        FunctionName: myLambdaFunction
        UserParameters: 'http://192.0.2.4'
    OutputArtifacts: []
    InputArtifacts: []
    Region: us-west-2
```

```
{
    "Name": "Lambda",
    "Actions": [
        {
            "Name": "Lambda",
            "ActionTypeId": {
                "Category": "Invoke",
```

```
"Owner": "AWS",
    "Provider": "Lambda",
    "Version": "1"
    },
    "RunOrder": 1,
    "Configuration": {
        "FunctionName": "myLambdaFunction",
        "UserParameters": "http://192.0.2.4"
    },
    "OutputArtifacts": [],
    "InputArtifacts": [],
    "Region": "us-west-2"
    }
]
```

예제 JSON 이벤트

Lambda 작업은 작업 ID, 파이프라인 작업 구성, 입력 및 출력 아티팩트 위치, 아티팩트에 대한 모든 암 호화 정보가 포함된 JSON 이벤트를 전송합니다. 작업자가 이 세부 정보에 액세스하여 Lambda 작업을 완료합니다. 자세한 내용은 <u>작업 세부 정보</u>를 참조하십시오. 다음은 이벤트 예제입니다.

```
{
    "CodePipeline.job": {
        "id": "11111111-abcd-1111-abcd-111111abcdef",
        "accountId": "111111111111",
        "data": {
            "actionConfiguration": {
                "configuration": {
                    "FunctionName": "MyLambdaFunction",
                    "UserParameters": "input_parameter"
                }
            },
            "inputArtifacts": [
                {
                    "location": {
                         "s3Location": {
                             "bucketName": "bucket_name",
                             "objectKey": "filename"
                         },
                        "type": "S3"
                    },
```

```
"revision": null,
                     "name": "ArtifactName"
                }
            ],
            "outputArtifacts": [],
            "artifactCredentials": {
                "secretAccessKey": "secret_key",
                "sessionToken": "session_token",
                "accessKeyId": "access_key_ID"
            },
            "continuationToken": "token_ID",
            "encryptionKey": {
              "id": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
              "type": "KMS"
            }
        }
    }
}
```

JSON 이벤트는 CodePipeline의 Lambda 작업에 대한 다음과 같은 작업 세부 정보를 제공합니다.

- id: 작업의 고유한 시스템 생성 ID입니다.
- account Id: 작업과 연결된 AWS 계정 ID입니다.
- data: 작업자가 작업을 완료하는 데 필요한 기타 정보입니다.
 - actionConfiguration: Lambda 작업에 대한 작업 파라미터입니다. 정의는 <u>구성 파라미터</u> 단 원을 참조하십시오.
 - inputArtifacts: 작업에 제공된 아티팩트입니다.
 - location: 아티팩트 스토어 위치입니다.
 - s3Location: 작업에 대한 입력 아티팩트 위치 정보입니다.
 - bucketName: 작업에 대한 파이프라인 아티팩트 스토어의 이름입니다(예: 이름이 codepipeline-us-east-2-1234567890인 Amazon S3 버킷).
 - objectKey: 애플리케이션 이름입니다(예: CodePipelineDemoApplication.zip).
 - type: 위치에서 아티팩트의 유형입니다. 현재 유효한 아티팩트 유형은 S3뿐입니다.
 - revision: 아티팩트의 개정 ID입니다. 객체 유형에 따라 커밋 ID(GitHub) 또는 개정 ID(Amazon Simple Storage Service)일 수 있습니다. 자세한 내용은 <u>ArtifactRevision</u>을 참조하십 시오.
 - name: 작업할 아티팩트의 이름입니다(예: MyApp).

- outputArtifacts: 작업의 출력입니다.
 - location: 아티팩트 스토어 위치입니다.
 - s3Location: 작업에 대한 출력 아티팩트 위치 정보입니다.
 - bucketName: 작업에 대한 파이프라인 아티팩트 스토어의 이름입니다(예: 이름이 codepipeline-us-east-2-1234567890인 Amazon S3 버킷).
 - objectKey: 애플리케이션 이름입니다(예: CodePipelineDemoApplication.zip).
 - type: 위치에서 아티팩트의 유형입니다. 현재 유효한 아티팩트 유형은 S3뿐입니다.
 - revision: 아티팩트의 개정 ID입니다. 객체 유형에 따라 커밋 ID(GitHub) 또는 개정 ID(Amazon Simple Storage Service)일 수 있습니다. 자세한 내용은 <u>ArtifactRevision</u>을 참조하십 시오.
 - name: 아티팩트의 출력 이름입니다(예: MyApp).
- artifactCredentials: Amazon S3 버킷의 입력 및 출력 아티팩트에 액세스하는 데 사용되는 AWS 세션 자격 증명입니다. 이러한 자격 증명은 AWS Security Token Service (AWS STS)에서 발급한 임시 자격 증명입니다.
 - secretAccessKey: 세션에 대한 보안 액세스 키입니다.
 - sessionToken: 세션에 대한 토큰입니다.
 - accessKeyId: 세션에 대한 보안 액세스 키입니다.
- continuationToken: 작업에서 생성하는 토큰입니다. 향후 작업에서는 이 토큰을 사용하여 실 행 중인 작업 인스턴스를 식별합니다. 작업이 완료되면 구성 토큰이 제공되지 않아야 합니다.
- encryptionKey: 키와 같이 아티팩트 스토어의 데이터를 암호화하는 데 사용되는 암호화 AWS KMS 키입니다. 정의하지 않으면 Amazon Simple Storage Service의 기본 키가 사용됩니다.
 - id: 키를 식별하는 데 사용되는 ID입니다. AWS KMS 키의 경우 키 ID, 키 ARN 또는 별칭 ARN 을 사용할 수 있습니다.
 - type: 암호화 키의 유형입니다(예: AWS KMS 키).

다음 사항도 참조하세요.

이 작업을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

• <u>AWS CloudFormation 사용 설명서</u> - 파이프라인의 Lambda 작업 및 아 AWS CloudFormation 티팩 트에 대한 자세한 내용은 <u>CodePipeline 파이프라인에서 파라미터 재정의 함수 사용</u>, <u>Lambda 기반</u> 애플리케이션 및 아티팩트의 배포 자동화AWS CloudFormation 를 참조하세요. • <u>CodePipeline의 파이프라인에서 AWS Lambda 함수 호출</u> - 이 절차에서는 샘플 Lambda 함수를 제 공하며 콘솔을 사용하여 Lambda 간접 호출 작업이 있는 파이프라인을 생성하는 방법을 보여줍니다.

AWS OpsWorks 배포 작업 참조

파이프라인을 사용하여 AWS OpsWorks 작업을 사용하여 OpsWorks와 함께 배포합니다.

작업 유형

- 범주: Deploy
- 소유자: AWS
- 공급자: OpsWorks
- 버전:1

구성 파라미터

앱

필수 여부: 예

AWS OpsWorks 스택입니다. 스택은 애플리케이션 인프라의 컨테이너입니다.

스택

필수 여부: 예

AWS OpsWorks 앱. 앱은 배포하고 실행하려는 코드를 나타냅니다.

계층

필수 여부: 아니요

AWS OpsWorks 스택입니다. 계층은 인스턴스 세트에 대한 구성과 리소스를 지정합니다.

입력 아티팩트

- 아티팩트 수: 1
- 설명: 작업에 대한 입력 아티팩트입니다.

출력 아티팩트

- 아티팩트 수:0 to 1
- 설명: 이 작업 유형에는 출력 아티팩트가 적용되지 않습니다.

서비스 역할 권한: AWS OpsWorks 작업

AWS OpsWorks 지원을 받으려면 정책 설명에 다음을 추가합니다.



예제 작업 구성

YAML

```
Name: ActionName
ActionTypeId:
  Category: Deploy
  Owner: AWS
  Version: 1
  Provider: OpsWorks
InputArtifacts:
  - Name: myInputArtifact
Configuration:
  Stack: my-stack
  App: my-app
```

JSON

```
{
    "Name": "ActionName",
    "ActionTypeId": {
        "Category": "Deploy",
        "Owner": "AWS",
        "Version": 1,
        "Provider": "OpsWorks"
    },
    "InputArtifacts": [
        {
            "Name": "myInputArtifact"
        }
    ],
    "Configuration": {
        "Stack": "my-stack",
        "App": "my-app"
    }
}
```

다음 사항도 참조하세요.

이 작업을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

• <u>AWS OpsWorks 사용 설명서</u> -를 사용하여 배포하는 방법에 대한 자세한 내용은 AWS OpsWorks 사 용 설명서를 AWS OpsWorks참조하세요.

AWS Service Catalog 배포 작업 참조

AWS Service Catalog 작업을 사용하여 파이프라인을 사용하여 템플릿을 배포합니다. 다음은 Service Catalog에서 생성한 리소스 템플릿입니다.

작업 유형

- 범주: Deploy
- 소유자: AWS
- 공급자: ServiceCatalog
- 버전:1

구성 파라미터

TemplateFilePath

필수 여부: 예

소스 위치에 있는 리소스 템플릿의 파일 경로입니다.

ProductVersionName

필수 여부: 예

Service Catalog의 제품 버전입니다.

ProductType

필수 여부: 예

Service Catalog의 제품 유형입니다.

ProductId

필수 여부: 예

Service Catalog의 제품 ID입니다.

ProductVersionDescription

필수 여부: 아니요

Service Catalog의 제품 버전 설명입니다.

입력 아티팩트

- 아티팩트 수: 1
- 설명: 작업에 대한 입력 아티팩트입니다.

출력 아티팩트

- 아티팩트 수: 0
- 설명: 이 작업 유형에는 출력 아티팩트가 적용되지 않습니다.

서비스 역할 권한: Service Catalog 작업

Service Catalog 지원의 경우 다음을 정책 설명에 추가합니다.

```
{
    "Effect": "Allow",
    "Action": [
        "servicecatalog:ListProvisioningArtifacts",
        "servicecatalog:CreateProvisioningArtifact",
        "servicecatalog:DescribeProvisioningArtifact",
        "servicecatalog:DeleteProvisioningArtifact",
        "servicecatalog:UpdateProduct"
    ],
    "Resource": "resource_ARN"
},
{
    "Effect": "Allow",
    "Action": [
        "cloudformation:ValidateTemplate"
    ],
    "Resource": "resource_ARN"
}
```

구성 파일 유형별 작업 구성 예제

다음 예는 별도의 구성 파일 없이 콘솔에 생성된 파이프라인에 대해 Service Catalog를 사용하는 배포 작업의 유효한 구성을 보여줍니다.

```
"configuration": {
    "TemplateFilePath": "S3_template.json",
    "ProductVersionName": "devops S3 v2",
    "ProductType": "CLOUD_FORMATION_TEMPLATE",
    "ProductVersionDescription": "Product version description",
    "ProductId": "prod-example123456"
}
```

다음 예는 별도의 sample_config.json 구성 파일로 콘솔에 생성된 파이프라인에 대해 Service Catalog를 사용하는 배포 작업의 유효한 구성을 보여줍니다.

```
"configuration": {
    "ConfigurationFilePath": "sample_config.json",
    "ProductId": "prod-example123456"
```

}

예제 작업 구성

YAML

```
Name: ActionName
ActionTypeId:
  Category: Deploy
  Owner: AWS
  Version: 1
  Provider: ServiceCatalog
OutputArtifacts:
- Name: myOutputArtifact
Configuration:
  TemplateFilePath: S3_template.json
  ProductVersionName: devops S3 v2
  ProductType: CLOUD_FORMATION_TEMPLATE
  ProductType: CLOUD_FORMATION_TEMPLATE
  ProductVersionDescription: Product version description
  ProductId: prod-example123456
```

```
{
    "Name": "ActionName",
    "ActionTypeId": {
        "Category": "Deploy",
        "Owner": "AWS",
        "Version": 1,
        "Provider": "ServiceCatalog"
    },
    "OutputArtifacts": [
        {
            "Name": "myOutputArtifact"
        }
    ],
    "Configuration": {
        "TemplateFilePath": "S3_template.json",
        "ProductVersionName": "devops S3 v2",
        "ProductType": "CLOUD_FORMATION_TEMPLATE",
        "ProductVersionDescription": "Product version description",
        "ProductId": "prod-example123456"
    }
```

}

다음 사항도 참조하세요.

이 작업을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

- <u>Service Catalog 사용 설명서</u> Service Catalog의 리소스 및 템플릿에 대한 자세한 내용은 Service Catalog 사용 설명서를 참조하세요.
- <u>자습서: Service Catalog에 배포하는 파이프라인 생성</u> -이 자습서 자습서에서는 제품 템플릿을 Service Catalog에 배포하고 소스 리포지토리에서 변경한 내용을 전달하도록 파이프라인을 생성 및 구성하는 방법을 보여줍니다.

AWS Step Functions 작업 참조 호출

다음을 수행하는 AWS CodePipeline 작업입니다.

- 파이프라인에서 AWS Step Functions 상태 시스템 실행을 시작합니다.
- 작업 구성의 속성이나 입력으로 전달할 파이프라인 아티팩트에 있는 파일을 통해 상태 시스템에 초 기 상태를 제공합니다.
- 선택적으로 작업에서 시작된 실행을 식별하기 위한 실행 ID 접두사를 설정합니다.
- Standard 및 Express 상태 시스템을 지원합니다.

Note

Step Functions 작업은 Lambda에서 실행되므로 Lambda 함수에 대한 아티팩트 크기 할당량과 동일한 아티팩트 크기 할당량이 있습니다. 자세한 내용은 Lambda 개발자 안내서에서 <u>Lambda</u> 할당량을 참조하세요.

작업 유형

- 범주: Invoke
- 소유자: AWS
- 공급자: StepFunctions
- 버전:1

구성 파라미터

StateMachineArn

필수 여부: 예

호출할 상태 시스템의 Amazon 리소스 이름(ARN)입니다.

ExecutionNamePrefix

필수 여부: 아니요

기본적으로 작업 실행 ID는 상태 시스템 실행 이름으로 사용됩니다. 접두사가 제공되면 작업 실행 ID 앞에 하이픈이 추가되고 상태 시스템 실행 이름으로 함께 사용됩니다.

myPrefix-1624a1d1-3699-43f0-8e1e-6bafd7fde791

express 상태 시스템의 경우 이름에는 0-9, A-Z, a-z, - 및 _만 포함해야 합니다.

InputType

필수 여부: 아니요

• 리터럴(기본값): 지정되면 입력 필드의 값이 상태 시스템 입력으로 직접 전달됩니다.

리터럴을 선택한 경우의 입력 필드 예제 항목:

{"action": "test"}

• FilePath: 입력 필드에 의해 지정된 입력 아티팩트의 파일 내용은 상태 시스템 실행을 위한 입력 으로 사용됩니다. InputType이 FilePath로 설정된 경우 입력 아티팩트가 필요합니다.

파일 경로를 선택한 경우의 입력 필드 예제 항목:

assets/input.json

입력

필수 항목 여부: 조건부

• Literal: InputType이 리터럴(기본값)로 설정된 경우 이 필드는 선택 사항입니다.

제공되는 경우 입력 필드는 상태 시스템 실행의 입력으로 직접 사용됩니다. 그렇지 않으면 상태 시스템이 빈 JSON 객체 {}로 호출됩니다.

• FilePath: InputType이 FilePath로 설정된 경우 이 필드는 필수입니다.

InputType이 FilePath로 설정된 경우에도 입력 아티팩트가 필요합니다.

지정된 입력 아티팩트의 파일 내용은 상태 시스템 실행을 위한 입력으로 사용됩니다.

입력 아티팩트

- 아티팩트 수:0 to 1
- 설명: InputType이 FilePath로 설정된 경우 이 아티팩트는 필수 항목이며 상태 시스템 실행을 위해 입 력을 소스 지정하는 데 사용됩니다.

출력 아티팩트

- 아티팩트 수: 0 to 1
- 설명:
 - Standard 상태 시스템: 제공되는 경우 출력 아티팩트가 상태 시스템의 출력으로 채워집니다. 이는 상태 머신 실행이 성공적으로 완료된 후 <u>Step Functions DescribeExecution API</u> 응답의 output 속성에서 가져옵니다.
 - Express 상태 시스템: 지원되지 않습니다.

출력 변수

이 작업은 파이프라인에서 다운스트림 작업의 작업 구성에서 참조할 수 있는 출력 변수를 생성합니다.

자세한 내용은 변수 참조 단원을 참조하십시오.

StateMachineArn

상태 시스템의 ARN입니다.

ExecutionArn

상태 시스템 실행의 ARN입니다. Standard 상태 시스템에만 해당됩니다.

서비스 역할 권한: StepFunctions 작업

StepFunctions 작업의 경우 Step Functions 호출 작업을 통해 파이프라인을 만드는 데 필요한 최소 권한은 다음과 같습니다.

```
{
    "Effect": "Allow",
    "Action": [
        "states:DescribeStateMachine",
        "states:DescribeExecution",
        "states:StartExecution"
    ],
    "Resource": "resource_ARN"
},
```

예제 작업 구성

기본 입력의 예

YAML

```
{
    "Name": "ActionName",
    "ActionTypeId": {
        "Category": "Invoke",
        "Owner": "AWS",
        "Version": 1,
        "Provider": "StepFunctions"
    },
    "OutputArtifacts": [
        {
            "Name": "myOutputArtifact"
        }
}
```

```
}
],
"Configuration": {
    "StateMachineArn": "arn:aws:states:us-
east-1:111122223333:stateMachine:HelloWorld-StateMachine",
    "ExecutionNamePrefix": "my-prefix"
}
```

리터럴 입력의 예

YAML

```
Name: ActionName
ActionTypeId:
  Category: Invoke
  Owner: AWS
  Version: 1
  Provider: StepFunctions
OutputArtifacts:
  - Name: myOutputArtifact
Configuration:
  StateMachineArn: arn:aws:states:us-east-1:1112222333:stateMachine:HelloWorld-
StateMachine
  ExecutionNamePrefix: my-prefix
  Input: '{"action": "test"}'
```

```
{
    "Name": "ActionName",
    "ActionTypeId": {
        "Category": "Invoke",
        "Owner": "AWS",
        "Version": 1,
        "Provider": "StepFunctions"
    },
    "OutputArtifacts": [
        {
            "Name": "myOutputArtifact"
        }
    ],
```

```
"Configuration": {
    "StateMachineArn": "arn:aws:states:us-
east-1:111122223333:stateMachine:HelloWorld-StateMachine",
    "ExecutionNamePrefix": "my-prefix",
    "Input": "{\"action\": \"test\"}"
    }
}
```

입력 파일의 예

YAML

```
Name: ActionName
InputArtifacts:
  - Name: myInputArtifact
ActionTypeId:
 Category: Invoke
 Owner: AWS
 Version: 1
  Provider: StepFunctions
OutputArtifacts:
  - Name: myOutputArtifact
Configuration:
  StateMachineArn: 'arn:aws:states:us-east-1:111122223333:stateMachine:HelloWorld-
StateMachine'
  ExecutionNamePrefix: my-prefix
  InputType: FilePath
  Input: assets/input.json
```

```
{
    "Name": "ActionName",
    "InputArtifacts": [
        {
            "Name": "myInputArtifact"
        }
    ],
    "ActionTypeId": {
            "Category": "Invoke",
            "Owner": "AWS",
            "Version": 1,
        }
    }
}
```

```
"Provider": "StepFunctions"
},
"OutputArtifacts": [
        {
            "Name": "myOutputArtifact"
        }
     ],
     "Configuration": {
            "StateMachineArn": "arn:aws:states:us-
east-1:1112222333:stateMachine:HelloWorld-StateMachine",
            "ExecutionNamePrefix": "my-prefix",
            "InputType": "FilePath",
            "Input": "assets/input.json"
        }
}
```

동작

릴리스 중에 CodePipeline은 작업 구성에 지정된 입력을 사용하여 구성된 상태 시스템을 실행합니다.

InputType이 Literal로 설정된 경우 입력 작업 구성 필드의 내용이 상태 시스템의 입력으로 사용됩니다. 리터럴 입력이 제공되지 않으면 상태 시스템 실행은 빈 JSON 객체 {}를 사용합니다. 입력 없이 상태 시스템 실행을 실행하는 방법에 대한 자세한 내용은 <u>Step Functions StartExecution API</u>를 참조하십시 오.

InputType이 FilePath로 설정된 경우 작업은 입력 아티팩트의 압축을 풀고 입력 작업 구성 필드에 지정 된 파일의 내용을 상태 시스템의 입력으로 사용합니다. FilePath를 지정하면 입력 필드가 필요하며 입 력 아티팩트가 있어야 합니다. 그렇지 않으면 작업이 실패합니다.

성공적인 시작 실행 후, 동작은 두 가지 상태 시스템 유형인 standard 및 express에 대해 분기됩니다.

Standard 상태 시스템

Standard 상태 시스템 실행이 성공적으로 시작되면 실행이 터미널 상태에 도달할 때까지 CodePipeline이 DescribeExecution API를 폴링합니다. 실행이 성공적으로 완료되면 작업이 성공 하고 그렇지 않으면 실패합니다.

출력 아티팩트가 구성되면 아티팩트는 상태 시스템의 반환 값을 포함합니다. 이는 상태 머신 실행이 성 공적으로 완료된 후 <u>Step Functions DescribeExecution API</u> 응답의 output 속성에서 가져옵니다. 이 API에는 출력 길이 제약 조건이 적용됩니다. 오류 처리

- 작업이 상태 시스템 실행을 시작하지 못하면 작업 실행이 실패합니다.
- CodePipeline Step Functions 작업이 제한 시간(기본값: 7일)에 도달하기 전에 상태 시스템 실행이 터미널 상태에 도달하지 못하면 작업 실행이 실패합니다. 이렇게 실패해도 상태 시스템이 계속될 수 있습니다. Step Functions의 상태 시스템 실행 제한 시간에 대한 자세한 내용은 <u>표준 워크플로와</u> Express 워크플로 비교를 참조하세요.

Note

작업이 있는 계정의 호출 작업 시간 제한에 대한 할당량 증가를 요청할 수 있습니다. 그러나 할당량 증가는 해당 계정의 모든 리전에서 이 유형의 모든 작업에 적용됩니다.

• 상태 시스템 실행이 FAILED, TIMED_OUT 또는 ABORTED 터미널 상태에 도달하면 작업 실행이 실 패합니다.

Express 상태 시스템

express 상태 시스템 실행이 성공적으로 시작되면 invoke 작업 실행이 성공적으로 완료됩니다.

express 상태 시스템에 대해 구성된 작업에 대한 고려 사항:

- 출력 아티팩트를 지정할 수 없습니다.
- 작업은 상태 시스템 실행이 완료될 때까지 기다리지 않습니다.
- CodePipeline에서 작업 실행이 시작되면 상태 시스템 실행이 실패하더라도 작업 실행이 성공합니다.

오류 처리

 CodePipeline이 상태 시스템 실행을 시작하지 못하면 작업 실행이 실패합니다. 그렇지 않으면 작 업이 즉시 성공합니다. 작업은 상태 시스템 실행이 완료되는 데 걸리는 시간이나 결과에 관계없이 CodePipeline에서 성공합니다.

다음 사항도 참조하세요.

이 작업을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

• <u>AWS Step Functions 개발자 안내서</u> - 상태 시스템, 실행 및 상태 시스템 입력에 대한 자세한 내용은 AWS Step Functions 개발자 안내서를 참조하세요.

• <u>자습서: 파이프라인에서 AWS Step Functions 호출 작업 사용</u> - 이 자습서에서는 샘플 standard 상태 시스템으로 시작하고 콘솔을 사용하여 Step Functions 간접 호출 작업을 추가하여 파이프라인을 업 데이트하는 방법을 보여줍니다.

규칙 구조 참조

이 섹션은 규칙 구성에 대한 참조일 뿐입니다. 파이프라인 구조의 개념적 개요는 <u>CodePipeline 파이프</u> <u>라인 구조 참조</u> 단원을 참조하십시오.

CodePipeline의 각 규칙 공급자는 파이프라인 구조에서 필수 및 옵션 구성 필드 세트를 사용합니다. 이 섹션에는 다음과 같은 규칙 공급자별 참조가 나와 있습니다.

- 파이프라인 구조 규칙 블록에 포함된 RuleType 필드의 유효한 값(예: Owner 및 Provider).
- 파이프라인 구조 규칙 섹션에 포함된 Configuration 파라미터(필수 및 옵션)에 대한 설명 및 기타 참조 정보.
- 유효한 예제 JSON 및 YAML 규칙 구성 필드.

참조 정보는 다음과 같은 규칙 공급자에 대해 사용할 수 있습니다.

주제

- <u>CloudWatchAlarm</u>
- <u>CodeBuild 규칙</u>
- 명령
- DeploymentWindow
- Lambdalnvoke
- VariableCheck

CloudWatchAlarm

조건을 생성할 때 CloudWatchAlarm 규칙을 추가할 수 있습니다. 이 섹션에서는 규칙 파라미터에 대 한 참조를 제공합니다. 규칙 및 조건에 대한 자세한 내용은 <u>스테이지 조건은 어떻게 작동하나요?</u> 섹션 을 참조하세요.

Amazon CloudWatch에 별도의 리소스로 경보가 이미 생성되어 있어야 합니다.

주제

- <u>규칙 타입</u>
- 구성 파라미터

CloudWatchAlarm

- 규칙 구성의 예
- 다음 사항도 참조하세요.

규칙 타입

- 범주: Rule
- 소유자: AWS
- 공급자: CloudWatchAlarm
- 버전:1

구성 파라미터

AlarmName

필수 여부: 예

CloudWatch 경보의 이름입니다. 이는 CloudWatch에서 생성된 별도의 리소스입니다.

AlarmStates

필수 여부: 아니요

모니터링할 규칙에 대해 원하는 CloudWatch 경보 상태입니다. 유효한 값은 ALARM, OK 및 INSUFFICIENT_DATA입니다.

WaitTime

필수 여부: 아니요

규칙 결과를 실행하기 전에 상태 변경을 허용하는 분 단위의 대기 시간입니다. 예를 들어 규칙 결과 를 적용하기 전에 경보 상태가 OK로 변경될 때까지 기다리도록 20분을 구성합니다.

규칙 구성의 예

YAML

rules:

- name: MyMonitorRule
```
ruleTypeId:
   category: Rule
   owner: AWS
   provider: CloudWatchAlarm
   version: '1'
configuration:
   AlarmName: CWAlarm
   WaitTime: '1'
inputArtifacts: []
region: us-east-1
```

JSON

```
"rules": [
        {
            "name": "MyMonitorRule",
            "ruleTypeId": {
                "category": "Rule",
                "owner": "AWS",
                "provider": "CloudWatchAlarm",
                "version": "1"
            },
            "configuration": {
                "AlarmName": "CWAlarm",
                "WaitTime": "1"
            },
            "inputArtifacts": [],
            "region": "us-east-1"
        }
    ]
}
```

다음 사항도 참조하세요.

이 규칙을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

• 실패 시 조건 생성 - 이 섹션에서는 경보 규칙으로 실패 시 조건을 생성하는 단계를 제공합니다.

CodeBuild 규칙

조건을 생성할 때 CodeBuild 규칙을 추가할 수 있습니다. 이 섹션에서는 규칙 파라미터에 대한 참조를 제공합니다. 규칙 및 조건에 대한 자세한 내용은 <u>스테이지 조건은 어떻게 작동하나요?</u> 섹션을 참조하 세요.

CodeBuild 규칙을 사용하여 빌드 프로젝트의 성공적인 실행이 beforeEntry 조건에 대한 빌드 실행 성 공과 같은 규칙 기준을 충족하는 조건을 생성할 수 있습니다.

1 Note

건너뛰기 결과로 구성된 beforeEntry 조건의 경우 LambdaInvoke 및 규칙만 사용할 수 있습니 다VariableCheck.

주제

- 서비스 역할 정책 권한
- <u>규칙 타입</u>
- <u>구성 파라미터</u>
- <u>규칙 구성의 예</u>
- 다음 사항도 참조하세요.

서비스 역할 정책 권한

이 규칙에 대한 권한의 경우 CodePipeline 서비스 역할 정책 문에 다음을 추가합니다. 권한을 리소스 수준으로 축소합니다.

```
{
    "Effect": "Allow",
    "Action": [
        "codebuild:BatchGetBuilds",
        "codebuild:StartBuild"
    ],
    "Resource": "resource_ARN"
},
```

규칙 타입

- 범주: Rule
- 소유자: AWS
- 공급자: CodeBuild
- 버전:1

구성 파라미터

ProjectName

필수 여부: 예

ProjectName은 CodeBuild에서 빌드 프로젝트의 이름입니다.

PrimarySource

필수 항목 여부: 조건부

PrimarySource 파라미터 값은 작업에 대한 입력 아티팩트 중 하나의 이름이어야 합니다. CodeBuild는 buildspec 파일을 찾고이 아티팩트의 압축 해제된 버전이 포함된 디렉터리에서 buildspec 명령을 실행합니다.

CodeBuild 작업에 대해 여러 입력 아티팩트가 지정되는 경우 이 파라미터는 필수 항목입니다. 작업 에 대해 소스 아티팩트가 하나만 있는 경우 해당 아티팩트의 기본값은 PrimarySource 아티팩트 입니다.

BatchEnabled

필수 여부: 아니요

BatchEnabled 파라미터의 부울 값을 사용하면 작업이 동일한 빌드 실행에서 여러 빌드를 실행할 수 있습니다.

이 옵션을 활성화하면 CombineArtifacts 옵션을 사용할 수 있습니다.

배치 빌드가 활성화된 파이프라인 예제는 <u>CodePipeline과 CodeBuild 및 배치 빌드 통합을 참조하</u> <u>세요</u>.

CombineArtifacts

필수 여부: 아니요

CombineArtifacts 파라미터의 부울 값은 배치 빌드의 모든 빌드 아티팩트를 빌드 작업을 위한 단일 아티팩트 파일로 결합합니다.

이 옵션을 사용하려면 BatchEnabled 파라미터를 활성화해야 합니다.

EnvironmentVariables

필수 여부: 아니요

이 파라미터의 값은 파이프라인의 CodeBuild 작업에 대한 환경 변수를 설정하는 데 사용됩니다. EnvironmentVariables 파라미터 값은 환경 변수 객체의 JSON 배열 형식을 취합니다. <u>작업 선</u> 언(CodeBuild 예제)의 예제 파라미터를 참조하십시오.

각 객체에는 세 부분이 있으며 모두 문자열입니다.

- name: 환경 변수의 이름 또는 키입니다.
- value: 환경 변수의 값입니다. PARAMETER_STORE 또는 SECRETS_MANAGER 유형을 사용하는 경우이 값은 AWS Systems Manager Parameter Store에 이미 저장한 파라미터의 이름이거나 Secrets Manager에 AWS 이미 저장한 보안 암호여야 합니다.

(i) Note

환경 변수를 사용하여 민감한 값, 특히 AWS 자격 증명을 저장하지 않는 것이 좋습니다. CodeBuild 콘솔 또는 AWS CLI를 사용하면 환경 변수가 일반 텍스트로 표시됩니다. 중요 한 값의 경우 대신 SECRETS_MANAGER 유형을 사용하는 것이 좋습니다.

• type: (선택 사항) 환경 변수의 유형입니다. 유효한 값은 PARAMETER_STORE, SECRETS_MANAGER또는 PLAINTEXT입니다. 지정하지 않으면 기본적으로 PLAINTEXT가 사용 됩니다.

Note

환경 변수 구성에 대해 name, value 및 type을 입력할 때, 특히 환경 변수에 CodePipeline 출력 변수 구문이 포함된 경우 구성 값 필드에 1000자 제한을 초과하지 마십시오. 이 제한 을 초과하면 확인 오류가 반환됩니다.

자세한 내용은 AWS CodeBuild API 참조의 <u>EnvironmentVariable</u>을 참조하세요. GitHub 브랜치 이름으로 확인되는 환경 변수를 포함하는 CodeBuild 작업의 예는 <u>예: CodeBuild 환경 변수와 함께</u> BranchName 변수 사용 섹션을 참조하세요.

규칙 구성의 예

YAML

```
name: codebuild-rule
ruleTypeId:
  category: Rule
  owner: AWS
  provider: CodeBuild
  version: '1'
configuration:
    ProjectName: my-buildproject
    EnvironmentVariables: '[{"name":"VAR1","value":"variable","type":"PLAINTEXT"}]'
inputArtifacts:
    name: SourceArtifact
region: us-east-1
```

JSON

```
{
    "name": "codebuild-rule",
    "ruleTypeId": {
        "category": "Rule",
        "owner": "AWS",
        "provider": "CodeBuild",
        "version": "1"
    },
    "configuration": {
        "ProjectName": "my-buildproject"
    },
    "inputArtifacts": [
        {
            "name": "SourceArtifact",
            "EnvironmentVariables": "[{\"name\":\"VAR1\",\"value\":\"variable\",
\"type\":\"PLAINTEXT\"}]"
        }
    ],
    "region": "us-east-1"
}
```

다음 사항도 참조하세요.

이 규칙을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

 규칙 및 조건에 대한 자세한 내용은 CodePipeline API 가이드의 <u>조건</u>, <u>RuleTypeld</u> 및 RuleExecution을 참조하세요.

명령

조건을 생성할 때 Commands 규칙을 추가할 수 있습니다. 이 섹션에서는 규칙 파라미터에 대한 참조를 제공합니다. 규칙 및 조건에 대한 자세한 내용은 <u>스테이지 조건은 어떻게 작동하나요?</u> 섹션을 참조하 세요.

Commands 규칙을 사용하여 성공한 명령이 beforeEntry 조건에 성공하는 명령의 출력 및 파일 경로와 같은 규칙 기준을 충족하는 조건을 생성할 수 있습니다.

Note

건너뛰기 결과로 구성된 beforeEntry 조건의 경우 LambdaInvoke 및 규칙만 사용할 수 있습니 다VariableCheck.

주제

- 명령 규칙에 대한 고려 사항
- 서비스 역할 정책 권한
- 규칙 타입
- <u>구성 파라미터</u>
- 규칙 구성의 예
- 다음 사항도 참조하세요.

명령 규칙에 대한 고려 사항

명령 규칙에는 다음 고려 사항이 적용됩니다.

• 명령 규칙은 CodeBuild 작업과 유사한 CodeBuild 리소스를 사용하는 동시에 빌드 프로젝트를 연결 하거나 생성할 필요 없이 가상 컴퓨팅 인스턴스에서 셸 환경 명령을 허용합니다. Note

명령 규칙을 실행하면 별도의 요금이 발생합니다 AWS CodeBuild.

- CodePipeline의 명령 규칙은 CodeBuild 리소스를 사용하기 때문에 작업에서 실행되는 빌드는 CodeBuild의 계정에 대한 빌드 제한에 기인합니다. 명령 규칙에 의해 실행되는 빌드는 해당 계정에 구성된 동시 빌드 제한에 포함됩니다.
- CodeBuild 빌드를 기반으로 할 때 명령 규칙을 사용하는 빌드의 제한 시간은 55분입니다.
- 컴퓨팅 인스턴스는 CodeBuild에서 격리된 빌드 환경을 사용합니다.

Note

격리된 빌드 환경은 계정 수준에서 사용되므로 인스턴스를 다른 파이프라인 실행에 재사용 할 수 있습니다.

- 다중 라인 형식을 제외한 모든 형식이 지원됩니다. 명령을 입력할 때는 단일 줄 형식을 사용해야 합니다.
- 이 규칙의 경우 CodePipeline은 파이프라인 서비스 역할을 수임하고 해당 역할을 사용하여 런타임
 시 리소스에 대한 액세스를 허용합니다. 권한 범위가 작업 수준까지 내려가도록 서비스 역할을 구성
 하는 것이 좋습니다.
- CodePipeline 서비스 역할에 추가된 권한은 <u>CodePipeline 서비스 역할에 권한 추가</u>에 자세히 설명되 어 있습니다.
- 콘솔에서 로그를 보는 데 필요한 권한은 <u>CodePipeline 콘솔에서 컴퓨팅 로그를 보는 데 필요한 권</u>
 한에 자세히 설명되어 있습니다. 다음 예제 화면에서 로그 링크를 사용하여 CloudWatch 로그에서 성공적인 명령 규칙에 대한 로그를 봅니다.

	Condition exe Execution ID: 9ace7b	ecution details 55-da5d-426d-8451-fe3e92b	o1c1e6				×
	Condition type: B	BeforeEntry Result: FAIL	Status: 🕑 S	ucceeded			
	Rule states	Rule Configuration					
	Name	Rule Execution ID				Status	Reason
	cmdsrule	05f51200-cf09-4d				Succeeded	Logs 2
							Done
	2024-11-04T15:	:47:07.928Z	[Container]	2024/11/04	15:47:07.469537	Entering phase BUILD	
	2024-11-04T15:	47:07.928Z	[Container]	2024/11/04	15:47:07.514998	Running command echo	"hello world"
	2024-11-04T15:	:47:07.928Z	hello world				
	2024-11-04T15:	47:07.928Z					
•	2024-11-04T15:	47:07.928Z	[Container]	2024/11/04	15:47:07.522410	Phase complete: BUIL	D State: SUCCEE
•	2024-11-04T15:	47:07.928Z	[Container]	2024/11/04	15:47:07.522428	Phase context status	code: Message:
•	2024-11-04T15:	:47:07.928Z	[Container]	2024/11/04	15:47:07.565724	Entering phase POST_	BUILD
•	2024-11-04T15:	:47:07.928Z	[Container]	2024/11/04	15:47:07.566944	Phase complete: POST	_BUILD State: S

• CodePipeline의 다른 작업과 달리 작업 구성에서는 필드를 설정하지 않고 작업 구성 외부에서 작업 구성 필드를 설정합니다.

서비스 역할 정책 권한

CodePipeline이 규칙을 실행하면 CodePipeline은 다음과 같이 파이프라인 이름을 사용하여 로그 그룹 을 생성합니다. 이렇게 하면 파이프라인 이름을 사용하여 리소스를 로깅할 수 있는 권한을 범위 아래로 지정할 수 있습니다.

/aws/codepipeline/MyPipelineName

기존 서비스 역할을 사용하는 경우 명령 작업을 사용하려면 서비스 역할에 대해 다음 권한을 추가해야 합니다.

logs:CreateLogGroup

- logs:CreateLogStream
- logs:PutLogEvents

서비스 역할 정책 설명에서 다음 예제와 같이 파이프라인 수준까지 권한의 범위를 좁힙니다.

```
{
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
    ],
    "Resource": "arn:aws:logs:*:YOUR_AWS_ACCOUNT_ID:log-group:/aws/
codepipeline/YOUR_PIPELINE_NAME:*"
}
```

작업 세부 정보 대화 상자를 사용하여 콘솔에서 로그를 보려면 로그를 볼 수 있는 권한을 콘솔 역할에 추가해야 합니다. 자세한 내용은 <u>CodePipeline 콘솔에서 컴퓨팅 로그를 보는 데 필요한 권한</u>에서 콘솔 권한 정책 예제를 참조하세요.

규칙 타입

- 범주: Rule
- 소유자: AWS
- 공급자: Commands
- 버전:1

구성 파라미터

명령

필수 여부: 예

Commands 규칙을 실행할 셸 명령을 제공할 수 있습니다. 콘솔에서 명령은 별도의 줄에 입력됩니다. CLI에서 명령은 별도의 문자열로 입력됩니다.

Note

다중 라인 형식은 지원되지 않으며 오류 메시지가 표시됩니다. 명령 필드에 명령을 입력하 려면 단일 줄 형식을 사용해야 합니다.

다음 세부 정보는 명령 규칙에 사용되는 기본 컴퓨팅을 제공합니다. 자세한 내용은 CodeBuild 사용 설명서의 빌드 환경 컴퓨팅 모드 및 유형을 참조하세요.

- CodeBuild 이미지: aws/codebuild/amazonlinux2-x86_64-standard:5.0
- 컴퓨팅 유형: Linux Small
- 환경 computeType 값: BUILD_GENERAL1_SMALL
- 환경 유형 값: LINUX_CONTAINER

규칙 구성의 예

YAML

```
result: FAIL
rules:
- name: CommandsRule
ruleTypeId:
    category: Rule
    owner: AWS
    provider: Commands
    version: '1'
configuration: {}
commands:
    - ls
    - printenv
inputArtifacts:
    - name: SourceArtifact
region: us-east-1
```

JSON

```
"name": "CommandsRule",
            "ruleTypeId": {
                 "category": "Rule",
                 "owner": "AWS",
                 "provider": "Commands",
                 "version": "1"
            },
            "configuration": {},
            "commands": [
                 "ls",
                 "printenv"
            ],
            "inputArtifacts": [
                 {
                     "name": "SourceArtifact"
                 }
            ],
            "region": "us-east-1"
        }
    ]
}
```

다음 사항도 참조하세요.

이 규칙을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

• 규칙 및 조건에 대한 자세한 내용은 CodePipeline API 가이드의 <u>조건</u>, <u>RuleTypeld</u> 및 RuleExecution을 참조하세요.

DeploymentWindow

조건을 생성할 때 DeploymentWindow 규칙을 추가할 수 있습니다. 이 섹션에서는 규칙 파라미터에 대한 참조를 제공합니다. 규칙 및 조건에 대한 자세한 내용은 <u>스테이지 조건은 어떻게 작동하나요?</u> 섹 션을 참조하세요.

주제

- <u>규칙 타입</u>
- <u>구성 파라미터</u>
- 규칙 구성의 예

• <u>다음 사항도 참조하세요.</u>

규칙 타입

- 범주: Rule
- 소유자: AWS
- 공급자: DeploymentWindow
- 버전:1

구성 파라미터

Cron

필수 여부: 예

배포가 허용되는 날짜와 시간을 정의하는 표현식입니다. Cron 표현식은 6개의 필수 필드와 공백으 로 구분된 하나의 선택적 필드로 구성됩니다. cron 표현식 필드를 사용하면 다음과 같이 cron 표현 식을 사용하여 일정 패턴을 지정할 수 있습니다.

필드 이름	허용된 값	허용되는 특수 문자
초	N/A	*
분	0~59	, - * /
시간	0~23	, - * /
날짜	1~31	, - * ? / L W
월	1-12 또는 JAN-DEC	, - * /
요일	1-7 또는 SUN-SAT	, - * ? / L #
연도(선택 사항)	비어 있음, 1970~2199	, - * /

• '*' 문자는 모든 값을 지정하는 데 사용됩니다. 예를 들어 분 필드의 '*'는 '매분'을 의미합니다.

- 날짜 및 요일 필드에는 '?' 문자가 허용됩니다. 이 문자는 '특정 값 없음'을 지정하는 데 사용됩니다.
 다. 두 필드 중 하나에 무언가를 지정해야 하지만 다른 필드에는 지정하지 않아도 되는 경우에 유용합니다.
- '-' 문자는 범위를 지정하는 데 사용됩니다. 예를 들어 시간 필드의 '10~12'는 '10, 11, 12시간'을 의 미합니다.
- ',' 문자는 추가 값을 지정하는 데 사용됩니다. 예를 들어 요일 필드의 'MON,WED,FRI'는 '월요일, 수요일, 금요일'을 의미합니다.
- '/' 문자는 증분을 지정하는 데 사용됩니다. 예를 들어 초 필드의 '0/15'는 '0, 15, 30, 45초'를 의미 합니다. 초 필드의 '5/15'는 '5, 20, 35, 50초'를 의미합니다. '/' 앞에 '*'를 지정하는 것은 시작할 값 을 0으로 지정하는 것과 같습니다.
- 날짜 및 요일 필드에는 'L' 문자가 허용됩니다. 이 문자는 'last'의 약자이지만 두 필드 각각에서 의 미가 다릅니다. 예를 들어, 날짜 필드의 값 'L'은 '해당 월의 마지막 날'을 의미합니다. 1월의 경우 31일, 윤년이 아닌 2월의 경우 28일입니다. 요일 필드에 단독으로 사용되는 경우, 단순히 '7' 또 는 'SAT'를 의미합니다. 하지만 다른 값 뒤에 있는 요일 필드에 사용하는 경우 '해당 월의 마지막 <specified_day>일'을 의미합니다. 예를 들어 '6L'은 '월의 마지막 금요일'을 의미합니다. 또한 해 당 월의 마지막 날부터의 오프셋을 지정할 수도 있습니다. 예를 들어 'L-3'은 달력 월의 셋째날부 터 마지막 날까지를 의미합니다.
- 날짜 필드에는 'W' 문자가 허용됩니다. 이 문자는 지정된 날짜에 가장 가까운 평일(월요일~금요일)을 지정하는 데 사용됩니다. 예를 들어 '15W'를 날짜 필드 값으로 지정하면 '해당 월에서 15일에 가장 가까운 평일'을 의미합니다. 따라서 15일이 토요일인 경우 트리거는 14일 금요일에 실행됩니다. 15일이 일요일인 경우 트리거는 16일 월요일에 실행됩니다. 15일이 화요일인 경우 15일 화요일에 실행됩니다.
- 'L' 문자와 'W' 문자를 결합하여 'LW'를 만들 수도 있습니다. 이 표현식은 '해당 월의 마지막 평일'로 해석됩니다.
- 요일 필드에는 '#' 문자가 허용됩니다. 이 문자는 해당 월의 'n 번째' <specified_day>일을 지정하는 데 사용됩니다. 예를 들어, 요일 필드의 '6#3' 값은 해당 월의 세 번째 금요일(6번째 날 = 금요일, '#3' = 해당 월의 세 번째 금요일)을 의미합니다.
- 적합한 문자와 월 및 일의 이름은 대소문자를 구분하지 않습니다.

TimeZone

필수 여부: 아니요

배포 창의 시간대입니다. 정규식은 다음 형식의 패턴과 일치합니다.

• 리전/도시 형식. 값은 Region/City 또는 Region/City_City 형식의 시간대와 일치합니다. 예: America/New_York또는 Europe/Berlin.

- UTC 형식. 값은 문자열 UTC와 선택적으로 일치하고, 그 다음에 +HH:MM 또는 -HH:MM 형식의 오프셋이 옵니다. 예를 들면 UTC, UTC+05:30 또는 UTC-03:00입니다. 이는 파라미터가 다르게 설정되지 않은 경우 기본값입니다.
- 약어 형식. 값은 시간대의 3~5자 약어와 일치합니다. 예: EST또는 IST.

유효한 TimeZoneID 값의 표는 <u>https://docs.oracle.com/middleware/1221/wcs/tag-ref/MISC/</u> <u>TimeZones.html</u>을 참조하세요. 중부 표준시, 중국 표준시, 쿠바 표준시를 나타내는 CST와 같이 특정 약어는 중복됩니다.

규칙 구성의 예

```
YAML
```

```
- name: MyDeploymentRule
ruleTypeId:
   category: Rule
   owner: AWS
   provider: DeploymentWindow
   version: '1'
configuration:
   Cron: 0 0 9-17 ? * MON-FRI *
   TimeZone: PST
   inputArtifacts: []
   region: us-east-1
```

JSON

"region": "us-east-1"
}

다음 사항도 참조하세요.

]

이 규칙을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

- <u>성공 시 조건 생성</u> 이 섹션에서는 배포 창 규칙을 사용하여 성공 사 조건을 생성하는 단계를 제공합 니다.
- 규칙 및 조건에 대한 자세한 내용은 CodePipeline API 가이드의 <u>조건</u>, <u>RuleTypeld</u> 및 <u>RuleExecution</u>을 참조하세요.

Lambdalnvoke

조건을 생성할 때 LambdaInvoke 규칙을 추가할 수 있습니다. 이 섹션에서는 규칙 파라미터에 대한 참조를 제공합니다. 규칙 및 조건에 대한 자세한 내용은 <u>스테이지 조건은 어떻게 작동하나요?</u> 섹션을 참조하세요.

Lambda에 별도의 리소스로 함수가 이미 생성되어 있어야 합니다.

주제

- <u>규칙 타입</u>
- 구성 파라미터
- <u>규칙</u> 구성의 예
- 다음 사항도 참조하세요.

규칙 타입

- 범주: Rule
- 소유자: AWS
- 공급자: LambdaInvoke
- 버전:1

구성 파라미터

FunctionName

```
필수 여부: 예
```

Lambda 함수의 이름입니다.

UserParameters

필수 여부: 아니요

키 값 페어 형식의 함수에 대한 입력으로 제공되는 파라미터입니다.

규칙 구성의 예

YAML

```
- name: MyLambdaRule
ruleTypeId:
   category: Rule
   owner: AWS
   provider: LambdaInvoke
   version: '1'
configuration:
   FunctionName: my-function
inputArtifacts:
   - name: SourceArtifact
region: us-east-1
```

JSON

```
[
{
        "name": "MyLambdaRule",
        "ruleTypeId": {
            "category": "Rule",
            "owner": "AWS",
            "provider": "LambdaInvoke",
            "version": "1"
        },
        "configuration": {
    }
}
```

```
"FunctionName": "my-function"
},
"inputArtifacts": [
        {
            "name": "SourceArtifact"
        }
    ],
    "region": "us-east-1"
    }
]
```

다음 사항도 참조하세요.

이 규칙을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

• 실패 시 조건 생성 - 이 섹션에서는 경보 규칙으로 실패 시 조건을 생성하는 단계를 제공합니다.

VariableCheck

조건을 생성할 때 VariableCheck 규칙을 추가할 수 있습니다. 이 섹션에서는 규칙 파라미터에 대한 참조를 제공합니다. 규칙 및 조건에 대한 자세한 내용은 <u>스테이지 조건은 어떻게 작동하나요?</u> 섹션을 참조하세요.

VariableCheck 규칙을 사용하여 출력 변수가 제공된 표현식에 대해 확인되는 조건을 생성할 수 있 습니다. 규칙은 변수 값이 지정된 출력 변수와 같거나 큰 값과 같은 규칙 기준을 충족할 때 확인을 전달 합니다.

주제

- 규칙 타입
- 구성 파라미터
- <u>규칙 구성의 예</u>
- <u>다음 사항도 참조하세요.</u>

규칙 타입

- 범주: Rule
- 소유자: AWS

- 공급자: VariableCheck
- 버전:1

구성 파라미터

연산자

필수 여부: 예

변수 확인을 위해 수행할 작업을 나타내는 연산자입니다.

다음 예제에서는 리포지토리 이름의 출력 변수가 MyDemoRepo와 동일한지 확인합니다.

```
"configuration": {
    "Variable": "#{SourceVariables.RepositoryName}",
    "Value": "MyDemoRepo",
    "Operator": "EQ"
},
```

다음 연산자를 사용하여 다음과 같이 표현식을 생성할 수 있습니다.

• 같음 - 변수가 문자열 값과 동일한지 확인하려면 이 연산자를 선택합니다.

```
CLI 파라미터: EQ
```

- 포함 변수에 문자열 값이 하위 문자열로 포함되어 있는지 확인하려면 이 연산자를 선택합니다. CLI 파라미터: CONTAINS
- 일치 변수가 문자열 값으로 지정된 정규식과 일치하는지 확인하려면 이 연산자를 선택합니다.

CodePipeline의 모든 정규식은 Java regex 구문을 따릅니다. Java regex 구문과 그 구조에 대한 포괄적인 설명은 java.util.regex.Pattern을 참조하세요.

CLI 파라미터: MATCHES

• 같이 않음 - 변수가 문자열 값과 같지 않은지 확인하려면 이 연산자를 선택합니다.

CLI 파라미터: NE

변수

필수 여부: 예

확인할 파이프라인 변수입니다.

값

필수 여부: 예

확인할 표현식의 값입니다.

다음 예제에서는 리포지토리 이름의 출력 변수가 MyDemoRepo와 동일한지 확인합니다.

```
"configuration": {
    "Variable": "#{SourceVariables.RepositoryName}",
    "Value": "MyDemoRepo",
    "Operator": "EQ"
},
```

다음 JSON 예제에서는 두 가지 별도의 규칙이 정의되어 있습니다. 하나는

#{SourceVariables.RepositoryName} 형식의 리포지토리 및 브랜치 이름을 확인하는 EQ(같음) 문이고, 다른 하나는 #{SourceVariables.CommitMessage} 형식의 커밋 메시지 출력 변수를 'update' 값에 대해 확인하는 CONTAINS입니다.

```
"beforeEntry": {
                  "conditions": [
                       {
                           "result": "FAIL",
                           "rules": [
                               {
                                   "name": "MyVarCheckRule",
                                   "ruleTypeId": {
                                       "category": "Rule",
                                       "owner": "AWS",
                                       "provider": "VariableCheck",
                                       "version": "1"
                                   },
                                   "configuration": {
                                       "Operator": "EQ",
                                       "Value": "MyDemoRepo",
                                       "Variable": "#{SourceVariables.RepositoryName}"
                                   },
                                   "inputArtifacts": [],
                                   "region": "us-east-1"
                               },
                               {
                                   "name": "MyVarCheckRuleContains",
```



규칙 구성의 예

YAML

```
- name: MyVariableCheck
ruleTypeId:
    category: Rule
    owner: AWS
    provider: VariableCheck
    version: '1'
configuration:
    Variable: "#{SourceVariables.RepositoryName}"
    Value: MyDemoRepo
    Operator: EQ
inputArtifacts: []
region: us-west-2
```

JSON

```
"rules": [
    {
        "name": "MyVariableCheck",
```

```
"ruleTypeId": {
    "category": "Rule",
    "owner": "AWS",
    "provider": "VariableCheck",
    "version": "1"
    },
    "configuration": {
        "Variable": "#{SourceVariables.RepositoryName}",
        "Value": "MyDemoRepo",
        "Operator": "EQ"
    },
    "inputArtifacts": [],
    "region": "us-west-2"
    }
]
```

다음 사항도 참조하세요.

이 규칙을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

- <u>자습서: 입력 조건으로 파이프라인에 대한 변수 확인 규칙 생성</u> 이 섹션에서는 가변 확인 규칙을 사 용하여 입력 시 조건을 생성하는 단계를 포함하는 자습서를 제공합니다.
- 변수 참조 이 섹션에서는 파이프라인 변수에 대한 참조 정보와 예제를 제공합니다.
- 규칙 및 조건에 대한 자세한 내용은 CodePipeline API 가이드의 <u>조건</u>, <u>RuleTypeld</u> 및 RuleExecution을 참조하세요.

통합 모델 참조

기존 고객 도구를 파이프라인 릴리스 프로세스에 통합하는 데 도움이 되도록 타사 서비스를 위한 몇 가지 사전 구축된 통합 기능이 있습니다. 파트너 또는 타사 서비스 공급자는 통합 모델을 사용하여 CodePipeline에서 사용할 작업 유형을 구현합니다.

CodePipeline에서 지원되는 통합 모델로 관리되는 작업 유형을 계획하거나 작업할 때 이 참조를 사용 하세요.

타사 작업 유형을 CodePipeline과의 파트너 통합으로 인증하려면 AWS 파트너 네트워크(APN)를 참조 하세요. 이 정보는 AWS CLI 참조를 보완한 것입니다.

주제

- 타사 작업 유형이 통합자와 함께 작동하는 방식
- 개념
- 지원되는 통합 모델
- Lambda 통합 모델
- <u>작업자 통합 모델</u>

타사 작업 유형이 통합자와 함께 작동하는 방식

고객 파이프라인에 타사 작업 유형을 추가하여 고객 리소스에 대한 작업을 완료할 수 있습니다. 통합자 는 CodePipeline을 사용하여 작업 요청을 관리하고 작업을 실행합니다. 다음 다이어그램은 고객이 파 이프라인에서 사용할 수 있도록 만든 타사 작업 유형을 보여줍니다. 고객이 작업을 구성하면 작업이 실 행되고 통합자의 작업 엔진에서 처리하는 작업 요청이 생성됩니다.



- 2. 공급자의 고객이 CodePipeline에서 작업을 선택하고 구성합니다.
- 3. 작업이 실행되고 작업이 CodePipeline에 대기됩니다. CodePipeline에서 작업이 준비되면 작업 요청 을 보냅니다.
- 4. 통합자(타사 폴링 API의 작업자 또는 Lambda 함수)가 작업 요청을 수신하여 확인을 반환하고 작업 에 대한 아티팩트를 처리합니다.
- 5. 통합자는 작업 결과 및 연속 토큰과 함께 성공/실패 출력(작업자가 성공/실패 API를 사용하거나 Lambda 함수가 성공/실패 출력을 전송)을 반환합니다.

작업 유형을 요청, 확인 및 업데이트하는 데 사용할 수 있는 단계에 대한 자세한 내용은 <u>작업 유형 처</u> <u>리</u>을 참조하세요.

개념

이 섹션에서는 타사 작업 유형에 대해 다음과 같은 용어를 사용합니다.

작업 유형

동일한 지속적 전송 워크로드를 수행하는 파이프라인에서 재사용할 수 있는 반복 가능한 프로세스 입니다. 작업 유형은 Owner, Category, Provider 및 Version으로 식별됩니다. 예시:

```
{
    "Category": "Deploy",
    "Owner": "AWS",
    "Provider": "CodeDeploy",
    "Version": "1"
},
```

동일한 유형의 모든 작업은 동일한 구현을 공유합니다.

작업

작업 유형의 단일 인스턴스로, 파이프라인 단계 내에서 발생하는 개별 프로세스 중 하나입니다. 여 기에는 일반적으로 이 작업이 실행되는 파이프라인 고유의 사용자 값이 포함됩니다. 작업 정의

작업 및 입력/출력 아티팩트를 구성하는 데 필요한 속성을 정의하는 작업 유형의 스키마입니다. 작업 실행

고객 파이프라인에서의 작업 성공 여부를 확인하기 위해 실행된 작업 모음입니다.

작업 실행 엔진

작업 유형에 사용되는 통합 유형을 정의하는 작업 실행 구성의 속성입니다. 유효 값은 JobWorker및 Lambda입니다.

통합

작업 유형을 구현하기 위해 통합자가 실행하는 소프트웨어를 설명합니다. CodePipeline은 지원되 는 두 작업 엔진 JobWorker 및 Lambda에 해당하는 두 가지 통합 유형을 지원합니다.

Integrator

작업 유형의 구현을 소유한 사람입니다.

작업

파이프라인 및 고객 컨텍스트를 바탕으로 통합을 실행하는 작업입니다. 작업 실행은 하나 이상의 작업으로 구성됩니다.

작업자

고객 입력을 처리하고 작업을 실행하는 서비스입니다.

지원되는 통합 모델

CodePipeline에는 두 가지 통합 모델이 있습니다.

- Lambda 통합 모델: 이 통합 모델은 CodePipeline의 작업 유형을 처리하는 데 선호되는 방법입니다. Lambda 통합 모델은 Lambda 함수를 사용하여 작업이 실행될 때 작업 요청을 처리합니다.
- 작업자 통합 모델: 작업자 통합 모델은 이전에 타사 통합에 사용되었던 모델입니다. 작업자 통합 모 델은 CodePipeline API에 접속하여 작업 실행 시 작업 요청을 처리하도록 구성된 작업자를 사용합니다.

비교를 위해 다음 표에서는 두 모델의 기능을 설명합니다.

	Lambda 통합 모델	작업자 통합 모델
설명	통합자는 Lambda 함수로 통합을 작성합니다. 이 함수는 작업에 사 용할 수 있는 작업이 있을 때마다 CodePipeline에 의해 호출됩니다. Lambda 함수는 사용 가능한 작업 을 폴링하지 않고 대신 다음 작업 요청이 수신될 때까지 기다립니다.	통합자는 고객의 파이프라인에서 사용 가능한 작업을 지속적으로 폴링하는 작업자로 통합을 작성합 니다. 그러면 작업자는 CodePipel ine API를 사용하여 작업을 실행 하고 CodePipeline에 작업 결과를 다시 제출합니다.
인프라	AWS Lambda	Amazon EC2 인스턴스와 같은 통 합자의 인프라에 작업자 코드를 배포합니다.
개발 노력	통합에는 비즈니스 로직만 포함됩 니다.	통합은 비즈니스 로직을 포함하는 것 외에도 CodePipeline API와 상 호 작용해야 합니다.
운영 노력	인프라는 AWS 리소스일 뿐이므로 운영 노력이 줄어듭니다.	작업자에게 독립형 하드웨어가 필 요하기 때문에 운영 노력이 더 많 이 듭니다.
최대 작업 런타임	통합을 15분 이상 활발하게 실행해 야 하는 경우 이 모델을 사용할 수 없습니다. 이 작업은 프로세스를 시 작(예: 고객의 코드 아티팩트를 기 반으로 빌드 시작)하고 프로세스가 완료되면 결과를 반환해야 하는 통 합자를 위한 것입니다. 통합자는 빌 드가 완료될 때까지 계속 기다리지 않는 것이 좋습니다. 대신 연속을 반환하세요. CodePipeline은 통합 자의 코드에서 작업이 완료될 때까 지 작업을 확인하라는 메시지가 계 속 전송되면 30초 후에 새 작업을 생성합니다.	이 모델을 사용하면 매우 오래 실 행되는 작업(시간/일)을 지속할 수 있습니다.

Lambda 통합 모델

지원되는 Lambda 통합 모델에는 Lambda 함수 생성 및 타사 작업 유형에 대한 출력 정의가 포함됩니 다.

CodePipeline의 입력을 처리하도록 Lambda 함수를 업데이트하세요.

새 Lambda 함수를 생성할 수 있습니다. 파이프라인에서 작업 유형에 사용할 수 있는 작업이 있을 때마 다 실행되는 Lambda 함수에 비즈니스 로직을 추가할 수 있습니다. 인스턴스의 경우, 고객 및 파이프라 인의 컨텍스트를 고려하면 고객을 위해 서비스 빌드를 시작하는 것이 좋습니다.

다음 파라미터를 사용하여 CodePipeline의 입력을 처리하도록 Lambda 함수를 업데이트하세요.

형식:

- jobId:
 - 작업의 고유한 시스템 생성 ID입니다.
 - 유형: 문자열
 - 패턴: [0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}
- accountId:
 - 작업을 수행할 때 사용할 고객 AWS 계정의 ID입니다.
 - 유형: 문자열
 - 패턴: [0-9]{12}
- data:
 - 통합이 작업을 완료하는 데 사용하는 작업에 대한 기타 정보입니다.
 - 다음과 같은 맵을 포함합니다.
 - actionConfiguration:
 - 작업에 대한 구성 데이터. 작업 구성 필드는 고객이 값을 입력할 수 있도록 키-값 페어를 매핑 한 것입니다. 키는 작업을 설정할 때 작업 유형 정의 파일의 주요 파라미터에 의해 결정됩니다.
 다. 이 예제에서 값은 작업 사용자가 Username 및 Password 필드에 정보를 지정하여 결정 합니다.
 - 유형: 문자열 간 맵, 선택적으로 표시

예제:

```
사용자 가이드
```

```
"configuration": {
    "Username": "MyUser",
    "Password": "MyPassword"
},
```

- encryptionKey:
 - 키와 같이 아티팩트 스토어의 데이터를 암호화하는 데 사용되는 AWS KMS 키에 대한 정보를 나타냅니다.
 - 내용: 데이터 유형 encryptionKey의 유형, 선택적으로 표시
- inputArtifacts:
 - 작업할 아티팩트에 대한 정보 목록입니다(예: 테스트 또는 빌드 아티팩트).
 - 내용: 데이터 유형 Artifact의 목록, 선택적으로 표시
- outputArtifacts:
 - 작업의 출력에 대한 정보 목록입니다.
 - 내용: 데이터 유형 Artifact의 목록, 선택적으로 표시
- actionCredentials:
 - AWS 세션 자격 증명 객체를 나타냅니다. 이러한 자격 증명은 AWS STS에서 발급한 임시 자 격 증명입니다. CodePipeline에 파이프라인의 아티팩트를 저장하는 데 사용되는 S3 버킷의 입력 및 출력 아티팩트에 액세스하는 데 사용할 수 있습니다.

또한 이러한 자격 증명은 작업 유형 정의 파일에 지정된 정책 설명 템플릿과 동일한 권한을 가 집니다.

- 내용: 데이터 유형 AWSSessionCredentials의 유형, 선택적으로 표시
- actionExecutionId:
 - 작업 실행의 외부 ID.
 - 유형: 문자열
- continuationToken:
 - 작업을 비동기적으로 계속하기 위해 작업에 필요한 시스템 생성 토큰(예: 배포 ID)입니다.
 - 유형: 문자열, 선택적으로 표시

데이터 형식:

• encryptionKey:

- 키를 식별하는 데 사용되는 ID입니다. AWS KMS 키의 경우 키 ID, 키 ARN 또는 별칭 ARN을 사용할 수 있습니다.
- 유형: 문자열
- type:
 - 키와 같은 암호화 AWS KMS 키의 유형입니다.
 - 타입: 문자열
 - 유효값: KMS
- Artifact:
 - name:
 - 아티팩트의 이름입니다.
 - 유형: 문자열, 선택적으로 표시
 - revision:
 - 아티팩트의 개정 ID입니다. 객체 유형에 따라 커밋 ID(GitHub) 또는 개정 ID(Amazon S3)일 수 있습니다.
 - 유형: 문자열, 선택적으로 표시
 - location:
 - 아티팩트의 위치입니다.
 - 내용: 데이터 유형 ArtifactLocation의 유형, 선택적으로 표시
- ArtifactLocation:
 - type:
 - 위치에서 아티팩트의 유형입니다.
 - 유형: 문자열, 선택적으로 표시
 - 유효값: S3
 - s3Location:
 - 개정이 포함된 S3 버킷의 위치입니다.
 - 내용: 데이터 유형 S3Location의 유형, 선택적으로 표시
- S3Location:
 - bucketName:
 - S3 버킷의 이름.
 - 유형: 문자열
 - objectKey:

- 버킷의 객체를 고유하게 식별하는 S3 버킷의 객체 키입니다.
- 유형: 문자열
- AWSSessionCredentials:
 - accessKeyId:
 - 세션에 대한 액세스 키입니다.
 - 유형: 문자열
 - secretAccessKey:
 - 세션에 대한 비밀 액세스 키입니다.
 - 유형: 문자열
 - sessionToken:
 - 세션에 대한 토큰입니다.
 - 유형: 문자열

예제:

```
{
    "jobId": "01234567-abcd-abcd-abcd-012345678910",
    "accountId": "012345678910",
    "data": {
        "actionConfiguration": {
            "key1": "value1",
            "key2": "value2"
        },
        "encryptionKey": {
            "id": "123-abc",
            "type": "KMS"
        },
        "inputArtifacts": [
            {
                "name": "input-art-name",
                "location": {
                    "type": "$3",
                    "s3Location": {
                         "bucketName": "inputBucket",
                         "objectKey": "inputKey"
                    }
                }
            }
```

```
],
        "outputArtifacts": [
            {
                "name": "output-art-name",
                "location": {
                     "type": "$3",
                     "s3Location": {
                         "bucketName": "outputBucket",
                         "objectKey": "outputKey"
                     }
                }
            }
        ],
        "actionExecutionId": "actionExecutionId",
        "actionCredentials": {
            "accessKeyId": "access-id",
            "secretAccessKey": "secret-id",
            "sessionToken": "session-id"
        },
        "continuationToken": "continueId-xxyyzz"
    }
}
```

Lambda 함수에서 CodePipeline에 결과 반환

통합자의 작업 작업자 리소스는 성공, 실패 또는 지속의 경우 유효한 페이로드를 반환해야 합니다.

형식:

- result: 작업 결과입니다.
 - 필수
 - 유효한 값(대소문자 구분 안 함):
 - Success: 작업이 성공했으며 터미널임을 나타냅니다.
 - Continue: 작업이 성공했으며 계속되어야 함을 나타냅니다(예: 동일한 작업 실행을 위해 작업 자가 다시 호출되는 경우).
 - Fail: 작업이 실패했고 터미널임을 나타냅니다.
- failureType: 실패한 작업과 관련된 실패 유형입니다.

파트너 작업의 failureType 범주는 작업을 실행하는 동안 발생한 실패 유형을 설명합니다. 통합자 는 CodePipeline에 작업 실패 결과를 반환할 때 실패 메시지와 함께 유형을 설정합니다.

- 선택 사항. 결과가 Fail인 경우 필수입니다.
- result가 Success 또는 Continue인 경우 null이어야 합니다.
- 유효한 값:
 - ConfigurationError
 - JobFailed
 - PermissionsError
 - RevisionOutOfSync
 - RevisionUnavailable
 - SystemUnavailable
- continuation: 현재 작업 실행 중 다음 작업으로 전달되는 연속 상태입니다.
 - 선택 사항. 결과가 Continue인 경우 필수입니다.
 - result가 Success 또는 Fail인 경우 null이어야 합니다.
 - 속성:
 - State: 전달될 상태의 해시입니다.
- status: 작업 실행 상태입니다.
 - 선택 사항.
 - 속성:
 - ExternalExecutionId: 작업과 연결할 선택적 외부 실행 ID 또는 커밋 ID입니다.
 - Summary: 발생한 상황에 대한 선택적 요약입니다. 실패 시나리오에서는 이 메시지가 사용자에 게 표시되는 실패 메시지가 됩니다.
- outputVariables: 다음 작업 실행 시 전달되는 키/값 페어 세트입니다.
 - 선택 사항.
 - result가 Continue 또는 Fail인 경우 null이어야 합니다.

예제:

```
{
    "result": "success",
    "failureType": null,
    "continuation": null,
    "status": {
        "externalExecutionId": "my-commit-id-123",
        "summary": "everything is dandy"
Lambda 함수에서 CodePipeline에 결과 반환
```

```
},
"outputVariables": {
    "FirstOne": "Nice",
    "SecondOne": "Nicest",
    ...
}
```

연속 토큰을 사용하여 비동기 프로세스의 결과를 기다립니다.

continuation 토큰은 Lambda 함수의 페이로드 및 결과의 일부입니다. 이는 CodePipeline에 작업 상태를 전달하고 작업을 계속해야 함을 나타내는 방법입니다. 예를 들어 통합자는 리소스에서 고객을 위한 빌드를 시작한 후 빌드가 완료될 때까지 기다리지 않고 result를 continue로 반환하고 빌드의 고유 ID를 CodePipeline에 continuation 토큰으로 반환하여 터미널 결과가 없음을 CodePipeline에 알립니다.

Note

Lambda 함수는 최대 15분 동안만 실행할 수 있습니다. 작업을 더 오래 실행해야 하는 경우 연 속 토큰을 사용할 수 있습니다.

CodePipeline 팀은 30초 후에 페이로드에 동일한 continuation 토큰으로 통합자를 호출하여 완료 여부를 확인할 수 있도록 합니다. 빌드가 완료되면 통합자는 터미널 성공/실패 결과를 반환하고 그렇지 않으면 계속됩니다.

CodePipeline에 런타임 시 통합자 Lambda 함수를 호출할 수 있는 권한을 제 공합니다.

CodePipeline 서비스 보안 주체: codepipeline.amazonaws.com을 사용하여 CodePipeline 서비스 를 호출할 수 있는 권한을 제공하는 권한을 통합자 Lambda 함수에 추가합니다. AWS CloudFormation 또는 명령줄을 사용하여 권한을 추가할 수 있습니다. 예시는 <u>작업 유형 처리</u>에서 확인하십시오.

작업자 통합 모델

상위 수준 워크플로우를 설계한 후 작업자를 생성할 수 있습니다. 타사 작업의 세부 사항이 작업자에 필요한 사항을 결정하지만 타사 작업의 대다수 작업자에는 다음 기능이 포함되어 있습니다.

• PollForThirdPartyJobs를 사용하여 CodePipeline의 작업을 폴링합니다.

- AcknowledgeThirdPartyJob, PutThirdPartyJobSuccessResult 및 PutThirdPartyJobFailureResult를 사용하여 작업 승인 및 CodePipeline에 결과 반환.
- 파이프라인에 대한 아티팩트를 Amazon S3 버킷에서 검색 및/또는 버킷에 배치. Amazon S3 버킷에 서 아티팩트를 다운로드하려면 서명 버전 4 서명(Sig V4)을 사용하는 Amazon S3 클라이언트를 생성해야 합니다. 에는 Sig V4가 필요합니다 AWS KMS.

Amazon S3 버킷에 아티팩트를 업로드하려면 AWS Key Management Service (AWS KMS). AWS KMS uses를 통한 암호화를 사용하도록 Amazon S3 <u>PutObject</u> 요청도 구성해야 합니다 AWS KMS keys. AWS 관리형 키 또는 고객 관리형 키를 사용하여 아티팩트를 업로드할지 여부를 확인하 려면 작업자가 <u>작업 데이터를</u> 살펴보고 <u>암호화 키</u> 속성을 확인해야 합니다. 속성이 설정된 경우 구성 할 때 해당 고객 관리형 키 ID를 사용해야 합니다 AWS KMS. 키 속성이 null인 경우 AWS 관리형 키 를 사용합니다. CodePipeline은 달리 구성 AWS 관리형 키 되지 않은 한를 사용합니다.

Java 또는 .NET에서 AWS KMS 파라미터를 생성하는 방법을 보여주는 예제는 <u>SDK를 사용하여</u> <u>Amazon S3 AWS Key Management Service 에서 지정을 참조하세요 AWS SDKs</u>. CodePipeline의 Amazon S3 버킷에 대한 자세한 내용은 CodePipeline 개념 섹션을 참조하세요.

작업자에 대한 권한 관리 전략 선택 및 구성

CodePipeline에서 타사 작업에 대한 작업자를 개발하려면 사용자 및 권한 관리의 통합을 위한 전략이 필요합니다.

가장 간단한 전략은 통합에 필요한 리소스를 쉽게 확장할 수 있도록 AWS Identity and Access Management (IAM) 인스턴스 역할로 Amazon EC2 인스턴스를 생성하여 작업자에게 필요한 인프라를 추가하는 것입니다. 와의 기본 통합을 사용하여 작업자와 CodePipeline 간의 상호 작용을 간소화 AWS 할 수 있습니다.

Amazon EC2에 대해 자세히 알아보고 해당 통합 사례에 적합한 선택인지 확인합니다. 자세한 내용은 <u>Amazon EC2 - 가상 서버 호스팅</u>을 참조하세요. Amazon EC2 인스턴스 설정에 대한 자세한 내용은 <u>Amazon EC2 Linux 인스턴스 시작하기</u>를 참조하세요.

고려해야 할 다른 전략은 IAM에 ID 페더레이션을 사용하여 기존의 자격 증명 공급자 시스템 및 리소스 를 통합하는 것입니다. 이 전략은 기업 자격 증명 공급자가 이미 있거나 웹 자격 증명 공급자를 사용하 여 사용자를 지원하도록 이미 구성되어 있는 경우에 유용합니다. ID 페더레이션을 사용하면 IAM 사용 자를 생성하거나 관리할 필요 없이 CodePipeline을 포함한 AWS 리소스에 대한 보안 액세스 권한을 부 여할 수 있습니다. 암호 보안 요구 사항 및 자격 증명 교체에 대한 기능과 정책을 사용할 수 있습니다. 샘플 애플리케이션을 고유한 설계를 위한 템플릿으로 사용할 수 있습니다. 자세한 내용은 <u>연동 관리</u>를 참조하십시오. 액세스 권한을 제공하려면 사용자, 그룹 또는 역할에 권한을 추가하세요:

• 의 사용자 및 그룹 AWS IAM Identity Center:

권한 세트를 생성합니다. AWS IAM Identity Center 사용 설명서의 <u>권한 세트 생성</u>의 지침을 따릅니 다.

• 보안 인증 공급자를 통해 IAM에서 관리되는 사용자:

ID 페더레이션을 위한 역할을 생성합니다. IAM 사용 설명서의 <u>Create a role for a third-party identity</u> provider (federation)의 지침을 따릅니다.

- IAM 사용자:
 - 사용자가 맡을 수 있는 역할을 생성합니다. IAM 사용 설명서에서 <u>Create a role for an IAM user</u>의 지침을 따릅니다.
 - (권장되지 않음)정책을 사용자에게 직접 연결하거나 사용자를 사용자 그룹에 추가합니다. IAM 사용 설명서에서 사용자(콘솔)에 권한 추가의 지침을 따르세요.

다음은 타사 작업자와 함께 사용하기 위해 생성할 수 있는 예제 정책입니다. 이 정책은 예로만 사용되며 있는 그대로 제공됩니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codepipeline:PollForThirdPartyJobs",
        "codepipeline:AcknowledgeThirdPartyJob",
        "codepipeline:GetThirdPartyJobDetails",
        "codepipeline:PutThirdPartyJobSuccessResult",
        "codepipeline:PutThirdPartyJobFailureResult"
      ],
      "Resource": [
        "arn:aws:codepipeline:us-east-2::actionType:ThirdParty/Build/Provider/1/"
      ]
    }
  ]
}
```

이미지 정의 파일 참조

이 단원은 참조용입니다. 컨테이너의 소스 또는 배포 작업이 포함된 파이프라인을 만드는 방법에 대한 자세한 내용은 <mark>파이프라인 스테이지 및 작업 생성</mark> 단원을 참조하십시오.

AWS CodePipeline Amazon ECR 소스 작업 또는 Amazon ECS 배포 작업과 같은 컨테이너 작업에 대 한 작업자는 정의 파일을 사용하여 이미지 URI 및 컨테이너 이름을 작업 정의에 매핑합니다. 각 정의 파일은 다음과 같이 작업 공급자가 사용하는 JSON 형식의 파일입니다.

- Amazon ECS 표준 배포에서는 imagedefinitions.json 파일을 배포 작업의 입력으로 사용해야 합니다. CodePipeline의 Amazon ECS 표준 배포 작업을 사용하는 자습서는 <u>자습서:</u> <u>CodePipeline을 사용한 Amazon ECS 표준 배포</u> 단원을 참조하세요. ECRBuildAndPublish 작업과 함께 CodePipeline의 Amazon ECS 표준 배포 작업을 사용하는 또 다른 예제 자습서는 섹션을 참조 하세요<u>자습서: CodePipeline을 사용하여 Docker 이미지를 빌드하고 Amazon ECR에 푸시(V2 유형).</u>
- Amazon ECS 블루/그린 배포에서는 imageDetail.json 파일을 배포 작업의 입력으로 사용해야 합니다. 샘플 블루/그린 배포가 포함된 자습서는 <u>자습서: Amazon ECR 소스 및 ECS-to-CodeDeploy</u> <u>배포를 사용하여 파이프라인 생성</u> 섹션을 참조하세요.
 - Amazon ECR 소스 작업은 소스 작업의 출력으로 제공되는 imageDetail.json 파일을 생성합 니다.

주제

- Amazon ECS 표준 배포 작업을 위한 imagedefinitions.json 파일
- Amazon ECS 블루/그린 배포 작업을 위한 imageDetail.json 파일

Amazon ECS 표준 배포 작업을 위한 imagedefinitions.json 파일

이미지 정의 문서는 Amazon ECS 컨테이너 이름과 이미지 및 태그를 설명하는 JSON 파일입니다. 컨 테이너 기반 애플리케이션을 배포하려는 경우 CodePipeline 작업자에게 Amazon ECR와 같은 이미지 리포지토리에서 검색할 수 있는 Amazon ECS 컨테이너와 이미지 ID를 제공하도록 이미지 정의 파일 을 생성해야 합니다.

Note

이 파일의 기본 파일 이름은 imagedefinitions.json입니다. 다른 파일 이름을 사용하려면 파이프라인 배포 단계를 만들 때 이름을 제공해야 합니다.

다음 사항을 고려하여 imagedefinitions.json 파일을 만듭니다.

- 파일은 UTF-8 인코딩을 사용해야 합니다.
- 이미지 정의 파일의 최대 파일 크기 한도는 100KB입니다.
- 배포 작업에 대한 입력 아티팩트가 되도록 파일을 소스 또는 빌드 아티팩트로 생성해야 합니다. 즉, 파일이 CodeCommit 리포지토리와 같은 소스 위치에 업로드되거나 빌드된 출력 아티팩트로 생성되 어야 합니다.

imagedefinitions.json 파일은 컨테이너 이름과 이미지 URI를 제공합니다. 이 파일은 다음 키-값 페어 세트로 생성해야 합니다.

7	값
name	container_name
imageUri	imageUri

```
₲ Note
이름 필드는 컨테이너 이미지 이름, 즉 Docker 이미지의 이름에 사용됩니다.
```

다음은 JSON 구조를 나타내는데, 여기서 컨테이너 이름은 sample-app이고, 이미지 URI는 ecsrepo이며, 태그는 latest입니다.

```
[
    {
        "name": "sample-app",
        "imageUri": "11111EXAMPLE.dkr.ecr.us-west-2.amazonaws.com/ecs-repo:latest"
    }
]
```

컨테이너-이미지 페어를 여러 개 나열하도록 파일을 생성할 수도 있습니다.

JSON 구조:

] }
```
"name": "simple-app",
    "imageUri": "httpd:2.4"
},
{
    "name": "simple-app-1",
    "imageUri": "mysql"
},
{
    "name": "simple-app-2",
    "imageUri": "java1.8"
}
```

파이프라인을 만들기 전에 다음 단계를 사용하여 imagedefinitions.json 파일을 설정합니다.

- 파이프라인에 대한 컨테이너 기반 애플리케이션 배포를 계획하는 과정에서 해당되는 경우 소스 단계와 빌드 단계를 계획합니다.
- 2. 다음 중 하나를 선택합니다.
 - a. 파이프라인이 생성되어 빌드 단계를 건너뛴 경우 JSON 파일을 수동으로 만들고 소스 작업이 아티팩트를 제공할 수 있도록 이를 소스 리포지토리에 업로드해야 합니다. 텍스트 편집기를 사용하여 파일을 만든 후, 파일 이름을 지정하거나 기본 imagedefinitions.json 파일 이 름을 사용합니다. 이미지 정의 파일을 소스 리포지토리에 푸시합니다.

1 Note

소스 리포지토리가 Amazon S3 버킷인 경우, JSON 파일을 압축해야 합니다.

b. 파이프라인에 빌드 단계가 있는 경우 빌드 단계 중에 소스 리포지토리의 이미지 정의 파일 을 출력하는 명령을 빌드 사양 파일에 추가합니다. 다음 예제에서는 printf 명령을 사용하여 imagedefinitions.json 파일을 생성합니다. buildspec.yml 파일의 post_build 섹 션에 이 명령을 나열합니다.

```
printf '[{"name":"container_name","imageUri":"image_URI"}]' >
imagedefinitions.json
```

이미지 정의 파일을 buildspec.yml 파일에 출력 아티팩트로 포함시켜야 합니다.

 콘솔에서 파이프라인을 생성할 때 파이프라인 생성 마법사의 배포 페이지에 있는 이미지 파일 이 름에 이미지 정의 파일 이름을 입력합니다. Amazon ECS를 배포 공급자로 사용하는 파이프라인 생성에 대한 단계별 자습서는 <u>자습서:</u> CodePipeline을 사용하여 연속 배포 단원을 참조하세요.

Amazon ECS 블루/그린 배포 작업을 위한 imageDetail.json 파일

imageDetail.json 문서는 Amazon ECS 이미지 URI를 설명하는 JSON 파일입니다. 블루/그 린 배포를 위해 컨테이너 기반 애플리케이션을 배포하는 경우 Amazon ECS 및 CodeDeploy 작 업자에게 Amazon ECR과 같은 이미지 리포지토리에서 검색할 수 있는 이미지 ID를 제공하도록 imageDetail.json 파일을 생성해야 합니다.

Note

파일의 이름은 imageDetail.json이어야 합니다.

작업 및 해당 파라미터의 설명은 <u>Amazon Elastic Container Service(Amazon ECS) 및 CodeDeploy 블</u> 루-그린 배포 작업 참조을 참조하세요.

배포 작업에 대한 입력 아티팩트가 되도록 imageDetail.json 파일을 소스 또는 빌드 아티팩트로 생 성해야 합니다. 다음과 같은 방법 중 하나를 통해 파이프라인에 imageDetail.json 파일을 제공할 수 있습니다.

• Amazon ECS 블루/그린 배포 작업에 대한 입력으로 파이프라인에 제공되도록 소스 위치에 imageDetail.json 파일을 포함합니다.

Note

소스 리포지토리가 Amazon S3 버킷인 경우, JSON 파일을 압축해야 합니다.

• Amazon ECR 소스 작업은 자동으로 imageDetail.json 파일을 다음 작업에 대한 입력 아티팩트 로 생성합니다.

Note

Amazon ECR 소스 작업이 이 파일을 작성하므로 Amazon ECR 소스 작업이 있는 파이프라 인은 imageDetail.json 파일을 수동으로 제공할 필요가 없습니다.

Amazon ECR 소스 단계가 포함된 파이프라인을 만드는 방법에 대한 자습서는 <u>자습서:</u> <u>Amazon ECR 소스 및 ECS-to-CodeDeploy 배포를 사용하여 파이프라인 생성</u> 단원을 참조 하세요.



imageDetail.json 파일은 이미지 URI를 제공합니다. 이 파일은 다음 키-값 페어로 생성해야 합니 다.

7	값
imageURI	image_URI

imageDetail.json

다음은 이미지 URI가 ACCOUNTID.dkr.ecr.us-west-2.amazonaws.com/dk-imagerepo@sha256:example3인 JSON 구조입니다.

```
{
"ImageURI": "ACCOUNTID.dkr.ecr.us-west-2.amazonaws.com/dk-image-
repo@sha256:example3"
}
```

imageDetail.json (generated by ECR)

변경 사항이 이미지 리포지토리로 푸시될 때마다 Amazon ECR 소스 작업에 의해 imageDetail.json 파일이 자동으로 생성됩니다. Amazon ECR 소스 작업에 의해 생성된 imageDetail.json은 소스 작업의 출력 아티팩트로 파이프라인의 다음 작업에 제공됩니다.

다음은 리포지토리 이름이 dk-image-repo이고, 이미지 URI가 ecs-repo이며, 이미지 태그가 latest인 JSON 구조입니다.

```
{
    "ImageSizeInBytes": "44728918",
    "ImageDigest":
    "sha256:EXAMPLE11223344556677889900bfea42ea2d3b8a1ee8329ba7e68694950afd3",
    "Version": "1.0",
    "ImagePushedAt": "Mon Jan 21 20:04:00 UTC 2019",
    "RegistryId": "EXAMPLE12233",
    "RepositoryName": "dk-image-repo",
    "ImageURI": "ACCOUNTID.dkr.ecr.us-west-2.amazonaws.com/dk-image-
repo@sha256:example3",
    "ImageTags": [
        "latest"
    ]
}
```

imageDetail.json 파일은 다음과 같이 이미지 URI와 컨테이너 이름을 Amazon ECS 작업 정의 에 매핑합니다.

- ImageSizeInBytes: 리포지토리의 이미지 크기(바이트)
- ImageDigest: 이미지 매니페스트의 sha256 다이제스트
- Version: 이미지 버전

- ImagePushedAt: 최신 이미지가 리포지토리로 푸시된 날짜와 시간
- RegistryId: 리포지토리가 포함된 레지스트리와 연결된 AWS 계정 ID입니다.
- RepositoryName: 이미지가 푸시된 Amazon ECR 리포지토리의 이름입니다.
- ImageURI: 이미지의 URI
- ImageTags: 이미지에 사용된 태그

파이프라인을 만들기 전에 다음 단계를 사용하여 imageDetail.json 파일을 설정합니다.

- 파이프라인에 대한 컨테이너 기반 애플리케이션 블루/그린 배포를 계획하는 과정에서 해당하는 경우 소스 단계와 빌드 단계를 계획합니다.
- 2. 다음 중 하나를 선택합니다.
 - a. 파이프라인에서 빌드 단계를 건너뛴 경우 JSON 파일을 수동으로 만들고 소스 작업이 아티팩 트를 제공할 수 있도록 이를 소스 리포지토리(예: CodeCommit)에 업로드해야 합니다. 텍스트 편집기를 사용하여 파일을 만든 후, 파일 이름을 지정하거나 기본 imageDetail.json 파일 이름을 사용합니다. imageDetail.json 파일을 해당 소스 리포지토리에 푸시합니다.
 - b. 파이프라인에 빌드 단계가 있는 경우 다음을 수행하십시오.
 - i. 빌드 단계 중에 소스 리포지토리의 이미지 정의 파일을 출력하는 명령을 빌드 사양 파일 에 추가합니다. 다음 예제에서는 printf 명령을 사용하여 imageDetail.json 파일을 생 성합니다. buildspec.yml 파일의 post_build 섹션에 이 명령을 나열합니다.

```
printf '{"ImageURI":"image_URI"}' > imageDetail.json
```

imageDetail.json 파일을 buildspec.yml 파일에 출력 아티팩트로 포함시켜야 합니다.

ii. imageDetail.json을 buildspec.yml 파일에 아티팩트 파일로 추가합니다.

```
artifacts:
files:
- imageDetail.json
```

변수 참조

이 단원은 참조용입니다. 변수 생성에 대한 자세한 내용은 변수 작업 단원을 참조하십시오.

변수를 사용하면 파이프라인 실행 또는 작업 실행 시 결정되는 값으로 파이프라인 작업을 구성할 수 있 습니다.

일부 작업 공급자는 정의된 변수 집합을 생성합니다. 해당 작업 공급자의 기본 변수 키에서 선택합니다 (예: 커밋 ID).

A Important

보안 파라미터를 전달할 때는 값을 직접 입력하지 마십시오. 값은 일반 텍스트로 렌더링되므로 읽을 수 있습니다. 보안상의 이유로 암호가 포함된 일반 텍스트는 사용하지 마십시오. 보안 암 호를 저장하는 AWS Secrets Manager 데를 사용하는 것이 좋습니다.

변수 사용에 대한 단계별 예제를 보려면

- 파이프라인 실행 시 전달되는 파이프라인 수준 변수에 대한 자습서는 <u>자습서: 파이프라인 수준 변수</u> 사용을 참조하세요.
- 업스트림 작업(CodeCommit)의 변수를 사용하고 출력 변수를 생성하는 Lambda 작업에 대한 자습서 는 자습서: Lambda 호출 작업과 함께 변수 사용 섹션을 참조하세요.
- 업스트림 CloudFormation AWS CloudFormation 작업의 스택 출력 변수를 참조하는 작업이 포함된 자습서는 섹션을 참조하세요<u>자습서: AWS CloudFormation 배포 작업의 변수를 사용하는 파이프라</u> 인 생성.
- CodeCommit 커밋 ID 및 커밋 메시지로 확인되는 출력 변수를 참조하는 메시지 텍스트가 포함된 수 동 승인 작업의 예는 예: 수동 승인에 변수 사용 섹션을 참조하세요.
- GitHub 브랜치 이름으로 확인되는 환경 변수를 포함하는 CodeBuild 작업의 예는 <u>예: CodeBuild 환경</u> 변수와 함께 BranchName 변수 사용 섹션을 참조하세요.
- CodeBuild 작업은 빌드의 일부로 내보낸 모든 환경 변수를 변수로 생성합니다. 자세한 내용은 <u>CodeBuild 작업 출력 변수</u> 단원을 참조하십시오.

변수 제한

제한 정보는 in AWS CodePipeline 할당량 단원을 참조하십시오.

Note

작업 구성 필드에 변수 구문을 입력할 때 구성 필드의 1000자 제한을 초과하지 마십시오. 이 제 한을 초과하면 확인 오류가 반환됩니다.

주제

- <u>개념</u>
- 변수 사용 사례
- 변수 구성
- 변수 확인
- 변수에 대한 규칙
- 파이프라인 작업에 사용할 수 있는 변수

개념

이 단원에서는 변수 및 네임스페이스와 관련된 주요 용어와 개념을 설명합니다.

변수

변수는 파이프라인에서 작업을 동적으로 구성하는 데 사용할 수 있는 키-값 페어입니다. 현재 이러한 변수를 사용할 수 있는 세 가지 방법이 있습니다.

- 각 파이프라인 실행을 시작할 때 묵시적으로 사용할 수 있는 변수 집합이 있습니다. 이 집합에는 현 재 파이프라인 실행의 ID인 PipelineExecutionId이 포함되어 있습니다.
- 파이프라인 수준의 변수는 파이프라인이 생성되고 파이프라인 런타임에서 해결될 때 정의됩니다.

파이프라인 생성 시 파이프라인 수준 변수를 지정하고 파이프라인 실행 시 값을 제공할 수 있습니다.

 작업이 실행될 때 변수 집합을 생성하는 작업 유형이 있습니다. ListActionExecutions API의 일부인 outputVariables 필드를 검사하여 작업에 의해 생성된 변수를 확인할 수 있습니다. 작업 공급자 별로 사용 가능한 키 이름의 목록은 <u>파이프라인 작업에 사용할 수 있는 변수</u> 단원을 참조하십시오. 각 작업 유형이 생성하는 변수를 확인하려면 CodePipeline 작업 구조 참조 단원을 참조하세요.

작업 구성에서 이러한 변수를 참조하려면 올바른 네임스페이스와 함께 변수 참조 구문을 사용해야 합 니다. 변수 워크플로의 예는 변수 구성 단원을 참조하십시오.

네임스페이스

변수를 고유하게 참조할 수 있도록 하려면 해당 변수를 네임스페이스에 할당해야 합니다. 네임스페이 스에 할당된 변수 집합이 있다면 다음 구문과 함께 네임스페이스 및 변수 키를 사용하여 작업 구성에서 이를 참조할 수 있습니다.

#{namespace.variable_key}

세 가지 유형의 네임스페이스 하에서 변수를 할당할 수 있습니다.

• 코드 파이프라인 예약 네임스페이스

각 파이프라인 실행을 시작할 때 사용할 수 있는 암묵적 변수 집합에 할당되는 네임스페이스입니다. 이 네임스페이스는 codepipeline입니다. 변수 참조 예:

#{codepipeline.PipelineExecutionId}

• 파이프라인 수준의 변수 네임스페이스

이는 파이프라인 수준의 변수에 할당된 네임스페이스입니다. 파이프라인 수준의 모든 변수에 대한 네임스페이스는 variables입니다. 변수 참조 예:

#{variables.variable_name}

• 작업에 할당된 네임스페이스

작업에 할당하는 네임스페이스입니다. 작업에서 생성되는 모든 변수가 이 네임스페이스에 속합니 다. 작업에서 생성되는 변수를 다운스트림 작업 구성에서 사용할 수 있도록 하려면 네임스페이스를 사용하여 생성 작업을 구성해야 합니다. 네임스페이스는 파이프라인 정의에서 고유해야 하며 어떤 아티팩트 이름과도 충돌해서는 안 됩니다. 다음은 네임스페이스 SourceVariables로 구성된 작업 에 대한 변수 참조 예제입니다.

#{SourceVariables.VersionId}

변수 사용 사례

다음은 파이프라인 수준의 가장 일반적인 몇 가지 변수 사용 사례로, 특정 요구 사항에 맞게 변수를 사 용하는 방법을 결정하는 데 도움이 됩니다.

- 파이프라인 수준의 변수는 매번 동일한 파이프라인을 사용하면서 작업 구성 입력에 약간의 변형을 주고자 하는 CodePipeline 고객을 위한 것입니다. 파이프라인을 시작하는 모든 개발자는 파이프라인 시작 시 UI에 변수 값을 추가합니다. 이 구성에서는 해당 실행에 대한 파라미터만 전달합니다.
- 파이프라인 수준 변수를 사용하면 파이프라인의 작업에 동적 입력을 전달할 수 있습니다. 동일한 파 이프라인의 다른 버전을 유지 관리하거나 복잡한 파이프라인을 생성할 필요 없이 파라미터화된 파 이프라인을 CodePipeline으로 마이그레이션할 수 있습니다.
- 파이프라인 수준 변수를 사용하여 입력 파라미터를 전달하면 각 실행에서 파이프라인을 재사용할 수 있습니다. 예를 들어 프로덕션 환경에 배포할 버전을 지정하려는 경우 파이프라인을 복제하지 않 아도 됩니다.
- 단일 파이프라인을 사용하여 여러 빌드 및 배포 환경에 리소스를 배포할 수 있습니다. 예를 들어 CodeCommit 리포지토리가 있는 파이프라인의 경우 파이프라인 수준에서 전달된 CodeBuild 및 CodeDeploy 파라미터를 사용하여 지정된 브랜치 및 대상 배포 환경에서 배포할 수 있습니다.

Note

Amazon ECR, Amazon S3 또는 CodeCommit 소스의 경우 입력 변환 항목을 사용하여 파이프 라인 이벤트에 EventBridgerevisionValue의를 사용하는 소스 재정의를 생성할 수도 있습니 다. 여기서 revisionValue는 객체 키, 커밋 또는 이미지 ID에 대한 소스 이벤트 변수에서 파 생됩니다. 자세한 내용은, <u>Amazon ECR 소스 작업 및 EventBridge 리소스 이벤트에 대해 활성</u> <u>화된 소스를 사용하여 Amazon S3 소스 작업에 연결</u>또는의 절차에 포함된 입력 변환 항목의 선 택적 단계를 참조하세요CodeCommit 소스 작업 및 EventBridge.

변수 구성

파이프라인 구조의 파이프라인 수준 또는 작업 수준에서 변수를 구성할 수 있습니다.

파이프라인 수준에서 변수 구성

파이프라인 수준에서 변수를 하나 이상 추가할 수 있습니다. CodePipeline 작업 구성에서 이 값을 참조 할 수 있습니다. 파이프라인을 생성할 때 변수 이름, 기본값, 설명을 추가할 수 있습니다. 변수는 실행 시 확인됩니다.

Note

파이프라인 수준에서 변수의 기본값이 정의되지 않은 경우 변수는 필요한 것으로 간주됩니다. 파이프라인을 시작할 때 모든 필수 변수에 대한 재정의를 지정해야 합니다. 그렇지 않으면 파 이프라인 실행이 실패하고 검증 오류가 발생합니다.

파이프라인 구조의 변수 속성을 사용하여 파이프라인 수준에서 변수를 제공합니다. 다음 예제에서 변 수 Variable1 값은 Value1입니다.

파이프라인 JSON 구조의 예는 파이프라인 스테이지 및 작업 생성 단원을 참조하세요.

파이프라인 실행 시 전달되는 파이프라인 수준 변수에 대한 자습서는 <u>자습서: 파이프라인 수준 변수 사</u> 용을 참조하세요.

단, 모든 종류의 소스 작업에서 파이프라인 수준 변수를 사용하는 것은 지원되지 않습니다.

Note

파이프라인 내 일부 작업에 variables 네임스페이스가 이미 사용되고 있는 경우 작업 정의를 업데이트하고 충돌하는 작업에 사용할 다른 네임스페이스를 선택해야 합니다.

작업 수준에서 변수 구성

작업의 네임스페이스를 선언하여 변수를 생성하도록 작업을 구성합니다. 작업은 변수를 생성하는 작 업 공급자 중 하나여야 합니다. 그렇지 않으면 파이프라인 수준의 변수가 사용 가능한 변수가 됩니다.

다음과 같이 네임스페이스를 선언합니다.

• 콘솔의 작업 편집 페이지에서 Variable namespace(변수 네임스페이스)에 네임스페이스를 입력합니다.

• JSON 파이프라인 구조의 namespace 파라미터 필드에 네임스페이스를 입력합니다.

이 예제에서는 이름 SourceVariables를 사용하여 CodeCommit 소스 작업에 namespace 파라미터 를 추가합니다. 이렇게 하면 CommitId 같은 작업 공급자가 사용할 수 있는 변수를 생성하도록 작업이 구성됩니다.

```
{
    "name": "Source",
    "actions": [
        {
            "outputArtifacts": [
                {
                     "name": "SourceArtifact"
                }
            ],
            "name": "Source",
            "namespace": "SourceVariables",
            "configuration": {
                "RepositoryName": "MyRepo",
                "BranchName": "mainline",
                "PollForSourceChanges": "false"
            },
            "inputArtifacts": [],
            "region": "us-west-2",
            "actionTypeId": {
                "provider": "CodeCommit",
                "category": "Source",
                "version": "1",
                "owner": "AWS"
            },
            "runOrder": 1
        }
    ]
},
```

그런 다음, 이전 작업에서 생성된 변수를 사용하도록 다운스트림 작업을 구성합니다. 그 방법은 다음과 같습니다.

- 콘솔의 작업 편집 페이지에서 작업 구성 필드에 변수 구문을 입력(다운스트림 작업의 경우)
- JSON 파이프라인 구조의 작업 구성 필드에 변수 구문을 입력(다운스트림 작업의 경우)

이 예제에서 빌드 작업의 구성 필드에는 작업 실행 시 업데이트되는 환경 변수가 표시됩니다. 이 예제 에서는 #{codepipeline.PipelineExecutionId}을 사용하여 실행 ID에 대한 네임스페이스와 변 수를 지정하고 #{SourceVariables.CommitId}를 사용하여 커밋 ID에 대한 네임스페이스 및 변수 를 지정합니다.

```
{
    "name": "Build",
    "actions": [
        {
            "outputArtifacts": [
                {
                    "name": "BuildArtifact"
                }
            ],
            "name": "Build",
            "configuration": {
                "EnvironmentVariables": "[{\"name\":\"Release_ID\",\"value\":
\"#{codepipeline.PipelineExecutionId}\", \"type\":\"PLAINTEXT\"}, {\"name\":\"Commit_ID
\",\"value\":\"#{SourceVariables.CommitId}\",\"type\":\"PLAINTEXT\"}]",
                "ProjectName": "env-var-test"
            },
            "inputArtifacts": [
                {
                    "name": "SourceArtifact"
                }
            ],
            "region": "us-west-2",
            "actionTypeId": {
                "provider": "CodeBuild",
                "category": "Build",
                "version": "1",
                "owner": "AWS"
            },
            "runOrder": 1
        }
    ]
},
```

변수 확인

작업이 파이프라인 실행의 일부로 실행될 때마다 생성되는 변수를 생성 작업 이후에 수행되도록 보 장된 모든 작업에 사용할 수 있습니다. 소비 작업에서 이러한 변수를 사용하려면 앞의 예제에 나와 있는 구문을 사용하여 소비 작업의 구성에 변수를 추가할 수 있습니다. 소비 작업을 수행하기 전에 CodePipeline에서는 작업 실행을 시작하기 전에 구성에 존재하는 모든 변수 참조를 확인합니다.



변수에 대한 규칙

다음 규칙은 변수 구성에 도움이 됩니다.

- 새 작업 속성을 사용하거나 작업을 편집하여 작업의 네임스페이스와 변수를 지정합니다.
- 파이프라인 생성 마법사를 사용하면 콘솔에서 마법사에서 생성된 각 작업에 대해 네임스페이스가 생성됩니다.
- 네임스페이스가 지정되지 않은 경우, 해당 작업에서 생성된 변수는 작업 구성에서 참조할 수 없습니
 다.
- 작업에서 생성된 변수를 참조하려면 변수를 생성하는 작업 이후에 참조 작업이 이루어져야 합니다.
 이는 해당 작업이 변수를 생성하는 작업보다 나중 단계에 있거나 동일한 단계이지만 실행 순서가 거 빠르다는 것을 의미합니다.

파이프라인 작업에 사용할 수 있는 변수

작업 공급자는 작업에서 생성할 수 있는 변수를 결정합니다.

변수 관리를 위한 단계별 절차는 변수 작업 단원을 참조하십시오.

정의된 변수 키가 있는 작업

선택이 가능한 네임스페이스와 달리 다음과 같은 작업은 편집이 불가능한 변수 키를 사용합니다. 예를 들어 Amazon S3 작업 공급자의 경우, ETag 및 VersionId 변수 키만 사용할 수 있습니다.

또한 각 실행은 파이프라인 릴리스 ID와 같이 실행에 대한 데이터를 포함하는 CodePipeline에서 생성 된 파이프라인 변수 집합을 가지고 있습니다. 이러한 변수는 파이프라인의 모든 작업에서 사용될 수 있 습니다.

주제

- <u>CodePipeline 실행 ID 변수</u>
- Amazon ECR 작업 출력 변수
- AWS CloudFormation StackSets 작업 출력 변수
- CodeCommit 작업 출력 변수
- CodeStarSourceConnection 작업 출력 변수
- GitHub 작업 출력 변수(GitHub(OAuth 앱 사용) 작업)
- S3 작업 출력 변수

CodePipeline 실행 ID 변수

CodePipeline 실행 ID 변수

공급자	변수 키	예시 값	예제 변수 구문
CodePipeline	PipelineE xecutionId	8abc75f0-fbf8-4f4c- bfEXAMPLE	<pre>#{codepip eline.Pip elineExec utionId}</pre>

변수 키	예시 값	예제 변수 구문
ImageDigest	sha256:EXAMPLE1122334455	#{SourceVariables. ImageDigest}
ImageTag	최신	#{SourceVariables. ImageTag}
ImageURI	11111EXAMPLE.dkr.ecr.us-wes t-2.amazonaws.com/ecs-repo: latest	#{SourceVariables. ImageURI}
RegistryId	EXAMPLE12233	#{SourceVariables. RegistryId}
RepositoryName	my-image-repo	<pre>#{SourceVariables. RepositoryName}</pre>

AWS CloudFormation StackSets 작업 출력 변수

AWS CloudFormation StackSets 변수

변수 키	예시 값	예제 변수 구문
OperationId	11111111-2bbb-111-2bbb-1111 1example	#{DeployVariables. OperationId}
StackSetId	my-stackset:1111aaaa-1111-2 222-2bbb-11111example	#{DeployVariables. StackSetId}

CodeCommit 작업 출력 변수

CodeCommit 변수

변수 키	예시 값	예제 변수 구문
AuthorDate	2019-10-29T03:32:21Z	#{SourceVariables. AuthorDate}
BranchName	개발	#{SourceVariables. BranchName}
CommitId	exampleb01f91b31	#{SourceVariables. CommitId}
CommitMessage	버그 수정(최대 크기 100KB)	#{SourceVariables. CommitMessage}
CommitterDate	2019-10-29T03:32:21Z	#{SourceVariables. CommitterDate}
RepositoryName	myCodeCommitRepo	#{SourceVariables. RepositoryName}

CodeStarSourceConnection 작업 출력 변수

CodeStarSourceConnection 변수(Bitbucket Cloud, GitHub, GitHub Enterprise Repository 및 GitLab.com)

변수 키	예시 값	예제 변수 구문
AuthorDate	2019-10-29T03:32:21Z	#{SourceVariables. AuthorDate}
BranchName	개발	#{SourceVariables. BranchName}
CommitId	exampleb01f91b31	#{SourceVariables. CommitId}

변수 키	예시 값	예제 변수 구문
CommitMessage	버그 수정(최대 크기 100KB)	#{SourceVariables. CommitMessage}
ConnectionArn	arn:aws:codestar-connection s:region:account-id :connecti on/connection-id	#{SourceVariables. ConnectionArn}
FullRepositoryName	사용자 이름/GitHubRepo	<pre>#{SourceVariables. FullRepositoryName}</pre>

GitHub 작업 출력 변수(GitHub(OAuth 앱 사용) 작업)

GitHub 변수(GitHub(OAuth 앱 사용) 작업)

변수 키	예시 값	예제 변수 구문
AuthorDate	2019-10-29T03:32:21Z	#{SourceVariables. AuthorDate}
BranchName	기본	#{SourceVariables. BranchName}
CommitId	exampleb01f91b31	#{SourceVariables. CommitId}
CommitMessage	버그 수정(최대 크기 100KB)	#{SourceVariables. CommitMessage}
CommitterDate	2019-10-29T03:32:21Z	#{SourceVariables. CommitterDate}
CommitUrl		#{SourceVariables. CommitUrl}
RepositoryName	myGitHubRepo	<pre>#{SourceVariables. RepositoryName}</pre>

S3 작업 출력 변수

S3 변수

변수 키	예시 값	예제 변수 구문
ETag	example28be1c3	#{SourceVariables. ETag}
VersionId	exampleta_IUQCv	#{SourceVariables. VersionId}

사용자가 구성한 변수 키를 사용한 작업

CodeBuild AWS CloudFormation및 Lambda 작업의 경우 사용자가 변수 키를 구성합니다.

주제

- <u>CloudFormation 작업 출력 변수</u>
- CodeBuild 작업 출력 변수
- Lambda 작업 출력 변수

CloudFormation 작업 출력 변수

AWS CloudFormation 변수

변수 키 예제 빈	변수 구문
AWS CloudFormation 작업의 경우 변수는 스택 템플릿의 #{Dep Outputs 섹션에 지정된 모든 값에서 생성됩니다. 출력을 생성 하는 유일한 CloudFormation 작업 모드는 스택 생성, 스택 업데 이트 및 변경 세트 실행과 같이 스택을 생성하거나 업데이트하 는 모드입니다. 변수를 생성하는 해당 작업 모드는 다음과 같습 니다.	oloyVariables. <name}< td=""></name}<>
CREATE_UPDATECHANGE_SET_EXECUTE	
CHANGE_SET_REPLACE	

변수 키	예제 변수 구문
REPLACE_ON_FAILURE	
이러한 작업 모드에 대한 자세한 내용은 <u>AWS CloudFormation</u> <u>배포 작업 참조</u> 단원을 참조하세요. AWS CloudFormation 출력 변수를 사용하는 파이프라인에서 AWS CloudFormation 배포 작업을 사용하여 파이프라인을 생성하는 방법을 보여주는 자 습서는 섹션을 참조하세요 <u>자습서: AWS CloudFormation 배포</u> <u>작업의 변수를 사용하는 파이프라인 생성</u> .	

CodeBuild 작업 출력 변수

CodeBuild 변수

변수 키	예제 변수 구문
CodeBuild 작업의 경우 내보낸 환경 변수로 생성된 값에서 변 수가 생성됩니다. CodePipeline에서 CodeBuild 작업을 편집하 거나 빌드 사양에 환경 변수를 추가하여 CodeBuild 환경 변수 를 설정합니다.	<pre>#{BuildVariables.EnvVar}</pre>
CodeBuild 빌드 사양에 지침을 추가하여 내보낸 변수 섹션 아 래에 환경 변수를 추가하세요.AWS CodeBuild 사용 안내서의 <u>env/exported-variables</u> 를 참조하세요.	

Lambda 작업 출력 변수

Lambda 변수

변수 키	예제 변수 구문
Lambda 작업을 수행하면 <u>PutJobSuccessResult API</u> 요청의 outputVariables 섹션에 포함된 모든 키-값 페어가 변수로 생성됩니다.	#{TestVariables.te stRunId}

변수 키	예제 변수 구문
업스트림 작업(CodeCommit)의 변수를 사용하고 출력 변수를	
생성하는 Lambda 작업에 대한 자습서는 <u>자습서: Lambda 호출</u>	
<u>작업과 함께 변수 사용</u> 섹션을 참조하세요.	

구문에서 glob 패턴 작업

파이프라인 아티팩트 또는 소스 위치에서 사용되는 파일 또는 경로를 지정하는 경우 작업 유형에 따라 아티팩트를 지정할 수 있습니다. 예를 들어, S3 작업의 경우 S3 객체 키를 지정합니다.

트리거의 경우 필터를 지정할 수 있습니다. glob 패턴을 사용하여 필터를 지정할 수 있습니다. 예를 들 면 다음과 같습니다.

구문이 "glob"인 경우 정규 표현식과 유사한 구문을 가진 제한된 패턴 언어를 사용하여 경로의 문자열 표현이 일치됩니다. 예시:

- *. java: .java로 끝나는 파일 이름을 나타내는 경로를 지정합니다.
- *.*: 점이 포함된 파일 이름을 지정합니다.
- *.{java, class}: .java 또는.class로 끝나는 파일 이름을 지정합니다.
- foo.?: foo.로 시작하고 단일 문자 확장자를 가진 파일 이름을 지정합니다.

glob 패턴을 해석하는 데 사용되는 규칙은 다음과 같습니다.

- 디렉터리 경계에서 이름 구성 요소의 문자를 0개 이상 지정하려면 *를 사용합니다.
- 디렉터리 경계를 가로지르는 이름 구성 요소의 문자를 0개 이상 지정하려면 **를 사용합니다.
- 이름 구성 요소의 한 문자를 지정하려면 ?를 사용합니다.
- 특수 문자로 해석될 수 있는 문자를 이스케이프하려면 백슬래시 문자(\)를 사용합니다.
- 문자 집합 중에서 단일 문자를 지정하려면 []를 사용합니다.
- 빌드 위치 또는 소스 리포지토리 위치의 루트에 있는 단일 파일을 지정하려면 my-file.jar를 사용 합니다..
- 하위 디렉터리에 단일 파일을 지정하려면 directory/my-file.jar 또는 directory/ subdirectory/my-file.jar를 사용합니다.
- 모든 파일을 지정하려면 "**"를 사용합니다. ** glob 패턴은 임의의 수의 하위 디렉터리와 일치함 을 나타냅니다.
- directory라는 디렉터리에 있는 모든 파일 및 디렉터리를 지정하려면 "directory/**"를 사용 합니다. ** glob 패턴은 임의의 수의 하위 디렉터리와 일치함을 나타냅니다.
- directory라는 디렉터리의 모든 파일을 지정하되 해당 하위 디렉터리는 지정하지 않으려면 "directory/*"를 사용합니다.

- 괄호 표현식 내에 *, ? 및 \ 문자는 자체로 대응됩니다. 취소할 때 (-) 문자가 괄호 내 첫 번째 문자이 거나 ! 다음 첫 번째 문자이면 (-) 문자는 자체로 대응합니다.
- 부패턴이 그룹 내에서 대등하면 { } 문자는 그룹과 대응하는 부패턴 그룹입니다. "," 문자는 부 패턴을 구분하는 기호입니다. 그룹은 중첩될 수 없습니다.

폴링 파이프라인을 권장되는 변경 감지 방법으로 업데이트

소스 변경에 대응하기 위해 폴링을 사용하는 파이프라인이 있는 경우, 권장되는 감지 방법을 사용하도 록 파이프라인을 업데이트할 수 있습니다. 권장되는 이벤트 기반 변경 감지 방법을 사용하도록 폴링 파 이프라인을 업데이트하는 방법에 대한 지침이 포함된 마이그레이션 가이드는 <u>이벤트 기반 변경 감지</u> 를 사용하도록 폴링 파이프라인 마이그레이션을 참조하세요.

GitHub(OAuth 앱 사용) 소스 작업을 GitHub(GitHub 앱 사용) 소스 작업으로 업데이트

에는 GitHub 소스 작업의 지원되는 두 AWS CodePipeline가지 버전이 있습니다.

- 권장 사항: GitHub(GitHub App을 통해) 작업은 <u>Bitbucket Cloud, GitHub, GitHub Enterprise Server,</u> <u>GitLab.com 및 GitLab 자체 관리형 작업용 CodeStarSourceConnection</u> 리소스가 지원하는 Github 앱 기반 인증을 사용합니다. GitHub에서 액세스를 관리할 수 있도록 GitHub 조직에 AWS CodeStar Connections 애플리케이션을 설치합니다.
- 권장되지 않음: GitHub(OAuth 앱 사용) 작업은 OAuth 토큰을 사용하여 GitHub로 인증하고 별도의 웹후크를 사용하여 변경 사항을 감지합니다. 이 방법은 더 이상 권장되지 않습니다.

Note

아시아 태평양(홍콩), 아시아 태평양(하이데라바드), 아시아 태평양(자카르타), 아시아 태 평양(멜버른), 아시아 태평양(오사카), 아프리카(케이프타운), 중동(바레인), 중동(UAE), 유 럽(스페인), 유럽(취리히), 이스라엘(텔아비브) 또는 AWS GovCloud(미국 서부) 리전에서 는 연결을 사용할 수 없습니다. 사용 가능한 다른 작업을 참조하려면 <u>CodePipeline과 제</u> 품 및 서비스 통합을 참조하세요. 유럽(밀라노) 리전에서 이 조치를 고려할 경우 <u>Bitbucket</u> <u>Cloud, GitHub, GitHub Enterprise Server, GitLab.com 및 GitLab 자체 관리형 작업용</u> <u>CodeStarSourceConnection</u>의 참고 사항을 참조하세요.

GitHub(OAuth 앱 사용) 작업 대신 GitHub(GitHub GitHub 앱 사용) 작업을 사용하면 다음과 같은 몇 가 지 중요한 이점이 있습니다.

- 연결을 통해 CodePipeline은 더 이상 OAuth 앱이나 개인 액세스 토큰이 없어도 리포지토리에 액세 스할 수 있습니다. 연결을 만들 때 GitHub 리포지토리에 대한 인증을 관리하고 조직 수준에서 권한을 허용하는 GitHub 앱을 설치합니다. 리포지토리에 액세스하려면 사용자로서 OAuth 토큰을 승인해야 합니다. 앱 기반 GitHub 액세스와 대조되는 OAuth 기반 GitHub 액세스에 대한 자세한 내용은 <u>https://</u> <u>docs.github.com/en/developers/apps/differences-between-github-apps-and-oauth-apps</u>을 참조하세 요.
- CLI 또는 CloudFormation에서 GitHub(GitHub App을 통해) 작업을 관리할 때 더 이상 Secrets Manager에 개인 액세스 토큰을 보안 암호로 저장할 필요가 없습니다. 더 이상 CodePipeline 작업 구 성에서 저장된 암호를 동적으로 참조할 필요가 없습니다. 대신 작업 구성에 연결 ARN을 추가합니다.

예제 작업 구성은 <u>Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab.com 및 GitLab 자체 관</u> 리형 작업용 CodeStarSourceConnection을 참조하세요.

- CodePipeline에서 GitHub(GitHub App을 통해) 작업에 사용할 연결 리소스를 생성할 때 동일한 연결 리소스를 사용하여 CodeGuru Reviewer와 같은 지원되는 다른 서비스를 리포지토리와 연결할 수 있 습니다.
- Github(GitHub 앱을 통해)에서는 리포지토리를 복제하여 후속 CodeBuild 작업에서 git 메타데이터에 액세스할 수 있지만, Github(OAuth 앱을 통해)에서는 소스만 다운로드할 수 있습니다.
- 관리자가 조직의 리포지토리용 앱을 설치합니다. 토큰을 생성한 개인에 따라 달라지는 OAuth 토큰 을 더 이상 추적할 필요가 없습니다.

조직에 설치된 모든 앱은 동일한 리포지토리 세트에 액세스할 수 있습니다. 각 리포지토리에 액세스할 수 있는 사용자를 변경하려면 각 연결에 대한 IAM 정책을 수정하세요. 예를 들면 <u>예: 지정된 리포지토</u> 리에 연결을 사용하기 위한 권한 범위 축소 정책을 참조하세요.

이 주제의 단계를 사용하여 GitHub(OAuth 앱 사용) 소스 작업을 삭제하고 CodePipeline 콘솔에서 GitHub(GitHub 앱 사용) 소스 작업을 추가할 수 있습니다.

주제

- 1단계: (OAuth 앱을 통해) GitHub 작업 교체
- <u>2단계: GitHub에 대한 연결 생성</u>
- <u>3단계: GitHub 소스 작업 저장</u>

1단계: (OAuth 앱을 통해) GitHub 작업 교체

파이프라인 편집 페이지를 사용하여 (OAuth 앱을 통해) GitHub 작업을 (GitHub 앱을 통해) GitHub 작 업으로 바꿉니다.

(OAuth 앱을 통해) GitHub 작업을 교체하려면

- 1. CodePipeline 콘솔에 로그인합니다.
- 파이프라인을 선택한 다음 편집을 선택합니다. 소스 단계에서 단계 편집을 선택합니다. 작업 업데 이트를 권장하는 메시지가 표시됩니다.
- 3. 작업 공급자에서 GitHub(GitHub 앱을 통해)를 선택합니다.
- 4. 다음 중 하나를 수행합니다.

- 연결에서 공급자와의 연결을 아직 생성하지 않은 경우 GitHub에 연결을 선택합니다. 2단계: GitHub에 대한 연결 생성으로 이동합니다.
- 연결에서 공급자와의 연결을 이미 생성한 경우 연결을 선택합니다. 3단계: 연결에 대한 소스 작 업 저장으로 이동합니다.

2단계: GitHub에 대한 연결 생성

연결을 생성하도록 선택하면 GitHub에 연결 페이지가 표시됩니다.

GitHub에 대한 연결을 생성하려면

1. GitHub 연결 설정 아래의 연결 이름에 연결 이름이 표시됩니다.

GitHub 앱에서 앱 설치를 선택하거나 새 앱 설치를 선택하여 앱을 새로 만듭니다.

Note

특정 공급자에 대한 모든 연결에 대해 하나의 앱을 설치합니다. GitHub 앱을 이미 설치한 경우 선택한 다음 이 단계를 건너뜁니다.

- 2. GitHub의 승인 페이지가 표시되면 자격 증명으로 로그인한 다음 계속하도록 선택합니다.
- 앱 설치 페이지에서 AWS CodeStar 앱이 GitHub 계정에 연결을 시도하고 있다는 메시지가 표시됩니다.

Note

각 GitHub 계정마다 앱을 한 번만 설치합니다. 이전에 앱을 설치한 경우 [구성(Configure)] 을 선택하여 앱 설치의 수정 페이지로 이동하거나 뒤로 버튼을 사용하여 콘솔로 돌아갈 수 있습니다.

- 4. AWS CodeStar설치 페이지에서 설치를 선택합니다.
- 5. GitHub에 연결 페이지에서 새 설치의 연결 ID가 표시됩니다. 연결을 선택합니다.

3단계: GitHub 소스 작업 저장

작업 편집 페이지에서 업데이트를 완료하여 새 소스 작업을 저장합니다.

GitHub 소스 작업을 저장하려면

리포지토리에서 타사 리포지토리의 이름을 입력합니다. 브랜치에서, 파이프라인에서 소스 변경
 사항을 감지할 브랜치를 입력합니다.

Note

리포지토리에 다음 예와 같이 owner-name/repository-name을 입력합니다.

my-account/my-repository

- 2. 출력 아티팩트 형식에서 아티팩트의 형식을 선택합니다.
 - 기본 방법을 사용하여 GitHub 작업의 출력 아티팩트를 저장하려면 CodePipeline 기본 방법을 선택합니다. 그러면 GitHub 리포지토리의 파일에 액세스하여 파이프라인 아티팩트 스토어에 ZIP 파일로 아티팩트가 저장됩니다.
 - 다운스트림 작업이 Git 명령을 직접 수행할 수 있도록 리포지토리에 대한 URL 참조가 포함된 JSON 파일을 저장하려면 Full clone(전체 복제)을 선택합니다. 이 옵션은 CodeBuild 다운스트림 작업에서만 사용할 수 있습니다.

이 옵션을 선택하면 <u>Bitbucket, GitHub, GitHub Enterprise Server 또는 GitLab.com 연결을 위한</u> <u>CodeBuild GitClone 권한 추가</u>과 같이 CodeBuild 프로젝트 서비스 역할에 대한 권한을 업데이 트해야 합니다. 전체 복제 옵션을 사용하는 방법을 보여주는 자습서는 <u>자습서: GitHub 파이프라</u> <u>인 소스와 함께 전체 복제 사용</u>을 참조하세요.

- 3. 출력 아티팩트에서 이 작업에 대한 출력 아티팩트의 이름을 유지할 수 있습니다(예: SourceArtifact). 완료를 선택하여 작업 편집 페이지를 닫습니다.
- 완료를 선택하여 단계 편집 페이지를 닫습니다. 저장을 선택하여 파이프라인 편집 페이지를 닫습 니다.

in AWS CodePipeline 할당량

CodePipeline에는 계정이 각 AWS 리전에서 보유할 수 있는 파이프라인, 단계, 작업 및 웹후크 수에 AWS 대한 할당량이 있습니다.

이러한 할당량은 리전당 적용되며 이 할당량을 늘릴 수 있습니다. 할당량 증대 요청을 처리하려면 최대 2주가 걸릴 수 있습니다.

리소스	Default
디소스 작업 만료까지 남은 기간 (구성 가능한 제한 시간입니다. 구성할 수 없는 제한 시간은 다음 표를 참조하세요.)	Default AWS CloudFormation 배포 작업: 3일 CodeDeploy 및 CodeDeploy ECS(파란색/초 록색) 배포 작업: 5일 AWS Lambda 호출 작업: 24시간

리소스

Default

Amazon S3 배포 작업: 90분

Note

대용량 ZIP 파일을 배포하는 동안 S3로의 업로드 시간이 초과되면 시 간 초과 오류와 함께 작업이 실패합 니다. ZIP 파일을 작은 파일로 분할 해 보세요.

수동 승인 작업 계정 수준 기본 제한 시간: 7 일

Note

수동 승인 작업의 기본 제한 시간은 파이프라인의 특정 작업에 대해 재 정의할 수 있으며, 최소 5분에서 최 대 86,400분(60일)까지 구성할 수 있 습니다. 자세한 내용은 CodePipeline API 참조의 <u>ActionDeclaration</u>을 참 조하세요. 이 제한 시간을 구성하면 작업에 적 용됩니다. 그렇지 않으면 계정 수준 기본값이 사용됩니다.

기타 모든 작업: 1시간

Note

Amazon ECS 배포 작업 제한 시간은 최대 1시간(기본 제한 시간)으로 구 성할 수 있습니다.

리소스	Default
AWS 계정의 리전당 최대 총 파이프라인 수	1000
	Solution Note 폴링 또는 이벤트 기반 변경 감지에 대해 구성된 파이프라인은 이 할당량 에 포함됩니다.
AWS 리전당 소스 변경에 대한 폴링으로 설정된 최 대 파이프라인 수	300
	Solution Note 이 할당량은 고정되어 있으며 변경할 수 없습니다. 폴링 파이프라인 한도에 도달하면 이벤트 기반 변경 감지를 사용하는 추가 파이프라인을 구성할 수 있습니다. 자세한 내용은 소스 작업 및 변경 감지 방법 단원을 참조하십시오. ¹
AWS 계정의 리전당 최대 웹후크 수	300
AWS 계정의 리전당 사용자 지정 작업 수	50

¹소스 공급자에 따라 다음 지침에 따라 이벤트 기반 변경 감지를 사용하도록 폴링 파이프라인을 업 데이트합니다.

- CodeCommit 소스 작업을 업데이트하려면 <u>폴링 파이프라인 마이그레이션(CodeCommit 또는</u> Amazon S3 소스)(콘솔) 단원을 참조하세요.
- Amazon S3 소스 작업을 업데이트하려면 <u>폴링 파이프라인 마이그레이션(CodeCommit 또는</u> <u>Amazon S3 소스)(콘솔)</u> 단원을 참조하세요.
- GitHub 소스 작업을 업데이트하려면 <u>폴링 파이프라인을 Webhook로 마이그레이션(GitHub(OAut</u> <u>h 앱 사용) 소스 작업)(콘솔)</u> 단원을 참조하십시오.

의 다음 할당량은 리전 가용성, 이름 지정 제약 조건 및 허용되는 아티팩트 크기에 AWS CodePipeline 적용됩니다. 이러한 할당량은 고정되어 있으므로 변경할 수 없습니다.

각 리전의 CodePipeline 서비스 엔드포인트 목록은AWS 일반 참조에서 <u>AWS CodePipeline 엔드포인</u> 트 및 할당량을 참조하세요.

구조적 요구 사항에 대한 자세한 내용은 CodePipeline 파이프라인 구조 참조 단원을 참조하십시오.

AWS 파이프라인을 생성할 수 있는 리전	미국 동부(오하이오)
	미국 동부(버지니아 북부)
	미국 서부(캘리포니아 북부)
	미국 서부(오레곤)
	캐나다(중부)
	유럽(프랑크푸르트)
	유럽(취리히)*
	이스라엘(텔아비브)
	유럽(아일랜드)
	유럽(런던)
	유럽(밀라노)*
	유럽(파리)
	유럽(스페인)
	유럽(스톡홀름)
	아프리카(케이프타운)*
	아시아 태평양(홍콩)*
	아시아 태평양(하이데라바드)
	아시아 태평양(뭄바이)

아시아 태평양(도쿄)

아시아 태평양(서울)

아시아 태평양(오사카)

아시아 태평양(싱가포르)

아시아 태평양(시드니)

아시아 태평양(자카르타)

아시아 태평양(멜버른)

남아메리카(상파울루)

중동(바레인)*

중동(UAE)

AWS GovCloud(미국 서부)

AWS GovCloud(미국 동부)

작업 이름에 허용되는 문자

작업 이름은 100자를 초과할 수 없습니다. 허용 되는 문자는 다음과 같습니다.

a부터 z까지의 소문자

A부터 Z까지의 대문자

0부터 9까지의 숫자

특수문자 .(마침표), @(골뱅이 기호), -(빼기 기 호), _(밑줄).

그 외에 공백을 비롯한 다른 문자는 허용되지 않 습니다.

작업 유형에 허용되는 문자	작업 유형 이름은 25자를 초과할 수 없습니다. 허용되는 문자는 다음과 같습니다.
	a부터 z까지의 소문자
	A부터 Z까지의 대문자
	0부터 9까지의 숫자
	특수문자 .(마침표), @(골뱅이 기호), -(빼기 기 호), _(밑줄).
	그 외에 공백을 비롯한 다른 문자는 허용되지 않 습니다.
아티팩트 이름에 허용되는 문자	아티팩트 이름은 100자를 초과할 수 없습니다. 허용되는 문자는 다음과 같습니다.
	a부터 z까지의 소문자
	A부터 Z까지의 대문자
	0부터 9까지의 숫자
	특수문자 -(마이너스 기호), _(밑줄).
	그 외에 공백을 비롯한 다른 문자는 허용되지 않 습니다.

파트너 작업 이름에 허용되는 문자	파트너 작업 이름은 CodePipeline의 다른 작업 이름과 동일한 명명 규정과 제한을 준수해야 합 니다. 특히 100자를 초과할 수 없습니다. 허용되 는 문자는 다음과 같습니다.
	a부터 z까지의 소문자
	A부터 Z까지의 대문자
	0부터 9까지의 숫자
	특수문자 .(마침표), @(골뱅이 기호), -(빼기 기 호), _(밑줄).
	그 외에 공백을 비롯한 다른 문자는 허용되지 않 습니다.
파이프라인 이름에 허용되는 문자	파이프라인 이름은 100자를 초과할 수 없습니 다. 허용되는 문자는 다음과 같습니다.
	a부터 z까지의 소문자
	A부터 Z까지의 대문자
	0부터 9까지의 숫자
	특수문자 .(마침표), @(골뱅이 기호), -(빼기 기 호), _(밑줄).
	그 외에 공백을 비롯한 다른 문자는 허용되지 않 습니다.

단계 이름에 허용되는 문자	단계 이름은 100자를 초과할 수 없습니다. 허용 되는 문자는 다음과 같습니다.
	a부터 z까지의 소문자
	A부터 Z까지의 대문자
	0부터 9까지의 숫자
	특수문자 .(마침표), @(골뱅이 기호), -(빼기 기 호), _(밑줄).
	그 외에 공백을 비롯한 다른 문자는 허용되지 않 습니다.
작업 만료까지 남은 기간	CodeBuild 빌드 작업: 36시간
	테스트 작업: 8시간
	사용자 지정 작업: 24시간
	Step Functions 호출 작업: 7일
	명령 작업에 대한 빌드 제한 시간: 55분
작업 구성 키의 최대 길이(예를 들어 CodeBuild 구성 키는 ProjectName ,PrimarySource 및 EnvironmentVariables 임)	50자
작업 구성 값의 최대 길이(예를 들어 CodeCommit 작업 구성의 RepositoryName 구성 값은 1000자 미만이어야 함:	1000자
"RepositoryName": "my-repo-name- less-than-1000-characters")	
파이프라인당 최대 작업 수	1000
파이프라인당 최대 동시 파이프라인 실행 수 (QUEUED PARALLEL 모드)	50

PARALLEL 모드 파이프라인 실행당 최대 동시 작업 실행 수	5
Amazon S3 객체의 최대 파일 수	100,000건
단계의 최대 병렬 작업 수	100
단계의 최대 순차 작업 수	100
소스 단계의 최대 아티팩트 크기	Amazon S3 버킷에 저장되는 아티팩트: 7GB
	CodeCommit 또는 GitHub 리포지토리에 저장되 는 아티팩트: 1GB
	예외: AWS Elastic Beanstalk 를 사용하여 애플 리케이션을 배포하는 경우 최대 아티팩트 크기 는 항상 512MB입니다.
	예외: AWS CloudFormation 를 사용하여 애플리 케이션을 배포하는 경우 최대 아티팩트 크기는 항상 256MB입니다.
	예외: 애플리케이션 배포에 CodeDep1o yToECS 작업 사용 시 아티팩트의 최대 크기는 항상 3MB입니다.
Amazon ECS 컨테이너와 이미지를 배포하는 파 이프라인에 사용되는 이미지 정의 JSON 파일의 최대 크기	100KB
AWS CloudFormation 작업에 대한 입력 아티팩 트의 최대 크기	256MB
CodeDeployToECS 작업용 입력 아티팩트의 최대 크기	3MB
Step Functions 작업용 입력 아티팩트의 최 대 크기	Step Functions 작업은 Lambda에서 실행되므로 Lambda 함수에 대한 아티팩트 크기 할당량과 동일한 아티팩트 크기 할당량이 있습니다. 자세 한 내용은 Lambda 개발자 안내서에서 <u>Lambda</u> <u>할당량</u> 을 참조하세요.
--	--
Parameter0verrides 속성에 저장할 수 있 는 JSON 객체의 최대 크기	를 공급자 AWS CloudFormation 로 사용하는 CodePipeline 배포 작업의 경우 Parameter Overrides 속성은 AWS CloudFormation 템플릿 구성 파일의 값을 지정하는 JSON 객 체를 저장하는 데 사용됩니다. Parameter Overrides 속성에 저장할 수 있는 JSON 객 체의 최대 크기 제한은 1킬로바이트입니다.
단계의 작업 수	최소 1, 최대 50
각 작업에 허용되는 아티팩트 수	각 작업에 허용되는 입력 및 출력 아티팩트의 수 는 <u>각 작업 유형에 대한 유효한 입력 및 출력 아</u> <u>티팩트</u> 단원을 참조하십시오.
파이프라인 실행 내역 정보가 보관되는 개월 수	12
파이프라인의 단계 수	최소 2, 최대 50
파이프라인 태그	태그는 대/소문자를 구분합니다. 리소스당 최대 50개입니다.

파이프라인 태그 키 이름	유니코드 문자, 숫자, 공백 및 UTF-8 형식의 허 용되는 문자 조합(1 - 128 문자) 사용할 수 있는 문자: + - = : / @	
	태그 키 이름은 고유해야 하며 각 키는 하나의 값만 가질 수 있습니다. 태그는 다음을 허용하지 않습니다.	
	 다음으로 시작합니다 AWS. 공백으로만 이루어짐 공백으로 끝남 이모티콘 또는 다음 문자를 포함: ?^*[\~!# \$%&*()>< "' 	
파이프라인 태그 값	유니코드 문자, 숫자, 공백 및 UTF-8 형식의 허 용되는 문자 조합(1 - 256 문자) 사용할 수 있는 문자: + - = : / @	
	키는 하나의 값만 가질 수 있지만 많은 키가 동 일한 값을 가질 수 있습니다. 태그는 다음을 허 용하지 않습니다.	
	• 다음으로 시작합니다 AWS. • 공백으로만 이루어짐 • 공백으로 끝남	
	 이모티콘 또는 다음 문자를 포함: ? ^ * [\~! # \$ % & *() > < " ' 	

트리거

push 및 pull request 구성에서 파이프라인 정의에는 최대 50개의 트리거가 있습니다.

푸시 트리거 및 풀 요청 트리거당 최대 3개의 필 터가 있습니다.

Note

동일한 이벤트 유형 배열에서 필터 중복 은 허용되지 않습니다.

각 이벤트 유형(푸시, 풀 요청)에 대해 최대 8개 의 포함 및 8개의 제외 패턴, 브랜치 및 파일 경 로를 추가할 수 있습니다.

패턴 값에 허용되는 문자에는 모든 문자 유형이 포함됩니다.

포함 및 제외 패턴의 경우, 최대 길이는 255자입 니다.

태그 이름의 경우, 최대 길이는 255자입니다.

triggers 배열의 최대 크기는 200KB를 초과해 서는 안 됩니다.

트리거 필터

이름 고유성

파일 경로:

- 패턴 수: 최대 8개의 포함 및 8개의 제외 패턴 을 추가할 수 있습니다.
- 패턴 크기: 각 포함 또는 제외 패턴의 크기는 최대 255자일 수 있습니다.

브랜치:

- 패턴 수: 최대 8개의 포함 및 8개의 제외 패턴 을 추가할 수 있습니다.
- 패턴 크기: 각 포함 또는 제외 패턴의 크기는 최대 255자일 수 있습니다.

풀 요청:

브랜치:

- 패턴 수: 최대 8개의 포함 및 8개의 제외 패턴 을 추가할 수 있습니다.
- 패턴 크기: 각 포함 또는 제외 패턴의 크기는 최대 255자일 수 있습니다.

단일 AWS 계정 내에서 AWS 리전에서 생성하 는 각 파이프라인에는 고유한 이름이 있어야 합 니다. 다른 AWS 리전의 파이프라인에서 이름을 재사용할 수 있습니다.

파이프라인 내의 단계 이름은 고유해야 합니다.

단계 내의 작업 이름은 고유해야 합니다.

출력 변수 및 네임스페이스 할당량

특정 작업에 대해 결합된 모든 출력 변수의 최대 크기 제한은 122,880바이트입니다.

특정 작업에 대해 해결된 전체 작업 구성의 최대 크기 제한은 100KB입니다.

출력 변수 이름은 대소문자를 구분합니다.

네임스페이스는 대소문자를 구분합니다.

허용되는 문자는 다음과 같습니다.

- a부터 z까지의 소문자
- A부터 Z까지의 대문자
- 0부터 9까지의 숫자
- 특수문자 ^(캐럿), @(골뱅이 기호), -(빼기 기 호), _(밑줄), [(왼쪽 대괄호),](오른쪽 대괄호), *(별표), \$(달러 기호).

그 외에 공백을 비롯한 다른 문자는 허용되지 않습니다.

파이프라인 수준의 변수 할당량	파이프라인당 최대 50개의 파이프라인 수준 변 수가 있습니다.
	파이프라인 수준의 변수 이름은 다음과 같아야 합니다.
	• 최대 128자 길이
	• a부터 z까지의 소문자
	• A부터 Z까지의 대문자
	• 0부터 9까지의 숫자
	• 특수 문자 @\]+
	그 외에 공백을 비롯한 다른 문자는 허용되지 않습니다.
	변수 값의 최대 길이는 1000자입니다.
	변수 값에는 모든 문자를 사용할 수 있습니다.
	변수 설명의 최대 길이는 200자입니다.

* 이 리전을 사용하려면 먼저 활성화해야 합니다.

부록 A: GitHub(OAuth 앱 사용) 소스 작업

이 부록에서는 CodePipeline에서 GitHub 작업의 (OAuth 앱을 통해)에 대한 정보를 제공합니다.

Note

GitHub(OAuth 앱 사용) 작업을 사용하지 않는 것이 좋지만 GitHub(OAuth 앱 사용) 작업이 있 는 기존 파이프라인은 아무런 영향 없이 계속 작동합니다. GitHub(OAuth 앱 사용) 작업이 있 는 파이프라인의 경우 CodePipeline은 OAuth 기반 토큰을 사용하여 GitHub 리포지토리에 연 결합니다. 반대로 GitHub 작업(GitHub 앱을 통해)은 연결 리소스를 사용하여 AWS 리소스를 GitHub 리포지토리에 연결합니다. 연결 리소스는 앱 기반 토큰을 사용하여 연결합니다. 연결 을 사용하는 권장 GitHub 작업으로 파이프라인을 업데이트하는 방법에 대한 자세한 내용은 <u>GitHub(OAuth 앱 사용) 소스 작업을 GitHub(GitHub 앱 사용) 소스 작업으로 업데이트</u>을 참조 하세요. 앱 기반 GitHub 액세스와 대조되는 OAuth 기반 GitHub 액세스에 대한 자세한 내용 은 <u>https://docs.github.com/en/developers/apps/differences-between-github-apps-and-oauth-apps을 참조하세요.</u>

GitHub와 통합하기 위해 CodePipeline은 파이프라인용 GitHub OAuth 애플리케이션을 사용합니다. CodePipeline은 웹후크를 사용하여 GitHub(OAuth 앱 사용) 소스 작업을 통해 파이프라인에 대한 변경 감지를 관리합니다.

Note

에서 GitHub(GitHub 앱을 통해) 소스 작업을 구성할 때 GitHub 토큰 정보를 포함 하거나 웹후크 리소스를 추가하지 AWS CloudFormation않습니다. 사용 설명서의 <u>AWS::CodeStarConnections::Connection</u>에 표시된 대로 연결 리소스를 구성합니다. AWS CloudFormation

이 참조에는 GitHub(OAuth 앱 사용) 작업에 대한 다음 섹션이 포함되어 있습니다.

- GitHub(OAuth 앱 사용) 소스 작업 및 웹후크를 파이프라인에 추가하는 방법에 대한 자세한 내용은 섹션을 참조하세요<u>GitHub(OAuth 앱 사용) 소스 작업 추가</u>.
- GitHub(OAuth 앱 사용) 소스 작업의 구성 파라미터 및 예제 YAML/JSON 코드 조각에 대한 자세한 내용은 섹션을 참조하세요GitHub(OAuth 앱 사용) 소스 작업 참조.

▲ Important

CodePipeline 웹후크를 생성할 때 자체 자격 증명을 사용하거나 여러 웹후크에서 동일한 보 안 암호 토큰을 재사용하지 마세요. 최적의 보안을 위해 생성하는 각 웹후크에 대해 고유한 보 안 암호 토큰을 생성합니다. 보안 암호 토큰은 사용자가 제공하는 임의의 문자열로, GitHub는 CodePipeline으로 전송된 웹후크 페이로드를 계산하고 서명하는 데 이를 사용하여 웹후크 페 이로드의 무결성과 신뢰성을 보호합니다. 자체 자격 증명을 사용하거나 여러 웹후크에서 동일 한 토큰을 재사용하면 보안 취약성이 발생할 수 있습니다.

Note

보안 암호 토큰이 제공된 경우 응답에서 해당 토큰이 수정됩니다.

주제

- GitHub(OAuth 앱 사용) 소스 작업 추가
- <u>GitHub(OAuth 앱 사용) 소스 작업 참조</u>

GitHub(OAuth 앱 사용) 소스 작업 추가

다음을 통해 GitHub(OAuth 앱을 통해) 소스 작업을 CodePipeline에 추가합니다.

- CodePipeline 콘솔 파이프라인 생성 마법사(<u>사용자 지정 파이프라인 생성(콘솔)</u>) 또는 작업 편집 페이지를 사용하여 GitHub 공급자 옵션을 선택합니다. 콘솔은 소스가 변경될 때 파이프라인을 시작하는 웹후크를 생성합니다.
- CLI를 사용하여 GitHub 작업에 대한 작업 구성을 추가하고 다음과 같이 추가 리소스를 생성합니다.
 - <u>GitHub(OAuth 앱 사용) 소스 작업 참조</u>의 GitHub 예제 작업 구성을 사용하여 <u>파이프라인 생성</u> (CLI)과 같이 작업을 생성합니다.
 - 변경 감지 방법은 기본적으로 소스를 폴링하여 파이프라인을 시작하기 때문에 정기적 확인을 비 활성화하고 변경 감지를 수동으로 생성합니다. 폴링 파이프라인을 GitHub용 웹후크(OAuth 앱 사 용) 작업으로 마이그레이션합니다.

GitHub(OAuth 앱 사용) 소스 작업 참조

Note

GitHub(OAuth 앱 사용) 작업을 사용하지 않는 것이 좋지만 GitHub(OAuth 앱 사용) 작업이 있 는 기존 파이프라인은 아무런 영향 없이 계속 작동합니다. GitHub GitHub(OAuth 앱 사용) 소스 작업이 있는 파이프라인의 경우 CodePipeline은 OAuth 기반 토큰을 사용하여 GitHub 리포지 토리에 연결합니다. 반대로 새 GitHub 작업(GitHub 앱을 통해)은 연결 리소스를 사용하여 AWS 리소스를 GitHub 리포지토리에 연결합니다. 연결 리소스는 앱 기반 토큰을 사용하여 연결합니 다. 연결을 사용하는 권장 GitHub 작업으로 파이프라인을 업데이트하는 방법에 대한 자세한 내 용은 <u>GitHub(OAuth 앱 사용) 소스 작업을 GitHub(GitHub 앱 사용) 소스 작업으로 업데이트</u>을 참조하세요.

구성된 GitHub 리포지토리와 분기에서 새 커밋이 이루어지면 파이프라인을 트리거합니다.

GitHub와 통합하기 위해 CodePipeline은 파이프라인에서 OAuth 애플리케이션이나 개인 액세스 토큰 을 사용합니다. 콘솔에서 파이프라인을 생성하거나 편집하는 경우, CodePipeline은 리포지토리에서 변 경이 발생할 때 파이프라인을 시작하는 GitHub 웹후크를 생성합니다.

GitHub 작업을 통해 파이프라인을 연결하기 전에 미리 GitHub 계정과 리포지토리를 생성해야 합니다.

리포지토리에 대한 CodePipeline의 액세스 권한을 제한하고 싶으면 GitHub 계정을 생성하고 CodePipeline에 통합하고 싶은 리포지토리에만 액세스할 수 있도록 계정에 권한을 부여합니다. 파이프 라인의 소스 단계에서 GitHub 리포지토리를 사용하도록 CodePipeline을 구성할 때 이 계정을 사용합 니다.

자세한 내용은 GitHub 웹 사이트의 GitHub 개발자 설명서를 참조하십시오.

주제

- 작업유형
- 구성 파라미터
- 입력 아티팩트
- 출력 아티팩트
- <u>출력 변수</u>
- 작업 선언(GitHub 예제)
- <u>GitHub에 연결(OAuth)</u>

• <u>다음 사항도 참조하세요.</u>

작업 유형

- 범주: Source
- 소유자: ThirdParty
- 공급자:GitHub
- 버전:1

구성 파라미터

소유자

필수 여부: 예

GitHub 리포지토리를 소유한 GitHub 사용자 또는 조직의 이름입니다.

Repo

필수 여부: 예

소스 변경 사항을 감지할 리포지토리의 이름입니다.

브랜치

필수 여부: 예

소스 변경 사항을 감지할 분기의 이름입니다.

OauthToken

필수 여부: 예

CodePipeline이 GitHub 리포지토리에서 작업을 수행할 수 있게 해주는 GitHub 인증 토큰을 나타냅니다. 항목은 항상 네 개의 별표로 표시됩니다. 다음 값 중 하나를 나타냅니다.

- 콘솔을 사용하여 파이프라인을 생성할 때 CodePipeline은 OAuth 토큰을 사용하여 GitHub 연결 을 등록합니다.
- AWS CLI 를 사용하여 파이프라인을 생성할 때이 필드에 GitHub 개인 액세스 토큰을 전달할 수 있습니다. 별표(****) 를 GitHub에서 복사한 개인 액세스 토큰으로 바꿉니다. 작업 구성을 보기 위 해 get-pipeline을 실행하면 이 값에 대해 별표 4개 마스크가 표시됩니다.

 AWS CloudFormation 템플릿을 사용하여 파이프라인을 생성할 때는 먼저 토큰을 보안 암호로 저장해야 합니다 AWS Secrets Manager. 이 필드의 값 을 Secrets Manager에 저장된 보안 암호에 대한 동적 참조로 포함합니다(예: {{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}).

GitHub 범위에 대한 자세한 내용은 GitHub 웹 사이트의 <u>GitHub 개발자 API 참조</u>를 참조하십시오. PollForSourceChanges

필수 여부: 아니요

PollForSourceChanges는 CodePipeline이 GitHub 리포지토리에서 소스 변경 사항을 폴링할지 여부를 제어합니다. 대신 웹후크를 사용하여 소스 변경 사항을 감지하는 것이 좋습니다. 웹후크 구 성에 대한 자세한 내용은 <u>폴링 파이프라인을 웹후크로 마이그레이션(GitHub(OAuth 앱 사용) 소스</u> <u>작업)(CLI)</u> 또는 <u>푸시 이벤트에 대한 파이프라인 업데이트(GitHub(OAuth 앱 사용) 소스 작업)(AWS</u> <u>CloudFormation 템플릿)</u> 단원을 참조하십시오.

A Important

웹후크를 구성하려는 경우 중복된 파이프라인 실행이 발생하지 않도록 PollForSourceChanges을 false로 설정해야 합니다.

이 파라미터에 유효한 값은 다음과 같습니다.

• True: 이 값으로 설정된 경우 CodePipeline은 리포지토리에서 소스 변경 사항을 폴링합니다.

1 Note

PollForSourceChanges를 생략하면 CodePipeline은 기본적으로 소스 변경 사항에 대 해 리포지토리를 폴링합니다. 이러한 동작은 PollForSourceChanges이 true로 설정 된 경우와 똑같습니다.

• False: 이 값으로 설정된 경우 CodePipeline은 리포지토리에서 소스 변경 사항을 폴링하지 않습니다. 소스 변경 사항을 감지하도록 웹후크를 구성하려면 이 설정을 사용합니다.

입력 아티팩트

- 아티팩트 수: 0
- 설명: 이 작업 유형에는 입력 아티팩트가 적용되지 않습니다.

출력 아티팩트

- 아티팩트 수: 1
- 설명: 이 작업의 출력 아티팩트는 파이프라인 실행을 위한 소스 개정으로 지정된 커밋에서 구성된 리 포지토리 및 분기의 내용을 포함하는 ZIP 파일입니다. 리포지토리에서 생성된 아티팩트는 GitHub 작업에 대한 출력 아티팩트입니다. 소스 코드 커밋 ID는 트리거된 파이프라인 실행의 소스 개정으로 CodePipeline에 표시됩니다.

출력 변수

이 작업을 구성하면 파이프라인에서 다운스트림 작업의 작업 구성에서 참조할 수 있는 변수가 생성됩 니다. 이 작업은 작업에 네임스페이스가 없는 경우에도 출력 변수로 볼 수 있는 변수를 생성합니다. 이 러한 변수를 다운스트림 작업 구성에서 사용할 수 있도록 네임스페이스를 사용하여 작업을 구성합니 다.

CodePipeline의 변수에 대한 자세한 내용은 변수 참조 단원을 참조하세요.

CommitId

파이프라인 실행을 트리거한 GitHub 커밋 ID입니다. 커밋 ID는 커밋의 전체 SHA입니다.

CommitMessage

파이프라인 실행을 트리거한 커밋과 연관된 설명 메시지입니다(존재하는 경우).

CommitUrl

파이프라인을 트리거한 커밋의 URL 주소입니다.

RepositoryName

파이프라인을 트리거한 커밋이 만들어진 GitHub 리포지토리의 이름입니다.

BranchName

소스 변경이 이루어진 GitHub 리포지토리의 분기 이름입니다.

AuthorDate

커밋이 작성된 날짜입니다(타임스탬프 형식).

CommitterDate

커밋이 수행된 날짜입니다(타임스탬프 형식).

작업 선언(GitHub 예제)

YAML

```
Name: Source
Actions:
  - InputArtifacts: []
    ActionTypeId:
      Version: '1'
      Owner: ThirdParty
      Category: Source
      Provider: GitHub
    OutputArtifacts:
      - Name: SourceArtifact
    RunOrder: 1
    Configuration:
      Owner: MyGitHubAccountName
      Repo: MyGitHubRepositoryName
      PollForSourceChanges: 'false'
      Branch: main
      OAuthToken: '{{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}'
    Name: ApplicationSource
```

JSON

```
{
    "Name": "Source",
    "Actions": [
        {
            "InputArtifacts": [],
            "ActionTypeId": {
                "Version": "1",
                "Owner": "ThirdParty",
                "Category": "Source",
                "Provider": "GitHub"
            },
            "OutputArtifacts": [
                {
                     "Name": "SourceArtifact"
                }
            ],
            "RunOrder": 1,
            "Configuration": {
```

```
"Owner": "MyGitHubAccountName",
    "Repo": "MyGitHubRepositoryName",
    "PollForSourceChanges": "false",
    "Branch": "main",
    "OAuthToken":
    "QAuthToken":
    "{{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}"
    },
    "Name": "ApplicationSource"
    }
]
```

GitHub에 연결(OAuth)

GitHub 리포지토리를 처음 파이프라인에 추가하면 사용자의 리포지토리에 대한 CodePipeline 액세스 를 승인하라는 메시지가 나타납니다. 토큰에는 다음과 같은 GitHub 범위가 필요합니다.

- repo 범위(퍼블릭 및 프라이빗 리포지토리에서 파이프라인으로 아티팩트를 읽고 가져올 수 있도록 완전히 제어하는 데 사용됨).
- admin:repo_hook 범위(리포지토리 후크를 완전히 제어하는 데 사용됨).

CLI 또는 AWS CloudFormation 템플릿을 사용하는 경우 GitHub에서 이미 생성한 개인 액세스 토큰의 값을 제공해야 합니다.

다음 사항도 참조하세요.

이 작업을 수행할 때 참조할 수 있는 관련 리소스는 다음과 같습니다.

- <u>AWS CloudFormation 사용 설명서 AWS::CodePipeline::Webhook</u>에 대한 리소스 참조 여기에는의 리소스에 대한 필드 정의, 예제 및 코드 조각이 포함됩니다 AWS CloudFormation.
- <u>AWS CloudFormation 사용 설명서 AWS::CodeStar::GitHubRepository</u>를 위한 리소스 참조 여기에 는 AWS CloudFormation의 리소스에 대한 필드 정의, 예제 및 코드 조각이 포함됩니다.
- <u>자습서:를 사용하여 Android 앱을 빌드하고 테스트하는 파이프라인 생성 AWS Device Farm</u> 이 자 습서에서는 GitHub 소스를 사용하여 파이프라인을 생성하기 위한 샘플 빌드 사양 파일과 샘플 애플 리케이션을 제공합니다. CodeBuild 및 AWS Device Farm를 사용해 Android 앱을 빌드 및 테스트합 니다.

AWS CodePipeline 사용 설명서 문서 기록

다음 표에서는 CodePipeline 사용 설명서의 각 릴리스에서 변경된 중요 사항에 대해 설명합니다. 이 설 명서에 대한 업데이트 알림을 받으려면 RSS 피드를 구독하면 됩니다.

- API 버전: 2015-07-09
- 최종 설명서 업데이트: 2025년 5월 8일

변경 사항	설명	날짜
<u>컴퓨팅 유형, 환경 유형 및 환경</u> <u>변수 유형에 대한 명령 작업에</u> 필드 추가	명령 작업에 대한 빌드 환경 및 컴퓨팅 유형을 지정할 수 있는 새 필드에 대한 정보가 추가되 었습니다. 명령 작업에 대한 환 경 변수의 새 유형 필드도 추 가되었습니다. <u>CodePipeline</u> <u>Comands 작업의 작업 참조 페</u> 이지를 참조하세요.	2025년 5월 8일
<u>기본 서비스 역할 정책을 문서</u> <u>화하는 새 주제</u>	최소 서비스 역할 정책에 대한 정보를 추가하고 CodePipel ine의 각 작업에 대한 추가 서 비스 역할 권한에 대한 링크 테이블을 업데이트했습니다. CodePipeline 서비스 역할 정 책에서 새 규칙 참조 페이지를 참조하세요.	2025년 3월 26일
<u>새 CodePipeline 호출 작업</u>	새 CodePipeline 호출 작업 에 대한 정보가 추가되었습니 다. <u>CodePipeline 호출 작업 참</u> <u>조의 작업 참조 페이지를 참조</u> <u>하세요</u> .	2025년 3월 14일
<u>새CodeBuild 규칙</u>	단계 조건에 대한 CodeBuild 규칙으로 빌드 프로젝트를 실	2025년 3월 14일

	행하는 데 사용할 수 있는 새 규 칙에 대한 정보가 추가되었습 니다. <u>CodeBuild 규칙 참조의</u> <u>새 규칙 참조</u> 페이지를 참조하 세요.	
<u>에서 연결 공유 AWS</u> <u>CodeConnections</u>	를 사용하는 연결이 있는 파이프라인의 경우 AWS CodeConnections, 간에 공유 하도록 구성된 연결을 사용할 수 있습니다 AWS 계정. 에서 공유 연결을 구성할 수 있습 니다 AWS Resource Access Manager. 자세한 내용은 <u>다른</u> <u>와 공유된 연결 사용을 AWS 계</u> <u>정</u> 참조하세요.	2025년 3월 6일
<u>소스 개정에 대한 EventBridge</u> <u>입력 변환 항목</u>	CodeCommit, Amazon ECR 및 Amazon S3 소스가 있는 파이프라인의 경우 소스 개정 에 EventBridge 입력 변환 항 목을 사용할 수 있습니다. 여 기서 revisionValue 는 객 체키(S3), 커밋(CodeCommit) 또는 이미지 ID(ECR)의 소스 이벤트 변수에서 파생됩니다. Amazon ECR 소스 작업 및 EventBridge 리소스, 이벤트에 대해 활성화된 소스를 사용하 여 Amazon S3 소스 작업에 연 결, CodeCommit 소스 작업 및 EventBridge를 참조하세요.	2025년 3월 3일

<u>AWS CodeBuild 작업에 대한</u> <u>빌드 사양 재정의</u>	명령을 직접 입력하는 대신 CodeBuild 작업의 빌드 사양을 재정의하도록 선택할 수 있습 니다. <u>AWS CodeBuild 빌드 및</u> 테스트 작업 참조의 작업 참조 페이지를 참조하세요. <u>파이프</u> 라인, 단계 및 작업 생성을 참조 하세요.	2025년 3월 3일
<u>새 EC2 배포 작업</u>	새 EC2 배포 작업에 대한 정보 가 추가되었습니다. <u>Amazon</u> <u>EC2 작업 참조의 작업 참조</u> 페 이지를 참조하세요. 자습서는 <u>자습서: CodePipeline을 사용</u> <u>하여 EC2 인스턴스에 배포</u> 를 참조하세요.	2025년 2월 21일
<u>새 EKS 배포 작업</u>	새 EKS 배포 작업에 대한 정 보가 추가되었습니다. <u>EKS 배</u> 포 작업의 작업 참조 페이지 를 참조하세요. 자습서는 <u>자습</u> <u>서: CodePipeline을 사용하여</u> <u>Amazon EKS에 배포</u> 를 참조하 세요.	2025년 2월 20일
<u>CodePipeline에 대한 새로운</u> <u>CloudWatch 지표 및 차원</u>	CloudWatch 지표에 파이프라 인에 대한 지표가 추가되었습 니다. <u>CodePipeline CloudWatc</u> <u>h 지표</u> 를 참조하세요.	2025년 2월 13일
<u>새 Commands 규칙</u>	단계 조건에 대한 Commands 규칙으로 셸 명령을 실행하는 데 사용할 수 있는 새 규칙에 대 한 정보가 추가되었습니다. <u>명</u> 령 규칙 참조의 새 규칙 참조 페 이지를 참조하세요.	2024년 12월 17일

<u>트리거에 대한 확장된 예제 및</u> <u>참조 정보</u>	공급자별 트리거에 대한 풀 요 청 이벤트에 <u>공급자별 풀 요청</u> 이벤트 필터에 대한 설명이 추 가되었습니다. <u>공급자별 트리</u> 거에 대한 풀 요청 이벤트의 푸 시 이벤트 및 풀 요청 이벤트 포 함 및 제외에 대한 자세한 정보 가 포함된 트리거에 대한 확장 된 예제가 추가되었습니다. <u>트</u> 리거에서 포함 및 제외에 대한	2024년 12월 17일
<u>작업 카탈로그의 새 작업</u>	추가 JSON 참조 정보가 추가 되었습니다. 이제 ECRBuildAndPublish 및 InspectorScan 작업 을 사용할 수 있습니다. 자세한 내용은 <u>ECRBuildAndPublish</u> 및 <u>InspectorScan</u> 작업 참조 페 이지를 참조하세요.	2024년 11월 22일
<u>실패 시 스테이지 재시도를 위</u> <u>한 새 자동 구성</u>	실패한 스테이지 또는 스테이 지에서 실패한 작업을 자동으 로 재시도하도록 스테이지를 구성할 수 있습니다. 자세한 내 용은 <u>실패 시 자동 재시도를 위</u> <u>한 스테이지 구성</u> 을 참조하세 요.	2024년 10월 15일

<u>입력 조건에 대한 새 Skip 결</u> 과	입력 조건에 사용할 수 있 는 Skip 결과에 대한 정보 가 추가되었습니다. 이 구 성에 VariableCheck 및 LambdaInvoke 규칙을 사용 할 수 있습니다. 건너뛰기 결과 가 있는 입력 조건 생성의 단계 를 참조하세요. 건너뛰기 결과 가 있는 조건에 대한 고려 사항 목록은 <u>스테이지 조건에 대해</u> 구성된 결과 고려 사항을 참조 하세요.	2024년 10월 15일
<u>정적 템플릿에서 파이프라인을</u> 생성하는 새로운 콘솔 단계	CodePipeline 콘솔에서 새 파 이프라인 생성 마법사를 사용 하여 여러 정적 템플릿 중에 서 선택하여 AWS CloudForm ation에서 파이프라인 리소스를 생성할 수 있습니다. 자세한 내 용은 <u>정적 템플릿에서 파이프</u> <u>라인 생성</u> 을 참조하세요.	2024년 10월 9일
<u>새로운 Commands 작업</u>	파이프라인의 작업으로 쉘 명 령을 실행하는 데 사용할 수 있 는 새 Commands 작업에 대한 정보가 추가되었습니다. 명령 작업의 새 작업 참조 페이지와 CodePipeline 서비스 역할에 권한 추가의 서비스 역할 권한 을 참조하세요. 자습서는 <u>자습</u> 서: 컴퓨팅을 사용하여 명령을 실행하는 파이프라인 생성을 참조하세요.	2024년 10월 3일

<u>조건에 대한 새 VariableC</u> <u>heck 규칙</u>	스테이지 조건의 VariableC heck 규칙에 대한 정보가 추가되었습니다. <u>VariableC</u> <u>heck</u> 의 새 규칙 참조 페이지를 참조하세요. 자습서는 <u>자습서:</u> <u>파이프라인에 대한 변수 확인</u> <u>규칙을 입력 조건으로 생성</u> 을 참조하세요.	2024년 9월 27일
<u>GitHub 연결 업데이트</u>	GitHub 사용자 액세스 토큰을 GitHub(GitHub V2 작업)에 연 결하여 사용하는 방법에 대한 정보가 추가되었습니다. 사용 자 액세스 토큰은 CodeBuild 프로젝트와 함께 사용됩니다. GitHub 연결 및 <u>자습서: GitHub</u> 파이프라인 소스와 함께 전체 복제 사용을 참조하세요.	2024년 9월 16일
<u>파이프라인 JSON 참조 및 가</u> 이드 목차를 재구성하기 위한 업데이트	참조 및 작업 섹션의 사용성을 향상시키기 위해 일부 섹션 제 목 변경을 포함하여 가이드가 재구성되었습니다.	2024년 8월 16일
<u>PutWebhook 및 ListWebho</u> oks 작업에 대한 응답의 보안 암호 토큰 필드 업데이트	PutWebhook 및 ListWebho oks 작업에 대한 보안 암호 토큰 필드가 업데이트되었습니 다. 보안 암호 토큰이 제공된 경 우 응답에서 해당 토큰이 수정 됩니다. <u>부록 A: GitHub 버전 1</u> 소스 작업에 추가된 참고 사항 을 참조하세요. CodePipeline API 가이드의 관련 업데이트 는 <u>PutWebhook 및 ListWebho</u> oks를 참조하세요.	2024년 8월 6일

<u>스테이지 조건 및 규칙에 대한</u> <u>새 콘텐츠 추가</u>	이제 V2 유형 파이프라인에 대 한 스테이지 조건 및 규칙을 구 성할 수 있습니다. <u>개념, 스테이</u> 지 <u>조건은 어떻게 작동하나요?</u> 및 <u>스테이지에 대한 조건 구</u> 성을 참조하세요. 참조 정보를 제공하는 규칙에 대한 참조 장 이 추가되었습니다. <u>CodePipel</u> <u>ine 규칙 참조</u> 를 참조하세요.	2024년 7월 30일
<u>파이프라인 유형 및 관련 기능</u> <u>에 대한 새로운 참조 정보 추가</u>	V2 유형 파이프라인으로 이동 하는 데 드는 비용을 평가하기 위한 새로운 분석 스크립트를 사용할 수 있습니다. <u>나에게 적</u> <u>합한 파이프라인 유형은 무엇</u> 인가요?를 참조하세요. 모든 CodePipeline 서비스 설명서 에 기능별 링크를 제공하는 참 조 테이블이 추가되었습니다. <u>CodePipeline 기능 참조</u> 를 참 조하세요.	2024년 7월 11일
<u>소스 재정의에 새 옵션을 추가</u> <u>하기 위해 S3 소스 작업 업데이</u> 트	S3_OBJECT_KEY 라는 소스 재정의에 대한 새 옵션을 S3 소 스 작업에 사용할 수 있습니다. S3 소스 작업에 새 AllowOver rideForS30bjectKey 파라미터가 추가되었습니다. <u>Amazon S3 소스 작업</u> 참조 페 이지 및 <u>소스 개정 재정의로 파</u> 이프라인 시작을 참조하세요.	2024년 6월 7일
<u>새 출력 변수를 추가하기 위한</u> <u>S3 소스 작업 업데이트</u>	BucketName 및 ObjectKey 라는 새 출력 변수를 S3 소 스 작업에 사용할 수 있습니다. <u>Amazon S3 소스 작업</u> 참조 페 이지를 참조하세요.	2024년 6월 5일

<u>CloudFormationStac</u> <u>kSet 및 CloudForm</u> <u>ationStackInstances</u> 작업에 대한 업데이트	CloudFormationStac kSet 및 CloudForm ationStackInstances 작업에 CallAs 파라미터가 추 가되었습니다. <u>작업 참조 페이</u> <u>지</u> 를 참조하십시오.	2024년 5월 2일
<u>스테이지 수준 롤백 지원</u>	스테이지에 대해 이전에 성공 한 파이프라인 실행으로 스테 이지를 수동 또는 자동으로 롤 백할 수 있습니다. <u>스테이지 롤</u> <u>백 구성</u> 및 <u>개념</u> 을 참조하세요.	2024년 4월 26일
<u>StackSets 및 Step Functions</u> <u>작업에 대한 리전 가용성 업데</u> 이트	이제 CodePipeline을 사용 할 수 있는 모든 리전에서 StackSets 및 Step Functions 작업을 사용할 수 있습니 다. <u>AWS CloudFormation</u> <u>StackSets 작업 참조</u> 및 <u>AWS</u> <u>Step Functions 작업 참조</u> 를 참 조하세요.	2024년 3월 27일
<u>관리형 정책으로 업데이트</u>	AWS 관리형 정책이 업데이 트AWSCodePipeline_Fu 11Access 되었습니다. <u>AWS</u> <u>CodePipeline에 대한AWS 관</u> <u>리형 정책</u> 참조.	2024년 3월 15일
<u>수동 승인 작업을 위해 구성 가</u> 능한 제한 시간 지원	수동 승인 작업을 위해 구성 가 능한 새 제한 시간 필드에 할당 량 정보가 추가되었습니다. 자 세한 내용은 <u>할당량</u> 을 참조하 세요.	2024년 2월 15일

<u>브랜치 및 파일 경로별 트리거</u> 필터링 지원	V2 유형 파이프라인의 풀 요청 상태, 브랜치 및 파일 경로를 필 터링할 수 있는 트리거 구성에 대한 지원이 추가되었습니다. 자세한 내용은 <u>코드 푸시 또는</u> <u>풀 요청의 트리거 필터링, 트리</u> 거, <u>파이프라인을 시작하기 위</u> 한 기능 브랜치 필터링, 할당 량을 참조하세요.	2024년 2월 8일
<u>새 파이프라인 실행 모드 지원</u>	PARALLEL 및 QUEUED 파 이프라인 실행 모드에 지원이 추가되었습니다. 자세한 내용 은 <u>파이프라인 실행 모드 설정,</u> QUEUED 모드에서 실행을 처 리하는 방법, PARALLEL 모드 에서 실행을 처리하는 방법 및 할당량을 참조하세요.	2024년 2월 8일
작업 세부 정보 보기, 수동 승인 작업 검토를 위한 콘솔 페이지, 목록 파이프라인 페이지 업데 이트	콘솔 업데이트는 새로운 세부 정보 보기 버튼 및 대화 상자, 새 수동 승인 대화 상자, 목록 파이프라인 페이지의 최근 실 행에 대한 새 열에 대해 문서화 되어 있습니다. 자세한 내용은 파이프라인 보기(콘솔), 파이프 라인의 작업 세부 정보 보기, 파 이프라인의 승인 작업 관리를 참조하십시오.	2024년 1월 10일
<u>GitLab 자체 관리형 지원</u>	AWS 리소스가 GitLab 자체 관 리형과 상호 작용하도록 연결 을 구성하기 위한 지원이 추 가되었습니다. 자세한 정보는 <u>GitLab 자체 관리형 연결</u> 을 참 조하십시오.	2023년 12월 28일

CloudFormationStac <u>kSet 및CloudForm</u> ationStackInstances 작업에 대한 업데이트	CloudFormationStac kSet 및 CloudForm ationStackInstances 작업에 ConcurrencyMode 파라미터가 추가되었습니다. <u>작업 참조 페이지</u> 를 참조하십 시오.	2023년 12월 19일
<u>CodePipeline의 AWS Device</u> Farm 작업 파라미터 업데이트	CodePipeline의 AWS Device Farm 작업에 대한 파라미터가 업데이트되었습니다. 자세한 내용은 <u>AWS Device Farm 작</u> <u>업 참조</u> 를 참조하십시오.	2023년 12월 18일
<u>CodePipeline의 AWS</u> <u>CloudFormation 작업에 대한</u> <u>자세한 오류 메시지에 대한 지</u> 원이 추가되었습니다.	AWS CloudFormation 작업 오 류 메시지는 이제 실패한 리소 스에 대한 세부 정보를 표시할 수 있습니다. 자세한 내용은 <u>AWS CloudFormation 작업 참</u> <u>조</u> 를 참조하십시오.	2023년 12월 15일
CodePipeline에서 소스 개정 재정의를 사용하여 파이프라인 을 시작하기 위한 업데이트	이제 지정된 소스 개정으로 파 이프라인을 시작할 수 있습니 다. 자세한 내용은 <u>소스 개정 재</u> <u>정의로 파이프라인 시작</u> 을 참 조하십시오.	2023년 11월 17일

<u>지원되는 새로운 리전</u>	이제 아시아 태평양(하이데 라바드), 아시아 태평양(자카 르타), 아시아 태평양(멜버 른), 아시아 태평양(오사카), 중동(UAE), 유럽(스페인), 이 스라엘(텔아비브) 리전에서 CodePipeline을 사용할 수 있 습니다. <u>이벤트 자리 표시자 버</u> <u>킷 참조</u> 주제 및 <u>AWS 서비스</u> <u>엔드포인트</u> 주제가 업데이트되 었습니다.	2023년 11월 13일
<u>Amazon EventBridge의 이벤트</u> 필드 업데이트	이제 Amazon EventBridge에 서 업데이트된 이벤트 필드를 볼 수 있습니다. 자세한 내용은 <u>CodePipeline 이벤트 모니터</u> <u>링</u> 을 참조하세요.	2023년 11월 9일
<u>새 파이프라인 유형 V2 파이 프라인, Git 태그의 트리거,</u> CodePipeline의 파이프라인 변 수에 대한 업데이트	이제 CodePipeline에서 파이프 라인 유형을 선택할 수 있습니 다. V2 유형 파이프라인의 경우 이제 트리거 구성을 사용하여 Git 태그에서 파이프라인을 시 작할 수 있습니다. V2 유형 파 이프라인을 사용하면 파이프라 인 수준에서 변수를 사용하여 파이프라인 실행을 위한 입력 파라미터를 전달할 수도 있습 니다. 자세한 내용은 변수, 자습 서: 파이프라인 수준 변수 사용 및 자습서: Git 태그를 사용하 여 파이프라인 시작을 참조하 세요. 파이프라인 유형에 대한 자세한 내용은 파이프라인 유 형 단원을 참조하세요.	2023년 10월 24일

<u>CodePipeline을 사용하면 실패</u> <u>한 단계에서 모든 작업을 재시</u> <u>도할 수 있습니다.</u>	CodePipeline에서 실패한 단계 의 경우 파이프라인을 다시 실 행하지 않고 단계를 재시도할 수 있습니다. 이렇게 하려면 단 계에서 실패한 작업을 다시 시 도하거나 단계의 첫 번째 작업 부터 시작하여 단계의 모든 작 업을 다시 시도하면 됩니다. 자 세한 내용은 다음을 참조하세 요.	2023년 10월 17일
<u>GitLab 그룹 지원</u>	AWS 리소스가 GitLab 그룹과 상호 작용하도록 연결을 구성 하기 위한 지원이 추가되었습 니다. 자세한 정보는 <u>GitLab 연</u> <u>결</u> 을 참조하세요.	2023년 9월 15일
<u>CodePipeline의 GitLab.com과</u> <u>의 연결 지원</u>	연결을 사용하여 GitLab.com 과 상호 작용하도록 AWS 리 소스를 구성할 수 있습니다. GitLab.com. 또한 다운스트림 작업에 Git 명령 및 메타데이 터를 사용하기 위한 전체 복 제 옵션을 선택할 수 있습니다. 자세한 내용은 <u>GitLab 연결</u> 및 <u>CodeStarSourceConnection 작</u> 업 구조 참조 주제를 참조하세 요.	2023년 8월 10일
<u>CloudFormationStac</u> <u>kInstances 작업에 대한</u> <u>업데이트</u>	CloudFormationStac kInstances 작업에 RegionConcurrencyT ype 파라미터가 추가되 었습니다.CloudForm ationStackInstances 작업에 대해서는 <u>작업 참조 페</u> 이지를 참조하세요.	2023년 8월 8일

사용자	가이	
-----	----	--

<u>CloudFormationStac</u> <u>kSet 작업에대한업데이트</u>	CloudFormationStac kSet 작업에 RegionCon currencyType 파라미터가 추가되었습니다. CloudForm ationStackSet 작업에 대 해서는 <u>작업 참조 페이지</u> 를 참 조하세요.	2023년 7월 24일
<u>관리형 정책으로 업데이트</u>	AWS 관리형 정책이 업데이 트AWSCodePipeline_Fu 11Access 되었습니다. <u>AWS</u> <u>CodePipeline에 대한AWS 관</u> <u>리형 정책</u> 참조.	2023년 6월 21일
<u>폴링 파이프라인의 마이그레이</u> <u>션 절차 업데이트</u>	이벤트 기반 변경 감지를 사용 하기 위해 폴링 파이프라인을 마이그레이션(업데이트)하는 절차가 EventBridge에 대한 알 림용으로 활성화된 Amazon S3 버킷을 사용하는 파이프라인 의 단계로 업데이트되었습니다 . 자세한 정보는 <u>이벤트 기반 변</u> <u>경 감지를 사용하도록 폴링 파</u> <u>이프라인 마이그레이션</u> 을 참조 하세요.	2023년 6월 12일
<u>관리형 정책으로 업데이트</u>	AWS 관리형 정책 AWSCodePi peline_FullAccess 및가 추가 권한으로 업데이 트AWSCodePipeline_Re adOnlyAccess 되었습니다. 자세한 내용은 <u>AWS CodePipel</u> ineAWS 관리형 정책에 대한 업 데이트를 참조하세요.	2023년 5월 16일

<u>관리형 정책으로 업데이트</u>	AWS 관리형 정책 AWSCodePi pelineFullAccess 및 AWSCodePipelineRea dOnlyAccess 는 더 이상 사 용되지 않습니다. AWSCodePi peline_FullAccess 및 AWSCodePipeline_Re adOnlyAccess 정책을 사 용하세요. <u>AWS CodePipeline</u> <u>가 AWS 관리형 정책으로 업데</u> 이트를 참조하세요.	2022년 11월 17일
<u>CloudTrail을 사용하는 절차에</u> <u>대한 업데이트</u>	S3 소스가 있는 파이프라 인의 모든 콘솔 절차, 샘플 CLI 명령, 샘플 코드 AWS CloudFormation 조각 및 템플 릿이 CloudTrail에서 관리 이벤 트에 대해 쓰기 및 false를 선택 하는 옵션으로 업데이트되었 습니다. 파이프라인 시작, 자습 서: 로 파이프라인 시작, 자습 서: 로 파이프라인 생성 AWS CloudFormation, 푸시 이벤트 를 사용하도록 파이프라인 편 집, 폴링 파이프라인 업데이트 의 업데이트된 샘플을 참조하 세요.	2022년 4월 27일
<u>Snyk와의 새로운 지원 통합</u>	CodePipeline의 Snyk 호출 작 업을 사용하여 오픈 소스 코 드에 대한 보안 검사를 자동화 할 수 있습니다. 자세한 내용은 <u>Snyk 작업 참조</u> 및 <u>통합</u> 을 참조 하세요.	2021년 6월 10일

<u>신규 지원 리전 유럽(밀라노)</u>	이제 유럽(밀라노)에서 CodePipeline을 사용할 수 있 습니다. <u>제한</u> 주제와 <u>AWS 서비</u> <u>스 엔드포인트</u> 주제가 업데이 트되었습니다.	2021년 1월 27일
<u>연결이 있는 소스 작업의 경우</u> 변경 감지를 끌 수 있습니다.	CLI 또는 SDK를 사용하여 CodeStarSourceConn ection 소스 작업을 업데이 트하여 소스 리포지토리에 대 한 자동 변경 감지를 끌 수 있 습니다. <u>CodeStarSourceConn</u> <u>ection 작업 구조 참조</u> 주제가 DetectChanges 파라미터에 대한 설명으로 업데이트되었습 니다.	2021년 1월 8일
<u>CodePipeline에서 이제 AWS</u> <u>CloudFormation StackSets 배</u> 포 작업 지원	새로운 자습서인 <u>자습서: 배</u> <u>포 공급자로 AWS CloudForm</u> ation StackSets를 사용하는 <u>파이프라인 생성</u> 에서는 AWS CloudFormation StackSets를 사용하여 파이프라인으로 스 택 세트 및 스택 인스턴스를 생 성하고 업데이트하는 단계를 제공합니다. <u>AWS CloudForm</u> ation StackSets 작업 구조 참 조 주제도 추가되었습니다.	2020년 12월 30일
<u>새로 지원되는 아시아 태평양</u> (홍콩) 리전	이제 아시아 태평양(홍콩) 리전 에서 CodePipeline을 사용할 수 있습니다. <u>제한</u> 주제와 <u>AWS</u> <u>서비스 엔드포인트</u> 주제가 업 데이트되었습니다.	2020년 12월 22일

<u>CodePipeline에서 업데이트된</u> EventBridge 이벤트 패턴 보기	파이프라인, 단계, 작업 수준 이 벤트에 대한 업데이트된 이벤 트 패턴과 상태가 <u>CodePipel</u> <u>ine 이벤트 모니터링</u> 에 추가되 었습니다.	2020년 12월 21일
<u>CodePipeline에서 인바운드 파</u> <u>이프라인 실행 보기</u>	콘솔 또는 CLI를 사용하여 인바 운드 실행을 확인할 수 있습니 다. 자세한 내용은 <u>인바운드 실</u> <u>행 보기(콘솔)</u> 및 <u>인바운드 실</u> <u>행 상태 보기(CLI)</u> 를 참조하세 요.	2020년 11월 16일
<u>CodePipeline의 CodeCommit</u> <u>소스 작업은 전체 복제 옵션을</u> <u>지원합니다.</u>	CodeCommit 소스 작업을 사용 할 때 다운스트림 CodeBuild 작 업에 Git 명령 및 메타데이터를 사용하기 위한 전체 복제 옵션 을 선택할 수 있습니다. 자세한 내용은 <u>CodeCommit 작업 참조</u> 및 <u>자습서: CodeCommit 파이</u> 프라인 소스와 함께 전체 복제 사용을 참조하세요.	2020년 11월 11일

<u>CodePipeline은 GitHub 및</u> <u>GitHub Enterprise Server에 대</u> 한 연결을 지원합니다.	연결을 사용하여 GitHub, GitHub Enterprise Cloud 및 GitHub GitHub Enterprise Server와 상호 작용하도록 AWS 리소스를 구성할 수 있 습니다. 또한 다운스트림 작업 에 Git 명령 및 메타데이터를 사용하기 위한 전체 복제 옵션 을 선택할 수 있습니다. 자세 한 내용은 <u>GitHub 연결, GitHub</u> Enterprise Server 연결 및 <u>자습</u> 서: GitHub 파이프라인 소스와 함께 전체 복제 사용을 참조하 세요. GitHub 소스 작업이 있는 기존 파이프라인이 있는 경우 GitHub(OAuth 앱을 통해) 소스 작업을 GitHub(GitHub 앱을 통 해) 소스 작업으로 업데이트를 참조하세요.	2020년 9월 30일
<u>CodeBuild 작업은에서 배치 빌</u> <u>드 활성화를 지원합니다. AWS</u> <u>CodePipeline</u>	파이프라인의 CodeBuild 작업 의 경우 배치 빌드를 활성화하 여 한 번의 실행으로 여러 빌드 를 실행할 수 있습니다. 자세한 내용은 <u>CodeBuild 작업 구조 참</u> 조 및 <u>파이프라인 생성(콘솔)</u> 을 참조하세요.	2020년 7월 30일
<u>AWS CodePipeline 는 이제</u> <u>AWS AppConfig 배포 작업을</u> 지원합니다.	새로운 자습서인 <u>자습서: AWS</u> AppConfig를 배포 공급자로 사 용하는 파이프라인 생성에서는 AWS AppConfig를 사용하여 파이프라인과 함께 구성 파일 을 배포하는 단계를 제공합니 다. <u>AWS AppConfig 작업 구조</u> 참조 주제도 추가되었습니다.	2020년 6월 25일

AWS CodePipeline 는 이 제 Amazon VPC in AWS GovCloud(미국 서부)를 지원합 니다.	이제 AWS GovCloud(미국 서 부)의 프라이빗 Amazon VPC 엔드포인트를 AWS CodePipel ine 통해에 직접 연결할 수 있습 니다. 자세한 내용은 <u>Amazon</u> <u>Virtual Private Cloud에서</u> <u>CodePipeline 사용</u> 을 참조하세 요.	2020년 6월 2일
<u>AWS CodePipeline 에서 이제</u> <u>AWS Step Functions 호출 작</u> <u>업 지원</u>	이제를 호출 작업 공급자 AWS Step Functions 로 사용하는 파 이프라인을 CodePipeline에서 생성할 수 있습니다. 새로운 자 습서인 <u>자습서: 파이프라인에</u> <u>서 AWS Step Functions 호출</u> <u>작업을 사용하면</u> 파이프라인 에서 상태 시스템 실행을 시작 하는 단계를 제공합니다. <u>AWS</u> <u>Step Functions 작업 구조 참조</u> 주제도 추가되었습니다.	2020년 5월 28일
<u>연결 보기, 나열 및 업데이트</u>	콘솔에서 연결을 나열, 삭제 및 업데이트할 수 있습니다. <u>CodePipeline의 연결 나열</u> 을 참조하세요.	2020년 5월 21일
<u>연결은 CLI에서 연결 리소스에</u> <u>대한 태그 지정 지원</u>	연결 리소스는 이제 AWS CLI 에서 태그 지정을 지원합니다. 이제 연결은 AWS CodeGuru와 통합됩니다. <u>연결을 위한 IAM</u> <u>권한 참조</u> 를 참조하십시오.	2020년 5월 6일
<u>이제 AWS GovCloud(미국 서</u> <u>부)에서 CodePipeline을 사용</u> <u>할 수 있습니다.</u>	이제 CodePipeline in AWS GovCloud(미국 서부)를 사용할 수 있습니다. 자세한 내용은 <u>할</u> <u>당량</u> 을 참조하세요.	2020년 4월 8일

<u>할당량 주제에서는 구성할 수</u> <u>있는 CodePipeline 서비스 할</u> <u>당량을 보여줍니다.</u>	CodePipeline 할당량 주제의 콘텐츠가 재구성되었습니다. 문서에는 구성할 수 있는 서비 스 할당량과 구성할 수 없는 할 당량이 나와 있습니다. <u>할당량</u> <u>in AWS CodePipeline</u> 을 참조 하세요.	2020년 3월 12일
<u>Amazon ECS 배포 작업 제한</u> <u>시간을 구성할 수 있습니다.</u>	Amazon ECS 배포 작업 제한 시간은 최대 1시간(기본 제한 시간)으로 구성할 수 있습니 다. <u>AWS CodePipeline의 할당</u> <u>량</u> 을 참조하세요.	2020년 2월 5일
<u>새 항목에서는 파이프라인 실</u> <u>행을 중지하는 방법에 대해 설</u> <u>명합니다.</u>	CodePipeline에서 파이프라인 실행을 중지할 수 있습니다. 진 행 중인 작업이 완료될 때까지 기다린 후 실행을 중지하도록 지정하거나, 즉시 실행을 중지 하고 진행 중인 작업을 중단하 도록 지정할 수 있습니다. <u>파이</u> <u>프라인 실행을 중지하는 방법</u> 및 <u>CodePipeline에서 파이프라</u> 인 실행 중지를 참조하세요.	2020년 1월 21일
<u>CodePipeline의 연결 지원</u>	연결을 사용하여 외부 코드 리 포지토리와 상호 작용하도록 AWS 리소스를 구성할 수 있 습니다. 각 연결은 CodePipel ine 같은 서비스에서 타사 리 포지토리(예: Bitbucket Cloud) 에 연결하는 데 사용할 수 있는 리소스입니다. 자세한 내용은 <u>CodePipeline에서 연결 관련</u> 작업을 참조하십시오.	2019년 12월 18일

<u>업데이트된 보안, 인증 및 액세</u> <u>스 제어 주제</u>	CodePipeline에 대한 보안, 인 증 및 액세스 제어 정보가 새로 운 보안 장으로 구성되었습니 다. 자세한 내용은 <u>보안</u> 을 참조 하세요.	2019년 12월 17일
<u>새 항목에서는 파이프라인에서</u> 변수를 사용하는 방법에 대해 설명합니다.	이제 작업에 대한 네임스페이 스를 구성하고 작업 실행이 완 료될 때마다 변수를 생성할 수 있습니다. 이러한 네임스페이 스 및 변수를 참조하도록 다운 스트림 작업을 설정할 수 있습 니다. <u>변수 작업</u> 및 <u>변수</u> 를 참조 하십시오.	2019년 11월 14일
<u>새 항목에서는 파이프라인 실</u> 행의 작동 방식, 실행 중에 단계 <u>가 잠기는 이유 및 파이프라인</u> 실행이 교체되는 시기를 설명 합니다.	시작 섹션에는 실행 중에 단계 가 잠기는 이유와 파이프라인 실행이 교체될 때 발생하는 일 을 비롯하여 파이프라인 실행 이 작동하는 방식을 설명하는 여러 항목이 추가되었습니다. 이러한 항목에는 개념 목록, DevOps 워크플로 예제 및 파이 프라인 구성 방법에 대한 권장 사항이 포함되어 있습니다. 파 이프라인 용어, DevOps 파이프 라인 예제 및 파이프라인 실행 의 작동 방식 주제가 추가되었 습니다.	2019년 11월 11일
<u>CodePipeline의 알림 규칙 지</u> <u>원</u>	이제 알림 규칙을 사용하여 파 이프라인의 중요한 변경 사항 을 사용자에게 알릴 수 있습니 다. 자세한 내용은 <u>알림 규칙 생</u> <u>성</u> 을 참조하세요.	2019년 11월 5일

<u>CodePipeline에서 사용할 수</u> <u>있는 CodeBuild 환경 변수</u>	파이프라인의 CodeBuild 빌 드 작업에서 CodeBuild 환경 변수를 설정할 수 있습니다. 콘솔이나 CLI를 사용하여 파 이프라인 구조에 Environme ntVariables 파라미터를 추가할 수 있습니다. 파이프라 인 생성(콘솔) 주제가 업데이트 되었습니다. CodeBuild에 대한 작업 참조의 작업 구성 예제도 업데이트되었습니다.	2019년 10월 14일
<u>새로운 리전</u>	이제 유럽(스톡홀름)에서 CodePipeline을 사용할 수 있 습니다. <u>제한</u> 주제와 <u>AWS 서비</u> <u>스 엔드포인트</u> 주제가 업데이 트되었습니다.	2019년 9월 5일
<u>Amazon S3 배포 작업에 대한</u> 표준 ACL 및 캐시 제어를 지정 합니다.	이제 CodePipeline에서 Amazon S3 배포 작업을 생 성할 때 표준 ACL 및 캐시 제 어 옵션을 지정할 수 있습니 다. <u>파이프라인 생성(콘솔),</u> <u>CodePipeline 파이프라인 구조</u> 참조 및 자습서: Amazon S3를 <u>배포 공급자로 사용하는 파이</u> <u>프라인 생성</u> 주제가 업데이트 되었습니다.	2019년 6월 27일

이제의 리소스에 태그를 이제 태그 지정을 사용하여 파 2019년 5월 15일 추가할 수 있습니다. AWS 이프라인, 사용자 지정 작업 및 **CodePipeline** 웹후크와 같은 AWS CodePipel ine 리소스를 추적하고 관리할 수 있습니다. 리소스 태그 지 정, 태그를 사용하여 CodePipel ine 리소스에 대한 액세스 제어, CodePipeline에서 파이프라인 태그 지정, CodePipeline에서 사용자 지정 작업 태그 지정 및 CodePipeline에서 Webhook 태 그 지정 주제가 새로 추가되었 습니다. CLI를 사용하여 리소 스에 태그를 지정하는 방법을 보여주기 위해 파이프라인 생 성(CLI), 사용자 지정 작업 생 성(CLI) 및 GitHub 소스에 대한 Webhook 생성 주제가 업데이 트되었습니다. 이제에서 작업 실행 기록을 볼 2019년 3월 20일 이제 파이프라인에서 모든 작 수 있습니다. AWS CodePipel 업의 과거 실행에 대한 세부 정 보를 볼 수 있습니다. 이러한 세 ine 부 정보에는 시작 및 종료 시간. 기간, 작업 실행 ID, 상태, 입력 및 출력 아티팩트 위치 세부 정 보 및 외부 리소스 세부 정보가 포함됩니다. 파이프라인 세부 정보 및 내역 보기 주제는 이 지 원을 반영하도록 업데이트되었 습니다.
AWS CodePipeline 는 이제에 애플리케이션 게시를 지원합니 다. AWS Serverless Applicati on Repository	이제 서버리스 애플리케이션 을 AWS Serverless Applicati on Repository에 게시하는 CodePipeline 파이프라인을 생 성할 수 있습니다. 새로운 자습 서인 <u>자습서:에 애플리케이션</u> 게시 AWS Serverless Applicati on Repository에서는 서버리스 애플리케이션을에 지속적으로 전달하도록 파이프라인을 생성 하고 구성하는 단계를 제공합 니다 AWS Serverless Applicati on Repository.	2019년 3월 8일
<u>AWS CodePipeline 는 이제 콘</u> <u>솔에서 교차 리전 작업을 지원</u> <u>합니다.</u>	이제 AWS CodePipeline 콘솔 에서 리전 간 작업을 관리할 수 있습니다. <u>교차 리전 작업 추</u> 가는 파이프라인과 다른 AWS 리전에 있는 작업을 추가, 편 집 또는 삭제하는 단계로 업 데이트되었습니다. <u>파이프라</u> 인 생성, <u>파이프라인 편집</u> 및 <u>CodePipeline 파이프라인 구조</u> 참조 주제를 업데이트했습니 다.	2019년 2월 14일

<u>AWS CodePipeline 는 이제</u> <u>Amazon S3 배포를 지원합니</u> <u>다.</u>	이제 Amazon S3를 배포 작업 공급자로 사용하는 CodePipel ine에서 파이프라인을 생성할 수 있습니다. 새로운 자습서 <u>자</u> 습서: Amazon S3를 배포 공급 <u>자로 사용하는 파이프라인 생</u> 성에서는 CodePipeline을 사용 하여 Amazon S3 버킷에 샘플 파일을 배포하는 단계를 제공 합니다. <u>CodePipeline 파이프</u> <u>라인 구조 참조</u> 주제도 업데이 트되었습니다.	2019년 1월 16일
<u>AWS CodePipeline 는 이제</u> <u>Alexa Skills Kit 배포를 지원합</u> 니다.	이제 Alexa Skill의 지속적 배포 를 위해 CodePipeline 및 Alexa Skills Kit를 사용할 수 있습니 다. 새로운 자습서인 <u>자습서</u> : Amazon Alexa 스킬을 배포 <u>하는 파이프라인 생성</u> 에는가 Alexa Skills Kit 개발자 계정에 AWS CodePipeline 연결할 수 있는 자격 증명을 생성한 다음 샘플 스킬을 배포하는 파이프 라인을 생성하는 단계가 포함 되어 있습니다. <u>CodePipeline</u> 파이프라인 구조 참조 주제가 업데이트되었습니다.	2018년 12월 19일
<u>AWS CodePipeline 는 이제</u> <u>AWS PrivateLink로 구동되는</u> <u>Amazon VPC 엔드포인트를 지</u> 원합니다.	이제 VPC의 프라이빗 엔드포 인트를 AWS CodePipeline 통 해에 직접 연결하여 모든 트래 픽을 VPC와 AWS 네트워크 내 에 유지할 수 있습니다. 자세한 내용은 <u>Amazon Virtual Private</u> <u>Cloud에서 CodePipeline 사</u> 용을 참조하세요.	2018년 12월 6일

AWS CodePipeline 는 이제 Amazon ECR 소스 작업 및 ECS-to-CodeDeploy 배포 작업 을 지원합니다.	컨테이너 기반 애플리케이션의 지속적 배포를 위해 Amazon ECR 및 Amazon ECS와 함께 CodePipeline 및 CodeDeplo y를 사용할 수 있습니다. 새 로운 자습서인 Amazon ECR 소스 및 ECS-to-CodeDeploy 배포를 사용하여 파이프라인 생성에서는 콘솔을 사용하여 CodeDeploy 트래픽 라우팅을 통해 이미지 리포지토리에 저 장된 컨테이너 애플리케이션을 Amazon ECS 클러스터에 배포 하는 파이프라인을 생성하는 절차를 설명합니다. 파이프라 인 생성 및 CodePipeline 파이 프라인 구조 참조 주제를 업데 이트했습니다.	2018년 11월 27일
AWS CodePipeline 는 이제 파 이프라인에서 리전 간 작업을 지원합니다.	새 주제인 <u>교차 리전 작업 추</u> 가에는 AWS CLI 또는를 사 용하여 파이프라인과 다른 리전에 있는 작업을 AWS CloudFormation 추가하는 단계 가 포함되어 있습니다. <u>파이프</u> 라인 생성, <u>파이프라인 편집</u> 및 <u>CodePipeline 파이프라인 구조</u> 참조 주제를 업데이트했습니 다.	2018년 11월 12일

<u>AWS CodePipeline 이제가</u> <u>Service Catalog와 통합됩니다.</u>	이제 Service Catalog를 배포 작업으로 파이프라인에 추가 할 수 있습니다. 이렇게 하면 소 스 리포지토리에서 변경 사항 을 적용할 때 Service Catalog 에 제품 업데이트를 게시하 는 파이프라인을 설정할 수 있 습니다. 통합 주제는 Service Catalog를 위해 이 지원을 반 영하도록 업데이트되었습니다 . 두 개의 Service Catalog 자습 서가 <u>AWS CodePipeline 자습</u> 서 단원에 추가되었습니다.	2018년 10월 16일
<u>AWS CodePipeline 이제가</u> <u>와 통합됩니다. AWS Device</u> <u>Farm</u>	이제를 파이프라인에 테스트 작업 AWS Device Farm 으로 추가할 수 있습니다. 이렇게 하면 모바일 애플리케이션을 테스트하기 위해 파이프라인 을 설정하도록 허용합니다. 통 합 주제가이 지원을 반영하도 록 업데이트되었습니다 AWS Device Farm 자습서가 AWS Device Farm 자습서가 AWS CodePipeline 자습서 단원에 추가되었습니다.	2018년 7월 19일

2018년 6월 30일

AWS CodePipeline 이제 RSS 를 통해 사용 설명서 업데이트 알림 사용 가능 CodePipeline 사용 설명서의 HTML 버전이 설명서 업데이트 기록 페이지에 명시된 업데이 트의 RSS 피드를 지원합니다. RSS 피드에는 2018년 6월 30 일 이후의 업데이트가 포함됩 니다. 이전에 발표한 업데이트 도 설명서 업데이트 기록 페이 지에서 이용할 수 있습니다. 피 드를 구독하려면 상단 메뉴판 에서 RSS 버튼을 누릅니다.

이전 업데이트

다음 표에서는 2018년 6월 30일 이전 CodePipeline 사용 설명서의 각 릴리스에서 변경된 중요 사항에 대해 설명합니다.

변경 사항	설명	변경 날짜
Webhook를 사용하 여 GitHub 파이프라 인의 소스 변경 감지	콘솔에서 파이프라인을 생성하거나 편집하면 CodePipeline 이 GitHub 소스 리포지토리 변경을 감지하는 Webhook를 생 성한 다음 파이프라인을 시작합니다. 파이프라인 마이그레 이션에 대한 자세한 내용은 <u>Webhook를 사용하여 변경을 감</u> 지하도록 GitHub 파이프라인 구성을 참조하십시오. 자세한 내용은 <u>CodePipeline의 파이프라인 실행 시작</u> 을 참조하세 요.	2018년 5월 1일
업데이트된 주제	콘솔에서 파이프라인을 생성하거나 편집할 때 CodePipeline 은 이제 Amazon CloudWatch Events 규칙과 Amazon S3 소 스 버킷의 변경 사항을 감지한 다음 파이프라인을 시작하는 AWS CloudTrail 추적을 생성합니다. 파이프라인 마이그레 이션에 대한 자세한 내용은 <u>소스 작업 및 변경 감지 방법</u> 단 원을 참조하십시오. <u>자습서: 간단한 파이프라인 생성(S3 버킷)</u> 단원이 Amazon S3 소스를 선택할 때 Amazon CloudWatch Events 규칙 및	2018년 3월 22일

변경 사항	설명	변경 날짜
	추적이 생성되는 방법을 보여주도록 업데이트되었습니다. <u>파이프라인 스테이지 및 작업 생성 및 CodePipeline에서 파</u> <u>이프라인 편집</u> 역시 업데이트되었습니다. 자세한 내용은 <u>CodePipeline에서 파이프라인 시작</u> 단원을 참조하십시오.	
업데이트된 주제	이제 유럽(파리)에서 CodePipeline을 사용할 수 있습니다. <u>in</u> <u>AWS CodePipeline 할당량</u> 주제가 업데이트되었습니다.	2018년 2월 21일
업데이트된 주제	이제 컨테이너 기반 애플리케이션의 지속적 배포를 위해 CodePipeline 및 Amazon ECS를 사용할 수 있습니다. 파이 프라인을 만들 때 Amazon ECS를 배포 공급자로 선택할 수 있습니다. 소스 제어 리포지토리의 코드를 변경하면 파이프 라인이 새 도커 이미지를 빌드하고 이를 컨테이너 레지스트 리에 푸시한 후 업데이트된 이미지를 Amazon ECS 서비스 에 배포합니다. <u>CodePipeline과 제품 및 서비스 통합, 파이프라인 스테이지</u> 및 작업 생성 및 <u>CodePipeline 파이프라인 구조 참조</u> 주제가 Amazon ECS에 대한 이 지원을 반영하도록 업데이트되었 습니다.	2017년 12월 12일

변경 사항	설명	변경 날짜
업데이트된 주제	콘솔에서 파이프라인을 만들거나 편집하면 CodePipeline 이 CodeCommit 리포지토리 변경을 감지하는 Amazon CloudWatch Events 규칙을 만든 후 파이프라인을 자동으로 시작합니다. 기존 파이프라인 마이그레이션에 대한 자세한 내용은 <u>소스 작업 및 변경 감지 방법</u> 단원을 참조하십시오.	2017년 10월 11일
	<u>자습서: 간단한 파이프라인 생성(CodeCommit 리포지토</u> 리) 단원이 CodeCommit 리포지토리 및 브랜치를 선택할 때 Amazon CloudWatch Events 규칙 및 역할이 생성되는 방법 을 보여주도록 업데이트되었습니다. <u>파이프라인 스테이지</u> <u>및 작업 생성</u> 및 <u>CodePipeline에서 파이프라인 편집</u> 역시 업 데이트되었습니다.	
	자세한 내용은 <u>CodePipeline에서 파이프라인 시작</u> 단원을 참조하십시오.	
신규 및 업데이트된 주제	CodePipeline이 이제 Amazon CloudWatch Events 및 Amazon Simple Notification Service(SNS)를 통해 파이프라 인 상태 변경 알림을 기본적으로 지원합니다. 새 자습서인 <u>자습서: 파이프라인 상태 변경에 대한 이메일 알림을 수신하</u> <u>도록 CloudWatch Events 규칙 설정</u> 이 추가되었습니다. 자 세한 내용은 <u>CodePipeline 이벤트 모니터링</u> 단원을 참조하 십시오.	2017년 9월 8일
신규 및 업데이트된 주제	이제 CodePipeline을 Amazon CloudWatch Events 작업 의 대상으로 추가할 수 있습니다. 소스 변경을 감지하도록 Amazon CloudWatch Events 규칙을 설정하여 그러한 변경 이 발생하자마자 파이프라인이 시작되도록 하거나, 예약된 파이프라인을 실행하도록 설정할 수 있습니다. PollForSo urceChanges 소스 작업 구성 옵션에 대한 정보가 추가되었 습니다. 자세한 내용은 <u>CodePipeline에서 파이프라인 시작</u> 단원을 참조하십시오.	2017년 9월 5일

변경 사항	설명	변경 날짜
새로운 리전	CodePipeline을 이제 아시아 태평양(서울) 및 아시아 태평양 (뭄바이)에서 사용할 수 있습니다. <u>in AWS CodePipeline 할</u> <u>당량</u> 주제와 <u>리전 및 엔드포인트</u> 주제가 업데이트되었습니 다.	2017년 7월 27일
새로운 리전	이제 CodePipeline을 미국 서부(캘리포니아 북부), 캐나 다(중부), 유럽(런던)에서 사용할 수 있습니다. <u>in AWS</u> <u>CodePipeline 할당량</u> 주제와 <u>리전 및 엔드포인트</u> 주제가 업 데이트되었습니다.	2017년 6월 29일
업데이트된 주제	이제 파이프라인의 가장 최근의 실행 내역이 아닌, 지난 실 행 내역에 대한 세부 정보도 볼 수 있습니다. 이 세부 정보에 는 시작 및 종료 시간, 실행 기간 및 실행 ID가 포함됩니다. 세부 정보는 최근 12개월 동안 실행된 100개 이하의 파이 프라인 실행에 대해 표시됩니다. 주제 <u>CodePipeline에서 파</u> 이프라인 및 세부 정보 보기, <u>CodePipeline 권한 참조</u> 및 in <u>AWS CodePipeline 할당량</u> 가 이 지원 사항을 반영하도록 업 데이트되었습니다.	2017년 6월 22일
업데이트된 주제	<u>Nouvola</u> 가 <u>테스트 작업 통합</u> 에서 사용 가능한 작업의 목록 에 추가되었습니다.	2017년 5월 18일
업데이트된 주제	AWS CodePipeline 마법사에서 4단계: 베타의 이름이 4단 계: 배포로 변경되었습니다. 이 단계에서 생성된 단계의 기 본 이름이 "베타"에서 "스테이징"으로 변경되었습니다. 여러 주제와 스크린샷이 이러한 변경 사항을 반영하도록 업데이 트되었습니다.	2017년 7월 4일

변경 사항	설명	변경 날짜
업데이트된 주제	이제 파이프라인의 모든 단계에 테스트 작업 AWS CodeBuild 으로를 추가할 수 있습니다. 이렇게 하면를 사 용하여 코드 AWS CodeBuild 에 대한 단위 테스트를 보 다 쉽게 실행할 수 있습니다. 이 릴리스 이전에는 AWS CodeBuild 를 사용하여 빌드 작업의 일부로만 단위 테스트 를 실행할 수 있습니다. 빌드 작업을 수행하려면 단위 테스 트가 일반적으로 생성되지 않는 빌드 출력 아티팩트가 필요 합니다. CodePipeline과 제품 및 서비스 통합, CodePipeline에서 파이프라인 편집및 주제가이 지원을 반영하도록 업데이 트CodePipeline 파이프라인 구조 참조되었습니다 AWS CodeBuild.	2017년 3월 8일
신규 및 업데이트된 주제	목차가 파이프라인, 작업, 단계 전환에 대한 섹션을 포함하 도록 재구성되었습니다. CodePipeline 자습서에 대한 새 단 원이 추가되었습니다. 활용도 향상을 위해 <u>CodePipeline과</u> 제품 및 서비스 통합이 더 짧은 주제로 분류되었습니다. 새 단원인 권한 부여 및 액세스 제어는 <u>AWS Identity and</u> <u>Access Management (IAM)</u> 및 CodePipeline을 사용하여 자 격 증명의 사용을 통해 리소스에 대한 액세스를 보호할 수 있는 방법에 대한 포괄적인 정보를 제공합니다. 이러한 자격 증명은 Amazon S3 버킷에서 아티팩트를 입력 및 검색하고 AWS OpsWorks 스택을 파이프라인에 통합하는 등 AWS 리 소스에 액세스하는 데 필요한 권한을 제공합니다.	2017년 2월 8일
새로운 리전	이제 아시아 태평양(도쿄)에서 CodePipeline을 사용할 수 있 습니다. <u>in AWS CodePipeline 할당량</u> 주제와 <u>리전 및 엔드</u> <u>포인트</u> 주제가 업데이트되었습니다.	2016년 14월 12일
새로운 리전	이제 남아메리카(상파울루)에서 CodePipeline을 사용할 수 있습니다. <u>in AWS CodePipeline 할당량</u> 주제와 <u>리전 및 엔</u> <u>드포인트</u> 주제가 업데이트되었습니다.	2016년 7월 12일

변경 사항	설명	변경 날짜
업데이트된 주제	이제를 파이프라인의 모든 단계에 빌드 작업 AWS CodeBuild 으로 추가할 수 있습니다. AWS CodeBuild 는 소 스 코드를 컴파일하고 단위 테스트를 실행하며 배포할 준비 가 된 아티팩트를 생성하는 클라우드의 완전 관리형 빌드 서 비스입니다. 기존의 빌드 프로젝트를 사용하거나 CodePipel ine 콘솔에서 하나를 만들 수 있습니다. 그런 다음 빌드 프로 젝트의 출력을 파이프라인의 일부로 배포할 수 있습니다. CodePipeline과 제품 및 서비스 통합, 파이프라인 스테이지 및 작업 생성, 인증 및 액세스 제어 주제가이 지원을 반영하 도록 업데이트CodePipeline 파이프라인 구조 참조되었습니 다 AWS CodeBuild. 이제 CodePipeline을 AWS CloudFormation 및 AWS 서버리 스 애플리케이션 모델과 함께 사용하여 서버리스 애플리케 이션을 지속적으로 제공할 수 있습니다. 주제 <u>CodePipeline</u> 과 제품 및 서비스 통합이 이 지원을 반영하도록 업데이트되 었습니다.	2016년 1월 12일
	<u>CodePipeline과 제품 및 서비스 통합</u> 는 작업 유형별로 AWS 및 파트너 상품을 그룹화하도록 재구성되었습니다.	
새로운 리전	이제 유럽(프랑크푸르트)에서 CodePipeline을 사용할 수 있 습니다. <u>in AWS CodePipeline 할당량</u> 주제와 <u>리전 및 엔드</u> <u>포인트</u> 주제가 업데이트되었습니다.	2016년 11월 16일
업데이트된 주제	AWS CloudFormation 이제를 파이프라인의 배포 공급자 로 선택할 수 있으므로 파이프라인 실행의 일부로 AWS CloudFormation 스택에 대한 작업을 수행하고 세트를 변경 할 수 있습니다. <u>CodePipeline과 제품 및 서비스 통합, 파이</u> <u>프라인 스테이지 및 작업 생성</u> , 인증 및 액세스 제어 주제가 이 지원을 반영하도록 업데이트 <u>CodePipeline 파이프라인 구</u> <u>조 참조</u> 되었습니다 AWS CloudFormation.	2016년 11월 3일

변경 사항	설명	변경 날짜
새로운 리전	이제 아시아 태평양(시드니) 리전에서 CodePipeline을 사용 할 수 있습니다. <u>in AWS CodePipeline 할당량</u> 주제와 <u>리전</u> <u>및 엔드포인트</u> 주제가 업데이트되었습니다.	2016년 10월 26일
새로운 리전	이제 아시아 태평양(싱가포르)에서 CodePipeline을 사용할 수 있습니다. <u>in AWS CodePipeline 할당량</u> 주제와 <u>리전 및</u> <u>엔드포인트</u> 주제가 업데이트되었습니다.	2016년 10월 20일
새로운 리전	이제 미국 동부(오하이오) 리전에서 CodePipeline을 사용할 수 있습니다. <u>in AWS CodePipeline 할당량</u> 주제와 <u>리전 및</u> <u>엔드포인트</u> 주제가 업데이트되었습니다.	2016년 10월 17일
업데이트된 주제	<u>파이프라인 스테이지 및 작업 생성</u> 이 [Source provider] 및 [Build provider] 목록에 사용자 지정 작업의 버전 식별자를 표시하기 위한 지원을 반영하도록 업데이트되었습니다.	2016년 9월 22일
업데이트된 주제	<u>스테이지에 수동 승인 작업 추가</u> 섹션이 승인 작업 검토자가 이메일 알림에서 직접 [Approve or reject the revision] 양식 을 열 수 있는 개선 사항을 반영하도록 업데이트되었습니다.	2016년 9월 14일
신규 및 업데이트된 주제	새 주제는 소프트웨어 릴리스 파이프라인을 통해 현재 통과 하고 있는 코드 변경 사항에 대한 세부 정보를 보는 방법을 설명합니다. 이 정보에 대한 빠른 액세스는 수동 승인 작업 을 검토하거나 파이프라인의 오류를 해결할 때 유용할 수 있 습니다.	2016년 9월 08일
	새 섹션인 <u>Monitoring pipelines</u> 은 파이프라인의 진행 상황 및 상태 모니터링과 관련된 모든 주제에 대한 중앙 위치를 제공합니다.	
신규 및 업데이트된 주제	새 단원인 <u>스테이지에 수동 승인 작업 추가</u> 는 파이프라인의 수동 승인 작업 구성 및 사용에 대한 정보를 제공합니다. 이 단원의 주제는 승인 프로세스, 필요한 IAM 권한을 설정하고 승인 작업을 생성하며 승인 작업을 승인 또는 거부하기 위한 지침, 승인 작업이 파이프라인에 도달한 경우 생성된 JSON 데이터의 샘플에 대한 개념적 정보를 제공합니다.	2016년 7월 06일

변경 사항	설명	변경 날짜
새로운 리전	이제 유럽(아일랜드) 리전에서 CodePipeline을 사용할 수 있 습니다. <u>in AWS CodePipeline 할당량</u> 주제와 <u>리전 및 엔드</u> <u>포인트</u> 주제가 업데이트되었습니다.	2016년 6월 23일
새 주제	새 주제인 가 단계에서 실패한 작업 또는 병렬 실패한 작업 의 그룹을 다시 시도하는 방법을 설명하도록 추가되었습니 다.	2016년 6월 22일
업데이트된 주제	파이프라인 스테이지 및 작업 생성, 인증 및 액세스 제어, CodePipeline 파이프라인 구조 참조, CodePipeline과 제품 및 서비스 통합 등의 여러 주제가 AWS OpsWorks에서 생 성된 사용자 지정 Chef 쿡북 및 애플리케이션과 함께 코드 를 배포하도록 파이프라인을 구성하기 위한 지원을 반영하 도록 업데이트되었습니다. 에 대한 CodePipeline 지원 AWS OpsWorks 은 현재 미국 동부(버지니아 북부) 리전(us-eas t-1)에서만 사용할 수 있습니다.	2016년 6월 2일
신규 및 업데이트된 주제	새 주제인 <u>자습서: 간단한 파이프라인 생성(CodeCommit 리</u> <u>포지토리)</u> 이 추가되었습니다. 이 주제에서는 CodeCommit 리포지토리 및 브랜치를 파이프라인에서 소스 작업에 대한 소스 위치로 사용하는 방법을 보여 주는 샘플 안내서를 제공 합니다. 다른 여러 주제가 인증 및 액세스 제어, <u>CodePipel</u> ine과 제품 및 서비스 통합, <u>자습서: 4단계 파이프라인 생성</u> , <u>CodePipeline 문제 해결</u> 등 CodeCommit과의 통합을 반영 하도록 업데이트되었습니다.	2016년 4월 18일
새 주제	새 주제인 <u>CodePipeline의 파이프라인에서 AWS Lambda</u> <u>함수 호출</u> 이 추가되었습니다. 이 주제에는 파이프라인에 Lambda AWS Lambda 함수를 추가하기 위한 샘플 함수와 단계가 포함되어 있습니다.	2016년 1월 27일
업데이트된 주제	인증 및 액세스 제어, 리소스 기반 정책에 새로운 단원이 추 가되었습니다.	2016년 1월 22일

변경 사항	설명	변경 날짜
새 주제	새 주제인 <u>CodePipeline과 제품 및 서비스 통합</u> 이 추가되었 습니다. 파트너 및 기타 와의 통합에 대한 정보가이 주제로 이동 AWS 서비스 되었습니다. 블로그와 비디오에 대한 링 크도 추가되었습니다.	2015년 12월 17일
업데이트된 주제	Solano CI와의 통합 세부 정보가 <u>CodePipeline과 제품 및 서</u> <u>비스 통합</u> 에 추가되었습니다.	2015년 17월 11일
업데이트된 주제	이제 Jenkins용 CodePipeline 플러그인을 Jenkins용 플러그 인 라이브러리의 일부로 Jenkins Plugin Manager를 통해 사 용할 수 있습니다. 플러그인 설치 단계가 <u>자습서: 4단계 파이</u> <u>프라인 생성</u> 에서 업데이트되었습니다.	2015년 9월 11일
새로운 리전	CodePipeline을 이제 미국 서부(오레곤) 리전에서 사용할 수 있습니다. <u>in AWS CodePipeline 할당량</u> 주제가 업데이트되 었습니다. 링크가 <u>리전 및 엔드포인트</u> 에 추가되었습니다.	2015년10월 22일
새 주제	두 가지 새 주제인 <u>CodePipeline의 Amazon S3에 저장된 아</u> 티팩트에 대해 서버 측 암호화 구성 및 <u>CodePipeline에서 다</u> 른 AWS 계정의 리소스를 사용하는 파이프라인 생성이 추가 되었습니다. 인증 및 액세스 제어, <u>예 8: 파이프라인에서 다</u> 른 계정과 연결된 AWS 리소스 사용에 새로운 단원이 추가 되었습니다.	2015년 8월 25일
업데이트된 주제	CodePipeline에서 사용자 지정 작업 생성 및 추가 주제가 inputArtifactDetails 및 outputArtifactDeta ils 등 구조의 변경 사항을 반영하도록 업데이트되었습니 다.	2015년 8월 17일
업데이트된 주제	<u>CodePipeline 문제 해결</u> 주제가 서비스 역할 및 Elastic Beanstalk의 문제를 해결하기 위해 수정된 단계로 업데이트 되었습니다.	2015년 8월 11일
업데이트된 주제	인증 및 액세스 제어 주제가 <u>CodePipeline에 대한 서비스 역</u> <u>할</u> 의 최신 변경 사항을 포함하도록 업데이트되었습니다.	2015년 8월 6일

변경 사항	설명	변경 날짜
새 주제	<u>CodePipeline 문제 해결</u> 주제가 추가되었습니다. <u>자습서: 4</u> <u>단계 파이프라인 생성</u> 에서 IAM 역할 및 Jenkins에 대해 업 데이트된 단계가 추가되었습니다.	2015년 7월 24일
주제 업데이트	<u>자습서: 간단한 파이프라인 생성(S3 버킷)</u> 및 <u>자습서: 4단계</u> <u>파이프라인 생성</u> 에서 샘플 파일 다운로드에 대해 업데이트 된 단계가 추가되었습니다.	2015년 7월 22일
주제 업데이트	<u>자습서: 간단한 파이프라인 생성(S3 버킷)</u> 에서 샘플 파일의 다운로드 문제에 대한 임시 해결 방법이 추가되었습니다.	2015년 7월 17일
주제 업데이트	제한 사항을 변경할 수 있는 정보를 가리키기 위해 <u>in AWS</u> <u>CodePipeline 할당량</u> 에서 링크가 추가되었습니다.	2015년 7월 15일
주제 업데이트	인증 및 액세스 제어의 관리형 정책 단원이 업데이트되었습 니다.	2015년 7월 10일
최초 공개 릴리스	이는 CodePipeline 사용 설명서의 최초 공개 릴리스입니다.	2015년 7월 9일

CodePipeline 기능 참조

이 섹션은 CodePipeline 설명서에서 기능에 대한 상위 수준 참조입니다. 파이프라인 구조의 개념적 개 요는 <u>CodePipeline 파이프라인 구조 참조</u> 단원을 참조하십시오.

이 표는 기능 참조 정보로 주기적으로 업데이트됩니다.

테이블 데이 터 날짜	Feature	CodePipeline 사용 설명서	CodePipeline API 가이드	AWS CloudForm ation 참조	AWS CDK 참 조 - L1 구문
2024년 10월 15일	스테이지 실 패 시 자동 재 시도	실패 시 자동 재시도를 위 한 스테이지 구성	RetryConf iguration		
2024년 10월 15일	beforeEntry 조건에 대한 건너뛰기 결 과	<u>건너뛰기</u> <u>결과 및</u> VariableC <u>heck 규칙</u> 을 사용하여 입력 조건 생 성(콘솔)	FailureCo nditions		
2024년 10월 3일	새 컴퓨팅 범 주를 사용한 명령 작업	<u>명령 작업 참</u> 조	ActionDec laration		
2024년 8월 28일	스테이지 조 건에 대한 beforeEntry 유형	<u>스테이지에</u> <u>대한 조건 구</u> 성	BeforeEnt ryConditions	AWS::Code Pipeline: :Pipeline BeforeEnt ryConditions	<u>beforeEntry</u>
2024년 8월 28일	스테이지 조 건에 대한 onSuccess 유 형	<u>스테이지에</u> <u>대한 조건 구</u> <u>성</u>	SuccessCo nditions	AWS::Code Pipeline: :Pipeline	onSuccess

테이블 데이 터 날짜	Feature	CodePipeline 사용 설명서	CodePipeline API 가이드	AWS CloudForm ation 참조	AWS CDK 참 조 - L1 구문
				SuccessCo nditions	
2024년 4월 26일	스테이지 조 건에 대한 스 테이지 롤백 및 onFailure 유형	<u>스테이지 롤</u> <u>백 구성</u>	<u>RollbackS</u> tage	<u>onFailure</u>	<u>onFailure</u>
2024년 2월 8 일	PARALLEL 및 QUEUED 실행 모드	<u>파이프라인</u> 실행 모드 설 정 또는 변경	PipelineE xecution	Execution Mode	<u>실행 모드</u>
2023년 11월 17일	소스 재정의	<u>소스 개정 재</u> 정의로 파이 프라인 시작	SourceRev isionOverride	N/A	N/A
2023년 10월 24일	파이프라인 유형	나에게 적합 <u>한 파이프라</u> 인 유형은 무 엇인가요?	PipelineD eclaration	<u>PipelineType</u>	<u>enum</u> PipelineType
2023년 10월 24일	파이프라인 수준 변수	<u>자습서: 파이</u> <u>프라인 수준</u> <u>변수 사용</u>	PipelineV ariable	AWS::Code Pipeline: :Pipeline VariableD eclaration	<u>파이프라인</u> <u>수준 변수</u>

테이블 데이 터 날짜	Feature	CodePipeline 사용 설명서	CodePipeline API 가이드	AWS CloudForm ation 참조	AWS CDK 참 조 - L1 구문
2023년 10월 24일	파일 경로/브 랜치/풀 요청 트리거 및 필 터링	<u>트리거 및 필</u> 터링을 사용 하여 시작 파 이프라인 자 동화 자습서: 파이프라인을 시작하기 위 한 풀 요청의 브랜치 이름 필터링(V2 유 형)	ActionDec laration	AWS::Code Pipeline: :Pipeline PipelineT riggerDec laration	<u>트리거</u>
2023년 10월 17일	스테이지 재 시도	실패한 스테 이지 또는 실 <u>패한 작업에</u> 대해 스테이 지 재시도 구 성	<u>ActionDec</u> <u>laration</u>	N/A	N/A

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